Restrições Complexas sobre Álgebras de Árvores e Aplicação a Gramáticas Lógicas

Tese submetida à Faculdade de Ciências da Universidade do Porto para obtenção do grau de Doutor em Ciência de Computadores

por

Nelma Resende Araújo Moreira

Departamento de Ciência de Computadores

Faculdade de Ciências da Universidade do Porto

1997

Índice

Sumário

Nos últimos anos, a resolução de restrições simbólicas tem conhecido um interesse crescente em várias áreas das Ciência de Computadores e, em particular, na área da Linguística Computacional. Na verdade, as restrições sobre álgebras de árvores racionais, podem contribuir significativamente para a redução do espaço de procura em problemas de Processamento de Linguagem Natural, ao mesmo tempo que aumentam a expressividade dos formalismos. Nesta dissertação desenvolvemos um modelo formal e computacional para linguagens de restrições sobre árvores finitas, racionais e de estruturas de atributos. Embora as teorias de 1ª ordem destas álgebras sejam decidíveis, elas são computacionalmente intratáveis. O nosso objectivo foi desenhar um modelo teórico e algoritmos de decisão para lidar com este problema, e que foram usados na implementação de vários formalismos gramaticais, designados por *gramáticas lógicas com restrições*.

As contribuições principais desta dissertação são:

- Resolutores completos de restrições complexas sobre álgebras de árvores finitas e racionais. Para reduzir o custo computacional do teste da satisfazibilidade de restrições complexas, a estratégia utilizada baseia-se na factorização de informação determinística contida numa restrição complexa que é utilizada para simplificar a restrição resultante, retardando a explosão das disjunções. Desenvolvemos também sistemas de reescrita para a detecção de factores comuns a elementos de disjunções e sistemas de reescrita que utilizam informação negativas para a simplificação. Investigamos o problema da eliminação de restrições negativas e apresentamos um algoritmo de decisão para assinaturas finitas.

- Estabelecimento duma relação entre a álgebra de árvores racionais e a álgebra de estruturas de atributos, de tal modo que o problema da satisfazibilidade numa se reduz ao da outra.

- Um método de obtenção de modelos minimais para certas classes de restrições satisfazíveis. Estas caracterizações são importantes para a simplificação e detecção de restrições equivalentes.

- Uma implementação de baixo nível dos algoritmos principais dos resolutores completos.

- Um *Calculus* de ordem superior para descrições de árvores baseado num λ-*calculus* tipificado, que estende as técnicas desenvolvidas para a resolução de restrições complexas. As descrições de ordem superior foram utilizadas para as representações semânticas em gramáticas de categorias para Linguagem Natural.

Abstract

In recent years, symbolic constraints have raised a considerable interest in several areas of computation and particularly in the Computational Linguistics community. As a matter of fact constraints over the algebra of rational trees can significantly contribute to the reduction of the search space of problems in Natural Language Processing, while increasing the formalisms expressive power. In this thesis we developed a formal and computational framework for constraint languages over the algebras of finite, rational and feature trees. Although, the first-order theories of these algebras are decidable, they are computationally intractable. Our goal was to design a formal model and decision procedures to face that problem, and that have been (successfully) used in the implementation of several grammar formalisms (called Constraint Logic Grammars).

The main contributions of this thesis are:

- Complete solvers for complex constraints in the algebras of finite and rational trees. In order to face the NP-hardness of the satisfiability problem for complex constraints, our approach was based in factoring out, in polynomial time, deterministic information contained in a complex constraint, simplifying the remaining formula using that information and delaying as much as possible the explosion of disjunctions. We also developed rewrite systems for detection of common factors in disjuncts and rewrite systems that use negative information for the simplification process. We investigated the problem of eliminating negative constraints and presented a decision procedure for finite signatures.

- Establishment of a relation between the algebra of rational trees and the algebra of feature trees, such that the satisfiability problem in one can be reduced to the satisfiability problem in other.

- A method for the computation of the minimal models of certain classes of satisfiable constraints. These characterizations are useful to simplify and to detect equivalent constraints.

- A low-level implementation of the main features of the complete solvers.

- A higher order tree descriptions *Calculus* based on a typed λ-*calculus*, which extended the previous techniques developed for resolving complex constraints. The higher order tree descriptions were used as semantic representations in categorial grammars for Natural Language.

Sommaire

Récemment, les contraintes symboliques ont connaît un intérêt grandissant en plusieurs domaines de computation et, particulièrement, dans la communauté de Linguistique Informatique. En effet, les contraintes dans des algèbres de arbres rationnels peut contribuer significativement pour la réduction de l'espace de recherche en problèmes de Traitement de la Langage Naturelle, tandis que elles augmente l'expressivité des formalismes. Dans cette thèse nous avons développé un modèle théorique et logiciel des langages de contraintes dans des algèbres de arbres finis, rationnels et d'attributs. Bien que les théories de premier ordre de ces algèbres sont décidables, elles ont une complexité intraitable. Notre objectif était de définir un modèle théorique et des algorithmes de décision, en vue de maîtriser cet problème, et qui ont été utilisés dans l'implantation de plusieurs formalismes grammaticales nommés *Constraint Logic Grammars*.

Les contributions originales de cette thèse sont:

- Solveurs complètes de contraintes complexes dans des algèbres de arbres finis et rationnels. Pour réduire le coût du test de satisfaisabilité pour les contraintes complexes, notre approche été de extraire la information déterministe contenu dans une contrainte complexe, simplifiant la formule obtenu avec cette information et retardant le plus possible l'explosion des disjonctions. Nous avons développé aussi des systèmes de réécriture pour la détection d'information déterministe commun à de éléments des disjonctions et des systèmes de réécriture qui utilisent information négative dans le processus de simplification. Nous avons abordé le problème de la élimination de négations et présenté un algorithme de décision dans le cas des signatures finis.

- L'établissement d'une relation entre l'algèbres de arbres rationnels et l'algèbre de arbres d'attributs, tel que le problème de la satisfaisabilité d'une, peut être réduit au d'autre.

- Un schéma pour obtenir tous les modèles minimaux de certains classes de contraintes satisfaisantes. Ces caractérisations sont utiles pour simplifier et détecter la équivalence de contraintes.

- Une implantation de baisse niveau des algorithmes principaux des solveurs complètes.

- Un *Calculus* d'ordre supérieur basé dans un λ-*calculus* typé de descriptions d'attributs, qui généralise les techniques développé antérieurement. Les descriptions d'attributs ont été utilisé comme représentations sémantiques en grammaires catégorielles pour langages naturelles.

Agradeço especialmente,

Ao meu orientador Luís Damas, a oportunidade que me deu de participar no projecto $\mathcal{CLG}$ e pela orientação, pela disponibilidade e pelo seu encorajamento constante.

Ao Rogério a paciência e o apoio.

Aos meus colegas do N.C.C.U.P, em particular, Ana Paula Tomás, Sabine Broda, Zé Paulo Leal, Rogério Reis, Vítor Costa, Fernando Silva, Armando Matos e Miguel Filgueiras, pela camaradagem e pelo bom ambiente de trabalho.

Ao Miguel Filgueiras, por ter teimosamente acumulado, neste último ano, parte do meu serviço docente no Departamento de Ciências de Computadores da F.C.U.P.

Ao Nino Varile, a colaboração no Projecto $\mathcal{CLG}$.

Ao Armando Matos, à Sabine Broda e ao Rogério Reis, pela ingrata tarefa de ler e comentar partes desta dissertação.

Ao Departamento de Matemática Aplicada da F.C.U.P. e ao Centro de Informática da Universidade do Porto, as condições e os meios disponibilizados para a realização deste trabalho.

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Nesta dissertação estudamos linguagens de restrições sobre estruturas complexas de dados, nomeadamente, termos de lógica de 1ª ordem (árvores finitas), árvores racionais e estruturas de atributos. Nos formalismos gramaticais *baseados em unificação* para Linguagem Natural, a informação linguística é representada através de restrições sobre estruturas de atributos, que são basicamente conjuntos de pares *atributo-valor*. Apesar da diversidade de formalizações da noção de estrutura de atributos e das suas descrições, iremos estabelecer uma relação entre estruturas de atributos e árvores de racionais, que permite obter uma teoria formal e computacional única para as linguagens de descrições de ambas as estruturas de dados.

Vamos considerar linguagens de restrições que correspondem a teorias da lógica de 1ª ordem tendo a igualdade como único símbolo predicado, axiomatizadas de forma a admitirem as álgebras das árvores finitas ($\mathcal{T}$) ou racionais ($\mathcal{RT}$), parametrizadas pelo conjunto de símbolos de função, como modelos canónicos. As correspondentes axiomatizações são completas [Mah88], donde as teorias decidíveis. Isto é, a unificação de termos de 1ª ordem, que pode ser vista como a resolução de conjunções de igualdades entre termos, é estendida dum modo decidível a fórmulas de lógica de 1ª ordem tendo a igualdade como único símbolo predicado. Assim, um dos nossos objectivos é a obtenção e a especialização de algoritmos de decisão da satisfazibilidade e da equivalência de fórmulas nestas linguagens. Este objectivo é especialmente relevante dado que a complexidade do problema da decisão da satisfazibilidade é (pelo menos) **NP**-completo.

As fórmulas destas linguagens são designadas *restrições complexas* e os respectivos algoritmos para a satisfazibilidade, *resolutores*.

Na origem deste trabalho esteve a colaboração no projecto $\mathcal{CLG}$ [DV89a, BDMV90, DMV91, Dam91, DV92, DMV94, DM95a], para o desenvolvimento de gramáticas lógicas com restrições para Linguagem Natural, que será descrito na Secção 1.4. Em [DV92, DMV94, DMB93] são descritas técnicas de factorização que permitem reduzir, em tempo polinomial, o tamanho das restrições complexas extraindo informação determinística, antes da aplicação dum algoritmo completo para a satisfazibilidade. Nesta dissertação pretende-se apresentar, formalizar e generalizar os modelos formais e computacionais para a resolução de restrições complexas sobre álgebras de árvores, dos formalismos $\mathcal{CLG}$.

1.1 Contribuições da Dissertação

As contribuições principais desta dissertação são:

- Os resolutores completos de restrições complexas sobre álgebras de árvores.

  Para reduzir o custo computacional, a estratégia utilizada nos resolutores baseia-se na factorização dum modelo parcial que é utilizado para a simplificação da restrição restante. Para uma manipulação computacional conveniente de fórmulas das teorias das álgebras de árvores, foram definidas linguagens sem quantificadores que envolvem expressões com caminhos em árvores, e que permitem a referência a sub-árvores. Para o estudo do problema da satisfazibilidade em $\mathcal{T}$ ou $\mathcal{RT}$, apenas é necessário considerar um subconjunto destas linguagens, a linguagem $\mathcal{LU}$. Foram definidas formas normais que caracterizam as restrições satisfazíveis de $\mathcal{LU}$ e apresentados algoritmos para as obter. Os modelos duma restrição podem ser expressos por disjunções de formas normais para conjunções de restrições atómicas positivas (*formas resolvidas*) e negativas (*formas primárias*). Em particular, um modelo parcial é uma forma resolvida ou uma forma primária.

  Este método de processamento de restrições complexas foi apresentado em [DMV91, DV92, DMV94], e a generalização a um resolutor completo foi feita em [DMV94].

  Nesta dissertação, foram formalizados e demonstrados alguns dos resultados enunciados nos trabalhos citados. As noções de *formas resolvidas normais* e *formas primárias*, assim como os algoritmos de obtenção destas formas, o estudo da eliminação de negações para assinaturas finitas e a caracterização de restrições em termos daquelas formas, são originais. A apresentação dos resolutores completos difere das de [DMV94, DMB93], pelo facto de se usarem *formas resolvidas normais*, *formas primárias* e sistemas de reescrita para a satisfazibilidade incremental. A correcção, a terminação, a completude e a complexidade dos sistemas de reescrita, embora indicadas anteriormente, não tinham sido apresentadas formalmente. Os resolutores completos foram generalizados, considerando formas primárias como modelos parciais, e foram discutidos os problemas levantados pela introdução de informação negativa no modelo. Investigamos o problema da eliminação de restrições negativas e apresentamos um algoritmo de decisão para assinaturas finitas.

Os resolutores completos apenas são apresentados para as álgebras de árvores racionais, embora para a sua adaptação a árvores finitas baste considerar as respectivas formas normais.

- A relação entre a teoria $\mathcal{RT}$ e as teorias das árvores de atributos.

  Apesar das variadas lógicas existentes na literatura, para a formalização de descrições de estruturas de atributos, considerando lógicas de árvores de atributos desenvolvidas por G. Smolka e seus colegas [Smo92, AKPS94, ST94], mostramos que o problema da satisfazibilidade nessas lógicas se reduz a um problema de satisfazibilidade na teoria $\mathcal{RT}$. Assim, a maioria das descrições de estruturas de atributos usadas em processamento de Linguagem Natural e linguagens de programação em lógica por restrições, podem ser formalizadas por um domínio que é bem conhecido quer do ponto de vista formal, quer computacional. Em particular, a similitude entre a linguagem $\mathcal{LU}$ e as linguagens de lógicas de árvores de atributos, permite que os resolutores desenvolvidos para $\mathcal{LU}$ sejam adaptáveis directamente para essas linguagens.

  Estes resultados foram apresentados em [DMV94].

- A caracterização de representações minimais para restrições satisfazíveis. Mostramos como obter modelos minimais de alguns tipos de restrições de $\mathcal{LU}$. Estas representações são úteis para a simplificação e a detecção de restrições equivalentes.

- A implementação dos resolutores completos para restrições complexas.

  Uma descrição parcial das implementações foi apresentada em [DMB93].

- Uma linguagem de ordem superior parametrizada por uma linguagem de restrições sobre árvores. Definimos o *Calculus* de descrições Λ_{LR} e mostramos que o seu problema de satisfazibilidade é decidível. Os resolutores para restrições complexas obtidos anteriormente, foram generalizados para $\Lambda_{\mathcal{LU}}$.

  Definimos e implementamos, também, o formalismo linguístico $\mathcal{CCLG}$, das gramáticas de categorias com restrições, que utiliza o *Calculus* Λ_{LR}, para o tratamento das representações semânticas.

  O *Calculus* Λ_{LR} e o formalismo $\mathcal{CCLG}$ foram parcialmente apresentados em [DM95a].

1.2 Estrutura da Dissertação

Capítulo 1: Introdução. As restantes secções deste capítulo enquadram e motivam o trabalho desta dissertação. Na Secção 1.3 relacionamos as linguagens de restrições com o modo como é descrita a informação em formalismos linguísticos actuais. Na Secção 1.3, como já referido, são apresentadas as gramáticas lógicas com restrições ($\mathcal{CLG}$), a sua metodologia e os seus modelos formal e de processamento.

Capítulo 2: Linguagens de Restrições. Neste capítulo, após alguns preliminares de natureza matemática, são apresentadas as linguagens da lógica de 1ª ordem tendo a igualdade como único símbolo de predicado ($\mathcal{L}$), interpretadas em álgebras de árvores finitas ($\mathcal{T}$) e racionais ($\mathcal{RT}$). São definidas as linguagens de restrições apropriadas para o estudo formal e computacional do problema da satisfazibilidade, nomeadamente a linguagens $\mathcal{LC}_0$, $\mathcal{LC}$ e $\mathcal{LU}$. Para as álgebras de árvores finitas e racionais, mostramos que as linguagens $\mathcal{LC}_0$, $\mathcal{LC}$ e $\mathcal{L}$ são equivalentes. Mostramos ainda que a linguagem $\mathcal{LU}$, que é um subconjunto da linguagem $\mathcal{LC}$, é suficiente para o estudo da satisfazibilidade.

Capítulo 3: $\mathcal{RT}$ e Linguagens de Árvores de Atributos. Neste capítulo relacionamos a linguagem $\mathcal{L}$ interpretada em $\mathcal{RT}$ e linguagens de estruturas de atributos. É apresentada a equivalência entre as respectivas teorias e evidenciada a similitude entre essas linguagens de estruturas de atributos e a linguagem $\mathcal{LU}$. Nos restantes capítulos iremos considerar essencialmente a linguagem $\mathcal{LU}$ interpretada em $\mathcal{RT}$.

Capítulo 4: Satisfazibilidade em $\mathcal{LU}$. Caracterizamos as restrições satisfazíveis em $\mathcal{LU}$ em termos de formas normais e apresentamos algoritmos para a obtenção de tais formas. Abordamos, ainda, o problema da eliminação de restrições negativas.

Capítulo 5: Modelos Minimais em $\mathcal{LU}$. Um pouco independentemente dos restantes capítulos, investigamos os problemas da existência e obtenção de representações minimais de restrições satisfazíveis. As noções de modelo, modelo parcial e modelo minimal de uma restrição satisfazível serão também utilizadas nos capítulos seguintes.

Capítulo 6: Resolutores para $\mathcal{LU}$. Sendo a determinação da satisfazibilidade de restrições de $\mathcal{LU}$ um problema **NP**-completo, apresentamos neste capítulo alguns métodos que permitam a redução do seu custo computacional. Apresentamos resolutores completos para

$\mathcal{LU}$, que usam um processo incremental de satisfazibilidade, e métodos de factorização de informação comum a restrições.

Capítulo 7: Implementação dos Resolutores. Neste capítulo são feitas algumas considerações sobre a implementação dos resolutores de restrições apresentados no capítulo anterior. Em particular, é salientado o algoritmo de satisfazibilidade incremental que permite uma implementação eficiente dos resolutores de restrições complexas.

Capítulo 8: O *Calculus* Λ_{LR}. Neste capítulo, as linguagens de restrições sobre árvores são estendidas a linguagens de ordem superior, pela combinação com um λ-*calculus* tipificado (simples). Depois de definirmos a noção de satisfazibilidade, mostramos que o problema da satisfazibilidade para esta linguagem é decidível. Os resolutores definidos no Capítulo 6 para $\mathcal{LU}$, são generalizados para $\Lambda_{\mathcal{LU}}$. A motivação para a introdução desta linguagem foi a inadequação dos formalismos linguísticos baseados em unificação no tratamento das representações semânticas. A praticabilidade e expressividade do *Calculus* Λ_{LR} é ilustrada na definição e implementação das gramáticas de categorias com restrições ($\mathcal{CCLG}$).

Capítulo 9: Conclusão. É sumariado o trabalho realizado e discutidas algumas direcções de trabalho futuro.

1.3 Processamento de Linguagem Natural e Restrições

Os formalismos gramaticais para Linguagem Natural (LN) têm como objectivo principal descrever o conjunto de frases – sequências de palavras – da linguagem, as propriedades estruturais dessas frases – sintaxe – e o seu significado – semântica. Nas últimas duas décadas vários destes formalismos, distanciando-se do tipo de gramáticas usadas para caracterizar linguagens formais, utilizam estruturas complexas de dados para descrever a informação linguística. Mais que aumentar o poder generativo das gramáticas, *i.e*, o tipo de linguagens que geram, o objectivo é aumentar a sua expressividade e permitir uma modelação mais adequada das LNs.

Nas secções seguintes fazemos uma revisão de algumas dessas abordagens, focando-nos no tipo de informação que manipulam. A exposição é um pouco longa, mas tem como objectivo enquadrar e motivar o trabalho realizado nesta dissertação.

1.3.1 Gramáticas Lógicas

O paradigma da Programação em Lógica baseia-se no uso de *lógica* para representar o conhecimento e num mecanismo de *inferência* para controlar e manipular o conhecimento. Esta associação foi expressa por R. Kowalski [Kow79]:

$$programa = lógica + control$$

Teve como origem as investigações em demonstração automática de teoremas, realizadas durante os anos 60 e 70 e baseadas essencialmente nos resultados de Jacques Herbrand [Her30] em teoria da lógica em forma clausal. Especialmente relevante foi a introdução da regra da resolução de J. A. Robinson [Rob65] como regra de inferência. Central a este método foi a noção de *unificação* de termos de lógica de 1ª ordem. Diz-se que dois termos *unificam* se existe uma substituição de variáveis por termos que os torna idênticos. Por exemplo, $f(x, a)$ e $f(g(a, z), w)$, são *unificáveis* substituindo x por $g(a, z)$ e w por a.

Para um subconjunto da lógica de 1ª ordem - as cláusulas de Horn - foi demonstrada a existência dum sistema de inferência completo, no sentido em que as cláusulas, que se deduzem a partir dum dado conjunto P, são exactamente as que são válidas em todos os modelos de P, e esse sistema pode ser eficientemente implementado em computador. Este resultado está na base de linguagens como o Prolog.

Intimamente associado com o desenvolvimento da linguagem Prolog, esteve o desenvolvimento de sistemas para o processamento de Linguagem Natural (PLN). Baseado no seu sistema *Q-system*, A. Colmerauer desenvolveu uma linguagem que pudesse simultaneamente ser usada para análise de LN e para implementar mecanismos dedutivos de pergunta/resposta. O sistema dedutivo resultante, concebido como um instrumento para aplicações de PLN [CKPR73], constituiu a primeira implementação dum sistema Prolog, o "Marseille Prolog". Para além da aplicação inicial, Colmerauer desenvolveu as *gramáticas de metamorfose* [Col78] que estão na base de vários outros formalismos de gramáticas lógicas, nomeadamente das *gramáticas de cláusulas definidas* (DCG) introduzidas por Pereira e Warren [PW80] e que se encontram implementadas na maioria dos sistemas Prolog. Nestas abordagens a informação linguística é representada por termos da lógica de 1ª ordem e as variáveis lógicas e a unificação de termos são mecanismos de transmissão de informação. Estes sistemas podem ser vistos como formalismos linguísticos baseados em restrições, que a seguir apresentaremos, onde a linguagem de restrições é constituída por conjunções de equações de termos interpretados no Universo de Herbrand[1]. Uma gramática é constituída por um conjunto de axiomas, que expri-

[1]Tal como o Prolog, visto como uma linguagem de programação lógica por restrições [JM94]. Ver também Capítulo 2.

mem o conhecimento sobre a linguagem, e mostrar que uma frase pertence a uma linguagem (descrita por essa gramática) corresponde a usar um demonstrador de teoremas, que produza uma demonstração de que a frase é um teorema. Este paradigma, usado inicialmente para analisadores sintácticos, é conhecido por *parsing as deduction* [PW83] e foi generalizado para *language-processing as deduction* [Shi88, Dah94], uma vez que outros aspectos do PLN podem ser tratados como dedução, nomeadamente a geração e a interpretação (análise semântica) de frases.

Pereira e Shieber em [PS87a] apresentam uma excelente introdução dos métodos de programação em lógica para PLN e V. Dahl em [Dah94] faz uma revisão da conexão destas duas áreas, onde são incluídos para além das gramáticas lógicas, diversos formalismos linguísticos baseados em restrições.

1.3.2 Gramáticas de Unificação ou Baseadas em Restrições

Desde os fins dos anos 70 e paralelamente ao desenvolvimento da Programação em Lógica e das gramáticas lógicas, os linguístas teóricos e computacionais desenvolveram formalismos onde a noção central de unificação de estruturas é muito semelhante à usada em Programação em Lógica. Os formalismos gramaticais *baseados em unificação*, também genericamente designados por *baseados em restrições*, têm a sua origem, independentemente, nas *gramáticas de unificação* de Martin Kay [Kay85b], nas *gramáticas funcionais léxicas* (LFG) de Bresnan e Kaplan [KB82] e nas *gramáticas generalizadas de estrutura de frase* (GPSG) de G. Gadzar [GKPS85]. Como características comuns a estes formalismos salientamos:

- o modo *declarativo* e, mais ou menos uniforme, usado na representação dos vários tipos de informação *lexical*, *sintáctica* e *semântica.* Em oposição ao modo procedimental usado, por exemplo, nos *diagramas de transição aumentados* (ATN) de Woods [Woo70], a separação entre a descrição da informação linguística e o modo como ela é processada, permite uma maior flexibilidade na escolha de estratégias de implementação e uma maior reutilização das gramáticas. Em princípio, as gramáticas são reversíveis podendo ser mais facilmente usadas para análise e geração de LN.

- a cada constituinte da frase é associada *informação parcial* sobre os objectos linguísticos, e que é combinada de modo hierárquico, indutivo e monótono. As estruturas que representam os objectos linguísticos designam-se por *estruturas de atributos* e têm semelhanças com as estruturas de dados usadas em Ciência de Computadores e os enquadramentos usados em Inteligência Artificial.

- a relação entre os constituintes duma frase e a informação linguística associada é descrita por uma *linguagem de restrições* sobre os objectos linguísticos.

Alguns destes formalismos foram desenhados como ferramentas computacionais para implementar sistemas de PLN, outros são parte de teorias linguísticas. Como teorias linguísticas, para além das já referidas *gramáticas funcionais léxicas* (LFG) [KB82] e das *gramáticas generalizadas de estrutura de frase* (GPSG) [GKPS85] têm especial relevância actualmente *as gramáticas de estrutura de frase guiadas por cabeça* (HPSG) [PS87b, PS94], que são baseadas nas GPSGs. Como ferramentas computacionais, citamos as *gramáticas funcionais de unificação* (FUG) [Kay85a], o formalismo PATR-II [SUP$^+$83, Shi86], as gramáticas de unificação que são extensões de gramáticas de categorias, UCG [ZKC87] e CUG [Usz86], e as *gramáticas de árvores adjacentes baseadas em estruturas de atributos* FTAG [VS87, VSJ88]. Shieber em [Shi86] apresenta, do ponto de vista das estruturas de informação e como são manipuladas, uma primeira introdução uniforme a estes formalismos, e Sells em [Sel85] faz uma introdução a algumas das teorias linguísticas referidas.

1.3.2.1 Estruturas de Atributos

As estruturas de atributos, para descrever informação linguística, aparecem em teorias linguísticas desde Chomsky e Halle [CH57] e na teoria transformacional [Cho65], antes de serem usadas extensivamente nos formalismos teóricos ou computacionais baseados em restrições, já citados.

Informalmente e geralmente, as *estruturas de atributos* são pares de *atributo–valor* onde os atributos têm um caracter funcional e cada valor pode ser uma constante (valor atómico) ou uma estrutura de atributos (valor complexo). Um valor pode ainda ser partilhado por vários atributos.

Um exemplo duma estrutura de atributos, na denominada *notação matricial*, é

$$\begin{bmatrix} cat & np \\ agr & \begin{bmatrix} pers & 3rd \end{bmatrix} \end{bmatrix} \tag{1.1}$$

Esta estrutura de atributos caracteriza uma frase nominal na terceira pessoa. O atributo *cat* (categoria) tem um valor atómico e o atributo *agr* (concordância sujeito-predicado) tem um valor complexo.

A operação básica para combinar estruturas de atributos é a *unificação* que a partir de um conjunto de estruturas compatíveis produz uma estrutura que contém toda a informação presente em cada elemento do conjunto e nada mais.

A unificação de (1.1) com a estrutura

(1.2) $$\begin{bmatrix} agr & \begin{bmatrix} num & sg \end{bmatrix} \end{bmatrix}$$

é a estrutura

(1.3) $$\begin{bmatrix} cat & np \\ agr & \begin{bmatrix} num & sg \\ pers & 3rd \end{bmatrix} \end{bmatrix}$$

A informação contida nas estruturas de atributos permite ordená-las de modo que uma estrutura de atributos é mais específica que outra se contém toda a informação da segunda e eventualmente mais informação. Em particular, o resultado da unificação de duas estruturas de atributos compatíveis é mais específica que cada uma delas. Esta relação de pré-ordem (transitiva e reflexiva) é denominada *subsunção*.

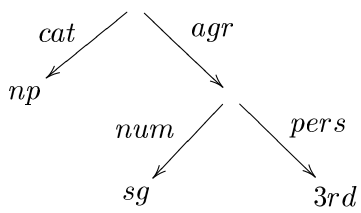
Duas estruturas são *incompatíveis* se os seus respectivos valores para um mesmo atributo são distintos, nomeadamente se são valores atómicos diferentes ou um é um valor atómico e outro é complexo. A estrutura de atributos (1.3) é incompatível com a estrutura

(1.4) $$\begin{bmatrix} agr & \begin{bmatrix} num & pl \\ pers & 3rd \end{bmatrix} \end{bmatrix}$$

Se as estruturas são incompatíveis a operação de unificação *falha*.

As estruturas de atributos podem também ser representadas por grafos dirigidos onde os nós são valores e os arcos são etiquetados por atributos. Para a estrutura de atributos (1.3) temos o grafo

(1.5)



Identificando um caminho no grafo pela sequência de atributos que etiquetam os arcos, uma estrutura de atributos pode ainda ser representada por uma conjunção de equações da forma $< sequência\_de\_atributos > \doteq valor$. Por exemplo, para a estrutura de atributos (1.3)

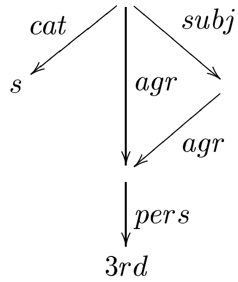
(1.6) $$< cat > \doteq np \wedge < agr\ num > \doteq sg \wedge < agr\ pers > \doteq 3rd$$

Se uma estrutura de atributos contém valores que são partilhados por diversos atributos isso pode ser representado por co-indexação, na notação matricial

(1.7)
$$\begin{bmatrix} cat & s \\ agr & \boxed{1}\begin{bmatrix} pers & 3rd \end{bmatrix} \\ subj & \begin{bmatrix} agr & \boxed{1} \end{bmatrix} \end{bmatrix}$$

ou, num grafo, por um nó onde chegam arcos etiquetados com atributos distintos – nó reentrante –

(1.8)



ou por equações da forma $<$ *sequência_de_atributos* $> \doteq <$ *sequência_de_atributos* $>$,

(1.9)
$$< agr > \doteq < subj\; agr >$$

A partilha de valores permite referir explicitamente que dois objectos coincidem, *i.e*, que são *extensivamente iguais*. Supondo que se pode sempre acrescentar informação (consistente) a uma estrutura de atributos, duas estruturas de atributos que tenham os mesmos atributos e os mesmos valores para esses atributos não são iguais. Nesta perspectiva, as estruturas de atributos são consideradas em geral *objectos intencionais*. Pelo contrário, os termos da lógica de 1ª ordem são sempre extensivos: dois termos são iguais se têm o mesmo functor e valores idênticos para cada um dos argumentos.

Descrevendo as estruturas de atributos como uma conjunção de equações a operação de unificação reduz-se a um problema de satisfazibilidade, para o qual são conhecidos algoritmos eficientes [NO80, Jaf84, Hue76, MM82].

No entanto, para a descrição dos objectos linguísticos é ainda desejável o uso de especificações negativas e disjuntivas de estruturas de atributos [Kar84, Kay85b].

Por exemplo, em Inglês, para a descrição dos verbos regulares no *present tense* apenas duas formas são necessárias: uma que represente a "terceira pessoa do singular" ("`runs`") e outra para as restantes, isto é, "não terceira pessoa do singular" ("`run`"). Para esta última teríamos uma representação matricial que incluiria a especificação

(1.10) $$\begin{bmatrix} agr & \neg \begin{bmatrix} num & sg \\ pers & 3rd \end{bmatrix} \end{bmatrix}$$

Karttunen interpreta procedimentalmente estas negações, dizendo que o valor de *agr* não pode unificar com uma estrutura de atributos que subsuma a estrutura de atributos precedida do operador $\neg$. Esta interpretação complica-se quando, por exemplo, a estrutura de atributos negada contém valores partilhados.

A conveniência do uso da disjunção de valores para um dado atributo é ilustrada, em língua alemã, pela descrição do artigo `die`

(1.11) $$\begin{bmatrix} agr & \left\{ \begin{array}{l} \begin{bmatrix} num & sg \\ gen & fem \end{bmatrix} \\ \begin{bmatrix} num & pl \end{bmatrix} \end{array} \right\} \\ case & \{nom\, acc\} \end{bmatrix}$$

onde a disjunção dos valores possíveis para cada atributo é indicada pelo uso de chavetas.

A disjunção foi também usada no formalismo FUG [Kay85b] para especificar grupos alternativos de atributos e estruturas de atributos.

A extensão da operação de unificação para a negação e a disjunção levantou problemas de carácter formal e computacional, que obrigaram a uma formalização precisa das noções envolvidas. A negação trouxe problemas relacionados com o significado dos valores negativos dum atributo e com a parcialidade da informação associada a uma estrutura de atributos. O tratamento de especificações com disjunções como em (1.11) pode ser feito pela expansão em forma normal disjuntiva, o que introduz uma explosão exponencial na complexidade dos algoritmos para a satisfazibilidade. Em particular, o problema da satisfazibilidade de fórmulas do cálculo proposicional é um problema **NP**-completo [GJ79] e a maior parte das lógicas de estruturas de atributos que iremos apresentar têm um problema de satisfazibilidade a que aquele se reduz.

1.3.2.2 Lógicas de Estruturas de Atributos

A formalização da noção de estrutura de atributos e das suas descrições fornece uma base matemática que permite a obtenção de resultados sobre a sua consistência. Do ponto de vista computacional é essencial uma descrição formal que permita desenhar e garantir a correcção dos algoritmos, assim como a obtenção de estratégias eficientes de processamento.

Várias foram as formalizações das estruturas de atributos e das suas descrições propostas para os diversos formalismos linguísticos existentes e, consequentemente, várias as tentativas de resolver os problemas computacionais.

Estas formalizações conduziram antes de mais à clarificação entre as *estruturas de atributos* e as suas *descrições*. Em formalismos linguísticos como o FUG tal distinção não era feita o que tornava ainda mais difícil um tratamento conveniente da negação e da disjunção. Por sua vez, numa linguagem lógica estas especificações podem ser incorporadas como combinações booleanas de restrições básicas (p.e. de equações). Assim, as representações em notação matricial ou por grafos acima apresentadas representam fórmulas que exprimem restrições em objectos linguísticos, e não os objectos linguísticos que satisfazem essas restrições. Esta distinção corresponde à distinção entre a sintaxe duma linguagem lógica (descrições) e a sua semântica (objectos linguísticos). Em cada formalização é dada uma noção de estrutura de atributos e uma linguagem lógica de descrições dessas estruturas, normalmente, denominada *linguagem de restrições*.

As abordagens divergem no modo como tratam a noção de *parcialidade* da informação. Na denominada "abordagem baseada em unificação" as estruturas de atributos são representações parciais dos objectos linguísticos, isto é, são a informação disponível sobre os objectos (num dado momento da computação). A unificação é vista como uma operação que permite a criação de estruturas de atributos sucessivamente mais específicas, que preservam a informação e que representam um objecto linguístico. Nesta perspectiva, se uma estrutura de atributos satisfaz uma restrição, então todas as estruturas de atributos subsumidas por ela devem satisfazer essa restrição. No entanto, considerar as estruturas de atributos objectos parciais conduz a dificuldades no tratamento das especificações negativas e, em particular, perde-se essa monotonia. Por exemplo, a estrutura de atributos

(1.12) $$\begin{bmatrix} agr & \begin{bmatrix} pers & 3rd \end{bmatrix} \end{bmatrix}$$

satisfaz a especificação negativa

(1.13) $$\begin{bmatrix} agr & \neg\begin{bmatrix} num & pl \end{bmatrix} \end{bmatrix}$$

Mas a estrutura de atributos (1.4) subsume (1.12) e não satisfaz (1.13).

O facto de uma estrutura de atributos não satisfazer uma descrição não significa que satisfaça a negação dessa descrição. Interpretações não-clássicas da negação, por exemplo intuicionistas, foram consideradas em [MR87, DVS90] mas o problema de satisfazibilidade associado embora decidível era computacionalmente intratável (**PSPACE**-completo). Historicamente, e sem ligação directa com formalismos linguísticos, o *Calculus* de ψ-termos de H. Aït-Kaci [AK86] foi a primeira formalização da unificação de estruturas de atributos interpretadas como objectos parciais. Os ψ-termos são classes de equivalência de estruturas de dados parcialmente ordenadas. Foram desenvolvidos para formalismos de representação de conhecimento, como generalização de termos de lógica 1ª ordem e são baseados nas teorias de tipos de linguagens de programação. Mas neste *Calculus* não é feita a distinção entre o domínio das estruturas de dados e o das suas descrições o que conduziu também a dificuldades técnicas no tratamento da negação e da disjunção.

Alternativamente, as estruturas de atributos podem ser consideradas objectos linguísticos completamente definidos (instanciados) e apenas as suas descrições conterem informação parcial sobre esses objectos. Esta abordagem foi adoptada por Kaplan e Bresnan [KB82], M. Johnson [Joh88] e G. Smolka [Smo89a]. Nas gramáticas lógicas com restrições ($\mathcal{CLG}$), que iremos apresentar na Secção 1.4, e que estão na base do trabalho desta dissertação, esta perspectiva foi também a adoptada. Uma estrutura de atributos satisfaz ou não uma descrição e as descrições são parciais no sentido em que caracterizam um conjunto de estruturas de atributos admissíveis. A relação entre estruturas de atributos e descrições (restrições) é a mesma que entre uma interpretação e uma fórmula que é verdadeira ou não nessa interpretação. A unificação corresponde ao teste da satisfazibilidade de uma conjunção de equações (restrições básicas). A subsunção corresponde à relação de vínculo entre restrições. A interpretação clássica (numa lógica de 1ª ordem) das descrições negativas não apresenta dificuldades, estas apenas indicam a ausência de determinados atributos e valores. A complexidade do problema da satisfazibilidade (em geral) não aumenta em relação ao duma linguagem que apenas envolva conjunções e disjunções.

Contudo, como é analisado em [Joh88], estas duas abordagens coincidem quando apenas se consideram conjunções de descrições sem negações.

Finalmente, as formalizações divergem no tipo de lógicas utilizadas para as linguagens de restrições. Ou são fragmentos da lógica de 1ª ordem com igualdade [Joh88, Smo89a, Joh91, DMV94] ou são linguagens "feitas por medida" (*custom built*) [KR86, RK86, Rea91, BS93b] que correspondem a lógicas não-clássicas, por exemplo, fragmentos de lógicas modais. As vantagens das primeiras parecem evidentes. Os resultados clássicos da lógica de 1ª ordem são directamente aplicados a estas linguagens, por exemplo, a completude do sistema dedutivo, a (in)decidibilidade do fragmento ou os algoritmos para determinar a satisfazibilidade das fórmulas. M. Johnson [Joh91, Joh94] argumenta que fragmentos decidíveis da lógica de 1ª ordem são suficientes para axiomatizar uma grande variedade de estruturas de atributos e para expressar restrições sobre elas. Ele exemplifica com a classe de fórmulas de 1ª ordem de *Schönfinkel-Bernays* que é constituída por fórmulas da forma $\exists x_1 \ldots x_n \forall y_1 \ldots y_m \phi$, onde ϕ é uma fórmula sem quantificadores nem símbolos funcionais.

Por outro lado, não há evidências que as potencialidades não-clássicas das linguagens "feitas por medida" sejam mesmo necessárias. L. Damas e outros em [DMV94] referem ainda que do ponto de vista computacional e em relação a modelos de processamento enquanto que os resultados obtidos nas áreas de Programação em Lógica e Programação por Restrições para teorias lógicas de 1ª ordem podem ser usados para implementações correctas e eficientes das gramáticas baseadas em restrições, o mesmo não acontece em relação a mecanismos de ordem superior que em geral não têm relação directa com as lógicas não-clássicas. Excepção são as linguagens de programação baseadas em lógicas de ordem superior como o λ-Prolog [NM88] e fragmentos de lógicas lineares. No entanto, e em geral, estas linguagens sofrem de problemas de eficiência e clareza semântica.

Todavia estes diferentes tipos de linguagens podem-se relacionar entre si, salientando as suas semelhanças. Para além de Johnson, vários autores evidenciaram a relação entre as várias linguagens de descrições de estruturas de atributos. G. Smolka [Smo89a] foi o primeiro a mostrar que a lógica de Kasper-Rounds corresponde a um fragmento da lógica de 1ª ordem com igualdade e, em particular, pode ser expressa na lógica de estruturas de atributos que desenvolveu. Reape [Rea91] e Blackburn [Bla91] estudaram lógicas de descrições de estruturas de atributos do ponto de vista modal, onde os atributos são modalidades, e mostraram que a lógica de Kasper-Rounds e suas extensões, correspondem a fragmentos de linguagens modais. Blackburn em [Bla94] apresenta uma abordagem, dita estrutural, em que as estruturas de atributos são modelos[2] para vários tipos de linguagens lógicas[3] e são construídas traduções, que preservam a satisfazibilidade, entre várias delas. Blackburn argumenta que as lógicas "feitas por medida", ao corresponderem a fragmentos de linguagens clássicas, podem ser vistas como estratégias para obter um balanço adequado entre expressividade e tratabilidade computacional. L. Damas e outros [DMV94] estabelecem uma relação entre as axiomatizações das álgebras de árvores racionais sobre símbolos funcionais e a axiomatização das lógicas de estruturas de atributos apresentada por Smolka. No Capítulo 3 apresentaremos essa relação considerando extensões da lógica de estruturas de atributos de Smolka.

[2]Álgebras da teoria de modelos, como serão apresentadas no Capítulo 2. Nesse capítulo também serão revistas outras noções básicas da lógica de 1ª ordem usadas nesta secção.

[3]de 1ª ordem, modal e proposicional dinâmica [Har84]

Apresentamos a seguir uma breve descrição de algumas das formalizações acima referidas. Para cada uma serão indicados os modelos para as estruturas de atributos e as linguagens de descrições associadas. Para uma revisão e comparação destas diferentes lógicas citamos [Kel93]. Rounds em [Rou97] faz uma revisão mais completa e actualizada das lógicas de estruturas de atributos, com ênfase na visão de estruturas de atributos como *objectos parciais*, e considerando várias maneiras de modelar matematicamente as estruturas de atributos (por exemplo, em relação às suas propriedades algébricas ou numa teoria de domínios). Curiosamente, Rounds considera como questões fundamentais em aberto a reconciliação entre as abordagens *objectos totais/parciais* e saber o que realmente significa considerar os objectos linguísticos *objectos parciais*.

Semântica denotacional de Pereira e Shieber. Em [PS84] Pereira e Shieber descrevem uma semântica denotacional que é considerada a primeira formalização das estruturas de atributos usadas nos formalismos linguísticos baseados em unificação. As estruturas de atributos são consideradas objectos parciais com um domínio distinto do das suas descrições. É usada a teoria de domínios das semânticas denotacionais de linguagens de programação [Sco82, Sto77]. Considerando um domínio L de atributos e um domínio A de constantes, o domínio dos valores V é dado como a solução duma equação recursiva de domínios

$$V \cong [L \longrightarrow V] + A$$

onde $[L \longrightarrow V]$ é um domínio de funções parciais de L em V. As estruturas de atributos, elementos de V, correspondem a árvores dirigidas cujos nós são valores e os arcos são etiquetados por atributos. Os termos da linguagem de descrições são *caminhos* $p \in L^\star$ e *constantes* $a \in A$. A denotação de um caminho $f_1 \dots f_n$ num valor complexo $v \in V$ é obtido por $(\dots v(f_1) \dots)(f_n)$. As fórmulas são conjunções de equações da forma $p_1 \doteq p_2$ ou $p_1 \doteq a$. Duas equações são inconsistentes se são da forma $p \doteq a$ e $p \doteq b$ ou $pq \doteq a$ e $p \doteq b$, onde a e b são constantes distintas e p e q são caminhos. Em [PS84] são apresentadas regras de inferência que permitem deduzir se uma descrição (conjunto de equações) é consistente. Pereira e Shieber constroem um domínio D, ordenado pela inclusão, que corresponde ao conjunto de todas as equações consistentes e fechadas para o sistema dedutivo, e onde as noções de subsunção e unificação são bem definidas.

Este modelo denotacional é suficiente para caracterizar as descrições do formalismo PATR-II [SUP+83], mas a sua generalização para linguagens de descrições com negação e disjunções apresenta dificuldades. Uma extensão a especificações disjuntivas, usando a teoria dos domínios e, em particular, o domínio potência de Smyth [GS90], é apresentada em [Rou97].

Lógica de Kasper-Rounds. Foi introduzida em [KR86, RK86, KR90] para caracterizar a semântica das especificações disjuntivas e a partilha de valores em estruturas de atributos, vistas como objectos parciais, que ocorriam no formalismo FUG. As descrições de estruturas de atributos são fórmulas duma lógica proposicional interpretadas em conjuntos de grafos dirigidos. Partindo dum conjunto L de atributos e um conjunto A de constantes, as estruturas de atributos são representadas por autómatos finitos determinísticos (AFD) $\mathcal{A} = (S, L, \delta, s_0, F)$ que reconhecem sequências de *atributos* (caminhos), com estados finais que podem ser elementos de A e onde δ é uma função parcial de $S \times L$ em S. Os AFDs são conexos e não têm transições dos estados finais. Têm várias propriedades que são desejáveis para o domínio das estruturas de atributos: o valor de um caminho é identificado por um estado; dois caminhos que unificam têm o mesmo estado como valor; e as técnicas e resultados da teoria de autómatos podem ser directamente aplicadas. A relação de subsunção entre AFDs pode ser definida por um morfismo entre os seus grafos, constituindo uma pré-ordem. As fórmulas atómicas da linguagem de descrições são da forma $p_1 \doteq p_2$, com $p_1, p_2 \in L^\star$, e a, com $a \in A$. Duas constantes especiais são $\top$ (Verdade) e $\perp$ (Falso). Se ϕ e ψ são fórmulas e l um atributo então $\phi \ \wedge \ \psi$, $\phi \ \vee \ \psi$ e $l : \phi$ são fórmulas. Nas fórmulas da forma $l : \phi$, l corresponde a uma modalidade das lógicas modais como a lógica proposicional dinâmica [Har84]. Num estado s, $l : \phi$ verifica-se se $\delta(s, l) = s'$ está definido e ϕ verifica-se em s'. Por exemplo, a estrutura de atributos representada em (1.11) corresponde à fórmula

(1.14) $$agr : ((num : sg \ \wedge \ gen : fem) \ \vee \ num : pl) \ \wedge \ case : (nom \ \vee \ acc)$$

A semântica desta lógica é definida pela relação de satisfação entre um AFD $\mathcal{A}$ e uma fórmula ϕ, $\mathcal{A} \models \phi$, dada por

(i) $\mathcal{A} \models \top$;

(ii) $\mathcal{A} \not\models \perp$;

(iii) $\mathcal{A} \models a$, se $\mathcal{A}$ é um autómato com apenas um estado (final) a e sem transições;

(iv) $\mathcal{A} \models p_1 \doteq p_2$ se $p_1 \mathcal{N}_{\mathcal{A}} p_2$, onde $\mathcal{N}_{\mathcal{A}}$ é a relação de equivalência de Nerode definida em $P(\mathcal{A}) = \{p \in L^\star \mid \delta(s_0, p)\text{está definida}\}$ por $p \mathcal{N}_{\mathcal{A}} p'$ se e só se $\delta(s_0, p) = \delta(s_0, p')$;

(v) $\mathcal{A} \models l : \phi$ se $\mathcal{A}/l \models \phi$, onde $\mathcal{A}/l$ é o subgrafo de $\mathcal{A}$ com estado inicial $\delta(s_0, l)$;

(vi) $\mathcal{A} \models \phi \ \vee \ \psi$ se $\mathcal{A} \models \phi$ ou $\mathcal{A} \models \psi$;

(vii) $\mathcal{A} \models \phi \ \wedge \ \psi$ se $\mathcal{A} \models \phi$ e $\mathcal{A} \models \psi$.

A conjunção corresponde à unificação e $\perp$ indica que a unificação falhou. A relação de satisfação $\models$ é monótona em relação à subsunção de autómatos. Kasper e Rounds apresentam um *Calculus* com regras de equivalência lógica entre fórmulas que é usado para determinar a satisfazibilidade. Mostram que o problema da satisfazibilidade para esta lógica é **NP**-completo e Kasper [Kas87] apresenta um algoritmo para testar a satisfazibilidade de duas descrições que evita a expansão em forma normal disjuntiva, que será analisado na Secção 1.4.3.2. A extensão desta lógica a fórmulas negativas seria imediata acrescentando à definição da relação de satisfação $\models$ a condição $\mathcal{A} \models \neg\phi$ se $\mathcal{A} \not\models \phi$. No entanto, como atrás referido, a relação $\models$ deixa de ser monótona, o que não é desejável. O problema reside, mais uma vez, no facto de se pretender que as estruturas de atributos sejam objectos parciais. Em certos casos, mesmo linguisticamente esta exigência não parece muito razoável. Como sugere Pereira [Per87], o conjunto de atributos admissíveis para algumas estruturas de atributos deveriam ser especificados. Por exemplo, para caracterizar a concordância *agr* os atributos admissíveis seriam apenas *num* e *pers*, cada um deles admitindo apenas valores constantes. Algumas soluções para a extensão desta lógica à negação foram propostas em [MR87, Kas88, DVS90, Car90, Car92]. Em particular, Carpenter [Car90, Car92], baseado no formalismo HPSG, apresenta um sistema de tipos para as estruturas de atributos, onde os tipos para além de formarem uma hierarquia de herança múltipla ([Car84]) possuem uma condição de admissibilidade de atributos, que permite distinguir entre atributos cujos valores são desconhecidos e atributos que não são apropriados para um dado tipo. Deste modo, algumas especificações negativas são tratadas directamente pelo sistema de tipos. Especificando os atributos apropriados para cada tipo, Carpenter admite também estruturas de atributos *extensivas*, isto é, que são identificadas pelos seus atributos e valores.

Lógica de Atributos-Valores de Johnson. M. Johnson [Joh88] desenvolveu esta lógica em parte como formalização das gramáticas funcionais léxicas LFG. Supõe que as estruturas de atributos representam informação completa sobre os objectos linguísticos. Na definição de estruturas de atributos, que são designadas de *estruturas atributo-valor*, não é feita a distinção[4] entre atributos e valores. Uma estrutura atributo-valor é um triplo $\mathcal{A} = (F, C, \delta)$ onde F é o conjunto dos elementos atributo-valor, $C \subseteq F$ é um conjunto de constantes e $\delta : F \times F \rightharpoonup F$ é uma função parcial tal que $\delta(c, f)$ é indefinida para todos $c \in C$ e $f \in F$.

[4] Por requisitos do formalismo LFG.

Por exemplo, a estrutura de atributos

$$(1.15)\qquad {}_{a}\begin{bmatrix} cat & np \\ agr & {}_{b}\begin{bmatrix} num & sg \\ pers & 3rd \end{bmatrix} \end{bmatrix}$$

pode ser formalizada por $\mathcal{A} = (F, C, \delta)$ onde $F = \{a, b, cat, np, agr, num, sg, pers, 3rd\}$, $C = F \setminus \{a, b\}$, $\delta(a, cat) = np$, $\delta(a, agr) = b$, $\delta(b, num) = sg$ e $\delta(b, pers) = 3rd$.

A linguagem de descrições de estruturas atributo-valor utiliza variáveis que denotam elementos atributo-valor. Sendo V um conjunto de variáveis e A um conjunto de constantes, os termos da linguagem são as variáveis, as constantes e expressões da forma $t_1(t_2)$, onde t_1 e t_2 são termos. As fórmulas atómicas são $\top$, $\perp$ e $t_1 \approx t_2$, onde t_1 e t_2 são termos. Se ϕ e ψ são fórmulas então $\neg\phi$, $\phi \wedge \psi$ e $\phi \vee \psi$ também são fórmulas. Um modelo atributo-valor é um quíntuplo $\mathcal{M} = (F, C, \delta, \phi, \chi)$, onde (F, C, δ) é uma estrutura atributo-valor, ϕ uma função de V em F e χ é uma função injectiva entre A e C. A denotação $[\![t]\!]$ de um termo t num modelo $\mathcal{M}$ é um elemento de F. Se t é uma variável $[\![t]\!] = \phi(t)$, se é uma constante $[\![t]\!] = \chi(t)$ e se for da forma $t_1(t_2)$, $[\![t]\!] = [\![t_1(t_2)]\!] = \delta([\![t_1]\!], [\![t_2]\!])$. A relação de satisfação entre modelos e fórmulas é definida por

(i) $\mathcal{M} \models \top$;

(ii) $\mathcal{M} \not\models \perp$;

(iii) $\mathcal{M} \models t_1 \approx t_2$ se $[\![t_1]\!]$ e $[\![t_2]\!]$ estão definidas e $[\![t_1]\!] = [\![t_2]\!]$;

(iv) $\mathcal{M} \models \neg\phi$ se $\mathcal{M} \not\models \phi$;

(v) $\mathcal{M} \models \phi \vee \psi$ se $\mathcal{M} \models \phi$ ou $\mathcal{M} \models \psi$;

(vi) $\mathcal{M} \models \phi \wedge \psi$ se $\mathcal{M} \models \phi$ e $\mathcal{M} \models \psi$.

Johnson apresenta um sistema dedutivo para as fórmulas válidas e prova a sua integridade e completude para os modelos atributo-valor. Construindo uma tradução desta lógica na teoria de lógica de 1ª ordem sem quantificadores, com símbolos funcionais e com igualdade [NO80], mostra que o problema da satisfazibilidade é **NP**-completo. Em [Joh90], Johnson conclui que a lógica de atributos-valores é essencialmente um subconjunto da lógica de 1ª ordem. Considerando-se que a unificação é apenas uma operação para manipular informação linguística, o facto de se "abdicar" do conceito de parcialidade da informação parece razoável comparado com o ganho na clareza no tratamento da negação e da disjunção e na possibilidade de usar técnicas e algoritmos bem conhecidos e pelo menos tão eficientes como os métodos de unificação. Johnson mostra ainda como se integra essa lógica na representação das relações gramaticais do estilo das do formalismo LFG, obtendo um sistema formal único para as gramáticas desse formalismo.

Lógica de Estruturas de Atributos de Smolka. O trabalho de Smolka [Smo89b, Smo89a, Smo92] é uma tentativa de capturar num só formalismo lógico a lógica de Kasper-Rounds, o *Calculus* de ψ-termos de Aït-Kaci e a lógica de atributos-valores de Johnson e, de um modo geral, caracterizar as propriedades matemáticas das estruturas de atributos e das suas descrições usadas nas gramáticas de unificação.

O seu principal mérito foi mostrar que as diferentes linguagens de descrições podem ser consideradas sub-linguagens da lógica de 1ª ordem com igualdade. Por outro lado, em vez de um único domínio de interpretação[5] para as fórmulas das linguagem de descrições, Smolka considera uma classe de interpretações. Essas interpretações são as *álgebras de atributos*. Partindo duma linguagem da lógica de 1ª ordem com igualdade e onde as constantes e os símbolos de predicado binários (atributos) são os únicos símbolos não lógicos, as álgebras de atributos são definidas como habitualmente com a condição dos atributos serem interpretados como funções parciais unárias e as constantes serem interpretadas como elementos distintos do universo e para os quais não existem atributos definidos. Um exemplo de álgebra de atributos é a álgebra dos grafos de atributos onde o universo é o conjunto dos grafos de atributos. Os grafos de atributos são grafos dirigidos com raiz, conexos e finitos, cujos arcos são etiquetados por atributos e as folhas etiquetadas com constantes. Coincidem essencialmente com os AFDs da lógica de Kasper-Rounds. Em particular, a relação de subsunção em grafos de atributos pode ser generalizada, via homomorfismos, para álgebras de grafos de atributos e mesmo para toda a classe das álgebras de atributos.

A linguagem das descrições é uma linguagem da lógica de 1ª ordem com igualdade construída a partir de um conjunto de variáveis e um conjunto de símbolos constantes e de símbolos de predicado binários. Os termos são as variáveis e as constantes. As fórmulas atómicas são $f(s,t)$ e $s \doteq t$, onde s e t são termos e f é um atributo. As fórmulas $f(s,t)$ são normalmente representadas por sft ou $s.f \doteq t$ para evidenciar o carácter funcional dos atributos: o termo s tem o valor t para o atributo f. As fórmulas da linguagem, denominadas restrições, são as que se formam a partir das fórmulas atómicas usando as conectivas e os quantificadores habituais. Para uma álgebra de atributos $\mathcal{A}$, uma atribuição de valores às variáveis e para

[5] que nas lógicas anteriores era identificado com o conjunto das estruturas de atributos

uma restrição ϕ a relação de satisfação $\mathcal{A} \models_\rho \phi$ define-se como habitualmente na lógica de 1ª ordem com igualdade. Em particular, ϕ é satisfazível se existe uma álgebra $\mathcal{A}$ e uma atribuição de valores às variáveis ρ tal que $\mathcal{A} \models_\rho \phi$. A classe das álgebras de atributos pode ser axiomatizada pelas seguintes fórmulas:

FA1. $\forall(xfy \wedge xfz \Rightarrow y \doteq z)$ para todo o atributo f

FA2. $\forall(afx \Rightarrow \bot)$, para toda a constante a

FA3. $a \neq b$, se a e b são constantes distintas

Smolka mostrou que o problema da satisfazibilidade de restrições sem quantificadores é **NP**-completo, mas para restrições em geral o problema é indecidível. Mais ainda, a álgebra de grafos de atributos é canónica no sentido em que uma restrição é satisfazível numa qualquer álgebra de atributos se e só se o for na álgebra de grafos.

Para relacionar a linguagem de restrições com a lógica de Kasper-Rounds, Smolka introduziu *termos de atributos* que são expressões que denotam conjuntos nas álgebras de atributos e que são uma generalização das descrições da lógica de Kasper-Rounds. Um grafo de atributos (na lógica de Kasper-Rounds) satisfaz um *termo de atributos* se e só se o grafo pertence à denotação do termo na álgebra dos grafos de atributos. Para usar termos de atributos na linguagem de restrições, a linguagem é estendida a restrições da forma $x : S$, onde x é uma variável e S um termo de atributos, e que denotam que a interpretação de x pertence à denotação de S.

No entanto, mostrando que restrições que envolvam restrições da forma $x : S$, podem ser transformadas, em tempo linear, em restrições equivalentes sem restrições daquela forma, Smolka conclui que os termos de atributos são apenas uma extensão sintáctica da sua linguagem de restrições.

Embora os axiomas FA1-FA3 capturem o essencial das propriedades das estruturas de atributos, são demasiado fracos no sentido que admitem álgebras em que predicados são sempre falsos. É necessário mais um axioma que garanta a satisfazibilidade de certas restrições consistentes. As álgebras canónicas para a axiomatização resultante, ditas álgebras de árvores de atributos têm similitudes com as álgebras das árvores racionais axiomatizadas por M. Maher [Mah88]. Essa relação foi apresentada em [DMV94] e constitui o assunto central do Capítulo 3.

1.3.2.3 Comparação com as Gramáticas Lógicas e Termos

Nesta secção comparamos as gramáticas de unificação e as estruturas de atributos com as gramáticas lógicas e os termos da lógica de 1ª ordem. Como se disse na Secção 1.3.1, as gramáticas lógicas podem ser consideradas formalismos baseados em restrições, onde a linguagem de restrições é constituída por conjunções de equações de termos da lógica de 1ª ordem. Os termos (finitos ou racionais) podem ser vistos como estruturas de atributos com um número fixo de atributos e onde os atributos são identificados com as posições dos argumentos. Por exemplo, a estrutura de atributos (1.1) pode ser representada pelo termo *np(agr(3rd,X))*, onde X é uma variável. A tradução de estruturas de atributos como termos envolve a introdução de variáveis para especificar valores dos atributos que não estão instanciados.

A co-indexação de informação é feita pela partilha de variáveis e os termos racionais correspondem a estruturas de atributos que admitem ciclos. O puro mecanismo de unificação de termos não permite, por exemplo, unificar a estrutura de atributos (1.3) com a estrutura de atributos (1.2), mas isso pode ser feito pela resolução duma conjunção de equações entre termos,

$$Y \doteq np(Z) \ \wedge \ Z \doteq agr(3rd, X) \ \wedge \ W \doteq agr(U, sg) \ \wedge \ Z \doteq W \tag{1.16}$$

Apesar da aparente limitação, supondo que as estruturas de atributos são objectos *totalmente especificados* e são conhecidos os seus atributos admissíveis, as estruturas de atributos são apenas um *embelezamento* sintáctico dos termos. Os objectos *intencionais* podem ser representados com termos, considerando que cada termo tem um argumento suplementar que não é especificado (instanciado) e que permite a adição de mais "informação". Em vez de se identificar as posições de um termo com os atributos podemos explicitar cada atributo e o seu valor. Por exemplo, usando termos da forma $f(A, V, S)$ onde A é o atributo, V é o seu valor e S ou é variável ou é um termo da mesma forma que representa o resto da estrutura de atributos. Para a estrutura de atributos (1.1) teríamos o termo

$$f(cat, np, f(agr, f(pers, 3rd, X), Y))$$

com X e Y variáveis. A unificação com a estrutura de atributos (1.3) conduziria ao termo

$$f(cat, np, f(agr, f(pers, 3rd, f(num, sg, W)), Y))$$

considerando a resolução duma conjunção de equações semelhante a (1.16).

Assim, a unificação de estruturas de atributos e as suas extensões à resolução de restrições complexas corresponde, essencialmente, à extensão de linguagens lógicas como o Prolog ao esquema $\mathcal{CLP}$ [JM94] da programação lógica por restrições sobre os domínios dos termos finitos ou racionais.

Em particular, as linguagens de restrições que serão introduzidas no Capítulo 2, permitem manipular termos evitando a introdução de variáveis supérfluas e, coincidem, sob condições que serão expostas no Capítulo 3, com as linguagens de descrições de estruturas de atributos.

1.3.2.4 Estrutura das Gramáticas de Unificação

Conceptualmente uma gramática de unificação pode ser dividida numa componente que representa a estrutura de frase e numa linguagem de restrições. A estrutura de frase pode ser especificada por uma gramática independente de contexto em que as regras de produção são anotadas por restrições. Essencialmente, as estruturas de atributos e as suas descrições são uma generalização das categorias atómicas (não-terminais) usadas nessas gramáticas[6]. Uma das primeiras formalizações das gramáticas de unificação foi apresentada por Johnson em [Joh88] para o LFG.

Informalmente, vamos ilustrar apresentando regras duma mini-gramática, para o Inglês, baseada no formalismo PART-II:

$$x_0 \rightarrow x_1\ x_2 \tag{1.17}$$
$$x_0.cat \doteq s \ \wedge\ x_1.cat \doteq np \ \wedge\ x_2.cat \doteq vp \ \wedge\ x_1.agr \doteq x_2.agr$$

$$x_0 \rightarrow x_1 \tag{1.18}$$
$$x_0.cat \doteq vp \ \wedge\ x_1.cat \doteq v \ \wedge\ x_0.agr \doteq x_1.agr$$

$$x_0 \rightarrow \texttt{John} \tag{1.19}$$
$$x_0.cat \doteq np \ \wedge\ x_0.agr \doteq z \ \wedge\ z.pers \doteq 3rd \ \wedge\ z.num \doteq sg$$

$$x_0 \rightarrow \texttt{runs} \tag{1.20}$$
$$x_0.cat \doteq v \ \wedge\ x_0.agr \doteq z \ \wedge\ z.pers \doteq 3rd \ \wedge\ z.num \doteq sg$$

$$x_0 \rightarrow \texttt{run} \tag{1.21}$$
$$x_0.cat \doteq v \ \wedge\ x_0.agr \doteq z \ \wedge\ (z.pers \neq 3rd \ \vee\ z.num \neq sg)$$

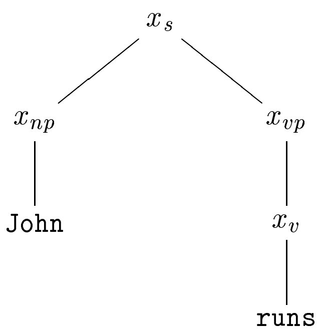
Os x_i são variáveis lógicas que correspondem às estruturas de atributos de cada constituinte da frase (não-terminais), e que têm de satisfazer as restrições respectivas. `John`, `runs` e `run`

[6]Sobre gramáticas e linguagens independentes de contexto ver [HU79]. Note-se que, embora existam raras evidências de que as (algumas) LNs não são linguagens independentes contexto [Shi85], alguns autores [PG82, GKPS85, GP85] são da opinião que este tipo de gramáticas são suficientes para descrever a maioria das construções sintácticas das LNs.

são terminais. Uma regra de produção (independente de contexto), indica que a sub-frase do constituinte à esquerda é a concatenação das sub-frases à direita. As restrições da forma $x.f \doteq v$ indicam que a variável x tem o valor v para o atributo f. Normalmente, e para eficiência da análise sintáctica, identificam-se os não-terminais com valores de categorias atómicas (principais). A regra seguinte é uma abreviatura da regra (1.17)

$$
\begin{array}{lcl}
s & \rightarrow & np\ vp \\
 & & x_1.agr \doteq x_2.agr
\end{array}
$$

Uma derivação pode ser representada por uma árvore (de derivação) sintáctica onde os nós são anotados com restrições. Por exemplo, uma árvore sintáctica correspondente à frase "`John runs`" é:



$$
\begin{array}{l}
x_s.cat \doteq s \ \wedge\ x_{np}.cat \doteq np \ \wedge\ x_{vp}.cat \doteq vp \\
\wedge\ x_{np}.agr \doteq x_{vp}.agr \\
\wedge\ x_{np}.agr \doteq z_1 \ \wedge\ z_1.pers \doteq 3rd \ \wedge\ z_1.num \doteq sg \\
\wedge\ x_v.cat \doteq v \ \wedge\ x_{vp}.agr \doteq x_v.agr \\
\wedge\ x_v.agr \doteq z_2 \ \wedge\ z_2.pers \doteq 3rd \ \wedge\ z_2.num \doteq sg
\end{array}
$$

Quando uma regra é aplicada a uma folha duma árvore sintáctica anotada, a árvore é expandida segundo a regra, sendo a variável desse nó identificada com a variável do lado esquerdo da regra e as restantes variáveis da regra renomeadas para variáveis novas. As restrições anotadas na regra são conjugadas com as já associadas à árvore. Uma frase `s` é aceite pela gramática se é possível gerar, aplicando as regras, (pelo menos) uma árvore sintáctica cuja *sequência de terminais* é `s`, isto é, cujas folhas são etiquetadas pelas palavras de `s` (ordenadas da esquerda para a direita) e as restrições associadas à árvore são satisfazíveis.

Por exemplo, a frase "`John run`" não é aceite pela gramática apresentada porque a conjunção de restrições associada à sua árvore sintáctica não é satisfazível.

Para este tipo de gramáticas, o problema de determinar se uma frase é aceite, denominado o *problema de reconhecimento universal*, é em geral indecidível[7]. Isto, apesar de tanto o problema de reconhecimento universal para gramáticas independentes de contexto como o problema da satisfazibilidade das restrições serem ambos decidíveis. Por exemplo, Johnson [Joh88] mostrou a indecidibilidade desse problema para as gramáticas em que as regras de estrutura de frase são anotadas por fórmulas da lógica de atributos-valores e que

[7] embora, semi-decidível

correspondem essencialmente às gramáticas funcionais léxicas LFG. O problema torna-se decidível se se impuserem condições sobre as árvores sintácticas que são construídas. Uma dessas condições, conhecida por *análise sintáctica off-line* [KB82, PW83, Joh88] garante que o número de árvores sintácticas associadas a uma frase seja limitado por uma função computável no tamanho da frase.

No entanto, como salienta Johnson [Joh88, Joh94] não há nenhuma evidência que o problema de reconhecimento universal para linguagens naturais seja indecidível. Até porque nós somos capazes de o fazer ... Portanto parece razoável a investigação de formalismos linguísticos que embora usem um sistema decidível de restrições sejam linguisticamente expressivos e computacionalmente tratáveis.

1.3.2.5 Extensões e Decidibilidade

Alguns formalismos linguísticos baseados em restrições, em especial o HPSG, aboliram o *esqueleto* independente de contexto considerando apenas um sistema lógico único, autónomo[8] e baseado em descrições de estruturas de atributos. Nestes formalismos, o problema do reconhecimento universal reduz-se a um problema de satisfazibilidade.

Dado que as estruturas de frase são representadas por árvores, estas podem ser descritas por estruturas de atributos que codifiquem listas de elementos. Mesmo sem tomar aquela posição radical, alguns formalismos linguísticos utilizam listas nas descrições de estruturas de atributos para representar a informação sobre a estrutura sintáctica das frases. Uma lista de elementos $< x_1, \ldots, x_n >$ pode ser representada por uma estrutura de atributos cuja descrição, em notação matricial, é

$$\begin{bmatrix} \textit{cabeça} & x_1 \\ \textit{resto} & \begin{bmatrix} \textit{cabeça} & x_2 \\ \textit{resto} & \begin{bmatrix} \ldots & & \\ & \textit{resto} & \begin{bmatrix} \textit{cabeça} & x_n \\ \textit{resto} & \textit{nil} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

onde *nil* representa a lista vazia.

Para simplificar a notação, vamos supor que um atributo pode admitir como valor uma lista. Para codificar uma árvore sintáctica usamos o atributo *dtrs* para a lista das estruturas de frase que ocorrem directamente como sucessores dum nó e o atributo *phon* para a

[8] Esta classe de formalismos gramaticais é denominada na literatura em língua inglesa por *stand-alone*.

sequência de palavras (terminais) correspondente ao constituinte. Assim, a árvore sintáctica representada anteriormente para a frase "`John runs`" corresponde à seguinte descrição, em notação matricial,

$$\begin{bmatrix} cat & s \\ phon & \langle \texttt{John runs} \rangle \\ dtrs & \left\langle \begin{bmatrix} cat & np \\ phon & \langle \texttt{John} \rangle \end{bmatrix} , \begin{bmatrix} cat & vp \\ phon & \langle \texttt{runs} \rangle \\ dtrs & \left\langle \begin{bmatrix} cat & v \\ phon & \langle \texttt{runs} \rangle \end{bmatrix} \right\rangle \end{bmatrix} \right\rangle \end{bmatrix}$$

onde foram omitidas as restrições referentes à concordância sujeito-verbo.

No entanto, para exprimir o facto de uma estrutura de atributos ser uma lista ou para se definir a concatenação de duas listas, por exemplo, para obter o valor do atributo *phon* acima, a expressividade das linguagens de descrições que apresentamos até aqui não é suficiente. Por outro lado, para exprimir as regras gramaticais (ou princípios gramaticais) vai ser necessário referir um determinado nó de uma árvore, que pode ser expresso por um caminho de atributos *dtrs* de tamanho, possivelmente, variável. Em resumo, a linguagem de restrições tem de permitir a expressão de algum tipo de definição indutiva.

Uma extensão natural das linguagens de restrições é a incorporação de expressões regulares [HU79] que denotem conjuntos, possivelmente infinitos, de caminhos de atributos. Sendo α uma expressão regular sobre o conjunto de atributos, uma restrição da forma $x\alpha \doteq y$ denota que existe um caminho representado pela expressão que satisfaz a equação.

Conjunções deste tipo de restrições, foram usadas no formalismo LFG [KZ86, KI88] para descrever um fenómeno linguístico denominado *dependências de longa distância*. Esse tipo de construções ocorre quando existe uma relação de dependência entre dois constituintes que não estão na relação antecessor-sucessor na árvore sintáctica. Exemplos disso ocorrem em frases relativas, em *topicalizações*, frases interrogativas ou frases com *buracos*. Em Inglês, temos os seguintes exemplos de topicalizações

a) `Mary, Paul saw ___`

b) `Mary Peter said that Paul saw ___`

c) `Mary John believed that Peter said that Paul saw ___`

d) ...

O tópico das frases – "`Mary`" – coincide com o objecto directo da frase em a),com o objecto directo da frase relativa em b) e com o objecto directo da frase relativa da frase relativa em c) .... Identificando s com a representação de cada uma das frases e x com a representação do tópico, podemos exprimir este tipo de fenómeno por uma restrição da forma

$$s.topic \doteq x \;\wedge\; s.comp^{\star}obj \doteq x$$

onde $comp^{\star}$ é uma expressão regular que corresponde a um caminho com zero ou mais ocorrências do atributo *comp*.

Kaplan e Maxwell descrevem em [KI88] um algoritmo para determinar a satisfazibilidade de restrições com expressões regulares cujos modelos admissíveis não contenham ciclos. Keller [Kel93] estende a lógica de Kasper-Rounds a expressões regulares, obtendo a lógica regular de Kasper-Rounds, e mostra a decidibilidade do problema da satisfazibilidade, mantendo a condição das estruturas de atributos serem acíclicas. Backofen [Bac94a, Bac94b, Bac96] generaliza estes resultados eliminando a condição adicional.

Embora as restrições com expressões regulares sobre atributos aumentem a expressividade da linguagem de restrições, permitindo "disjunções infinitas" de restrições, as conjunções dessas restrições não são suficientes para, por exemplo, exprimir regras de estrutura de frase ou outras generalizações sobre as estruturas de atributos. No entanto, a combinação dessas restrições com outras operações e quantificadores lógicos torna a linguagem suficientemente expressiva mas (com o problema de satisfazibilidade) indecidível. Keller [Kel93] mostra que o problema da satisfazibilidade para a lógica regular de Kasper-Rounds acrescida dum operador de negação, interpretado classicamente, é indecidível embora o formalismo lógico resultante seja suficiente para descrever vários aspectos das estruturas de frase, por exemplo, as relações de dominância entre nós.

Backofen [Bac94a] obtém o mesmo resultado de indecidibilidade estendendo a restrições com expressões regulares sobre caminhos, uma linguagem de lógica de 1ª ordem semelhante à lógica de estruturas de atributos de Smolka, e que será apresentada no Capítulo 3. Considerando ainda uma generalização dessas linguagens que admite variáveis que denotam atributos, Backofen mostra que é possível definir relações recursivas sobre estruturas de atributos, semelhantes às usadas em Programação em Lógica e suficientes para a expressividade dos formalismos linguísticos autónomos. No entanto, para essa linguagem de estruturas de atributos generalizada, Treinen [Tre93] já tinha demonstrado a sua indecidibilidade.

Ainda na tentativa de obter lógicas de estruturas de atributos suficientemente expressivas, Blackburn e Spaan [BS93b, BS93a] apresentam uma linguagem da lógica modal, correspondente à lógica de Kasper-Rounds com negação e aumentada com um operador implicação.

Esta linguagem possuí duas das características fundamentais dos formalismos autónomos: a possibilidade de descrever estruturas de atributos reentrantes e a possibilidade de exprimir generalizações sobre estruturas de atributos. Mas, mais uma vez, o problema da satisfazibilidade é indecidível para esta linguagem. Blackburn e Spaan [BS93a] consideram alguns fragmentos decidíveis da linguagem mas nenhum deles é suficientemente expressivo para formalismos autónomos.

Contudo, estes resultados de indecidibilidade não devem surpreender dado que, como foi referido na Secção 1.3.2.4, o problema de reconhecimento universal para as gramáticas de unificação existentes é em geral indecidível. No entanto, nos formalismos autónomos e ao contrário dos formalismos híbridos, é mais difícil a imposição de condições, como a condição da *análise sintáctica off-line*, que garantam a decidibilidade [Joh94, BS93a]. Isto é, a elegância e a simplicidade dum formalismo uniforme, que é desejável do ponto de vista linguístico e mesmo formal, dificulta o acesso a informação estruturada que seria, do ponto de vista computacional, fundamental. Ao contrário, nos formalismos híbridos cada componente pode ser independentemente optimizada, assim como a sua interacção.

Claro que isso poderá ser obtido nos formalismos autónomos se as lógicas de estruturas de atributos permitirem uma análise mais fina da estrutura interna da informação linguística. Uma das vantagens seria poder estudar as características computacionais das gramáticas como um todo. A favor desta perspectiva é o facto de que a complexidade de cada componente dum sistema híbrido, nomeadamente da linguagem de restrições, não permitir estabelecer limites nem inferiores nem superiores da complexidade do sistema. Trautwein [Tra95] mostra que assim como combinando duas linguagens decidíveis se obtém uma indecidível, também considerando uma linguagem de descrições cujo problema da satisfazibilidade é de complexidade quadrática, e uma estrutura de frase definida por uma gramática regular, a gramática de unificação resultante tem um problema do reconhecimento universal decidível mas **NP**-completo. Por outro lado, se uma componente contiver informação sobre a outra ou se a gramática não usar toda a expressividade duma componente, a complexidade das componentes podem não ser limites inferiores da complexidade do sistema total.

Terminamos esta secção apresentando um exemplo dum formalismo autónomo e referindo ainda outras extensões.

HPSG: Exemplo de um formalismo autónomo. No formalismo HPSG [PS87b, PS94] a informação linguística é lexicalizada[9] e as generalizações são utilizadas para definir um

[9]No sentido em que o léxico contém informação que determina a combinação de componentes.

sistema de tipos e para exprimir *princípios* universais que definem relações gramaticais que reduzem drasticamente o número de regras de estrutura de frase. Os princípios determinam a projecção da informação léxica e definem recursivamente o fluxo de informação na estrutura global. O sistema de tipos permite que cada tipo descreva um género de objectos linguísticos para os quais um conjunto de atributos é apropriado. Os subtipos são definidos a partir duma ordem parcial nos tipos básicos e se um atributo é apropriado para um tipo é apropriado para qualquer subtipo desse tipo.

No HPSG o tipo básico é `sign` que tem como atributos *PHON*, *SYN* e *SEM*, que correspondem respectivamente à informação fonológica, sintáctica e semântica dos objectos linguísticos. Os subtipos de `sign` são o tipo `lexical`, para as entradas lexicais, e o tipo `pharsal`, para representar frases, e o qual admite também o atributo *DTRS*, que corresponde à estrutura de frase. A informação sintáctica é dividida em *local* e *não-local*. A local contém informação sobre a categoria sintáctica do objecto, incluindo informação que será usada pela cabeças de frase (*HEAD*), e a sua subcategorização[10] (*SUBCAT*), constituída por uma lista de `signs`. Esta informação, em conjunto com alguns princípios, permite reduzir o número das regras de estrutura de frase. A informação não-local corresponde a dependências de longa distância como topicalização, frases relativas, etc e sobre a qual também actuam princípios gerais.

Simplificando, as entradas lexicais para as palavras "`John`" e "`runs`", usadas na gramática anterior, podem ser descritas respectivamente por

$$\begin{bmatrix} PHON & \langle \texttt{John} \rangle \\ SYN|LOC & \begin{bmatrix} HEAD|CAT & n \\ LEX & + \\ SUBCAT & \langle\rangle \end{bmatrix} \end{bmatrix} \quad \text{e} \quad \begin{bmatrix} PHON & \langle \texttt{runs} \rangle \\ SYN|LOC & \begin{bmatrix} HEAD|CAT & v \\ LEX & + \\ SUBCAT & \langle NP \rangle \end{bmatrix} \end{bmatrix}$$

onde *NP* é uma abreviatura de

$$\begin{bmatrix} SYN|LOC & \begin{bmatrix} HEAD|CAT & n \\ SUBCAT & \langle\rangle \end{bmatrix} \end{bmatrix}$$

Como exemplo de princípios universais, consideramos o Princípio da Cabeça e da Subcategorização, que controlam o fluxo de informação nos atributos *HEAD* e *SUBCAT*. Estes princípios podem ser formulados conjuntamente da seguinte forma

$$\begin{array}{lll} [DTRS] & \Rightarrow & SYN.LOC.HEAD \doteq DTRS.H\_DTR.SYN.LOC.HEAD \quad \& \\ & & \texttt{append}(DTRS.C\_DTRS, SYN.LOC.SUBCAT, \\ & & \qquad DTRS.H\_DTR.SYN.LOC.SUBCAT) \end{array}$$

[10] Para verbos, a subcategorização corresponde ao conjunto dos seus complementos.

onde o objecto parcialmente especificado do lado esquerdo tem de verificar as restrições expressas do lado direito. Como habitualmente, a relação `append` definida em listas verifica-se se o seu terceiro argumento é a concatenação dos dois primeiros. Estes princípios podem ser traduzidos num sistema de tipos, considerando que a interpretação do tipo do objecto é restringida ao conjunto de elementos que verificam as restrições.

Finalmente nestas gramáticas deste formalismo existe um pequeno conjunto de regras de estrutura de frase que especificam como é estruturada em termos de dominância a informação do atributo $DTRS$, isto é, como são combinados os constituintes.

As regras

$$\begin{array}{lcl} [SUBCAT\ \langle \_ \rangle] & \to & C^\star,\ H\_DTR\,[LEX\ \ +] \\ [SUBCAT\ \langle\rangle] & \to & C,\ H\_DTR\,[LEX\ \ -] \end{array}$$

permitem combinar os dois constituintes da frase "`John runs`". Note-se contudo que estas regras não fazem referência nem a categorias sintácticas nem à ordem linear da ocorrência dos constituintes na frase, podendo ser aplicadas em muitos outros tipos de frases.

Aplicando também os princípios acima definidos, obtemos a descrição de "`John runs`" que é parcialmente representada na Figura 1.1.

$$\left[\begin{array}{ll}
PHON & \langle \texttt{John runs}\rangle \\
SYN|LOC & \left[\begin{array}{ll} HEAD & \boxed{1} \\ SUBCAT & \langle\rangle \\ LEX & - \end{array}\right] \\
DTRS & \left[\begin{array}{ll}
C_DTRS & \left\langle \boxed{2} \left[\begin{array}{ll} PHON & \langle\texttt{John}\rangle \\ SYN|LOC & \left[\begin{array}{ll} HEAD|CAT & n \\ SUBCAT & \langle\rangle\end{array}\right]\end{array}\right]\right\rangle \\
H_DTR & \left[\begin{array}{ll} PHON & \langle\texttt{runs}\rangle \\ SYN|LOC & \left[\begin{array}{ll} HEAD & \boxed{1} \\ SUBCAT & \langle\boxed{2}\rangle \\ LEX & - \end{array}\right]\end{array}\right] \\
DTRS & \left[\begin{array}{ll} C_DTRS & \langle\rangle \\ H_DTR & \left[\begin{array}{ll} PHON & \langle\texttt{runs}\rangle \\ SYN|LOC & \left[\begin{array}{ll} HEAD & \boxed{1}\,[CAT\ \ v] \\ SUBCAT & \langle\boxed{2}\rangle \\ LEX & + \end{array}\right]\end{array}\right]\end{array}\right]
\end{array}\right]
\end{array}\right]$$

Figura 1.1: Representação da frase "`John runs`"

Em conclusão, uma gramática G do formalismo HPSG pode ser definida como a conjunção de todos os princípios conjugada com a disjunção das entradas lexicais e das regras

$$G = P_1 \wedge \ldots \wedge P_n \wedge (L_1 \vee \ldots \vee L_m \vee R_1 \vee \ldots R_r)$$

Nomeadamente, para a representação de informação semântica, o HPSG usa ainda descrições de conjuntos – ou multiconjuntos – de estruturas de atributos. Existem na literatura algumas extensões das lógicas de estruturas de atributos para incorporar descrições de conjuntos, que preservam a decidibilidade. Johnson [Joh91] utilizando fórmulas de Schönfinkel-Bernays axiomatiza algumas operações sobre conjuntos não extensivos. Manandhar [Man93b, Man94] generaliza os *termos de atributos* de Smolka, a descrições de conjuntos mas omitindo a negação e a disjunção. Manandhar mostra que nessas condições, o problema da satisfazibilidade é **NP**-completo e apresenta uma tradução da sua lógica em fórmulas de Schönfinkel-Bernays. A semântica de lógicas de estruturas de atributos com descrições de conjuntos foram também estudadas em [Rou88, PM90, Rea94, Lan94].

Baseados em sistema de tipos para estruturas de atributos, existem vários formalismos computacionais cujo principal objectivo foi a implementação de gramáticas do estilo do HPSG. Entre esses sistemas salientamos o CUF [Joc93], o ALE [Car94a], o TFS [Zaj92] e o $\mathcal{TDL}/\mathcal{UDINE}$ [Kri95, Han94, BK93]. Para uma descrição destes e de outros formalismos computacionais para gramáticas de unificação referimos [BKSU93]. Manandhar [Man93a] faz uma comparação entre os três primeiros focando essencialmente os aspectos de expressividade e eficiência. O facto de incorporarem duma maneira mais ou menos uniforme a definição de tipos recursivos leva a que sejam, em geral, computacionalmente ineficientes. Nos sistemas TFS e $\mathcal{TDL}/\mathcal{UDINE}$ as definições de tipo, são suficientemente gerais para permitir a definição de princípios gramaticais. Em ALE e CUF isso não é possível e os princípios são introduzidos através de *cláusulas definidas* à lá Prolog. Deste modo, estes sistemas podem ser considerados como instâncias do paradigma de Programação em Lógica por Restrições $\mathcal{CLP}$. Em particular, $\mathcal{UDINE}$ é um resolutor de restrições sobre o fragmento existencial da lógica de estruturas de atributos de Smolka. ALE e CUF usam lógicas de descrições de estruturas de atributos que são generalizações da lógica de Kasper-Rounds. No entanto, ao se aproximarem de linguagens de programação em lógica gerais, e usando apenas algumas estratégias de controlo da resolução e esquemas de retardamento específicos, estes sistemas para além de serem bons ambientes para o desenvolvimento de protótipos gramaticais, dificilmente são formalismos computacionais eficientes para o PLN. Mais uma vez, de algum modo, os formalismos usados parecem poderosos demais para os fins desejados embora a expressividade pareça a conveniente.

1.4 $\mathcal{CLG}$: Gramáticas Lógicas com Restrições

$\mathcal{CLG}$ (*Constraint Logic Grammars*) é uma família de formalismos gramaticais computacionais baseados na resolução de restrições complexas e que foram desenvolvidos e implementados por L. Damas e outros desde 1988 [DV89a, BDMV90, DMV91, Dam91, DV92, DMV94, DM95a].

Um dos objectivos do projecto $\mathcal{CLG}$ foi testar em que medida e com que benefícios as técnicas usadas em Programação em Lógica por Restrições podiam ser adaptadas para o PLN. Em particular, considerando as características principais dos formalismos linguísticos baseados em unificação existentes e usando resolução de restrições complexas, pretendia-se construir sistemas com uma semântica denotacional íntegra e computacionalmente eficientes. Ao contrário de abordagens mais recentes que advogam usar directamente linguagens de programação (lógica) por restrições, como o Prolog III [Col90], o LIFE [AKP93a] ou o Oz [HSW93], para codificar directamente as gramáticas de LN, pretendia-se utilizar a metodologia e as técnicas de implementação dessas linguagens mas restringí-las aos formalismos linguísticos.

Nas Secções 1.4.1 e 1.4.2 são apresentados dois elementos desta família. Na Secção 1.4.3 é apresentada a metodologia do $\mathcal{CLG}$, sendo focados o seu modelo formal e de processamento.

1.4.1 $\mathcal{CLG}(0)$

Na origem deste projecto esteve o desenho de um formalismo linguístico computacional suficientemente expressivo mas computacionalmente aceitável para implementar as (complicadas) gramáticas do projecto europeu de tradução automática multilíngua Eurotra [Com90]. Mantendo o desenho base das gramáticas de unificação e do formalismo existente, as gramáticas do $\mathcal{CLG}(0)$ [Dam88, DV89a] são constituídas pela especificação das estruturas de atributos admissíveis e por um conjunto de regras gramaticais anotadas com restrições complexas. Estas anotações, que também ocorrem nas entradas lexicais, permitem uma diminuição substancial no número de regras nessas gramáticas, em relação às do formalismo do Eurotra. A incorporação de um resolutor de restrições complexas a par de um analisador sintáctico, semelhante ao usado pelas DCGs mas com controlo da recursão à esquerda, levou a um aumento drástico do desempenho do sistema em relação ao do formalismo existente. O protótipo $\mathcal{CLG}(0)$ [DV89a, DV89b, LMD91] foi testado para gramáticas relativamente grandes de várias línguas europeias e os resultados mostraram que uma implementação eficiente de gramáticas de unificação com restrições complexas era exequível. Em [DV89a] é apresentada uma semântica denotacional, não só para a linguagem de restrições complexas, mas também para toda a gramática. Esta semântica é simplificada pelo facto das estruturas de atributos serem consideradas objectos totalmente especificados e os valores dos atributos serem apenas constantes[11] ou estruturas de atributos complexas não recursivas. Os objectos linguísticos, correspondentes a constituintes de frase, são representados por árvores de estruturas de atributos. O significado duma gramática $\mathcal{CLG}(0)$, definida por um *gerador*, é dado por uma função de reconhecimento do domínio dos objectos, nos valores de verdade.

Cada regra duma gramática $\mathcal{CLG}(0)$ é constituída por um esqueleto independente de contexto[12] e restrições da forma

(1.22) $$X_0[X_1, \ldots, X_n] : c_1 \ \wedge \ \ldots \ \wedge \ c_k$$

Por exemplo, a regra (1.17) da gramática apresentada na Secção 1.3.2.4 seria

(1.23) $$\begin{aligned} X_0[X_1, X_2] :\ & X_0.cat \doteq s \ \wedge \ X_1.cat \doteq np \ \wedge \\ & X_2.cat \doteq vp \ \wedge \ X_1.num = X_2.num \ \wedge \ X_1.pers = X_2.pers \end{aligned}$$

Para além de conjunções de equações, a linguagem de restrições admite disjunções e negações, interpretadas classicamente.

O facto de não se usarem estruturas de atributos recursivas, garante a decidibilidade do problema do reconhecimento universal, para estas gramáticas.

1.4.2 $\mathcal{CLG}(2)$

O $\mathcal{CLG}(2)$ foi desenhado para a implementação de gramáticas do estilo do formalismo HPSG. É usado um sistema de tipos para as estruturas de atributos que é codificado em termos Prolog. A linguagem de restrições admite como fórmulas atómicas equações $t_1 \doteq t_2$ entre termos e predicados $r(t_1, \ldots, t_n)$, que são definidos por expressões da forma

$$r(x_1, \ldots, x_n) \ \Leftrightarrow \ \mathit{Restrições}$$

A linguagem de restrições incluí ainda expressões com caminhos que permitem a referência a argumentos específicos dum termo complexo, sem a necessidade de introduzir variáveis extra. Por exemplo, se t representar o termo $f(a, g(c, d))$ as seguintes restrições são satisfeitas:

$$t.f^2 \qquad t.f_2^2.g^2 \qquad t.f_1^2 \doteq a$$

[11] como no formalismo existente de Eurotra

[12] A sintaxe dos esqueletos é aqui simplificada pela omissão de um operador prefixo associado a cada X_i e que indicaria *opcionalidade* ou *iteração* desse elemento.

Em [DMV91] é apresentado o modelo de processamento do $\mathcal{CLG}(2)$ para as restrições que não envolvem relações definidas recursivamente. Esse processamento baseia-se no uso dum modelo parcial (uma conjunção de restrições atómicas ou um termo) e na reescrita (incompleta) das restrições complexas usando esse modelo.

Para as restrições que envolvem relações definidas recursivamente são exigidas condições de aplicação que garantam que a sua avaliação "falha finitamente" [BDMV90].

Na Figura 1.2 apresenta-se um fragmento duma gramática do $\mathcal{CLG}(2)$. Cada gramática é constituída por:

- uma declaração do sistema de tipos;
- uma especificação de sub-tipos considerando restrições da forma

  `tipo := <tipo> where <restrições>`
- uma especificação de regras usando descrições parciais de objectos

  `<phrasal_sign> -> <constituintes> where <restrições>`
- uma definição de relações
- restrições globais que implementam princípios do estilo dos do HPSG

  `<princ> := <descrição_parcial_objecto> -> <restrições>`

Em [BDMV90] é apresentada uma semântica denotacional para estas gramáticas, a sua expressividade é ilustrada pelo tratamento de certos fenómenos linguísticos e a implementação do formalismo $\mathcal{CLG}(2)$ é descrita genéricamente. Em particular, são resolvidos alguns problemas levantados por uma implementação "directa" dos princípios do HPSG. É também sugerido que a complexidade envolvida na formalização e implementação deste sistema é demasiada e, talvez, desnecessária para o PLN.

1.4.3 Metodologia do $\mathcal{CLG}$

Baseado no desenvolvimento e implementação dos sistemas $\mathcal{CLG}(0)$, $\mathcal{CLG}(2)$ e de vários outros protótipos [BDMV90, Dam91, DMV91], e no estudo das suas características formais e computacionais foi proposto em [Dam91, DMV94] uma teoria formal e um esquema de processamento para formalismos gramaticais que descrevemos sumariamente nesta secção.

Consideram-se essenciais para um formalismo linguístico computacional os requisitos seguintes:

```
sign_definition
        sign = (phon, syn, dtrs);
        phon = word list;
        syn = (synf, local, bind);
                synf = {lex, compl};
                local = (head, subcat, lex);
                        head = (maj, nform, vform, agr);
                                maj = {n, v, adj, adv, det};
                                agr = (num, pers);
                                        num = {sg, pl};
                                        pers = {fst, snd, thrd};
                                nform = {there, it, norm};
                                vform = {fin, bse};
                        subcat = sign list;
                        lex = {+, -};
                bind = (slash, wh);
                        slash = sign list;
                        wh = {rel, que};
        dtrs = (hdtr, odtr);
                hdtr = sign;
                odtr =  union([cdtrs, adtr]);
                        cdtrs = sign list;
                        adtr = sign;
constrainted_types
        lexical :=  sign   where dtrs = nil & syn.local.lex = +;
rules
        h_comp = [syn.synf=compl]  -> dtrs.hdtr, dtrs.odtr.cdtrs
                where
                dtrs.hdtr type lexical,
                syn.local.lex = -,
                syn.local.subcat = [],
                forall(dtrs.odtr.cdtrs,[syn.local.subcat = []]);
user_relations
        iterate([],Z,Z);
        iterate(X:L1,X:L2,X.syn.bind.slash::R1) <- iterate(L1,L2,R1);
        iterate(X:L1,Y:L2,Y:R) <- iterate(X:L1,L2,R);

principles
        % princípio da cabeça
        hfp = ([] where syn.local.head~=nil) ->
                        syn.local.head = dtrs.hdtr.syn.local.head;
        % princípio da subcategorização
        cgi = [syn.synf=compl] ->
            iterate(dtrs.odtr.cdtrs,dtrs.hdtr.syn.local.subcat,syn.bind.slash);
        neh = [syn.synf=compl] -> ~(dtrs.hdtr.syn.local.subcat=[]);
```

Figura 1.2: Fragmento duma gramática do $\mathcal{CLG}(2)$

ser declarativo reflectindo o facto dum sistema declarativo permitir a separação entre a descrição da informação e o modo como é processada, munindo o utilizador duma semântica clara e deixando ao implementador do sistema liberdade suficiente para escolher estratégias de implementação efectivas;

expressividade o poder expressivo da linguagem de descrições deve permitir o uso de restrições complexas;

integridade deve ser fornecida uma semântica denotacional que caracterize os objectos linguísticos que a gramática reconhece ou gera;

tratabilidade computacional o formalismo deve ser *implementável* e em particular permitir usar técnicas da Programação por Restrições.

Para os requisitos serem atingidos os formalismos deveriam incluir contribuições da Lógica de 1ª ordem e da Programação em Lógica. A metodologia proposta, que é uma generalização das dos sistemas $\mathcal{CLG}$ existentes, baseia-se em considerar qualquer gramática uma teoria particular da lógica de 1ª ordem com igualdade.

A Lógica Computacional, e a Programação em Lógica em particular, é um paradigma ideal para este formalismo uma vez que suporta uma tradução directa entre as gramáticas e a sua implementação. Simultaneamente, fornece um esquema computacional formalmente íntegro e eficiente. Esta abordagem não só é compatível com a tradição das gramáticas de unificação mas também permite generalizar de um modo simples a noção de unificação à de resolução de restrições complexas. E o carácter declarativo é obtido ao não se fazer referência a nenhuma operação como a *unificação*.

Para além de ser feita uma clara distinção entre os objectos linguísticos e as suas descrições parciais, os objectos são *entidades totais* e, computacionalmente, apenas são manipuladas descrições parciais de objectos.

Um aspecto inovador desta metodologia é constatar que não é suficiente desenhar formalismos com uma semântica íntegra e estudar a sua complexidade formal, é essencial fazer uma análise mais fina das suas propriedades computacionais. Isto é, num formalismo gramatical computacional a sua teoria formal deve ser acompanhada de esquemas de processamento, devendo ser estudadas as suas características computacionais.

1.4.3.1 Álgebra de Árvores Racionais

É necessário determinar como e em que circunstâncias as gramáticas de unificação com restrições complexas podem ser consideradas teorias da lógica de 1ª ordem com igualdade.

Essas teorias são extensões da teoria $\mathcal{RT}$, a axiomatização completa das álgebras de árvores racionais apresentada por Maher [Mah88]. Dito de outro modo, dada uma linguagem de 1ª ordem com a igualdade como único símbolo de predicado, as únicas interpretações admissíveis são as álgebras de árvores racionais. A escolha destas interpretações tem várias motivações:

- têm sido usadas como modelo em vários domínios computacionais e existem alguns resultados formais e computacionais que podem ser utilizados e generalizados;
- a teoria $\mathcal{RT}$ é decidível;
- sob certas condições existe uma relação entre $\mathcal{RT}$ e as lógicas de estruturas de atributos de Smolka; e, deste modo, $\mathcal{RT}$ pode ser usada como a teoria para as descrições de estruturas de atributos;
- existe uma linguagem sem quantificadores em que se podem exprimir as fórmulas de $\mathcal{RT}$ e que é muito semelhante às linguagens das lógicas de estruturas de atributos; essa linguagem será introduzida no Capítulo 2;
- usando essa linguagem, o modelo de processamento de restrições sobre termos de 1ª ordem desenvolvido para as gramáticas $\mathcal{CLG}(2)$ [DMV91] pode ser adaptado para um algoritmo de eliminação de quantificadores e permitir um tratamento exequível de restrições expressas como fórmulas de $\mathcal{RT}$.

1.4.3.2 Modelo de Processamento de Restrições Complexas

Como já se disse, a introdução de restrições complexas sobre estruturas de atributos levou a que a verificação da sua satisfazibilidade passasse a ser um problema **NP**-hard.

Do ponto de vista computacional e desde a primeira implementação do $\mathcal{CLG}(0)$ [DV89a], o formalismo $\mathcal{CLG}$ assenta na suposição de que a maior parte das especificações linguísticas sejam conjunções de restrições. Esta suposição é razoável pois é de esperar que o conhecimento linguístico seja expresso primordialmente duma forma afirmativa.

Assim, o processamento de restrições complexas baseia-se num esquema de reescrita parcial que constrói uma descrição parcial dum objecto, representada por uma conjunção de equações[13] satisfazíveis, e um conjunto de restrições complexas envolvendo disjunções e

[13] Ou por um termo da lógica de 1ª ordem com variáveis não instanciadas que ocorrem nas restrições complexas [DMV91].

negações, que vai sendo parcialmente (re)avaliado, e que o objecto tem de satisfazer. A avaliação das restrições complexas é retardada até que elas possam ser avaliadas como falsas, como verdadeiras ou levarem à adição de alguma equação à descrição parcial do objecto. Deste modo é assegurada a tratabilidade computacional do algoritmo. Este esquema de avaliação é íntegro mas não é completo no sentido em que não é suficiente para garantir a satisfazibilidade da descrição parcial do objecto, quando ainda existem restrições complexas que não foram avaliadas como falsas. Isto é, pelo menos num passo final, para obter a completude é necessário aplicar um conjunto de regras que permitam a obtenção duma forma normal disjuntiva (ou forma normal conjuntiva).

O facto de que no fim do processamento poderem ainda permanecer restrições complexas, que não foram avaliadas como verdadeiras, permite que o sistema não falhe por não ter resolvido todas as restrições. Do ponto de vista linguístico, isto é desejável já que as descrições gramaticais não são suficientemente precisas para que todas e só as restrições que são necessárias sejam expressas pela gramática e toda e só a informação que é necessária para satisfazer as restrições é que está disponível. Por outro lado, o facto de que as estruturas de dados intermédias e final são descrições parciais, que representam uma classe de objectos e não apenas um, corresponde ao facto das descrições gramaticais serem, em geral, incompletas.

Este modelo contrasta claramente com os modelos que se baseiam no cálculo sistemático de formas normais para avaliarem as restrições complexas [Kay85b] e modelos em que as restrições são parte integrante dos objectos [Kas87, Car90, Car92]. Note-se que, apesar dos inconvenientes, a óbvia a vantagem da redução a forma normal disjuntiva seria que os algoritmos de unificação poderiam ser aplicados directamente a cada elemento da disjunção.

No entanto, na tentativa de minimizar essa redução foram propostas na literatura algumas outras técnicas das quais salientamos as seguintes:

Disjunção restrita a valores [Kar84] São apenas admitidas disjunções nos valores de um atributo. A localidade das disjunções, conduz a algoritmos mais eficientes mas perde-se o poder expressivo da linguagem de descrições. O método torna-se complicado e menos eficiente quando os valores são expressos por caminhos de atributos.

Aproximações sucessivas [Kas87] Representando as estruturas de atributos como grafos dirigidos, Kasper estende o algoritmo de unificação (usual) de duas estruturas de atributos[14] à unificação de duas descrições de estruturas de atributos com disjunções. Cada descrição é escrita na forma $conj \wedge disj_1 \wedge \ldots \wedge disj_n$, onde $conj$ não contém

[14] Considerando que uma descrição não disjuntiva corresponde a uma estrutura de atributos minimal [Kas87].

disjunções e $disj_i$, para $1 \leq i \leq n$, são disjunções de descrições da mesma forma. O algoritmo constrói uma nova descrição em 3 passos: no primeiro são unificadas as partes não disjuntivas de cada uma das descrições; no segundo é verificada a consistência (satisfazibilidade) de cada elemento das disjunções com a nova descrição não disjuntiva; no terceiro é verificada a consistência de grupos de k elementos de uma disjunção com as restantes disjunções (k-consistência), $k = 1, \ldots, m$. Este método permite detectar inconsistências dum modo incremental. Para aumentar a eficiência, o último passo, que pode conduzir à expansão em forma normal disjuntiva, é só efectuado em pontos estratégicos da computação.

Factorização de restrições em subconjuntos independentes [MK91] Baseia-se na observação de que em geral numa frase os constituintes distantes contribuem com informação linguística independente, de modo que as suas possíveis alternativas podem ser "resolvidas" localmente. Assim, numa frase, o número de descrições alternativas consistentes é muito menor do que aquele obtido por redução a forma normal disjuntiva. Retomando a vantagem das disjunções restritas a valores, o objectivo é embeber as disjunções nos atributos a que se referem as alternativas. O método é constituído pelos seguintes passos:

(i) As descrições com disjunções são transformadas numa conjunção de restrições com contexto, da forma $P \Rightarrow \phi$ onde ϕ é uma descrição não disjuntiva e P é uma combinação booleana de variáveis proposicionais (contexto). A regra básica usada na transformação é

$$\phi_1 \vee \phi_2 \rightarrow (p \Rightarrow \phi_1) \wedge (\neg p \Rightarrow \phi_2) \tag{1.24}$$

onde p é uma variável proposicional nova.

(ii) Os algoritmos para testar a satisfazibilidade de sistemas puramente conjuntivos são adaptados para as restrições com contexto permitindo obter uma forma normal com contexto.

(iii) Na forma normal, se para $P \Rightarrow \phi$, a fórmula ϕ não for satisfazível então para a conjunção ser satisfazível $\neg P$ tem de ser verdade. A conjunção destas expressões é chamada *resíduo disjuntivo*.

(iv) Para cada solução do resíduo disjuntivo, usando (1.24), é possível obter uma conjunção de fórmulas do sistema inicial que corresponde a um elemento da forma normal disjuntiva (da descrição inicial).

Note-se que no passo (iii) é necessário determinar a satisfazibilidade duma fórmula proposicional o que é também um problema **NP**-completo.

Disjunção distribuída [ED88, DE90] Este método é uma generalização da disjunção de valores e do método anterior, permitindo qualquer tipo de disjunções mas mantendo-as localizadas pela associação de um nome a cada disjunção. Por exemplo, na lógica de Kasper-Rounds em vez da descrição

$$(syn : agr : case : dat \wedge sem : rel : s\_in) \vee (syn : agr : case : acc \wedge sem : rel : d\_in)$$

tem-se a disjunção distribuída

$$syn : agr : case : (dat \vee_{d_1} acc) \wedge sem : rel : (st\_in \vee_{d_1} d\_in)$$

A partir dos termos de atributos, introduzidos por Smolka [Smo89a, Smo92], Eisele e Dörre utilizam uma linguagem de descrições que permite exprimir disjunções com contexto e apresentam um *Calculus* para testar a satisfazibilidade. Ao contrário do método anterior, os contextos não são combinações booleanas de variáveis proposicionais mas apenas variáveis booleanas (cujos valores correspondem a cada uma das alternativas duma disjunção).

Todos estes métodos tentam evitar a redução a forma normal disjuntiva e/ou diminuir o tamanho das fórmulas às quais tal redução tem de se aplicar. O método usado nas implementações dos formalismos $\mathcal{CLG}$, de factorização das restrições pela construção de um modelo parcial, pode ser visto como uma generalização dos 2 primeiros passos do método descrito em [Kas87]. Por outro lado, a factorização de restrições em conjunções de restrições independentes, por exemplo, que não tenham variáveis em comum, permite à semelhança do método descrito em [MK91] e das disjunções distribuídas, *localizar* as disjunções que têm de ser expandidas. Para ilustrar o método do $\mathcal{CLG}$ exposto acima, vamos considerar um exemplo de [MK91] expresso na lógica de atributos de Smolka e que foi apresentado em [DV92]. A fórmula seguinte corresponde à conjunção dos atributos referentes à concordância artigo–nome da informação lexical associada às palavras, em língua alemã, *die* e *Koffer*:

(1.25)

$$\begin{gathered}(x.case \doteq nom \vee x.case \doteq acc) \wedge ((x.gend \doteq fem \wedge x.num \doteq sg) \vee x.num \doteq pl) \\ \wedge \\ x.gend \doteq masc \wedge x.pers \doteq 3 \wedge ((x.num \doteq sg \wedge x.case \neq gen) \vee \\ \hfill (x.num \doteq pl \wedge x.case \neq dat))\end{gathered}$$

Considerando as conjunções de equações de topo, podemos concluir que qualquer modelo da fórmula acima subsume a estrutura de atributos

$$
{}_{x}\begin{bmatrix} gend & masc \\ pers & 3 \end{bmatrix}
$$

Usando esta informação e os Axiomas FA1–FA3 da lógica de estruturas de atributos, podemos reescrever (1.25) para

$$
\begin{array}{ll}
(1.26) & (x.case \doteq nom \ \vee\ x.case \doteq acc) \ \wedge\ ((\bot \ \wedge\ x.num \doteq sg) \ \vee\ x.num \doteq pl) \\
& \wedge \\
& \top \ \wedge\ \top \ \wedge\ ((x.num \doteq sg \ \wedge\ x.case \neq gen) \ \vee \\
& \qquad (x.num \doteq pl \ \wedge\ x.case \neq dat))
\end{array}
$$

que, usando teoremas clássicos da lógica de 1ª ordem, pode ser simplificada para

$$
\begin{array}{ll}
(1.27) & (x.case \doteq nom \ \vee\ x.case \doteq acc) \ \wedge\ (x.num \doteq pl) \\
& \wedge \\
& ((x.num \doteq sg \ \wedge\ x.case \neq gen) \ \vee\ (x.num \doteq pl \ \wedge\ x.case \neq dat))
\end{array}
$$

Mais uma vez, considerando as conjunções de equações de topo, podemos concluir que qualquer modelo da fórmula original subsume a estrutura de atributos

$$
{}_{x}\begin{bmatrix} gend & masc \\ pers & 3 \\ num & pl \end{bmatrix}
$$

Usando esta informação e simplificando (1.27) reduz-se a

$$
(1.28) \qquad (x.case \doteq nom \ \vee\ x.case \doteq acc) \ \wedge\ x.case \neq dat
$$

Não havendo mais conjunções de restrições equacionais, e não sendo possível factorizar as restrições, seria necessário expandir esta fórmula para forma normal disjuntiva. Contudo o tamanho da fórmula (1.28) é substancialmente menor do que o da fórmula original e foi obtida, em tempo polinomial, preservando a equivalência. Note-se ainda que neste exemplo, como qualquer modelo de cada um dos elementos da disjunção é modelo do segundo elemento da conjunção, podemos concluir que a fórmula (1.27) é satisfazível sem efectuar a expansão. Este exemplo mostra ainda que existem várias estratégias que podem ser utilizadas e exploradas antes da expansão em forma normal disjuntiva.

1.4.3.3 $\mathcal{CLG}$ e $\mathcal{CLP}(\mathcal{RT})$

Embora $\mathcal{RT}$ seja uma teoria adequada para descrever classes de objectos linguísticos, as regras gramaticais e os princípios, excepto para os casos mais simples, requerem alguma forma de definições recursivas. Uma possibilidade é introduzir símbolos de predicados para exprimir classes de objectos na gramática e regras gramaticais como axiomas definindo esses predicados. Contudo, é desejável restringir os axiomas gramaticais a cláusulas de Horn com restrições da forma

$$A \leftarrow C \ \wedge \ A_1 \ \wedge \ \ldots \ \wedge \ A_n$$

onde A e A_i, para $1 \leq i \leq n$, denotam fórmulas atómicas da forma $p(t_1, \ldots, t_k)$ com p um símbolo de predicado de aridade k e $t_1, \ldots, t_k$ termos; e C é uma restrição, *i.e*, uma fórmula lógica envolvendo apenas o símbolo de predicado igualdade. Uma teoria lógica deste tipo pode ser vista como um programa de programação em lógica por restrições, p.e, em $\mathcal{CLP}(\mathcal{RT})$ [JM94], e, portanto, admite uma semântica clara e bem definida. Em [DMV94], é apresentado um modelo de processamento para gramáticas do $\mathcal{CLG}$ vistas como programas em $\mathcal{CLP}(\mathcal{RT})$, e que generaliza o sistema de reescrita de restrições complexas.

Como já se disse, uma vantagem desta escolha é podermos beneficiar dos resultados teóricos e de técnicas de implementação eficientes desenvolvidas na área de Programação Lógica por Restrições. No entanto, será conveniente fornecer sempre condições de aplicabilidade dos predicados definidos recursivamente, que garantam a decidibilidade do sistema.

Capítulo 2

Linguagens de Restrições

Neste capítulo introduzimos as linguagens de restrições que vamos estudar neste trabalho. Nomeadamente, linguagens da lógica de 1ª ordem com a igualdade como único símbolo de predicado interpretadas em álgebras de árvores. Na Secção 2.1 recordamos alguns conceitos e resultados matemáticos que iremos utilizar. Na Secção 2.2 serão definidas as linguagens de restrições apropriadas para o estudo formal e computacional do problema da satisfazibilidade. Finalmente, na Secção 2.3, relacionamos essas linguagens com linguagens de restrições utilizadas actualmente na área de Programação por Restrições.

2.1 Linguagem $\mathcal{L}$

Introduzimos nesta secção alguns conceitos e resultados fundamentais da lógica de 1ª ordem [And86] e notação a usar. Apresentamos também alguns resultados gerais sobre sistemas de reescrita e a sua aplicação em problemas de satisfazibilidade. Na Secção 2.1.4 serão apresentadas as interpretações para as linguagens de lógica 1ª ordem que iremos adoptar neste trabalho.

2.1.1 Sintaxe

Vamos apenas considerar linguagens da lógica de 1ª ordem com a igualdade como único símbolo de predicado.

Seja $\mathcal{L}$ uma linguagem de 1ª ordem construída a partir de um conjunto infinito numerável de variáveis $X = \{x, y, z, x_1, y_1, \ldots\}$; uma assinatura $\Sigma = \{f^n, g^n, h^n, \ldots\}$ constituída, para cada $n \geq 0$, por um conjunto numerável de símbolos de função de aridade n; o símbolo de predicado $=$; as conectivas $\neg$, $\vee$ e $\wedge$; o quantificador $\forall$; os parênteses '(' e ')' e o símbolo $\top$ (valor lógico verdade). Os símbolos de função de aridade zero são denotados por $a, b, c, d, a_1, \ldots$ e designados por *constantes* e os de aridades superiores são designados *functores* (ou funcionais). Utilizamos também as conectivas $\Rightarrow$ e $\Leftrightarrow$ sendo as expressões $\phi \Rightarrow \psi$ e $\phi \Leftrightarrow \psi$ abreviaturas de $(\neg\phi \vee \psi)$ e $((\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi))$, respectivamente. Analogamente, $\exists x\phi$ é uma abreviatura de $\neg\forall x\neg\phi$, $t_1 \neq t_2$ é uma abreviatura de $\neg(t_1 = t_2)$ e $\perp$ de $\neg\top$. Se omitirmos parênteses numa fórmula, as prioridades das conectivas são as usuais, *i.e*, consideramos por ordem decrescente de precedência os quantificadores, as conectivas $\neg$, $\wedge$, $\vee$, $\Rightarrow$ e $\Leftrightarrow$.

Uma linguagem $\mathcal{L}$ fica completamente caracterizada por X e Σ. Designaremos por $\mathcal{L}$ qualquer uma dessas linguagens.

Uma sequência finita de símbolos de $\mathcal{L}$ diz-se uma *expressão*. A identidade de símbolos ou expressões de $\mathcal{L}$ é denotada por $\equiv$ e a sua distinção por $\not\equiv$.

Definição 2.1 *O conjunto de termos $T = \{t, r, s, u, t_1, t_2, \ldots\}$ sobre Σ e X é o menor conjunto tal que:*

(i) qualquer variável é um termo;

(ii) qualquer constante é um termo;

(iii) se f^n é um functor de aridade $n \geq 1$ e $t_1, \ldots, t_n$ são termos então $f(t_1, \ldots, t_n)$ é um termo.

Um *termo complexo* é um termo que não é uma variável nem uma constante. Um *termo fechado* é um termo que não contém variáveis.

O facto de u ser um sub-termo de t denota-se por $t[u]$. A $t[\ ]$ chama-se um contexto. Um termo da forma $f(t_1, \ldots, t_n)$ pode ser denotado por $f(\bar{t})$, para qualquer símbolo de função f^n.

Definição 2.2 *O tamanho $|t|$ um termo t, é definido indutivamente na estrutura de t:*

(i) $|x| = 1$ se x é uma variável;

(ii) $|a| = 1$ se a é uma constante;

(iii) $|f(t_1, \ldots, t_n)| = 1 + \sum_{i=1}^{n} |t_i|$.

Definição 2.3 *Uma fórmula atómica é $\top$ ou uma expressão da forma $t_1 = t_2$ onde t_1 e t_2 são termos; uma expressão desta forma designa-se também por equação. O conjunto das fórmulas de $\mathcal{L}$ é o menor conjunto tal que:*

(i) uma fórmula atómica é uma fórmula;

(ii) se ϕ é uma fórmula, $\neg\phi$ é uma fórmula;

(iii) se ϕ e ψ são fórmulas, $(\phi \wedge \psi)$ e $(\phi \vee \psi)$ são fórmulas;

(iv) se ϕ é uma fórmula e x é uma variável, então $\forall x\phi$ é uma fórmula;

Se $\phi \equiv Qx_1Qx_2 \ldots Qx_n\psi$, onde Q representa um mesmo quantificador ($\exists$ ou $\forall$) e x_i são variáveis, escreve-se $\phi \equiv Qx_1, \ldots, x_n\psi$ ou apenas $\phi \equiv Q\bar{x}.\psi$. Uma fórmula ϕ está em *forma prenexa normal* se não contiver quantificadores ou se for da forma $Q_1x_1 \ldots Q_nx_n.\psi$, onde cada Q_i é $\exists$ ou $\forall$ e ψ é uma fórmula sem quantificadores, denominada *matriz* de ϕ. Uma fórmula sem quantificadores está em *forma normal disjuntiva* (*forma normal conjuntiva*) se for uma disjunção (conjunção) de conjunções (disjunções) de fórmulas atómicas ou das suas negações.

O facto de ϕ ser uma sub-fórmula de ψ denota-se por $\psi[\phi]$. O *tamanho* duma fórmula ϕ define-se como o número de símbolos de ϕ.

Definição 2.4 *Uma variável x ocorre livre numa fórmula ϕ se:*

(i) ϕ é uma fórmula atómica e x ocorre em ϕ;

(ii) ϕ é $\neg\psi$ e x ocorre livre em ψ;

(iii) ϕ é $\psi \wedge \gamma$ ou $\psi \vee \gamma$ e x ocorre livre em ψ ou γ;

(iv) ϕ é $\forall y\psi$ e $x \not\equiv y$ e x ocorre livre em ψ.

Uma variável ocorre ligada *numa fórmula ϕ se não ocorrer livre.*

Denotamos por $\mathsf{Vars}(\phi)$ o conjunto de todas as variáveis que ocorrem livres numa fórmula ϕ. Uma *fórmula fechada* é uma fórmula em que nenhuma variável ocorre livre. Se ϕ é uma fórmula, então $\forall\phi$ denota o *fecho universal* de ϕ, que é a fórmula fechada que se obtém de ϕ adicionando um quantificador universal para cada variável que ocorre livre em ϕ. Analogamente, definimos $\exists\phi$, o *fecho existencial* de ϕ.

Definição 2.5 *Uma substituição é uma função $\theta : X \longrightarrow T$, tal que, se $x\theta$ denotar o valor de θ em x o conjunto*

$$\mathcal{D}(\theta) = \{x \in X \mid x\theta \not\equiv x\}$$

é finito e denomina-se domínio *de θ. Se $\mathcal{D}(\theta) = \{x_1, \ldots, x_n\}$ e para todo $1 \leq i \leq n$, $x_i\theta \equiv t_i$, θ identifica-se com o conjunto $\{t_1/x_1, \ldots, t_n/x_n\}$. Uma* instância *duma expressão E por θ é a expressão $E\theta$ que se obtém substituindo simultaneamente cada ocorrência livre de x_i em E por t_i, para todo $1 \leq i \leq n$.*

A substituição $\theta = \emptyset$ é a substituição *identidade* e denotamos por ϵ. A composição de duas substituições $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ e $\sigma = \{s_1/y_1, \ldots, s_m/y_m\}$ escreve-se $\theta\sigma$ e define-se por:

$$\begin{aligned} \theta\sigma \quad = \quad & \{t_i\sigma/x_i : 1 \leq i \leq n \text{ e } x_i \not\equiv t_i\sigma\} \quad \cup \\ & \{s_j/y_j : 1 \leq j \leq m \text{ e } y_j \not\equiv x_i \text{ para todo } 1 \leq i \leq n\} \end{aligned}$$

Sendo θ, σ e γ substituições e E uma expressão é fácil verificar que:

(i) $\theta\epsilon = \epsilon\theta = \theta$

(ii) $E(\theta\sigma) = (E\theta)\sigma$

(iii) $(\theta\sigma)\gamma = \theta(\sigma\gamma)$

Uma substituição θ é *idempotente* se $\theta\theta = \theta$. Dada uma expressão E, uma substituição $\theta = \{y_1/x_1, \ldots, y_n/x_n\}$ é uma *renomeação de variáveis* se o conjunto $\{x_1, \ldots, x_n\}$ está contido em $\mathsf{Vars}(E)$, todos y_i são distintos e $(\mathsf{Vars}(E) \setminus \{x_1, \ldots, x_n\}) \cap \{y_1, \ldots, y_n\} = \emptyset$.

No conjunto das substituições define-se a seguinte relação (pré-ordem):

$$\theta \preceq \sigma \quad \text{se e só se} \quad \exists\gamma\ \sigma = \theta\gamma$$

Se $\theta \preceq \sigma$ diz-se que θ é *mais geral* que σ, ou, equivalentemente, que σ é *mais específica* que θ.

A noção de substituição estende-se naturalmente a todo o conjunto de termos, isto é, cada substituição θ tem uma única extensão homomórfica $\hat{\theta} : T \longrightarrow T$, que denotaremos também por θ.

A teoria de demonstração dum sistema axiomático (dedutivo) de $\mathcal{L}$ (p.e. no estilo de Hilbert) é dada como habitualmente por um conjunto de axiomas lógicos, entre eles os axiomas da igualdade, e eventualmente um conjunto de axiomas não lógicos; e por um conjunto de regras de inferência (p.e. *Modus Ponens* e Generalização). Uma *demonstração* (ou derivação) duma fórmula ϕ é uma sequência de fórmulas de $\mathcal{L}$, em que cada uma é um axioma ou pode ser inferida das fórmulas precedentes, por uma das regras de inferência, e ϕ é a última fórmula da sequência. Uma fórmula ϕ é um teorema num sistema axiomático se tiver uma demonstração, e denota-se esse facto por $\vdash \phi$.

Neste trabalho, vamos apenas utilizar alguns teoremas bem conhecidos da lógica de 1ª ordem. Em particular, alguns referentes às propriedades booleanas das conectivas lógicas.

2.1.2 Semântica

A teoria de modelos tem como objectivo associar uma interpretação às expressões de $\mathcal{L}$.

Definição 2.6 *Uma álgebra (ou estrutura*[1]*) $\mathcal{A}$ de uma linguagem $\mathcal{L}$ é definida por:*

(i) *um conjunto não vazio $\mathcal{U}_{\mathcal{A}}$, que se diz o universo de $\mathcal{A}$;*

(ii) *um elemento $a^{\mathcal{A}} \in \mathcal{U}_{\mathcal{A}}$, para cada constante a de $\mathcal{L}$;*

(iii) *uma função de aridade n, $f^{\mathcal{A}} : \mathcal{U}_{\mathcal{A}}^n \longrightarrow \mathcal{U}_{\mathcal{A}}$ para cada functor f^n de $\mathcal{L}$;*

(iv) *o símbolo de predicado $=$ é interpretado como a identidade $=_{\mathcal{U}_{\mathcal{A}}}$ em $\mathcal{U}_{\mathcal{A}}$.*

Uma vez que $\mathcal{L}$ é caracterizada pela assinatura Σ, designam-se estas álgebras por Σ-álgebras. Vamos apenas considerar Σ-álgebras em que Σ é não singular, isto é, com pelo menos dois elementos.

Definição 2.7 *Dada uma álgebra $\mathcal{A}$ de $\mathcal{L}$, uma atribuição de (valores às) variáveis ρ é uma função de X em $\mathcal{U}_{\mathcal{A}}$. Dada uma atribuição de variáveis ρ, uma variável $x \in X$ e um elemento $\alpha \in \mathcal{U}_{\mathcal{A}}$ define-se $\rho[\alpha/x]$ como uma função que coincide com ρ em todas as variáveis diferentes de x e tal que $\rho[\alpha/x](x) = \alpha$. Dado $V \subseteq X$ e uma atribuição de variáveis ρ, denotamos por $\rho|_V$ a restrição de ρ a V.*

Definição 2.8 *Dada uma álgebra $\mathcal{A}$ de $\mathcal{L}$ e uma atribuição de variáveis ρ, estende-se ρ ao conjunto dos termos de $\mathcal{L}$, definindo recursivamente a função $\hat{\rho}$:*

(i) $\hat{\rho}(x) = \rho(x)$ *para toda a variável x;*

(ii) $\hat{\rho}(a) = a^{\mathcal{A}}$ *para toda a constante a;*

[1] *Preferimos a designação de álgebra, pelos demasiados significados da palavra "estrutura" em Ciência de Computadores e porque é a designação habitual na literatura para este tipo de interpretações.*

(iii) $\hat{\rho}(f(t_1,\ldots,t_n)) = f^{\mathcal{A}}(\hat{\rho}(t_1),\ldots,\hat{\rho}(t_n))$, *se* $t_1,\ldots,t_n$ *são termos e* f^n *um símbolo de função de aridade* n.

Habitualmente identificaremos a função $\hat{\rho}$ com ρ.

O valor (ou interpretação) das fórmulas de $\mathcal{L}$ em $\mathcal{A}$ é dado pela relação de satisfação a seguir definida. Por uma questão de comodidade apresentamos a definição desta relação para algumas das abreviaturas de fórmulas de $\mathcal{L}$, atrás referidas.

Definição 2.9 *Seja $\mathcal{A}$ uma álgebra de $\mathcal{L}$ e seja ρ uma atribuição de variáveis. O facto de $\mathcal{A}$* satisfazer uma fórmula ϕ para a atribuição de variáveis ρ *denota-se por $\mathcal{A} \models_\rho \phi$ e define-se por indução na estrutura de ϕ do seguinte modo:*

(i) $\mathcal{A} \models_\rho \top$;

(ii) $\mathcal{A} \not\models_\rho \bot$, *isto é,* $\mathcal{A} \models_\rho \bot$ *não se verifica;*

(iii) $\mathcal{A} \models_\rho t_1 = t_2$ *se* $\rho(t_1) = \rho(t_2)$;

(iv) $\mathcal{A} \models_\rho \phi \wedge \psi$ *se* $\mathcal{A} \models_\rho \phi$ e $\mathcal{A} \models_\rho \psi$;

(v) $\mathcal{A} \models_\rho \phi \vee \psi$ *se* $\mathcal{A} \models_\rho \phi$ *ou* $\mathcal{A} \models_\rho \psi$;

(vi) $\mathcal{A} \models_\rho \neg\phi$ *se* $\mathcal{A} \not\models_\rho \phi$, *isto é, se* $\mathcal{A} \models_\rho \phi$ *não se verifica;*

(vii) $\mathcal{A} \models_\rho \phi \Rightarrow \psi$ *se* $\mathcal{A} \not\models_\rho \phi$ *ou* $\mathcal{A} \models_\rho \psi$;

(viii) $\mathcal{A} \models_\rho \forall x \phi$ *se para todo* $\alpha \in \mathcal{U}_{\mathcal{A}}$ *se tem* $\mathcal{A} \models_{\rho[\alpha/x]} \phi$;

(ix) $\mathcal{A} \models_\rho \exists x \phi$ *se existe um* $\alpha \in \mathcal{U}_{\mathcal{A}}$ *tal que* $\mathcal{A} \models_{\rho[\alpha/x]} \phi$.

Uma fórmula ϕ é *satisfazível em* $\mathcal{A}$ se existe uma atribuição de variáveis ρ, tal que $\mathcal{A} \models_\rho \phi$. Ao conjunto de todas essas atribuições de variáveis designamos por conjunto de $\mathcal{A}$-soluções de ϕ e denotamos por $\mathsf{Sol}(\phi, \mathcal{A})$. Uma fórmula ϕ é *satisfazível* se existe uma álgebra $\mathcal{A}$ em que ϕ é satisfazível. Uma fórmula ϕ é *válida em* $\mathcal{A}$ se para toda a atribuição de variáveis ρ se tem $\mathcal{A} \models_\rho \phi$. Neste caso, $\mathcal{A}$ é um *modelo* de ϕ, e escrevemos $\mathcal{A} \models \phi$. Uma fórmula ϕ é *válida* se for válida em todas as álgebras e escrevemos $\models \phi$. Uma fórmula ϕ é uma *contradição* se não existe nenhuma álgebra $\mathcal{A}$ e nenhuma atribuição de variáveis ρ tal que $\mathcal{A} \models_\rho \phi$. Neste caso, escrevemos $\not\models \phi$.

Proposição 2.10 *Seja ϕ uma fórmula de $\mathcal{L}$ e $\mathcal{A}$ uma álgebra de $\mathcal{L}$.*

(i) ϕ é válida (em $\mathcal{A}$) se e só se $\neg\phi$ é não satisfazível (em $\mathcal{A}$).

(ii) ϕ é satisfazível (em $\mathcal{A}$) se e só se $\neg\phi$ é não válida (em $\mathcal{A}$).

(iii) ϕ *é válida (em* $\mathcal{A}$*) se e só se* $\forall\phi$ *é válida (em* $\mathcal{A}$*).*

(iv) ϕ *é satisfazível (em* $\mathcal{A}$*) se e só se* $\exists\phi$ *é satisfazível (em* $\mathcal{A}$*).*

Definição 2.11 *Para qualquer álgebra* $\mathcal{A}$*, duas fórmulas* ϕ *e* ψ *são* independentes para a satisfazibilidade em $\mathcal{A}$, *se a satisfazibilidade em* $\mathcal{A}$ *de* ϕ *e* ψ *implica a satisfazibilidade em* $\mathcal{A}$ *de* $\phi \wedge \psi$.

Definição 2.12 *Sendo* θ *uma substituição e* ρ *uma atribuição de variáveis, numa álgebra* $\mathcal{A}$ *de* $\mathcal{L}$*, define-se* $\rho\theta$ *a atribuição de variáveis tal que* $\rho\theta(x) = \rho(x\theta)$ *para todo* $x \in X$*. Esta noção estende-se naturalmente a todo o conjunto dos termos.*

Proposição 2.13 *Se* θ *é uma substituição e* ϕ *uma fórmula de* $\mathcal{L}$ *tal que para cada variável livre* x *de* ϕ*, o termo* $x\theta$ *não contém nenhuma variável que após substituição fique ligada a um quantificador de* ϕ*, então para toda a atribuição de variáveis* ρ*,*

$$\mathcal{A} \models_\rho \phi\theta \text{ se e só se } \mathcal{A} \models_{\rho\theta} \phi$$

É óbvio que o facto de uma fórmula fechada ϕ ser satisfazível numa álgebra $\mathcal{A}$ não depende das atribuições de variáveis. Dado um conjunto de fórmulas fechadas S e uma álgebra $\mathcal{A}$, diz-se que $\mathcal{A}$ é *um modelo para* S, e escreve-se $\mathcal{A} \models S$, se $\mathcal{A}$ for um modelo para cada fórmula de S. Se todo o modelo de S for um modelo duma fórmula fechada ϕ, diz-se que ϕ é uma *consequência (lógica)* de S, e escreve-se $S \models \phi$. Este conceito generaliza-se para qualquer fórmula ϕ considerando o seu fecho universal, $\forall\phi$. Pela completude e integridade da lógica de 1ª ordem uma fórmula ϕ é uma *consequência lógica* de S se e só se $S \vdash \phi$, isto é, se ϕ é um teorema num sistema axiomático onde os elementos de S são axiomas não-lógicos.

Definição 2.14 *Seja* $\mathcal{A}$ *uma álgebra e* ϕ *e* ψ *duas fórmulas de* $\mathcal{L}$*. A fórmula* ϕ vincula ψ *em* $\mathcal{A}$*, e escreve-se* $\phi \models_\mathcal{A} \psi$*, se* $\forall(\phi \Rightarrow \psi)$ *é válida em* $\mathcal{A}$*, ou em termos de conjuntos de soluções, se* $\mathsf{Sol}(\phi,\mathcal{A}) \subseteq \mathsf{Sol}(\psi,\mathcal{A})$*. As fórmulas* ϕ *e* ψ *dizem-se* (logicamente) equivalentes *em* $\mathcal{A}$*, e escreve-se* $\phi \approx_\mathcal{A} \psi$*, se* $\forall(\phi \Leftrightarrow \psi)$ *é válida em* $\mathcal{A}$*, ou em termos de conjuntos de soluções se* $\mathsf{Sol}(\phi,\mathcal{A}) = \mathsf{Sol}(\psi,\mathcal{A})$*. Dado um conjunto fechado de fórmulas* S*, se* $S \models \phi \Rightarrow \psi$ *então diz-se que* ϕ vincula *(ou* implica*)* ψ *em* S *e escreve-se* $\phi \models_S \psi$*. Analogamente, diz-se que duas fórmulas* ϕ*,* ψ *são equivalentes em* S*,* $\phi \approx_S \psi$*.*

Proposição 2.15 *Para qualquer álgebra* $\mathcal{A}$ *e fórmulas* ϕ*,* ψ *e* ψ' *de* $\mathcal{L}$ *tem-se que:*

(i) se ϕ *é satisfazível em* $\mathcal{A}$ *então não se pode ter simultaneamente* $\phi \models_\mathcal{A} \psi$ *e* $\phi \models_\mathcal{A} \neg\psi$*;*

(ii) $\mathcal{A} \models \phi \wedge \psi \Leftrightarrow \phi \wedge \psi'$ *se e só se* $\phi \models_\mathcal{A} \psi \Leftrightarrow \psi'$*;*

(iii) $\mathcal{A} \models \phi \wedge \psi \Leftrightarrow \phi$ *se e só se* $\phi \models_{\mathcal{A}} \psi$*;*

(iv) $\mathcal{A} \not\models \phi \wedge \psi$ *se e só se* $\phi \models_{\mathcal{A}} \neg\psi$*.*

Se ϕ for satisfazível em $\mathcal{A}$ e $\phi \models_{\mathcal{A}} \neg\psi$ ou $\phi \models_{\mathcal{A}} \psi$ diz-se que ϕ *determina* ψ em $\mathcal{A}$ ([AKPS92, AKP93b, JM94]).

2.1.3 Teorias e Sistemas de Reescrita

A *teoria* (de 1ª ordem) dum conjunto de fórmulas fechadas S é o conjunto $\mathrm{Th}(S)$ de todas as fórmulas fechadas ϕ tal que $S \models \phi$. O conjunto $\mathrm{Th}(S)$ é *decidível* (recursivo) se existe um algoritmo que dada uma fórmula ϕ permite decidir se $\phi \in \mathrm{Th}(S)$. O conjunto S diz-se *completo* se para toda a fórmula fechada ϕ ou se verifica $S \models \phi$ ou $S \models \neg\phi$. Um conjunto S é semi-decidível (recursivamente enumerável) se existe um algoritmo que termina se $\phi \in S$. Se S é recursivamente enumerável então $\mathrm{Th}(S)$ é recursivamente enumerável. Se S é recursivamente enumerável e completo então $\mathrm{Th}(S)$ é decidível. A teoria $\mathrm{Th}(\mathcal{A})$ duma álgebra $\mathcal{A}$ é o conjunto de todas as fórmulas fechadas ϕ, tal que $\mathcal{A} \models \phi$. Uma *axiomatização* duma álgebra $\mathcal{A}$ é um conjunto recursivo de fórmulas fechadas S, usualmente designadas por *axiomas*, tal que $\mathrm{Th}(S)=\mathrm{Th}(\mathcal{A})$. Então, se uma álgebra é *completamente axiomatizável* a sua teoria é decidível.

Duas álgebras $\mathcal{A}$ e $\mathcal{B}$ são *elementarmente equivalentes* se para cada fórmula fechada ϕ,

$$\mathcal{A} \models \phi \text{ se e só se } \mathcal{B} \models \phi$$

O lema seguinte é apresentado em [Sho67].

Lema 2.16 *Seja S um conjunto fechado de fórmulas que admite pelo menos um modelo. São equivalentes as seguintes afirmações:*

(i) S é completo.

(ii) Quaisquer dois modelos de S são elementarmente equivalentes.

(iii) Para qualquer modelo $\mathcal{A}$ de S, tem-se que $Th(S)=Th(\mathcal{A})$.

Por abuso de linguagem, designamos um conjunto de axiomas S por *teoria*. Associados a uma teoria será possível deduzir conjuntos de *regras de reescrita* que serão usados para demonstrar resultados sobre essa teoria.

Sistemas de reescrita sobre termos são usados extensivamente para resolver problemas que envolvem igualdades entre termos. Para um estudo detalhado, aconselhamos [DJ90].

De modo análogo, definem-se sistemas de reescrita em $\mathcal{L}$. Em particular, as fórmulas de $\mathcal{L}$ podem ser vistas como termos numa álgebra.

Definição 2.17 *Uma regra de reescrita em $\mathcal{L}$ é um par ordenado de fórmulas, denotado por $\phi_1 \mapsto \phi_2$. Uma fórmula ϕ* reescreve-se *numa fórmula ψ por uma regra de reescrita R, e diz-se que R* aplica-se *a ϕ, se uma das seguintes condições se verifica:*

(i) $\phi \mapsto \psi$ *é uma instância de R;*

(ii) existe uma sub-fórmula ϕ' de ϕ, tal que $\phi' \mapsto \psi'$ é uma instância de R e ψ é ϕ onde uma ocorrência de ϕ' é substituída por ψ'.

Uma fórmula ϕ reescreve-se *numa fórmula ψ, se ϕ se reescreve em ψ pela aplicação finita de regras de reescrita, isto é, se $\phi \mapsto^\star \psi$ onde $\mapsto^\star$ é o fecho reflexivo e transitivo da relação binária $\mapsto$. Um sistema de reescrita $\mathcal{R}$ é um conjunto de regras de reescrita sobre $\mathcal{L}$. Uma derivação em $\mathcal{R}$ é uma sequência $\phi_1 \mapsto \phi_2 \mapsto \ldots$ de aplicações de regras de reescrita de $\mathcal{R}$. Uma regra R é* correcta *em relação a uma álgebra $\mathcal{A}$ se $\phi_1 \mapsto \phi_2$ implica que $\phi_1 \approx_{\mathcal{A}} \phi_2$. Um sistema de reescrita $\mathcal{R}$ é* correcto *em relação a $\mathcal{A}$ se cada regra de $\mathcal{R}$ o for. Um sistema de reescrita $\mathcal{R}$* termina *se não existir nenhuma derivação infinita $\phi_1 \mapsto \phi_2 \mapsto \ldots$. Uma fórmula ϕ é* irredutível *para $\mathcal{R}$ ou* fechada *para $\mathcal{R}$, se nenhuma regra $R \in \mathcal{R}$ se aplica a ϕ. Dado um conjunto de fórmulas $\mathcal{C}$, um conjunto de fórmulas $\mathcal{FR}$, e uma álgebra $\mathcal{A}$, um conjunto de regras $\mathcal{R}$ é* completo *em relação a $\mathcal{FR}$* e *$\mathcal{C}$ se $\mathcal{R}$ é correcto em relação a $\mathcal{A}$* e *para toda a fórmula $\phi \in \mathcal{C}$ existe $\psi \in \mathcal{FR}$ tal que $\phi \mapsto^\star \psi$* e *ψ é fechada para $\mathcal{R}$.*[2]

A demonstração da correcção duma regra de reescrita R é simplificada notando-se que se $\mathsf{Sol}(\phi_1, \mathcal{A}) = \mathsf{Sol}(\phi_2, \mathcal{A})$ então, para qualquer $\psi \in \mathcal{L}$ tem-se que:

$$\begin{array}{lcl}
\mathsf{Sol}(\phi_1 \ \wedge \ \psi, \mathcal{A}) & = & \mathsf{Sol}(\phi_2 \ \wedge \ \psi, \mathcal{A}) \\
\mathsf{Sol}(\phi_1 \ \vee \ \psi, \mathcal{A}) & = & \mathsf{Sol}(\phi_2 \ \vee \ \psi, \mathcal{A}) \\
\mathsf{Sol}(\exists \bar{z}.\, \phi_1, \mathcal{A}) & = & \mathsf{Sol}(\exists \bar{z}.\, \phi_2, \mathcal{A}) \\
\mathsf{Sol}(\neg \phi_1, \mathcal{A}) & = & \mathsf{Sol}(\neg \phi_2, \mathcal{A})
\end{array}$$

Para a demonstração da terminação dos sistemas de reescrita serão ainda usadas as seguintes noções sobre relações de ordem e multiconjuntos [DM79, DJ90].

Definição 2.18 *Dado um conjunto parcialmente ordenado $(\mathcal{S}, \succeq)$, um* multiconjunto *(finito) sobre $\mathcal{S}$ é uma aplicação $\mu : \mathcal{S} \longrightarrow \mathbb{N}$ tal que o conjunto $\{x \in \mathcal{S} : \mu(x) \neq 0\}$ é finito. O*

[2] *Estas noções são usadas em [Com91b]* e *[Com88] para obter algoritmos de decidibilidade de teorias.*

multiconjunto μ pode ser denotado por um conjunto em que o elemento x ocorre n vezes, se $\mu(x) = n$. O conjunto de todos os multiconjuntos sobre $\mathcal{S}$ denota-se por $\mathsf{Mul}(\mathcal{S})$. Dois multiconjuntos (sobre $\mathcal{S}$) μ e μ' são iguais *se para todo $x \in \mathcal{S}$ se tem $\mu(x) = \mu'(x)$. A* reunião *de dois multiconjuntos μ e μ' é o multiconjunto $\mu \cup \mu'$ definido para todo $x \in \mathcal{S}$ por $(\mu \cup \mu')(x) = \mu(x) + \mu'(x)$. A ordem parcial $\succeq$ em $\mathcal{S}$ induz uma ordem parcial $\ggcurly$ em $\mathsf{Mul}(\mathcal{S})$, tal que $\mu_1 \ggcurly \mu_2$, se uma das seguintes condições se verifica:*

(i) $\mu_1 = \mu_2$;

(ii) existem multiconjuntos ζ e ω sobre $\mathcal{S}$ com $\emptyset \neq \zeta \subseteq \mu_1$ tal que:

- *(a)* $\mu_2 = (\mu_1 \setminus \zeta) \cup \omega$;
- *(b)* $\forall w \in \omega\ \exists z \in \zeta\ z \succ w$;

Por exemplo, para $(\mathbb{N}, \geq)$, temos $\{3,3,4,5\} \gg \{3,4\}$ e $\{3,3,4,5\} \gg \{2,3,3,3,3,0\}$.

Definição 2.19 *Dado um conjunto parcialmente ordenado $(\mathcal{S}, \succeq)$, pode-se estender $\succeq$ ao conjunto $\mathcal{S}^n$ de n-uplos de elementos de $\mathcal{S}$, considerando a ordem* lexicográfica, *$\succeq_n$ que se define por: $(s_1, \ldots, s_n) \succeq_n (s'_1, \ldots, s'_n)$ se*

$$\forall\, 1 \leq k \leq n,\ s_k = s'_k \quad \textit{ou} \quad \exists\, 1 \leq i \leq n,\ \forall\, 1 \leq k \leq i-1,\ s_k = s'_k \text{ e } s_i \succ s'_i$$

Definição 2.20 *Um conjunto parcialmente ordenado $(\mathcal{S}, \succeq)$ é* bem-fundado *se não existe nenhuma sequência estritamente decrescente $s_1 \succ s_2 \succ \ldots$ de elementos de $\mathcal{S}$.*

Por exemplo, o conjunto parcialmente ordenado $(\mathbb{N}, \geq)$ é bem-fundado. O conjunto $(\mathcal{S}, \succeq)$ é bem-fundado se e só se a extensão $\ggcurly$ a multiconjuntos sobre $\mathcal{S}$ é bem-fundado [DM79]. Analogamente, se o conjunto $(\mathcal{S}, \succeq)$ é bem-fundado, o conjunto $(\mathcal{S}^n, \succeq_n)$ também é bem-fundado. O teorema seguinte é habitualmente usado para provar a terminação de sistemas de reescrita [MN70]:

Teorema 2.21 *Um sistema de reescrita $\mathcal{R}$ sobre um conjunto de expressões $\mathcal{E}$ termina se e só se existe um conjunto bem-fundado $(\mathcal{W}, \succeq)$ e uma função $\tau : \mathcal{E} \longrightarrow \mathcal{W}$ tal que:*

$$\forall e, e' \in \mathcal{E},\ e \mapsto e' \quad \textit{implica que} \quad \tau(e) \succ \tau(e')$$

Métodos para provar a terminação de sistemas de reescrita e propriedades das relações de ordem envolvidas, são estudados extensivamente em [Der82] e [DJ90].

2.1.4 Árvores Finitas, Racionais e Infinitas

Nesta secção introduzimos as álgebras de $\mathcal{L}$ que iremos considerar neste trabalho.

Seja $\mathbb{N}^\star$ o conjunto (monóide) das sequências finitas de inteiros positivos munido com a operação concatenação e tendo, como elemento neutro a sequência vazia, denotada por ϵ. Os seus elementos designam-se por *posições* ou *nós*. Se α é uma posição então $|\alpha|$ denota o seu comprimento. Um conjunto $D \subseteq \mathbb{N}^\star$ é *fechado para prefixos* se $\alpha\beta \in D$ implica que $\alpha \in D$. Seja F um conjunto dos símbolos de função e D um subconjunto não vazio de $\mathbb{N}^\star$, fechado para prefixos. Uma árvore sobre F é uma função $t : D \longrightarrow F$ tal que se f tem aridade n, então $t(\alpha) = f$ implica $\alpha i \in D$ se e só se $1 \leq i \leq n$. A posição ϵ é a *raiz* duma árvore. Uma sub-árvore duma árvore t na posição $\alpha \in D$ é a árvore $t|_\alpha$: $D' \longrightarrow F$ tal que $D' = \{\beta \in \mathbb{N}^\star : \alpha\beta \in D\}$ e $t|_\alpha\ (\beta) = t(\alpha\beta)$. Se u é uma árvore, $t[u]_\alpha$ é a árvore que se obtém substituindo $t|_\alpha$ por u em t. Uma árvore racional é uma árvore com um número finito de sub-árvores. Uma árvore finita é uma árvore cujo domínio é finito. O *tamanho* duma árvore t, designa-se por $|t|$ e define-se como o cardinal do domínio de t. Na Figura 2.1 são apresentados alguns exemplos de árvores sobre um conjunto de símbolos funcionais.

Os termos podem ser considerados árvores finitas sobre $F = \Sigma \cup X$, onde os elementos de X são considerados símbolos de aridade 0. Uma árvore racional sobre $F = \Sigma \cup X$, denomina-se um *termo racional*. Na Figura 2.1, a árvore em a) é finita e corresponde ao termo $f(g(a), b)$; as árvores em b) são racionais e correspondem, respectivamente, aos termos $f(g(a), g(f(g(a), \ldots)))$ e $h(h(\ldots))$. A árvore em c) não é racional, porque não tem um número de sub-árvores finito.

As *álgebras de árvores sobre* F são definidas por:

(i) o universo é conjunto das árvores sobre F;

(ii) a árvore $t(\epsilon) = a$ para cada constante a de F;

(iii) para cada $f^n \in F$, define-se uma função de aridade n que associa a n árvores $t_1, \ldots, t_n$ uma árvore t tal que $t(\epsilon) = f$ e $t(k\beta) = t_k(\beta)$ para todo $1 \leq k \leq n$.

Os conjuntos das árvores infinitas, racionais e finitas são fechados para toda a função associada a $f^n \in \Sigma$. As álgebras correspondentes denotam-se por $\mathcal{IT}(F)$, $\mathcal{RT}(F)$ e $\mathcal{T}(F)$, respectivamente, e F será omitida sempre que coincidir com Σ. A $\mathcal{T}$ chama-se habitualmente o *Universo de Herbrand*.

M. Maher [Mah88] provou que as álgebras $\mathcal{T}$, $\mathcal{IT}$, e $\mathcal{RT}$ são completamente axiomatizáveis, e portanto, que as suas teorias são decidíveis.

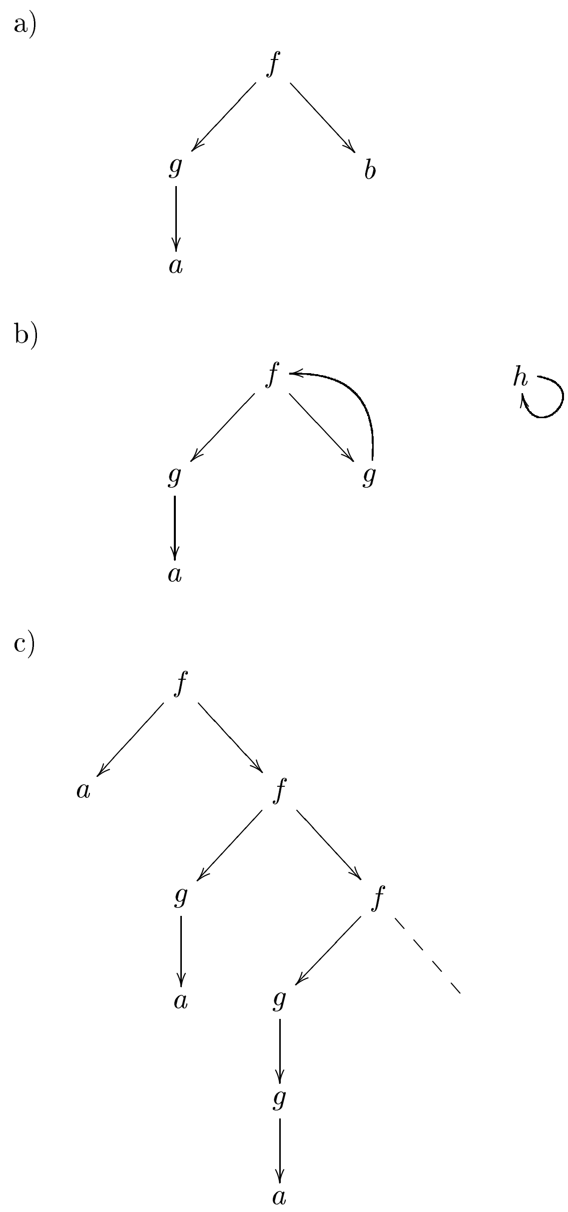


Figura 2.1: Exemplos de árvores sobre $\{f^2, g^1, h^1, a, b\}$.

Para cada uma das álgebras o problema é sub-dividido em dois casos consoante a assinatura Σ é finita ou infinita. Mal'cev [Mal71] ao estudar axiomatizações completas para classes de álgebras localmente livres, tinha já proposto uma axiomatização do Universo de Herbrand que coincide com o que é hoje conhecido, na comunidade de Programação em Lógica, por teoria equacional de Clark [Cla78] e provado a sua decidibilidade. K. Kunen [Kun87] prova que a teoria equacional de Clark é completa para o caso de Σ ser infinita. H. Comon [CL89, Com88] obtém os mesmos resultados para o caso de Σ ser finita. Independentemente e com base nos sistemas de reescrita apresentados em [DMV91][3] e na eliminação de quantificadores, foi descrito em [DM91] um algoritmo para a decidibilidade de fórmulas do Universo de Herbrand para Σ infinita. Este algoritmo pode ser modificado para os restantes casos, como se apresenta no Apêndice A.

O esquema básico das demonstrações de decidibilidade da teoria duma álgebra $\mathcal{A}$ é o seguinte:

- estabelecer um conjunto (recursivo) de axiomas T tal que $\text{Th}(T)=\text{Th}(\mathcal{A})$.
- definir uma classe de fórmulas $\mathcal{FR}$ para as quais é trivial decidir se pertencem a $\text{Th}(T)$; as fórmulas de $\mathcal{FR}$ denominam-se *formas resolvidas.*
- determinar um conjunto de regras de reescrita, definidas a partir dos axiomas, que permitem transformar qualquer fórmula ϕ numa disjunção ψ de fórmulas $\phi' \in \mathcal{FR}$, e tal que $\phi \approx_T \psi$.

Procede-se por eliminação de quantificadores. Seja ϕ uma fórmula de $\mathcal{L}$. Qualquer que seja a álgebra $\mathcal{A}$, ϕ é equivalente a uma fórmula em forma prenexa normal, $Q_1 x_1 \ldots Q_n x_n.\gamma$, onde γ não contém quantificadores. A fórmula γ pode ser transformada (por exemplo) em forma normal disjuntiva onde as fórmulas de $\mathcal{FR}$ são consideradas fórmulas atómicas, isto é, $\gamma \approx_T \gamma_1 \ \vee \ \ldots \ \vee \ \gamma_m$. Se $Q_n \equiv \exists$ então $\exists x_n.\gamma \approx_T \exists x_n.\gamma_1 \ \vee \ \ldots \ \vee \ \exists x_n.\gamma_m$ e prova-se que para cada i, $\exists x_n.\gamma_i \approx_T \phi_i$ onde ϕ_i é uma fórmula de $\mathcal{FR}$. Se $Q_n \equiv \forall$ basta considerar $\exists x_n.\neg\gamma$ e no fim negar a fórmula obtida.

Na Tabela 2.1 apresentamos o conjunto de axiomas $E_{\mathcal{T}}$ que constituem uma axiomatização completa de $\mathcal{T}$ (Universo de Herbrand) para Σ infinita.

Os axiomas da igualdade podiam ser omitidos porque estamos a considerar álgebras $\mathcal{A}$ onde o símbolo '=' é interpretado como a identidade em $\mathcal{U}_{\mathcal{A}}$. No entanto, estes axiomas são apresentados porque serão usados em regras de reescrita.

[3]E que iremos detalhar e generalizar nos Capítulo 4 e 6.

A1. Para todo o termo t, $t = t$

A2. Para todos os termos t e s, $t = s \Rightarrow s = t$

A3. Para todos os termos t, s e u, $(t = s \wedge s = u) \Rightarrow t = u$

A4. Para todo $f^n \in \Sigma$, $\forall \bar{x} \forall \bar{y}.\ (f(\bar{x}) = f(\bar{y}) \Leftrightarrow x_1 = y_1 \wedge \ldots \wedge x_n = y_n)$

A5. Para todo $f^n, g^k \in \Sigma$, $f^n \not\equiv g^k$, $\forall \bar{x} \forall \bar{y}.\ f(\bar{x}) \neq g(\bar{y})$

A6. Para todo o termo t, $\forall x\ x \neq t[x]$, se $t[x]$ não coincide com x

Tabela 2.1: Axiomas das árvores finitas, $E_{\mathcal{T}}$

No caso de Σ ser finita, $E_{\mathcal{T}}$ não é completo. Por exemplo, se $\Sigma = \{a, b\}$, a fórmula

$$x \neq a \wedge x \neq b$$

não é satisfazível em $\mathcal{T}(\Sigma)$ mas não é verdade que

$$E_{\mathcal{T}} \models \forall x\, (x = a \vee x = b)$$

Para obter uma axiomatização completa quando a assinatura Σ é finita, é necessário mais um axioma, denominado *Axioma do Domínio Fechado* (DCA):

$$\forall x \bigvee_{f^n \in \Sigma} \exists z_1, \ldots, z_n.\ x = f(z_1, \ldots, z_n)$$

onde se $n = 0$, f denota uma constante.

O conjunto $E_{\mathcal{T}} \cup \{\mathsf{DCA}\}$ axiomatiza completamente o Universo de Herbrand para Σ finita, com pelo menos dois elementos[4].

Na Tabela 2.2 apresentamos o conjunto de axiomas $E_{\mathcal{RT}}$ que constituem uma axiomatização completa de $\mathcal{RT}$ para Σ infinita.

No Axioma A6 da Tabela 2.2, a expressão $\exists! x\, \phi$ é uma abreviatura de

$$\exists x \phi \wedge \forall x \forall y (\phi \wedge \phi[y/x] \Rightarrow x = y)$$

O quantificador $\exists!$ satisfaz a seguinte propriedade para quaisquer fórmulas ϕ e ψ de $\mathcal{L}$ e álgebra $\mathcal{A}$

$$\forall \exists! \bar{x} \phi \wedge \forall \exists \bar{x} (\phi \wedge \psi) \models_{\mathcal{A}} \forall (\phi \Rightarrow \psi) \tag{2.1}$$

[4] Se Σ contém apenas uma constante ou um functor unário o resultado também é válido, mas o problema está em aberto para os restantes casos [Mah88].

O Axioma A6 traduz a unicidade de solução das formas resolvidas racionais, cuja definição é dada na secção seguinte.

Se Σ for finita com pelo menos dois elementos, o conjunto $E_{\mathcal{RT}} \cup \{\mathsf{DCA}\}$ é uma axiomatização completa de $\mathcal{RT}$.

O facto de $\mathcal{IT}$ ser também um modelo de $E_{\mathcal{RT}}$ (ou, respectivamente, de $E_{\mathcal{RT}} \cup \{\mathsf{DCA}\}$, quando $2 \leq |\Sigma| < \infty$) e as axiomatizações serem completas permite concluir que $\mathcal{RT}$ e $\mathcal{IT}$ são elementarmente equivalentes.

A1. Para todo o termo t, $t = t$

A2. Para todos os termos t e s, $t = s \;\Rightarrow\; s = t$

A3. Para todos os termos t, s e u, $(t = s \;\wedge\; s = u) \;\Rightarrow\; t = u$

A4. Para todo $f^n \in \Sigma$, $\forall\bar{x}\forall\bar{y}.\ (f(\bar{x}) = f(\bar{y}) \;\Leftrightarrow\; x_1 = y_1 \;\wedge\; \ldots \;\wedge\; x_n = y_n)$

A5. Para todo $f^n, g^k \in \Sigma$, $f^n \not\equiv g^k$, $\forall\bar{x}\forall\bar{y}.\ f(\bar{x}) \neq g(\bar{y})$

A6. Para toda a forma resolvida racional $\bar{x} = \bar{t}[\bar{x}]$, $\forall\exists!\bar{x}.\bar{x} = \bar{t}[\bar{x}]$

Tabela 2.2: Axiomas das árvores racionais, $E_{\mathcal{RT}}$

Iremos identificar $E_{\mathcal{T}}$ (ou $E_{\mathcal{T}} \cup \{\mathsf{DCA}\}$) com $\mathcal{T}$ e $E_{\mathcal{RT}}$ (ou $E_{\mathcal{RT}} \cup \{\mathsf{DCA}\}$) com $\mathcal{RT}$, e com as teorias correspondentes.

2.1.5 Fórmulas Resolvidas para a Unificação

Vamos recordar um subconjunto de $\mathcal{L}$, vastamente estudado na literatura, e que possui uma classe de fórmulas para a qual o problema da satisfazibilidade, em $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$, é trivialmente decidido.

Um *problema de unificação* numa álgebra $\mathcal{A}$ consiste em resolver uma conjunção e de equações $s_1 = t_1 \;\wedge\; \ldots \;\wedge\; s_n = t_n$, abreviando $\bar{s} = \bar{t}$, *i.e*, determinar se $\mathcal{A} \models \exists e$.

Se $\mathcal{A} = \mathcal{T}$, uma *forma resolvida* para a unificação é ou $\top$ ou uma conjunção de equações

$$x_1 = u_1 \;\wedge\; \ldots \;\wedge\; x_n = u_n$$

onde $x_1, \ldots, x_n$ são variáveis distintas e que não ocorrem em nenhum u_j, para $1 \leq j \leq n$. Uma forma resolvida é satisfazível em $\mathcal{T}$. Quaisquer que sejam os valores atribuídos por ρ

às variáveis que ocorrem em u_i, fazendo $\rho(x_i) = \rho(u_i)$, para todo $1 \leq i \leq n$, obtém-se uma $\mathcal{T}$-solução. E vice-versa. Uma forma resolvida define uma substituição idempotente, chamada *unificador mais geral* de e:

$$\theta = \{u_1/x_1, \ldots, u_n/x_n\}$$

tal que se $\bar{s}\theta \equiv \bar{t}\theta$ e $\bar{s}\sigma \equiv \bar{t}\sigma$ (unificador), então θ é mais geral que σ. Qualquer conjunção de equações e pode ser transformada numa forma resolvida ϕ, tal que $e \approx_{\mathcal{T}} \phi$, se for satisfazível, ou em $\perp$, caso contrário [Her30, LMM87]. A fórmula ϕ é única a menos duma relação de equivalência definida entre variáveis $x_i, y_i \in \mathsf{Vars}(e)$ tais que $x_i = y_i$ pertence a ϕ. Um estudo extensivo sobre a unificação no Universo de Herbrand é apresentado em [LMM87].

Se $\mathcal{A} = \mathcal{RT}$, dado que uma equação do tipo $x = t[x]$ tem soluções, uma *forma resolvida racional* é ou $\top$ ou uma conjunção de equações

$$x_1 = t_1 \ \wedge \ \ldots \ \wedge \ x_n = t_n$$

onde $x_1, \ldots, x_n$ são variáveis distintas e se para algum $1 \leq i \leq n$, t_i é uma variável então x_i ocorre uma só vez na fórmula [Com91b, Mah88]. A última condição garante a não existência de ciclos de variáveis da forma

$$x_1 = x_2 \ \wedge \ x_2 = x_3 \ \wedge \ \ldots \ \wedge \ x_{m-1} = x_m \ \wedge \ x_m = x_1$$

Cada forma resolvida define um único unificador mais geral

$$\theta = \{r_1/x_1, \ldots, r_n/x_n\}$$

onde $(r_1, \ldots, r_n)$ é um n-tuplo de árvores racionais sobre $\mathcal{RT}(\Sigma \ \cup \ V)$ onde $V = \mathsf{Vars}(e) \setminus \{x_1, \ldots, x_n\}$, e cada $\mathcal{RT}$-solução ρ satisfaz $\rho(x_i) = \rho(r_i)$[5], isto é, existe uma atribuição de variáveis ρ' tal que $\rho = \rho'\theta$. Em particular, se $t_i \in \mathcal{T}(\Sigma \ \cup \ \{x_1, \ldots, x_n\})$ e $x_i \not\equiv t_i$, para todo $1 \leq i \leq n$, então $x_1 = t_1 \ \wedge \ \ldots \ \wedge \ x_n = t_n$, ou abreviando, $\bar{x} = \bar{t}$ tem uma única solução em $\mathcal{RT}(\Sigma)$. E, inversamente, toda a árvore racional pode ser solução dum sistema de equações dessa forma [Cou83].

Como no caso das árvores finitas, existe um algoritmo que transforma qualquer conjunção de equações e numa forma resolvida ϕ, tal que $e \approx_{\mathcal{RT}} \phi$, se e for satisfazível, ou em $\perp$, caso contrário [Hue76].

Em implementações de linguagens de programação em lógica a unificação de termos é feita sem a regra de verificação de ocorrências (Axioma $\mathsf{A6}$ de $E_{\mathcal{T}}$) o que levou à consideração

[5] Onde $\rho(r_i)$ corresponde à árvore que se obtém aplicando ρ às variáveis de r_i.

árvores racionais, cuja semântica foi estudada por A. Colmerauer [Col94] para sistemas de equações e desigualdades, *i.e*, negações de equações.

Quando se introduzem desigualdades e, no caso do universo da álgebra $\mathcal{A}$ ser infinito, é especialmente relevante a propriedade da *Independência das Desigualdades*, que é caracterizada pelo seguinte lema:

Lema 2.22 *Seja $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$ e $|\mathcal{U}_{\mathcal{A}}| = \infty$. Sendo e uma conjunção de equações e d_i desigualdades, para $1 \leq i \leq n$,*

$$e \ \wedge \ d_1 \ \wedge \ \dots \ \wedge \ d_n$$

tem solução em $\mathcal{A}$ se e só se para cada $1 \leq i \leq n$, a conjunção $e \ \wedge \ d_i$ é satisfazível em $\mathcal{A}$.

Esta propriedade permite testar duma maneira simples a satisfazibilidade dum sistema de equações e desigualdades. O lema foi estudado e generalizado por vários autores [LMM87, Com88, LM90] e em especial por Maher [Mah88], como um lema preparatório para a demonstração da axiomatização completa de $\mathcal{T}$ e $\mathcal{RT}$.

A noção de forma resolvida é ainda usada noutras classes de álgebras para estudar o problema da satisfazibilidade. Um exemplo são as álgebras quociente $\mathcal{T}/_{=_E}$, onde E é um conjunto finito de equações [Com91b].

2.1.6 Formas Resolvidas para $\mathcal{T}$ e $\mathcal{RT}$

A partir das formas resolvidas para a unificação podemos obter as classes de fórmulas $\mathcal{FR}$ usadas para estabelecer a decidibilidade de $\mathcal{T}$ e $\mathcal{RT}$. Vamos referir as usadas por Maher em [Mah88]. Uma *forma básica* é ou $\perp$ ou a fórmula obtida quantificando existencialmente as variáveis que ocorrem no lado direito das equações duma forma resolvida para a unificação. Uma *forma resolvida* é uma conjunção de formas básicas e de negações de formas básicas.

No Capítulo 4, iremos definir as *formas primárias* que como se mostra no Apêndice A, também constituem uma classe de fórmulas $\mathcal{FR}$. A razão principal da introdução desta classe de fórmulas é serem mais adequadas para a manipulação computacional.

2.2 Linguagens de Restrições $\mathcal{LC}_0$, $\mathcal{LC}$ e $\mathcal{LU}$

Nesta secção definimos três linguagens apropriadas para o estudo do problema da satisfazibilidade em $\mathcal{L}$, interpretada em $\mathcal{T}$ ou $\mathcal{RT}$. As linguagens são $\mathcal{LC}_0$, $\mathcal{LC}$ e $\mathcal{LU}$. As fórmulas destas linguagens serão denominadas *restrições*. Para $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$, as linguagens $\mathcal{LC}_0$ e $\mathcal{LC}$ são equivalentes e equivalentes a $\mathcal{L}$ no sentido em que para qualquer fórmula ϕ duma das linguagens existe uma fórmula ψ nas outras duas que é equivalente em $\mathcal{A}$ a ϕ. Estas linguagens foram introduzidas em [DMV91, DM91]. A linguagem $\mathcal{LU}$ é um subconjunto da linguagem $\mathcal{LC}$ que é suficiente para o estudo da satisfazibilidade em $\mathcal{L}$ [DMV94]. Como será estudado no Capítulo 3, a linguagem $\mathcal{L}$ interpretada em $\mathcal{RT}$ é, sobre certas condições, equivalente a linguagens usadas nas teorias de árvores de atributos. Em particular, a linguagem $\mathcal{LU}$ é muito semelhante a linguagens usadas para as árvores de atributos.

Para todas estas linguagens, apenas vamos considerar interpretações nas álgebras de árvores e, portanto, sempre que nos referirmos a problemas de satisfazibilidade poderemos omitir a álgebra.

2.2.1 Linguagens $\mathcal{LC}_0$ e $\mathcal{LC}$

As linguagens $\mathcal{LC}_0$ e $\mathcal{LC}$, a seguir apresentadas, diferem da linguagem $\mathcal{L}$ pela introdução de *caminhos* que permitirão trabalhar com fórmulas sem quantificadores, mais apropriadas para manipulação computacional. Informalmente, os *caminhos* permitem a referência a um qualquer sub-termo dum termo sem a necessidade de introduzir expressamente variáveis adicionais quantificadas existencialmente. Conduzem a uma representação compacta e à manipulação de termos com "informação parcial" sobre os seus sub-termos. Independentemente, uma representação análoga foi usada por Kunen em [Kun87].

A noção de caminho é também usada para a representação de árvores. Associamos a uma assinatura Σ, o conjunto $\Pi(\Sigma) = \{f_i^n \mid f^n \in \Sigma \text{ e } 1 \leq i \leq n\}$. Os caminhos serão definidos como as sequências finitas de elementos de $\Pi(\Sigma)$. A cada árvore podemos associar um subconjunto de sequências, eventualmente infinitas, de elementos de $\Pi(\Sigma) \cup \{a \mid a \in \Sigma\}$. Uma árvore finita t pode ser descrita pelo conjunto (linguagem) dos seus *ramos* (caminhos maximais)[6] do seguinte modo:

$$\begin{aligned}\Pi(t) \quad = \quad & \{\tau_{i_1}^1 \ldots \tau_{i_n}^n \tau \mid \text{ onde } \tau \text{ é uma constante, } i_1 \ldots i_n \text{ pertence ao domínio} \\ & \text{de } t,\ t(i_1 \ldots i_{j-1}) = \tau^j \in \Sigma, \text{ para todo } 1 \leq j \leq n \text{ e } t(i_1 \ldots i_n) = \tau\}\end{aligned}$$

Analogamente, podemos caracterizar as árvores racionais e infinitas, considerando a linguagem $\Pi_\omega(t)$ de caminhos infinitos, ou a linguagem $\Pi_p(t)$ de todos os prefixos finitos de

[6]Um caminho p é maximal numa árvore t, se é um caminho em t e não existe q tal que $q \neq \epsilon$ e pq seja um caminho em t. A designação de *ramos* corresponde à de *branches* em inglês, usada por exemplo em [Cou83].

elementos de $\Pi_\omega(t) \cup \Pi(t)$ [Cou83]. Em particular, uma árvore t é racional se e só se a linguagem $\Pi_p(t)$ é regular. A noção de *ramo* pode ser generalizada ao conjunto dos termos, considerando as árvores finitas sobre $\Sigma \cup X$.

Começamos por definir a linguagem $\mathcal{LC}_0$. Sejam X o conjunto de variáveis , Σ o conjunto de símbolos de função e $\Pi(\Sigma)$ como acima. Definimos os *termos* $t, t', t_0, \ldots$, os *caminhos* $p, p', q, p_0, \ldots$, os *valores* $v, v', v_0, \ldots$ e as *restrições* $c, c', c_0, \ldots$ da linguagem $\mathcal{LC}_0$ pela seguinte gramática em notação (semelhante à) de Backus-Naur (BNF):

$$
\begin{array}{rcll}
t & ::= & x & \\
 & | & a & \\
 & | & f(t_1, \ldots, t_n) & n > 0 \\
p & ::= & \epsilon & \\
 & | & pf_i^n & n > 0,\ 1 \leq i \leq n \\
v & ::= & t & \\
 & | & t.p & \\
c & ::= & v.f^n & n > 0 \\
 & | & v \doteq v & \\
 & | & \top & \\
 & | & \bot & \\
 & | & \neg c & \\
 & | & c \wedge c & \\
 & | & c \vee c &
\end{array}
$$

As restrições da forma $v.f^n$, $v \doteq v$, $\top$ e $\bot$ são designadas *restrições atómicas positivas* e as negações $\neg v.f^n$ e $\neg v \doteq v$, *restrições atómicas negativas.* Para qualquer termo t, identificamos o valor t com $t.\epsilon$.

Intuitivamente, e considerando a correspondência entre caminhos e posições (e entre termos e árvores), um valor da forma $t.p$ é o sub-termo de t na "posição" p e uma restrição da forma $t.p.f^n$ indica que o termo t tem na "posição" p um sub-termo cujo functor é f^n. Por exemplo, sendo t o termo $f(a, g(b, c))$ as seguintes restrições são satisfeitas $t.f^2$, $t.f_2^2.g^2$ ou $t.f_1^2 \doteq a$.

Em vez de introduzir uma axiomatização ou uma álgebra para a linguagem $\mathcal{LC}_0$, consideramos cada restrição c como uma abreviatura de fórmulas de $\mathcal{L}$. Por exemplo, temos as seguintes correspondências:

$$
\begin{array}{rcl}
f(x,a).f_1^2.g^1 & \longrightarrow & \exists z_1, z_2.\, f(x,a) = f(g(z_1), z_2) \\
x.f_2^2 g_1^2 \doteq y & \longrightarrow & \exists z_1, z_2.\, x = f(z_1, g(y, z_2)) \\
x.f_1^2 \doteq y.g_1^2 h_1^1 & \longrightarrow & \exists z_1, z_2, z_3.\, (x = f(z_1, z_2) \ \wedge\ y = g(h(z_1), z_3))
\end{array}
$$

Para definir estas abreviaturas são necessárias as duas noções a seguir introduzidas.

Definição 2.23 *O* comprimento $|p|$ *dum caminho* p*, é definido indutivamente na estrutura de* p*:*

(i) $|\epsilon| \,=\, 0$

(ii) $|pf_i^n| \,=\, 1 + |p|$

Definição 2.24 *Sendo* p *um caminho e* t *um termo, o termo* $\pi(p,t)$*, expansão de* p *em* t*, é definido, a menos de renomeação de variáveis novas, por indução na estrutura de* p*:*

(i) $\pi(\epsilon, t) = t$*;*

(ii) $\pi(f_i^n p, t) \,=\, f(z_1, \ldots, z_{i-1}, \pi(p,t), z_{i+1}, \ldots, z_n)$ *onde* $z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n$ *são variáveis (novas) que não ocorrem em* $\pi(p,t)$*.*

É fácil ver, por indução no comprimento de p, que se t for um termo então $\pi(p,t)$ é um termo.

Usaremos ainda o seguinte teorema da lógica de 1ª ordem com igualdade:

Teorema 2.25 *Seja* $\phi[t/z]$ *a fórmula que resulta de* ϕ *substituindo todas as ocorrências de* z *por* t*. Se* z *não ocorre em* t*, então*

$$\vdash\ \phi[t/z] \ \Leftrightarrow\ \exists z\ (\phi \ \wedge\ z = t)$$

Definição 2.26 *As restrições de* $\mathcal{LC}_0$ *da forma* $v.f^n$ *e* $v \doteq v$ *são definidas, por indução na* *estrutura de v, como abreviaturas de fórmulas em $\mathcal{L}$*

$$\begin{array}{rcl}
t.f^n & \equiv & \exists z_1,\ldots,z_n.\ t = f(z_1,\ldots,z_n) \\
t.p.f^n & \equiv & \exists z_1,\ldots,z_m.\ t = \pi(p, f(z_1,\ldots,z_n)) \\
& & \textit{onde } m \geq n \textit{ e } z_{n+1},\ldots,z_m \textit{ são as variáveis introduzidas por } \pi \textit{ em } p; \\
t_1 \doteq t_2 & \equiv & t_1 = t_2; \\
t_1.p \doteq t_2 & \equiv & \exists z, z_1,\ldots,z_m.\ (t_1 = \pi(p,z)\ \wedge\ z = t_2) \\
& & \textit{onde } m \geq 0 \textit{ e } z_1,\ldots,z_m \textit{ são as variáveis introduzidas por } \pi \textit{ em } p; \\
t_2 \doteq t_1.p & \equiv & t_1.p \doteq t_2; \\
t_1.p \doteq t_2.q & \equiv & \exists z_1,\ldots,z_m.\ (t_1 = \pi(p,z_1)\ \wedge\ t_2 = \pi(q,z_1)) \\
& & \textit{onde } m \geq 1 \textit{ e } z_2,\ldots,z_m \textit{ são as variáveis introduzidas por } \pi \textit{ em } p \textit{ e } q.
\end{array}$$

Na definição anterior, como identificamos t com $t.\epsilon$, bastaria considerar apenas a segunda e a sexta abreviatura. As restantes são casos particulares daquelas.

Com estas definições é evidente que a cada restrição de $\mathcal{LC}_0$ corresponde uma fórmula de $\mathcal{L}$.

Sempre que v e v' sejam termos escrevemos $v = v'$ em vez de $v \doteq v'$. As noções de ocorrência livre de variáveis, conjunto de variáveis $\mathsf{Vars}(c)$ e substituição em $\mathcal{L}$ são definidas em $\mathcal{LC}_0$ considerando as abreviaturas da Definição 2.26. Considerando as fórmulas de $\mathcal{LC}_0$ fórmulas sem quantificadores, definem-se, analogamente, forma normal disjuntiva e forma normal conjuntiva em $\mathcal{LC}_0$.

A principal propriedade da linguagem $\mathcal{LC}_0$ é que qualquer fórmula de $\mathcal{L}$ é equivalente a uma restrição de $\mathcal{LC}_0$, quando se consideram as interpretações nas álgebras das árvores $\mathcal{T}$, $\mathcal{RT}$ ou $\mathcal{IT}$. Isto é, qualquer fórmula das teorias $E_{\mathcal{T}}$ ou $E_{\mathcal{RT}}$ é equivalente a uma restrição de $\mathcal{LC}_0$. Este resultado foi apresentado em [DMV94] e é consequência do Teorema 2.30 e do Teorema 2.32.

Vamos introduzir um subconjunto de $\mathcal{LC}_0$ onde não ocorrem termos complexos e que será suficiente para provar aquele resultado. Denominamos $\mathcal{LC}$ aquele subconjunto e sejam X e Σ como anteriormente. Definimos os *caminhos* $p, p', q, p_0, \ldots$, os *valores* $v, v', v_0, \ldots$ e as *restrições* $c, c', c_0, \ldots$ da linguagem $\mathcal{LC}$ pela seguinte gramática em BNF:

$$
\begin{array}{rcll}
p & ::= & \epsilon & \\
 & | & pf_i^n & n > 0,\ 1 \leq i \leq n \\
v & ::= & x.p & \\
 & | & a & \\
c & ::= & v.f^n & n > 0 \\
 & | & v \doteq v & \\
 & | & \bot & \\
 & | & \top & \\
 & | & \neg c & \\
 & | & c \ \wedge \ c & \\
 & | & c \ \vee \ c &
\end{array}
$$

Os Lemas 2.27 e 2.29 justificam a “eliminação” de termos complexos.

Lema 2.27 *Sendo $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$, para toda a variável x, functor f^n, termos t_i, para $1 \leq i \leq n$, e caminho p, tem-se*[7]*:*

$$\mathcal{A} \ \models \ x = f(t_1, \ldots, t_n) \ \Leftrightarrow \ \bigwedge_{i=1}^{n} x.f_i^n \doteq t_i \qquad n > 0 \tag{2.2}$$

$$\mathcal{A} \ \models \ x.p \doteq f(t_1, \ldots, t_n) \ \Leftrightarrow \ \bigwedge_{i=1}^{n} x.pf_i^n \doteq t_i \qquad n > 0 \tag{2.3}$$

Dem. (2.2) Basta provar as seguintes equivalências:

$$
\begin{aligned}
\mathcal{A} \models x = f(t_1, \ldots, t_n) \quad & \Leftrightarrow \quad \exists z_1, \ldots, z_n.\ (x = f(z_1, \ldots, z_n) \ \wedge \ \bigwedge_{i=1}^{n} z_i = t_i) \\
& \Leftrightarrow \quad \bigwedge_{i=1}^{n} \exists z_1, \ldots, z_n.\ (x = f(z_1, \ldots, z_n) \ \wedge \ z_i = t_i) \\
& \Leftrightarrow \quad \bigwedge_{i=1}^{n} x.f_i^n \doteq t_i
\end{aligned}
$$

onde $z_1, \ldots, z_n \notin \bigcup_{i=1}^{n} \mathsf{Vars}(t_i) \ \cup \ \{x\}$. A primeira equivalência resulta do facto da igualdade ser interpretada como a identidade em $\mathcal{A}$. A terceira resulta da definição das restrições com caminhos. Na segunda,

($\Rightarrow$)
É evidente.

[7] *A separação em dois casos é apenas para clareza da demonstração.*

($\Leftarrow$)
Seja ρ uma atribuição de variáveis tal que

$$\mathcal{A} \models_\rho \bigwedge_{i=1}^{n} \exists z_1, \ldots, z_n.\ (x = f(z_1, \ldots, z_n)\ \wedge\ z_i = t_i)$$

Para cada $1 \leq i \leq n$ existe uma atribuição de variáveis ρ_i que difere de ρ nos valores atribuídos às variáveis de $\bar{z}$ e tal que $\mathcal{A} \models_{\rho_i} x = f(z_1, \ldots, z_n) \wedge z_i = t_i$. Em particular, verifica-se $\rho_i(x) = \rho(x) = t_x$, para $1 \leq i \leq n$. Basta ver que ρ_i é a mesma (única) para todos os i. Por construção sabemos que $t_x(k) = \rho(t_k)$, donde $\rho_i(z_k) = \rho(t_k)$ para todo $1 \leq k \leq n$, o que prova que existe uma única extensão ρ' de ρ a $\bar{z}$ tal que

$$\mathcal{A} \models_{\rho'} x = f(z_1, \ldots, z_n)\ \wedge\ \bigwedge_{i=1}^{n} z_i = t_i$$

Equivalentemente, os Axiomas A4 e A5 das árvores (finitas ou racionais) garantem a unicidade de ρ'.

(2.3) Se $p = \epsilon$ resulta da alínea anterior. Caso contrário, seja

$$\begin{aligned}
\mathcal{A} \models x.p \doteq f(t_1, \ldots, t_n) & \\
\Leftrightarrow\quad & \exists z_1, \ldots, z_m.\ (x = \pi(p, f(z_1, \ldots, z_n))\ \wedge\ \bigwedge_{i=1}^{n} z_i = t_i),\ m \geq n \\
\Leftrightarrow\quad & \bigwedge_{i=1}^{n} \exists z_1, \ldots, z_m.\ (x = \pi(p, f(z_1, \ldots, z_n))\ \wedge\ z_i = t_i) \\
\Leftrightarrow\quad & \bigwedge_{i=1}^{n} \exists z_1, \ldots, z_m.\ x = \pi(p, f(z_1, \ldots, t_i, \ldots, z_n)) \\
\Leftrightarrow\quad & \bigwedge_{i=1}^{n} \exists z_1, \ldots, z_m.\ x = \pi(pf_i^n, t_i) \\
\Leftrightarrow\quad & \bigwedge_{i=1}^{n} x.pf_i^n \doteq t_i
\end{aligned}$$

Pelos mesmos motivos que na alínea anterior, a primeira, segunda, terceira e quinta equivalências verificam-se. Falta apenas ver que com a apropriada renomeação de variáveis:

$$\pi(pf_i^n, t_i) = \pi(p, f(z_1, \ldots, t_i, \ldots, z_n))$$

Por indução na estrutura de p:

Base. $\pi(\epsilon f_i^n, t_i) = \pi(f_i^n, t_i) = f(z_1, \ldots, t_i, \ldots, z_n) = \pi(\epsilon, f(z_1, \ldots, t_i, \ldots, z_n))$

Indução. Seja $p = g_j^k q$ com $|q| \geq 0$,

$$\begin{aligned}
\pi(pf_i^n, t_i) &= \pi(g_j^k q f_i^n, t_i) = g(z_{i_1}, \ldots, \pi(qf_i^n, t_i), \ldots, z_{i_k}) \\
&= g(z_{i_1}, \ldots, \pi(q, f(z_1, \ldots, t_i, \ldots, z_n)), \ldots, z_{i_k}) \\
&= \pi(g_j^k q, f(z_1, \ldots, t_i, \ldots, z_n)) \\
&= \pi(p, f(z_1, \ldots, t_i, \ldots, z_n))
\end{aligned}$$

2

Proposição 2.28 *Sendo $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$ e dado um termo t, uma variável x e um caminho q, tem-se que*

$$\mathcal{A} \models x.q \doteq t \;\Leftrightarrow\; c'$$

onde c' é uma conjunção de restrições da forma $x.p \doteq v$, com v uma constante ou uma variável que ocorre em t. O número de restrições atómicas em c' é igual ao número de ramos de t e o comprimento de cada caminho p é igual ao comprimento de q mais o comprimento de cada ramo de t menos um.

Dem. Provamos por indução no tamanho de t.

Base. Se $|t| = 1$ então t é uma variável ou uma constante; em ambos os casos v é t e c' é $x.q \doteq v$. O número de ramos de t é 1, o número de restrições atómicas é 1 e $|q| = |q| + (1 - 1)$.

Indução. Suponhamos que o lema é válido para todo o termo cujo tamanho é menor que k e seja $t = f(t_1, \ldots, t_n)$ tal que $|t| = k$. Pelo Lema 2.27,

$$\mathcal{A} \models x.q \doteq f(t_1, \ldots, t_n) \;\Leftrightarrow\; \bigwedge_{i=1}^{n} x.qf_i^n \doteq t_i$$

Por hipótese de indução, cada restrição $x.qf_i^n \doteq t_i$ é equivalente a uma conjunção de restrições da forma $x.p \doteq v$, com v uma variável ou uma constante. Se r_i for o número de ramos para cada t_i, então t tem $\sum_{i=1}^{n} r_i$ ramos e, usando a hipótese de indução, esse é o número de restrições atómicas resultantes. Como cada ramo de t é obtido de um ramo de t_i concatenando f_i^n, usando a hipótese de indução, obtém-se que o comprimento de cada caminho é igual ao comprimento de q mais o comprimento de cada ramo de t menos um.

2

Lema 2.29 *Seja $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$. Para todo o valor v de $\mathcal{LC}$, functores $f^n, g^k \in \Sigma$, caminho p e termos t_i, para $1 \leq i \leq n$, tem-se que:*

$$
\begin{array}{lllll}
\mathcal{A} & \models & f(t_1,\ldots,t_n).f_i^n p \doteq v & \Leftrightarrow \ t_i.p \doteq v & \\
\mathcal{A} & \models & f(t_1,\ldots,t_n).g_i^k p \doteq v & \Leftrightarrow \ \perp & \text{se } f^n \not\equiv g^k \\
\mathcal{A} & \models & f(t_1,\ldots,t_n).f_i^n p.g^k & \Leftrightarrow \ t_i.p.g^k & \\
\mathcal{A} & \models & f(t_1,\ldots,t_n).g_i^k p.f^n & \Leftrightarrow \ \perp & \text{se } f^n \not\equiv g^k
\end{array}
$$

Dem. As equivalências resultam directamente das definições das restrições e dos Axiomas A4 e A5 para $E_{\mathcal{A}}$. Vamos supor que v é um termo t. O caso de v ser da forma $t.p$ prova-se de maneira análoga atendendo à definição das restrições.

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathcal{A} \models f(t_1,\ldots,t_n).f_i^n p \doteq t} \\
\Leftrightarrow & \exists z_1,\ldots,z_m.\ (f(t_1,\ldots,t_n) = f(z_1,\ldots,z_{i-1},\pi(p,z_i),z_{i+1},\ldots,z_n)\ \wedge\ z_i = t) \\
\Leftrightarrow & \exists z_1,\ldots,z_m.\ (t_1 = z_1\ \wedge\ \ldots\ \wedge\ t_i = \pi(p,z_i)\ \wedge\ \ldots\ \wedge\ t_n = z_n\ \wedge\ z_i = t) \\
\Leftrightarrow & \exists z_i, z_{n+1},\ldots,z_m.\ (t_i = \pi(p,z_i)\ \wedge\ z_i = t)\ \wedge \\
& \exists z_1,\ldots,z_{i-1},z_{i+1},\ldots,z_n.\ (z_1 = t_1\ \wedge\ \ldots\ \wedge\ z_n = t_n) \\
\Leftrightarrow & t_i.p \doteq t
\end{array}
$$

A segunda equivalência resulta do Axioma A4 de $E_{\mathcal{A}}$; a quarta resulta do facto de que $\exists z_1,\ldots,z_{i-1},z_{i+1},\ldots,z_n.\ (z_1 = t_1\ \wedge\ \ldots\ \wedge\ z_n = t_n)$, com $z_i \notin \cup_{i=1}^{n}\mathsf{Vars}(t_i)$ e distintos, ser válida em $\mathcal{A}$.

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathcal{A} \models f(t_1,\ldots,t_n).g_i^k p \doteq t} \\
\Leftrightarrow & \exists z_1,\ldots,z_m.\ (f(t_1,\ldots,t_n) = g(z_1,\ldots,\pi(p,z_i),\ldots,z_k)\ \wedge\ z_i = t) \\
\Leftrightarrow & \perp
\end{array}
$$

pelo Axioma A5 de $E_{\mathcal{A}}$.

Analogamente se provam as outras duas equivalências do enunciado. 2

O Lema 2.29 garante que, para $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$, as seguintes regras de reescrita para valores são[8] correctas:

$$
\begin{array}{lll}
f(t_1,\ldots,t_n).f_i^n p & \mapsto & t_i.p \\
f(t_1,\ldots,t_n).g_i^k p & \mapsto & \omega,\ \text{se } f^n \not\equiv g^k
\end{array}
$$

onde ω é uma constante especial que, indica que a restrição onde ocorre é equivalente a $\perp$.

Temos, então o seguinte teorema:

[8] Isto é, as restrições onde ocorram são equivalentes em $\mathcal{A}$.

Teorema 2.30 *Sendo $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$, é possível transformar qualquer restrição $c \in \mathcal{LC}_0$ numa restrição $c' \in \mathcal{LC}$, tal que $\mathcal{A} \models c \Leftrightarrow c'$.*

Dem. É evidente que $\mathcal{LC} \subset \mathcal{LC}_0$. Toda a restrição $c \in \mathcal{LC}_0$ reduz-se a uma restrição $c' \in \mathcal{LC}$ por aplicação das seguintes regras de reescrita:

L1. $f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n) \;\mapsto\; t_1 = u_1 \;\wedge\; \ldots \;\wedge\; t_n = u_n$

L2. $f(t_1, \ldots, t_n) = g(u_1, \ldots, u_k) \;\mapsto\; \bot$, se $f^n \not\equiv g^k$

L3. $x.p \doteq f(t_1, \ldots, t_n) \;\mapsto\; \bigwedge_{i=1}^{n} x.pf_i^n \doteq t_i$ se $n > 0$

L4. $f(t_1, \ldots, t_n).f_i^n p \;\mapsto\; t_i.p$

L5. $f(t_1, \ldots, t_n).g_i^k p \;\mapsto\; \omega$, se $f^n \not\equiv g^k$

L6. $f(t_1, \ldots, t_n).f^n \;\mapsto\; \top$

L7. $f(t_1, \ldots, t_n).g^k \;\mapsto\; \bot$, se $f^n \not\equiv g^k$

L8. $\omega.f^n \;\mapsto\; \bot$

L9. $v \doteq v' \;\mapsto\; \bot$, se v ou v' é ω

As regras L1 e L2 correspondem a axiomas de $E_{\mathcal{A}}$ e portanto são correctas em $\mathcal{A}$. A regra L3 é correcta pelo Lema 2.27 e as regras L4 a L9 pelo Lema 2.29. Para todas as regras o número de termos complexos que ocorrem na restrição do lado direito é menor do que o número de termos complexos que ocorrem na restrição do lado esquerdo e cada regra só pode ser aplicada um número finito de vezes. Se nenhuma regra se aplicar a restrição pertence a $\mathcal{LC}$. 2

Para provar a equivalência entre restrições de $\mathcal{LC}$ e fórmulas de $\mathcal{L}$, para $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$, usaremos ainda o facto de $\doteq$ ser uma relação de equivalência nos valores de $\mathcal{LC}$. Em particular,

Lema 2.31 *Sendo $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$ e v, v' e v'' valores de $\mathcal{LC}$,*

$$\mathcal{A} \models v' \doteq v \;\wedge\; v'' \doteq v \;\Leftrightarrow\; v' \doteq v \;\wedge\; v' \doteq v''$$

Dem. Se v, v', v'' são termos o lema resulta da simetria e transitividade da igualdade. Os restantes casos são todos semelhantes e, portanto, apenas analisaremos um:

$$\begin{aligned}
\mathcal{A} \models x.p \doteq a \;\wedge\; y.q \doteq a \quad & \Leftrightarrow \quad \exists \bar{z}.\; x = \pi(p, a) \;\wedge\; \exists \bar{w}.\; y = \pi(q, a) \\
& \Leftrightarrow \quad \exists z_0, \bar{z}\bar{w}.\; (x = \pi(p, z_0) \;\wedge\; y = \pi(q, z_0) \;\wedge\; z_0 = a) \\
& \Leftrightarrow \quad \exists z_0, \bar{z}.\; (x = \pi(p, z_0) \;\wedge\; z_0 = a) \;\wedge \\
& \qquad\;\; \exists z_0, \bar{z}\bar{w}.\; (x = \pi(p, z_0) \;\wedge\; y = \pi(q, z_0)) \\
& \Leftrightarrow \quad x.p \doteq a \;\wedge\; x.p \doteq y.q
\end{aligned}$$

onde a terceira equivalência resulta, como na demonstração do Lema 2.27, do facto de que se $\mathcal{A} \models_\rho \exists z_0, \bar{z}\bar{w}.\ x = \pi(p, z_0) \ \wedge\ y = \pi(q, z_0) \ \wedge\ z_0 = a$ e $z_0, \bar{z}, \bar{w}$ são distintas de x e y, então existe um única extensão ρ' de ρ a $\{z_0\} \cup \{\bar{z}\} \cup \{\bar{w}\}$, tal que $\mathcal{A} \models_{\rho'} x = \pi(p, z_0) \ \wedge\ y = \pi(q, z_0) \ \wedge\ z_0 = a$. 2

Teorema 2.32 *Sendo $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$, para cada fórmula de $\mathcal{L}$ existe uma restrição de $\mathcal{LC}$ equivalente.*

Dem. Maher [Mah88] provou que qualquer fórmula em $\mathcal{L}$ interpretada em $\mathcal{A}$ é equivalente a uma combinação booleana de fórmulas básicas e uma fórmula básica é satisfazível a menos que seja equivalente a $\perp$. Em particular, se uma fórmula é fechada, ela é equivalente a $\top$ ou $\perp$. Então, basta ver que cada fórmula básica é equivalente a uma fórmula em $\mathcal{LC}$. Se for $\perp$, está provado, porque $\perp \in \mathcal{LC}$.

Seja $\mathcal{A} = \mathcal{RT}$. Neste caso, as fórmulas básicas são da forma:

$$\exists u_1, \ldots, u_m.\ (x_1 = t_1 \ \wedge\ \ldots\ \wedge\ x_n = t_n \ \wedge\ u_1 = r_1 \ \wedge\ \ldots\ \wedge\ u_p = r_p)$$

onde $\bar{u}$ e $\bar{x}$ são conjuntos disjuntos de variáveis, a conjunção de equações está em forma resolvida racional e $p \leq m$. Sem falta de generalidade, podemos considerar que as únicas variáveis que ocorrem em t_i, para $1 \leq i \leq n$, e em r_j, para $1 \leq j \leq p$, são os u_k, para $1 \leq k \leq m$. Pela Proposição 2.28, cada equação pode ser substituída por uma conjunção de restrições da forma $x_i.p_{i_l} \doteq v_l$ ou $u_j.p_{j_m} \doteq v_m$, onde cada v_l ou v_m é uma constante ou uma variável u_k. Seja c_x a conjunção das restrições da primeira forma e c_u a conjunção das restrições da segunda forma. Em c_x aplique-se, enquanto possível, a seguinte regra de reescrita:

LC1. $x.p \doteq u_k \ \wedge\ y.q \doteq u_k \ \mapsto\ x.p \doteq y.q \ \wedge\ x.p \doteq u_k$

O número de aplicações desta regra é finito, porque de cada vez diminui a ocorrência de restrições da forma $x.p \doteq u_k$. Seja c a restrição resultante, pelo Lema 2.31 tem-se que $\mathcal{RT} \models c_x \ \Leftrightarrow\ c$. Então,

$$\begin{array}{rcl}
\mathcal{RT} & \models & \exists u_1, \ldots, u_m.\ (x_1 = t_1 \ \wedge\ \ldots\ \wedge\ x_n = t_n \ \wedge\ u_1 = r_1 \ \wedge\ \ldots\ \wedge\ u_p = r_p) \\
& \Leftrightarrow & \exists u_1, \ldots, u_m.\ (c \ \wedge\ c_u) \\
& \Leftrightarrow & \exists u_1, \ldots, u_m.\ c \ \wedge\ \exists u_1, \ldots, u_m.\ c_u \\
& \Leftrightarrow & \exists u_1, \ldots, u_m.\ c \\
& \Leftrightarrow & c'
\end{array}$$

A terceira equivalência é válida uma vez que $\exists u_1, \ldots, u_m.\ c_u$, é equivalente a $\top$, porque é equivalente a uma fórmula básica fechada (diferente de $\perp$). Para eliminar os quantificadores em $\exists u_1, \ldots, u_m.\ c$, e dado que cada u_k ocorre no máximo uma vez em c, basta aplicar para cada um, uma das seguintes regras de reescrita:

LC2. $\exists u\ x.pf_i^n \doteq u \ \mapsto\ x.p.f^n$

LC3. $\exists u\ x = u \ \mapsto\ \top$

LC4. $\exists u\ c \ \mapsto\ c$ se u não ocorre em c

Estas regras são correctas e aplicam-se um número finito de vezes. Seja c' a restrição resultante e, por construção, $c' \in \mathcal{LC}$.

No caso de $\mathcal{A} = \mathcal{T}$, as formas básicas são da forma:

$$\exists u_1, \ldots, u_m.\ (x_1 = t_1 \ \wedge\ \ldots\ \wedge\ x_n = t_n)$$

onde o conjunto de equações está em forma resolvida e os u_i, para $1 \le i \le m$, são as variáveis que ocorrem em t_j, para $1 \le j \le n$. Basta, então, substituir cada equação por uma conjunção de restrições da forma $x_i.p_{i_l} \doteq v_l$, onde v_l é uma constante ou uma variável u_k, $1 \le k \le m$. Usando-se primeiro a regra LC1 e depois as regras LC2-LC4 obtém-se uma restrição $c' \in \mathcal{LC}$ equivalente à forma básica dada. 2

Em aplicações práticas, pode interessar transformar uma fórmula $\phi \in \mathcal{L}$ numa restrição $c \in \mathcal{LC}$ sem a prévia obtenção da combinação booleana de formas básicas equivalente a ϕ. Usando a linguagem $\mathcal{LU}$, a seguir introduzida, e resultados do Capítulo 4 é possível obter essa transformação. Para o caso de $\mathcal{A} = \mathcal{RT}$, uma nova demonstração do Teorema 2.32 é apresentada no Apêndice A.

2.2.2 Linguagem $\mathcal{LU}$

Embora as linguagens das secções anteriores pudessem ser usadas directamente para o estudo das propriedades relacionadas com o problema da satisfazibilidade das restrições é suficiente considerar um seu subconjunto, que designaremos por linguagem de restrições unitária, $\mathcal{LU}$. Em particular, esta linguagem terá analogias com algumas linguagens de árvores de atributos, como será visto no Capítulo 3. Essas analogias permitirão que o *Calculus* de restrições apresentado para $\mathcal{LU}$ nos Capítulos 4 e 6, possa ser directamente aplicado para aquelas linguagens [DV92].

A linguagem $\mathcal{LU}$ difere da linguagem $\mathcal{LC}$ pela ausência de caminhos de tamanho superior a um e no facto de que a negação só ocorre para determinadas restrições atómicas. Uma vez que os únicos termos de $\mathcal{LU}$ são as variáveis e as constantes, quando não especificado de outro modo, as letras u, w, s e t denotarão também variáveis de $\mathcal{LU}$.

Sendo X o conjunto das variáveis e Σ o conjunto de símbolos de função, definimos os *valores* $v, v', v_0, \ldots$ e as *restrições* $c, c', c_0, \ldots$ da linguagem $\mathcal{LU}$ pela seguinte gramática em BNF:

$$
\begin{array}{rcll}
v & ::= & x & \\
 & | & a & \\
c & ::= & v.f^n & n > 0 \\
 & | & v = v & \\
 & | & v.f_i^n \doteq v & n > 0,\ 1 \leq i \leq n \\
 & | & v \neq v & \\
 & | & \neg v.f^n & n > 0 \\
 & | & \top & \\
 & | & \bot & \\
 & | & c \wedge c & \\
 & | & c \vee c &
\end{array}
$$

Mais uma vez, a cada restrição $c \in \mathcal{LU}$ corresponde uma fórmula de $\mathcal{L}$, considerando as seguintes abreviaturas:

$$
\begin{array}{rcl}
v.f^n & \equiv & \exists z_1, \ldots, z_n.\ v = f(z_1, \ldots, z_n) \\
v.f_i^n \doteq v' & \equiv & \exists z_1, \ldots, z_n.\ (v = f(z_1, \ldots, z_n) \wedge z_i = v')
\end{array}
$$

Para qualquer restrição $c \in \mathcal{LC}$ é possível construir uma restrição $c' \in \mathcal{LU}$ tal que c é satisfazível se e só se c' for satisfazível. Pela definição das abreviaturas, introduzindo variáveis quantificadas existencialmente podemos transformar cada restrição atómica positiva de $\mathcal{LC}$ numa conjunção de restrições de $\mathcal{LU}$. Para transformar as restrições atómicas negativas é necessário o seguinte lema:

Lema 2.33 *Sendo $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$, para toda a variável x, functor $f^n \in \Sigma$ e variável ou constante v tem-se que:*

$$\mathcal{A} \models \neg x.f_i^n \doteq v \ \Leftrightarrow\ \neg x.f^n \ \vee\ \exists z.\ (x.f_i^n \doteq z \ \wedge\ z \neq v)$$

para todo $1 \leq i \leq n$.

Dem. Sem falta de generalidade, podemos supor que v é uma constante a. Temos,

$$(2.4) \qquad \mathcal{A} \models \neg x.f_i^n \doteq a \;\Leftrightarrow\; \neg\exists\bar{z}.\,(x = f(\bar{z}) \;\wedge\; z_i = a)$$
$$\Leftrightarrow\; \forall\bar{z}.(x \neq f(\bar{z}) \;\vee\; (x = f(\bar{z}) \;\wedge\; z_i \neq a))$$

e

$$(2.5) \qquad \mathcal{A} \models \neg x.f^n \;\vee\; \exists z.\,(x.f_i^n \doteq z \;\wedge\; z \neq a)$$
$$\Leftrightarrow\; \forall\bar{z}.\, x \neq f(\bar{z}) \;\vee\; \exists z, \bar{z}.\; (x = f(\bar{z}) \;\wedge\; z_i = z \;\wedge\; z \neq a)$$
$$\Leftrightarrow\; \forall\bar{z}.\, x \neq f(\bar{z}) \;\vee\; \exists\bar{z}.\; (x = f(\bar{z}) \;\wedge\; z_i \neq a)$$

(2.5) $\Rightarrow$ (2.4)

Para toda a atribuição de variáveis ρ, vamos ver que se $\mathcal{A} \models_\rho$ (2.5) então $\mathcal{A} \models_\rho$ (2.4). Se $\mathcal{A} \models_\rho \forall\bar{z}.\, x \neq f(\bar{z})$ é evidente. Senão, suponhamos que existe uma atribuição de variáveis ρ tal que $\mathcal{A} \models_\rho \exists\bar{z}.\; (x = f(\bar{z}) \;\wedge\; z_i \neq a)$ e $\mathcal{A} \not\models_\rho \forall\bar{z}.(x \neq f(\bar{z}) \;\vee\; (x = f(\bar{z}) \;\wedge\; z_i \neq a))$. Sabemos que ρ admite uma única extensão às variáveis $\bar{z}$ que satisfaz $x = f(\bar{z}) \;\wedge\; z_i \neq a$ e seja $\rho(x) = t_x \in \mathcal{U}_\mathcal{A}$. Por outro lado, existe uma extensão ρ_0 de ρ, tal que $t_x = \rho(x) = \rho_0(x) = \rho_0(f(\bar{z}))$ e $\rho_0(z_i) = a$ o que é um absurdo, porque $t_x \mid i \neq a$.

(2.4) $\Rightarrow$ (2.5)

Se $\mathcal{A} \models_\rho$ (2.4) então $\forall\bar{d} \in \mathcal{U}_\mathcal{A}^n$, $\mathcal{A} \models_{\rho[\bar{d}/\bar{z}]} x \neq f(\bar{z}) \;\vee\; z_i \neq a$. Se para algum $\bar{d}_0 \in \mathcal{U}_\mathcal{A}^n$, $\mathcal{A} \models_{\rho[\bar{d}_0/\bar{z}]} x = f(\bar{z})$ então $\mathcal{A} \models_{\rho[\bar{d}_0/\bar{z}]} z_i \neq a$ e portanto $\mathcal{A} \models_\rho \exists\bar{z}.\; (x = f(\bar{z}) \;\wedge\; z_i \neq a)$. Se tal $\bar{d}_0$ não existir, o resultado é evidente.

2

Este lema permite-nos transformar, por exemplo

$$\neg x.f_1^2 g_1^1 \doteq y$$

em

$$\neg x.f^2 \;\vee\; (x.f_1^2 \doteq z_1 \;\wedge\; (\neg z_1.g^1 \;\vee\; (z_1.g_1^1 = z_2 \;\wedge\; z_2 \neq y)))$$

tal que a primeira restrição é satisfazível se e só se a segunda o for.

Teorema 2.34 *Sendo $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$, para toda a restrição $c \in \mathcal{LC}$ existe uma restrição $c' \in \mathcal{LU}$ tal que:*

$$\mathcal{A} \models c \;\Leftrightarrow\; \exists\bar{z}.\, c', \qquad \textit{onde } \bar{z} \notin \mathsf{Vars}(c),\ (\mathsf{Vars}(c') \backslash \{\bar{z}\}) \subseteq \mathsf{Vars}(c) \textit{ e para todo } z_i \in \{\bar{z}\} \textit{ existe } x \in \mathsf{Vars}(c) \textit{ e um caminho } p \not\equiv \epsilon \textit{ tal que } \mathcal{A} \models \exists\bar{z}.\, c' \;\Rightarrow\; x.p \doteq z_i$$

Dem. Seja $c \in \mathcal{LC}$. Começamos por transformar c de tal modo que só ocorram negações aplicadas a restrições atómicas ($i.e$as únicas negações sejam restrições atómicas negativas). Para tal usam-se as seguintes regras de reescrita correspondentes a teoremas da lógica de 1ª ordem:

U1. $\neg(c_0 \ \wedge \ c_1) \ \mapsto \ \neg c_0 \ \vee \ \neg c_1$

U2. $\neg(c_0 \ \vee \ c_1) \ \mapsto \ \neg c_0 \ \wedge \ \neg c_1$

U3. $\neg\neg c \ \mapsto \ c$

A restrição resultante é logicamente equivalente a c. A seguir, nas restrições atómicas positivas, são eliminados os caminhos de comprimento superior a um. Sendo $|p| \geq 1$, $|q| \geq 1$ e v uma variável ou uma constante, aplicam-se enquanto possível as seguintes regras de reescrita:

U4. $x.p \doteq y.q \quad \mapsto \quad \exists z.\ (x.p \doteq z \ \wedge \ y.q \doteq z)$
onde z é uma variável nova

U5. $v \doteq x.p \quad \mapsto \quad x.p \doteq v$

U6. $x.f_i^n p \doteq v \quad \mapsto \quad \exists z.\ (x.f_i^n \doteq z \ \wedge \ z.p \doteq v)$
onde z é uma variável nova

U7. $x.p.f^n \quad \mapsto \quad \exists z.\ (x.p \doteq z \ \wedge \ z.f^n)$
onde z é uma variável nova

A correcção destas regras resulta directamente das definições das restrições atómicas. Em nenhuma das regras aumenta o comprimento dos caminhos que ocorrem nas restrições do lado direito. Nas duas últimas regras, o comprimento dos caminhos nas restrições resultantes diminui. Pela primeira regra diminui o número de restrições atómicas envolvendo dois caminhos complexos. Destes factos resulta a terminação do sistema de reescrita, e que as restrições atómicas positivas irredutíveis pertencem a $\mathcal{LU}$.

Sendo v uma variável ou uma constante, para as restrições atómicas negativas as regras a aplicar são:

U8. $\neg x.f_i^n p \doteq y.g_j^m q \quad \mapsto \quad \neg x.f^n \ \vee \ \neg y.g^m \ \vee \ \exists z_1, z_2.\ (x.f_i^n \doteq z_1 \ \wedge \ y.g_j^m \doteq z_2$
$\wedge \ \neg z_1.p \doteq z_2.q)$
onde z_1 e z_2 são variáveis novas

U9. $\neg x.f_i^n p \doteq v \quad \mapsto \quad \neg x.f^n \ \vee \ \exists z.\ (x.f_i^n \doteq z \ \wedge \ \neg z.p \doteq v)$
onde z é uma variável nova

U10. $\neg x.g_j^m p.f^n \quad \mapsto \quad \neg x.g^m \ \vee \ \exists z.\ (x.g_j^m \doteq z \ \wedge \ \neg z.p.f^n)$
onde z é uma variável nova

A equivalência é garantida pelo Lema 2.33 e a terminação por argumentos idênticos aos usados no caso anterior.

Finalmente os quantificadores existenciais são "empurrados" para fora, usando teoremas da lógica de 1ª ordem, obtendo-se uma *forma prenexa normal*:

U11. $\exists z\ c \;\mapsto\; \exists z_1\ c[z_1/z]$ se z_1 não ocorre em c

U12. $\exists z\ c \vee c' \;\mapsto\; \exists z\ (c \vee c')$ se z não ocorre em c'

U13. $\exists z\ c \wedge c' \;\mapsto\; \exists z\ (c \wedge c')$ se z não ocorre em c'

Estas regras não alteram as restrições atómicas, a menos de renomeação de variáveis, e portanto as restrições irredutíveis são da forma $\exists \bar{z}.\ c'$, onde $c' \in \mathcal{LU}$.

Para a condição sobre as variáveis $\bar{z}$, basta ver que essas variáveis apenas são introduzidas pelas regras de reescrita U4–U10. Por indução no número de aplicações de alguma destas regras obtém-se o pretendido. 2

Limitando-nos ao problema da satisfazibilidade podíamos ter considerado directamente esta linguagem em vez de fazer referência a $\mathcal{LC}$ e a $\mathcal{LC}_0$. Para tal bastaria na demonstração do Teorema 2.32 considerar as seguintes equivalências:

$$\mathcal{T}, \mathcal{RT} \models x = f(t_1, \ldots, t_n) \Leftrightarrow \bigwedge_{i=1}^{n} \exists z_i.\ (x.f_i^n \doteq z_i \wedge z_i = t_i) \tag{2.6}$$

$$\mathcal{T}, \mathcal{RT} \models x \neq f(t_1, \ldots, t_n) \Leftrightarrow \neg x.f^n \vee \bigvee_{i=1}^{n} \exists z_i.\ (x.f_i^n \doteq z_i \wedge z_i \neq t_i) \tag{2.7}$$

No entanto, a utilização de linguagens mais gerais que $\mathcal{LU}$ é conveniente tanto em aplicações práticas (ver Capítulo 8), como para a obtenção de certos resultados teóricos (p.e. [DM91]).

Terminamos esta secção com um lema que será utilizado no Capítulo 4 na obtenção de restrições em formas normalizadas.

Lema 2.35 *Seja $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$. Para todas as variáveis $x, y \in X$, para todo o functor $f^n \in \Sigma$ e valores v_i, v_i', para $1 \leq i \leq n$, tem-se que*

$$\begin{aligned} \mathcal{A} \models\ & \bigwedge_{i=1}^{n} (x.f_i^n \doteq v_i \wedge y.f_i^n \doteq v_i') \wedge x \neq y \\ \Leftrightarrow\ & \bigwedge_{i=1}^{n} (x.f_i^n \doteq v_i \wedge y.f_i^n \doteq v_i') \wedge \bigvee_{i=1}^{n} v_i \neq v_i' \end{aligned} \tag{2.8}$$

Dem. Provamos um resultado mais geral. Para todo x, $y \in X$ e $f^n \in \Sigma$ tem-se que:

$$
\begin{aligned}
\mathcal{A} \models\ & x.f^n \ \wedge\ y.f^n \ \wedge\ x \neq y \\
\Leftrightarrow\ & \exists \bar{z}.\ x = f(z_1, \ldots, z_n) \ \wedge\ \exists \bar{w}.\ y = f(w_1, \ldots, w_n) \ \wedge\ x \neq y \\
\Leftrightarrow\ & \exists \bar{z}\bar{w}.\ (x = f(z_1, \ldots, z_n) \ \wedge\ y = f(w_1, \ldots, w_n) \ \wedge \\
& f(z_1, \ldots, z_n) \neq f(w_1, \ldots, w_n)) \\
\Leftrightarrow\ & \exists \bar{z}\bar{w}.\ (x = f(z_1, \ldots, z_n) \ \wedge\ y = f(w_1, \ldots, w_n) \ \wedge\ \bigvee_{j=1}^{n} z_j \neq w_j) \\
\Leftrightarrow\ & \exists \bar{z}\bar{w}.(\bigwedge_{j=1}^{n} (x.f_j^n \doteq z_j \ \wedge\ y.f_j^n \doteq w_j) \ \wedge\ \bigvee_{j=1}^{n} z_j \neq w_j)
\end{aligned}
$$

Então, dados v, v' valores e algum $1 \leq i \leq n$, tem-se que:

$$
\begin{aligned}
\mathcal{A} \models\ & x.f^n \ \wedge\ y.f^n \ \wedge\ x.f_i^n \doteq v \ \wedge\ y.f_i^n \doteq v' \ \wedge\ x \neq y \\
\Leftrightarrow\ & \exists \bar{z}\bar{w}.\ (x = f(z_1, \ldots, z_n) \ \wedge\ y = f(w_1, \ldots, w_n) \ \wedge \\
& z_i = v \ \wedge\ w_i = v' \ \wedge\ \bigvee_{j=1}^{n} z_j \neq w_j) \\
\Leftrightarrow\ & x.f^n \ \wedge\ y.f^n \ \wedge\ x.f_i^n \doteq v \ \wedge\ y.f_i^n \doteq v' \ \wedge \\
& (v \neq v' \ \vee\ \exists \bar{z}\bar{w}.\ (\bigwedge_{j=1, j\neq i}^{n} (x.f_j^n \doteq z_j \ \wedge\ y.f_j^n \doteq w_j) \ \wedge \bigvee_{j=1, j\neq i}^{n} z_j \neq w_j))
\end{aligned}
$$

Repetindo-se o processo para cada $1 \leq i \leq n$, obtém-se (2.8). 2

2.3 Linguagens de Programação e Restrições

Nesta secção enquadramos as linguagens $\mathcal{LC}_0$, $\mathcal{LC}$ e $\mathcal{LU}$, no conjunto das linguagens de *restrições*, em particular nas que estão directamente relacionadas com a Programação em Lógica *por Restrições*. A descrição geral das linguagens de restrições baseia-se em [JM94].

Em Programação em Lógica por Restrições, $\mathcal{CLP}(\mathcal{D})$[9], incorporam-se métodos de resolução de restrições num dado domínio $\mathcal{D}$, com métodos lógicos de computação. No esquema $\mathcal{CLP}$ [JL87], uma linguagem de programação em lógica, a sua semântica operacional e a sua semântica declarativa, podem ser parametrizadas por um 4-tuplo $(\Sigma, \mathcal{A}, \mathcal{C}, \mathcal{S})$, onde Σ é uma assinatura que pode conter outros símbolos de predicado além do símbolo '=', $\mathcal{A}$ é uma Σ-álgebra, $\mathcal{C}$ é uma classe de fórmulas da linguagem de 1ª ordem caracterizada por Σ e $\mathcal{S}$ é uma teoria (conjunto de axiomas).

[9] do inglês, *Constraint Logic Programming*

A classe $\mathcal{C}$ e a álgebra $\mathcal{A}$, definem um domínio $\mathcal{D}$ – domínio de restrições – e as fórmulas de $\mathcal{C}$ são chamadas *restrições*. As linguagens de restrições $\mathcal{C}$ contêm as fórmulas atómicas e são habitualmente fechadas para a conjunção e quantificação existencial. Quando se considera a negação ou outras conectivas lógicas denominam-se por *linguagens de restrições complexas*.

Claramente, os pares $(\mathcal{L},\mathcal{T})$ e $(\mathcal{L},\mathcal{RT})$ são domínios de restrições complexas. Na literatura, aparecem domínios associados a $\mathcal{T}$ e $\mathcal{RT}$, mas normalmente é só considerado um subconjunto de $\mathcal{L}$. Em particular, a linguagem Prolog pode ser vista como $\mathcal{CLP}(\mathcal{T})$[10], onde $\mathcal{C}$ é o subconjunto de $\mathcal{L}$ que contém as igualdades entre termos e é fechada para a conjunção e quantificação existencial. Relacionados com os domínios das árvores finitas e racionais são usados os domínios de estruturas de atributos (por exemplo, [AK86, AKP93a]), que serão analisados no Capítulo 3.

Podemos ainda citar outros domínios, usados, integralmente ou em parte, em várias implementações de $\mathcal{CLP}$:

Números reais: $\Re = (\mathcal{C},\mathcal{R})$, onde $\Sigma = \{0, 1, +^2, \times^2, =, <, \leq\}$, $\mathcal{U}_\mathcal{R} = \mathbb{R}$ é o conjunto de números reais e $\mathcal{R}$ é uma Σ-álgebra que interpreta as operações aritméticas e as relações de ordem como habitualmente. É usado nas linguagens de programação $\mathcal{CLP}(\mathcal{R})$ [JMSY90], Prolog-III [Col90] e CHIP [DvS+88], entre outras.

Sequências de caracteres: $\mathcal{WE} = (\mathcal{C},\mathcal{W})$, onde Σ contém a função binária '$\cdot$' e a constante λ, entre outras; $\mathcal{U}_\mathcal{W}$ é o conjunto de sequências finitas de constantes; o operador '$\cdot$' é interpretado como a concatenação de sequências e λ como a sequência vazia [Col90].

Valores booleanos: $\mathcal{BOOL} = (\mathcal{C},\mathcal{B})$, onde $\Sigma = \{0, 1, \neg, \wedge^2, \vee^2, \Rightarrow^2, \oplus^2, =\}$, $\mathcal{U}_\mathcal{B} = \{0, 1\}$ e $\mathcal{B}$ interpreta os símbolos de função de Σ como funções booleanas [Col90, DvS+88, ASS+88].

Domínios finitos: $\mathcal{FD} = (\mathcal{C},\mathcal{F})$, podem ser associados a domínios finitos de números inteiros, *i.e*, $\mathcal{U}_\mathcal{F} = \mathbb{Z}$, Σ contém $+^2$ e os símbolos de predicado $=$, $\neq$, $\leq$, e expressões que denotam que $m \leq x \leq n$, para m e n inteiros. Se $c \in \mathcal{C}$ então para cada $x \in \mathsf{Vars}(c)$ existe uma restrição da forma $m \leq x \leq n$, que indica a que intervalo de inteiros, a variável x pertence [DvS+88].

Em resumo, uma linguagem de restrições é um linguagem lógica de 1ª ordem que é associada a uma álgebra e para a qual é decidível o problema da satisfazibilidade. Exige-se, ainda, que as linguagens de restrições, definidas por um 4-tuplo $(\Sigma, \mathcal{A}, \mathcal{C}, \mathcal{S})$ possuam as seguintes propriedades:

[10]ou melhor, $\mathcal{CLP}(\mathcal{RT})$ dado que nas implementações se omite a verificação do Axioma A6 de $E_\mathcal{T}$, chamado habitualmente "occur check".

***Compacidade de soluções*:** Sendo I um conjunto de índices, possivelmente infinito:

$$\forall c \in \mathcal{C} \;\; \exists \{c_i\}_{i \in I} \in \mathcal{C} \quad \mathcal{A} \models \forall \neg c \;\Leftrightarrow\; \bigvee_{i \in I} c_i$$

***Completude para a satisfazibilidade*:** Definem-se noções mais fracas de axiomatização e de completude. $\mathcal{A}$ e $\mathcal{S}$ *correspondem* em $\mathcal{C}$, isto é:

(i) $\mathcal{A} \models \mathcal{S}$;

(ii) $\forall c \in \mathcal{C} \;\; \mathcal{A} \models \exists c$ se e só se $\mathcal{S} \models \exists c$.

$\mathcal{S}$ é *completo para a satisfação em relação a* $\mathcal{C}$, *i.e*,

$$\forall c \in \mathcal{C} \;\; \mathcal{S} \models \exists c \text{ ou } \mathcal{S} \models \neg \exists c$$

***Independência das Restrições Negativas*:** Para todas as c, $c_1, \ldots, c_n \in \mathcal{C}$ é verdade que

$$\mathcal{A} \models \exists\, (c \wedge \neg c_1 \wedge \ldots \wedge \neg c_n) \text{ se e só se } \mathcal{A} \models \exists\, (c \wedge \neg c_i) \text{ para } 1 \leq i \leq n$$

As duas primeiras garantem a completude e a integridade do esquema $\mathcal{CLP}$ [JL87]. A terceira permite simplificações nos testes de satisfazibilidade para restrições negativas e garante diversas propriedades nos modelos computacionais de $\mathcal{CLP}$ e que eram válidas em Programação em Lógica [Mah93].

Estas três propriedades são verificadas por $(\mathcal{L}, \mathcal{T})$ e por $(\mathcal{L}, \mathcal{RT})$. A segunda propriedade resulta das axiomatização completas que foram apresentadas para estas álgebras. A primeira e a terceira foram demonstradas por Maher em [Mah88], e serão estudadas para $\mathcal{LU}$ no Capítulo 4.

O problema fundamental a resolver num domínio de restrições é o da existência de soluções e a determinação de algoritmos completos eficientes para as calcular, que denominamos *resolutores* de restrições. Contudo em implementações de $\mathcal{CLP}$, surgem outros problemas relacionados com a satisfação, que permitem determinar relações entre restrições, simplificar restrições ou obter soluções parciais. Em resumo, alguns dos problemas a resolver são

Satisfazibilidade : Dada uma restrição c, determinar se $\mathcal{A} \models \exists c$

Vínculo: Dadas duas restrições c e c_1 determinar se $c \models_{\mathcal{A}} c_1$

Projecção: Dada uma restrição c e um conjunto de variáveis $X' \subseteq \mathsf{Vars}(c)$ determinar se existe uma restrição c', o mais simples possível, tal que $\mathcal{A} \models c' \;\Leftrightarrow\; \exists \bar{z}\; c$ e $\bar{z} \notin X'$

Determinação: Dada uma restrição c verificar se c *determina* uma variável $x \in \mathsf{Vars}(c)$, *i.e*, se x é *determinada* em c, ou ainda, se $\mathcal{A} \models \exists z \forall (c \Rightarrow x = z)$.

Equivalência: $c \approx_{\mathcal{A}} c'$

O último problema permite, por exemplo, simplificar restrições antes de aplicar um algoritmo de satisfazibilidade.

Suponhamos que $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$ e a linguagem é $\mathcal{L}$ ou um dos seus subconjuntos $\mathcal{LC}$, $\mathcal{LC}_0$ ou $\mathcal{LU}$. Recorde-se que $\mathcal{LU} \subset \mathcal{LC} \subset \mathcal{LC}_0 \subset \mathcal{L}$.

Pelo Teorema 2.34, para resolver o problema a satisfazibilidade em $\mathcal{LC}$ transforma-se esse problema num problema de $\mathcal{LU}$ e depois *projectam-se* as soluções nas variáveis de $\mathcal{LC}$, como será apresentado na Secção 7.7 do Capítulo 7.

Um problema relacionado com o da satisfazibilidade é o determinar se uma restrição c *vincula* c_1, $c_1 \models_{\mathcal{A}} c$. De acordo com a Proposição 2.15, se c_1 for satisfazível, não se pode ter simultaneamente $c_1 \models_{\mathcal{A}} c$ e $c_1 \models_{\mathcal{A}} \neg c$ e ainda, $c_1 \wedge c$ é não satisfazível se e só se $c_1 \models_{\mathcal{RT}} \neg c$. Portanto, determinar se c vincula c', pode reduzir-se a um problema de satisfazibilidade.

A *equivalência* entre restrições pode ser obtida mais facilmente se cada restrição tiver uma forma normal equivalente e essa forma for única. No Capítulo 5, abordaremos este último problema para $\mathcal{LU}$.

2.4 Sumário

As teorias $\mathcal{T}$ e $\mathcal{RT}$ das axiomatizações completas das álgebras de árvores sobre um conjunto de símbolos funcionais correspondem a linguagens de restrições complexas que designamos por $\mathcal{LC}_0$. Nestas linguagens, sem quantificadores, são usados caminhos (de símbolos funcionais) para a referência a sub-termos dum termo que evitam a quantificação existencial de certas variáveis. Este facto permite uma conveniente manipulação computacional das fórmulas dessas teorias.

Para estudar o problema da satisfazibilidade, mostramos que basta considerar um subconjunto de $\mathcal{LC}_0$, a linguagem $\mathcal{LU}$, onde apenas são considerados caminhos de tamanho um. Esta linguagem será estudada em detalhe no Capítulo 4.

Concluímos com uma breve revisão de outras linguagens de restrições e enquadramos as linguagens apresentadas nesse conjunto.

Capítulo 3

$\mathcal{RT}$ e Linguagens de Árvores de Atributos

Neste capítulo relacionamos a linguagem $\mathcal{L}$ interpretada em $\mathcal{RT}$ e as linguagens de árvores de atributos. Para pôr em evidência a similitude entre estas linguagens vamos utilizar a linguagem $\mathcal{LU}$, considerando por vezes restrições quantificadas.

Como foi mencionado no Capítulo 1, Secção 1.3.2.2, Smolka propôs em [Smo92] uma lógica de estruturas de atributos para a teorização matemática dos formalismos linguísticos baseados em unificação. Esta lógica, difere de outras existentes, pelo facto de que as descrições de estruturas de atributos são consideradas fórmulas de linguagens de lógica de 1ª ordem com igualdade e são interpretadas numa classe de álgebras. Isto tem a vantagem de importar dela, em particular, a completude dos seus sistemas dedutivos.

Estas linguagens são construídas a partir dum conjunto numerável de variáveis e duma assinatura que contém símbolos de predicado binários (atributos) e constantes. São definidas álgebras de atributos onde os predicados binários são interpretados como relações funcionais, as constantes distintas como objectos distintos e, ainda, onde as constantes não admitem atributos. Aos atributos associam-se predicados binários $f(x, y)$ que indicam que o objecto x tem para o atributo f o valor y, e representam-se normalmente por xfy. Um caso particular destas álgebras é a álgebra de grafos de atributos, onde o universo é o conjunto de grafos de atributos. Em [Smo92], Smolka apresenta uma axiomatização destas álgebras para aquelas linguagens, cujos axiomas coincidem com os Axiomas FT1-FT3, da Tabela 3.1. Smolka demonstra que a satisfazibilidade de fórmulas sem quantificadores é um problema decidível mas para fórmulas em geral, o problema é indecidível. Para o fragmento existencial, a álgebra de grafos de atributos é canónica no sentido em que uma fórmula é satisfazível numa qualquer álgebra de atributos se e só se o for na álgebra de grafos.

Esta axiomatização, embora capture o essencial das propriedades das estruturas de atributos, não é completa. Introduzindo novos axiomas e considerando como modelos canónicos álgebras de árvores de atributos, Smolka e outros apresentaram em [AKPS94, BS92, ST94, BS95, Bac94a] várias teorias completas para as linguagens de estruturas de atributos. Entre essas linguagens destacam-se a linguagem $\mathcal{FT}$ e linguagem $\mathcal{CFT}$, que estão na base de linguagens de programação em lógica por restrições como o LIFE [AKP93a] e o Oz [HSW93], respectivamente. A diferença entre estas duas linguagens é que a primeira é *intencional* e a segunda é *extensiva*. Isto é, em $\mathcal{CFT}$ é possível caracterizar completamente um objecto (do universo de interpretação) por uma fórmula, enquanto que em $\mathcal{FT}$ isso não é possível.

Como será apresentado na Secção 3.3, as teorias $\mathcal{CFT}$ e $\mathcal{RT}$ são equivalentes no sentido em que existem transformações entre as suas fórmulas tais que uma fórmula é válida numa das teorias se e só se a transformada é válida na outra. Estes factos permitem que $\mathcal{RT}$ seja tomado como modelo canónico para as teorias das descrições de estruturas de atributos em formalismos linguísticos, em particular nas gramáticas lógicas com restrições.

Do ponto vista prático, é indiferente usar qualquer uma destas duas teorias (linguagens) para as descrições das estruturas de atributos. Podemos considerar que as linguagens de estruturas de atributos são variantes sintácticas de $\mathcal{RT}$ adequadas para representar estruturas em linguagens de programação em lógica por restrições como LIFE [AKP93a] e Oz [HSW93] e, também, em formalismos linguísticos baseados em restrições. O estudo da satisfazibilidade em $\mathcal{LU}$ e os resolutores para $\mathcal{LU}$ que irão ser apresentados nos capítulos seguintes são adaptáveis directamente para $\mathcal{CFT}$, não só pela equivalência formal mas também pela similitude entre as fórmulas das duas linguagens.

Na Secção 3.1 definem-se as árvores de atributos e na Secção 3.2 as linguagens de 1ª ordem que são interpretadas nas álgebras de árvores de atributos. Esta apresentação baseia-se, fundamentalmente, em [BS95, Bac94a]. A Secção 3.3 constituí o essencial deste capítulo, onde se estabelece uma equivalência entre as linguagens $\mathcal{CFT}$ e $\mathcal{RT}$. Esta equivalência foi apresentada em [DMV94]. Na Secção 3.4 são descritas outras linguagens de árvores de atributos, evidenciando-se a diferença entre as álgebras de árvores de atributos e as de árvores sobre símbolos funcionais, mas notando que as propriedades de algumas destas linguagens não são interessantes para o trabalho apresentado nesta dissertação.

3.1 Árvores de Atributos

As *árvores de atributos* são árvores cujos arcos são etiquetados por *atributos* e as folhas por *constantes*[1]. Seja $\mathcal{FEA}$ um conjunto numerável de símbolos de predicado binários, denominados *atributos* (*features*) e $\mathcal{CON}$ um conjunto numerável de símbolos de *constantes*, disjunto de $\mathcal{FEA}$. Um *domínio de árvore* é um conjunto não vazio $D \subseteq \mathcal{FEA}^\star$ que é fechado para prefixos, *i.e*, se $pq \in D$ então $p \in D$, com p e q elementos (caminhos) de $\mathcal{FEA}^\star$. Dado um domínio de árvore D define-se $l(D)$ como o conjunto de caminhos maximais de D, *i.e*,

$$l(D) = \{p \in D \mid \forall f \in \mathcal{FEA} \quad pf \notin D\}$$

Uma *árvore de atributos* é um par $\sigma = (D, \lambda)$, onde D é um domínio de árvore e λ é uma função total $\lambda : l(D) \longrightarrow \mathcal{CON}$. Os caminhos em D representam os nós de σ; o caminho ϵ corresponde à raiz da árvore e λ representa a etiquetagem das folhas de σ (constantes). Uma árvore de atributos é *finita* (respectivamente *infinita*) se o seu domínio é finito (infinito). Um elemento $f \in \mathcal{FEA}$ é um atributo duma árvore de atributos $\sigma = (D, \lambda)$, se existe $p \in \mathcal{FEA}^\star$ tal que $pf \in D$. Uma sub-árvore $p^{-1}\sigma$ duma árvore de atributos $\sigma = (D, \lambda)$ é a árvore de atributos $\sigma' = (D', \lambda')$ definida por

$$D' = \{q \mid pq \in D\} \text{ e } \lambda' = \{(q, a) \mid (pq, a) \in \lambda\}$$

Uma árvore de atributos $\sigma = (D, \lambda)$ é *racional* se σ tem apenas um número finito de sub-árvores e para todo $p \in D$ o conjunto $\{f \in \mathcal{FEA} \mid pf \in D\}$ é finito. A *aridade* duma árvore de atributos σ é o conjunto de atributos dos arcos que ocorrem directamente da sua raiz, e designa-se por *aridade*(σ). Identifica-se $a \in \mathcal{CON}$ com a árvore $(\{\epsilon\}, \{(\epsilon, a)\})$, cuja *aridade* é $\emptyset$. Na Figura 3.1 apresenta-se um exemplo duma árvore de atributos.

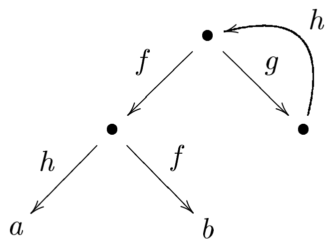


Figura 3.1: Exemplo duma árvore de atributos

[1] Esta formalização difere ligeiramente da apresentada em [AKPS94, ST94], onde cada nó é etiquetado com um *tipo*, mas é a formalização usada em [BS95, Bac94a].

Em particular, as árvores sobre um conjunto de símbolos funcionais são exemplos de árvores de atributos. Dada uma assinatura Σ, suponhamos que $\mathcal{FEA}$ contém o atributo 'fun' (o seu valor corresponderá a um functor) e os inteiros 1, 2, ..., e que $\mathcal{CON}$ contém uma constante f para cada $f^n \in \Sigma$. Para cada functor $f^n \in \Sigma$ considera-se uma árvore σ tal que $\text{fun} \in aridade(\sigma)$, $i \in aridade(\sigma)$ para $1 \leq i \leq n$ e $\lambda(\text{fun}) = f$. Na Figura 3.2 apresentam-se dois exemplos de árvores de atributos, que correspondem às árvores sobre $\{f^2, g^1, a, b\}$ apresentadas na Figura 2.1a) e b), no Capítulo 2.

No entanto, as árvores de atributos admitem, por exemplo, árvores com aridade infinita (um número infinito de sub-árvores directas) o que não é possível em árvores sobre um conjunto de símbolos funcionais.

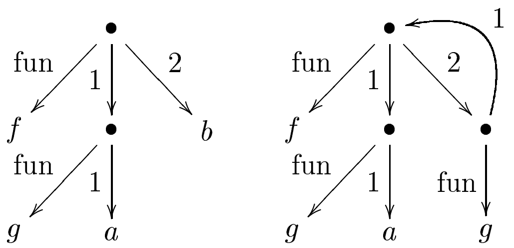


Figura 3.2: Árvores de atributos correspondentes a árvores sobre funcionais

3.2 Linguagens de Árvores de Atributos

As linguagens de árvores de atributos são linguagens de lógica de 1ª ordem com igualdade cuja assinatura contém símbolos de constantes, designados por a, b, ..., símbolos de predicados binários (atributos), designados por f, g, ... e, eventualmente, outros símbolos de predicado. Supõe-se um conjunto infinito de variáveis, x, y, ... O conjunto de variáveis livres duma fórmula ϕ designa-se por $\mathcal{V}(\phi)$.

Cada linguagem é interpretada numa álgebra $\mathcal{A}$ que tem como universo o conjunto das árvores de atributos e que é denominada *a álgebra canónica.*

O conjunto $\text{Th}(\mathcal{A})$ de fórmulas válidas nessa álgebra é considerado a teoria associada à linguagem. Neste sentido essa álgebra diz-se o *modelo canónico.*

Essa teoria e a linguagem são designamos pelo mesmo nome, assim como, o conjunto de axiomas que axiomatiza completamente essa teoria.

A linguagem $\mathcal{FT}$ [AKPS94] tem como assinatura $\mathcal{FEA} \cup \mathcal{CON}$. As fórmulas atómicas são as *restrições de atributos* xfy (que correspondem ao predicado $f(x, y)$, na notação habitual) e as *equações* $t = t'$, com t, t' variáveis ou constantes.

A álgebra de árvores de atributos $\mathfrak{T}_{\mathcal{FT}}$ é a $\mathcal{FT}$-álgebra[2] definida por:

(i) o universo $\mathcal{U}_{\mathfrak{T}_{\mathcal{FT}}}$ de $\mathfrak{T}_{\mathcal{FT}}$ é o conjunto das árvores de atributos sobre $\mathcal{FEA} \cup \mathcal{CON}$;

(ii) $a^{\mathfrak{T}_{\mathcal{FT}}} = (\{\epsilon\}, \{(\epsilon, a)\})$ para todo o símbolo constante $a \in \mathcal{CON}$;

(iii) para todo o $f \in \mathcal{FEA}$, $(\sigma, \tau) \in f^{\mathfrak{T}_{\mathcal{FT}}}$ se e só se $f \in aridade(\sigma)$ e $\tau = f^{-1}\sigma$.

A sub-álgebra constituída apenas por árvores racionais é denotada por $\mathfrak{R}_{\mathcal{FT}}$.

A Tabela 3.1 apresenta a teoria $\mathcal{FT}$ que constitui uma axiomatização completa de $\mathfrak{T}_{\mathcal{FT}}$ para o caso de $\mathcal{FEA}$ e $\mathcal{CON}$ serem conjuntos infinitos [BS92]. Os Axiomas FT1-FT3 correspondem aos apresentados para lógicas de estruturas de atributos em [Smo92]. O quarto axioma garante que certas descrições de atributos consistentes são satisfazíveis. Para tal, foi introduzida a noção de *cláusula resolvida*. Denota-se por $xf\uparrow$ ("f é indefinida em x") uma abreviatura de $\neg\exists y(xfy)$.

FT1. Para todo $f \in \mathcal{FEA}$, $\forall(xfy \wedge xfz \Rightarrow y = z)$

FT2. Para todo $a \in \mathcal{CON}$, $\forall(afx \Rightarrow \perp)$

FT3. $a \neq b$, se a e b são constantes distintas

FT4. Para toda a cláusula resolvida ϕ, $\forall\exists\mathcal{CV}(\phi)\phi$

Tabela 3.1: Axiomas de $\mathcal{FT}$

Definição 3.1 *Uma* cláusula resolvida *é uma conjunção ϕ, possivelmente vazia, de fórmulas atómicas da forma xft e $xf\uparrow$ tal que as seguintes condições são satisfeitas:*

(i) nenhuma fórmula ocorre duas vezes em ϕ;

(ii) não ocorrem fórmulas da forma aft ou $af\uparrow$ em ϕ;

(iii) se $xft \in \phi$ então não existe $t' \neq t$ tal que $xft' \in \phi$;

[2]Usamos aqui $\mathcal{FT}$ como abreviatura de $\mathcal{FEA} \cup \mathcal{CON}$.

(iv) se $x f t \in \phi$, *então* $x f \uparrow \notin \phi$.

Uma variável diz-se restringida *numa cláusula resolvida* ϕ, *se* ϕ *contém uma restrição da forma* $x f t$ *ou* $x f \uparrow$. *Designa-se por* $\mathcal{CV}(\phi)$ *o conjunto de todas as variáveis que são* restringidas *em* ϕ. *As variáveis de* $\mathcal{V}(\phi) \setminus \mathcal{CV}(\phi)$ *são denominadas* parâmetros *de* ϕ.

O Axioma FT4 afirma que as variáveis restringidas duma cláusula resolvida têm soluções para todos os valores dos parâmetros. Como a teoria é completa, resulta que $\mathfrak{T}_{\mathcal{FT}}$ e $\mathfrak{R}_{\mathcal{FT}}$ são modelos elementarmente equivalentes.

A teoria $\mathcal{FT}$ formaliza as propriedades essenciais das descrições de estruturas de atributos usadas em vários formalismos linguísticos e linguagens de programação.

Contudo, $\mathcal{FT}$ não é uma teoria *extensiva*, isto é, não é possível identificar um objecto do domínio por uma fórmula. É sempre possível adicionar atributos sem que a descrição se torne inconsistente. Por outro lado, a propriedade da *extensão* garante que duas estruturas de atributos são iguais se o são as suas descrições, isto é, se têm os mesmos atributos e, para um mesmo atributo, os mesmos valores. As teorias $E_{\mathcal{T}}$ e $E_{\mathcal{RT}}$, apresentadas no Capítulo 2, são extensivas no sentido em que dois termos são iguais se têm o mesmo functor e os mesmos sub-termos. Em particular, isto significa, que as linguagens de programação em lógica, como o Prolog, onde as estruturas de dados são os termos interpretados em $\mathcal{T}$ ou $\mathcal{RT}$, são, primordialmente[3] extensivas. Como foi discutido no Capítulo 1, em formalismos linguísticos esta propriedade pode também ser conveniente.

A linguagem $\mathcal{CFT}$ permite que a descrição de estruturas de atributos seja extensiva. A assinatura da linguagem $\mathcal{CFT}$ [ST94] é a da linguagem $\mathcal{FT}$ acrescida de mais um símbolo de predicado unário para todo o conjunto $F \subseteq \mathcal{FEA}$ de atributos, e que é denominado *aridade*. As fórmulas atómicas de $\mathcal{CFT}$ são as de $\mathcal{FT}$ e, também, as *restrições de aridade* do tipo xF. A restrição $x\emptyset$ é interpretada como uma constante.

A álgebra de árvores de atributos $\mathfrak{T}_{\mathcal{CFT}}$ é a $\mathcal{CFT}$-álgebra cujo universo, interpretação de símbolos de constantes e de símbolos de atributos são definidos como para $\mathfrak{T}_{\mathcal{FT}}$ e $\sigma \in F^{\mathfrak{T}_{\mathcal{CFT}}}$ se e só se $aridade(\sigma) = F$. A sub-álgebra constituída apenas por árvores racionais é denotada por $\mathfrak{R}_{\mathcal{CFT}}$.

A Tabela 3.2 apresenta a teoria $\mathcal{CFT}$ que constitui uma axiomatização completa de $\mathfrak{T}_{\mathcal{CFT}}$ para o caso de $\mathcal{FEA}$ e $\mathcal{CON}$ serem conjuntos infinitos [BS95]. Os Axiomas CFT4–CFT6 garantem que se x tem aridade F então um atributo f está definido para x se e só se $f \in F$. Para garantir a existência e a unicidade de soluções, foi introduzida a noção de determinante.

[3]Como já se viu, considerando argumentos extra podemos simular a *intencionalidade* com termos.

Definição 3.2 *Um* determinante *para x é uma fórmula da forma*

$$x\{f_1, \ldots, f_n\} \wedge xf_1t_1 \wedge \ldots \wedge xf_nt_n$$

onde cada t_i é uma variável ou uma constante. Abrevia-se a fórmula anterior para

$$x \doteq (f_1 : t_1, \ldots f_n : t_n)$$

O determinante *de variáveis distintas $x_1, \ldots, x_n$ é a conjunção dos determinantes de cada uma delas. Para uma conjunção de fórmulas atómicas ϕ define-se $\mathcal{D}(\phi)$ como o conjunto de variáveis que contêm um determinante em ϕ.*

O Axioma CFT7 afirma que cada determinante admite uma única solução para as suas variáveis determinadas.

A álgebra $\mathfrak{R}_{\mathcal{CFT}}$ também é um modelo de $\mathcal{CFT}$, logo, é elementarmente equivalente a $\mathfrak{T}_{\mathcal{CFT}}$.

É de realçar a similitude destes axiomas e os da teoria $E_{\mathcal{RT}}$ especialmente se usarmos fórmulas de $\mathcal{LU}$ para os exprimir.

Como no caso de árvores racionais sobre símbolos funcionais [Cou83], qualquer árvore de atributos racional τ é solução duma das variáveis duma fórmula ϕ_τ de $\mathcal{CFT}$. Para cada sub-árvore σ de τ seja x_σ uma variável distinta. Se $aridade(\sigma) = \emptyset$ então σ é uma constante a e $x_\sigma = a \in \phi_\tau$. Se $aridade(\sigma) = \{f_1, \ldots, f_n\}$ e $\sigma_i = f_i^{-1}\sigma$, o determinante $x_\sigma \doteq (f_1 : x_{\sigma_1}, \ldots f_n : x_{\sigma_n}) \in \phi_\tau$. Então, ou τ é uma constante ou τ é solução (única) para o determinante de x_τ.

CFT1. Para todo $f \in \mathcal{FEA}$, $\forall(xfy \wedge xfz \Rightarrow y = z)$

CFT2. Para todo $a \in \mathcal{CON}$, $\forall(afx \Rightarrow \perp)$

CFT3. $a \neq b$, se a e b são constantes distintas

CFT4. $\forall(xF \wedge xfy \Rightarrow \perp)$, se $f \notin F$

CFT5. Para todo $a \in \mathcal{CON}$ e F aridade, $aF \Rightarrow \perp$

CFT6. $\forall(xF \Rightarrow \exists y(xfy))$, se $f \in F$

CFT7. $\forall(\exists! D(\delta)\delta)$, se δ é um determinante

Tabela 3.2: Axiomas de $\mathcal{CFT}$

3.3 $\mathcal{RT}$ e $\mathcal{CFT}$

Nesta secção vamos mostrar que as teorias $\mathcal{CFT}$ e $\mathcal{RT}$ são equivalentes no sentido em que existem transformações efectivas, parametrizadas pelas assinaturas, tais que uma fórmula é válida numa das teorias se e só se a transformada é válida na outra. Este resultado, foi apresentado por L. Damas em [Dam91, DMV94]. Nesse sentido, podemos dizer que a teoria $\mathcal{CFT}$ não tem um poder expressivo maior que $\mathcal{RT}$. Em [ST94] é também demonstrado que $\mathcal{CFT}$ é uma extensão conservativa de $\mathcal{RT}$. Isto é, sendo Σ uma assinatura de símbolos funcionais define-se uma assinatura de Σ^F de $\mathcal{CFT}$ tal que existe uma tradução efectiva duma Σ-fórmula ϕ numa Σ^F-fórmula ϕ^F tal que $\mathcal{RT} \models \phi$ se e só se $\mathcal{CFT} \models \phi^F$.

Se as assinaturas forem finitas as duas linguagens têm a mesma expressividade. No entanto, em [ST94] argumenta-se que, considerando um número infinito de atributos, $\mathcal{CFT}$ é mais expressivo que $\mathcal{RT}$. Em particular, fórmulas do tipo

$$x\ f\ y$$

que indicam apenas que x tem o atributo f, e nada dizem sobre os restantes atributos, não podem ser expressas em $\mathcal{RT}$ se Σ for infinito. Contudo, como será visto a seguir, uma fórmula daquele tipo só será válida em $\mathcal{CFT}$ se o for numa álgebra com um número finito de atributos. E para estas álgebras é possível construir uma álgebra de árvores racionais "equivalente". Como uma fórmula de $\mathcal{FT}$ é uma fórmula de $\mathcal{CFT}$, o resultado estende-se a essa linguagem.

Em consequência, do ponto vista prático, é indiferente usar qualquer uma destas duas teorias (linguagens) para as descrições das estruturas de atributos e $\mathcal{RT}$ pode ser considerado como o modelo canónico para as teorias das descrições de estruturas de atributos.

No que se segue, identificaremos $\mathcal{RT}$ com o conjunto de axiomas $E_{\mathcal{RT}}$ excepto nos casos em que se pretenda expressamente referir àquele conjunto de fórmulas.

3.3.1 Modelos Finitos e Infinitos

Um primeiro resultado apresentado por L. Damas em [Dam91, DMV94] foi que, supondo um conjunto infinito de constantes, uma fórmula $\phi \in \mathcal{CFT}$ é válida numa álgebra de árvores (racionais) de atributos com um número infinito de atributos se e só se é válida para uma álgebra de árvores com um número finito de atributos. Na realidade, numa fórmula ϕ só podem ocorrer um número finito de atributos e não se pode referir de outro modo a atributos. Suponhamos que esses atributos são $f_1, \ldots, f_n$. Se uma solução de ϕ envolver outros atributos, estes podem ser "codificados" com a introdução de apenas mais um atributo f_{n+1} e pela identificação de cada um deles com uma constante distinta.

Este tipo de representação de estruturas de atributos é semelhante à usada em implementações de programação em lógica usando termos, e referida na Secção 1.3.2.3 do Capítulo 1. Para representar uma estrutura de atributos por termos, em vez de se considerar um termo da forma $f(V_1, \ldots, V_n)$ em que cada argumento corresponde a um valor dum atributo, podemos representá-la por um termo da forma $f(A, V, S)$ tal que A é o nome dum atributo, V é o seu valor e S ou é variável ou um termo da mesma forma que codifica os restantes atributos. Deste modo, podemos representar uma estrutura de atributos com um número não determinado de atributos. Em particular, se S é variável, é possível definir objectos "intencionalmente", correspondendo a fórmulas da linguagem $\mathcal{FT}$.

Analogamente, qualquer estrutura de atributos (árvore de atributos) pode ser transformada numa estrutura de atributos "equivalente" com um número fixo de atributos. Considere-se a estrutura de atributos

$$\boxed{1}\begin{bmatrix} f_1 & \boxed{2}\begin{bmatrix} f_3 & \boxed{2} \end{bmatrix} \\ f_4 & \begin{bmatrix} f_1 & a \\ f_5 & b \end{bmatrix} \\ f_5 & \begin{bmatrix} f_2 & \boxed{1} \end{bmatrix} \end{bmatrix}$$

Supondo que o conjunto de atributos disponíveis era $\{f_1, f_2\}$, podemos transformar a estrutura de atributos anterior considerando apenas mais um atributo f_3 e introduzindo constantes para identificar os restantes atributos: a_3, a_4 e a_5. A estrutura de atributos transformada seria

$$\boxed{1}\begin{bmatrix} f_1 & \boxed{2}\begin{bmatrix} f_3 & \begin{bmatrix} f_1 & a_3 \\ f_2 & \boxed{2} \end{bmatrix} \end{bmatrix} \\ f_3 & \begin{bmatrix} f_1 & a_4 \\ f_2 & \begin{bmatrix} f_1 & a \\ f_3 & \begin{bmatrix} f_1 & a_5 \\ f_2 & b \end{bmatrix} \end{bmatrix} \\ f_3 & \begin{bmatrix} f_1 & a_5 \\ f_2 & \begin{bmatrix} f_2 & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Para cada atributo f_i, com $i > 2$, associa-se o atributo f_1 cujo valor é a constante a_i e o atributo f_2 cujo valor é o valor do atributo f_i transformado. O atributo f_3 permite em cada "nível" codificar o resto da estrutura de atributos transformada. Esta transformação está na base do resultado que iremos apresentar nesta secção.

Antes de estabelecer esse resultado vamos estender as axiomatizações apresentadas na secção anterior para o caso do conjunto de atributos $\mathcal{FEA}$ ser finito (e $\mathcal{CON}$ infinito). Neste caso, para se obter uma axiomatização completa para álgebras de árvores de atributos é necessário mais um axioma, análogo ao Axioma do Domínio Fechado:

$$\forall x \bigvee_{F \subseteq \mathcal{FEA}} xF$$

Vamos denominar por $\mathcal{CFT}^{\text{fin}}$ a teoria constituída pelos Axiomas CFT1-CFT7 e pelo axioma anterior (para $\mathcal{FEA}$ finito).

Passamos a designar por $\mathfrak{T}^{\infty}_{\mathcal{CFT}}$ (ou $\mathfrak{R}^{\infty}_{\mathcal{CFT}}$) uma $\mathcal{CFT}$-álgebra tal que $\mathcal{FEA}$ é um conjunto infinito e, para todo $n > 0$, por $\mathfrak{T}^{n}_{\mathcal{CFT}}$ (ou $\mathfrak{R}^{n}_{\mathcal{CFT}}$) uma $\mathcal{CFT}$-álgebra tal que $\mathcal{FEA}$ é um conjunto com n elementos.

Proposição 3.3 *As álgebras $\mathfrak{T}^{n}_{\mathcal{CFT}}$ e $\mathfrak{R}^{n}_{\mathcal{CFT}}$ são modelos de $\mathcal{CFT}^{\text{fin}}$.*

Como já se disse, no que segue vamos restringir-nos às sub-álgebras $\mathfrak{R}_{\mathcal{CFT}}$ de árvores de atributos racionais.

A demonstração da proposição seguinte é uma adaptação da de [DMV94], Secção 2.2.

Proposição 3.4 *Seja $\phi \in \mathcal{CFT}$ e sejam $f_1, \ldots, f_n$ os atributos que ocorrem em ϕ. Então, $\mathfrak{R}^{\infty}_{\mathcal{CFT}} \models \phi$ se e só se $\mathfrak{R}^{n+1}_{\mathcal{CFT}} \models \phi$.*

Dem. Vamos construir funções injectivas H_0 e H_1 de $\mathcal{U}_{\mathfrak{R}^{n+1}_{\mathcal{CFT}}}$ para $\mathcal{U}_{\mathfrak{R}^{\infty}_{\mathcal{CFT}}}$ e vice-versa, tal que para $1 \leq i \leq n$ se $\tau = f_i^{-1}\tau'$ então $H_k(\tau) = f_i^{-1}H_k(\tau')$, para $k = 0, 1$. A função H_0 é definida como a inclusão de $\mathcal{U}_{\mathfrak{R}^{n+1}_{\mathcal{CFT}}}$ em $\mathcal{U}_{\mathfrak{R}^{\infty}_{\mathcal{CFT}}}$. Para a definição de H_1, associamos para cada atributo $f_{n+1}, f_{n+2}, \ldots$ uma constante distinta $a_1, a_2, \ldots$.

Para cada $\tau \in \mathcal{U}_{\mathfrak{R}^{\infty}_{\mathcal{CFT}}}$, a função H_1 é definida por:

(i) se $aridade(\tau) = \emptyset$ então τ é uma constante $(\{\epsilon\}, \{(\epsilon, a)\})$ e $H_1(\tau) = \tau$;

(ii) se $aridade(\tau) \neq \emptyset$ seja ϕ_τ a fórmula em $\mathcal{CFT}$ associada a τ e sejam $\sigma_1, \ldots, \sigma_s$ as suas sub-árvores.

Para cada sub-árvore σ de τ, se $aridade(\sigma) = \{f_{j_1}, \ldots, f_{j_k}, f_{n+i_1}, \ldots, f_{n+i_m}\}$, com $0 < j_1 < j_2 < \ldots < j_k \leq n$, $0 < i_1 < i_2 < \ldots < i_m$ e, para cada um destes índices g, $f_g^{-1}\sigma = \sigma_g$, temos que

$$x_\sigma \doteq (f_{j_1} : x_{\sigma_{j_1}}, \ldots, f_{j_k} : x_{\sigma_{j_k}}, f_{n+i_1} : x_{\sigma_{n+i_1}}, \ldots, f_{n+i_m} : x_{\sigma_{n+i_m}}) \in \phi_\tau$$

onde cada variável x_{σ_g} corresponde a uma sub-árvore σ_g de τ. Se σ_g é uma constante a temos que $x_{\sigma_g} = a \in \phi_\tau$.

Vamos transformar ϕ_τ numa fórmula $\phi_{\tilde{\tau}}$ tal que $\tilde{\tau} \in \mathcal{U}_{\mathfrak{R}^{n+1}_{\mathcal{CFT}}}$ e definimos $H_1(\tau) = \tilde{\tau}$. Para cada x_{σ_l}, para $1 \leq l \leq s$, seja $\tilde{x}_{\sigma_l}$ a variável correspondente em $\phi_{\tilde{\tau}}$. Definimos

$$\tilde{x}_\sigma \doteq (f_{j_1} : \tilde{x}_{\sigma_{j_1}}, \ldots, f_{j_k} : \tilde{x}_{\sigma_{j_k}}, f_{n+1} : y^{i_1}_\sigma) \in \phi_{\tilde{\tau}}$$

e

$$\begin{array}{lcll} y^{i_j}_\sigma & \doteq & (f_1 : a_{i_j}, f_2 : \tilde{x}_{\sigma_{n+i_j}}, f_{n+1} : y^{i_{j+1}}_\sigma) \in \phi_{\tilde{\tau}} & \text{para } j = 1, 2, \ldots, m-1 \\ y^{i_m}_\sigma & \doteq & (f_1 : a_{i_m}, f_2 : \tilde{x}_{\sigma_{n+i_m}}) \in \phi_{\tilde{\tau}} & \end{array}$$

onde, para cada σ, $y^{i_1}_\sigma, \ldots\ y^{i_m}_\sigma$ são variáveis novas.

Se $x_\sigma = a \in \phi_\tau$ então $\tilde{x}_\sigma = a \in \phi_{\tilde{\tau}}$. Resulta da definição, que $\phi_{\tilde{\tau}}$ tem uma única solução $\tilde{\tau}$ em $\mathfrak{R}^{n+1}_{\mathcal{CFT}}$.

Um exemplo da transformação H_1 é apresentado na Figura 3.3 e coincide com a transformação apresentada acima para estruturas de atributos.

Por indução na estrutura de ϕ prova-se que

$$\forall \rho \quad \mathfrak{R}^{n+1}_{\mathcal{CFT}} \models_\rho \phi \text{ se e só se } \mathfrak{R}^{\infty}_{\mathcal{CFT}} \models_{H_0(\rho)} \phi$$

e

$$\forall \rho \quad \mathfrak{R}^{\infty}_{\mathcal{CFT}} \models_\rho \phi \text{ se e só se } \mathfrak{R}^{n+1}_{\mathcal{CFT}} \models_{H_1(\rho)} \phi$$

sendo n escolhido adequadamente em cada caso. 2

Considerando os modelos canónicos podemos dizer que uma fórmula ϕ é válida em $\mathcal{CFT}$ se e só se for válida em $\mathcal{CFT}^{\text{fin}}$, para um conveniente conjunto de atributos $\mathcal{FEA}$.

3.3.2 Equivalência entre $\mathcal{RT}$ e $\mathcal{CFT}$

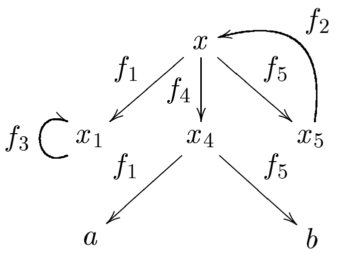
Nesta secção vamos apresentar transformações $.^c : \mathcal{CFT} \longrightarrow \mathcal{RT}$ e $.^f : \mathcal{RT} \longrightarrow \mathcal{CFT}$ tais que para qualquer fórmula ϕ

$$\mathcal{CFT} \models \phi \quad \text{se e só se} \quad \mathcal{RT} \models \phi^c$$

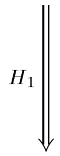
e

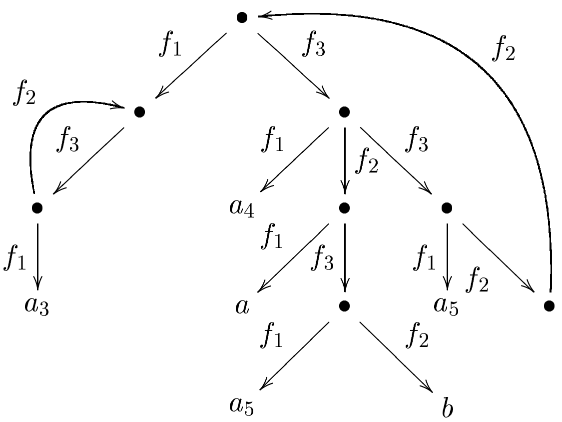
$$\mathcal{RT} \models \phi \quad \text{se e só se} \quad \mathcal{CFT} \models \phi^f$$

Para obter a equivalência entre as duas teorias, bastará considerar a primeira implicação em cada uma das equivalências acima.



$$x \doteq (f_1 : x_1, f_4 : x_4, f_5 : x_5) \;\wedge\; x_1 \doteq (f_3 : x_1) \;\wedge\; x_4 \doteq (f_1 : a, f_5 : b) \;\wedge\; x_5 \doteq (f_2 : x)$$





$$\begin{aligned}
&\tilde{x} \doteq (f_1 : \tilde{x}_1, f_3 : y^4) \;\wedge\; y^4 \doteq (f_1 : a_4, f_2 : \tilde{x}_4, f_3 : y^5) \;\wedge\; y^5 \doteq (f_1 : a_5, f_2 : \tilde{x}_5) \;\wedge\\
&\tilde{x}_1 \doteq (f_3 : y_1^3) \;\wedge\; y_1^3 \doteq (f_1 : a_3, f_2 : \tilde{x}_1) \;\wedge\\
&\tilde{x}_4 \doteq (f_1 : a, f_3 : y_4^5) \;\wedge\; y_4^5 \doteq (f_1 : a_5, f_2 : b) \;\wedge\\
&\tilde{x}_5 \doteq (f_2 : \tilde{x})
\end{aligned}$$

Figura 3.3: Exemplo da transformação H_1 para $n = 2$

Enquanto que a transformação $.^c$ depende de ϕ, isto é, dos atributos que ocorrem em ϕ e da assinatura de $\mathcal{CFT}$, a transformação $.^f$ apenas depende da assinatura. Variantes destas transformações foram apresentadas em [DMV94].

Seja $\mathcal{FEA} \cup \mathcal{CON}$ uma assinatura com $\mathcal{FEA} = \{f_1, \ldots, f_n\}$. Construímos uma assinatura Σ^c para $\mathcal{RT}$ constituída por $\mathcal{CON}$ e um conjunto finito de símbolos funcionais F, para cada subconjunto não vazio de $\{1, \ldots, n\}$. A aridade de cada símbolo F é igual à cardinalidade do conjunto F, e representa-se por n_F. A transformação $.^c$ é definida para as fórmulas de $\mathcal{CFT}$ como a extensão homomórfica de

$(xf_iy)^c$	$:=$	$\bigvee_{F\subseteq\{1,\ldots,n\}, i\in F}\ x.F^{n_F}_{i_F} \doteq y$	para $1 \leq i \leq n$ e onde i_F é o índice de i em F
$(x\emptyset)^c$	$:=$	$\bigwedge_{F\subseteq\{1,\ldots,n\}}\ \neg x.F^{n_F}$	
$(xF)^c$	$:=$	$x.F^{n_F}$	onde $F \not\equiv \emptyset$ é identificado com os subconjuntos de $\mathcal{FEA}$ e de $\{1, \ldots, n\}$ correspondentes
$(t = t')^c$	$:=$	$t = t'$	se t e t' são variáveis ou constantes

As imagens dos axiomas de $\mathcal{CFT}^{\text{fin}}$ por $.^c$ são Σ^c-fórmulas válidas em $\mathcal{RT}$. Podemos então concluir que $\mathcal{RT}$ é uma extensão de $\mathcal{CFT}^{\text{fin}}$, isto é, se ϕ for válida em $\mathcal{CFT}^{\text{fin}}$ então ϕ^c é válida em $\mathcal{RT}$.

A transformação $.^f$ pode ser definida com base na construção, apresentada na Secção 3.1, de árvores sobre símbolos funcionais como árvores de atributos.

Sendo Σ uma assinatura para $\mathcal{RT}$, seja $\Sigma^f = \mathcal{FEA} \cup \mathcal{CON}$, onde $\mathcal{FEA} = \{\text{fun}\} \cup \{1, 2, \ldots\}$ e $\mathcal{CON} = \Sigma$. Note-se que neste caso não é necessário considerar $\mathcal{FEA}$ um conjunto finito e a transformação $.^f$ não depende de ϕ.

A transformação $.^f$ é definida para as fórmulas de $\mathcal{LU}$ como a extensão homomórfica de

$(x.f^n_i \doteq v)^f$	$:=$	$x\,\text{fun}\,f\ \wedge\ x\{1, \ldots, n\}\ \wedge\ x\,i\,v$	
$(x.f^n)^f$	$:=$	$x\,\text{fun}\,f\ \wedge\ x\{1, \ldots, n\}$	
$(v = v')^f$	$:=$	$v = v'$	se v e v' são variáveis ou constantes

Usando $.^f$ podemos verificar que as imagens dos axiomas de $E_{\mathcal{RT}}$ (considerando as matrizes como fórmulas de $\mathcal{LU}$) são Σ^f-fórmulas válidas em $\mathcal{CFT}$. Concluímos que $\mathcal{CFT}$ é uma extensão de $\mathcal{RT}$, isto é, se ϕ for uma fórmula válida em $\mathcal{RT}$ então ϕ^f é válida em $\mathcal{CFT}$.

Como corolário destas transformações, podemos concluir que $\mathcal{CFT}^{\text{fin}}$ é completa uma vez que $\mathcal{RT}$ o é. É ainda possível decidir se uma fórmula $\phi \in \mathcal{CFT}$ é válida em $\text{Th}(\mathfrak{R}_{\mathcal{CFT}})$.

Com este corolário e usando a completude de $\mathcal{CFT}$, é possível construir mais explicitamente as equivalências anteriores. A apresentação que se segue é baseada em [ST94], generalizada para a transformação $.^c$.

Cada Σ^c-álgebra $\mathcal{A}$ pode ser traduzida numa $\mathcal{CFT}$-álgebra $\widehat{\mathcal{A}}$ considerando que:

(i) o universo de $\widehat{\mathcal{A}}$ é o de $\mathcal{A}$;

(ii) para todo $f_i \in \mathcal{FEA}$, $(\sigma, \sigma_i) \in f_i^{\widehat{\mathcal{A}}}$ se e só se existe $F^{\mathcal{A}}$ tal que

$$(\sigma_{n_1}, \ldots, \sigma_{i_F}, \ldots, \sigma_{n_F}, \sigma) \in F^{\mathcal{A}}$$

e $\sigma_{i_F} = \sigma_i$, onde i_F é o índice de i em F;

(iii) $\sigma \in F^{\widehat{\mathcal{A}}}$ se e só se $\sigma \in F^{\mathcal{A}}$

Se $\mathcal{A} \models E_{\mathcal{RT}}$ então para $\phi \in \mathcal{CFT}$, $\mathcal{A} \models \phi^c$ se e só se $\widehat{\mathcal{A}} \models \phi$ e $\widehat{\mathcal{A}} \models \mathcal{CFT}^{\text{fin}}$.

Proposição 3.5 *Para toda a fórmula $\phi \in \mathcal{CFT}$, $\mathcal{CFT} \models \phi$ se e só se $\mathcal{RT} \models \phi^c$.*

Dem. $(\Rightarrow)$

Seja ϕ tal que $\mathcal{CFT} \models \phi$ e sejam $f_1, \ldots, f_n$ os atributos que ocorrem em ϕ. Pela completude de $\mathcal{CFT}$ e pela Proposição 3.4, $\mathfrak{R}^{n+1}_{\mathcal{CFT}} \models \phi$. Então, por completude, $\mathcal{CFT}^{\text{fin}} \models \phi$. Como $\mathcal{RT} \models (\mathcal{CFT}^{\text{fin}})^c$ então $\mathcal{RT} \models \phi^c$.

$(\Leftarrow)$

Suponhamos que $\mathcal{RT} \models \phi^c$. Então, $\widehat{\mathcal{RT}} \models \phi$ e $\mathcal{CFT}^{\text{fin}} \models \phi$, por completude. Temos então que $\mathcal{CFT} \models \phi$. 2

Como no caso anterior é possível definir para cada Σ^f-álgebra $\mathcal{A}$ de $\mathcal{CFT}$, uma Σ-álgebra $\widehat{\mathcal{A}}$ tal que:

(i) o universo de $\widehat{\mathcal{A}}$ é o de $\mathcal{A}$;

(ii) para todo $f^n \in \Sigma$, $(\sigma_1, \ldots, \sigma_n, \sigma) \in f^{\widehat{\mathcal{A}}}$ se e só se $\sigma \in \{\text{fun}, 1, \ldots, n\}^{\mathcal{A}}$, $(\sigma, \sigma_f) \in \text{fun}^{\mathcal{A}}$ e σ_f é a constante f e $(\sigma, \sigma_i) \in i^{\mathcal{A}}$ para todo $1 \leq i \leq n$;

Se $\mathcal{A} \models \mathcal{CFT}$ então para toda a fórmula ϕ, $\mathcal{A} \models \phi^f$ se e só se $\widehat{\mathcal{A}} \models \phi$ e $\widehat{\mathcal{A}} \models E_{\mathcal{RT}}$.

Proposição 3.6 *Para toda a* Σ*-fórmula* ϕ, $\mathcal{RT} \models \phi$ *se e só se* $\mathcal{CFT} \models \phi^f$.

Dem. $(\Rightarrow)$

Seja ϕ tal que $\mathcal{RT} \models \phi$. Temos que $(E_{\mathcal{RT}})^f \models \phi^f$. Como $\mathcal{CFT} \models (E_{\mathcal{RT}})^f$, então $\mathcal{CFT} \models \phi^f$.

$(\Leftarrow)$

Suponhamos que $\mathcal{CFT} \models \phi^f$. Então, $\mathfrak{T}_{\mathcal{CFT}} \models \phi^f$. Como $\mathcal{RT}$ é completo e consistente, ou $\mathcal{RT} \models \phi$ ou $\mathcal{RT} \models \neg\phi$. Como $\widehat{\mathfrak{T}_{\mathcal{CFT}}} \models \phi$ temos que $\mathcal{RT} \models \phi$. 2

3.4 Outras Linguagens de Árvores de Atributos

As linguagens de árvores de atributos podem ser generalizadas para permitir quantificar sobre atributos, isto é, ter variáveis que denotem atributos. Para isso, as restrições de atributos xfy são generalizadas para predicados ternários, com símbolo de predicado $\cdot[\cdot]\cdot$, onde $x[z]y$ representa o facto de y ser uma sub-árvore de x no atributo z. A linguagem de atributos $\mathcal{F}$ definida em [Tre93, Bac94a] tem como assinatura símbolos constantes (que podem ser interpretados como atributos ou constantes) e um único símbolo de predicado ternário $\cdot[\cdot]\cdot$. A álgebra de árvores de atributos $\mathfrak{T}_{\mathcal{F}}$ é a $\mathcal{F}$-álgebra definida por:

(i) o universo de $\mathfrak{T}_{\mathcal{F}}$, $\mathcal{U}_{\mathfrak{T}_{\mathcal{F}}}$, é o conjunto das árvores de atributos sobre $\mathcal{FEA} \cup \mathcal{CON}$

(ii) $a^{\mathfrak{T}_{\mathcal{F}}} = (\{\epsilon\}, \{(\epsilon, a)\})$ para todo o símbolo constante $a \in \mathcal{CON} \cup \mathcal{FEA}$

(iii) $(\sigma_1, \sigma_2, \sigma_3) \in \cdot[\cdot]\cdot^{\mathfrak{T}_{\mathcal{F}}}$ se e só se existe $f \in \mathcal{FEA}$ tal que $\sigma_2 = f^{\mathfrak{T}_{\mathcal{F}}}$ e $\sigma_3 = f^{-1}\sigma_1$

A álgebra $\mathfrak{T}_{\mathcal{F}}$ é um modelo canónico para $\mathcal{F}$. Nesta linguagem, é possível, a quantificação sobre as sub-árvores duma árvore. Em particular, os modelos das árvores infinitas e das árvores racionais não são equivalentes, uma vez que é possível exprimir o facto de uma árvore ter um número infinito de filhos (sub-árvores directas). A expressividade de $\mathcal{F}$ é obviamente muito maior do que a das linguagens anteriores, sendo estas equivalentes por definições a fragmentos de $\mathcal{F}$ [Tre93, Bac94a]. A linguagem $\mathcal{F}$, permite a definição de relações recursivas sobre estruturas de atributos, suficientes para os formalismos linguísticos autónomos baseados em unificação [Bac94a]. No entanto, o facto da teoria correspondente ser indecidível [Tre93] torna-a, de acordo com os objectivos definidos no Capítulo 1, desinteressante para o trabalho apresentado nesta tese. Em [Tre93], Treinen apresenta uma axiomatização de $\mathcal{F}$ que é completa para a satisfazibilidade de fórmulas sem quantificadores, e portanto, este fragmento de $\mathcal{F}$ é decidível. Em particular, decidir a satisfazibilidade do fragmento de conjunções de fórmulas atómicas positivas é um problema **NP**-completo.

Em [Bac94a], Backofen apresenta ainda a linguagem $\mathcal{RFT}$ que permite expressões regulares sobre $\mathcal{FEA}$. Usando esta linguagem é possível especificar propriedades de sub-árvores cujo caminho pertence a um conjunto regular. Como vimos no Capítulo 1, este tipo de expressões foi introduzido por Kaplan [KI88] em formalismos linguísticos, para tratar fenómenos de longa-distância. Backofen [Bac94a] provou que a teoria associada a $\mathfrak{R}_{\mathcal{FT}}$ é indecidível, mas o fragmento existencial de conjunções de fórmulas atómicas positivas é decidível.

Mais recentemente, Treinen em [Tre95] apresenta uma linguagem entre $\mathcal{F}$ e $\mathcal{CFT}$ que é decidível. Esta linguagem admite a quantificação sobre atributos (e aridades) mas não sobre sub-árvores. Uma fórmula do tipo $xf\downarrow$ indica que o atributo f está definido para x, podendo f ser uma variável. Estabelece-se assim uma relação entre conjuntos (finitos) de atributos (conjunto infinito) e fórmulas duma linguagem de árvores de atributos. Generalizando esta correspondência, Treinen apresenta uma transformação, por eliminação de quantificadores, de fórmulas da linguagem de árvores de atributos em fórmulas equivalentes da teoria de atributos associada. Se esta última teoria for decidível, a correspondente teoria de árvores de atributos também o é. No caso anterior, essa teoria é a teoria dos conjuntos finitos sobre um alfabeto infinito, que é decidível [Büc60].

3.5 Sumário

Existe uma tradução efectiva entre fórmulas de $\mathcal{RT}$ e da linguagem de árvores de atributos $\mathcal{CFT}$, de tal modo que para a validade de fórmulas é indiferente qual das duas teorias se use (para descrições de estruturas de atributos). A similitude entre as fórmulas de $\mathcal{LU}$ e de $\mathcal{CFT}$ permite ainda que os resolutores que vão ser apresentados para $\mathcal{LU}$ sejam facilmente adaptados para $\mathcal{CFT}$.

Capítulo 4

Satisfazibilidade em $\mathcal{LU}$

Neste capítulo vamos estudar a satisfazibilidade de $\mathcal{LU}$ nas álgebras de árvores, isto é, dada uma restrição $c \in \mathcal{LU}$ determinar se c é satisfazível e obter o seu conjunto de soluções.

Como exposto nas Secções 2.2.1 e 2.2.2 do Capítulo 2, este problema é decidível. Resulta da decidibilidade de $\mathcal{T}$ e de $\mathcal{RT}$, da equivalência de $\mathcal{LC}_0$ e $\mathcal{L}$ e de que $\mathcal{LU} \subset \mathcal{LC}_0$. No entanto, como veremos, é um problema **NP**-completo. Um resultado equivalente foi apresentado por Maher [Mah] e por Smolka [Smo89a], para as lógicas de estruturas de atributos. Um estudo da complexidade computacional das teorias $E_{\mathcal{RT}}$ e $E_{\mathcal{T}}$ foi feito por Maher [Mah], como complemento a [Mah88], onde se prova que estas teorias são **PSPACE**-completas.

O carácter exponencial dos algoritmos para decidir a satisfazibilidade de $\mathcal{LU}$, reside na transformação de fórmulas em forma normal disjuntiva (DNF), as quais podem ter um tamanho exponencial em relação ao tamanho da fórmula inicial. O nosso objectivo será apresentar, no Capítulo 6, resolutores de restrições em $\mathcal{LU}$ que tentam reduzir o custo desta transformação. Para tal serão usadas as caracterizações de restrições obtidas neste capítulo.

Neste e, nos restantes capítulos, vamos considerar essencialmente a satisfazibilidade e outras propriedades de $\mathcal{LU}$ na álgebra das árvores racionais $\mathcal{RT}$. Portanto, por vezes, será omitida a referência a essa álgebra. Esta escolha baseia-se, como exposto no Capítulo 3, no facto de considerarmos $\mathcal{RT}$ o modelo canónico das descrições das estruturas de atributos.

No entanto, como será referido em alguns casos, as diversas formas normais e o *Calculus* de restrições apresentado adaptam-se facilmente para a álgebra das árvores finitas e, mesmo, para as álgebras de árvores de atributos.

Sempre que necessário será feita a distinção entre os casos da assinatura Σ ser infinita ou finita.

Como no caso dos problemas de unificação, na Secção 4.1 vamos ver que as restrições em forma resolvida caracterizam as soluções duma conjunção de restrições atómicas positivas. Contudo, estas formas não garantem a igualdade entre duas variáveis que tenham as mesmas soluções. Para tal, uma "normalização" será introduzida na Secção 4.2 que permitirá, na Secção 4.3, a generalização das formas resolvidas para englobar conjunções de restrições atómicas negativas a que chamaremos formas primárias. Na Secção 4.3.1 serão estudadas formas primárias para uma assinatura infinita e na Secção 4.3.2 para uma assinatura finita. Neste último caso, na Secção 4.3.2.1, analisamos o problema de determinar se uma conjunção de restrições atómicas é equivalente a uma disjunção de conjunções de restrições atómicas positivas. Finalmente, a Secção 4.4 apresenta a caracterização de soluções para restrições em geral. Para cada tipo de restrições, serão ainda apresentadas algumas propriedades das formas "normais" encontradas.

Em geral, estaremos interessados em soluções duma restrição c confinadas a um conjunto finito de variáveis V tal que $\mathsf{Vars}(c) \subseteq V$,

$$\mathsf{Sol}_V(c, \mathcal{RT}) = \{\rho_{|V} \mid \mathcal{RT} \models_\rho c\}$$

A comutatividade e a associatividade da conjunção e da disjunção preservam, em qualquer álgebra $\mathcal{A}$, a satisfazibilidade das restrições de $\mathcal{LU}$. Seja $\equiv_\mathcal{B}$ a menor relação de equivalência definida em $\mathcal{LU}$, tal que

$$\begin{aligned}
c_1 \ \wedge \ c_2 \ &\equiv_\mathcal{B} \ c_2 \ \wedge \ c_1 \\
c_1 \ \wedge \ (c_2 \ \wedge \ c_3) \ &\equiv_\mathcal{B} \ (c_1 \ \wedge \ c_2) \ \wedge \ c_3 \\
c_1 \ \vee \ c_2 \ &\equiv_\mathcal{B} \ c_2 \ \vee \ c_1 \\
c_1 \ \vee \ (c_2 \ \vee \ c_3) \ &\equiv_\mathcal{B} \ (c_1 \ \vee \ c_2) \ \vee \ c_3
\end{aligned}$$

Salvo indicação em contrário, serão consideradas restrições a menos da relação $\equiv_\mathcal{B}$. Este facto permite-nos tratar as conjunções de restrições como multiconjuntos. A restrição $\top$, como conjunção vazia, corresponde ao multiconjunto vazio e a conjunção à reunião de multiconjuntos. A expressão $c \in c'$ denota que c' é da forma $c \ \wedge \ c''$, isto é, que c é um elemento (ou sub-conjunto!) do multiconjunto c'. Em geral, os multiconjuntos serão denotados por $\mathcal{C}$, $\mathcal{C}'$, $\mathcal{C}_1$, ...

A disjunção vazia corresponderá à restrição $\perp$. Por vezes, como é usual, designaremos a conjunção por multiplicação e a disjunção por adição.

Considerando ainda que a idempotência da conjunção, $c \wedge c \equiv c$, também preserva, em qualquer álgebra $\mathcal{A}$, a satisfazibilidade das restrições, certas conjunções de restrições atómicas são tratadas como conjuntos.

Como indicado na definição da linguagem $\mathcal{LU}$, para simplificar a notação, evitando índices, nos exemplos de restrições de $\mathcal{LU}$ usaremos para além de x, y e z, as letras u, w, s e t para denotar variáveis. Sempre que não especificado doutro modo, uma *restrição* será um elemento de $\mathcal{LU}$.

4.1 Formas Resolvidas

Definição 4.1 *Uma conjunção $\mathcal{C}$ de restrições atómicas está em (ou é uma)* forma resolvida *se é $\top$ ou se satisfaz as seguintes condições:*

(i) toda a restrição em $\mathcal{C}$ é do tipo $x.f^n$, $x.f_i^n \doteq v$ ou $x = v$ e só ocorre uma vez em $\mathcal{C}$;

(ii) se $x = v \in \mathcal{C}$ então x ocorre exactamente uma vez em $\mathcal{C}$;

(iii) se $x.f_i^n \doteq v \in \mathcal{C}$ então não existe em $\mathcal{C}$ outra restrição $x.f_i^n \doteq v'$;

(iv) se $x.f_i^n \doteq v \in \mathcal{C}$ então $x.f^n \in \mathcal{C}$;

(v) se $x.f^n \in \mathcal{C}$ então não existe em $\mathcal{C}$ nenhuma outra restrição do tipo $x.g^m$;

Proposição 4.2 *Se $\mathcal{C}$ é uma forma resolvida, então $\mathcal{C}$ é satisfazível em $\mathcal{RT}$.*

Dem. Vamos provar que uma restrição em forma resolvida é equivalente a uma fórmula básica racional de $\mathcal{L}$, ver Teorema 2.32, e portanto é satisfazível. Iremos ainda mostrar que se pode construir uma solução para $\mathcal{C}$, directamente a partir duma forma resolvida.

Recordemos que uma fórmula básica é fórmula resolvida racional onde certas variáveis estão existencialmente quantificadas. Seja $\mathcal{C}$ uma forma resolvida. Construímos a fórmula $\phi \in \mathcal{L}$ do seguinte modo:

(i) se $x = v \in \mathcal{C}$ então $x = v \in \phi$;

(ii) se $y.f^n \in \mathcal{C}$ então $\exists \bar{z}.\ y = t \in \phi$ onde, t é um termo tal que, visto como uma árvore, $t(\epsilon) = f^n$ e $t(i) = v_i$ se $y.f_i^n \doteq v_i \in \mathcal{C}$ ou, caso contrário, $t(i) = z_i$ onde z_i é uma variável nova e $z_i \in \{\bar{z}\}$;

Seja $\hat{\phi}$ a forma prenexa normal de ϕ. Pela definição das restrições e pelo Lema 2.27, temos que $\hat{\phi} \approx_{\mathcal{RT}} \mathcal{C}$. As variáveis que aparecem do lado esquerdo de cada equação são todas distintas e se $x = v \in \phi$ então x não ocorre mais em ϕ. Estas condições garantem que a conjunção de equações (matriz) de $\hat{\phi}$, que designaremos por $\phi_{\mathcal{C}}$, está em forma resolvida racional e que $\hat{\phi}$ é uma fórmula básica (racional).

Alternativamente, podemos construir a partir duma forma resolvida $\mathcal{C}$, uma atribuição de variáveis ρ tal que $\mathcal{RT} \models_\rho \mathcal{C}$. Sendo[1] $a \in \Sigma$, ρ é definida recursivamente por:

(i) $\rho(x) = a$ para toda a variável $x \notin \mathsf{Vars}(\mathcal{C})$;

(ii) $\rho(x) = a$ para toda a variável $x \in \mathsf{Vars}(\mathcal{C})$ e tal que não existe para x em $\mathcal{C}$ nenhuma restrição do tipo $x = v$ ou $x.f^n$;

(iii) $\rho(x) = \rho(v)$ para toda a restrição $x = v \in \mathcal{C}$;

(iv) se $x.f^n \in \mathcal{C}$ então $\rho(x) = t_x$ onde t_x é uma árvore racional tal que $t_x(\epsilon) = f^n$ e para todo $1 \leq i \leq n$, se $x.f_i^n \doteq v_i \in \mathcal{C}$ então $t_x(i) = \rho(v_i)$, ou, caso contrário, $t_x(i) = a$

Para cada variável x, a atribuição $\rho(x)$ está bem definida e é uma árvore racional porque é finito o número de variáveis que verificam as condições (iii) e (iv). Temos que $\mathcal{RT} \models_\rho \mathcal{C}$. 2

As formas resolvidas serão designadas por $\mathcal{M}$, $\mathcal{M}'$, $\mathcal{M}_0 \ldots$. Estas letras também designaram, por vezes, explicitamente a restrição $\perp$.

Como na demonstração da proposição anterior, para qualquer forma resolvida $\mathcal{M}$ podemos definir recursivamente uma substituição $\theta_{\mathcal{M}}$ em $\mathcal{RT}(\Sigma \cup X)$, do seguinte modo:

$$
\begin{array}{lll}
x\theta_{\mathcal{M}} = v\theta_{\mathcal{M}} & \text{se } x = v \in \mathcal{M} & \\
x\theta_{\mathcal{M}} = f(t_1, \ldots, t_n) & \text{se } x.f^n \in \mathcal{M} \text{ e } t_i = & \left\{ \begin{array}{ll} v_i\theta_{\mathcal{M}} & \text{se } x.f_i^n \doteq v_i \in \mathcal{M} \\ z_i & \text{caso contrário e } z_i \notin \mathsf{Vars}(\mathcal{M}) \end{array} \right. \\
x\theta_{\mathcal{M}} = x & \text{caso contrário} &
\end{array}
$$

Pelo Axioma $\mathsf{A6}$ de $E_{\mathcal{RT}}$, a substituição $\theta_{\mathcal{M}}$ está bem definida e é única a menos de renomeação das variáveis $\bar{z}$. Para toda a atribuição de variáveis ρ em $\mathcal{RT}$, $\rho\theta_{\mathcal{M}}$ é uma solução de $\mathcal{M}$, e, inversamente, qualquer solução ρ de $\mathcal{M}$ em $\mathcal{RT}$ verifica $\rho = \rho\theta_{\mathcal{M}}$.

Definição 4.3 *Se $\mathcal{C}$ é uma forma resolvida e $x = v \in \mathcal{C}$ diz-se que x é uma* variável resolvida *em $\mathcal{C}$.*

[1] Se Σ não contiver constantes, sendo $f^n \in \Sigma$ denota-se por a a árvore racional tal que $a(\alpha) = f^n$ para todas as posições α.

Vamos ver que uma conjunção de restrições atómicas positivas satisfazível pode sempre ser transformada numa forma resolvida equivalente. As regras de reescrita a seguir apresentadas são baseadas nas originalmente usadas por Herbrand [Her30] para a resolução de equações na álgebra das árvores finitas e retomadas por Robinson no algoritmo da unificação de termos [Rob65]. Correspondem às regras de simplificação dadas por Colmerauer em [Col82] para a obtenção de formas resolvidas racionais e são, ainda, análogas às apresentadas por Smolka em [Smo89a, ST92], para estruturas de atributos.

Proposição 4.4 *Qualquer conjunção $\mathcal{C}$ de restrições atómicas positivas pode ser transformada numa restrição $\mathcal{M}$ tal que $\mathcal{C} \approx_{\mathcal{RT}} \mathcal{M}$, pela aplicação do sistema de reescrita* R*, constituído pelas seguintes regras:*

R1. $\{\top\} \cup \mathcal{C} \mapsto \mathcal{C}$

R2. $\{v = v\} \cup \mathcal{C} \mapsto \mathcal{C}$

R3. $\{a = b\} \cup \mathcal{C} \mapsto \bot$

R4. $\{x = v\} \cup \mathcal{C} \mapsto \{x = v\} \cup \mathcal{C}[v/x]$ *se x é distinto de v e $x \in \mathsf{Vars}(\mathcal{C})$*

R5. $\{a = x\} \cup \mathcal{C} \mapsto \{x = a\} \cup \mathcal{C}$

R6. $\{x.f_i^n \doteq v, x.f_i^n \doteq v'\} \cup \mathcal{C} \mapsto \{x.f_i^n \doteq v', v = v'\} \cup \mathcal{C}$

R7. $\{a.f_i^n \doteq v\} \cup \mathcal{C} \mapsto \bot$

R8. $\{a.f^n\} \cup \mathcal{C} \mapsto \bot$

R9. $\{x.g^m, x.f^n\} \cup \mathcal{C} \mapsto \bot$

R10. $\{x.f^n, x.f^n\} \cup \mathcal{C} \mapsto \{x.f^n\} \cup \mathcal{C}$

R11. $\{x.f_i^n \doteq v\} \cup \mathcal{C} \mapsto \{x.f^n, x.f_i^n \doteq v\} \cup \mathcal{C}$ *se $x.f^n \notin \mathcal{C}$*

O sistema de reescrita R *termina e se $\mathcal{M}$ é irredutível para este conjunto de regras e $\mathcal{M} \not\equiv \bot$ então $\mathcal{M}$ está em forma resolvida.*

Dem. As regras R1, R2, R4 e R5 são correctas em qualquer álgebra de $\mathcal{L}$. A correcção das regras R3, R6, R7, R8, R9, R10, R11 resulta dos axiomas de $\mathcal{RT}$ e das definições das restrições. Analisemos a terminação. Nenhuma regra faz aumentar o número de variáveis. A aplicação das regras R3, R7, R8 ou R9 causa a terminação. A regra R1 só se aplica se inicialmente $\top \in \mathcal{C}$ e só uma vez para cada uma dessas ocorrências.

A regras R2 e R10 fazem diminuir o número de restrições. A regra R6 faz diminuir o número de restrições do tipo $x.f_i^n \doteq v$, portanto a sua aplicação é finita (sem a aplicação da regra R4). Se a regra R11 se aplica mais que uma vez para a mesma variável então a terminação resulta da aplicação da regra R9. As regras R2, R5, R6, R10 e R11 só se aplicam um número finito de vezes antes da aplicação da regra R4 e, portanto se esta regra não se aplicar o processo termina. A regra R4 só pode ser aplicada no máximo uma vez para cada variável. Donde, a regra aplica-se um número finito de vezes, garantindo a terminação. Finalmente, falta provar que as restrições irredutíveis, diferentes de $\perp$, estão em forma resolvida. Se a primeira condição não se verifica, a restrição não é irredutível: se uma restrição atómica for do tipo $v.f^n$ ou $v.f^n \doteq v'$ e v não é uma variável então a regra R8 ou a regra R7 aplica-se; se for do tipo $v = v'$ e v não for variável então uma das regras R2, R3 ou R5 aplicam-se. Analogamente, prova-se que se alguma das outras condições não se verifica, a restrição não é irredutível. 2

Sendo $\mathcal{C}$ uma conjunção de restrições atómicas positivas, N o número de variáveis de $\mathcal{C}$ e M o número de restrições atómicas de $\mathcal{C}$, o sistema de reescrita R tem complexidade, no pior[2] caso, da ordem $O(NM)$.

Corolário 4.5 *Uma conjunção $\mathcal{C}$ de restrições atómicas positivas é satisfazível em $\mathcal{RT}$ se e só se $\mathcal{C} \overset{\mathsf{R}}{\mapsto}^\star \mathcal{M}$, $\mathcal{M}$ é irredutível para o sistema de reescrita* R *e* $\mathcal{M} \not\equiv \perp$.

Aplicando o sistema de reescrita R à restrição

$$(\mathsf{c}_1) \qquad z.f_1^4 \doteq b \ \wedge \ z = y \ \wedge \ z = x \ \wedge \ w = b \ \wedge \ x.f_1^4 \doteq u \ \wedge \ x.f_1^4 \doteq b$$

obtemos a forma resolvida

$$(\mathsf{c}_2) \qquad x.f^4 \ \wedge \ z = x \ \wedge \ y = x \ \wedge \ w = b \ \wedge \ x.f_1^4 \doteq b \ \wedge \ u = b$$

A definição seguinte será usada na caracterização de conjunções de restrições atómicas não satisfazíveis.

Definição 4.6 *Uma restrição atómica positiva diz-se* contraditória *se é do tipo $a = b$, com $a \not\equiv b$, $a.f^n$ ou $a.f_i^n \doteq v$. A restrição $\perp$ também é designada de* contraditória.

Uma restrição atómica contraditória não é satisfazível em $\mathcal{RT}$.

[2] Na prática pode-se implementar um algoritmo da ordem $O(N \log N)$ [Jaf84].

4.2 Formas Resolvidas Normais

Uma forma resolvida $\mathcal{M}$ define uma relação de equivalência $=_{\mathcal{M}}$ entre as suas variáveis tal que $x =_{\mathcal{M}} y$ se $x = y \in \mathcal{M}$ e $=_{\mathcal{M}}$ é a menor relação de equivalência que verifica esta condição. Resulta da Proposição 4.4 que

Corolário 4.7 *Se $\mathcal{C}$ é uma conjunção de restrições atómicas positivas e $\mathcal{M}$ é uma forma resolvida de $\mathcal{C}$, irredutível para o sistema de reescrita* R*, então $\mathcal{M}$ é única a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{M}}$.*

No entanto, duas formas resolvidas distintas podem ser equivalentes em $\mathcal{RT}$.

Por exemplo, as restrições[3] c_3 e c_4 são equivalentes em $\mathcal{RT}$:

$$(\mathsf{c}_3) \qquad y.f^1 \;\wedge\; y.f_1^1 \doteq y \;\wedge\; x = y$$

$$(\mathsf{c}_4) \qquad x.f^1 \;\wedge\; y.f^1 \;\wedge\; x.f_1^1 \doteq y \;\wedge\; y.f_1^1 \doteq x$$

Em ambos os casos, a solução das variáveis x e y é árvore racional cuja representação gráfica é:

Por outro lado, numa forma resolvida duas variáveis podem corresponder à mesma árvore sem que essa identidade seja obtida na redução a forma resolvida.

A restrição $x = y$ é consequência de (é vinculada por) cada uma das restrições em forma resolvida

$$(\mathsf{c}_5) \qquad x.f^1 \;\wedge\; y.f^1 \;\wedge\; x.f_1^1 \doteq x \;\wedge\; y.f_1^1 \doteq y$$

$$(\mathsf{c}_6) \qquad x.f^1 \;\wedge\; y.f^1 \;\wedge\; x.f_1^1 \doteq a \;\wedge\; y.f_1^1 \doteq a$$

sendo a restrição c_5 também equivalente em $\mathcal{RT}$ às restrições c_3 e c_4.

À restrição c_6 podemos associar a seguinte representação gráfica:

$$\begin{array}{ccc} [x] & \qquad & [y] \\ \big\downarrow{\scriptstyle f_1^1} & & \big\downarrow{\scriptstyle f_1^1} \\ a & & a \end{array}$$

[3]Passaremos a identificar a referência da fórmula com a denotação da restrição correspondente.

Para esta restrição, a única solução das suas variáveis é a árvore racional cuja representação gráfica é:

$$\begin{tikzcd} f \arrow[d] \\ a \end{tikzcd}$$

No entanto, essa igualdade não pode ser deduzida pelo sistema de reescrita R, apresentado na Proposição 4.4.

Note-se que este facto não é relevante para a satisfazibilidade de conjunções de restrições atómicas positivas, mas é-o para a satisfazibilidade de restrições que envolvam negações.

Assim, o facto da conjunção da restrição c_6 com a restrição $x \neq y$ ser não satisfazível não é "imediato" da restrição c_6, mas torna-se explícito na restrição

$$(\mathsf{c}_7) \qquad x.f^1 \ \wedge \ x.f_1^1 \doteq a \ \wedge \ x = y$$

que é equivalente em $\mathcal{RT}$ à restrição c_6. A esta restrição podemos associar a seguinte representação gráfica:

$$\begin{tikzcd} {[x, y]} \arrow[d, "f_1^1"] \\ a \end{tikzcd}$$

Nesta secção iremos considerar formas resolvidas $\mathcal{M}$ tal que, se a igualdade de duas variáveis é vinculada por $\mathcal{M}$, então essas variáveis pertencem à mesma classe de equivalência de $=_{\mathcal{M}}$.

Lema 4.8 *Se $\mathcal{M}$ e $\mathcal{M}'$ são formas resolvidas tais que $\mathcal{M} \approx_{\mathcal{RT}} \mathcal{M}'$ então*

$$\mathsf{Vars}(\mathcal{M}) = \mathsf{Vars}(\mathcal{M}')$$

Dem. Provamos por redução ao absurdo. Suponhamos que $x \in \mathsf{Vars}(\mathcal{M})$ e $x \notin \mathsf{Vars}(\mathcal{M}')$. Temos os seguintes casos:

(i) se $x = v \in \mathcal{M}$, seja $\rho \in \mathsf{Sol}(\mathcal{M}', \mathcal{RT})$ tal que $\rho(x) = a$ e $\rho(v) \neq a$. Então $\rho \notin \mathsf{Sol}(\mathcal{M}, \mathcal{RT})$;

(ii) se $x.f^n \in \mathcal{M}$ para algum $f^n \in \Sigma$ prova-se como no caso anterior, sendo $a \neq f^n$;

(iii) se $y = x \in \mathcal{M}$ ou $y.f_i^n \doteq x$ para algum $y \in \mathsf{Vars}(\mathcal{M})$, algum $f^n \in \Sigma$ e algum $1 \leq i \leq n$, podemos construir $\rho \in \mathsf{Sol}(\mathcal{M}', \mathcal{RT})$ tal que $\rho(x) \neq \rho(y)$ ou $\rho(y) = t_y$ e $\rho(x) \neq t_y|_i$.

2

Em analogia com a minimização de autómatos finitos determinísticos [HU79, Col82], vamos obter uma forma resolvida mínima correspondente a uma conjunção de restrições atómicas positivas satisfazível, que identifique expressões equivalentes, no sentido de corresponderem às mesmas árvores racionais.

Definição 4.9 *Sendo $\mathcal{M}$ uma forma resolvida, o conjunto de valores de $\mathcal{M}$ é dado por*

$$\mathsf{Val}(\mathcal{M}) = \mathsf{Vars}(\mathcal{M}) \ \cup \ \{a \mid a \in \Sigma,\, x = a \in \mathcal{M} \text{ ou } x.f_i^n \doteq a \in \mathcal{M}\}$$

Sendo $\mathcal{M}$ uma forma resolvida, a relação de equivalência $=_{\mathcal{M}}$ pode ser estendida a $\mathsf{Val}(\mathcal{M})$, dividindo-o duma maneira única, em classes de equivalência. Se $v \in \mathsf{Val}(\mathcal{M})$ e v não ocorre em nenhuma igualdade, a classe $[v]_{\mathcal{M}}$ diz-se *unitária*. Pela Definição 4.1, condição *(ii)*, cada classe de equivalência de $=_{\mathcal{M}}$ tem como representante um valor v que corresponde ao lado direito das equações que definem a classe e denota-se essa classe por $[v]_{\mathcal{M}}$. Salvo especificação contrária, iremos considerar a relação $=_{\mathcal{M}}$ definida em $\mathsf{Val}(\mathcal{M})$.

Pretendemos encontrar uma forma resolvida equivalente a $\mathcal{M}$ cujo número de classes de equivalência de $=_{\mathcal{M}}$ seja mínimo.

Definição 4.10 *Seja $\mathcal{M}$ uma forma resolvida e considere-se a partição de* $\mathsf{Val}(\mathcal{M})$ *em classes de equivalência de $=_{\mathcal{M}}$. Cada classe $[v]_{\mathcal{M}}$ designa-se por:*

(i) constante, *se contém uma constante (e só uma);*

(ii) funcional, *se $v \equiv x$ e $x.f^n \in \mathcal{M}$; f^n diz-se o seu* functor*; x é designada por* variável funcional*; neste caso não existe outra variável $y \in [v]_{\mathcal{M}}$ tal que $y.g^k \in \mathcal{M}$ e $f^n \not\equiv g^k$;*

(iii) livre *em $\mathcal{M}$, se as condições (i) e (ii) não se verificarem; a variável v é designada por* livre *em $\mathcal{M}$.*

Sendo $\mathcal{M}$ uma forma resolvida, cada classe de equivalência de $=_{\mathcal{M}}$ verifica uma e uma só das condições (i)-(iii) da definição anterior.

Note-se que a satisfazibilidade de $\mathcal{M}$ não depende dos valores atribuídos às suas variáveis *livres*.

Na forma resolvida

$$(\mathsf{c}_8) \qquad x.f^4 \ \wedge\ z = x \ \wedge\ y = x \ \wedge\ w = b \ \wedge\ x.f_1^4 \doteq b \ \wedge\ x.f_2^4 \doteq t \ \wedge\ u = b$$

$\mathsf{Val}(\mathsf{c}_8) = \{x, y, z, w, u, t, b\}$ e as classes de equivalência de $=_{\mathsf{c}_8}$ são

$$\{\{x, y, z\}, \{w, u, b\}, \{t\}\}$$

onde a primeira classe é funcional com functor f^4, a segunda é constante e a terceira é livre em c_8.

Definição 4.11 *Para qualquer forma resolvida $\mathcal{M}$, uma variável $x \in \mathsf{Vars}(\mathcal{M})$ diz-se* definida em $\mathcal{M}$ *se $x.f^n \in \mathcal{M}$ e para todo $1 \leq i \leq n$, existe $v \in \mathsf{Val}(\mathcal{M})$ tal que $x.f_i^n \doteq v \in \mathcal{M}$. Se para algum $1 \leq j \leq n$, não existe $v \in \mathsf{Val}(\mathcal{M})$ tal que $x.f_j^n \doteq v \in \mathcal{M}$ diz-se que a variável x* não é definida para j em $\mathcal{M}$.

Uma classe funcional $[x]_{\mathcal{M}}$ diz-se (não) definida, *se a variável x* (não) *o for em $\mathcal{M}$.*

No exemplo anterior, a variável x é não definida em c_8. As variáveis x e y dos exemplos c_5 e c_6 são ambas definidas.

Definição 4.12 *Sendo $\mathcal{M}$ uma forma resolvida, os valores $v, v' \in \mathsf{Val}(\mathcal{M})$ são* equivalentes *em $\mathcal{M}$, se $\mathcal{M} \models_{\mathcal{RT}} v = v'$ e denota-se este facto por $v \sim_{\mathcal{M}} v'$. Se $v, v' \in \mathsf{Val}(\mathcal{M})$ e $v \not\sim_{\mathcal{M}} v'$, então os valores v e v' dizem-se* distinguíveis *em $\mathcal{M}$. Duas classes $[v]_{\mathcal{M}}$ e $[v']_{\mathcal{M}}$ são distinguíveis se $v \not\sim_{\mathcal{M}} v'$.*

Como vimos, pelo exemplo c_7, as variáveis x e y são equivalentes em c_6. No exemplo c_8 todas as variáveis de classes diferentes são distinguíveis.

É importante notar que o facto de duas variáveis serem distinguíveis numa forma resolvida não implica que elas não possam ter uma mesma solução. Isto é, não são necessariamente *incompatíveis*, como será visto na Definição 4.37.

Seja $\mathcal{M}$ uma forma resolvida. A relação $\sim_{\mathcal{M}}$ definida em $\mathsf{Val}(\mathcal{M})$ é uma relação de equivalência. Obviamente, todos os valores duma mesma classe $[v]_{\mathcal{M}}$ são equivalentes em $\mathcal{M}$. Por outro lado, duas constantes distintas são sempre distinguíveis e duas variáveis x e y são distinguíveis em $\mathcal{M}$ se:

- pertencem a classes constantes distintas ou se só uma delas pertence a uma tal classe;
- pertencem a classes livres em $\mathcal{M}$ distintas ou se só uma delas pertence a uma tal classe;
- pertencem a classes funcionais com functores distintos;
- pertencem a classes funcionais distintas com o mesmo functor, mas pelo menos uma delas não é definida em $\mathcal{M}$.

Nestes casos é fácil exibir[4] uma atribuição de variáveis ρ tal que $\mathcal{RT} \models_\rho \mathcal{M}$ e $\mathcal{RT} \not\models_\rho x = y$.

No exemplo c_8 todas as classes são distinguíveis por uma das condições acima. A primeira e a segunda porque uma é funcional e a outra é constante; a primeira e a terceira porque uma é funcional e a outra é livre; a segunda e a terceira porque uma é livre e a outra é constante.

Mas duas variáveis podem ser distinguíveis e não satisfazerem nenhuma das condições anteriores.

Por exemplo, na restrição

$$(\mathsf{c}_9) \qquad x.f^2 \ \wedge \ x.f_1^2 \doteq a \ \wedge \ x.f_2^2 \doteq w \ \wedge \ y.f^2 \ \wedge \ y.f_1^2 \doteq a \ \wedge \ y.f_2^2 \doteq u$$

as variáveis x e y são distinguíveis e não verificam as condições.

Para caracterizar todas as variáveis distinguíveis em $\mathcal{M}$, reintroduzimos a noção de caminhos, mas restringindo-nos às variáveis que ocorrem em $\mathcal{M}$.

Definição 4.13 *Para qualquer forma resolvida $\mathcal{M}$, o conjunto $\mathsf{Paths}(\mathcal{M})$ dos caminhos de $\mathcal{M}$ é o menor conjunto tal que:*

(i) se $x \in \mathsf{Vars}(\mathcal{M})$ então $x\langle\epsilon\rangle x \in \mathsf{Paths}(\mathcal{M})$, onde ϵ denota a sequência vazia;

(ii) se $x = v \in \mathcal{M}$ então $x\langle\epsilon\rangle v \in \mathsf{Paths}(\mathcal{M})$;

(iii) se $x.f_i^n \doteq v \in \mathcal{M}$ então $x\langle f_i^n\rangle v \in \mathsf{Paths}(\mathcal{M})$;

(iv) se $x\langle p\rangle z \in \mathsf{Paths}(\mathcal{M})$ e $z\langle q\rangle v \in \mathsf{Paths}(\mathcal{M})$ então $x\langle pq\rangle v \in \mathsf{Paths}(\mathcal{M})$, onde p e q denotam caminhos em $\Pi(\Sigma)$.

Para cada $x \in \mathsf{Vars}(\mathcal{M})$ definem-se o conjunto de caminhos de x em $\mathcal{M}$ e os conjuntos dos valores e das variáveis acessíveis de x em $\mathcal{M}$, respectivamente por:

$$\begin{aligned}
\mathsf{Paths}_x(\mathcal{M}) &= \{p \mid \textit{existe } v \textit{ tal que } x\langle p\rangle v \in \mathsf{Paths}(\mathcal{M})\} \\
\mathsf{Val}_x(\mathcal{M}) &= \{v \in \mathsf{Val}(\mathcal{M}) \mid \textit{existe } p \textit{ tal que } x\langle p\rangle v \in \mathsf{Paths}(\mathcal{M})\} \\
\mathsf{Vars}_x(\mathcal{M}) &= \mathsf{Val}_x(\mathcal{M}) \cap \mathsf{Vars}(\mathcal{M})
\end{aligned}$$

Se $z \in \mathsf{Vars}_x(\mathcal{M})$ diz-se que z depende *de x em $\mathcal{M}$.*

[4] No caso de Σ ser finita, supomos que Σ contém pelo menos dois símbolos.

Noções como comprimento, maximalidade, etc, de caminhos são generalizadas naturalmente para elementos de $\mathsf{Paths}(\mathcal{M})$.

Para qualquer $x \in \mathsf{Vars}(\mathcal{M})$, o conjunto $\mathsf{Val}_x(\mathcal{M})$ é finito e pode ser determinado a partir da definição do conjunto $\mathsf{Paths}(\mathcal{M})$. Em particular, se $x\langle\epsilon\rangle v \in \mathsf{Paths}(\mathcal{M})$ então $x \in [v]_{\mathcal{M}}$.

Lema 4.14 *O conjunto* $\mathsf{Paths}(\mathcal{M})$ *é infinito se e só se existe* $x \in \mathsf{Vars}(\mathcal{M})$ *e* $p \not\equiv \epsilon$ *tal que* $x\langle p\rangle x \in \mathsf{Paths}(\mathcal{M})$.

Dem. ($\Leftarrow$)

Se $x\langle p\rangle x \in \mathsf{Paths}(\mathcal{M})$ então por indução sobre n prova-se que $\forall n \leq 1, x\langle p^n\rangle x \in \mathsf{Paths}(\mathcal{M})$.

($\Rightarrow$)

Por redução ao absurdo. Sendo $m = |\mathsf{Vars}(\mathcal{M})|$, se a condição não se verificar o comprimento de cada p seria no máximo m, logo $\mathsf{Paths}(\mathcal{M})$ seria finito. 2

Lema 4.15 *Seja* $\mathcal{M}$ *uma forma resolvida e* x *e* y *variáveis distintas de* $\mathsf{Vars}(\mathcal{M})$. *As variáveis* x *e* y *são distinguíveis em* $\mathcal{M}$ *se e só se existem* p, v *e* v' *tal que* $x\langle p\rangle v$, $y\langle p\rangle v' \in \mathsf{Paths}(\mathcal{M})$, $v \not\equiv v'$ *e as classes* $[v]_{\mathcal{M}}$ *e* $[v']_{\mathcal{M}}$ *satisfazem uma das seguintes condições:*

1. *pelo menos uma é constante;*
2. *pelo menos uma é livre em* $\mathcal{M}$*;*
3. *são classes funcionais com functores distintos;*
4. *são classes funcionais com o mesmo functor mas pelo menos uma delas não é definida.*

Dem. ($\Leftarrow$)

Provamos por indução na estrutura de p que $x \not\sim_{\mathcal{M}} y$.

Base. Se $p \equiv \epsilon$, então $x \in [v]_{\mathcal{M}}$ e $y \in [v']_{\mathcal{M}}$ e é imediato que se $[v]_{\mathcal{M}}$ e $[v']_{\mathcal{M}}$ satisfazem uma das condições *1–4*, então $v \not\sim_{\mathcal{M}} v'$, e consequentemente $x \not\sim_{\mathcal{M}} y$.

Indução. Se $p \equiv f_i^n q$ então existem valores v_1, $v_2 \in \mathsf{Val}(\mathcal{M})$ tal que $x.f_i^n \doteq v_1 \in \mathcal{M}$ e $y.f_i^n \doteq v_2 \in \mathcal{M}$. Se v_1 ou v_2 são constantes, verificam a condição *1* e $q \equiv \epsilon$. Podemos concluir facilmente que $x \not\sim_{\mathcal{M}} y$. Senão v_1 e v_2 são variáveis, sejam z_1 e z_2, respectivamente. Então, temos $x.f_i^n \doteq z_1 \in \mathcal{M}$, $z_1\langle q\rangle v$, $y.f_i^n \doteq z_2 \in \mathcal{M}$ e $z_2\langle q\rangle v'$. Por hipótese de indução, sabemos que $z_1 \not\sim_{\mathcal{M}} z_2$, logo existe ρ tal que $\mathcal{RT} \models_\rho \mathcal{M}$ e $\rho(z_1) \neq \rho(z_2)$. Sendo $\rho(x) = t_x$ e $\rho(y) = t_y$ como $\rho(z_1) = t_x |_i \neq t_y |_i = \rho(z_2)$ então $\rho(x) \neq \rho(y)$.

($\Rightarrow$)
Sendo $x \not\sim_{\mathcal{M}} y$ existe uma atribuição das variáveis ρ tal que $\mathcal{RT} \models_\rho \mathcal{M}$ e $t_x = \rho(x) \neq \rho(y) = t_y$. Seja α a posição de comprimento mínimo tal que $t_x(\alpha) \neq t_y(\alpha)$. Provamos por indução no comprimento de α, a existência de p, v e v' que verificam as condições.

Base. Se $\alpha = \epsilon$ então $t_x(\epsilon) \neq t_y(\epsilon)$, isto é, são elementos distintos de Σ. Para todos os casos, $p \equiv \epsilon$, $x \in [v]_{\mathcal{M}}$, $y \in [v']_{\mathcal{M}}$ e uma das condições *1–3* é satisfeita por $[v]_{\mathcal{M}}$ e $[v']_{\mathcal{M}}$. Suponhamos que $t_x(\epsilon) = a$ e $t_y(\epsilon) = b$ então

(i) $x \in [a]_{\mathcal{M}}$ e $y \in [b]_{\mathcal{M}}$ (a condição *1* verifica-se); ou

(ii) $x \in [a]_{\mathcal{M}}$, y é livre em $\mathcal{M}$ e $\rho(y) = b$, ou vice-versa (a condição *2* verifica-se); ou

(iii) x e y são livres em $\mathcal{M}$ e $\rho(y) = b$ e $\rho(x) = a$ (a condição *2* verifica-se);

(iv) e não há mais possibilidades

Analogamente, se prova para o caso de $t_x(\epsilon)$ ou $t_y(\epsilon)$ serem functores distintos.

Indução. Seja $\alpha = i\beta$ e suponhamos[5] que $t_x(\epsilon) = t_y(\epsilon) = f^n$ e $1 \leq i \leq n$. Se x ou y é livre em $\mathcal{M}$ então, mais uma vez, a condição *2* é verificada e $p \equiv \epsilon$. Caso contrário, sabemos que $x.f^n \in \mathcal{M}$ e $y.f^n \in \mathcal{M}$. Se x ou y é não definida em $\mathcal{M}$, a condição *4* verifica-se e $p \equiv \epsilon$. Caso contrário, $x.f_i^n \doteq v_1 \in \mathcal{M}$ e $y.f_i^n \doteq v_2 \in \mathcal{M}$. Então $\rho(v_1) = t_x|_i$, $\rho(v_2) = t_y|_i$ e β é a posição de comprimento mínimo tal que $t_x|_i\,(\beta) \neq t_y|_i\,(\beta)$. Se v_1 ou v_2 são constantes então, necessariamente, $\beta \equiv \epsilon$ e consideramos $p \equiv f_i^n$. Senão, v_1 e v_2 são variáveis, sejam z_1 e z_2, respectivamente. Por hipótese de indução, existe q tal que $z_1\langle q\rangle v \in \mathsf{Paths}(\mathcal{M})$ e $z_2\langle q\rangle v' \in \mathsf{Paths}(\mathcal{M})$ e uma das condições *1-4* verifica-se para $[v]_{\mathcal{M}}$ e $[v']_{\mathcal{M}}$. Neste caso, consideramos $p \equiv f_i^n q$.

2

Corolário 4.16 *Sendo $\mathcal{M}$ uma forma resolvida e x e y variáveis distintas definidas em $\mathcal{M}$ tem-se que $x \sim_{\mathcal{M}} y$ se e só se para todo $p \not\equiv \epsilon$, v e v' tal que $x\langle p\rangle v$, $y\langle p\rangle v' \in \mathsf{Paths}(\mathcal{M})$, $v =_{\mathcal{M}} v'$ ou as classes $[v]_{\mathcal{M}}$ e $[v']_{\mathcal{M}}$ são definidas com o mesmo functor.*

O Corolário 4.16 permite concluir que $x = y$ é vinculado pelas restrições dos exemplos c_5 e c_6.

O lema seguinte apresenta um algoritmo que a partir de uma forma resolvida $\mathcal{M}$ permite determinar todas as variáveis x, $y \in \mathsf{Vars}(\mathcal{M})$ tal que $x \sim_{\mathcal{M}} y$.

[5] Nem t_x nem t_y podem ser constantes.

Lema 4.17 *Sendo $\mathcal{M}$ uma forma resolvida é possível determinar todos os pares de classes de equivalência $([v]_{\mathcal{M}}, [v']_{\mathcal{M}})$ que são distinguíveis em $\mathcal{M}$.*

Dem. Considere-se o algoritmo de identificação de classes distinguíveis (Figura 4.1), que é uma adaptação do usado na minimização de autómatos finitos, em [HU79].

Entrada: Conjunto de classes de equivalência duma forma resolvida $\mathcal{M}$.

Saída: Pares de classes de equivalência distinguíveis.

Passo 1

Marcar todos os pares de classes distintas $([v]_{\mathcal{M}}, [v']_{\mathcal{M}})$ que verifiquem uma das seguintes condições:

- pelo menos uma delas é constante;
- pelo menos uma delas é livre em $\mathcal{M}$;
- são classes funcionais com functores distintos;
- são classes funcionais com o mesmo functor mas pelo menos uma delas não é definida em $\mathcal{M}$.

Passo 2 [6]

Para cada par de classes funcionais definidas $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$ com um mesmo functor f^n e não marcadas **faça** {

  Se para algum $1 \leq i \leq n$, $x.f_i^n \doteq v \in \mathcal{M}$, $y.f_i^n \doteq v' \in \mathcal{M}$ e $([v]_{\mathcal{M}}, [v']_{\mathcal{M}})$ está marcado **então** {

    marcar $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$ e marcar recursivamente todos os pares de classes não marcados $([w]_{\mathcal{M}}, [z]_{\mathcal{M}})$ tais que sendo g^l o seu functor, $w.g_k^l \doteq x \in \mathcal{M}$ e $z.g_k^l \doteq y \in \mathcal{M}$ para algum $1 \leq k \leq l$

  }

}

Figura 4.1: Identificação de classes distinguíveis numa forma resolvida $\mathcal{M}$

O algoritmo termina dado que é finito o número classes e a aridade de cada functor, e não há a possibilidade de ciclo (infinito) no passo 2. O facto de $x \not\sim_{\mathcal{M}} y$ se e só se $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$ foi marcado, é consequência do Lema 4.15.

$(\Rightarrow)$

Suponhamos que $x \not\sim_{\mathcal{M}} y$ e que p é o caminho de comprimento mínimo tal que o

[6] Ou simplesmente

Enquanto existe um par não marcado de classes funcionais definidas $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$ com um mesmo functor f^n, tal que para algum $1 \leq i \leq n$, $x.f_i^n \doteq v \in \mathcal{M}$, $y.f_i^n \doteq v' \in \mathcal{M}$ e $([v]_{\mathcal{M}}, [v']_{\mathcal{M}})$ está marcado **faça** { marcar o par $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$ }

Lema 4.15 é satisfeito. Provamos por indução na estrutura do caminho p.

Base. Se $p \equiv \epsilon$ então o par $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$ é marcado no passo 1

Indução. Se $p \equiv f_i^n q$ então existem valores v_1 e v_2 tais que $x.f_i^n \doteq v_1 \in \mathcal{M}$ e $y.f_i^n \doteq v_2 \in \mathcal{M}$. Se v_1 ou v_2 são constantes então $q \equiv \epsilon$ e o par $([v_1]_{\mathcal{M}}, [v_2]_{\mathcal{M}})$ é marcado no passo 1. Logo, o par $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$ é marcado no passo 2. Senão v_1 e v_2 são variáveis e q é o caminho mínimo que distingue v_1 de v_2. Por hipótese de indução, $([v_1]_{\mathcal{M}}, [v_2]_{\mathcal{M}})$ é marcado pelo algoritmo. Se já o foi, então o par $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$ é marcado imediatamente. Senão quando $([v_1]_{\mathcal{M}}, [v_2]_{\mathcal{M}})$ for marcado, $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$ é-o também.

($\Leftarrow$)
Por indução, supondo que o resultado é válido para os pares já marcados. 2

Sendo $\mathcal{M}$ uma forma resolvida, N o número de classes de equivalência de $\mathcal{M}$ e

$$K = \max\{n \in \mathbb{N} \mid f^n \in \Sigma \text{ e } f^n \text{ é o functor de alguma classe } [x]_{\mathcal{M}}\}$$

o algoritmo de identificação de classes distinguíveis (Figura 4.1) tem complexidade da ordem $O(KN^2)$.

Se x e y são variáveis distintas de $\mathcal{M}$ e o par $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$ não é marcado pelo algoritmo de identificação de classes distinguíveis (Figura 4.1), então $x \sim_{\mathcal{M}} y$.

Definição 4.18 *Uma forma resolvida $\mathcal{M}$ diz-se* normal *se para todas as variáveis x, $y \in$* $\mathsf{Vars}(\mathcal{M})$ *tais que $x \sim_{\mathcal{M}} y$, então $x =_{\mathcal{M}} y$.*

Proposição 4.19 *Toda a forma resolvida $\mathcal{M}$ pode ser transformada numa forma normal equivalente pela aplicação da seguinte regra de reescrita*

N1. $\mathcal{M} \mapsto \mathcal{M} \cup \{x = y\}$ *se* $x,\ y \in \mathsf{Vars}(\mathcal{M})$, $x \sim_{\mathcal{M}} y$ e $x \neq_{\mathcal{M}} y$

e reduzindo a restrição irredutível resultante a forma resolvida, usando o sistema de reescrita R.

Dem. Sendo $\mathcal{M}$ uma forma resolvida aplique-se o algoritmo de identificação de classes distinguíveis (Figura 4.1), para a identificação de classes distinguíveis em $\mathcal{M}$. Para cada par de classes não marcadas $([x]_{\mathcal{M}}, [y]_{\mathcal{M}})$, x e y verificam a condição da regra de reescrita e são as únicas variáveis de $\mathcal{M}$ que a verificam. Como é finito o número de tais pares, a aplicação da regra termina. Se não existirem tais pares, $\mathcal{M}$ já está em forma normal, e nada há a fazer. Senão, pela definição de $\sim_{\mathcal{M}}$ é evidente que a forma resultante é equivalente à inicial, *i.e*, $\mathcal{M} \approx_{\mathcal{RT}} \mathcal{M} \wedge x = y$, para cada par de variáveis, x e y. 2

Iremos designar por sistema de reescrita N, o sistema de reescrita que resulta do sistema de reescrita R acrescentando a regra $\mathsf{N1}$, com o controlo indicado na Proposição 4.19.

Vamos ilustrar a aplicação do sistema de reescrita N para a restrição c_5. Como é uma forma resolvida, basta aplicar o algoritmo para identificar classes distinguíveis. As classes de equivalência de $=_{\mathsf{c}_5}$ são $[x]_{\mathsf{c}_5}$ e $[y]_{\mathsf{c}_5}$, cada uma funcional com functor f^1 e definida em c_5. Assim, no primeiro passo do algoritmo o par $([x]_{\mathsf{c}_5}, [y]_{\mathsf{c}_5})$ não é marcado. No segundo passo, também não se verifica a condição da instrução condicional, porque $v \equiv x$ e $v' \equiv y$ e as classes correspondentes não estão marcadas. Como o algoritmo não marca o par $([x]_{\mathsf{c}_5}, [y]_{\mathsf{c}_5})$, concluí-se que $x \sim_{\mathsf{c}_5} y$ e a regra $\mathsf{N1}$ aplica-se. A forma resolvida (normal) resultante é a restrição c_3. De modo análogo, concluía-se que a forma resolvida normal da restrição c_4 é c_3, e que a restrição c_9 está em forma normal.

Para a restrição em forma resolvida

$$(\mathsf{c}_{10}) \qquad x.f^2 \wedge y.f^2 \wedge x.f^2_1 \doteq w \wedge x.f^2_2 \doteq z \wedge y.f^2_1 \doteq w \wedge y.f^2_2 \doteq z$$

a aplicação do sistema de reescrita N permite concluir, do mesmo modo, que x e y são equivalentes, resultando a forma normal

$$(\mathsf{c}_{11}) \qquad x.f^2 \wedge x.f^2_1 \doteq w \wedge x.f^2_2 \doteq z \wedge y = x$$

Proposição 4.20 *Se $\mathcal{M}$ e $\mathcal{M}'$ são formas normais tais que $\mathcal{M} \approx_{\mathcal{RT}} \mathcal{M}'$, então as relações $=_{\mathcal{M}}$ e $=_{\mathcal{M}'}$ são a mesma relação e $\mathcal{M}$ e $\mathcal{M}'$ são idênticas a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{M}}$.*

Dem. Pelo Lema 4.8, temos que $\mathsf{Vars}(\mathcal{M}) = \mathsf{Vars}(\mathcal{M}')$. Sejam x e y tal que $x =_{\mathcal{M}} y$ e $x \neq_{\mathcal{M}'} y$. Mas então $x \sim_{\mathcal{M}} y$ e como $\mathcal{RT} \models \mathcal{M} \Leftrightarrow \mathcal{M}'$, $x \sim_{\mathcal{M}'} y$ o que contradiz o facto de $\mathcal{M}'$ ser normal. Logo $x =_{\mathcal{M}} v$ se e só se $x =_{\mathcal{M}'} v$. A identidade resulta do facto de $\mathcal{M}$ e $\mathcal{M}'$ serem formas resolvidas equivalentes. 2

O corolário seguinte resulta da Proposição 4.19.

Corolário 4.21 *Se $\mathcal{M}$ é uma forma resolvida, a sua forma normal é obtida pela conjunção de $\mathcal{M}$ com restrições do tipo $x = y$, tal que $x \sim_{\mathcal{M}} y$, e eliminando restrições do tipo $v = v$, $a = x$, $x.f^n$ e $x.f^n_i \doteq v$.*

Dem. A redução a forma normal é feita por aplicação finita das regras $\mathsf{R2}$, $\mathsf{R4}$, $\mathsf{R5}$, $\mathsf{R6}$ ou $\mathsf{R10}$ do sistema de reescrita R de redução a forma resolvida. 2

Teorema 4.22 *Toda a conjunção $\mathcal{C}$ de restrições atómicas positivas ou não é satisfazível ou $\mathcal{C} \approx_{\mathcal{RT}} \mathcal{M}$, onde $\mathcal{M}$ é uma forma normal e $\mathcal{M}$ é única a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{M}}$.*

Dem. É consequência das Proposições 4.2, 4.4, 4.19 e 4.20. 2

Pelo Teorema 4.22, uma forma normal $\mathcal{M}$ será considerada, salvo indicação em contrário, a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{M}}$. Usaremos, a relação $\equiv$ para indicar que duas formas normais $\mathcal{M}_1$ e $\mathcal{M}_2$ são idênticas a menos dessa permutação. Esta notação aplica-se também a formas resolvidas.

4.2.1 Formas Normais para $\mathcal{T}$

Nesta secção vamos apenas referir quais as formas normais de $\mathcal{LU}$ para a álgebra de árvores finitas $\mathcal{T}$.

Uma restrição $\mathcal{M} \in \mathcal{LU}$ é uma *forma resolvida finita* se $\mathcal{M}$ é uma forma resolvida e para além disso verifica a seguinte condição:

(vi) para todo $x \in \mathsf{Vars}(\mathcal{M})$, se $x\langle p\rangle x \in \mathsf{Paths}(\mathcal{M})$ então $p \equiv \epsilon$

Proposição 4.23 *Se $\mathcal{M}$ é uma forma resolvida finita então $\mathcal{M}$ é satisfazível em $\mathcal{T}$.*

Dem. Resulta da Proposição 4.2 e do Lema 4.14. 2

Uma forma resolvida finita $\mathcal{M}$ é *normal* se e só se verifica a seguinte condição:

(i) se existem variáveis distintas $x, y \in \mathsf{Vars}(\mathcal{M})$ e um valor v tal que $x.f_i^n \doteq v \in \mathcal{M}$ e $y.f_i^n \doteq v \in \mathcal{M}$ então para algum $1 \leq j \leq n$ não existe v' tal que $x.f_j^n \doteq v' \in \mathcal{M}$ e $y.f_j^n \doteq v' \in \mathcal{M}$

A proposição seguinte assegura que a definição anterior corresponde à definição de formas normais.

Proposição 4.24 *Se $\mathcal{M}$ é uma forma resolvida finita normal então $\forall x, y \in \mathsf{Vars}(\mathcal{M})$, $x \sim_{\mathcal{M}} y$ se e só se $x =_{\mathcal{M}} y$.*

Dem. Pelo Lema 4.15 e Corolário 4.16. 2

O Lema 4.17 pode ser adaptado facilmente para este caso uma vez que $\mathsf{Paths}(\mathcal{M})$ é finito e usar-se a Proposição 4.19 para obter formas normais.

4.3 Formas Primárias

Nesta secção vamos analisar a satisfazibilidade duma conjunção $\mathcal{C}_p \wedge \mathcal{C}_n$ de restrições atómicas positivas e negativas. Podemos supor que a conjunção $\mathcal{C}_p$ de restrições atómicas positivas está em forma normal. A noção de forma resolvida normal será generalizada, de modo a incluir restrições atómicas negativas, a uma forma que chamaremos *forma primária*.[7]

O estudo destas restrições será dividido em dois casos, consoante a assinatura Σ seja infinita ou finita. Enquanto que, no primeiro caso, cada conjunção de restrições atómicas satisfazível será caracterizada por uma forma primária, no segundo, será caracterizada por uma disjunção dessas formas. Para ambos os casos, serão ainda analisadas as condições em que as restrições negativas podem ser eliminadas ou substituídas por disjunções de restrições positivas.

Começamos por caracterizar as restrições negativas *triviais*.

Definição 4.25 *Uma restrição atómica negativa diz-se* trivial *se é do tipo $v \neq v$, $a \neq b$, com $a \not\equiv b$, ou $\neg a.f^n$. No primeiro caso, é ainda designada por restrição* contraditória

As restrições do tipo $v \neq v$ são não satisfazíveis em qualquer álgebra de $\mathcal{L}$. As restrições do tipo $a \neq b$ ou $\neg a.f^n$ são válidas em $\mathcal{RT}$ (e em $\mathcal{T}$).

Embora em $\mathcal{LU}$ não exista o operador de negação, se c for uma restrição atómica positiva do tipo $v = v'$ ou $v.f^n$, denotamos por $\neg c$ a restrição negativa correspondente, e, analogamente, se c for uma restrição atómica negativa, $\neg c$ representa a restrição positiva correspondente. Iremos denotar as restrições atómicas negativas por $n_1, n_2, \ldots$. Se n_i for uma restrição atómica negativa, denotamos a restrição atómica positiva $\neg n_i$ por c_i.

4.3.1 Formas Primárias para Σ Infinita

[7] Não se usa a designação de *básica* para evitar confusões com as *fórmulas básicas* tal como definidas por Maher [Mah88], embora outros autores usem o termo *básica* para outros tipos de fórmulas, por exemplo em [Com91b]. As propriedades essenciais destas fórmulas são: serem satisfazíveis e qualquer outra fórmula ser equivalente a uma sua combinação booleana.

Lema 4.26 *Seja $\mathcal{C}$ uma conjunção de restrições atómicas negativas não triviais do tipo $x \neq v$ ou $\neg x.f^n$. Se $\mid \Sigma \mid = \infty$, então $\mathcal{C}$ é satisfazível em $\mathcal{RT}$.*

Dem. Seja $\mathsf{Sym} = \{\tau \in \Sigma : \; x \neq \tau \in \mathcal{C}$ ou $\neg x.\tau \in \mathcal{C}\}$. Como Sym e $\mathsf{Vars}(\mathcal{C})$ são finitos, basta considerar uma solução ρ, tal que $\rho(x) = t_x$ e $t_x(\epsilon) \in \Sigma \setminus \mathsf{Sym}$ seja diferente para cada $x \in \mathsf{Vars}(\mathcal{C})$. 2

Para a caracterização da satisfazibilidade duma conjunção de restrições atómicas positivas e negativas usaremos a proposição seguinte.

Proposição 4.27 *Seja $\mid \Sigma \mid = \infty$ e sejam $\mathcal{M}$ e $\mathcal{M}_i$, para $1 \leq i \leq m$, formas resolvidas normais. Se $\mathcal{M} \approx_{\mathcal{RT}} \mathcal{M}_1 \;\vee\; \ldots \;\vee\; \mathcal{M}_m$ então, para algum $1 \leq i \leq m$, tem-se que $\mathcal{M} \approx_{\mathcal{RT}} \mathcal{M}_i$.*

Dem. Provamos por redução ao absurdo. Suponhamos que para todo $1 \leq i \leq m$, temos $\mathsf{Sol}(\mathcal{M}_i, \mathcal{RT}) \subset \mathsf{Sol}(\mathcal{M}, \mathcal{RT})$. Então

$$\forall\, 1 \leq i \leq m, \; \exists \rho_i \in \mathsf{Sol}(\mathcal{M}, \mathcal{RT}) \text{ e } \rho_i \notin \mathsf{Sol}(\mathcal{M}_i, \mathcal{RT})$$

Como $\mathsf{Vars}(\mathcal{M}) \subseteq \mathsf{Vars}(\mathcal{M}_i)$ e como qualquer solução ρ' de $\mathcal{M}_i$ verifica $\rho' = \rho'\theta_{\mathcal{M}}$, para cada $1 \leq i \leq m$ existe $x_i \in \mathsf{Vars}(\mathcal{M}_i)$ tal que uma das seguintes condições ocorre:

(i) $x_i \in [y]_{\mathcal{M}_i}$, $y \not\equiv x_i$, e tal não acontece em $\mathcal{M}$;

(ii) $x_i \notin \mathsf{Vars}(\mathcal{M})$ ou $[x_i]_{\mathcal{M}}$ é livre e $[x_i]_{\mathcal{M}_i}$ é constante ou funcional;

(iii) $[x_i]_{\mathcal{M}}$ é funcional com functor f^n, é não definida em j, para algum $1 \leq j \leq n$, e $[x_i]_{\mathcal{M}_i}$ é definida em j.

As condições anteriores garantem que existem p_i e v_i tal que $x_i\langle p_i\rangle v_i \in \mathsf{Paths}(\mathcal{M}_i)$, com $x_i \not\equiv v_i$ se $p_i \equiv \epsilon$, e para p_i não existe $v_i' \not\equiv v_i$ tal que $x_i\langle p_i\rangle v_i' \in \mathsf{Paths}(\mathcal{M})$, ou se existe, v_i' é livre e v_i é constante ou funcional.

Supondo que as variáveis introduzidas por $\theta_{\mathcal{M}}$ não ocorrem em nenhum dos $\mathcal{M}_i$, seja Y o seguinte conjunto de variáveis:

$$\begin{aligned} Y \;\; = \;\; & \{y \in X \mid \;\; y \text{ é livre em } \mathcal{M} \text{ ou } y \text{ foi introduzida por } \theta_{\mathcal{M}}\} \\ & \cup \bigcup_{i=1}^{n} \mathsf{Vars}(\mathcal{M}_i) \setminus \mathsf{Vars}(\mathcal{M}) \end{aligned}$$

e seja

$$\mathsf{Sym} = \{\tau \in \Sigma : \; x = \tau \in \mathcal{M}_i \text{ ou } x.\tau \in \mathcal{M}_i, \text{ para todo } 1 \leq i \leq m\}$$

Definimos a atribuição de variáveis ρ tal que para cada $y \in Y$, $\rho(y) = t_y$, $t_y(\epsilon) \in \Sigma \setminus \mathsf{Sym}$ e $t_y(\epsilon)$ é diferente para cada y, que não pertença a uma mesma classe de equivalência de $=_{\mathcal{M}}$. As variáveis de Y que pertencem a uma mesma classe de equivalência de $=_{\mathcal{M}}$ têm o mesmo valor por ρ. Então $\rho\theta_{\mathcal{M}}$ é solução de $\mathcal{M}$ e para todo $1 \leq i \leq m$, tem-se que $\rho\theta_{\mathcal{M}} \notin \mathsf{Sol}(\mathcal{M}_i, \mathcal{RT})$, já que o valor de $\rho(x_i)$ não satisfaz $\mathcal{M}_i$. Obtemos uma contradição. 2

O lema seguinte afirma que $\mathcal{LU}$ verifica a propriedade da *Independência das Restrições Negativas.* É uma generalização do Lema da *Independência das Desigualdades* (ver página 58) e é uma consequência da proposição anterior.

Lema 4.28 *Seja* $\mid \Sigma \mid = \infty$ *e* $\mathcal{C} \equiv \mathcal{C}_p \ \wedge \ n_1 \ \wedge \ \ldots \ \wedge \ n_m$, *onde* $\mathcal{C}_p$ *é uma conjunção de restrições atómicas positivas e cada* n_i *uma restrição atómica negativa. A restrição* $\mathcal{C}$ *é satisfazível em* $\mathcal{RT}$ *se e só se para todo* $1 \leq i \leq m$, $\mathcal{C}_p \ \wedge \ n_i$ *o for.*

Dem. $(\Rightarrow)$

É trivial.

$(\Leftarrow)$

Sem perda de generalidade, podemos supor que $\mathcal{C}_p$ é satisfazível em $\mathcal{RT}$. Analisemos o caso de algum n_i ser trivial. Se n_i é do tipo $v \neq v$, então $\mathcal{C}$ é não satisfazível em $\mathcal{RT}$. Se n_i é do tipo $\neg a.f^n$ ou $a \neq b$, então n_i pode ser omitido. Caso contrário, suponhamos que

$$\mathcal{RT} \models \mathcal{C}_p \ \wedge \ n_1 \ \wedge \ \ldots \ \wedge \ n_m \ \Leftrightarrow \ \perp$$

Então

$$\mathcal{RT} \models \mathcal{C}_p \ \Leftrightarrow \ (\mathcal{C}_p \ \wedge \ c_1) \ \vee \ \ldots \ \vee \ (\mathcal{C}_p \ \wedge \ c_m)$$

Pela Proposição 4.27, existe $1 \leq i \leq m$ tal que $\mathcal{C}_p \approx_{\mathcal{RT}} \mathcal{C}_p \ \wedge \ c_i$, isto é,

$$\mathcal{RT} \models \mathcal{C}_p \ \wedge \ n_i \ \Leftrightarrow \ \perp$$

2

Sendo $\mathcal{C}_p$ uma forma resolvida normal, $\mathcal{C}_p \wedge n_i$ é não satisfazível em $\mathcal{RT}$ se e só se $\mathcal{C}_p \models_{\mathcal{RT}} c_i$, isto é, se $\mathcal{C}_p$ vincula c_i em $\mathcal{RT}$. Podemos distinguir os seguintes casos:

- $\mathcal{C}_p \ \wedge \ c_i \overset{\mathsf{R}}{\mapsto}^{\star} \perp$ se e só se $\mathcal{C}_p \ \wedge \ n_i$ é satisfazível e $\mathcal{C}_p \models_{\mathcal{RT}} n_i$.
- $\mathcal{C}_p \ \wedge \ c_i \overset{\mathsf{R}}{\mapsto}^{\star} \mathcal{C}_p$ se e só se $\mathcal{C}_p \ \wedge \ n_i$ é não satisfazível e $\mathcal{C}_p \models_{\mathcal{RT}} c_i$.
- $\mathcal{C}_p \ \wedge \ c_i \overset{\mathsf{R}}{\mapsto}^{\star} \mathcal{C}'_p$ e $\mathcal{C}'_p \not\equiv \mathcal{C}_p$ é uma forma normal se e só se $\mathcal{C}_p \ \wedge \ n_i$ é satisfazível. Como veremos, no Capítulo 6, Corolário 6.5, resulta duma extensão do Lema 4.51 que neste caso $\mathcal{RT} \models \mathcal{C}'_p \ \Leftrightarrow \ \mathcal{C}_p \ \wedge \ c'$ e $c' \equiv c'_1 \ \wedge \ c'_2 \ \wedge \ \ldots \ \wedge \ c'_m$, com cada c'_i da forma $v = v'$ ou $v.f^n$. Isto é, $\mathcal{RT} \models \mathcal{C}_p \ \wedge \ n_i \ \Leftrightarrow \ \mathcal{C}_p \ \wedge \ (n'_1 \ \vee \ n'_2 \ \vee \ \ldots \ \vee \ n'_m)$.

No entanto, nesta secção iremos ver que se $\mathcal{C}_p$ é uma forma resolvida normal podemos caracterizar as conjunções de restrições atómicas satisfazíveis sem necessitar de avaliar $\mathcal{C}_p \wedge c_i$. Uma comparação entre os dois métodos será apresentada na Secção 6.1.

Definição 4.29 *Sendo* $\mid \Sigma \mid = \infty$, *uma conjunção de restrições* $\mathcal{C}$ *está em (ou é uma)* forma primária *se é* $\top$ *ou se satisfaz as seguintes condições:*

(i) *toda a restrição em* $\mathcal{C}$ *é do tipo* $x.f^n$, $x.f_i^n \doteq v$, $x = v$, $x \neq v$ *ou* $\neg x.f^n$ *e ocorre uma só vez em* $\mathcal{C}$*;*

(ii) *a conjunção das restrições atómicas positivas de* $\mathcal{C}$ *está em forma resolvida normal e denota-se por* $\mathcal{C}_p$*; a conjunção das restrições atómicas negativas denota-se por* $\mathcal{C}_n$*.*

(iii) *se* $x = v \in \mathcal{C}_p$ *então* x *ocorre exactamente uma vez em* $\mathcal{C}$*;*

(iv) *se* $x \neq a \in \mathcal{C}$ *ou* $\neg x.f^n \in \mathcal{C}$ *então ou* $x \notin \mathsf{Vars}(\mathcal{C}_p)$ *ou* $[x]_{\mathcal{C}_p}$ *é livre em* $\mathcal{C}_p$*;*

(v) *se* $x \neq y \in \mathcal{C}$ *então* $y \neq x \notin \mathcal{C}$ e $x \not\equiv y$.

Considerando a condições *(i)* e *(v)* da definição anterior, numa forma primária não vamos distinguir $v \neq v'$ de $v' \neq v$.

Proposição 4.30 *Se* $\mid \Sigma \mid = \infty$ *e* $\mathcal{C}$ *é uma forma primária, então* $\mathcal{C}$ *é satisfazível em* $\mathcal{RT}$*.*

Dem. Seja $\mathcal{C} \equiv \mathcal{C}_p \wedge \mathcal{C}_n$ uma forma primária. Os casos $\mathcal{C}_p \equiv \top$ ou $\mathcal{C}_n \equiv \top$ são triviais. Sendo $\mathcal{C}_n \equiv n_1 \wedge \ldots \wedge n_m$, resulta do Lema 4.28 que $\mathcal{C}$ é satisfazível se para todo $1 \leq i \leq m$, a conjunção $\mathcal{C}_p \wedge n_i$ o for. Por outro lado, $\rho\theta_{\mathcal{C}_p}$ é uma solução de $\mathcal{C}_p \wedge n_i$ se ρ for construída como na demonstração do Lema 4.26. No caso de n_i ser do tipo $x \neq a$ ou $\neg x.f^n$ a condição *(iv)* garante a existência de solução. No caso de n_i ser do tipo $x \neq y$ a existência de solução resulta do facto de $\mathcal{C}_p$ estar em forma normal e, portanto, $x \neq y \in \mathcal{C}$ se $\mathcal{C}_p$ não vincula $x = y$. 2

As formas primárias serão denotadas por $\mathcal{P}$, $\mathcal{P}'$, $\mathcal{P}_0$, ..., com $\mathcal{P}_p$, $\mathcal{P}'_p$, $\mathcal{P}^0_p$, ... representando a conjunção de restrições positivas e $\mathcal{P}_n$, $\mathcal{P}'_n$, $\mathcal{P}^0_n$, ... representando a conjunção de restrições negativas. Como no caso das formas resolvidas, por vezes, estas letras podem também denotar explicitamente a restrição $\perp$.

Proposição 4.31 *Seja* $\mid \Sigma \mid = \infty$. *Qualquer conjunção de restrições atómicas* $\mathcal{C} \equiv \mathcal{C}_p \wedge \mathcal{C}_n$ *pode ser transformada numa restrição* $\mathcal{P}$, *tal que* $\mathcal{C} \approx_{\mathcal{RT}} \mathcal{P}$ *e* $\mathcal{P}$ *ou é uma forma primária* $\mathcal{P}_p \wedge \mathcal{P}_n$ *ou é* $\perp$. *A transformação é efectuada pela redução da conjunção* $\mathcal{C}_p$ *de restrições* *atómicas positivas a forma resolvida normal $\mathcal{P}_p$ ou a $\perp$, e depois, no primeiro caso, pela aplicação das restantes regras do sistema reescrita* P, *a seguir apresentado, até obter uma restrição irredutível.*

P1. $\mathcal{C}_p \cup \mathcal{C}_n \mapsto \perp$ *se* $\mathcal{C}_p \overset{\mathsf{R}}{\mapsto}^{\star} \perp$

P2. $\mathcal{C}_p \cup \mathcal{C}_n \mapsto \mathcal{P}_p \cup \mathcal{C}_n$ *se* $\mathcal{C}_p \overset{\mathsf{N}}{\mapsto}^{\star} \mathcal{P}_p$ *e* $\mathcal{P}_p$ *é uma forma normal*

P3. $\{\top\} \cup \mathcal{C} \mapsto \mathcal{C}$

P4. $\{\perp\} \cup \mathcal{C} \mapsto \perp$ *se* $\mathcal{C} \not\equiv \top$

P5. $\{x = v\} \cup \mathcal{C} \mapsto \{x = v\} \cup \mathcal{C}[v/x]$ *se* $x \in VarsC$

P6. $\{a \neq x\} \cup \mathcal{C} \mapsto \{x \neq a\} \cup \mathcal{C}$

P7. $\{v \neq v\} \cup \mathcal{C} \mapsto \perp$

P8. $\{a \neq b\} \cup \mathcal{C} \mapsto \mathcal{C}$ *se* a *e* b *são constantes distintas*

P9. $\{\neg a.f^n\} \cup \mathcal{C} \mapsto \mathcal{C}$

P10. $\{x \neq a\} \cup \mathcal{C} \mapsto \mathcal{C}$ *se* $x.f^n \in \mathcal{P}_p$, *para algum* $f^n \in \Sigma$

P11. $\{\neg x.g^m\} \cup \mathcal{C} \mapsto \mathcal{C}$ *se* $x.f^n \in \mathcal{P}_p$, *para algum* $f^n \neq g^m$

P12. $\{\neg x.f^n\} \cup \mathcal{C} \mapsto \perp$ *se* $x.f^n \in \mathcal{P}_p$

P13. $\{x \neq y, y \neq x\} \cup \mathcal{C} \mapsto \{x \neq y\} \cup \mathcal{C}$ *se* $x \not\equiv y$

P14. $\{x \neq v, x \neq v\} \cup \mathcal{C} \mapsto \{x \neq v\} \cup \mathcal{C}$

P15. $\{\neg x.f^n, \neg x.f^n\} \cup \mathcal{C} \mapsto \{\neg x.f^n\} \cup \mathcal{C}$

Dem. A redução da conjunção de restrições atómicas positivas de $\mathcal{C}$ a forma resolvida normal garante que $\mathcal{C} \approx_{\mathcal{RT}} \mathcal{P}_p \wedge \mathcal{C}''$, onde $\mathcal{C}''$ é uma conjunção de restrições atómicas negativas, ou $\mathcal{C} \approx_{\mathcal{RT}} \perp$. No primeiro caso, vejamos que as regras P3–P15 são correctas em $\mathcal{RT}$, terminam e que as restrições irredutíveis ou são $\perp$ ou formas primárias. Como na Proposição 4.4, as regras P3–P7 e P13–P15 são correctas em qualquer álgebra de $\mathcal{L}$. A correcção das restantes resulta dos axiomas de $\mathcal{RT}$, da definição das restrições, como abreviaturas de fórmulas de $\mathcal{L}$, e do facto de $\mathcal{P}_p$ estar em forma normal. A análise da terminação é análoga à efectuada para a Proposição 4.4. Nenhuma regra faz aumentar o número de variáveis nem o número de restrições. As regras P4, P7 e

P12 fazem terminar. A regra P3 só se aplica se $\top \in \mathcal{C}$ inicialmente (logo, em número finito). As regras P8–P11 e P13–P15 fazem diminuir o número de restrições. As regras P6, P8–P11 e P13–P15 aplicam-se um número finito de vezes antes da aplicação da regra P5 e, portanto, se esta regra não se aplicar o processo termina. Ora, como $\mathcal{P}_p$ está em forma resolvida normal existe no máximo uma restrição do tipo $x = v$ para cada variável que ocorre em $\mathcal{C}$ e x não ocorre mais em $\mathcal{P}_p$. Assim , a regra P5 só se pode aplicar um número finito de vezes. Finalmente, é imediato que se para uma restrição, diferente de $\perp$, alguma das condições na definição de forma primária não se verifica, a restrição não é irredutível. 2

Sendo $\mathcal{C} \equiv \mathcal{C}_p \ \wedge \ \mathcal{C}_n$ uma conjunção de restrições atómicas e supondo que $\mathcal{C}_p$ está em forma resolvida normal, o sistema de reescrita P tem complexidade linear no número de restrições atómicas de $\mathcal{C}$.

Corolário 4.32 *Seja $\mid \Sigma \mid = \infty$. Se $\mathcal{C}$ é uma conjunção de restrições atómicas e $\mathcal{P}_p \ \wedge \ \mathcal{P}_n$ é uma forma primária de $\mathcal{C}$ obtida pelo algoritmo da Proposição 4.31, então $\mathcal{P}_p \ \wedge \ \mathcal{P}_n$ é única a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{P}_p}$.*

Corolário 4.33 *Seja $\mid \Sigma \mid = \infty$. Uma conjunção $\mathcal{C}$ de restrições atómicas é não satisfazível em $\mathcal{RT}$ se e só se $\mathcal{C} \overset{\mathsf{P}}{\mapsto}^{\star} \perp$.*

Aplicando o sistema de reescrita P à restrição

$$(\mathsf{c}_{12}) \qquad \begin{array}{l} x.f_1^2 \doteq w \ \wedge \ y.f_1^2 \doteq z \ \wedge \ w.g_1^1 \doteq u \ \wedge \ z.g_1^1 \doteq a \ \wedge \ u = a \\ \wedge \ x \neq y \ \wedge \ y \neq w \ \wedge \ y \neq a \ \wedge \ \neg u.f^2 \end{array}$$

obtemos a forma primária

$$(\mathsf{c}_{13}) \qquad \begin{array}{l} x.f^2 \ \wedge \ x.f_1^2 \doteq w \ \wedge \ y.f^2 \ \wedge \ y.f_1^2 \doteq w \ \wedge \ w.g^1 \ \wedge \ w.g_1^1 \doteq a \ \wedge \ z = w \ \wedge \ u = a \\ \wedge \ x \neq y \ \wedge \ y \neq w \end{array}$$

Por outro lado, a restrição

$$(\mathsf{c}_{14}) \qquad x.f_1^1 \doteq z \ \wedge \ z.f_1^1 \doteq x \ \wedge \ x \neq z$$

reduz-se pelo sistema de reescrita P a $\perp$.

Apesar da forma primária obtida pelo sistema de reescrita P ser única, pode conter restrições *redundantes*.

Definição 4.34 *Sendo $\mathcal{C}$ uma conjunção de restrições diz-se que $c_1 \in \mathcal{C}$, é* redundante *em $\mathcal{C}$, se $\mathsf{Sol}(\mathcal{C}, \mathcal{RT}) = \mathsf{Sol}(\mathcal{C} \setminus \{c_1\}, \mathcal{RT})$. Se $c_1 \notin \mathcal{C}$ diz-se que c_1 é redundante em $\mathcal{C}$ se c_1 é redundante em $\mathcal{C} \wedge c_1$.*

Pela definição anterior, temos que uma forma normal $\mathcal{M}$ pode conter restrições redundantes do tipo $x.f^n$ desde que contenha alguma do tipo $x.f_i^n \doteq v$. No entanto, a inclusão de restrições do tipo $x.f^n$, na condição (iv) da Definição 4.1 de forma resolvida é apenas uma questão técnica, que facilita a detecção da não satisfazibilidade e, portanto, esse tipo de restrições *não* serão consideradas *redundantes*[8].

Lema 4.35 *Uma forma normal $\mathcal{M}$ não contém restrições redundantes. Em particular, se c é uma restrição atómica positiva e se $\mathcal{M} \models c$, então $c \in \mathcal{M}$ a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{M}}$.*

Dem. Por indução na estrutura de $\mathcal{M}$. 2

Sendo $\mathcal{C}$ uma conjunção de restrições atómicas e $c_1 \notin \mathcal{C}$ uma restrição atómica positiva, se $\mathcal{RT} \not\models \mathcal{C} \wedge \neg c_1$ então c_1 é redundante em $\mathcal{C}$. Em particular, $\mathcal{C} \models_{\mathcal{RT}} c_1$.

Resulta[9] ainda da Proposição 4.27 que:

Proposição 4.36 *Seja $|\ \Sigma\ | = \infty$. Sendo $\mathcal{P} \equiv \mathcal{P}_p \wedge \mathcal{P}_n$ uma forma primária, $n_i \in \mathcal{P}_n$ é redundante em $\mathcal{P}$ se e só se ou $\mathcal{RT} \not\models \mathcal{P}_p \wedge c_i$ ou existe $n_k \in \mathcal{P}_n$, com $i \neq k$, tal que $\mathcal{P}_p \wedge n_k \models_{\mathcal{RT}} \mathcal{P}_p \wedge n_i$, ou equivalentemente, $\mathcal{P}_p \wedge c_i \models_{\mathcal{RT}} c_k$.*

Pela definição, uma forma primária só pode conter restrições redundantes do tipo $x \neq y$. Por exemplo, na forma primária c_{13} a restrição $y \neq w$ é redundante, enquanto que a restrição $y \neq x$ não o é.

Definição 4.37 *Se $\mathcal{C}$ é uma conjunção de restrições satisfazível e $\mathcal{C} \models_{\mathcal{RT}} x \neq y$ dizemos que x e y são variáveis* incompatíveis *em $\mathcal{C}$. Se $\mathcal{C} \equiv \mathcal{C}_p \wedge \mathcal{C}_n$ é uma forma primária, esta noção generaliza-se naturalmente* a valores e classes *de equivalência de $=_{\mathcal{C}_p}$.*

Corolário 4.38 *Seja $|\ \Sigma\ | = \infty$. Sendo $\mathcal{P} \equiv \mathcal{P}_p \wedge \mathcal{P}_n$ uma forma primária e $x, y \in \mathsf{Vars}(\mathcal{P})$, x e y são incompatíveis em $\mathcal{P}$ se e só se $\mathcal{P} \wedge x = y \overset{\mathsf{P}}{\mapsto}^{\star} \bot$.*

[8]Em termos da implementação computacional, essas restrições não alteram a representação duma forma resolvida.

[9]Esta proposição corresponde, para $\mathcal{LU}$, à Proposição 6.2 em [LMM87] para sistemas de equações e inequações considerando o Universo de Herbrand infinito.

Salvo referência em contrário, vamos supor que uma forma primária não contém restrições redundantes.

Note-se que a exigência de numa forma primária, a conjunção de restrições atómicas positivas $\mathcal{C}_p$ estar em forma normal pode ser enfraquecida, atendendo a que ela só é necessária para determinar a satisfazibilidade duma forma primária, quando ocorrem restrições do tipo $x \neq y$. Suponhamos que $\mathcal{C}_p$ é uma forma resolvida. Depois de se aplicar o sistema de reescrita P, excluindo as regras $\mathsf{P1}$ e $\mathsf{P2}$, para cada restrição do tipo $x \neq y$ aplica-se a seguinte versão restrita do algoritmo de identificação de classes distinguíveis (Figura 4.1), para identificação de classes distinguíveis: considera-se na entrada do algoritmo apenas o conjunto de todos os pares de classes $([w]_{\mathcal{C}_p}, [z]_{\mathcal{C}_p})$ para as quais existe p tal que $x\langle p\rangle w \in \mathsf{Paths}(\mathcal{C}_p)$ e $y\langle p\rangle z \in \mathsf{Paths}(\mathcal{C}_p)$. Se algum destes pares for "marcado" (no passo 1 ou no passo 2) conclui-se imediatamente que $x \not\sim_{\mathcal{C}_p} y$ e portanto o algoritmo pode terminar. Se todos esses pares não forem distinguíveis em $\mathcal{C}_p$, a restrição inicial não é satisfazível, caso contrário, é-o.

Este método apresenta as seguintes desvantagens, quando comparado com a obtenção duma forma normal:

- não se obtém uma forma única, e, habitualmente mais simples, para a conjunção de restrições atómicas positivas.
- repetição desnecessária de marcações de pares de classes distinguíveis.

A vantagem é, obviamente, evitar o cálculo da forma resolvida normal.

Vamos ver que em geral numa forma primária (sem redundâncias) as negações não podem ser substituídas por uma disjunção de conjunções de restrições atómicas positivas.

A proposição seguinte é uma consequência da Proposição 4.27.

Proposição 4.39 *Seja $\mid \Sigma \mid = \infty$ e $\mathcal{P}$ uma forma primária, sem negações redundantes. Então $\mathcal{P}$ não é equivalente em $\mathcal{RT}$ a nenhuma disjunção finita de formas resolvidas normais.*

Dem. Provamos por redução ao absurdo.

Suponhamos que $\mathcal{P} \approx_{\mathcal{RT}} \mathcal{M}_1 \vee \ldots \vee \mathcal{M}_n$, onde, para $1 \leq i \leq n$, $\mathcal{M}_i$ é uma forma normal. Seja $\mathcal{P} \equiv \mathcal{P}_p \wedge n_1 \wedge \ldots \wedge n_m$, então:

$$\begin{aligned}
\mathcal{RT} &\models \mathcal{P}_p \wedge n_1 \wedge \ldots \wedge n_m \Leftrightarrow \mathcal{M}_1 \vee \ldots \vee \mathcal{M}_n \\
\mathcal{RT} &\models \mathcal{P}_p \Leftrightarrow (\mathcal{P}_p \wedge c_1) \vee \ldots \vee (\mathcal{P}_p \wedge c_m) \vee \mathcal{M}_1 \vee \ldots \vee \mathcal{M}_n
\end{aligned}$$

onde $c_i \equiv \neg n_i$, para $1 \leq i \leq m$. Pela Proposição 4.27, existe $1 \leq j \leq n$ tal que $\mathcal{P}_p \approx_{\mathcal{RT}} \mathcal{M}_j$ ou existe $1 \leq i \leq m$ tal que $\mathcal{P}_p \approx_{\mathcal{RT}} \mathcal{P}_p \wedge c_i$. No primeiro caso, temos

$$\mathcal{RT} \models \mathcal{M}_j \wedge n_1 \wedge \ldots \wedge n_m \Leftrightarrow \mathcal{M}_j$$

Mas, então para todo $1 \leq i \leq m$, $\mathcal{M}_j \models_{\mathcal{RT}} n_i$ o que é impossível porque $\mathcal{P}_p \approx_{\mathcal{RT}} \mathcal{M}_j$ e $\mathcal{P}$ não contém negações redundantes. No segundo caso, ter-se-ia $\mathcal{P} \approx_{\mathcal{RT}} \perp$ o que também contradiz a hipótese de $\mathcal{P}$ ser uma forma primária. 2

4.3.2 Formas Primárias para Σ Finita

Se a assinatura Σ for finita o Lema da Independência das Restrições Negativas, Lema 4.28, não é válido. O exemplo seguinte ilustra este facto.

Se $\Sigma = \{a, f^1\}$ a conjunção de restrições

$$(\mathsf{c}_{15}) \qquad x \neq a \wedge \neg x.f^1$$

não é satisfazível em $\mathcal{RT}$, embora cada uma das restrições o seja.

O *Axioma do Domínio Fechado* DCA, necessário para garantir a completa axiomatização de $\mathcal{RT}$ para Σ finita, pode-se enunciar em $\mathcal{LU}$ da seguinte forma:

$$\forall x. \; (\bigvee_{a \in \Sigma} x = a) \vee (\bigvee_{f^n \in \Sigma} x.f^n) \tag{4.1}$$

Dada uma restrição $c \in \mathcal{LU}$ e $x \in \mathsf{Vars}(c)$, seja $\mathsf{DCA}(x) \in \mathcal{LU}$ a restrição

$$\mathsf{DCA}(x) \equiv (\bigvee_{a \in \Sigma} x = a) \vee (\bigvee_{f^n \in \Sigma} x.f^n)$$

Então

$$E_{\mathcal{RT}} \cup \{\mathsf{DCA}\} \models c \Leftrightarrow c \wedge \bigwedge_{x \in \mathsf{Vars}(c)} \mathsf{DCA}(x) \tag{4.2}$$

Utilizando a equivalência (4.2), o exemplo c_{15} é equivalente em $\mathcal{RT}$ a

$$(\mathsf{c}_{16}) \qquad x \neq a \wedge \neg x.f^1 \wedge (x = a \vee x.f^1)$$

Transformando c_{16} em forma normal disjuntiva e usando para cada elemento da disjunção, o sistema de reescrita P, obtém-se $\perp$.

Em certos casos, a equivalência (4.2) permite substituir restrições negativas por disjunções de conjunções de restrições atómicas positivas.

Proposição 4.40 *Supondo que Σ é finita e que apenas contém constantes, qualquer conjunção $\mathcal{C}$ de restrições atómicas é equivalente em $\mathcal{RT}$ (ou em $\mathcal{T}$) a uma disjunção de conjunções de restrições atómicas positivas. Em particular, $\mathcal{C}$ é satisfazível se e só se pelo menos uma das conjunções o for.*

Dem. Suponhamos que $\mathcal{C}$ só contém restrições atómicas negativas. Aplicando (4.2) a $\mathcal{C}$ e transformando a restrição obtida em[10] forma normal disjuntiva, resulta que cada elemento da disjunção é da forma $\mathcal{C} \wedge \mathcal{C}_i$, onde $\mathcal{C}_i$ é uma conjunção de restrições do tipo $x = a$, uma para cada $x \in \mathsf{Vars}(\mathcal{C})$ e algum $a \in \Sigma$. Então, cada $\mathcal{C}_i$ está em forma normal. Usando o sistema de reescrita P da Proposição 4.31, que é correcto em $\mathcal{RT}$ com Σ finita, é imediato que cada conjunção é equivalente a $\perp$ ou a uma conjunção de restrições do tipo $x = a$. Note-se que após a aplicação da regra P5 as restrições negativas ou são triviais.

Se $\mathcal{C}$ contiver restrições atómicas positivas, depois de se aplicar (4.2) e se transformar em forma normal disjuntiva, para cada elemento $\mathcal{C} \wedge \mathcal{C}_i$ da disjunção, é necessário obter a forma normal da conjunção de restrições atómicas positivas ou $\perp$. Essa forma normal, se existir, é $\mathcal{C}_i$ e a demonstração prossegue como no caso anterior. 2

Se Σ contiver pelo menos um functor, $\mathcal{RT}(\Sigma)$ é infinito e, em geral, não vai ser possível "substituir" restrições negativas por disjunções de conjunções de restrições atómicas positivas.

Sendo $\Sigma = \{a, f^1\}$ a restrição $x \neq y$ é equivalente em $\mathcal{RT}$ a

$$(\mathsf{c}_{17}) \qquad (x = a \wedge y.f^1) \vee (x.f^1 \wedge y = a) \vee (x.f^1 \wedge y.f^1 \wedge x \neq y)$$

e a negação não pode ser eliminada.

No que se segue supomos, a menos referência em contrário, que Σ contém pelo menos um functor e outro símbolo. Isto, porque se Σ contém apenas duas ou mais constantes, a caracterização de conjunções de restrições atómicas foi feita na Proposição 4.40.

A definição de formas primárias é modificada para traduzir a aplicação do Axioma DCA.

Definição 4.41 *Sendo $2 \leq |\Sigma| < \infty$, uma conjunção de restrições $\mathcal{C}$ está em (ou é uma)* forma primária *se é $\top$ ou se satisfaz as seguintes condições:*

(i) toda a restrição em $\mathcal{C}$ é do tipo $x.f^n$, $x.f_i^n \doteq v$, $x = v$ ou $x \neq y$ e ocorre uma só vez em $\mathcal{C}$;

[10] Ver Teorema 4.58.

(ii) *a conjunção de todas as restrições atómicas positivas de $\mathcal{C}$ está em forma resolvida normal e denota-se por $\mathcal{C}_p$;*

(iii) *se $x = v \in \mathcal{C}_p$ então x ocorre exactamente uma vez em $\mathcal{C}$;*

(iv) *para cada $x \in \mathsf{Vars}(\mathcal{C})$ ou x é resolvida em $\mathcal{C}_p$ ou $x.\tau \in \mathcal{C}_p$ para algum $\tau \in \Sigma$;*

(v) *se $x \neq y \in \mathcal{C}$ então $y \neq x \notin \mathcal{C}$ e $x \not\equiv y$.*

É de notar que numa forma primária para Σ finita não ocorrem variáveis livres.

Nas demonstrações seguintes serão usados os conceitos de "profundidade" em árvores a seguir definidos, e que são análogos aos apresentados em [Mah88] e [JL87]. Em particular, a profundidade duma árvore finita t corresponde ao maior comprimento duma posição de t mais um. Formalmente,

Definição 4.42 *A* profundidade *duma árvore $t \in \mathcal{T}$, $\mathcal{D}(t)$ é definida indutivamente por:*

(i) *$\mathcal{D}(t) = 1$ se $t(\epsilon)$ é uma constante;*

(ii) *$\mathcal{D}(t) = 1 + \max\{\mathcal{D}(t|_i) \,|$para todo $1 \leq i \leq n,$ sendo n a aridade de $t(\epsilon)\}$.*

Dado $t \in \mathcal{RT}$ (ou $t \in \mathcal{T}$) diz-se que $\tau \in \Sigma$ aparece à profundidade $d+1$ em t *se existe α tal que $t(\alpha) = \tau$ e $|\alpha| = d$.*

Seja $f^n \in \Sigma$ e a um elemento de Σ de aridade arbitrária, diferente de f^n. Definimos $G(h) \equiv F_h(a, \ldots, a) \in \mathcal{RT}$ como a árvore uniforme[11] de profundidade $h+1$, n^h folhas etiquetadas por a e nós internos com functor f^n, *i.e*, $G(h)(\alpha) = a$ se $|\alpha| = h$ e $G(h)(\alpha) = f^n$ se $0 \leq |\alpha| < h$. Por exemplo, para $n = 2$ e $h = 2$, $G(2) \equiv f(f(a,a), f(a,a))$.

Proposição 4.43 *Se $2 \leq |\Sigma| < \infty$ e $\mathcal{C}$ é uma forma primária, então $\mathcal{C}$ é satisfazível em $\mathcal{RT}$.*

Dem. Sejam $\mathcal{C} \equiv \mathcal{C}_p \wedge \mathcal{C}_n$, onde $\mathcal{C}_p$ é uma forma normal, $\mathcal{C}_n \equiv \top$ ou $\mathcal{C}_n \equiv n_1 \wedge \ldots \wedge n_k$, onde para $1 \leq i \leq k$ n_i é do tipo $x \neq y$ com $x, y \in \mathsf{Vars}(\mathcal{C}_p)$. Se $\mathcal{C}_n \equiv \top$ está provado. Para cada $x \neq y \in \mathcal{C}_n$, x e y pertencem a classes funcionais. Se pertencerem a classes funcionais com functores distintos a restrição $x \neq y$ pode ser eliminada. Podemos então supor que se $x \neq y \in \mathcal{C}_n$, x e y pertencem a classes funcionais com o mesmo functor mas distinguíveis em $\mathcal{C}_p$. Dado que $|\mathsf{Vars}(\mathcal{C}_n)|$ é finito, e Σ contém pelo menos um functor e outro símbolo, por exemplo f^n e a, é possível construir $\rho\theta_{\mathcal{C}_p}$ solução de $\mathcal{C}_p \wedge \mathcal{C}_n$.

[11] Os termos $G(h)$ são também usados em [Mah88] e [JL87] na demonstração de teoremas análogos.

A ideia da demonstração é a seguinte. Como $\mathcal{RT}(\Sigma)$ é infinito é possível construir árvores "suficientemente" profundas e todas distintas que assegurem valores diferentes para todas as variáveis que ocorrem em restrições negativas. A demonstração que se segue é baseada do método apresentado por Maher em [Mah88], para a demonstração do Lema 19.

Sendo $m = |\mathsf{Vars}(\mathcal{C}_n)|$ tem-se que $k \leq \frac{m(m-1)}{2}$. Cada $n_i \in \mathcal{C}_n$ é do tipo $x_d \neq x_e$, $1 \leq d, e \leq m$, $d \neq e$ e pelo Lema 4.15 existem p_i, v_d^i e v_e^i tal que $x_d\langle p_i\rangle v_d^i$, $x_e\langle p_i\rangle v_e^i \in \mathsf{Paths}(\mathcal{C}_p)$ e, as classes $[v_d^i]_{\mathcal{C}_p}$ e $[v_e^i]_{\mathcal{C}_p}$ satisfazem uma das seguintes condições:

(i) pelo menos uma é constante;

(ii) são classes funcionais com functores distintos;

(iii) são classes funcionais com o mesmo functor mas pelo menos uma delas não é definida.

Nos dois primeiros casos qualquer atribuição de variáveis que satisfaça $\mathcal{C}_p$ também satisfaz n_i.

Sejam $w_1, \ldots, w_l$, para algum $1 \leq l \leq m$, as variáveis não definidas em $\mathcal{C}_p$ que incluem os valores $v_e^i, v_d^i \in \mathsf{Vars}(\mathcal{C})$, para $1 \leq i \leq k$, que verifiquem a terceira condição acima. Para cada $1 \leq j \leq l$, suponhamos que $w_j.f^{n_j} \in \mathcal{C}_p$ e que w_j não é definida em $\mathcal{C}_p$ para $1 \leq r_1^j < \ldots < r_{l_j}^j \leq n_j$, com $l_j \leq n_j$. Seja $M = \max\{n_j \mid 1 \leq j \leq l\}$. Seja $h > 0$. Definimos uma atribuição de variáveis ρ tal que para cada $1 \leq j \leq l$ tem-se $\rho(w_j) = t_{w_j}$ e para cada $o = r_1^j, \ldots r_{l_j}^j$, $t_{w_j}|_o = G(h \times (M \times j + o))$. Isto garante que para todo $1 \leq i \leq k$, $\mathcal{RT} \models_{\rho\theta_{\mathcal{C}_p}} n_i$.

Então, a atribuição de variáveis $\rho\theta_{\mathcal{C}_p}$ satisfaz $\mathcal{C}$. 2

Dado que uma forma primária para Σ finita é uma forma primária para Σ infinita, a referência ao cardinal de Σ será omitida, sempre que isso seja explícito pelo contexto.

Na Definição 4.41 de forma primária para Σ finita, para garantir a satisfazibilidade, bastava exigir a aplicação do Axioma DCA apenas às variáveis que ocorrem em negações, isto é, a condição *(iv)* podia ser substituída pela condição seguinte:

$(iv)_a$ *para cada* $x \in \mathsf{Vars}(\mathcal{C}_n)$, $x.\tau \in \mathcal{C}_p$ *para algum* $\tau \in \Sigma$.

Nesse caso, a Proposição 4.45, a seguir apresentada, ainda seria válida. O uso da condição *(iv)* da Definição 4.41 apenas garante que não existem variáveis livres numa forma primária, o que simplificou a demonstração da Proposição 4.43.

Resulta da Proposição 4.43, que uma conjunção de restrições não triviais do tipo $x \neq y$ é sempre satisfazível em $\mathcal{RT}$, se Σ contiver pelo menos um functor e outro símbolo. O corolário seguinte, justifica a afirmação feita para o exemplo c_{17}.

Corolário 4.44 *Seja* $2 \leq \mid \Sigma \mid < \infty$ *. Qualquer conjunção de restrições atómicas negativas não triviais do tipo* $x \neq y$ *é satisfazível em* $\mathcal{RT}$*, mas não é equivalente em* $\mathcal{RT}$ *a nenhuma disjunção finita de formas resolvidas normais.*

Dem. Seja $\mathcal{C} \equiv n_1 \ \wedge \ \ldots \ \wedge \ n_k$ e $m = |\mathsf{Vars}(\mathcal{C})|$. Provamos a segunda parte por redução ao absurdo. Suponhamos que

$$\mathcal{C} \approx_{\mathcal{RT}} \exists \bar{z}. \ (\mathcal{M}_1 \ \vee \ \ldots \ \vee \ \mathcal{M}_n)$$

onde $\bar{z} \notin \mathsf{Vars}(\mathcal{C})$ e para cada $1 \leq i \leq n$, $\mathcal{M}_i$ é uma forma normal e $\mathsf{Vars}(\mathcal{M}_i) \setminus \{\bar{z}\} \subseteq \mathsf{Vars}(\mathcal{C})$.

Para cada $x \neq y \in \mathcal{C}$ e cada $1 \leq i \leq n$ existe p_i tal que $x\langle p_i \rangle v_i$, $y\langle p_i \rangle v_i' \in \mathsf{Paths}(\mathcal{M}_i)$ e as classes $[v_i]_{\mathcal{M}_i}$ e $[v_i']_{\mathcal{M}_i}$ são constantes distintas ou funcionais com functores distintos. Suponhamos que Σ contém pelo menos um functor f^n e outro símbolo. Para cada $1 \leq i \leq n$ e $x \in \mathsf{Vars}(\mathcal{C})$ seja

$$\mathcal{D}_i(x, f^n) \quad = \quad \begin{cases} \min\{|p| + 1 \mid x\langle p \rangle v \in \mathsf{Paths}(\mathcal{M}_i), \text{ e a classe } [v]_{\mathcal{M}_i} \\ \qquad\qquad \text{é constante } \tau \text{ ou o seu functor é } \tau, \\ \qquad\qquad \text{com } \tau \not\equiv f^n\} \\ 1 \qquad \text{se o conjunto anterior for vazio} \end{cases}$$

Seja

$$H \quad = \quad \max \bigcup_{1 \leq i \leq n} \ \bigcup_{x \in \mathsf{Vars}(\mathcal{C})} \mathcal{D}_i(x, f^n)$$

Então, a atribuição de variáveis ρ tal que $\rho(x_j) = G(H \times j)$, para cada $1 \leq j \leq m$, é solução de $\mathcal{C}$ e não o é de nenhum $\mathcal{M}_i$, com $1 \leq i \leq n$.

2

Proposição 4.45 *Seja* $2 \leq \mid \Sigma \mid < \infty$*. Qualquer conjunção* $\mathcal{C} \equiv \mathcal{C}_p \ \wedge \ \mathcal{C}_n$ *de restrições atómicas positivas e negativas é equivalente em* $\mathcal{RT}$ *a uma disjunção de formas primárias ou a* $\perp$*.*

Dem. Como anteriormente, podemos supor que $\mathcal{C}_p$ está em forma normal, porque se tal não for possível $\mathcal{C}$ é não satisfazível em $\mathcal{RT}$. Notando que as regras da Proposição 4.31 são correctas em $\mathcal{RT}$ para $2 \leq \mid \Sigma \mid < \infty$, podemos transformar $\mathcal{C}$ em $\perp$ ou numa forma primária (para Σ infinita) equivalente, $\mathcal{C}'_p \wedge \mathcal{C}'_n$ com $\mathcal{C}'_n \equiv n_1 \wedge \ldots \wedge n_m$, onde para cada $1 \leq i \leq m$, n_i é do tipo $x \neq v$ ou $\neg x.f^n$. No primeiro caso, vem

$$\begin{array}{rcl} \mathcal{RT} \models \mathcal{C} & \Leftrightarrow & \mathcal{C}'_p \wedge \mathcal{C}'_n \\ & \Leftrightarrow & \mathcal{C}'_p \wedge \mathcal{C}'_n \wedge (\bigwedge_{x \in \mathsf{Vars}(\mathcal{C}'_p \wedge \mathcal{C}'_n)} \mathsf{DCA}(x)) \\ & \Leftrightarrow & \bigvee_{1 \leq k \leq |\Sigma|^l} \mathcal{C}'_p \wedge \mathcal{C}'_n \wedge c^1_k \wedge \ldots \wedge c^l_k \end{array}$$

onde $l = |\mathsf{Vars}(\mathcal{C}'_p \wedge \mathcal{C}'_n)|$ e c^j_k é uma restrição do tipo $x = \tau$ ou $x.\tau$, para cada $1 \leq j \leq l$, cada $x \in \mathsf{Vars}(\mathcal{C}'_p \wedge \mathcal{C}'_n)$ e algum $\tau \in \Sigma$. Para cada $1 \leq k \leq |\Sigma|^l$, podemos reduzir $\mathcal{C}'_p \wedge c^1_k \wedge \ldots \wedge c^l_k$ a forma normal ou a $\perp$. No primeiro caso, aplicando mais uma vez as regras da Proposição 4.31 obtém-se uma forma primária ou $\perp$. Basta provar que cada uma dessas restrições é uma forma primária para Σ finita. Se for uma forma normal, verifica a condição (iv) e está demonstrado. Caso contrário, seja $\mathcal{C}^k \equiv \mathcal{C}''_p \wedge \mathcal{C}''_n$ uma dessas formas primárias. Todas as classes de equivalência da relação $=_{\mathcal{C}''_p}$ ou são constantes ou são funcionais. Se $x \in \mathsf{Vars}(\mathcal{C}''_n)$ então $x \in \mathsf{Vars}(\mathcal{C}''_p)$ e x não é livre em $\mathcal{C}''_p$. Então, $\mathcal{C}_n$ é constituído por negações do tipo $x \neq y$, onde x e y pertencem a classes funcionais que podemos supor com o mesmo functor, mas que são distinguíveis em $\mathcal{C}''_p$. Assim, são verificadas as condições *(i)* e *(iv)* da definição de formas primárias para Σ finita. As restantes condições resultam de $\mathcal{C}''_p \wedge \mathcal{C}''_n$ ser uma forma primária para Σ infinita.

Em particular, $\mathcal{C}$ não é satisfazível se e só se todos os elementos da disjunção se reduzirem a $\perp$. 2

Exemplificamos a seguir a aplicação desta proposição.

Sendo $\Sigma = \{a, f^1\}$, a restrição

(c18) $\quad x.f^1_1 \doteq a \wedge x \neq y$

é, aplicando o algoritmo da Proposição 4.45 e factorizando elementos comuns, equivalente à restrição

(c19) $\quad x.f^1 \wedge x.f^1_1 \doteq a \wedge (y = a \vee (y.f^1 \wedge x \neq y))$

Corolário 4.46 *Sendo $2 \leq |\Sigma| < \infty$, o problema de determinar a satisfazibilidade duma conjunção de restrições atómicas é* **NP***-completo.*

Dem. Seja $\mathcal{C} \equiv \mathcal{C}_p \wedge \mathcal{C}_n$ uma conjunção de restrições atómicas e suponhamos que $\mathcal{C}_p$ está em forma resolvida normal e $l = |\mathsf{Vars}(\mathcal{C})|$. Pela Proposição 4.45, para decidir a satisfazibilidade de $\mathcal{C}$ basta considerar o seguinte método. Para cada $x_i \in \mathsf{Vars}(\mathcal{C})$, com $1 \leq i \leq l$, escolha-se um $\tau \in \Sigma$ e considere-se a restrição c_i do tipo $x = \tau$ ou $x.\tau$. Usando o sistema de reescrita P, pode-se reduzir $\mathcal{C} \wedge c_1 \wedge \ldots \wedge c_l$ a forma primária ou a $\perp$. A restrição $\mathcal{C}$ é satisfazível se e só se alguma destas restrições admite forma primária. Todo este processo pode ser executado em tempo não-determinístico polinomial.

Para provar que é **NP-hard** exibimos uma redução polinomial (linear) ao problema da colorir um grafo com 3 cores, que é um problema **NP**-completo [GJ79]. Este último problema pode ser enunciado da seguinte forma:

(3-COLOR)

Instância: Um grafo não dirigido $G = (V, E)$.

Questão: É possível colorir G com 3 cores, isto é, existe uma função

$$\Phi : V \longrightarrow \{1, 2, 3\}$$

tal que se $(x, y) \in E$ então $\Phi(x) \neq \Phi(y)$?

Dada uma instância de **3-COLOR** vamos construir Σ e uma restrição $\mathcal{C}$, tal que $\mathcal{C}$ tem solução em $\mathcal{RT}$ se e só se essa instância tem solução. Considere-se $\Sigma = \{1, 2, 3\}$ e para cada $(x, y) \in E$, $x \neq y \in \mathcal{C}$. Temos que $\mathsf{Vars}(\mathcal{C}) \subseteq V$ e $\mathcal{C}$ é uma conjunção de restrições atómicas, cujo tamanho é linear no tamanho da representação da instância de **3-COLOR**. 2

4.3.2.1 Eliminação de Negações

O problema de determinar se uma conjunção $\mathcal{C}$ de restrições atómicas é equivalente a uma disjunção de conjunções de restrições atómicas positivas é resolvido pelo algoritmo da Proposição 4.45, se as restrições negativas em $\mathcal{C}$ são apenas do tipo $x \neq a$ ou $\neg x.f^n$ (ou reduzidas a esse tipo pela aplicação de alguma regra).

Por exemplo, se $\Sigma = \{a, f^1\}$ a restrição

$$(\mathsf{c}_{20}) \qquad y.f^1_1 \doteq x \wedge (\neg x.f^1 \vee (x.f^1_1 \doteq x_0 \wedge x_0 \neq a))$$

é reduzida a

$$(\mathsf{c}_{21}) \qquad y.f^1_1 \doteq x \ \wedge \ (x = a \ \vee \ (x.f^1_1 \doteq x_0 \ \wedge \ x_0.f^1))$$

Por sua vez, as restrições do tipo $x \neq y$ que ocorram numa disjunção de formas primárias podem ser eliminadas se forem redundantes em cada elemento da disjunção, usando o método do Corolário 4.38, uma vez que o sistema de reescrita P é correcto para assinaturas finitas.

No que segue e a menos especificação em contrário, vamos supor que uma forma primária não contém redundâncias.

Considerem-se, para $\Sigma = \{a, f^1\}$, as seguintes restrições:

$$(\mathsf{c}_{22}) \qquad x.f^1 \ \wedge \ x.f^1_1 \doteq a \ \wedge \ (y = a \ \vee \ (y.f^1 \ \wedge \ (y.f^1_1 \doteq y_1 \ \wedge \ y \neq x)))$$

$$(\mathsf{c}_{23}) \qquad x.f^1 \ \wedge \ x.f^1_1 \doteq a \ \wedge \ (y = a \ \vee \ (y.f^1 \ \wedge \ (y.f^1_1 \doteq y_1 \ \wedge \ y_1.f^1)))$$

Reduzindo a restrição c_{22} a formas primárias, pela Proposição 4.45, concluímos que as restrições c_{22} e c_{23} são equivalentes em $\mathcal{RT}$. Por outro lado, tem-se que $(\exists y_1 . \mathsf{c}_{22}) \approx_{\mathcal{RT}} \mathsf{c}_{19}$ e para cada atribuição de variáveis $\rho \in \mathsf{Sol}_{\mathsf{Vars}(\mathsf{c}_{19})}(\mathsf{c}_{19}, \mathcal{RT})$ existe uma única extensão a y_1 que é solução de c_{22}. Isto é, dada uma restrição $c \in \mathcal{LU}$ e introduzindo novas variáveis pode ser possível obter uma restrição c_1 sem negações e cujas soluções restritas às variáveis de c, coincidem com as de c.

Um algoritmo para obter uma tal forma, quando existe, será apresentado a seguir.

Recordemos, ver página 77, que

Definição 4.47 *Dada uma restrição $c \in \mathcal{LU}$, uma variável $x \in \mathsf{Vars}(c)$ é* determinada *em $c \in \mathcal{LU}$ se existe $t_x \in \mathcal{RT}$ tal que para toda a solução $\rho \in \mathsf{Sol}(c, \mathcal{RT})$, $\rho(x) = t_x$. Se t_x é finita diz-se que x é* determinada finita *em c. Se x é uma variável determinada finita o seu tamanho $\|x\|$ define-se como sendo o tamanho $|t_x|$ de t_x. Estas noções generalizam-se naturalmente a* valores. *Para uma forma resolvida $\mathcal{M}$, a classe $[v]_{\mathcal{M}}$ é* determinada *se v o for.*

Iremos necessitar de alguns lemas sobre formas resolvidas (normais) que são válidos em $\mathcal{RT}$ para qualquer assinatura Σ não singular, e que serão usados também em secções posteriores.

O lema seguinte é uma aplicação directa do Axioma $\mathsf{A6}$ de $E_{\mathcal{RT}}$.

Lema 4.48 *Seja Σ não singular e $\mathcal{M}$ uma forma resolvida. Todas as variáveis $x \in \mathsf{Vars}(\mathcal{M})$ são determinadas em $\mathcal{M}$ se e só se para todo $x \in \mathsf{Vars}(\mathcal{M})$, a classe $[x]_{\mathcal{M}}$ ou é constante ou é definida.*

Dem. ($\Rightarrow$)

Se $x \in \mathsf{Vars}(\mathcal{M})$ e a classe $[x]_{\mathcal{M}}$ é livre ou funcional não definida é óbvio que x não é determinada em $\mathcal{M}$.

($\Leftarrow$)

Suponhamos que para toda a variável $x \in \mathsf{Vars}(\mathcal{M})$, a classe $[x]_{\mathcal{M}}$ é constante ou definida. Neste caso, $\mathcal{M}$ é equivalente a uma forma resolvida racional de $\mathcal{L}$ cujas variáveis são as de $\mathcal{M}$ e onde todas as variáveis ocorrem do lado esquerdo duma equação. Logo, pelo Axioma A6 de $E_{\mathcal{RT}}$, todas as variáveis são determinadas.

2

A demonstração dos lemas a seguir apresentados é trivial.

Lema 4.49 *Sendo $\mathcal{M}$ é uma forma normal, $x \in \mathsf{Vars}(\mathcal{M})$ é determinada finita em $\mathcal{M}$ se e só se $\mathsf{Paths}_x(\mathcal{M})$ é finito e para cada caminho maximal $x\langle p\rangle v \in \mathsf{Paths}(\mathcal{M})$, a classe $[v]_{\mathcal{M}}$ é constante.*

Lema 4.50 *Sendo $\mathcal{M}$ uma forma normal, $x \in \mathsf{Vars}(\mathcal{M})$ é não determinada em $\mathcal{M}$ se e só se existem p e v tal que $x\langle p\rangle v \in \mathsf{Paths}(\mathcal{M})$ e a classe $[v]_{\mathcal{M}}$ é livre ou funcional não definida.*

Lema 4.51 *Seja $\mathcal{M}$ uma forma normal e x e y variáveis distinguíveis e definidas em $\mathcal{M}$. Se $\mathcal{M} \cup \{x = y\}$ é satisfazível em $\mathcal{RT}$ então $\mathcal{M} \models_{\mathcal{RT}} x = y \Leftrightarrow \mathcal{C}_i$ onde $\mathcal{C}_i$ é a conjunção de todas as restrições do tipo $v_i = v_i'$, com v_i e v_i' distintas não definidas em $\mathcal{M}$ e existe $p_i \not\equiv \epsilon$ tal que $x\langle p_i\rangle v_i,\ y\langle p_i\rangle v_i' \in \mathsf{Paths}(\mathcal{M})$.*

Dem. Pelo Corolário 4.16. 2

Lema 4.52 *Seja $\mathcal{M}$ uma forma normal e $x,\, y \in \mathsf{Vars}(\mathcal{M})$ determinadas em $\mathcal{M}$. Então $\mathcal{M}$ determina $x = y$, isto é, ou $x \sim_{\mathcal{M}} y$ ou $x \neq y$ é redundante em $\mathcal{M}$.*

Lema 4.53 *Seja $2 \leq |\ \Sigma\ | < \infty$ e $\mathcal{P} \equiv \mathcal{P}_p\ \wedge\ \mathcal{P}_n$ uma forma primária. Então, para todas as restrições $x \neq y \in \mathcal{P}_n$ e para todo p tal que $x\langle p\rangle v,\, y\langle p\rangle v' \in \mathsf{Paths}(\mathcal{P}_p)$ tem-se que $v \equiv v'$ ou v e v' são distinguíveis em $\mathcal{P}_p$.*

Lema 4.54 *Seja $2 \leq |\ \Sigma\ | < \infty$ e $\mathcal{P} \equiv \mathcal{P}_p\ \wedge\ \mathcal{P}_n$ uma forma primária. Para $x \neq y \in \mathcal{P}_n$ se existe p tal que $p \in \mathsf{Paths}_x(\mathcal{P}_p)$ e $p \notin \mathsf{Paths}_y(\mathcal{P}_p)$ então existem q, v e v' tal que $x\langle q\rangle v,\ y\langle q\rangle v' \in \mathsf{Paths}(\mathcal{P}_p)$ e as classes $[v]_{\mathcal{P}_p}$ e $[v']_{\mathcal{P}_p}$ verificam as condições 1, 3 ou 4 do Lema 4.15.*

Lema 4.55 *Seja $2 \leq |\Sigma| < \infty$ e $\mathcal{P} \equiv \mathcal{P}_p \wedge \mathcal{P}_n$ uma forma primária. Se $x \neq y \in \mathcal{P}_n$, a variável x é determinada não finita em $\mathcal{P}_p$ com functor f^n e a variável y é funcional com functor f^n e não é definida em $\mathcal{P}_p$ para nenhum $1 \leq i \leq n$, então $\mathcal{P}$ não é equivalente em $\mathcal{RT}$ a nenhuma disjunção de formas resolvidas normais.*

Dem. Se x é determinada não finita em $\mathcal{P}_p$ existe $w \in \mathsf{Vars}_x(\mathcal{P}_p)$ e $p \not\equiv \epsilon$ tal que $w\langle p\rangle w$. Suponhamos que $w.f^n \in \mathcal{P}_p$ e $x\langle q\rangle w \in \mathsf{Paths}(\mathcal{P}_p)$. Por redução ao absurdo, seja

$$\mathcal{P} \approx_{\mathcal{RT}} \exists\bar{z}.\ (\mathcal{M}_1 \vee \ldots \vee \mathcal{M}_m)$$

onde $\bar{z} \notin \mathsf{Vars}(\mathcal{P})$ e para cada $1 \leq j \leq m$, $\mathcal{M}_j$ é uma forma normal. Então, para cada $1 \leq j \leq m$, seja p_j o menor caminho tal que existem v_j, v'_j e $x\langle p_j\rangle v_j$, $y\langle p_j\rangle v'_j \in \mathsf{Paths}(\mathcal{M}_j)$, e as classes $[v_j]_{\mathcal{M}_j}$ e $[v'_j]_{\mathcal{M}_j}$ são constantes distintas ou funcionais com functores distintos. Seja $M = \max\{|p_j|, 1 \leq j \leq m\}$. Seja t_x o valor de x para qualquer solução de $\mathcal{P}$, $\beta = j_1 \ldots j_k$ e $\alpha = i_1 \ldots i_l$ posições tal que $q = \gamma^1_{j_1} \ldots \gamma^k_{j_k}$ e $p = \tau^1_{i_1} \ldots \tau^l_{i_l}$. Considere-se a atribuição de variáveis ρ tal que $\rho(y) = t_y$ coincide com t_x excepto que $t_y\,|_{\beta\alpha^{M+1}}\,(\epsilon) = a$, e $a \not\equiv f^n$. Então ρ é solução de $\mathcal{P}$ e não o é de nenhum $\mathcal{M}_j$, $1 \leq j \leq m$. 2

Iremos ainda usar o Lema 2.35, que é aqui enunciado como Lema 4.56:

Lema 4.56 *Seja $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$. Para todas as variáveis x, $y \in X$, functor $f^n \in \Sigma$ e valores v_i, v'_i, para $1 \leq i \leq n$, tem-se que*

$$\begin{aligned}(4.3)\qquad \mathcal{A} \models\ & \bigwedge_{i=1}^{n}(x.f^n_i \doteq v_i \ \wedge\ y.f^n_i \doteq v'_i) \ \wedge\ x \neq y \\ \Leftrightarrow\ & \bigwedge_{i=1}^{n}(x.f^n_i \doteq v_i \ \wedge\ y.f^n_i \doteq v'_i) \ \wedge\ \bigvee_{i=1}^{n} v_i \neq v'_i\end{aligned}$$

Na proposição seguinte apresentamos uma caracterização necessária e suficiente para a eliminação de negações numa forma primária. É definido um sistema de reescrita que dada uma forma primária permite obter, se existir, uma disjunção de formas resolvidas normais equivalente. Sendo $\mathcal{P} \equiv \mathcal{P}_p \wedge \mathcal{P}_n$ uma forma primária, se $x \neq y \in \mathcal{P}$ e x e y são variáveis funcionais com um mesmo functor f^n podemos "eliminar" $x \neq y$, usando o lema anterior, se x e y são definidas em $\mathcal{P}_p$ ou se para algum $1 \leq i \leq n$, x é não definida para i em $\mathcal{P}_p$, então $y.f^n_i \doteq v_i \in \mathcal{P}_p$ e v_i é determinada finita (ou vice-versa). Alguns exemplos da aplicação deste método serão apresentados no final desta secção.

Proposição 4.57 *Seja* $2 \leq \mid \Sigma \mid < \infty$*. Sendo* $\mathcal{P} \equiv \mathcal{P}_p \;\wedge\; \mathcal{P}_n$ *uma forma primária (sem redundâncias), existem formas resolvidas normais* $\mathcal{M}_1, \ldots, \mathcal{M}_n$ *tais que,*

$$(4.4) \qquad \mathcal{RT} \models \mathcal{P} \;\Leftrightarrow\; \exists\bar{z}.\,(\mathcal{M}_1 \;\vee\; \ldots \;\vee\; \mathcal{M}_m) \qquad \begin{array}{l} \bar{z} \notin \mathsf{Vars}(\mathcal{P}) \text{ e} \\ \mathsf{Vars}(\mathcal{M}_j) \setminus \{\bar{z}\} \subseteq \mathsf{Vars}(\mathcal{P}) \\ \text{para todo } 1 \leq j \leq m \end{array}$$

se e só se para cada negação $x \neq y \in \mathcal{P}_n$ *tem-se que*[12]*:*

para todo p*,* v *e* v' *tal que* $x\langle p\rangle v,\ y\langle p\rangle v' \in \mathsf{Paths}(\mathcal{P}_p)$ *e as classes* $[v]_{\mathcal{P}_p}$ *e* $[v']_{\mathcal{P}_p}$ *são funcionais com um mesmo functor* f^n*, uma das seguintes condições é verificada:*

(i) $[v]_{\mathcal{P}_p}$ e $[v']_{\mathcal{P}_p}$ *são definidas;*

(ii) se $[v]_{\mathcal{P}_p}$ *é não definida para algum* $1 \leq i \leq n$*, então* $v'.f_i^n \doteq v_i \in \mathcal{P}_p$ *e* v_i *é determinada finita para* i*. Ou vice-versa.*

Dem. ($\Rightarrow$)

Por redução ao absurdo. Seja $x \neq y \in \mathcal{P}$. Suponhamos que existem p, v e v' tal que $x\langle p\rangle v,\ y\langle p\rangle v' \in \mathsf{Paths}(\mathcal{P}_p)$, as classes $[v]_{\mathcal{P}_p}$ e $[v']_{\mathcal{P}_p}$ são funcionais com um mesmo functor f^n, existe $1 \leq i \leq n$ tal que a classe $[v]_{\mathcal{P}_p}$ (ou $[v']_{\mathcal{P}_p}$) é não definida para i em $\mathcal{P}_p$ e para a classe $[v']_{\mathcal{P}_p}$ (ou $[v]_{\mathcal{P}_p}$) verifica-se uma das seguintes condições:

a) $[v']_{\mathcal{P}_p}$ é não definida para i;

b) $v'.f_i^n \doteq v'_i \in \mathcal{P}_p$ e $[v'_i]_{\mathcal{P}_p}$ é não determinada;

c) $v'.f_i^n \doteq v'_i \in \mathcal{P}_p$ e $[v'_i]_{\mathcal{P}_p}$ é determinada não finita.

Provamos por indução na estrutura do caminho p, que $\mathcal{P}$ não é equivalente a nenhuma disjunção de formas resolvidas normais, isto é, que a relação (4.4) não se verifica. Seja $\mathcal{P} \equiv \mathcal{P}' \;\wedge\; x \neq y$.

Base. Seja $p \equiv \epsilon$. Pelo Lema 4.56 e pela definição das restrições em $\mathcal{LU}$ vem

$$\begin{aligned} \mathcal{RT} \;\models\; \mathcal{P}' \;\wedge\; x \neq y \quad &\Leftrightarrow \quad \exists\bar{z}.\,(\mathcal{P}'' \;\wedge\; \bigvee_{k=1}^{n} v_k \neq v'_k) \\ &\Leftrightarrow \quad \exists\bar{z}.\,(\bigvee_{k=1}^{n} \mathcal{P}'' \;\wedge\; v_k \neq v'_k) \end{aligned}$$

onde $\mathcal{P}''$ difere de $\mathcal{P}'$ pela eventual introdução de restrições do tipo $x.f_k^n \doteq z_k$ ou $y.f_k^n \doteq z'_k$, para alguns $1 \leq k \leq n$. Temos os seguintes casos:

[12] *Se existir* $p \in \mathsf{Paths}_x(\mathcal{P}_p)$ *e* $p \notin \mathsf{Paths}_y(\mathcal{P}_p)$*, então pelo Lema 4.54 e supondo que* $x \neq y$ *não é redundante, tínhamos um caso particular do enunciado. Como não há redundâncias mais nenhum caso há a considerar.*

- se se verificar o caso a), então para $k = i$, v_k e v'_k são variáveis novas. Pelo Corolário 4.44, $v_i \neq v'_i$ não é equivalente a uma disjunção de formas resolvidas normais. Como $\mathcal{P}''$ e $v_i \neq v'_i$ são independentes para a satisfazibilidade em $\mathcal{RT}$, a relação (4.4) não se verifica.
- analogamente se prova para o caso b), usando o Lema 4.50 e aplicando um número finito de vezes a equivalência do Lema 4.56.
- se se verificar o caso c), aplica-se o algoritmo da Proposição 4.45 a $\mathcal{P}'' \wedge z_i \neq v'_i$ e na restrição resultante o elemento da disjunção com $z_i.f^n$ está nas condições do Lema 4.55.

Indução. Seja $p \equiv g_j^m q$. Existem $z_1,\ z_2 \in \mathsf{Vars}(\mathcal{P})$ tais que $x.g_j^m \doteq z_1 \in \mathcal{P}_p$, $z_1\langle q\rangle v \in \mathsf{Paths}(\mathcal{P}_p)$, $y.g_j^m \doteq z_2 \in \mathcal{P}_p$ e $z_2\langle q\rangle v' \in \mathsf{Paths}(\mathcal{P}_p)$. Mais uma vez, aplicando o Lema 4.56 a $\mathcal{P}$ vem

$$\mathcal{RT} \models \mathcal{P}' \wedge x \neq y \quad \Leftrightarrow \quad \exists \bar{z}.\ (\mathcal{P}'' \wedge \bigvee_{k=1}^{n} v_k \neq v'_k)$$

onde, para $k = j$ tem-se a inequação $z_1 \neq z_2$. Aplicando a hipótese de indução a z_1 e z_2, conclui-se que (4.4) não se verifica.

($\Leftarrow$)

Seja $\mathcal{P} \equiv \mathcal{P}_p \wedge \mathcal{P}_n$ uma forma primária e $\mathcal{N}$ um conjunto, possivelmente vazio, de restrições atómicas negativas do tipo $x \neq y$. Começamos por definir duas regras do sistema de reescrita E para pares $\langle \mathcal{P}, \mathcal{N}\rangle$:

E1. $\langle \mathcal{P} \cup \{x \neq y\}, \mathcal{N}\rangle \mapsto \langle \exists \bar{z}.\ (\mathcal{P}' \wedge \bigvee_{i=1}^{n} v_i \neq v'_i), \mathcal{N} \cup \{x \neq y\}\rangle$

se $x \neq y \notin \mathcal{N}$ e sendo $x.f^n \in \mathcal{P}_p$ e $y.f^n \in \mathcal{P}_p$, para cada $1 \leq i \leq n$ verifica-se uma das seguintes condições:

(i) $x.f_i^n \doteq v_i \in \mathcal{P}_p$ e $y.f_i^n \doteq v'_i \in \mathcal{P}_p$;

(ii) se x (respectivamente, y) não está definida para i em $\mathcal{P}_p$, $y.f_i^n \doteq v'_i \in \mathcal{P}_p$ ($x.f_i^n \doteq v_i \in \mathcal{P}_p$) e v'_i (v_i) é determinada finita em $\mathcal{P}_p$, então v_i (v'_i) é uma variável z_i que não ocorria em $\mathcal{P}$;

$\mathcal{P}' \equiv \mathcal{P} \cup \bigcup_i \{x.f_i^n \doteq v_i (y.f_i^n \doteq v'_i)\}$ para cada i que verifica a condição (ii); $\bar{z}$ contém as variáveis introduzidas em (ii) ou omite-se $\exists \bar{z}$, se tais variáveis não ocorrem.

E2. $\langle \mathcal{P} \cup \{x \neq y\}, \mathcal{N}\rangle \mapsto \langle \bot, \mathcal{N}\rangle$ se $x \neq y \in \mathcal{N}$

Sendo $c \equiv \exists \bar{z}.(\mathcal{P}_1 \ \vee \ \ldots \ \vee \ \mathcal{P}_k)$ e para $1 \leq j \leq k$, $\mathcal{P}_j$ formas primárias ou $\perp$, podemos generalizar a disjunção e a quantificação a pares $\langle c, \mathcal{N} \rangle$ considerando que

$$\begin{aligned} \langle \mathcal{P}_1 \ \vee \ \mathcal{P}_2, \mathcal{N} \rangle & \stackrel{def}{=} \langle \mathcal{P}_1, \mathcal{N} \rangle \ \vee \ \langle \mathcal{P}_2, \mathcal{N} \rangle \\ \langle \exists \bar{z}.c, \mathcal{N} \rangle & \stackrel{def}{=} \exists \bar{z}.\langle c, \mathcal{N} \rangle \end{aligned}$$

Do mesmo modo podemos generalizar as noções de forma primária, forma normal disjuntiva e forma prenexa.

Assim, sendo $\mathcal{D} \equiv \langle \mathcal{P}_1, \mathcal{N}_1 \rangle \ \vee \ \ldots \ \vee \ \langle \mathcal{P}_k, \mathcal{N}_k \rangle$, define-se a seguinte regra de reescrita[13]:

E3. $\exists \bar{z}.\, \mathcal{D} \quad \mapsto \quad \exists \bar{z}.\, \mathcal{D}'$

se $\mathcal{D} \not\equiv \mathcal{D}'$ e $\mathcal{D}'$ resulta de $\mathcal{D}$ pela aplicação de uma das regras E1 ou E2 a um dos elementos da disjunção, pela transformação em forma prenexa e forma normal disjuntiva e pela redução de cada elemento da disjunção a (disjunção de) formas primárias ou a $\perp$.

Sendo $\mathcal{P}$ uma forma primária aplica-se a regra de reescrita E3 ao par $\langle \mathcal{P}, \top \rangle$.

Analisemos a correcção destas regras em $\mathcal{RT}$. Considerando as soluções da restrição associada a $\mathcal{D}$, a regra E3 é correcta se as restantes regras o forem. A correcção da regra E1 resulta da equivalência (4.3), do Lema 4.56, sendo as condições de aplicação apenas necessárias para garantir a terminação da regra E3. A correcção da regra E2 resulta do Lema 4.51, que garante a redundância de $x \neq y$.

Pretendemos ainda provar que a aplicação da regra de reescrita E3 termina e que se $\exists \bar{z}.\mathcal{D}$ com

$$\mathcal{D} \equiv \langle \mathcal{P}_1, \mathcal{N}_1 \rangle \ \vee \ \ldots \ \vee \ \langle \mathcal{P}_k, \mathcal{N}_k \rangle$$

é irredutível para essa regra, então $\mathcal{P}_1 \ \vee \ \ldots \ \vee \ \mathcal{P}_k$, com $k \geq 0$, é uma disjunção de formas resolvidas normais.

Se houvesse uma derivação infinita ela tinha de envolver a regra E1. Podemos supor, sem falta de generalidade, que a regra E2 é aplicada sempre que possível a seguir à regra E1 e que as várias formas normais são obtidas. Nenhuma das regras envolvidas nessas transformações pode conduzir a derivações infinitas. Em relação à regra E1, vamos ver, que podemos reforçar o controlo da sua aplicação, sem alterar a correcção. Note-se, ainda, que o número total de variáveis introduzidas por aplicação dessa regra é limitado pelo tamanho das variáveis determinadas finitas.

[13] Na realidade esta regra constitui um sistema de reescrita, distinto do das regras E1 e E2, mas para não sobrecarregar a notação iremos identificar o sistema de reescrita E com esta regra.

Seja $\mathcal{P} \equiv \mathcal{P}_p \ \wedge \ n_1 \ \wedge \ \ldots \ \wedge \ n_r$ uma forma primária e $\mathcal{D} \equiv \langle \mathcal{P}, \top \rangle$. Suponhamos que $r = 1$ e $n_1 \equiv x \neq y$. Como $x \neq y$ não é redundante em $\mathcal{P}$ e a regra E2 garante a não existência de ciclos[14], a regra E1 apenas com a condição (i) só se pode aplicar um número finito de vezes antes da condição (ii) se verificar[15]. Pelo Lema 4.53, se apenas a condição (i) se verificar, depois da redução a forma normal disjuntiva cada elemento ou é equivalente a $\perp$ ou está em forma primária. Assim, neste caso, como não são introduzidas variáveis novas, ao fim de um número finito de aplicações da regra E1 obtemos uma disjunção de formas resolvidas normais.

Caso contrário, a primeira vez que a condição (ii) se verifique, temos

$$\text{(4.5)} \qquad \mathcal{D} \ \mapsto^{\star}_{\mathsf{E1}.(i),\mathsf{E2},\mathsf{E3}} \ \langle \mathcal{P}_p \ \wedge \ x_1 \neq y_1, \mathcal{N}_1 \rangle \ \vee \ \ldots \ \vee \ \langle \mathcal{P}_p \ \wedge \ x_h \neq y_h, \mathcal{N}_h \rangle$$

com $h \geq 1$, omitindo o prefixo existencial e considerando que a aplicação da regra E3 apenas reduz a formas "normais".

Suponhamos que a condição (ii) se verifica para o par $\langle \mathcal{P}_p \ \wedge \ x_1 \neq y_1, \mathcal{N}_1 \rangle$, isto é, $x_1.f^n$, $y_1.f^n \in \mathcal{P}_p$ e que para alguns $1 \leq i \leq n$, x_1 (ou y_1) é não definida para i, $y_1.f_i^n \doteq v_i'$ $(x_1.f_i^n \doteq v_i)$ e v_i' (v_i) é determinada finita em $\mathcal{P}_p$.

Sejam u ou x_1 ou y_1, $l + k = n$, u_j variáveis novas para $1 \leq j \leq l$ e w_j ou v_j ou v_j' variáveis determinadas finitas em $\mathcal{P}_p$. Temos, omitindo os quantificadores existenciais,

$$\begin{array}{lll} \mathcal{RT} & \models & \mathcal{P}_p \ \wedge \ x_1 \neq y_1 \\ & \Leftrightarrow & \mathcal{P}_p \ \wedge \ \displaystyle\bigwedge_{i=1}^{l} (u.f_i^n \doteq u_i) \ \wedge \ \bigvee_{i=1}^{n} v_i \neq v_i' \\ & \Leftrightarrow & \displaystyle\bigvee_{i=1}^{k} (\mathcal{P}_p \ \wedge \ \bigwedge_{j=1}^{l} (u.f_j^n \doteq u_j) \ \wedge \ v_i \neq v_i') \ \vee \ \bigvee_{j=1}^{l} (\mathcal{P}_p^j \ \wedge \ u.f_j^n \doteq u_j \ \wedge \ u_j \neq w_j) \end{array}$$

onde $\mathcal{P}_p^j \equiv \mathcal{P}_p \wedge \bigwedge_{i=1, i \neq j}^{l} (u.f_i^n \doteq u_i)$. Para os primeiros k elementos da disjunção torna-se a aplicar a regra E1 se após redução a forma primária ainda contiverem negações (e se não for equivalente a $\perp$).

Provamos por indução no tamanho $\|w_j\|$ de w_j, para $1 \leq j \leq l$, que cada restrição

$$\mathcal{P}_p^j \ \wedge \ u.f_j^n \doteq u_j \ \wedge \ u_j \neq w_j$$

é equivalente a uma disjunção de formas resolvidas normais, as quais são obtidas pelo sistema de reescrita.

[14] Isto é, reduções do tipo $\langle \mathcal{P} \ \wedge \ x \neq y, \mathcal{N} \rangle \mapsto_{\mathsf{E1}} \ldots \mapsto_{\mathsf{E1}} \langle \mathcal{P}' \ \wedge \ x \neq y, \mathcal{N}' \rangle$.

[15] Note-se que se nem a condição (i) nem a condição (ii) se verificarem o sistema de reescrita termina, e podíamos concluir que $\mathcal{P}$ não era equivalente a uma disjunção de formas resolvidas normais. Isto é, o sistema de reescrita pode ser usado para determinar se essa disjunção existe ou não.

Base. Se $\|w_j\| = 1$ então w_j é uma constante a. Por redução a forma primária vem

$$\begin{array}{rcl} \mathcal{RT} & \models & \mathcal{P}_p^j \ \wedge \ u.f_j^n \doteq u_j \ \wedge \ u_j \neq w_j \\ & \Leftrightarrow & \mathcal{P}_p^j \ \wedge \ u.f_j^n \doteq u_j \ \wedge \ \mathsf{DCA}(u_j) \ \wedge \ u_j \neq a \\ & \Leftrightarrow & \displaystyle\bigvee_{\tau \in \Sigma \setminus \{a\}} \mathcal{P}_p^j \ \wedge \ u.f_j^n \doteq u_j \ \wedge \ u_j.\tau \end{array}$$

onde se τ é constante $u_j.\tau$ denota a restrição $u_j = \tau$.

Indução. Suponhamos $\|w_j\| > 1$ e que o resultado é válido para variáveis determinadas finitas de tamanho menor que $\|w_j\|$. Necessariamente, w_j é uma variável funcional e seja f^n o seu functor. Reduzindo a forma primária vem,

$$\begin{array}{rcl} \mathcal{RT} & \models & \mathcal{P}_p^j \ \wedge \ u.f_j^n \doteq u_j \ \wedge \ u_j \neq w_j \\ & \Leftrightarrow & \mathcal{P}_p^j \ \wedge \ u.f_j^n \doteq u_j \ \wedge \ \mathsf{DCA}(u_j) \ \wedge \ u_j \neq w_j \\ & \Leftrightarrow & \displaystyle\bigvee_{\tau \in \Sigma \setminus \{f^n\}} (\mathcal{P}_p^j \ \wedge \ u.f_j^n \doteq u_j \ \wedge \ u_j.\tau) \\ & & \vee \ (\mathcal{P}_p^j \ \wedge \ u.f_j^n \doteq u_j \ \wedge \ u_j.f^n \ \wedge \ u_j \neq w_j) \end{array}$$

Aplicando a regra $\mathsf{E1}$, condição (ii), ao último elemento da disjunção, tem-se necessariamente que $w_j.f_i^n \doteq w_{j_i} \in \mathcal{P}_p^j$ e $\|w_{j_i}\| < \|w_j\|$, para $1 \leq i \leq n$. Por hipótese de indução, cada uma das restrições resultantes é equivalente a uma disjunção de formas resolvidas normais obtidas pelo sistema de reescrita, logo cada $\mathcal{P}_p^j \ \wedge \ u.f_j^n \doteq u_j \ \wedge \ u_j \neq w_j$ também o é.

Repetindo o processo para cada $x_k \neq y_k$, $2 \leq k \leq h$, em (4.5), conclui-se que

$$\mathcal{RT} \models \mathcal{P} \ \Leftrightarrow \ \exists \bar{z}.\, (\mathcal{M}_1 \ \vee \ \ldots \ \vee \ \mathcal{M}_m).$$

Se $r > 1$ podemos repetir o processo anterior sucessivamente para cada n_i, com $2 \leq i \leq r$, bastando verificar de cada vez se $\mathcal{M}_j \ \wedge \ n_i$ é satisfazível, para $1 \leq j \leq m$. Note-se que é garantido que pelo menos um deles o é. 2

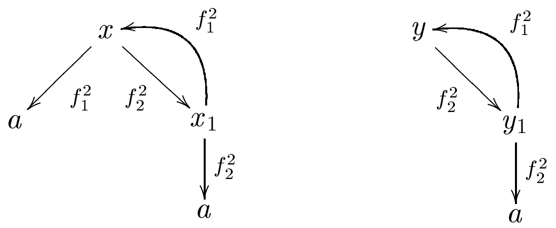
Esta proposição generaliza, para árvores racionais e $\mathcal{LU}$, os resultados sobre representações de termos definidos por contra-exemplos apresentados em [LM87] e também analisados em [Mah88] e [LMM92]. Generalizações da forma $t \setminus \{t_1 \ \vee \ \ldots \ \vee \ t_n\}$ representam o conjunto das instâncias do termo t que não são instâncias de nenhum t_i. Em [LM87], é apresentado um algoritmo (**uncover**) que permite determinar uma disjunção finita de termos equivalente a uma generalização, quando tal generalização existe. Várias aplicações deste algoritmo são apresentadas em [LMM92] e [Mah88], por exemplo para a negação construtiva em linguagens de programação em lógica [Cha88]. Em [CF92] e [Fer96], o problema da eliminação de negações foi estudado para casos particulares de álgebras quociente $\mathcal{T}/_{=_E}$, onde E é uma teoria (conjunto finito de equações) com propriedades comutativas e/ou associativas.

Para $\Sigma = \{a, f^2\}$ vamos considerar dois exemplos da aplicação do sistema de reescrita E, definido na demonstração da Proposição 4.57.

Seja a restrição

$$(\mathsf{c}_{24}) \qquad \begin{array}{l} x.f^2 \ \wedge \ x.f_1^2 \doteq a \ \wedge \ x.f_2^2 \doteq x_1 \ \wedge \ x_1.f^2 \ \wedge \ x_1.f_1^2 \doteq x \ \wedge \ x_1.f_2^2 \doteq a \\ \wedge \, y.f^2 \ \wedge \ y.f_2^2 \doteq y_1 \ \wedge \ y_1.f^2 \ \wedge \ y_1.f_1^2 \doteq y \ \wedge \ y_1.f_2^2 \doteq a \ \wedge \ x \neq y \end{array}$$

Seja $\mathcal{C}_p$ a conjunção de restrições atómicas positivas da restrição c_{24} cuja representação gráfica[16] é:



Vamos aplicar o sistema de reescrita E ao par $\langle \mathcal{C}_p \ \wedge \ x \neq y, \top \rangle$. Para simplificar, não vamos considerar a redução a forma prenexa e consideraremos reduções em paralelo em elementos duma disjunção. Supomos ainda que o quantificador existencial tem precedência menor que as conectivas.

$$\begin{array}{rl} \langle \mathcal{C}_p \ \wedge \ x \neq y, \top \rangle & \mapsto \ \ \langle \exists y_2.\, \mathcal{C}_p \ \wedge \ y.f_1^2 \doteq y_2 \ \wedge \ (a \neq y_2 \ \vee \ x_1 \neq y_1), \{x \neq y\} \rangle \\ & \mapsto \ \ \langle \exists y_2.\, \mathcal{C}_p \ \wedge \ y.f_1^2 \doteq y_2 \ \wedge \ a \neq y_2 \ \wedge \ \mathsf{DCA}(y_2), \{x \neq y\} \rangle \\ & \qquad \vee \, \langle \exists y_2.\, \mathcal{C}_p \ \wedge \ y.f_1^2 \doteq y_2 \ \wedge \ x_1 \neq y_1 \ \wedge \ \mathsf{DCA}(y_2), \{x \neq y\} \rangle \\ & \mapsto \ \ \langle \exists y_2.\, \mathcal{C}_p \ \wedge \ y.f_1^2 \doteq y_2 \ \wedge \ y_2.f^2, \{x \neq y\} \rangle \\ & \qquad \vee \, \langle \exists y_2.\, \mathcal{C}_p \ \wedge \ y.f_1^2 \doteq y_2 \ \wedge \ \mathsf{DCA}(y_2) \ \wedge \ (x \neq y \ \vee \ a \neq a), \{x \neq y, x_1 \neq y_1\} \rangle \\ & \mapsto \ \ \langle \exists y_2.\, \mathcal{C}_p \ \wedge \ y.f_1^2 \doteq y_2 \ \wedge \ y_2.f^2, \{x \neq y\} \rangle \\ & \qquad \vee \, \langle \bot, \{x \neq y\} \rangle \ \vee \ \langle \bot, \{x \neq y\} \rangle \\ & \mapsto \ \ \langle \exists y_2.\, \mathcal{C}_p \ \wedge \ y.f_1^2 \doteq y_2 \ \wedge \ y_2.f^2, \{x \neq y\} \rangle \end{array}$$

Concluímos que c_{24} é equivalente em $\mathcal{RT}$ a

[16]Nestas representações não se identificam classes de equivalência de $=_{\mathcal{C}_p}$.

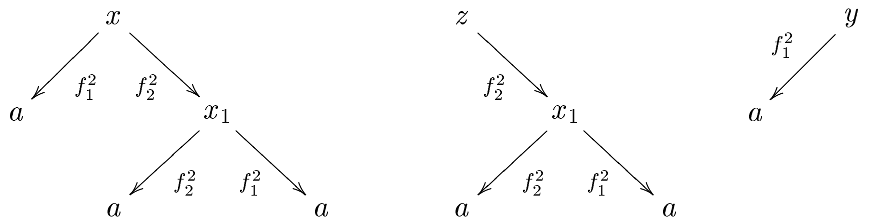
$$(\mathsf{c}_{25}) \qquad \exists y_2.\ (x.f^2 \ \wedge\ x.f_1^2 \doteq a \ \wedge\ x.f_2^2 \doteq x_1 \ \wedge\ x_1.f^2 \ \wedge\ x_1.f_1^2 \doteq x \ \wedge\ x_1.f_2^2 \doteq a \ \wedge\ y.f^2 \\ \wedge\ y.f_2^2 \doteq y_1 \ \wedge\ y_1.f^2 \ \wedge\ y_1.f_1^2 \doteq y \ \wedge\ y_1.f_2^2 \doteq a \ \wedge\ y.f_1^2 \doteq y_2 \ \wedge\ y_2.f^2)$$

que corresponde a que a "componente" $y.f_1^2$ seja diferente de a.

Seja a restrição

$$(\mathsf{c}_{26}) \qquad x.f^2 \ \wedge\ x.f_1^2 \doteq a \ \wedge\ x.f_2^2 \doteq x_1 \ \wedge\ x_1.f^2 \ \wedge\ x_1.f_1^2 \doteq a \\ \wedge\ x_1.f_2^2 \doteq a \ \wedge\ z.f^2 \ \wedge\ z.f_2^2 \doteq x_1 \ \wedge\ y.f^2 \ \wedge\ y.f_1^2 \doteq a \\ \wedge\ x \neq y \ \wedge\ x \neq z \ \wedge\ y \neq z$$

Como no exemplo anterior, seja $\mathcal{C}_p$ a conjunção de restrições atómicas positivas da restrição c_{26}, cuja representação gráfica é:



Aplicando do sistema de reescrita E ao par $\langle \mathcal{C}_p \ \wedge\ x \neq y \ \wedge\ x \neq z \ \wedge\ y \neq z, \top \rangle$ vem

$$\begin{aligned}
\langle \mathcal{C}_p \ \wedge\ x \neq y \ \wedge\ x \neq z \ \wedge\ y \neq z, \top \rangle \ &\mapsto\ \langle \exists y_2.\ \mathcal{C}_p \ \wedge\ y.f_2^2 \doteq y_2 \ \wedge\ x \neq z \ \wedge\ y \neq z \\
&\qquad \wedge\ (a \neq a \ \vee\ x_1 \neq y_2), \{x \neq y\} \rangle \\
&\mapsto\ \langle \exists y_2.\ \mathcal{C}_p \ \wedge\ y.f_2^2 \doteq y_2 \ \wedge\ \mathsf{DCA}(y_2) \ \wedge\ x \neq z \\
&\qquad \wedge\ y \neq z \ \wedge\ x_1 \neq y_2, \{x \neq y\} \rangle \\
&\mapsto\ \langle \exists y_2.\ \mathcal{C}_p \ \wedge\ y.f_2^2 \doteq y_2 \ \wedge\ y_2 = a \ \wedge\ x \neq z \\
&\qquad \wedge\ y \neq z, \{x \neq y\} \rangle \\
&\quad \vee\ \langle \exists y_2.\ \mathcal{C}_p \ \wedge\ y.f_2^2 \doteq y_2 \ \wedge\ y_2.f^2 \ \wedge\ x \neq z \\
&\qquad \wedge\ y \neq z \ \wedge\ x_1 \neq y_2, \{x \neq y\} \rangle
\end{aligned}$$

Vamos considerar os dois elementos da disjunção em separado. Para o primeiro elemento da disjunção e sendo $\mathcal{C}_p^1 \equiv \mathcal{C}_p \ \wedge\ y.f_2^2 \doteq a \ \wedge\ y_2 = a$, vem

$$\begin{aligned}
\cdots \ \mapsto\ &\langle \exists y_2 z_1.\ \mathcal{C}_p^1 \ \wedge\ z.f_1^2 \doteq z_1 \ \wedge\ y \neq z \ \wedge\ (z_1 \neq a \ \vee\ x_1 \neq x_1), \{x \neq y, x \neq z\} \rangle \\
\mapsto\ &\langle \exists y_2 z_1.\ \mathcal{C}_p^1 \ \wedge\ z.f_1^2 \doteq z_1 \ \wedge\ y \neq z \ \wedge\ \mathsf{DCA}(z_1) \ \wedge\ z_1 \neq a, \{x \neq y, x \neq z\} \rangle \\
\mapsto\ &\vee\ \langle \bot, \{x \neq y, x \neq z\} \rangle \\
\mapsto\ &\langle \exists y_2 z_1.\ \mathcal{C}_p^1 \ \wedge\ z.f_1^2 \doteq z_1 \ \wedge\ z_1.f^2 \ \wedge\ y \neq z, \{x \neq y, x \neq z\} \rangle \\
\mapsto\ &\langle \exists y_2 z_1.\ \mathcal{C}_p^1 \ \wedge\ z.f_1^2 \doteq z_1 \ \wedge\ z_1.f^2 \ \wedge\ (z_1 \neq a \ \vee\ a \neq x_1), \{x \neq y, x \neq z, y \neq z\} \rangle \\
\mapsto\ &\langle \exists y_2 z_1.\ \mathcal{C}_p^1 \ \wedge\ z.f_1^2 \doteq z_1 \ \wedge\ z_1.f^2, \{x \neq y, x \neq z, y \neq z\} \rangle
\end{aligned}$$

No segundo elemento da disjunção o tratamento da restrição $x \neq z$ produz restrições análogas, dado que y não depende em $\mathcal{C}_p$ de x nem de z. Sendo $\mathcal{C}_p^2 \equiv \mathcal{C}_p \ \wedge \ y.f_2^2 \doteq y_2 \ \wedge \ y_2.f^2$, vem

$$\begin{aligned}
\cdots \ \mapsto \ & \langle \exists y_2 z_1.\, \mathcal{C}_p^2 \ \wedge \ z.f_1^2 \doteq z_1 \ \wedge \ z_1.f^2 \ \wedge \ y \neq z \ \wedge \ x_1 \neq y_2, \{x \neq y, x \neq z\}\rangle \\
\mapsto \ & \langle \exists y_2 z_1.\, \mathcal{C}_p^3 \ \wedge \ (z_1 \neq a \ \vee \ x_1 \neq y_2) \ \wedge \ x_1 \neq y_2, \{x \neq y, x \neq z, y \neq z\}\rangle \\
\mapsto \ & \langle \exists y_2 z_1.\, \mathcal{C}_p^3 \ \wedge \ x_1 \neq y_2, \{x \neq y, x \neq z, y \neq z\}\rangle \\
\mapsto \ & \langle \exists y_2 z_1 y_3 y_4.\, \mathcal{C}_p^3 \ \wedge \ y_2.f_1^2 \doteq y_3 \ \wedge \ y_2.f_2^2 \doteq y_4 \ \wedge \ (y_3 \neq a \ \vee \ y_4 \neq a), \\
& \{x \neq y, x \neq z, y \neq z, x_1 \neq y_2\}\rangle \\
\mapsto \ & \langle \exists y_2 z_1 y_3 y_4.\, \mathcal{C}_p^3 \ \wedge \ y_2.f_1^2 \doteq y_3 \ \wedge \ y_2.f_2^2 \doteq y_4 \ \wedge \ (y_3.f^2 \ \vee \ y_4.f^2), \\
& \{x \neq y, x \neq z, y \neq z, x_1 \neq y_2\}\rangle
\end{aligned}$$

onde $\mathcal{C}_p^3 \equiv \mathcal{C}_p^2 \ \wedge \ z.f_1^2 \doteq z_1 \ \wedge \ z_1.f^2$.

Temos então que c_{26} é equivalente em $\mathcal{RT}$ à restrição sem negações

$$(\mathsf{c}_{27}) \qquad \begin{aligned}
& \exists y_2 z_1 y_3 y_4.\, (x.f^2 \ \wedge \ x.f_1^2 \doteq a \ \wedge \ x.f_2^2 \doteq x_1 \ \wedge \ x_1.f^2 \ \wedge \ x_1.f_1^2 \doteq a \ \wedge \ x_1.f_2^2 \doteq a \\
& \quad \wedge z.f^2 \ \wedge \ z.f_1^2 \doteq z_1 \ \wedge \ z_1.f^2 \ \wedge \ z.f_2^2 \doteq x_1 \ \wedge \ y.f^2 \ \wedge \ y.f_1^2 \doteq a \ \wedge \ y.f_2^2 \doteq y_2 \\
& \quad \wedge (y_2 = a \ \vee \ (y_2.f^2 \ \wedge \ y_2.f_1^2 \doteq y_3 \ \wedge \ y_2.f_2^2 \doteq y_4 \ \wedge \ (y_3.f^2 \ \vee \ y_4.f^2))))
\end{aligned}$$

No entanto, se na restrição c_{26} tivéssemos, por exemplo, $x_1.f_2^2 \doteq x_1$ tal não seria possível, pois x_1 era uma variável determinada não finita.

De modo análogo, sendo $\Sigma = \{a, f^1\}$, considere-se a restrição

$$(\mathsf{c}_{28}) \qquad x.f_1^1 \doteq x \ \wedge \ x \neq y$$

A restrição c_{28} não é equivalente em $\mathcal{RT}$ a nenhuma disjunção de formas resolvidas normais embora uma das variáveis seja determinada, mas não finita. A restrição c_{28} é equivalente em $\mathcal{RT}$ à seguinte disjunção de formas primárias

$$(\mathsf{c}_{29}) \qquad (x.f^1 \ \wedge \ x.f_1^1 \doteq x \ \wedge \ y = a) \ \vee \ (x.f^1 \ \wedge \ x.f_1^1 \doteq x \ \wedge \ y.f^1 \ \wedge \ x \neq y)$$

Usando a expressividade da linguagem $\mathcal{LU}$ é impossível eliminar a negação no segundo elemento da disjunção. Uma possibilidade seria introduzir restrições que permitissem a referência a pontos fixos, por exemplo, $x = \star(f)$ e a referência a possíveis caminhos em árvores, por exemplo, $\exists p \ y.p \doteq a$. Como foi referido, no Capítulo 3, Secção 3.4, estas extensões foram estudadas para árvores de atributos por Treinen em [Tre93] e Backofen em [Bac94a].

4.3.3 Formas Primárias para $\mathcal{T}$

No caso da assinatura Σ ser infinita a noção de forma primária para $\mathcal{T}$ coincide com a que foi dada para $\mathcal{RT}$, considerando que conjunção de restrições atómicas positivas está em forma resolvida finita normal. Em particular, são válidos os Lemas 4.26 e 4.28 e a Proposição 4.30.

No caso da assinatura Σ ser finita, a noção de forma primária também coincide podendo o algoritmo de eliminação de negações da Proposição 4.57 e os seus lemas preparatórios, serem adaptados (simplificando) para $\mathcal{T}$.

4.4 Caso Geral

Finalmente, as soluções duma restrição $c \in \mathcal{LU}$ podem ser caracterizadas pelo teorema seguinte.

Teorema 4.58 *Toda a restrição $c \in \mathcal{LU}$ é equivalente em $\mathcal{RT}$ (e em $\mathcal{T}$) a uma disjunção finita de formas primárias ou a $\perp$.*

Dem. Pode-se transformar $c \in \mathcal{LU}$ numa restrição equivalente em forma normal disjuntiva, *i.e*, numa disjunção de conjunções de restrições atómicas positivas e negativas, c_i, pela aplicação das regras usuais de distributividade[17]:

D1. $c_1 \wedge (c_2 \vee c_3) \mapsto (c_1 \wedge c_2) \vee (c_1 \wedge c_3)$

D2. $(c_1 \vee c_2) \wedge c_3 \mapsto (c_1 \wedge c_3) \vee (c_2 \wedge c_3)$

Estas regras são correctas e completas para formas normais disjuntivas em qualquer álgebra de $\mathcal{L}$. Como usualmente[18], a terminação pode ser provada pela seguinte função de interpretação de $\mathcal{LU}$ nos inteiros positivos, γ:

$$\begin{array}{ll} \gamma(c) = 2 & \text{se } c \text{ é atómica} \\ \gamma(c_1 \vee c_2) = \gamma(c_1) + \gamma(c_2) + 1 & \\ \gamma(c_1 \wedge c_2) = \gamma(c_1) \times \gamma(c_2) & \end{array}$$

A aplicação de qualquer das regras decresce estritamente o valor de γ.

Então $c \approx_{\mathcal{RT}} c_1 \vee \ldots \vee c_n$, isto é,

$$\forall \rho \;\; \mathcal{T}, \mathcal{RT} \models_\rho c \quad \text{se e só se} \quad \exists i \;\; \mathcal{T}, \mathcal{RT} \models_\rho c_i$$

Se c_i é uma conjunção de restrições atómicas positivas então ou é equivalente a $\perp$ ou a uma forma resolvida $\mathcal{M}$. Se c_i contiver restrições atómicas negativas é equivalente a

[17] Considerando $c \in \mathcal{LU}$ a menos da relação $\equiv_{\mathcal{B}}$ apenas uma regra é necessária.

[18] Ver, por exemplo, [DJ90].

uma disjunção de formas primárias (apenas uma forma primária, se $\mid \Sigma \mid = \infty$) ou a $\perp$. A restrição c é equivalente a $\perp$ se e só se todos os c_i o forem. 2

Corolário 4.59 *Determinar a satisfazibilidade de restrições de $\mathcal{LU}$ em $\mathcal{RT}$ é um problema* **NP***-completo.*

Dem. Seja $c \in \mathcal{LU}$. As regras do sistema de reescrita D, apresentado na demonstração do Teorema 4.58, podem, considerando a relação $\equiv_{\mathcal{B}}$, ser aplicadas não deterministicamente da seguinte forma:

Dn1. $c_1 \ \wedge \ (c_2 \ \vee \ c_3) \mapsto (c_1 \ \wedge \ c_2)$

A restrição c é satisfazível se e só se pelo menos uma das restrições resultantes é satisfazível. Como cada uma destas restrições tem tamanho polinomial em função do tamanho de c, a satisfazibilidade de c pode ser determinada em tempo não-determinístico polinomial se isso for possível para cada uma das restrições resultantes. No caso de Σ ser infinita, vimos que esse problema pode ser resolvido em tempo polinomial. No caso de Σ ser finita, esse problema é **NP**-completo. Portanto, em ambos os casos é garantido que determinar a satisfazibilidade em $\mathcal{LU}$ é um problema em **NP**.

Vamos provar que é **NP-hard** por redução polinomial do problema da satisfazibilidade duma fórmula da lógica proposicional em forma normal conjuntiva, normalmente conhecido por **SAT** e que é **NP**-completo [GJ79]. Uma fórmula proposicional está em forma normal conjuntiva se for uma conjunção de disjunções de variáveis proposicionais ou das suas negações. Podemos transformar, em tempo linear, uma fórmula destas numa restrição de $\mathcal{LU}$ tal que uma é satisfazível se e só se a outra o é. Para tal, basta associar a cada variável proposicional x uma restrição $x = a$ e à sua negação a restrição $x \neq a$[19].
2

Corolário 4.60 *Seja $2 \leq \mid \Sigma \mid < \infty$. Para toda a restrição $c \in \mathcal{LU}$ é possível determinar se c é equivalente a uma disjunção de formas resolvidas normais.*

Dem. Consequência do Teorema 4.58 e da Proposição 4.57. 2

4.5 Sumário

Neste capítulo obtivemos, para $\mathcal{LU}$, caracterizações das conjunções de restrições atómicas satisfazíveis em $\mathcal{RT}$: formas resolvidas se não ocorrem negações e formas primárias, caso

[19]Se Σ apenas contiver functores consideram-se as restrições $x.f^n$ e $\neg x.f^n$ para algum $f^n \in \Sigma$.

contrário. Para estas restrições, apresentamos algoritmos para decidir a satisfazibilidade e obter formas resolvidas ou (disjunções de) formas primárias equivalentes. No caso da assinatura Σ ser finita estes algoritmos têm complexidade polinomial. Se Σ é finita o problema da satisfazibilidade de conjunções de restrições atómicas é **NP**-completo.

Foi descrito um algoritmo para identificar variáveis equivalentes em formas resolvidas, isto é, variáveis com as mesmas soluções. Este algoritmo é usado para a obtenção de formas resolvidas normais e formas primárias.

Vimos também como eliminar negações redundantes numa forma primária.

Estudamos ainda o problema de determinar se uma conjunção de restrições atómicas é equivalente a uma disjunção de conjunções de restrições atómicas positivas. Se Σ é infinita não é possível obter uma disjunção de conjunções de restrições atómicas positivas equivalente a uma forma primária sem redundâncias. Para Σ finita, na Proposição 4.57 apresentamos uma condição necessária e suficiente para que tal seja possível.

Finalmente, por redução a forma normal disjuntiva, mostramos que qualquer restrição de $\mathcal{LU}$ é equivalente a uma disjunção de formas primárias ou a $\perp$.

Todas estas caracterizações permitem estudar o problema da satisfazibilidade de $\mathcal{L}$ em $\mathcal{RT}$ (ou em $\mathcal{T}$), tendo como principal novidade, em relação a outras formalizações existentes na literatura, a introdução de caminhos que permitem o uso de linguagens sem quantificadores.

Capítulo 5

Modelos Minimais em $\mathcal{LU}$

Abordamos, neste capítulo, o problema de obter para uma restrição satisfazível em $\mathcal{RT}$, conjuntos minimais de formas primárias cuja disjunção é equivalente em $\mathcal{RT}$, a essa restrição. Informalmente a noção de minimal corresponde a não ter informação redundante para caracterizar a restrição.

Do ponto de vista prático, a existência tais conjuntos (únicos) e de algoritmos para os obter, é útil para a determinação da equivalência de restrições e para a sua simplificação. Contudo, como é evidente dos resultados do Capítulo 4, não é necessária para resolver o problema da satisfazibilidade. No entanto, e em particular, os conjuntos minimais caracterizam as soluções duma restrição. Os sistemas de reescrita que serão descritos no capítulo seguinte para determinar a satisfazibilidade em $\mathcal{LU}$ factorizam a restrição inicial em restrições cuja conjunção é equivalente à inicial. Assim, surgiu o problema de relacionar e caracterizar as soluções da restrição inicial em termos das dos factores.

Como motivação para este estudo foi a existência de tais conjuntos quando apenas se consideram conjunções de restrições atómicas positivas (ou suas disjunções), à semelhança da existência de unificadores mais gerais para um conjunto de equações ou a existência de conjuntos completos de unificadores em algumas teorias equacionais [FH86, JK91]. Por outro lado, qualquer fórmula satisfazível da lógica proposicional é equivalente a uma disjunção minimal de implicantes primos, onde um implicante primo é uma conjunção de literais que vinculam a fórmula e tal que qualquer subconjunto próprio não tem essa propriedade [Qui52, McC56, SM54]. Este facto levou-nos a investigar o problema análogo para $\mathcal{LU}$. Contudo, apenas iremos apresentar uma solução parcial desse problema e algumas ideias para a sua resolução geral.

Começamos por introduzir as noções de modelo, modelo parcial e modelo minimal de uma restrição satisfazível e com elas caracterizar as restrições em que não ocorrem negações (não redundantes). Estas noções serão também utilizadas nos capítulos seguintes. A presença de negações e disjunções em restrições levou, na Secção 5.1, à noção de forma primária normal e versões "normalizadas" das noções de modelos. O problema de obter disjunções de modelos minimais é introduzido na Secção 5.2. Na Secção 5.2.1 é apresentada uma solução para o caso da assinatura apenas conter constantes. Um método para generalizar essa solução ao caso geral é sugerido na Secção 5.2.2. Finalmente, na Secção 5.3, fazemos uma pequena referência ao estudo deste problema para a linguagem $\mathcal{LC}$.

Para além da relação $\equiv_{\mathcal{B}}$, considerando a idempotência da disjunção vamos supor que uma disjunção não contém elementos repetidos e que se algum elemento é $\top$, então a disjunção é equivalente a $\top$.

Embora as formas primárias caracterizem as restrições satisfazíveis de $\mathcal{LU}$ essa caracterização, tal como acontecia para as formas resolvidas, não é única mesmo supondo que as formas primárias não contêm restrições redundantes. Duas formas primárias distintas podem ser equivalentes ou equivalentes a disjunções de formas primárias. Considerem-se os seguintes[1] exemplos:

(c_1) $$x.f_1^1 \doteq w_1 \ \wedge\ y.f_1^1 \doteq z_1 \ \wedge\ x \neq y$$

(c_2) $$x.f_1^1 \doteq w_1 \ \wedge\ y.f_1^1 \doteq z_1 \ \wedge\ w_1 \neq z_1$$

(c_3) $$x.f_1^2 \doteq w_1 \ \wedge\ x.f_2^2 \doteq w_2 \ \wedge\ y.f_1^2 \doteq z_1 \ \wedge\ y.f_2^2 \doteq z_2 \ \wedge\ x \neq y$$

(c_4) $$\begin{array}{l}(x.f_1^2 \doteq w_1 \ \wedge\ x.f_2^2 \doteq w_2 \ \wedge\ y.f_1^2 \doteq z_1 \ \wedge\ y.f_2^2 \doteq z_2 \ \wedge\ w_1 \neq z_1) \vee \\ (x.f_1^2 \doteq w_1 \ \wedge\ x.f_2^2 \doteq w_2 \ \wedge\ y.f_1^2 \doteq z_1 \ \wedge\ y.f_2^2 \doteq z_2 \ \wedge\ w_2 \neq z_2)\end{array}$$

As representações gráficas das formas normais de c_1 e de c_2 são respectivamente

$$\begin{tikzcd}[x] \arrow[d, "f_1^1"] & \text{e} & [y] \arrow[d, "f_1^1"] \\ {[w_1]} & & {[z_1]}\end{tikzcd}$$

[1] Para clareza da exposição, alguns exemplos não estarão em forma primária ou disjunção de tais formas, mas são convertíveis facilmente a essas formas.

e a de c_3, que coincide com as de c_4 é

$$\begin{tikzcd}
 & {[x]} \arrow[dl, "f_1^2"'] \arrow[dr, "f_2^2"] & & & {[y]} \arrow[dl, "f_1^2"'] \arrow[dr, "f_2^2"] & \\
{[w_1]} & & {[w_2]} & {[z_1]} & & {[z_2]}
\end{tikzcd}$$

Facilmente se verifica que $\mathsf{c}_1 \approx_{\mathcal{RT}} \mathsf{c}_2$ e $\mathsf{c}_3 \approx_{\mathcal{RT}} \mathsf{c}_4$.

Designaremos por *modelo* uma forma primária sem redundâncias. Como anteriormente, se o modelo for uma forma resolvida, será denotado por $\mathcal{M}$.

Definição 5.1 *Seja $c_1 \in \mathcal{LU}$ uma restrição satisfazível em $\mathcal{RT}$. Uma forma primária $\mathcal{P}$ é um* modelo parcial *de c_1 se c_1 vincula $\mathcal{P}$, ou seja, $c_1 \models_{\mathcal{RT}} \mathcal{P}$, ou ainda, $\mathsf{Sol}(c_1, \mathcal{RT}) \subseteq \mathsf{Sol}(\mathcal{P}, \mathcal{RT})$. Uma forma primária $\mathcal{P}$ é um* modelo *de c_1 se $\mathcal{P}$ vincula c_1, ou seja, $\mathcal{P} \models_{\mathcal{RT}} c_1$, ou ainda, $\mathsf{Sol}(\mathcal{P}, \mathcal{RT}) \subseteq \mathsf{Sol}(c_1, \mathcal{RT})$. Se estas inclusões forem próprias, os modelos designam-se por* próprios. *A relação de vínculo entre modelos designa-se por* subsunção. *Um modelo $\mathcal{P}$ de c_1 diz-se um* modelo minimal, *se $\mathcal{P}$ não subsume nenhum modelo de c_1. Dado um conjunto de variáveis $V \subseteq X$, diz-se que $\mathcal{P}$ é um* modelo de c_1 em V, *e escreve-se $\mathcal{P} \models^V_{\mathcal{RT}} c_1$, se $\mathsf{Sol}_V(\mathcal{P}, \mathcal{RT}) \subseteq \mathsf{Sol}_V(c_1, \mathcal{RT})$.*

Resulta da definição anterior que se $\mathcal{P}$ é modelo parcial de $c_1 \in \mathcal{LU}$, então $\mathsf{Vars}(\mathcal{P}) \subseteq \mathsf{Vars}(c_1)$. A relação de *subsunção* definida no conjunto dos modelos de $\mathcal{LU}$ é uma relação de pré-ordem.

Por exemplo, as restrições

$$(\mathsf{c}_5) \qquad x.f_1^2 \doteq w_1 \ \wedge \ x.f_2^2 \doteq w_2 \ \wedge \ y.f_1^2 \doteq z_1 \ \wedge \ y.f_2^2 \doteq z_2 \ \wedge \ w_1 \neq z_1$$

$$(\mathsf{c}_6) \qquad x.f_1^2 \doteq w_1 \ \wedge \ x.f_2^2 \doteq w_2 \ \wedge \ y.f_1^2 \doteq z_1 \ \wedge \ y.f_2^2 \doteq z_2 \ \wedge \ w_1 = a \ \wedge \ z_1 = b$$

são modelos de c_3. No entanto, não são modelos minimais de c_3, pois, as restrições c_5 e c_6 subsumem a restrição c_3, que é um modelo (e dele próprio).

Corolário 5.2 *Se $\mathcal{C}$ é uma conjunção de restrições atómicas positivas e $\mathcal{C}$ é satisfazível, então $\mathcal{C}$ tem um modelo minimal único. Se $\mid \Sigma \mid = \infty$ e $\mathcal{C}$ for uma conjunção de restrições atómicas, então $\mathcal{C}$ tem um modelo minimal.*

Dem. Basta considerar a forma primária associada e a unicidade de formas normais. 2

Corolário 5.3 *Seja $\mid \Sigma \mid = \infty$. Se $c \in \mathcal{LU}$ é satisfazível e c é equivalente a uma disjunção de formas normais, então c é equivalente a uma disjunção finita e única de modelos minimais.*

Dem. Pelo Teorema 4.58, pela Proposição 4.27 e excluindo da disjunção os modelos que subsumam outros modelos. A unicidade é garantida pela unicidade de formas normais, Proposição 4.20. 2

Isto é, se não ocorrem negações uma restrição satisfazível tem uma caracterização única em termos de modelos minimais.

No caso geral, se $c \in \mathcal{LU}$ é satisfazível, então existem formas primárias $\mathcal{P}_1, \ldots, \mathcal{P}_n$ tais que $c \approx_{\mathcal{RT}} \mathcal{P}_1 \vee \ldots \vee \mathcal{P}_n$, mas o facto de uma forma primária poder ser equivalente a uma disjunção de formas primárias, não permite garantir que as formas primárias $\mathcal{P}_i$ sejam modelos minimais de c. Por outro lado, a presença de negações e disjunções leva a que uma forma primária $\mathcal{P}$ seja equivalente a disjunções do tipo $(c_1 \wedge \mathcal{P}) \vee (\neg c_1 \wedge \mathcal{P})$.

Por exemplo, sendo

$$(\mathsf{c}_7) \qquad x.f_1^2 \doteq w_1 \wedge x.f_2^2 \doteq w_2 \wedge y.f_1^2 \doteq z_1 \wedge y.f_2^2 \doteq z_2 \wedge w_1 \neq a \wedge w_1 \neq z_1$$

$$(\mathsf{c}_8) \qquad x.f_1^2 \doteq w_1 \wedge x.f_2^2 \doteq w_2 \wedge y.f_1^2 \doteq z_1 \wedge y.f_2^2 \doteq z_2 \wedge z_1 \neq b \wedge w_1 \neq z_1$$

temos que $\mathsf{c}_5 \approx_{\mathcal{RT}} \mathsf{c}_6 \vee \mathsf{c}_7 \vee \mathsf{c}_8$.

5.1 Formas Primárias Normais

A forma primária c_3 pode ser "normalizada" para a restrição c_4, e então c_3 seria equivalente a uma disjunção (única) de modelos minimais "normais". Isso é possível porque as variáveis envolvidas na desigualdade estão *definidas* na forma normal de c_3 e pode-se, usando o Lema 4.56, associar à equivalência (4.3) a regra de reescrita

NP1. $\mathcal{P} \wedge x \neq y \;\mapsto\; \mathcal{P} \wedge (\bigvee_{i=1}^{n} v_i \neq v_i')$
se $\mathcal{P}$ é uma forma primária, $x.f^n, y.f^n \in \mathcal{P}$ e para todo $1 \leq i \leq n$, $x.f_i^n \doteq v_i \in \mathcal{P}$ e $y.f_i^n \doteq v_i' \in \mathcal{P}$

Definição 5.4 *Uma forma primária* $\mathcal{P} \equiv \mathcal{P}_p \wedge \mathcal{P}_n$ *diz-se* normal *se não contiver redundâncias e se* $x \neq y \in \mathcal{P}_n$ *então pelo menos uma das variáveis* x *ou* y *é não definida ou livre em* $\mathcal{P}_p$.

Proposição 5.5 *Toda a forma primária é equivalente a uma disjunção de formas primárias normais.*

Dem. Usando a regra $\mathsf{NP1}$, basta modificar o sistema de reescrita E, da demonstração da Proposição 4.57, restringindo-nos a negações $x \neq y$ com x e y definidas em $\mathcal{P}_p$ (em particular, sem introduzir variáveis novas). 2

Corolário 5.6 *Se $\mathcal{P} \equiv \mathcal{P}_p \ \wedge \ \mathcal{P}_n$ e $\mathcal{P}' \equiv \mathcal{P}'_p \ \wedge \ \mathcal{P}'_n$ são duas formas primárias normais tal que $\mathcal{P} \approx_{\mathcal{RT}} \mathcal{P}'$ então $\mathcal{P}$ e $\mathcal{P}'$ são idênticas a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{P}_p}$.*

Redefinimos os conceitos de modelos, modelos parciais e modelos minimais considerando *apenas* formas primárias normais. No que se segue, a menos referência em contrário, vamos apenas considerar modelos *normais*.

Apesar do Corolário 5.6, em certos casos, mesmo sem a ocorrência de variáveis definidas em negações, uma forma primária normal podia ainda ser "simplificada" com a introdução de variáveis novas, quantificadas existencialmente. Vamos ver contudo, que isso nem sempre é possível e portanto apenas consideraremos a normalização da Definição 5.4.

Supondo $\mid \Sigma \mid = \infty$, consideremos os seguintes modelos (normais):

(C_9) $\quad x.f_1^2 \doteq w_1 \ \wedge \ y.f_1^2 \doteq z_1 \ \wedge \ x \neq y$

(C_{10}) $\quad x.f_1^2 \doteq w_1 \ \wedge \ y.f_1^2 \doteq z_1 \ \wedge \ w_1 \neq z_1$

(C_{11}) $\quad x.f_1^2 \doteq w_1 \ \wedge \ y.f_2^2 \doteq z_2 \ \wedge \ x \neq y$

(C_{12}) $\quad x.f_1^2 \doteq w_1 \ \wedge \ w_1.g_1^2 \doteq u_1 \ \wedge \ y.f_1^2 \doteq z_1 \ \wedge \ z_1.g_1^2 \doteq t_1 \ \wedge \ x \neq y$

(C_{13}) $\quad x.f_1^2 \doteq w_1 \ \wedge \ w_1.g_1^2 \doteq u_1 \ \wedge \ y.f_1^2 \doteq z_1 \ \wedge \ z_1.g_1^2 \doteq t_1 \ \wedge \ u_1 \neq t_1$

(C_{14}) $\quad x.f_1^2 \doteq w_1 \ \wedge \ w_1.g_1^2 \doteq u_1 \ \wedge \ y.f_1^2 \doteq z_1 \ \wedge \ z_1.g_1^2 \doteq t_1 \ \wedge \ w_1 \neq z_1$

(C_{15}) $\quad x.f_1^2 \doteq w_1 \ \wedge \ y.f_1^2 \doteq z_1 \ \wedge \ x.f_2^2 \doteq w_2 \ \wedge \ y.f_2^2 \doteq z_2 \ \wedge \ x \neq y$

(C_{16}) $\quad x.f_1^2 \doteq w_1 \ \wedge \ y.f_1^2 \doteq z_1 \ \wedge \ x.f_2^2 \doteq w_2 \ \wedge \ y.f_2^2 \doteq z_2 \ \wedge \ w_1 \neq z_1 \ \wedge \ w_2 \neq y_2$

(C_{17}) $\quad x.f^2 \ \wedge \ x \neq y$

(C_{18}) $\quad x.f^2 \ \wedge \ y.f^2 \ \wedge \ x \neq y$

(C_{19}) $\quad x.f_1^1 \doteq x \ \wedge \ x \neq y$

Em relação aos exemplos anteriores, temos que:

- c_{10} é modelo de c_9 mas não é modelo minimal;

- c_9 e c_{11} admitem os próprios como modelo minimal único;

- Considerando a representação gráfica das formas normais de c_{12}, c_{13} e c_{14}

$$\begin{tikzcd}
 & {[x]} \arrow[dl, "f_1^2"'] & & & {[y]} \arrow[dl, "f_1^2"'] \\
{[w_1]} \arrow[d, "g_1^2"'] & & & {[z_1]} \arrow[d, "g_1^2"'] & \\
{[u_1]} & & & {[t_1]} &
\end{tikzcd}$$

  vemos que c_{13} e c_{14} são modelos de c_{12} mas não são minimais; e c_{13} é modelo não minimal de c_{14};

- c_{16} é modelo não minimal de c_{15} (que coincide com c_3); neste caso, apenas se pode concluir que $x \neq y$ é redundante em c_{16} e o único modelo minimal de c_{16} é ele próprio.

Introduzindo variáveis novas, $\bar{z}$, podíamos ainda transformar a restrição c_9 numa restrição c' em que as variáveis x e y fossem definidas e, tal que $\mathcal{RT} \models \mathsf{c}_9 \;\Leftrightarrow\; \exists \bar{z}.\, c'$. Neste caso, é possível garantir que se $z \neq w \in c'$ então z e w são variáveis livres (na forma normal de c'). Para tal, basta usar a regra $\mathsf{NP1}$. Vem para a restrição c_9,

$$\begin{array}{lll}
\mathcal{RT} & \models & x.f_1^2 \doteq w_1 \;\wedge\; y.f_1^2 \doteq z_1 \;\wedge\; x \neq y \\
 & \Leftrightarrow & \exists x_2, y_2\; x.f_1^2 \doteq w_1 \;\wedge\; y.f_1^2 \doteq z_1 \;\wedge\; x.f_2^2 \doteq w_2 \;\wedge\; y.f_2^2 \doteq z_2 \;\wedge\; (w_1 \neq z_1 \;\vee\; w_2 \neq z_2)
\end{array}$$

O mesmo processo podia-se aplicar às restrições c_{11}, c_{12} e c_{16}. Em relação, à restrição c_{17} podíamos obter o mesmo resultado considerando que $\mathsf{c}_{17} \models y.f^2 \;\vee\; \neg y.f^2$ e depois, procedendo como nos casos anteriores. E, analogamente, para a restrição c_{18}. No entanto, dada a expressividade de $\mathcal{LU}$, tal não seria possível para a restrição c_{19}. Este exemplo ilustra que não é sempre possível transformar uma forma primária numa disjunção de formas primárias normais em que as desigualdades só ocorrem entre variáveis livres.

Se $\mid \Sigma \mid < \infty$, uma forma primária é considerada sem variáveis livres, logo esse tipo de normalização também não é possível. Na Proposição 4.57, o sistema de reescrita E estabelece um método para eliminar as negações ou, apenas para transformar as formas primárias em disjunções de formas primárias normais.

5.2 Disjunções de Modelos Minimais

Uma forma primária normal pode ser ainda equivalente a uma disjunção de modelos.

Por exemplo,

$$(\mathsf{c}_{20}) \qquad (x \neq y \ \wedge \ x \neq a) \ \vee \ (x \neq y \ \wedge \ y \neq b) \ \vee \ (y = b \ \wedge \ x \neq b) \ \vee \ (x = a \ \wedge \ y \neq a)$$

é equivalente em $\mathcal{RT}$ (ou em $\mathcal{T}$) a $x \neq y$.

Um método para simplificar c_{20} para $x \neq y$ é obter os modelos minimais de c_{20}. Como iremos ver, normalmente, não existe apenas uma disjunção de modelos minimais equivalente a uma restrição $c \in \mathcal{LU}$. Assim a noção de simplificação terá de ser explicitada para que dadas duas restrições equivalentes em $\mathcal{RT}$, se possa decidir qual a mais "simples".

Numa disjunção de modelos $\mathcal{P}_1 \ \vee \ \ldots \ \vee \ \mathcal{P}_n$, se um modelo $\mathcal{P}_i$ vincular a disjunção dos restantes pode ser eliminado da disjunção. Contudo este método não garante que os elementos que restem sejam modelos minimais. Uma caracterização será "simples" se for uma disjunção de modelos minimais que não contenha modelos que vinculem a disjunção dos restantes. Isto é, se obtivermos a disjunção de todos os modelos minimais duma restrição e depois eliminarmos elementos que sejam redundantes obtemos uma caracterização "simples".

Começamos por considerar o problema de determinar todos os modelos minimais duma restrição.

Para a lógica proposicional, o problema de determinar todos os modelos minimais, isto é,os *implicantes primos*, duma fórmula, foi estudado inicialmente por W. V. Quine [Qui52, Qui55, Qui59]. Teve como motivação a minimização de funções booleanas para o desenho de circuitos lógicos [McC56, SM54], onde se pretendia obter a representação mais simples, segundo um dado critério, duma fórmula proposicional. Este problema, e o seu dual, também é relevante em Inteligência Artificial em problemas de diagnóstico [Rei87, dKMR90], manutenção de verdades [dKR87], aprendizagem [Rym94] e raciocínio não-monótono [dKR87, Prz89]. Alguns algoritmos (eficientes) para obter implicantes primos de fórmulas proposicionais foram apresentados mais recentemente em [JRSL70, JP90, KT90, dK92, Nga93].

Nesta secção, vamos generalizar (parcialmente) para $\mathcal{LU}$ o método apresentado por Quine em [Qui55, Qui59] e também por Samson e Mills em [SM54].

5.2.1 Modelos Minimais para Σ com Constantes

Suponhamos que Σ é infinita e apenas contém constantes. Note-se que no caso de Σ ser finita, e apenas com constantes, pela Proposição 4.40 (e o Teorema 4.58) podemos obter todos os modelos minimais duma restrição.

Dadas duas formas primárias $\mathcal{P}_1$ e $\mathcal{P}_2$, para a obtenção de todos os modelos minimais de $\mathcal{P}_1 \vee \mathcal{P}_2$ será necessário o conceito de resolvente de duas formas primárias. Esta noção corresponde à noção de *consensus* de duas conjunções de literais em [Qui55, Qui59, SM54].

O lema seguinte será utilizado para provar que a resolvente de duas formas primárias é um modelo da disjunção das duas. Reflecte para Σ com constantes os axiomas de $\mathcal{RT}$ (e de $\mathcal{T}$).

Lema 5.7 *Seja Σ um conjunto infinito de constantes, e $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$. Para todos os valores v, v',v'' e v''' de $\mathcal{LU}$ tem-se que*

(5.1) $\mathcal{A} \models v' = v \Rightarrow v'' \neq v' \vee v'' = v$

(5.2) $\mathcal{A} \models v'' = v''' \Rightarrow v'' \neq v \vee v''' \neq v'$ *se v e v' são valores incompatíveis*

(5.3) $\mathcal{A} \models v' \neq v'' \Rightarrow v' \neq v \vee v'' \neq v$

Dem. (5.1) Seja ρ é uma atribuição de variáveis tal que $\rho(v') = \rho(v)$. Então, $\rho(v'') = \rho(v)$ ou $\rho(v'') \neq \rho(v')$.

(5.2) Deduz-se do Axioma A5 de $E_{\mathcal{T}}$ ou $E_{\mathcal{RT}}$.

(5.3) Deduz-se da transitividade da igualdade, Axioma A3 de $E_{\mathcal{T}}$ ou $E_{\mathcal{RT}}$.

2

Definição 5.8 *Sendo $\mathcal{P}_1 \equiv \mathcal{P}_p^1 \wedge \mathcal{P}_n^1$ e $\mathcal{P}_2 \equiv \mathcal{P}_p^2 \wedge \mathcal{P}_n^2$ formas primárias, diferentes de $\top$, uma forma primária $\mathcal{P}$ é uma* resolvente *de $\mathcal{P}_1$ e $\mathcal{P}_2$ se uma das seguintes condições se verifica:*

(i) $v'' \neq v' \in \mathcal{P}_n^1$, $[v'']_{\mathcal{P}_p^1} \cap [v]_{\mathcal{P}_p^2} \neq \emptyset$ e $[v]_{\mathcal{P}_p^2}$ *não é unitária; $\mathcal{P}$ é, se existir, a forma primária de*

$$(\mathcal{P}_1 \setminus \{v'' \neq v'\}) \cup \mathcal{P}_2'$$

onde $\mathcal{P}_2'$ é uma forma primária que resulta de $\mathcal{P}_2$ excluindo da classe $[v]_{\mathcal{P}_p^2}$ pelo menos os elementos que pertençam à classe $[v'']_{\mathcal{P}_p^1}$ e acrescentando os que pertencem à classe $[v']_{\mathcal{P}_p^1}$ de tal forma que:

(a) $[v'']_{\mathcal{P}_p} \cap [v]_{\mathcal{P}_p} = \emptyset$

(b) $[v'']_{\mathcal{P}_p} \cup [v]_{\mathcal{P}_p} \supseteq [v'']_{\mathcal{P}_p^1} \cup [v]_{\mathcal{P}_p^1} \cup [v']_{\mathcal{P}_p^1} \cup [v'']_{\mathcal{P}_p^2} \cup [v']_{\mathcal{P}_p^2}$

(ii) $x \neq v \in \mathcal{P}_n^1$, $y \neq v' \in \mathcal{P}_n^2$ *e* v *e* v' *são valores incompatíveis em* $\mathcal{P}_1$ *e em* $\mathcal{P}_2$*;* $\mathcal{P}$ *é, se existir, a forma primária de*

$$(\mathcal{P}_1 \setminus \{x \neq v\}) \cup (\mathcal{P}_2 \setminus \{y \neq v'\}) \cup \{x = y\}$$

(iii) $v' \neq v \in \mathcal{P}_n^1$, $v'' \neq v''' \in \mathcal{P}_n^2$ *e* $v''' \in [v]_{\mathcal{P}_p^2}$*;* $\mathcal{P}$ *é, se existir, a forma primária de*

$$(\mathcal{P}_1 \setminus \{v' \neq v\}) \cup (\mathcal{P}_2 \setminus \{v'' \neq v'''\}) \cup \{v' \neq v''\}$$

Na definição anterior, a resolvente pela condição (i) seria, aplicando o Lema 5.7,

$$(\mathcal{P}_1 \setminus \{v'' \neq v'\}) \cup (\mathcal{P}_2 \setminus \{v'' = v\}) \cup \{v' = v\}$$

Mas, pela transitividade da igualdade é necessário considerar as respectivas classes de equivalência de $=_{\mathcal{C}_p^i}$, para $i = 1, 2$. Contudo, dadas essas classes de equivalência, na condição (i) podemos supor que $v'' \in [v]_{\mathcal{P}_p^2}$ e $v'' \not\equiv v$, e, na condição (iii) que $v''' \equiv v$.

Vamos ilustrar com alguns exemplos.

Pela condição (i), da definição anterior, a resolvente de $x = z$ e $y \neq z$ é $x = y$. Este facto permitirá concluir, usando o Corolário 5.10, que a restrição

(c$_{21}$) $\quad x = z \ \vee \ x \neq y \ \vee \ y \neq z$

é equivalente em $\mathcal{RT}$ a $\top$. O mesmo resultado é obtido, pela condição (iii), considerando a resolvente entre $x \neq y$ e $y \neq z$.

Para as restrições

(c$_{22}$) $\quad y_1 = a \ \wedge \ y_2 = a \ \wedge \ y_3 = a$

(c$_{23}$) $\quad y_1 \neq y_2$

todas as resolventes de c$_{22}$ e c$_{23}$ são

(c$_{24}$) $\quad y_1 = a \ \wedge \ y_3 = a$

(c$_{25}$) $\quad y_2 = a \ \wedge \ y_3 = a$

(c$_{26}$) $\quad y_1 = y_3 \ \wedge \ y_2 = a$

(c_{27}) $\quad y_2 = y_3 \ \wedge \ y_1 = a$

Isto é, considerando as classes de equivalência de $=_{\mathsf{c}_{22}}$ obtêm-se partições tais que y_1 e y_2 não pertençam simultaneamente ao mesmo conjunto (e todas as outras classes não sejam alteradas).

Para as restrições

(c_{28}) $\quad x_1 = y_1 \ \wedge \ x_2 = y_2 \ \wedge \ y_1 \neq y_2$

(c_{29}) $\quad y_1 = y_2 \ \wedge \ x_1 = x_2$

não existe nenhuma resolvente entre elas. Apenas se poderia aplicar a condição (i) da Definição 5.8, mas a condição (i)a não se verifica.

Lema 5.9 *Sejam $\mathcal{P}_1$ e $\mathcal{P}_2$ duas formas primárias. Se $\mathcal{P}$ é uma resolvente de $\mathcal{P}_1$ e $\mathcal{P}_2$ então*

$$\mathcal{P} \models_{\mathcal{RT}} \mathcal{P}_1 \ \vee \ \mathcal{P}_2$$

Dem. Consideramos cada uma das possíveis formas de obter resolventes, de acordo com a Definição 5.8.

(i) Seja $\rho \in \mathsf{Sol}(\mathcal{P}, \mathcal{RT})$. Como a restrição $\mathcal{P}$ é satisfazível apenas temos que analisar as restrições que envolvem variáveis nas classes $[v'']_{\mathcal{P}_p}$ e $[v]_{\mathcal{P}_p}$, pois as restantes ocorrem em $\mathcal{P}_1$ ou $\mathcal{P}_2$. Temos que $[v'']_{\mathcal{P}_p^1} \subseteq [v'']_{\mathcal{P}_p}$ e $[v']_{\mathcal{P}_p^1} \subseteq [v']_{\mathcal{P}_p}$. Seja

$$[v'']_{\mathcal{P}_p^2} = [v]_{\mathcal{P}_p^2} = \{v_1, \ldots, v_n\}$$

com

$$\{v_1, \ldots, v_k\} \subseteq [v'']_{\mathcal{P}_p^1} \text{ e } \{v_{k+1}, \ldots, v_l\} \subseteq [v']_{\mathcal{P}_p^1} \qquad 0 < k + l \leq n$$

e os restantes elementos v_i, para $l + 1 \leq i \leq n$, não pertencem a nenhuma dessas classes. Então,

$$[v'']_{\mathcal{P}_p} = \{v_1, \ldots, v_k, v_{l+1}, \ldots, v_{n-m}, w_1, \ldots, w_s\} \qquad 0 \leq m \leq n - l$$

onde para alguns $1 \leq i \leq s$, w_i pode pertencer a $[v'']_{\mathcal{P}_p^1}$. E

$$[v']_{\mathcal{P}_p} = [v]_{\mathcal{P}_p} = \{v_{k+1}, \ldots, v_l, v_{n-m+1}, \ldots, v_n, u_1, \ldots, u_r\}$$

onde para alguns $1 \leq j \leq r$, u_j pode pertencer a $[v']_{\mathcal{P}_p^1}$.

Suponhamos que $\rho(v'') = \rho(v)$. Para que $\rho \in \mathsf{Sol}(\mathcal{P}_2, \mathcal{RT})$ basta ver que os w_i, para $1 \leq i \leq s$, e u_j, para $1 \leq j \leq r$, não são incompatíveis em $\mathcal{P}_2$ com os elementos de $[v'']_{\mathcal{P}_p^2}$, nem entre si. Mas se tal acontecesse também o seriam em $\mathcal{P}$ e então $\rho \notin \mathsf{Sol}(\mathcal{P}, \mathcal{RT})$. Analogamente, se $\rho(v'') \neq \rho(v)$ então $\rho \in \mathsf{Sol}(\mathcal{P}_1, \mathcal{RT})$. Não pode haver w_i, para algum $1 \leq i \leq s$ nem u_j, para $1 \leq j \leq r$, que pertençam à mesma classe de $=_{\mathcal{P}_p^1}$, pois nesse caso a condição (i)a da Definição 5.8, não se verificava.

(ii) Corresponde ao caso (5.2) do Lema 5.7. Seja $\rho \in \mathsf{Sol}(\mathcal{P}, \mathcal{RT})$. Então, se $\rho(x) = \rho(v)$, como $\rho(v) \neq \rho(v')$, temos que $\rho(y) \neq \rho(v')$, logo $\rho \in \mathsf{Sol}(\mathcal{P}_2, \mathcal{RT})$. Se $\rho(x) \neq \rho(v)$ então $\rho \in \mathsf{Sol}(\mathcal{P}_1, \mathcal{RT})$.

(iii) Corresponde ao caso (5.3) do Lema 5.7. Seja $\rho \in \mathsf{Sol}(\mathcal{P}, \mathcal{RT})$ e suponhamos que $v \equiv v'''$. Se $\rho(v) = \rho(v')$ então $\rho(v) \neq \rho(v'')$ e $\rho \in \mathsf{Sol}(\mathcal{P}_2, \mathcal{RT})$. Caso contrário, $\rho \in \mathsf{Sol}(\mathcal{P}_1, \mathcal{RT})$.

2

Corolário 5.10 *Se $\mathcal{P}_1$ e $\mathcal{P}_2$ são formas primárias e $\mathcal{P}$ uma resolvente de $\mathcal{P}_1$ e $\mathcal{P}_2$ então*

$$\mathcal{P}_1 \vee \mathcal{P}_2 \approx_{\mathcal{RT}} \mathcal{P}_1 \vee \mathcal{P}_2 \vee \mathcal{P}$$

Recordemos a restrição c_{20}, agora referida por c_{30}

$$(\mathsf{c}_{30}) \qquad (x \neq y \wedge x \neq a) \vee (x \neq y \wedge y \neq b) \vee (y = b \wedge x \neq b) \vee (x = a \wedge y \neq a)$$

A resolvente entre o primeiro e o último elemento da disjunção é $(x \neq y \wedge y \neq a)$ e entre o segundo e terceiro é $(x \neq y \wedge x \neq b)$. Temos então que $\mathsf{c}_{30} \approx_{\mathcal{RT}} \mathsf{c}_{31}$, onde

$$(\mathsf{c}_{31}) \qquad (x \neq y \wedge x \neq a) \vee (x \neq y \wedge y \neq b) \vee (y = b \wedge x \neq b) \vee (x = a \wedge y \neq a) \vee (x \neq y \wedge y \neq a) \vee (x \neq y \wedge x \neq b)$$

Como $(x = a \wedge y \neq a)$ subsume $(x \neq y \wedge y \neq a)$ e $(y = b \wedge x \neq b)$ subsume $(x \neq y \wedge x \neq b)$ vem

$$(\mathsf{c}_{32}) \qquad (x \neq y \wedge x \neq a) \vee (x \neq y \wedge y \neq b) \vee (x \neq y \wedge y \neq a) \vee (x \neq y \wedge x \neq b)$$

Na restrição c_{32} podemos obter as resolventes entre o primeiro e o último elemento da disjunção e entre o segundo e terceiro, usando a condição (ii) da Definição 5.8. Ambas as resolventes são equivalentes a $x \neq y$. Como todos os elementos de c_{32} subsumem $x \neq y$, concluímos que

$$\mathsf{c}_{30} \approx_{\mathcal{RT}} x \neq y$$

Neste caso, não foi necessário obter todos os modelos minimais de c_{30} para a simplificar. Mas, em geral, isso poderá ser necessário.

Proposição 5.11 *Supondo que Σ é infinita e que contém apenas constantes, qualquer disjunção $\mathcal{C}$ de formas primárias pode ser transformada numa disjunção de modelos minimais de $\mathcal{C}$ pela aplicação do sistema de reescrita* M, *constituído pelas seguintes regras:*

M1. $\mathcal{P}_i \vee \mathcal{C} \mapsto \mathcal{C}$ *se $\mathcal{P}_i$ é uma forma primária e existe em $\mathcal{C}$ uma forma primária $\mathcal{P}_j$ tal que $\mathcal{P}_i$ subsume $\mathcal{P}_j$;*

M2. $\mathcal{C} \mapsto \mathcal{C} \vee \mathcal{P}$ *onde $\mathcal{P}$ é uma resolvente de duas formas primárias em $\mathcal{C}$ e $\mathcal{P}$ não subsume nenhuma forma primária de $\mathcal{C}$.*

Se $\mathcal{C} \equiv \mathcal{P}_1 \vee \ldots \vee \mathcal{P}_n$ é irredutível para M *então, para cada $1 \leq i \leq n$, $\mathcal{P}_i$ é um modelo minimal de $\mathcal{C}$* e *$\mathcal{C}$ contém todos os seus modelos minimais.*

Dem. A correcção da regra $\mathsf{M1}$ é imediata e a da regra $\mathsf{M2}$ resulta do Corolário 5.10. É também evidente que as regras transformam disjunções de formas primárias em disjunções de formas primárias (ou $\top$). Analisemos a terminação. Nenhuma regra introduz variáveis ou constantes, portanto, basta provar que não há ciclos pela aplicação das regras. A regra $\mathsf{M1}$ faz diminuir o número de elementos da disjunção e uma restrição eliminada não pode ser reintroduzida pela regra $\mathsf{M2}$. Se $\mathcal{P}$ foi eliminada pela regra $\mathsf{M1}$, então existe e existirá sempre em $\mathcal{C}$, uma forma primária $\mathcal{P}_j$ tal que $\mathcal{P}$ subsume $\mathcal{P}_j$ e $\mathcal{P}$ nunca será introduzida por $\mathsf{M2}$. Assim, não pode haver um ciclo que incluia a aplicação da regra $\mathsf{M1}$. Por outro lado, a regra $\mathsf{M2}$ aumenta o número de elementos da disjunção. Portanto não é possível haver um ciclo, usando apenas $\mathsf{M2}$,

$$\mathcal{C} \mapsto \cdots \mapsto \mathcal{C} \mapsto \cdots$$

Seja $\mathcal{C} \equiv \mathcal{P}_1 \vee \ldots \vee \mathcal{P}_n$ irredutível para M. Se $\mathcal{C} \equiv \top$ então $\mathcal{C}$ admite apenas um modelo minimal. Se, para todo $1 \leq i \leq n$, $\mathcal{P}_i$ é uma forma normal, pelo Corolário 5.3, são todos modelos minimais de $\mathcal{C}$ e $\mathcal{C}$ não admite outros modelos minimais.

Caso contrário, suponhamos que existe um modelo minimal $\mathcal{P}$ de $\mathcal{C}$ que não é nenhum dos $\mathcal{P}_i$, para $1 \leq i \leq n$. Temos que:

(i) $\mathcal{P} \models_{\mathcal{RT}} \mathcal{C}$

(ii) $\mathcal{P}$ não subsume nenhum modelo de $\mathcal{C}$, em particular nenhum $\mathcal{P}_i$, para $1 \leq i \leq n$; isto é, para todo $1 \leq i \leq n$ existe uma restrição atómica c_i tal que $\mathcal{P}_i \models_{\mathcal{RT}} c_i$ e c_i não é determinada em $\mathcal{P}$.

(iii) $\mathsf{Vars}(\mathcal{P}) \subseteq \mathsf{Vars}(\mathcal{C})$ e $\mathcal{P}$ não contém símbolos de Σ que não ocorram em $\mathcal{C}$.

Considere-se o conjunto

$$\mathsf{Mod}_{\mathcal{P}} = \{\mathcal{P}' \mid \mathcal{P}' \models_{\mathcal{RT}} \mathcal{P},\ \mathcal{P}' \text{ é um modelo e verifica as condições (i)–(iii)}\}$$

Atendendo à condição (iii) este conjunto é finito. Seja $\mathcal{P}_m$ tal que $\mathcal{P}_m \in Mod_{\mathcal{P}}$ e nenhum elemento de $Mod_{\mathcal{P}}$ subsume $\mathcal{P}_m$. Pela condição (ii) para todo $1 \leq i \leq n$, existe uma restrição atómica c_i tal que $\mathcal{P}_i \models_{\mathcal{RT}} c_i$ e c_i não é determinada em $\mathcal{P}_m$. Como $\mathcal{P}_m \wedge c_i$ satisfaz a condição (i) e a condição (iii), então $\mathcal{P}_m \wedge c_i \models_{\mathcal{RT}} \mathcal{P}_i$.

A restrição c_i pode ser dum dos seguintes tipos:

Caso $v = v''$**:** Como $\mathcal{P}_i$ é uma forma primária temos que $v'' \in [v]_{\mathcal{P}_p^i}$. Como $v = v''$ não é determinada em $\mathcal{P}_m$, $v \neq v''$ também não é determinada em $\mathcal{P}_m$. A restrição $\mathcal{P}_m \wedge v \neq v''$ é um modelo de $\mathcal{C}$ que verifica a condição (i) e a condição (iii) mas não verifica a condição (ii). Então existe $1 \leq j \leq n$ tal que $\mathcal{P}_m \wedge v \neq v'' \models_{\mathcal{RT}} \mathcal{P}_j$. Como Σ é infinita, $v' \neq v'' \in \mathcal{P}_j$ com $v' \in [v]_{\mathcal{P}_p^i}$. Seja

$$\mathcal{P}_r \equiv \mathcal{P}_j \setminus \{v'' \neq v'\} \cup \mathcal{P}_i'$$

a resolvente de $\mathcal{P}_j$ e de $\mathcal{P}_i$, pela condição (i) da Definição 5.8. A restrição $\mathcal{P}_r$ é satisfazível porque $\mathcal{P}_m$ é satisfazível e $\mathcal{P}_m$ subsume $\mathcal{P}_r$. Por esta última razão, também $\mathcal{P}_r$ não subsume nenhum $\mathcal{P}_k$, com $1 \leq k \leq n$.

Concluí-se que a regra $\mathsf{M2}$ é aplicável.

Caso $v'' \neq v'$**:** Como no caso anterior, existe pelo menos um $1 \leq j \leq n$ tal que $\mathcal{P}_m \wedge v = v'' \models_{\mathcal{RT}} \mathcal{P}_j$. Se $\mathcal{P}_j \models_{\mathcal{RT}} v'' = v$ e $v' \in [v]_{\mathcal{P}_p^j}$ a demonstração segue como no caso anterior. Senão, como foi visto no Lema 5.7, podemos ainda ter os seguintes casos:

a) $v''' \neq v \in \mathcal{P}_j$, v e v' são incompatíveis em $\mathcal{P}_i$ e $\mathcal{P}_j$ e $v'' \in [v''']_{\mathcal{P}_p^m}$

b) $v'' \neq v \in \mathcal{P}_j$ e $\mathcal{P}_m \models_{\mathcal{RT}} v' \neq v$

No primeiro caso, aplica-se a regra $\mathsf{M2}$ considerando a resolvente da condição (ii) da Definição 5.8. E, no segundo caso, a resolvente da condição (iii). Como nos casos anteriores as resolventes são satisfazíveis.

Suponhamos, agora, que $\mathcal{C}$ é irredutível para M, mas que para algum $1 \leq i \leq n$, $\mathcal{P}_i$ não é modelo minimal de $\mathcal{C}$. Como $\mathcal{P}_i$ é modelo de $\mathcal{C}$ existe um modelo minimal $\mathcal{P}$ de

$\mathcal{C}$ tal que $\mathcal{P}_i \models_{\mathcal{RT}} \mathcal{P}$. Se $\mathcal{P}$ é algum dos $\mathcal{P}_j$ para $1 \leq j \leq n$, e $j \neq i$, então aplica-se a regra M1. Senão, existe um modelo minimal que ainda não pertence à disjunção, e como vimos, acima a regra M2 aplica-se. 2

5.2.1.1 Representações *Simples* e Implementação

Como já se disse, a disjunção de todos os modelos minimais de uma restrição $c \in \mathcal{LU}$ pode conter restrições redundantes. Um modelo minimal (ou uma disjunção de modelos minimais) pode vincular a disjunção de dois ou mais modelos minimais. Os métodos descritos em [Qui55, Qui59], para eliminar fórmulas redundantes podem ser adaptados directamente para $\mathcal{LU}$.

Por exemplo, para a restrição

$$(x = a \ \wedge \ y = a) \ \vee \ x \neq y \tag{c33}$$

a disjunção de todos os seus modelos minimais é:

$$x = a \ \vee \ y = a \ \vee \ x \neq y \tag{c34}$$

mas ou $x = a$ ou $y = a$ podem ser eliminados.

A ideia geral é verificar individualmente se algum elemento da disjunção vincula a disjunção dos restantes. À disjunção dos elementos que não verifiquem esta propriedade denomina-se o *núcleo*. Se algum elemento que não pertence ao núcleo vincula o núcleo, então pode ser eliminado. Para os outros elementos que não pertencem ao núcleo pode-se ainda testar se combinações de disjunções desses elementos vinculam a disjunção dos restantes.

No entanto, o nosso o objectivo nesta secção foi essencialmente mostrar como é possível obter todos os modelos minimais duma restrição de $\mathcal{LU}$ satisfazível em $\mathcal{RT}$. Obviamente isso não será necessário se existirem critérios para obter uma disjunção "simples" de modelos minimais sem ter de os obter todos exaustivamente. No sentido de obter tais métodos, em [Qui59] é apresentado um critério para identificar o *núcleo* duma fórmula da lógica proposicional. No entanto nem todas as fórmulas têm um núcleo nem é óbvio quais os modelos minimais que se têm de adicionar ao núcleo para obter uma disjunção de modelos minimais equivalente à fórmula inicial. A investigação de métodos análogos para a $\mathcal{LU}$ será também desejável.

Por outro lado, para uma implementação eficiente da determinação dos modelos minimais em $\mathcal{LU}$ será conveniente estudar a possibilidade de adaptação dos algoritmos para a obtenção de implicantes primos, em lógica proposicional, propostos na literatura, e acima citados.

5.2.2 Modelos Minimais para Σ com Functores

Supomos que os resultados da secção anterior poderão ser generalizados para o caso de Σ ser infinita e conter functores de aridade superior a zero. No entanto, nesta secção apenas apresentaremos alguns exemplos e indicaremos como tal generalização poderá ser obtida. Neste caso é necessário distinguir se $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$. Nos exemplos seguintes supomos que $\mathcal{A} = \mathcal{RT}$. A noção de *resolvente* será usada sem contudo ser dada uma definição formal. As restrições, assim referidas, devem pelo menos verificar o Lema 5.9.

Consideremos as restrições

$$(\mathsf{c}_{35}) \qquad x.f_1^2 \doteq x_1 \ \wedge \ x.f_2^2 \doteq x_2 \ \wedge \ y = x \ \wedge \ y_1 = x_1 \ \wedge \ y_2 = x_2$$

$$(\mathsf{c}_{36}) \qquad x.f_1^2 \doteq x_1 \ \wedge \ x.f_2^2 \doteq x_2 \ \wedge \ y.f_1^2 \doteq y_1 \ \wedge \ y.f_2^2 \doteq y_2 \ \wedge \ y_1 \neq x_1$$

$$(\mathsf{c}_{37}) \qquad x.f_1^2 \doteq x_1 \ \wedge \ x.f_2^2 \doteq x_2 \ \wedge \ y.f_1^2 \doteq y_1 \ \wedge \ y.f_2^2 \doteq y_2 \ \wedge \ y_2 \neq x_2$$

$$(\mathsf{c}_{38}) \qquad x.f_1^2 \doteq x_1 \ \wedge \ x.f_2^2 \doteq x_2 \ \wedge \ y.f_1^2 \doteq y_1 \ \wedge \ y.f_2^2 \doteq y_2$$

A representação gráfica de c_{35} é

$$\begin{tikzcd}
& {[x,y]} \arrow[dl, "f_1^2"'] \arrow[dr, "f_2^2"] & \\
{[x_1,y_1]} & & {[x_2,y_2]}
\end{tikzcd}$$

e a forma normal de c_{36}, c_{37} e c_{38} é

$$\begin{tikzcd}
& {[x]} \arrow[dl, "f_1^2"'] \arrow[dr, "f_2^2"] & & & {[y]} \arrow[dl, "f_1^2"'] \arrow[dr, "f_2^2"] & \\
{[x_1]} & & {[x_2]} & {[y_1]} & & {[y_2]}
\end{tikzcd}$$

A disjunção das restrições c_{35}, c_{36} e c_{37} é equivalente em $\mathcal{RT}$ à restrição c_{38}. Contudo, considerando, por exemplo a resolvente entre c_{35} e c_{36}, usando a condição (i) da Definição 5.8, obtém-se novamente a restrição c_{35}, devido à presença da restrição $y = x$.

Por outro lado, se

(c_{39}) $\qquad x.f_1^2 \doteq x_1 \;\wedge\; x.f_2^2 \doteq x_2 \;\wedge\; y.f_1^2 \doteq y_1 \;\wedge\; y.f_2^2 \doteq x_2 \;\wedge\; y_2 = x_2$

cuja representação gráfica é

$$
\begin{tikzcd}
 & {[x]} \arrow[dl, "f_1^2"'] \arrow[dr, "f_2^2"] & & {[y]} \arrow[dl, "f_2^2"'] \arrow[dr, "f_1^2"] & \\
{[x_1]} & & {[x_2, y_2]} & & {[y_1]}
\end{tikzcd}
$$

for uma resolvente entre c_{35} e c_{36}, uma resolvente entre c_{39} e c_{37} é c_{38}.

Esta resolvente pode ser obtida, não só "separando", como na condição (i) da Definição 5.8, a classe $[x_1]_{\mathsf{c}_{35}}$ da de y_1, tornando x_1 e y_1 distinguíveis, mas também "separando" a classe $[x]_{\mathsf{c}_{35}}$ da de y.

A mesma técnica pode ser aplicada para as restrições

(c_{40}) $\qquad y = x \;\wedge\; z = x \;\wedge\; x.f_1^1 \doteq x$

(c_{41}) $\qquad x \neq y$

Qualquer resolvente de c_{40} e de c_{41} não pode vincular nem $x \neq y$ nem $y \neq z$ nem $x \neq z$. Sendo a representação gráfica de c_{40}

$$
\begin{tikzcd}
{[x, y, z]} \arrow[loop below, "f_1^1"]
\end{tikzcd}
$$

as resolventes de c_{40} e c_{41} são representadas graficamente por

$$
\begin{tikzcd}
{[x, z]} \arrow[d, "f_1^1"] & {[x]} \arrow[d, "f_1^1"] & {[x, z]} \arrow[loop below, "f_1^1"] & {[x]} \arrow[d, "f_1^1"] & {[x]} \arrow[loop below, "f_1^1"] & {[y, z]} & {[x]} \arrow[d, "f_1^1"] & {[y]} \arrow[loop below, "f_1^1"] \\
{[y]} & {[y, z]} & & {[y]} \arrow[d, "f_1^1"] & & & {[z]} & \\
 & & & {[z]} & & & & \\
\text{(a)} & \text{(b)} & \text{(c)} & \text{(d)} & \text{(e)} & & \text{(f)} &
\end{tikzcd}
$$

a menos renomeação de variáveis no conjunto $\{x, y\}$, para (a), (b), (c), e (e); e no conjunto $\{x, y, z\}$ para as duas restantes.

Para obter resolventes entre duas formas primárias e que não vinculem nenhuma delas, é necessário alterar a condição (i) da Definição 5.8, para "separar" recursivamente as classes dependentes. Para tal podemos usar um algoritmo semelhante ao da obtenção de classes distinguíveis numa foram resolvida, da Figura 4.1. Dado um par (v, v'), se $x.f_i^n \doteq v$ e $x.f_i^n \doteq v'$, temos que "separar" a classe $[x]_{\mathsf{c}_{35}}$ de modo a que v e v' não sejam dependentes duma mesma classe.

Por exemplo, a restrição c_{39} pode ser obtida de c_{35} e c_{36}, considerando o par (x_1, y_1).

Analogamente, para as restrições

$$(\mathsf{c}_{42}) \qquad y = x \wedge z = x \wedge x.f_1^2 \doteq w \wedge u = w \wedge w.g_1^1 \doteq s \wedge t = s \wedge s.h_1^2 \doteq w \wedge s.h_2^2 \doteq x$$

$$(\mathsf{c}_{43}) \qquad x \neq y$$

cuja representação gráfica da forma normal de c_{42} é

$$\begin{tikzcd}
{[x,y,z]} \arrow[d, "f_1^2"] \\
{[w,u]} \arrow[d, "g_1^1"] \\
{[s,t]} \arrow[u, bend right, "h_1^2"'] \arrow[uu, bend left=60, "h_2^2"]
\end{tikzcd}$$

Uma resolvente será, graficamente,

$$\begin{tikzcd}
{[x]} \arrow[d, "f_1^2"] & & {[y,z]} \arrow[d, "f_1^2"] \\
{[w]} \arrow[d, "g_1^1"] & & {[u]} \arrow[d, "g_1^1"] \\
{[s]} \arrow[u, bend right, "h_1^2"'] \arrow[uu, bend left=60, "h_2^2"] & & {[t]} \arrow[u, bend left, "h_1^2"] \arrow[uu, bend right=60, "h_2^2"']
\end{tikzcd}$$

Embora as "árvores" sejam semelhantes as classes de x e y são distinguíveis porque não são definidas.

No caso geral, esta "separação" conduz a uma explosão combinatória e será conveniente procurar representações[2] "compactas" mas que permitam identificar os modelos minimais. Note-se que estas separações podem ser encontradas por um método de "geração e teste".

Considerem-se as restrições

[2] Obviamente que a disjunção das formas primárias é uma dessas representações!

$$(\mathsf{c}_{44}) \qquad x.f_1^1 \doteq x_1 \ \wedge \ y_1 = x_1$$

$$(\mathsf{c}_{45}) \qquad x.f_1^1 \doteq y_1 \ \wedge \ y_1 \neq x_1$$

cuja representação gráfica de c_{44} é

$$\begin{tikzcd} {[x]} \arrow[d, "f_1^1"] \\ {[x_1, y_1]} \end{tikzcd}$$

Neste caso, não podemos aplicar o método anterior para "separar" a classe de x para obter uma resolvente de c_{44} e c_{45}. No entanto, a restrição $x.f_1^1 \doteq y_1$ vincula a disjunção $\mathsf{c}_{44} \vee \mathsf{c}_{45}$ e não subsume nenhum dos elementos. Para obter esta resolvente podemos considerar $v'' \equiv x_1$ e $v' \equiv y_1$ na condição (i) da Definição 5.8.

Consideremos ainda a disjunção

$$(\mathsf{c}_{46}) \qquad \neg x.f^n \ \vee \ y \neq x \ \vee \ y.f^n$$

Esta disjunção é equivalente em $\mathcal{RT}$ a $\top$. No entanto, nenhuma das condições anteriores para obter resolventes, se aplica.

Este facto sugere que ainda existem outros métodos para obter resolventes ... Para tal é necessário considerar os restantes axiomas de $\mathcal{RT}$ (ou de $\mathcal{T}$).

O lema seguinte é uma generalização do Lema 5.7, para Σ com functores.

Lema 5.12 *Seja $\mathcal{A} = \mathcal{T}$ ou $\mathcal{A} = \mathcal{RT}$. Para quaisquer variáveis x, y, valores de $\mathcal{LU}$ v, v', v''*

e v''', constante a e functores distintos f^n e g^k tem-se que

(5.4) $\mathcal{A} \models v' = v \Rightarrow v'' \neq v' \vee v'' = v$

(5.5) $\mathcal{A} \models v'' = v''' \Rightarrow v'' \neq v \vee v''' \neq v'$ *se v e v' são valores incompatíveis*

(5.6) $\mathcal{A} \models v' \neq v'' \Rightarrow v' \neq v \vee v'' \neq v$

(5.7) $\mathcal{A} \models x.f_i^n \doteq v \Rightarrow v' \neq v \vee x.f_i^n \doteq v'$ *se $\mathcal{A} = \mathcal{T}$, v e v' são distintos de x*

(5.8) $\mathcal{A} \models x = y \Rightarrow \neg x.f^n \vee y.f^n$

(5.9) $\mathcal{A} \models x = y \Rightarrow \neg x.f^n \vee y.g^k$ *se $f^n \not\equiv g^k$*

(5.10) $\mathcal{A} \models x = y \Rightarrow \neg x.f^n \vee y \neq a$

(5.11) $\mathcal{A} \models x = y \Rightarrow \neg x.f^n \vee (y \neq z \wedge z.g^k)$

(5.12) $\mathcal{A} \models x = y \Rightarrow \neg x.f^n \vee (y \neq z \wedge \neg z.f^n)$

(5.13) $\mathcal{A} \models x.f^n \Rightarrow y \neq x \vee y.f^n$

(5.14) $\mathcal{A} \models \neg y.f^n \Rightarrow \neg x.f^n \vee y \neq x$

Dem. As demonstrações são análogas às do Lema 5.7. 2

As três primeiras condições correspondem às do Lema 5.7. A condição (5.7) coincide com a condição (5.4) se se generalizar a noção de valor para expressões do tipo $x.f_i^n$. As condições (5.8)-(5.12) generalizam para functores a condição (5.5). E a condição (5.14) exprime a transitividade da igualdade.

Utilizando a condição (5.13) entre o segundo e o terceiro elementos da disjunção c_{46} obtemos a restrição $x.f^n$. Com esta restrição e com o primeiro elemento, usando a condição (5.8), podemos simplificar a disjunção para $\top$.

Conjectura Considerando uma noção de resolvente que na condição (i) da Definição 5.8, permita distinguir as classes dos valores envolvidos na negação e use as implicações do Lema 5.12 para obter resolventes, a Proposição 5.11 pode ser generalizada para Σ infinita e com functores de aridade superior a zero.

5.3 Modelos Minimais em $\mathcal{LC}$

Acabamos este capítulo com uma breve nota ao estudo de modelos minimais em $\mathcal{LC}$.

Se Σ apenas contiver constantes as linguagens $\mathcal{LC}_0$, $\mathcal{LC}$ e $\mathcal{LU}$ coincidem e portanto o estudo dos modelos minimais feito para $\mathcal{LU}$ é obviamente válido para as outras.

Vimos, no Capítulo 2, que podíamos reduzir o problema da satisfazibilidade em $\mathcal{LC}_0$ ou $\mathcal{LC}$ ao problema da satisfazibilidade em $\mathcal{LU}$. Contudo, essa redução não pode ser usada directamente para o estudo de modelos minimais em $\mathcal{LC}$.

O exemplo seguinte ilustra esse facto.

A restrição de $\mathcal{LC}$

$$(\mathsf{c}_{47}) \qquad x.f_1^1 \doteq y \ \vee \ \neg(x.f_1^1 \doteq y)$$

é equivalente em $\mathcal{RT}$ a $\top$.

Pelo Lema 2.33 temos que

$$\mathcal{RT} \models \neg x.f_1^1 \doteq y \ \Leftrightarrow \ \neg x.f^1 \ \vee \ \exists z\, (x.f_1^1 \doteq z \ \wedge \ z \neq y) \tag{5.15}$$

e, então,

$$(\mathsf{c}_{48}) \qquad \neg x.f^1 \ \vee \ (x.f^1 \ \wedge \ x.f_1^1 \doteq z \ \wedge \ z \neq y)$$

é satisfazível se e só se $\neg x.f_1^1 \doteq y$ o for.

Contudo a restrição de $\mathcal{LU}$

$$(\mathsf{c}_{49}) \qquad x.f_1^1 \doteq y \ \vee \ \neg x.f^1 \ \vee \ (x.f_1^1 \doteq z \ \wedge \ z \neq y)$$

não é equivalente em $\mathcal{RT}$ a $\top$. Por exemplo, a atribuição de variáveis ρ tal que $\rho(x) = f(b)$, $\rho(y) = a$ e $\rho(z) = c$ não pertence a $\mathsf{Sol}(\mathsf{c}_{49}, \mathcal{RT})$.

No entanto, identificando as variáveis quantificadas existencialmente introduzidas na redução de $\mathcal{LC}$ a $\mathcal{LU}$ e considerando o Lema 2.33, ou outras regras usadas no Teorema 2.34, podemos generalizar para $\mathcal{LC}$ os métodos para obter modelos minimais descritos para $\mathcal{LU}$.

Assim, bastava considerar que a variável z em c_{49} era quantificada existencialmente e usar (5.15) para concluir que c_{47} era equivalente em $\mathcal{RT}$ a $\top$.

5.4 Sumário

Os problemas da existência e obtenção de representações minimais de restrições satisfazíveis são relevantes para a simplificação e a detecção de restrições equivalentes. Contudo, o facto dos métodos existentes obrigarem à obtenção de todos os modelos minimais torna a sua resolução extremamente ineficiente. Neste capítulo, mostramos como obter todos os modelos minimais de restrições sem negações ou em que apenas ocorrem constantes e sugerimos a solução para o caso geral. As várias noções de modelos que foram definidas serão também usadas no capítulo seguinte, na descrição de algoritmos eficientes para o problema da satisfazibilidade.

Capítulo 6

Resolutores para $\mathcal{LU}$

Sendo a determinação da satisfazibilidade de restrições de $\mathcal{LU}$ um problema **NP**-completo, apresentaremos neste capítulo alguns métodos que permitam a redução do seu custo computacional. O Teorema 4.58, do Capítulo 4, estabelece um algoritmo exponencial para determinar a satisfazibilidade duma restrição de $\mathcal{LU}$. Se a assinatura Σ é infinita, o carácter exponencial do algoritmo resulta da transformação da restrição inicial em forma normal disjuntiva (multiplicação de disjunções). Para diminuir o custo desta transformação, utilizaremos neste capítulo uma estratégia baseada em técnicas de factorização que permite reduzir, em tempo polinomial, o tamanho das restrições às quais se tem de aplicar a transformação referida. Este método está na base do processamento de restrições nos formalismos de gramáticas lógicas com restrições $\mathcal{CLG}$, apresentados em [DMV91, DV92, DMV94]. Um exemplo da sua utilização foi apresentado no Capítulo 1, Secção 1.4.3.2.

Se a assinatura Σ é finita, como vimos no Corolário 4.46, a satisfazibilidade duma conjunção de restrições atómicas é um problema **NP**-completo. Embora, seu o carácter exponencial resulte da transformação em forma normal disjuntiva, os métodos acima referidos não contribuem para uma redução do tamanho das restrições. Neste caso, métodos usados em programação por restrições em domínios finitos ([vHSD92b, vHSD92a, CN95]) deverão ser adicionados. No entanto, esse estudo não será feito no trabalho aqui apresentado. Assim, neste capítulo vamos supor que a assinatura Σ é infinita.

Começamos por introduzir informalmente as técnicas utilizadas e ilustrar com um exemplo. Como nos capítulos anteriores, vamos considerar essencialmente a álgebra das árvores racionais $\mathcal{RT}$.

Qualquer restrição de $\mathcal{LU}$ pode ser transformada em tempo polinomial, e omitindo os parênteses, numa restrição equivalente da forma

$$\mathcal{C}_p \wedge \mathcal{C}_n \wedge \mathcal{D}isj_1 \wedge \ldots \wedge \mathcal{D}isj_m \tag{6.1}$$

onde, cada um dos factores é $\top$ ou

> $\mathcal{C}_p$ é uma conjunção de restrições atómicas positivas
> $\mathcal{C}_n$ é uma conjunção de restrições atómicas negativas
> $\mathcal{D}isj_i$ é uma disjunção de restrições da mesma forma, para $1 \leq i \leq m$

Deste modo cada restrição de $\mathcal{LU}$ é constituída por conjunções e disjunções de:

- restrições atómicas negativas
- conjunções de restrições atómicas positivas

Formalmente, sendo $\mathcal{P}$ uma conjunção de restrições atómicas (forma primária ou $\perp$), uma restrição $c \in \mathcal{LU}$ na forma (6.1) é definida pela seguinte gramática BNF:

$$\begin{array}{lcl}
\mathcal{C} & ::= & \mathcal{P} \wedge \mathcal{F} \\
 & | & \mathcal{P} \\
 & | & \mathcal{F} \\
\mathcal{F} & ::= & (\mathcal{D}) \\
 & | & \mathcal{F} \wedge (\mathcal{D}) \\
\mathcal{D} & ::= & \mathcal{C} \vee \mathcal{C} \\
 & | & \mathcal{C} \vee \mathcal{D}
\end{array}$$

Qualquer modelo parcial c_1 de c será denominado um *factor determinístico* de c. Se c está na forma geral (6.1) e se $\mathcal{C}_p$ for (ou admitir) uma forma normal, então $\mathcal{C}_p$ é um factor determinístico de c. O *factor determinístico*, teoremas clássicos da lógica de 1ª ordem, os axiomas de $\mathcal{RT}$ e a definição de $\mathcal{LU}$ podem ser utilizados – através de regras de reescrita – para simplificar cada um dos outros elementos da conjunção. Esta simplificação, que é de complexidade polinomial, pode ser repetida recursivamente no interior das disjunções. A detecção de factores determinísticos comuns nas disjunções $Disj_i$ e a sua factorização permitirão também reduzir o tamanho das restrições a multiplicar. Finalmente, para obter um algoritmo completo para a satisfazibilidade, será necessário multiplicar as disjunções $Disj_i$ que restarem. No entanto, bastará multiplicar disjunções que não forem independentes para a satisfazibilidade. Isto é, dada uma conjunção $c_1 \wedge c_2 \in \mathcal{LU}$, serão formulados critérios para que a conjunção seja satisfazível se e só se cada um dos factores o seja, independentemente.

Consideremos o seguinte exemplo:

$$
\begin{aligned}
(\mathsf{c}_1) \qquad & x.f_1^1 \doteq u \ \wedge \ y.f_1^3 \doteq w \ \wedge \ x \neq b \ \wedge \ (z = x \ \vee \ y = b) \\
& \wedge \ ((y.f_1^3 \doteq u \ \wedge \ u.g^2 \ \wedge \ (u = a \ \vee \ x_1 = z)) \ \vee \ (x_1 = x \ \wedge \ y \neq z)) \\
& \wedge \ (y.f_2^3 \doteq x \ \vee \ y.f_2^3 \doteq a) \ \wedge \ (y.f_3^3 \doteq u \ \vee \ y.f_3^3 \doteq b)
\end{aligned}
$$

Sendo $\mathcal{C}_p \equiv x.f^1 \ \wedge \ x.f_1^1 \doteq u \ \wedge \ y.f^3 \ \wedge \ y.f_1^3 \doteq w$ podemos simplificar c_1 para

$$
\begin{aligned}
(\mathsf{c}_2) \qquad & \mathcal{C}_p \ \wedge \ z = x \ \wedge \ ((w = u \ \wedge \ u.g^2 \ \wedge \ (u = a \ \vee \ x_1 = z)) \\
& \vee \ (x_1 = x \ \wedge \ y \neq z)) \\
& \wedge \ (y.f_2^3 \doteq x \ \vee \ y.f_2^3 \doteq a) \ \wedge \ (y.f_3^3 \doteq u \ \vee \ y.f_3^3 \doteq b)
\end{aligned}
$$

Porque: $\mathcal{RT} \models \mathcal{C}_p \ \wedge \ x \neq b \ \Leftrightarrow \ \mathcal{C}_p$; na primeira disjunção $\mathcal{RT} \models \mathcal{C}_p \ \wedge \ y = b \ \Leftrightarrow \ \perp$ e, utilizando teoremas clássicos da lógica de 1ª ordem, essa disjunção é equivalente a $z = x$; e na segunda disjunção, pelo Axioma A4 de $\mathcal{RT}$, como $y.f_1^3 \doteq w$ temos que $w = u$.

Sendo $\mathcal{C}'_p \equiv \mathcal{C}_p \ \wedge \ z = x$ podemos, usando argumentos análogos aos anteriores, simplificar a restrição c_2 para

$$
\begin{aligned}
(\mathsf{c}_3) \qquad & \mathcal{C}'_p \ \wedge \ ((w = u \ \wedge \ u.g^2 \ \wedge \ (u = a \ \vee \ x_1 = x)) \ \vee \ x_1 = x) \\
& \wedge \ (y.f_2^3 \doteq x \ \vee \ y.f_2^3 \doteq a) \ \wedge \ (y.f_3^3 \doteq u \ \vee \ y.f_3^3 \doteq b)
\end{aligned}
$$

Aplicando o mesmo processo ao primeiro elemento da primeira disjunção, e notando que

$$\mathcal{RT} \models \mathcal{C}'_p \ \wedge \ w = u \ \wedge \ u.g^2 \ \wedge \ u = a \ \Leftrightarrow \ \perp$$

obtemos

$$
\begin{aligned}
(\mathsf{c}_4) \qquad & \mathcal{C}'_p \ \wedge \ ((w = u \ \wedge \ u.g^2 \ \wedge \ x_1 = x) \ \vee \ x_1 = x) \\
& \wedge \ (y.f_2^3 \doteq x \ \vee \ y.f_2^3 \doteq a) \ \wedge \ (y.f_3^3 \doteq u \ \vee \ y.f_3^3 \doteq b)
\end{aligned}
$$

Factorizando $x_1 = x$ na primeira disjunção de c_4 e dado que um dos elementos dessa disjunção se reduz a $\top$, obtemos a restrição

$$
(\mathsf{c}_5) \qquad \mathcal{C}'_p \ \wedge \ (y.f_2^3 \doteq w \ \vee \ y.f_2^3 \doteq a) \ \wedge \ (y.f_3^3 \doteq u \ \vee \ y.f_3^3 \doteq b)
$$

Finalmente, é necessário analisar a independência para a satisfazibilidade das disjunções em c_5. Embora a variável y ocorra em ambas as disjunções, neste caso é fácil ver que elas são independentes, logo a restrição c_5 é satisfazível. Concluímos assim a satisfazibilidade da restrição c_1 sem ter de utilizar um algoritmo exponencial. Note-se que, no caso geral, para que $c_1 \ \vee \ c_2$ seja independente (para a satisfazibilidade) de c_3 , basta que um dos elementos da disjunção seja independente de c_3. Por exemplo, mesmo que $u = w$ pertencesse a $\mathcal{C}'_p$ não seria ainda necessário multiplicar as disjunções em c_5.

Na Secção 6.1 apresentamos algoritmos que dado um modelo (forma normal) permitem simplificar conjunções de restrições atómicas que sejam satisfazíveis nesse modelo. Dado um modelo parcial duma restrição c, esses algoritmos serão usados para simplificar c. Na Secção 6.2 é apresentado o sistema de reescrita $\mathsf{S}_{\mathcal{M}}$ que permite decidir a satisfazibilidade em $\mathcal{LU}$ e que traduz no essencial, as técnicas ilustradas no exemplo anterior. Versões deste sistema de reescrita foram apresentadas em [DMV94, DMB93]. Na Secção 6.3 é definida uma classe de restrições para a qual apenas um subconjunto do sistema de reescrita $\mathsf{S}_{\mathcal{M}}$ é necessário para decidir a satisfazibilidade e cuja complexidade é polinomial. A factorização de informação determinística comum a restrições é abordada na Secção 6.4. Finalmente, na Secção 6.5 os métodos estudados nas Secções 6.1 e 6.2 são estendidos, considerando que os modelos parciais são formas primárias.

6.1 Satisfazibilidade Incremental para Formas Normais

Começamos por apresentar um algoritmo para a satisfazibilidade duma conjunção $\mathcal{C}$ de restrições atómicas positivas, dada uma forma normal (modelo) $\mathcal{M}$, designada por *contexto*. Se $\mathcal{M} \ \wedge \ \mathcal{C}$ for satisfazível pretendemos obter as "diferenças" entre $\mathcal{M}$ e a forma normal de $\mathcal{M} \ \wedge \ \mathcal{C}$. Essas "diferenças" serão independentes para a satisfazibilidade de $\mathcal{M}$. Em particular, se $\mathcal{M}$ vincular $\mathcal{C}$ a "diferença" será $\top$.

Definição 6.1 *Sendo $\mathcal{M}$ um modelo, uma restrição $\mathcal{C} \in \mathcal{LU}$ é $\mathcal{M}$-satisfazível se $\mathcal{M} \ \wedge \ \mathcal{C}$ é satisfazível em $\mathcal{RT}$.*

A proposição seguinte generaliza o Lema 4.51, da página 127, e relaciona-o com a obtenção de formas resolvidas normais.

Proposição 6.2 *Seja $\mathcal{M}$ uma forma resolvida normal e $\mathcal{C}$ uma conjunção de restrições atómicas positivas. Se $\mathcal{C}$ é $\mathcal{M}$-satisfazível, então*

$$\mathcal{M} \models_{\mathcal{RT}} \mathcal{C} \ \Leftrightarrow \ \mathcal{M}_1$$

onde $\mathcal{M}_1$ resulta de $\mathcal{C}$ pela aplicação finita do sistema de reescrita $\mathsf{I}_{\mathcal{M}}$ constituído pelas seguintes regras de reescrita:

I1. $\mathcal{C} \mapsto_{\mathcal{M}} \mathcal{C}[v/x]$ *se* $x = v \in \mathcal{M}$ *e* $x \in \mathsf{Vars}(\mathcal{C})$;

I2. $\{y.f_i^n \doteq v'\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \{v = v'\} \cup \mathcal{C}$ *se* $x.f_i^n \doteq v \in \mathcal{M}$ *e* $x =_{\mathcal{C}} y$;

I3. $\{y.f^n\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \mathcal{C}$ *se* $x.f^n \in \mathcal{M}$, $x =_{\mathcal{C}} y$ *e* y *não é definido em* $\mathcal{C}$ *para nenhum* $1 \leq i \leq n$;

I4. $\{\top\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \mathcal{C}$

I5. $\{v = v\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \mathcal{C}$

I6. $\{a = x\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \{x = a\} \cup \mathcal{C}$

I7. $\{x.f_i^n \doteq v, x.f_i^n \doteq v'\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \{x.f_i^n \doteq v', v = v'\} \cup \mathcal{C}$

I8. $\{x = v\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \{x = v\} \cup \mathcal{C}[v/x]$ *se nenhuma regra anterior se aplica e* $x \in \mathsf{Vars}(\mathcal{C})$;

I9. $\mathcal{C} \mapsto_{\mathcal{M}} \{v = v'\} \cup \mathcal{C}$ *se nenhuma regra anterior se aplica,* $x.f_i^n \doteq v \in \mathcal{M}$ *e* $y.f_i^n \doteq v' \in \mathcal{M}$ *para* $f^n \in \Sigma$ e $1 \leq i \leq n$, $x =_{\mathcal{C}} y$, v *e* v' *não são simultaneamente constantes,* $v \not\equiv v'$ *e* $v \neq_{\mathcal{C}} v'$;

I10. $\{x.f^n, x.f^n\} \cup \mathcal{C} \mapsto \{x.f^n\} \cup \mathcal{C}$

I11. $\{x.f_i^n \doteq v\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \{x.f^n, x.f_i^n \doteq v\} \cup \mathcal{C}$ *se* $x.f^n \notin \mathcal{C}$; [1]

I12. $\mathcal{C} \mapsto_{\mathcal{M}} \mathcal{C} \cup \{x = y\}$ *se nenhuma regra anterior se aplica,* $x \sim_{\mathcal{M} \wedge \mathcal{C}} y$ *e* $x \neq_{\mathcal{M} \wedge \mathcal{C}} y$.

O sistema de reescrita termina e se $\mathcal{M}_1$ *é irredutível então* $\mathcal{M}_1$ *é uma forma resolvida normal e* $\mathcal{M} \wedge \mathcal{M}_1$ *coincide com a forma normal* $\mathcal{M}'$ *de* $\mathcal{M} \wedge \mathcal{C}$*, a menos de permutação de valores na mesma classe de equivalência de* $=_{\mathcal{M}'}$ *e repetição de restrições atómicas. Neste caso, escreve-se* $\mathcal{M}' \doteq \mathcal{M} \wedge \mathcal{M}_1$*. Em particular, se* $\mathcal{M} \equiv \top$ *então* $\mathcal{M}' \equiv \mathcal{M}_1$.

Dem. Cada regra do sistema de reescrita $\mathsf{I}_{\mathcal{M}}$ é correcta em relação a $\mathcal{M}$, se $\mathcal{C} \mapsto^{\mathsf{I}}_{\mathcal{M}} \mathcal{C}'$ implicar $\mathcal{M} \models_{\mathcal{RT}} \mathcal{C} \Leftrightarrow \mathcal{C}'$, ou, equivalentemente, se $\mathcal{RT} \models \mathcal{M} \wedge \mathcal{C} \Leftrightarrow \mathcal{M} \wedge \mathcal{C}'$. Excluindo as regras I2, I3, I8 e I9, o conjunto de regras definido acima coincide com o que seria usado na redução de $\mathcal{M} \wedge \mathcal{C}$ a forma resolvida normal, pelo sistema de reescrita R, porque $\mathcal{M}$ está em forma normal. Para esse conjunto de regras fica provada a correcção e que $\mathcal{M} \wedge \mathcal{M}_1$ corresponde à forma normal de $\mathcal{M} \wedge \mathcal{C}$. A regra I8 é correcta em qualquer álgebra de $\mathcal{L}$. A regra I3 é correcta porque se $x \not\equiv y$ corresponde, no sistema de reescrita R, ao resultado da aplicação da regra R4 seguida da regra R10. Cada aplicação das regras I2 ou I9 corresponde a uma aplicação da regra R6, na redução a forma resolvida. No caso da regra I2, se $x \not\equiv y$, essa aplicação seria precedida por outras regras em R, nomeadamente, duma aplicação da regra R4. Aplicando a regra I9

[1] *Para a satisfazibilidade bastaria considerar se* $y.f^n \notin \mathcal{M} \wedge \mathcal{C}$ *e* $x =_{\mathcal{C}} y$*, mas neste caso* $\mathcal{M}_1$ *não seria uma forma resolvida.*

não se elimina uma restrição da forma $x.f_i^n \doteq v$ em $\mathcal{M}$, mas essa restrição coincide com $y.f_i^n \doteq v'$ a menos de permutação de valores numa classe de equivalência de $=_{\mathcal{M} \wedge \mathcal{C}}$. Para cada aplicação da regra I8, uma vez que as regras I1 e I2 foram aplicadas primeiro, é garantido que se x ocorre em $\mathcal{M}$ então x é livre em $\mathcal{M}$ ou é funcional, e em ambos os casos $\mathcal{M} \wedge x = v$ é satisfazível e difere duma forma resolvida a menos de permutação de valores na mesma classe de equivalência de $=_{\mathcal{M} \wedge x=v}$. Isto prova a correcção desta regra. Temos que $\mathcal{M}_1$ é uma forma resolvida e a regra I12 restrita a $\mathcal{C}$ garante que $\mathcal{M}_1$ é normal. Apesar da semelhança entre este sistema de reescrita e o usado para redução a forma normal é necessário ainda garantir a sua terminação. A aplicação das regras I3, I4, I5, I6, I10 e I11 não podem conduzir a derivações infinitas, portanto apenas analisaremos a terminação das restantes regras. Nenhuma regra faz aumentar o número de variáveis. Dado um modelo $\mathcal{M}$, consideremos as seguintes funções de $\mathcal{LU}$ em $\mathbb{N}$:

$\gamma_1(\mathcal{C})$: número de variáveis não resolvidas em $\mathcal{M} \wedge \mathcal{C}$

$\gamma_2(\mathcal{C})$: número de restrições da forma $x.f_i^n \doteq v$ em $\mathcal{C}$

$\gamma_3(\mathcal{C})$: número de classes de equivalência da relação $=_{\mathcal{M} \wedge \mathcal{C}}$

$\gamma_4(\mathcal{C})$: número de variáveis não resolvidas em $\mathcal{C}$

O conjunto de elementos $\nu(\mathcal{C}) = (\gamma_1(\mathcal{C}), \gamma_2(\mathcal{C}), \gamma_3(\mathcal{C}), \gamma_4(\mathcal{C}))$ ordenado lexicograficamente é bem-fundado. Provando que ν é estritamente decrescente pela aplicação das regras de $\mathsf{I}_{\mathcal{M}}$ (excluindo as atrás referidas), garantimos a terminação, dado que uma derivação infinita corresponderia a uma sequência infinita estritamente decrescente de elementos ν, o que é impossível. A aplicação das regras atrás referidas não altera o valor de nenhuma destas funções. O quadro da tabela 6.1 resume a variação de cada γ_i, para $1 \leq i \leq 4$, pela aplicação de cada uma das restantes regras. A aplicação da regra I1 faz

$\overset{\mathsf{I}}{\mapsto}_{\mathcal{M}}$	γ_1	γ_2	γ_3	γ_4
I1	$<$			
I2	$\leq$	$<$		
I7	$\leq$	$<$		
I8	$\leq$	$=$	$\leq$	$<$
I9	$\leq$	$=$	$<$	
I12	$=$	$=$	$<$	

Tabela 6.1: Quadro da variação de γ_i

diminuir γ_1 porque resolve uma variável de $\mathcal{M} \wedge \mathcal{C}$, não resolvida em $\mathcal{C}$. A regra I2 não aumenta γ_1, pois se v ou v' são variáveis elas não estão resolvidas em $\mathcal{M} \wedge \mathcal{C}$, antes da sua aplicação. Esta regra decresce estritamente γ_2. Pelas mesmas razões a regra I7 não aumenta γ_1 e decresce estritamente γ_2. A regra I8 não pode fazer aumentar o número de variáveis não resolvidas em $\mathcal{M}$, portanto não aumenta γ_1. O valor de γ_2 não é alterado pois ou v é uma constante e como $\mathcal{M} \wedge \mathcal{C}$ é satisfazível não existe para esse x restrição da forma $x.f_i^n \doteq v'$, ou v é uma variável e no máximo há uma substituição de variável nas restrições daquela forma. O valor de γ_3 não pode aumentar e pode diminuir[2] se existir em $\mathcal{C}$ outra restrição da forma $x = v'$ e $v \neq_{\mathcal{C}} v'$. Esta regra decresce estritamente o valor de γ_4, porque resolve uma variável em $\mathcal{C}$. A regra I9 não aumenta γ_1, por razões análogas às da regra I2, e mantém o valor de γ_2. Pelas condições impostas à aplicação desta regra, γ_3 decresce estritamente. Note-se que como v e v' são distintas é garantido que $v \neq_{\mathcal{M}} v'$. Finalmente, a regra I12 não altera γ_1 nem γ_2 e decresce estritamente γ_3. 2

Consideremos as restrições

$$
\begin{array}{llll}
(\mathsf{c}_6) & \mathcal{M} & \equiv & x.f^2 \wedge y.f^2 \wedge x.f_2^2 \doteq z \wedge y.f_2^2 \doteq b \wedge u = a \\
 & \mathcal{C} & \equiv & x.f_1^2 \doteq w \wedge x.f_2^2 \doteq b \wedge y.f_1^2 \doteq u \wedge w = a
\end{array}
$$

Aplicando o sistema de reescrita $\mathsf{I}_{\mathcal{M}}$ a $\mathcal{C}$ obtemos a restrição

$$
(\mathsf{c}_7) \qquad \mathcal{M}_1 \equiv x.f^2 \wedge x.f_1^2 \doteq a \wedge y = x \wedge z = b \wedge w = a
$$

Corolário 6.3 *Sejam $\mathcal{M}$ uma forma resolvida normal, $\mathcal{C}$ uma conjunção de restrições atómicas positivas e $\mathcal{M}_1$ uma forma resolvida obtida de $\mathcal{C}$ e irredutível para o sistema de reescrita $\mathsf{I}_{\mathcal{M}}$. Então*

(i) $\mathcal{M}_1$ é única a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{M}_1}$;

(ii) $\mathcal{M}_1 \equiv \top$ se e só se $\mathcal{M} \models_{\mathcal{RT}} \mathcal{C}$;

(iii) Seja $c_1 \in \mathcal{LU}$ uma restrição atómica positiva satisfazível não equivalente a $\top$. Se $\mathcal{M}_1 \not\equiv \top$ e $\mathcal{M}_1 \models_{\mathcal{RT}} c_1$ então $\mathcal{M} \not\models_{\mathcal{RT}} c_1$.

Dem.

(i) É consequência da Proposição 4.20.

(ii) Pela correcção do sistema $\mathsf{I}_{\mathcal{M}}$, $\mathcal{C} \overset{\mathsf{I}}{\mapsto}{}^{\star}_{\mathcal{M}} \top$ se e só se $\mathcal{RT} \models \mathcal{M} \wedge \mathcal{C} \Leftrightarrow \mathcal{M}$, isto é, $\mathcal{M} \models_{\mathcal{RT}} \mathcal{C}$.

[2] Aliás γ_3 é não crescente pela aplicação de qualquer regra de $\mathsf{I}_{\mathcal{M}}$.

(iii) Por indução na estrutura de c_1.

Caso $c_1 \equiv x = v$**.** Temos que $x \in [v]_{\mathcal{M}_1}$. Se $\mathcal{M} \models_{\mathcal{RT}} x = v$ então $x \in [v]_{\mathcal{M}}$ e $\mathcal{M}_1$ não era irredutível para o sistema de reescrita $\mathsf{I}_{\mathcal{M}}$ uma vez que a regra I1 se aplicava.

Caso $c_1 \equiv x.f_i^n \doteq v$**.** Temos que $x \in [y]_{\mathcal{M}_1}$, $v \in [v']_{\mathcal{M}_1}$ e $y.f_i^n \doteq v' \in \mathcal{M}_1$. Se $\mathcal{M} \models_{\mathcal{RT}} x.f_i^n \doteq v$ então $x.f_i^n \doteq v \in \mathcal{M}$, uma vez que as variáveis de $\mathcal{M}_1$ não podem ser resolvidas em $\mathcal{M}$. Neste caso, a regra I2 aplicava-se.

Caso $c_1 \equiv x.f^n$**.** Temos que $x \in [y]_{\mathcal{M}_1}$ e $y.f^n \in \mathcal{M}_1$. Como no caso anterior, se $\mathcal{M} \models_{\mathcal{RT}} x.f^n$, então $x.f^n \in \mathcal{M}$ e a regra I3 aplicava-se.

2

Por outro lado, é imediato do Corolário 4.5 que

Corolário 6.4 *Sendo $\mathcal{M}$ uma forma normal e $\mathcal{C}$ uma conjunção de restrições atómicas positivas, $\mathcal{RT} \not\models \mathcal{M} \wedge \mathcal{C}$ se e só se $\mathcal{M} \wedge \mathcal{C} \overset{\mathsf{R}}{\mapsto}^{\star} \bot$.*

Dem. Pela correcção do sistema de reescrita R. 2

Os dois corolários anteriores, justificam as afirmações da página 113, referentes à satisfazibilidade de conjunções de restrições atómicas. O corolário seguinte resume essas afirmações.

Corolário 6.5 *Sendo $\mathcal{C}_p$ uma forma normal e n_i uma restrição negativa não trivial, verifica-se uma das seguintes condições, onde c_i denota $\neg n_i$:*

(i) $\mathcal{C}_p \wedge c_i \overset{\mathsf{R}}{\mapsto}^{\star} \bot$ *se e só se* $\mathcal{C}_p \wedge n_i$ *é satisfazível em* $\mathcal{RT}$ *e* $\mathcal{C}_p \models_{\mathcal{RT}} n_i$.

(ii) $c_i \overset{\mathsf{I}}{\mapsto}^{\star}_{\mathcal{C}_p} \top$ *se e só se* $\mathcal{C}_p \models_{\mathcal{RT}} c_i$ *e* $\mathcal{C}_p \wedge n_i$ *é não satisfazível em* $\mathcal{RT}$.

(iii) $c_i \overset{\mathsf{I}}{\mapsto}^{\star}_{\mathcal{C}_p} \mathcal{C}_i$, *onde* $\mathcal{C}_i$ *ou é da forma* $x.f^n$ *ou uma conjunção de restrições da forma* $v = v'$, *se e só se* $\mathcal{C}_p \models_{\mathcal{RT}} n_i \Leftrightarrow \neg c'_1 \vee \ldots \vee \neg c'_m$ *com* $\mathcal{C}_i \equiv c'_1 \wedge \ldots \wedge c'_m$.

Dem. Apenas se considera a demonstração do caso (iii), por análise da forma de n_i. Seja n_i da forma $\neg x.f^n$. Se $x = v \in \mathcal{C}_p$ então $\mathcal{C}_i \equiv v.f^n$, senão $\mathcal{C}_i \equiv x.f^n$. Se n_i é da forma $v \neq v'$ então é fácil ver que as regras de $\mathsf{I}_{\mathcal{C}_p}$ que se podem aplicar só envolvem a eliminação ou a adição de igualdades. Se v ou v' são constantes as únicas regras que se podem aplicar são a I1, I4, I5 ou I6. Se v e v' são variáveis, para além das regras referidas podem também aplicar-se as regras I8, I9 ou I12. 2

A condição (iii) do corolário anterior permite, se forem eliminadas as redundâncias, a obtenção duma disjunção de formas primárias normais equivalente em $\mathcal{RT}$ a $\mathcal{C}_p \wedge n_i$.

Os resultados sobre satisfazibilidade para uma conjunção de restrições atómicas positivas dado um *contexto* podem ser generalizados para a conjunção finita de conjunções de restrições atómicas positivas (ou formas normais). Esta generalização corresponde a um processo incremental de satisfazibilidade.

Corolário 6.6 *Sendo $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_m$ conjunções de restrições atómicas positivas, se*

$$\mathcal{C}_0 \wedge \mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_m$$

é satisfazível em $\mathcal{RT}$ então, a forma normal $\mathcal{M}'$ de $\mathcal{C}_0 \wedge \mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_m$ coincide, a menos de permutação de valores na mesma classe de equivalência de $=_{\mathcal{M}'}$ e repetição de restrições atómicas, com $\mathcal{M}_0 \wedge \mathcal{M}_1 \wedge \ldots \wedge \mathcal{M}_m$, onde $\mathcal{M}_i$ e $\mathcal{M}'_i$, para $0 \leq i \leq m$, são definidas por:

a) $\mathcal{M}'_0 \equiv \mathcal{M}_0 \quad$ *se* $\quad \mathcal{C}_0 \overset{\mathsf{R}}{\mapsto^\star} \mathcal{M}_0$

b) $\mathcal{M}'_{i+1} \doteq \mathcal{M}'_i \wedge \mathcal{M}_{i+1} \quad$ *se* $\quad \mathcal{C}_{i+1} \overset{\mathsf{I}}{\mapsto^\star_{\mathcal{M}'_i}} \mathcal{M}_{i+1}, \quad$ para $0 \leq i \leq m-1$

Dem. Por indução no número de conjunções de restrições atómicas positivas. 2

A satisfazibilidade incremental e o sistema de reescrita $\mathsf{I}_{\mathcal{M}}$ podem ser generalizados para o caso de $\mathcal{C}$ ser uma conjunção de restrições atómicas positivas e negativas.

Proposição 6.7 *Seja $\mathcal{M}$ uma forma resolvida normal e $\mathcal{C} \equiv \mathcal{C}_p \wedge \mathcal{C}_n$ uma conjunção de restrições atómicas. Se $\mathcal{C}$ é $\mathcal{M}$-satisfazível, então $\mathcal{M} \models_{\mathcal{RT}} \mathcal{C} \Leftrightarrow \mathcal{C}^1_p \wedge \mathcal{C}^1_n$, onde $\mathcal{C}^1_p$ é irredutível para o sistema de reescrita $\mathsf{I}_{\mathcal{M}}$ e $\mathcal{C}^1_p \wedge \mathcal{C}^1_n$ é irredutível para o sistema de reescrita $\mathsf{IP}_{\mathcal{M}}$, a seguir apresentado:*

IP1. $\mathcal{C}_p \cup \mathcal{C}_n \mapsto_{\mathcal{M}} \mathcal{C}^1_p \cup \mathcal{C}_n$ *se* $\mathcal{C}_p \overset{\mathsf{I}}{\mapsto^\star_{\mathcal{M}}} \mathcal{C}^1_p$ *e* $\mathcal{C}^1_p$ *é irredutível para* $\mathsf{I}_{\mathcal{M}}$

IP2. $\mathcal{C} \mapsto_{\mathcal{M}} \mathcal{C}[v/x]$ *se* $x = v \in \mathcal{M}$ *e* $x \in \mathsf{Vars}(\mathcal{C})$;

IP3. $\{x = v\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \{x = v\} \cup \mathcal{C}[v/x]$ *se* $x \in \mathsf{Vars}(\mathcal{C})$;

IP4. $\{a \neq x\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \{x \neq a\} \cup \mathcal{C}$

IP5. $\{a \neq b\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \mathcal{C}$ *se a e b são constantes distintas;*

IP6. $\{\neg a.f^n\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \mathcal{C}$

IP7. $\{x \neq a\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \mathcal{C}$ *se* $x.f^n \in \mathcal{C}^1_p$ *ou* $y.f^n \in \mathcal{M}$ *e* $x \in [y]_{\mathcal{C}^1_p}$, *para algum* $f^n \in \Sigma$;

IP8. $\{\neg x.g^m\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \mathcal{C}$ *se* $x.f^n \in \mathcal{C}_p^1$ *ou* $y.f^n \in \mathcal{M}$ *e* $x \in [y]_{\mathcal{C}_p^1}$, *para algum* $f^n \neq g^m$*;*

IP9. $\{x \neq y, y \neq x\} \cup \mathcal{C} \mapsto_{\mathcal{M}} \{x \neq y\} \cup \mathcal{C}$

IP10. $\{x \neq v, x \neq v\} \cup \mathcal{C} \mapsto \{x \neq v\} \cup \mathcal{C}$

IP11. $\{\neg x.f^n, \neg x.f^n\} \cup \mathcal{C} \mapsto \{\neg x.f^n\} \cup \mathcal{C}$

O sistema de reescrita $\mathsf{IP}_{\mathcal{M}}$ *termina e* $\mathcal{M} \wedge \mathcal{C}_p^1 \wedge \mathcal{C}_n^1$ *coincide com a forma primária* $\mathcal{P} \equiv \mathcal{P}_p \wedge \mathcal{P}_n$ *de* $\mathcal{M} \wedge \mathcal{C}$ *a menos de permutação de valores na mesma classe de equivalência de* $=_{\mathcal{P}_p}$ *e repetição de restrições atómicas. Neste caso, escreve-se* $\mathcal{P} \doteq \mathcal{M} \wedge \mathcal{C}_p^1 \wedge \mathcal{C}_n^1$*.*

Dem. Cada regra do sistema de reescrita $\mathsf{IP}_{\mathcal{M}}$ é correcta em relação a $\mathcal{M}$, se $\mathcal{C} \overset{\mathsf{IP}}{\mapsto}_{\mathcal{M}} \mathcal{C}'$ implicar $\mathcal{M} \models_{\mathcal{RT}} \mathcal{C} \Leftrightarrow \mathcal{C}'$, ou, equivalentemente, se $\mathcal{RT} \models \mathcal{M} \wedge \mathcal{C} \Leftrightarrow \mathcal{M} \wedge \mathcal{C}'$. A regra de reescrita IP1 é correcta pela correcção do sistema de reescrita $\mathsf{I}_{\mathcal{M}}$ e as regras IP2–IP11 coincidem com regras do sistema de reescrita P, para redução a forma primária, considerando a forma normal $\mathcal{P}_p \doteq \mathcal{M} \wedge \mathcal{C}_p^1$. A terminação resulta da terminação dos sistemas $\mathsf{I}_{\mathcal{M}}$ e P, e da sua ordem de aplicação. Resulta ainda que $\mathcal{C}_p^1$ está em forma normal e se $\mathcal{C}_p^1 \wedge \mathcal{C}_n^1$ é irredutível para $\mathsf{IP}_{\mathcal{M}}$ então é uma forma primária.
2

Consideremos as restrições

$$
(\mathsf{c}_8) \qquad \begin{aligned} \mathcal{M} &\equiv x.f^2 \wedge y.f^2 \wedge x.f_2^2 \doteq b \wedge y.f_2^2 \doteq z \wedge u = a \\ \mathcal{C} &\equiv x.f_1^2 \doteq a \wedge x.f_2^2 \doteq w \wedge y \neq b \wedge w \neq a \end{aligned}
$$

Aplicando o sistema de reescrita $\mathsf{IP}_{\mathcal{M}}$ a $\mathcal{C}$ obtemos a restrição

$$
(\mathsf{c}_9) \qquad \mathcal{C}_1 \equiv x.f^2 \wedge x.f_1^2 \doteq a \wedge w = b
$$

6.2 O Sistema de Reescrita Completo $\mathsf{S}_{\mathcal{M}}$

Nesta secção vamos apresentar o sistema de reescrita que está na base do processamento de restrições nos formalismos de gramáticas lógicas com restrições $\mathcal{CLG}$, [DMV91, DV92, DMV94]. Esta apresentação difere das de [DMV94, DMB93] pelo facto de se usarem os sistemas de reescrita para a satisfazibilidade incremental, da secção anterior, e a redução a forma primária. A correcção, a terminação, a completude e a complexidade do sistema de reescrita, embora indicadas em [DMV94], não tinham sido apresentadas formalmente.

Como anteriormente, denotamos por

- $\mathcal{C}_p$, $\mathcal{C}'_p$, ... as conjunções de restrições atómicas positivas;
- $\mathcal{C}_n$, $\mathcal{C}'_n$, ... as conjunções de restrições atómicas negativas;
- $\mathcal{M}$, $\mathcal{M}'$, $\mathcal{M}_0$, ... as formas resolvidas;
- $\mathcal{P}$, $\mathcal{P}'$, $\mathcal{P}_0$ ... as formas primárias;
- $\mathcal{C}$, $\mathcal{C}'$, $\mathcal{C}_0$, ... as conjunções de restrições.

Vamos supor que qualquer $\mathcal{C} \in \mathcal{LU}$ está na forma geral (6.1), definida na página 162,

$$\mathcal{C}_p \ \wedge \ \mathcal{C}_n \ \wedge \ \mathcal{D}isj_1 \ \wedge \ \ldots \ \wedge \ \mathcal{D}isj_m$$

As seguintes regras de reescrita podem ser usadas para essa transformação:

G1.	$(c_1 \ \wedge \ c_2) \ \wedge \ c_3$	$\mapsto$	$c_1 \ \wedge \ (c_2 \ \wedge \ c_3)$
G2.	$c_1 \ \wedge \ (c_2 \ \wedge \ c_3)$	$\mapsto$	$c_2 \ \wedge \ (c_1 \ \wedge \ c_3)$ se c_2 é atómica positiva e c_1 não é atómica ou se c_2 é atómica negativa e c_1 não é atómica;
G3.	$c_1 \ \wedge \ c_2$	$\mapsto$	$c_2 \ \wedge \ c_1$ se c_2 é atómica e c_1 não é atómica ou se c_1 é atómica negativa e c_2 é atómica positiva.

A correcção destas regras resulta da relação $\equiv_{\mathcal{B}}$. A terminação prova-se adaptando as técnicas em [DM79] ou [Der82] para o caso dum sistema de reescrita só com a regra G1. Uma restrição irredutível para este sistema de reescrita, está na forma geral (6.1) (omitindo os parênteses).

Para qualquer restrição $\mathcal{C}_0 \in \mathcal{LU}$ o sistema de reescrita $\mathsf{S}_{\mathcal{M}}$ que iremos apresentar ou produz $\perp$ ou um modelo parcial $\mathcal{M}$ de $\mathcal{C}_0$ e uma restrição $\mathcal{C}$, possivelmente mais simples, e tal que:

- $\mathcal{RT} \models \mathcal{C}_0 \ \Leftrightarrow \ \mathcal{M} \ \wedge \ \mathcal{C}$
- Qualquer modelo minimal de $\mathcal{C}_0$ pode ser obtido pela reunião de $\mathcal{M}$ com um modelo minimal de $\mathcal{C}$.
- Se $\mathcal{C}$ não contiver negações, qualquer modelo minimal $\mathcal{M}_c$ de $\mathcal{C}$ é $\mathcal{M}$-satisfazível. E, neste caso, a reunião de $\mathcal{M}_c$ com $\mathcal{M}$ é um modelo minimal de $\mathcal{C}_0$.
- O sistema de reescrita $\mathsf{S}_{\mathcal{M}}$ é completo no sentido em que $\mathcal{C}_0$ é satisfazível a menos que

seja produzido $\perp$, e, neste caso, $\mathcal{RT} \models \mathcal{C}_0 \Leftrightarrow \perp$.

- Se a regra S8 do sistema de reescrita $\mathsf{S}_{\mathcal{M}}$ for omitida, o sistema de reescrita é incompleto mas a sua complexidade é polinomial.

O sistema de reescrita pode ser subdividido em dois passos: um que "extrai" um *factor determinístico* e outro que "usa" esse factor (modelo) para simplificar a restrição resultante. No primeiro passo serão reescritos pares $\langle \mathcal{M}, \mathcal{C} \rangle$, correspondendo à conjunção $\mathcal{M} \wedge \mathcal{C}$, considerando-se inicialmente o par $\langle \top, \mathcal{C}_0 \rangle$.

O sistema de reescrita SP para pares $\langle \mathcal{M}, \mathcal{C} \rangle$ é definido pelas seguintes regras:

$$
\begin{array}{lll}
\langle \mathcal{M}, \{\perp\} \cup \mathcal{C} \rangle & \Rightarrow \langle \perp, \emptyset \rangle & \text{se } \mathcal{M} \not\equiv \perp \\
\langle \mathcal{M}, \{\top\} \cup \mathcal{C} \rangle & \Rightarrow \langle \mathcal{M}, \mathcal{C} \rangle & \\
\langle \mathcal{M}, \{x = v\} \cup \mathcal{C} \rangle & \Rightarrow \langle \mathcal{M} \cup \{x = v\}, \mathcal{C} \rangle & \\
\langle \mathcal{M}, \{x.f^n\} \cup \mathcal{C} \rangle & \Rightarrow \langle \mathcal{M} \cup \{x.f^n\}, \mathcal{C} \rangle & \\
\langle \mathcal{M}, \{x.f_i^n \doteq v\} \cup \mathcal{C} \rangle & \Rightarrow \langle \mathcal{M} \cup \{x.f_i^n \doteq v\}, \mathcal{C} \rangle & \\
\langle \perp, \mathcal{C} \rangle & \Rightarrow \langle \perp, \emptyset \rangle & \\
\langle \mathcal{M}, \mathcal{C} \rangle & \Rightarrow \langle \mathcal{M}, \mathcal{C}' \rangle & \text{se } \mathcal{C} \overset{\mathsf{S}}{\longmapsto}{}^{\star}_{\mathcal{M}} \mathcal{C}'
\end{array}
$$

com a convenção de que após cada aplicação duma das regras de reescrita o novo modelo parcial $\mathcal{M}$ é reduzido a forma resolvida normal ou a $\perp$.

Definição 6.8 *Seja $\mathcal{M}$ um modelo ou $\perp$ e $\mathcal{C}$, $\mathcal{C}' \in \mathcal{LU}$ restrições. Denota-se $\mathcal{C} \longmapsto_{\mathcal{M}} \mathcal{C}'$ se $\mathcal{M} \wedge \mathcal{C}$ se reescreve em $\mathcal{M} \wedge \mathcal{C}'$ pela aplicação de uma regra do sistema de reescrita $\mathsf{S}_{\mathcal{M}}$, a uma qualquer sub-expressão de $\mathcal{C}$.*

O conjunto de regras de reescrita $\mathsf{S}_{\mathcal{M}}$ para uma restrição $\mathcal{C}$ e modelo $\mathcal{M}$ (ou $\perp$), é apresentado na Tabela 6.3. Cada regra pode ser aplicada a uma sub-restrição da restrição $\mathcal{C}$, que satisfaça as condições de aplicabilidade. Salvo indicação em contrário, quando escrevemos $c \mapsto^\star c'$, então c' é irredutível para o sistema de reescrita $\mapsto$.

O sistema de reescrita SP pode ser englobado no sistema de reescrita $\mathsf{S}_{\mathcal{M}}$, se inicialmente considerarmos $\mathcal{M} \equiv \top$ e no final se obtiver, ou $\perp$ caso em que $\mathcal{M} \equiv \perp$, ou uma restrição da forma $\mathcal{M} \wedge \mathcal{C}$, pela regra S7. A introdução do sistema SP apenas facilita a exposição, sendo as suas propriedades as do sistema de reescrita $\mathsf{S}_{\mathcal{M}}$, a seguir apresentadas.

Na relação $\mathcal{M}' \doteq \mathcal{M} \wedge \mathcal{C}_p^2$, usada na regra S7, a restrição $\mathcal{C}_p^2$ corresponde às "diferenças" entre as formas normais $\mathcal{M}'$ e $\mathcal{M}$ pela "adição" de $\mathcal{C}_1$, como definido na Proposição 6.2.

Denotamos por $\mathsf{S}_{\mathcal{M}}^-$ o sistema de reescrita que resulta de $\mathsf{S}_{\mathcal{M}}$ excluindo a regra S8.

S1.	$\mathcal{C}$	$\longmapsto_{\perp}$	$\perp$
S2.	$\mathcal{C} \wedge \top$	$\longmapsto_{\mathcal{M}}$	$\mathcal{C}$
S3.	$\mathcal{C} \wedge \perp$	$\longmapsto_{\mathcal{M}}$	$\perp$
S4.	$\mathcal{C} \vee \top$	$\longmapsto_{\mathcal{M}}$	$\top$
S5.	$\mathcal{C} \vee \perp$	$\longmapsto_{\mathcal{M}}$	$\mathcal{C}$
S6.	$\mathcal{C}_1 \wedge \mathcal{C}$	$\longmapsto_{\mathcal{M}}$	$\perp$ se nenhuma das regras anteriores se aplica, $\mathcal{C}_1$ é uma conjunção, não vazia, de restrições atómicas, $\mathcal{C}$ é uma conjunção, possivelmente vazia, de restrições não atómicas e $\mathcal{M} \wedge \mathcal{C}_1 \overset{\mathsf{P}}{\mapsto}{}^{\star} \perp$;
S7.	$\mathcal{C}_1 \wedge \mathcal{C}$	$\longmapsto_{\mathcal{M}}$	$\mathcal{C}_p^2 \wedge \mathcal{C}_n^2 \wedge \mathcal{C}'$ se nenhuma das regras anteriores se aplica, $\mathcal{C}_1$ é uma conjunção, não vazia, de restrições atómicas, $\mathcal{C}$ é uma conjunção, possivelmente vazia, de restrições não atómicas, $\mathcal{M} \wedge \mathcal{C}_1 \overset{\mathsf{P}}{\mapsto}{}^{\star} \mathcal{M}' \wedge \mathcal{C}_n^2$ $\mathcal{C} \overset{\mathsf{S}}{\longmapsto}{}^{\star}_{\mathcal{M}'} \mathcal{C}'$ $\mathcal{M}' \dot{\equiv} \mathcal{M} \wedge \mathcal{C}_p^2$ e $\mathcal{C} \not\equiv \mathcal{C}'$ ou $\mathcal{C}_1 \not\equiv \mathcal{C}_p^2 \wedge \mathcal{C}_n^2$;
S8.	$(\mathcal{C}_1 \vee \mathcal{C}_2) \wedge \mathcal{C}_3$	$\longmapsto_{\mathcal{M}}$	$(\mathcal{C}_1 \wedge \mathcal{C}_3) \vee (\mathcal{C}_2 \wedge \mathcal{C}_3)$ se nenhuma das regras anteriores se aplica, $\mathcal{C}_3$ não é uma conjunção de restrições atómicas positivas e nem $\mathcal{C}_1$ nem $\mathcal{C}_2$ são $\mathcal{M}$-independentes com $\mathcal{C}_3$.

Tabela 6.3: Sistema de reescrita $\mathsf{S}_{\mathcal{M}}$

Proposição 6.9 *O sistema de reescrita $\mathsf{S}^{-}_{\mathcal{M}}$ é correcto (em $\mathcal{RT}$) e termina.*

Dem. O sistema de reescrita $\mathsf{S}^{-}_{\mathcal{M}}$ é correcto se $\mathcal{C} \in \mathcal{LU}$ e $\mathcal{C} \longmapsto_{\mathcal{M}} \mathcal{C}'$, então $\mathcal{M} \models_{\mathcal{RT}} \mathcal{C} \Leftrightarrow \mathcal{C}'$. As regras $\mathsf{S1}$ a $\mathsf{S5}$ são correctas em qualquer álgebra de $\mathcal{L}$. A correcção das regras $\mathsf{S6}$ e $\mathsf{S7}$ resulta da correcção dos sistemas de reescrita P e $\mathsf{IP}_{\mathcal{M}}$ e da correcção das restantes regras deste sistema. Para a terminação, começamos por dar uma justificação informal e depois construiremos uma ordenação bem-fundada associada a $\mathsf{S}^{-}_{\mathcal{M}}$. Considerando a forma geral (6.1) para cada $\mathcal{C} \in \mathcal{LU}$, podemos associar as seguintes funções de $\mathcal{LU}$ em $\mathbb{N}$:

$\gamma_c(\mathcal{C})$ **:** número de conjunções de restrições atómicas incluindo $\top$ e $\perp$

$\gamma_{np}(\mathcal{C})$ **:** número de conjunções de restrições atómicas não formas primárias

O sistema de reescrita reduz as conjunções de restrições atómicas a formas primárias e simplifica as restrições usando as propriedades booleanas. Assim, nenhuma aplicação de uma regra faz aumentar γ_c.

As regras $\mathsf{S2}$ a $\mathsf{S5}$ fazem diminuir γ_c e γ_{np}.

A aplicação das regras $\mathsf{S1}$, $\mathsf{S6}$ e $\mathsf{S7}$ se $\mathcal{C}^2_p \equiv \top$, obrigam à aplicação duma das regras $\mathsf{S2}$ a $\mathsf{S5}$ (caso contrário, mais nenhuma regra se aplica). Assim, simplificando, podemos supor que aquelas regras decrescem γ_c e γ_{np}.

A única regra que poderia conduzir a uma derivação infinita é a regra $\mathsf{S7}$.

A regra $\mathsf{S7}$, dado um modelo parcial, reduz conjunções de restrições atómicas a forma primária e constrói um novo modelo – temporário – para a restrições "abaixo". Pela unicidade das formas primárias, a regra $\mathsf{S7}$ aplica-se no máximo uma vez para cada conjunção de restrições atómicas positivas de $\mathcal{C}$, antes da aplicação de outra regra. Se na aplicação da regra $\mathsf{S7}$, $\mathcal{C} \equiv \top$, a terminação resulta da terminação do sistema de reescrita $\mathsf{IP}_{\mathcal{M}}$. Se $\mathcal{C} \not\equiv \top$, a não terminação só podia resultar por aplicação da regra $\mathsf{S7}$, pois γ_c e γ_{np} decresce em cada aplicação recursiva.

Por outro lado, a regra $\mathsf{S7}$ só se aplica sucessivamente no topo, isto é,

$$\mathcal{C}_1 \wedge \mathcal{C} \overset{\mathsf{S7}}{\longmapsto}_{\mathcal{M}} \mathcal{C}^2_p \wedge \mathcal{C}^2_n \wedge \mathcal{C}' \overset{\mathsf{S7}}{\longmapsto}_{\mathcal{M}} \dots$$

se $\mathcal{C}'$ for uma conjunção que contém restrições atómicas positivas (ou negativas), caso contrário, alguma regra de $\longmapsto_{\mathcal{M}'}$ ainda se aplicava. Neste caso, essas restrições serão "adicionadas" a $\mathcal{C}^2_p \wedge \mathcal{C}^2_n$, e, portanto, o número total de conjunções de restrições atómicas, γ_c, diminui. Podemos concluir que não pode haver nenhuma derivação infinita usando a regra $\mathsf{S7}$.

Para associar uma ordenação bem-fundada a $\mathsf{S}_{\mathcal{M}}^{-}$ e considerando o Teorema 4.58, definimos[3] $\Delta(\mathcal{C})$ como o menor conjunto tal que:

(i) $\Delta(\mathcal{C}) = \{\mathcal{C}\}$ se $\mathcal{C}$ é uma conjunção de restrições atómicas

(ii) $\Delta(\mathcal{C}_1 \vee \mathcal{C}_2) = \Delta(\mathcal{C}_1) \cup \Delta(\mathcal{C}_2)$

(iii) $\Delta(\mathcal{C}_1 \wedge \mathcal{C}_2) = \{\mathcal{C} \dot{\wedge} \mathcal{C}' \mid \mathcal{C} \in \Delta(\mathcal{C}_1) \text{ e } \mathcal{C}' \in \Delta(\mathcal{C}_2)\}$, onde $\mathcal{C}_1$ e $\mathcal{C}_2$ não são simultaneamente conjunções de restrições atómicas e $\dot{\wedge}$ indica a conjunção de conjunções de restrições atómicas, consideradas como fórmulas atómicas.

Note-se que a disjunção dos elementos de $\Delta(\mathcal{C})$ coincide com a forma normal disjuntiva de $\mathcal{C}$ (ver [And86]). Por outro lado, cada $\mathcal{D}_i \in \Delta(\mathcal{C})$ é da forma $\mathcal{C}_1 \dot{\wedge} \ldots \dot{\wedge} \mathcal{C}_{r_i}$, onde para cada $1 \leq j \leq r_i$, $\mathcal{C}_j \equiv \mathcal{C}_p^j \wedge \mathcal{C}_n^j$ é uma conjunção de restrições atómicas correspondendo a um dos factores obtidos pelo passo (iii), da definição de Δ. Para cada $\mathcal{D}_i$, dizemos que $\mathcal{C}_j$, para algum $1 \leq j \leq r_i$, é uma *forma semi-primária* se $\mathcal{C}_j$ é fechada para $\mathsf{IP}_{\mathcal{C}_p^1 \wedge \ldots \wedge \mathcal{C}_p^{j-1}}$.

Para um modelo $\mathcal{M}$ e para cada $\mathcal{D}_i \in \Delta(\mathcal{C})$ definimos as seguintes funções em $\mathbb{N}$:

$\gamma_1(\mathcal{D}_i)$ **:** número de conjunções de restrições atómicas (r_i)

$\gamma_2(\mathcal{D}_i)$ **:** número de conjunções de restrições atómicas não em forma primária

$\gamma_3(\mathcal{D}_i)$ **:** número de conjunções de restrições atómicas não formas semi-primárias

Seja $\upsilon(\mathcal{D}_i) = (\gamma_1(\mathcal{D}_i), \gamma_2(\mathcal{D}_i), \gamma_3(\mathcal{D}_i))$. Para cada $\mathcal{C}$, sendo $\Delta(\mathcal{C}) = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ define-se o multiconjunto $\Upsilon(\mathcal{C}) = \{\upsilon(\mathcal{D}_1), \ldots, \upsilon(\mathcal{D}_n)\}$. Atendendo ao sistema de reescrita $\mathsf{IP}_{\mathcal{M}}$ e às regras que envolvem as restrições $\top$ e $\perp$, Υ é estritamente decrescente pela aplicação de qualquer regra de $\mathsf{S}_{\mathcal{M}}^{-}$. 2

Proposição 6.10 *O sistema de reescrita $\mathsf{S}_{\mathcal{M}}^{-}$ é de complexidade polinomial.*

Dem. Seja $\mathcal{C} \in \mathcal{LU}$ e $\mathcal{M}$ um modelo. Seja A o número de conjunções de restrições atómicas de $\mathcal{C} \wedge \mathcal{M}$. Seja $P(n)$ um polinómio que majora o tempo de redução da conjunção de todas as restrições atómicas de $\mathcal{C}$ a forma primária, em função do número n de restrições atómicas e de variáveis de $\mathcal{C}$.

O sistema de reescrita $\mathsf{S}_{\mathcal{M}}^{-}$ reduz em $O(AP(n))$ todas as conjunções de restrições atómicas a forma primária ou $\perp$ (usando contexto) e simplifica em $O(A)$ a restrição obtida usando as regras $\mathsf{S1}$ a $\mathsf{S5}$. Para a restrição obtida o processo repete-se, e isto,

[3] É importante não confundir a notação de conjuntos normalmente usada para a conjunção de restrições, que aqui é usada para a disjunção de conjunções de restrições atómicas.

no máximo A vezes. Então, o sistema de reescrita tem complexidade, no pior caso, da ordem $O(A^2P(n))$. 2

Considerando o conjunto de fórmulas de $\mathcal{LU}$ satisfazíveis em $\mathcal{RT}$ e o facto de $\mathcal{RT}$ ser completo, podemos generalizar a noção de um sistema de reescrita completo para um conjunto de fórmulas (ver Definição 2.17) da seguinte forma:

Definição 6.11 *Um sistema de reescrita definido em $\mathcal{LU}$ diz-se* completo para a satisfazibilidade em $\mathcal{RT}$, *ou simplesmente* completo, *se toda a restrição $\mathcal{C}$ fechada para esse sistema de reescrita, e diferente de $\perp$, é satisfazível em $\mathcal{RT}$. Caso contrário, o sistema de reescrita diz-se* incompleto *(para a satisfazibilidade).*

Proposição 6.12 *O sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$ é incompleto.*

Dem. Considerem-se as restrições

$$\begin{array}{lll} \mathcal{C} & \equiv & (x = a \ \vee \ x = b) \ \wedge \ (x = c \ \vee \ x = d) \\ \mathcal{M} & \equiv & \top \end{array}$$

A restrição $\mathcal{C}$ é fechada para $\mathsf{S}^-_{\mathcal{M}}$ mas por redução a forma normal disjuntiva (ver Teorema 4.58) concluí-se que $\mathcal{C}$ não é satisfazível em $\mathcal{RT}$. 2

Se $\mathcal{C}$ é uma conjunção de restrições atómicas, $\mathcal{M}$ um modelo e $\mathcal{M} \wedge \mathcal{C}$ é satisfazível em $\mathcal{RT}$, fechar $\mathcal{C}$ para $\mathsf{S}_{\mathcal{M}}$ é equivalente a fechar $\mathcal{C}$ para o sistema de reescrita $\mathsf{IP}_{\mathcal{M}}$. Se $\mathcal{M} \wedge \mathcal{C}$ não é satisfazível o modelo final será $\perp$. Como consequência, para estas classes de restrições o sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$ é completo. Mais geralmente, podemos enunciar o seguinte corolário:

Corolário 6.13 *Seja $\mathcal{M}$ um modelo e $\mathcal{C} \in \mathcal{LU}$ uma restrição fechada para o sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$, então para toda a conjunção $\mathcal{C}'$ de restrições atómicas, diferente de $\perp$, que ocorre em $\mathcal{C}$ tem-se que $\mathcal{C}'$ é $\mathcal{M}$-satisfazível.*

Dem. Caso contrário, pelo menos uma das regras $\mathsf{S6}$ ou $\mathsf{S7}$ aplicar-se-ia, o que contradiz o facto de $\mathcal{C}$ ser fechada para $\mathsf{S}^-_{\mathcal{M}}$. 2

Para demonstrar que o sistema de reescrita $\mathsf{S}_{\mathcal{M}}$ é completo acrescentando ao sistema $\mathsf{S}^-_{\mathcal{M}}$ a regra $\mathsf{S8}$, que é um refinamento da regra da distributividade usada para reduzir uma restrição a forma normal disjuntiva (ver Teorema 4.58), é necessário estudar a independência de restrições em relação a um modelo.

Definição 6.14 *Sendo $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{LU}$ e $\mathcal{M}$ um modelo, $\mathcal{C}_1$ e $\mathcal{C}_2$ são $\mathcal{M}$-*independentes *se $\mathcal{C}_1 \wedge \mathcal{C}_2$ é $\mathcal{M}$-satisfazível se e só se $\mathcal{C}_1$ e $\mathcal{C}_2$ são $\mathcal{M}$-satisfazíveis.*

Lema 6.15 *Seja $\mathcal{M}$ um modelo e $\mathcal{C}_1,\ \mathcal{C}_2,\ \mathcal{C}_3 \in \mathcal{LU}$ restrições $\mathcal{M}$-satisfazíveis. Se $\mathcal{C}_1$ e $\mathcal{C}_3$ são $\mathcal{M}$-independentes então $(\mathcal{C}_1 \ \vee\ \mathcal{C}_2) \ \wedge\ \mathcal{C}_3$ é $\mathcal{M}$-satisfazível.*

Dem. Resulta directamente da distributividade e da $\mathcal{M}$-independência. 2

Pelo lema anterior, se $\mathcal{M}$ for um modelo e $\mathcal{C}_1,\ \mathcal{C}_3 \in \mathcal{LU}$ forem restrições $\mathcal{M}$-satisfazíveis e $\mathcal{C}_1$ uma disjunção, então $\mathcal{C}_1$ e $\mathcal{C}_3$ são $\mathcal{M}$-independentes se e só se algum elemento de $\mathcal{C}_1$ for $\mathcal{M}$-independente de $\mathcal{C}_3$. Este facto, pode ser utilizado conjuntamente com os critérios das Proposições 6.17, 6.18 e 6.19 para testar a $\mathcal{M}$-independência de restrições.

Iremos apresentar, algumas condições suficientes de $\mathcal{M}$-independência por análise das variáveis que ocorrem nas restrições.

Definição 6.16 *Sendo $\mathcal{M}$ uma forma normal e $\mathcal{C} \in \mathcal{LU}$ uma restrição, define-se $\mathsf{Var}_{\mathcal{M}}(\mathcal{C})$ como o menor conjunto que verifica as seguintes condições:*

(i) *se $x \in \mathsf{Vars}(\mathcal{C})$ então $x \in \mathsf{Var}_{\mathcal{M}}(\mathcal{C})$;*

(ii) *se $x \in \mathsf{Var}_{\mathcal{M}}(\mathcal{C})$ e $x =_{\mathcal{M}} z$ então $z \in \mathsf{Var}_{\mathcal{M}}(\mathcal{C})$;*

(iii) *se $x \in \mathsf{Var}_{\mathcal{M}}(\mathcal{C})$ e $x.f_i^n \doteq z \in \mathcal{M}$, para algum $f^n \in \Sigma$ e algum $1 \leq i \leq n$, então $z \in \mathsf{Var}_{\mathcal{M}}(\mathcal{C})$.*

*Se $x \in \mathsf{Var}_{\mathcal{M}}(\mathcal{C})$ então diz-se que x é $\mathcal{M}$-*ligada em *$\mathcal{C}$.*

Uma variável z é $\mathcal{M}$-ligada em $\mathcal{C}$ se z ocorre em $\mathcal{C}$ ou z depende de x em $\mathcal{M}$ e $x \in \mathsf{Vars}(\mathcal{C})$. Por exemplo, sendo

$$\begin{array}{lrcl} (\mathsf{c10}) & \mathcal{M} & \equiv & y.f^3 \ \wedge\ y.f_1^3 \doteq w \ \wedge\ \ \wedge w.g^2 \ \wedge\ w.g_1^2 \doteq z \\ & \mathcal{C}_1 & \equiv & x = w \\ & \mathcal{C}_2 & \equiv & y.f^3 \ \wedge\ y.f_1^3 \doteq u \end{array}$$

$\mathsf{Var}_{\mathcal{M}}(\mathcal{C}_1) = \{x, w, z\}$ e $\mathsf{Var}_{\mathcal{M}}(\mathcal{C}_2) = \{y, u, w, z\}$.

O valor duma variável que ocorra em $\mathcal{M}$ e não em $\mathcal{C}$, pode ser necessário para a satisfazibilidade de $\mathcal{C} \ \wedge\ \mathcal{M}$. No exemplo anterior, o valor da variável u de $\mathcal{C}_2$ tem de ser igual ao da variável w de $\mathcal{M}$.

E, assim, a $\mathcal{M}$-independência entre duas restrições $\mathcal{C}_1$ e $\mathcal{C}_2$ pode não ser garantida pelo facto da intersecção dos seus conjuntos de variáveis ser vazia.

No entanto, temos a seguinte proposição:

Proposição 6.17 *Sendo $\mathcal{M}$ um modelo e $\mathcal{M}_1,\ \mathcal{M}_2 \in \mathcal{LU}$ formas resolvidas normais fechadas para $\mathsf{S}^-_{\mathcal{M}}$, $\mathcal{M}_1$ e $\mathcal{M}_2$ são $\mathcal{M}$-independentes se* $\mathsf{Vars}(\mathcal{M}_1)\ \cap\ \mathsf{Vars}(\mathcal{M}_2) = \emptyset$.

Dem. Pela definição destas formas, e pelas hipóteses, $\mathcal{M}_1\ \wedge\ \mathcal{M}_2$ são formas resolvidas. Se $\mathcal{M}\ \wedge\ \mathcal{M}_1\ \wedge\ \mathcal{M}_2$ não fosse satisfazível então, pelo Corolário 4.5,

$$\mathcal{M}\ \wedge\ \mathcal{M}_1\ \wedge\ \mathcal{M}_2 \overset{\mathsf{R}}{\mapsto}{}^{\star} \perp$$

Vamos ver que tal não é possível. As regras do sistema de reescrita R que produzem $\perp$, não se podem aplicar directamente a $\mathcal{M}\ \wedge\ \mathcal{M}_1\ \wedge\ \mathcal{M}_2$. Para que se apliquem essas regras, é necessário aplicarem-se primeiro a regra $\mathsf{R4}$ ou a regra $\mathsf{R6}$. Começamos por ver que por aplicações apenas da regra $\mathsf{R4}$ (e eventualmente de outras diferentes da regra $\mathsf{R6}$), $\perp$ não pode ser produzido.

Se uma variável é resolvida em $\mathcal{M}$ então ela não pode ocorrer nem em $\mathcal{M}_1$ nem em $\mathcal{M}_2$, isto é,

$$\forall\, x,\, y \in \mathsf{Vars}(\mathcal{M}_1\ \cup\ \mathcal{M}_2)\ (\exists\, z.\ \ x,\, y \in [z]_{\mathcal{M}}\ \Rightarrow\ x \equiv y \equiv z)$$

Consequentemente, para cada classe de equivalência de $=_{\mathcal{M}}$ apenas uma variável pode ocorrer em $\mathcal{C}_1$ ou $\mathcal{C}_2$.

Então, a regra $\mathsf{R4}$ só de pode aplicar se

- $x = a \in \mathcal{M}_1$ (ou $\mathcal{M}_2$), $x \in \mathsf{Vars}(\mathcal{M})$, x é livre em $\mathcal{M}$ e $x \notin \mathsf{Vars}(\mathcal{M}_2)$ (ou $\mathsf{Vars}(\mathcal{M}_1)$)
- $x = z \in \mathcal{M}_1$ (ou $\mathcal{M}_2$), $x \in \mathsf{Vars}(\mathcal{M})$, x é ou funcional ou livre em $\mathcal{M}$ e $x,\, z \notin \mathsf{Vars}(\mathcal{M}_2)$ (ou $\mathsf{Vars}(\mathcal{M}_1)$)

Em todos os casos, apenas se efectuam em $\mathcal{M}$ permutações de valores nas mesmas classes de equivalência de $=_{\mathcal{M}_1}$ ou $=_{\mathcal{M}_2}$. Como $\mathcal{M} \wedge \mathcal{M}_1$ e $\mathcal{M} \wedge \mathcal{M}_2$ são satisfazíveis, $\perp$ não pode ser produzido.

Vamos considerar a primeira a aplicação da regra $\mathsf{R6}$. Para que esta regra se aplique uma das seguintes condições teria de se verificar:

- $x.f_i^n \doteq v \in \mathcal{M}_1$ e $x.f_i^n \doteq v' \in \mathcal{M}_2$, o que contradiz a hipótese;
- $x.f_i^n \doteq v \in \mathcal{M}$, $y.f_i^n \doteq v' \in \mathcal{M}$, $x \not\equiv y$, $x \in [z]_{\mathcal{M}_1}$ e $y \in [z]_{\mathcal{M}_2}$, o que é absurdo, porque ter-se-ia $z \in \mathsf{Vars}(\mathcal{M}_1)\ \cap\ \mathsf{Vars}(\mathcal{M}_2)$ (note-se que z não podia ocorrer em $\mathcal{M}$);

- $x.f_i^n \doteq v \in \mathcal{M}$, $y.f_i^n \doteq v' \in \mathcal{M}$, $x \not\equiv y$ e $x \in [y]_{\mathcal{M}_1}$; neste caso, $y \notin \mathsf{Vars}(\mathcal{M}_2)$, $v \in [v']_{\mathcal{M}_1}$ e, portanto, se v ou v' forem variáveis, também, não ocorrem em $\mathcal{M}_2$. Então, a aplicação da regra R6 apenas elimina uma restrição de $\mathcal{M}$ que é redundante em $\mathcal{M} \wedge \mathcal{M}_1$ e não acrescenta nenhuma restrição. Portanto não pode causar a produção de $\perp$ (no passo seguinte).
- análoga à anterior, para $\mathcal{M}_2$;
- $x.f_i^n \doteq v_2 \in \mathcal{M}_1$, $y.f_i^n \doteq v_1 \in \mathcal{M}$, $x \not\equiv y$, e $x \in [y]_{\mathcal{M}_2}$, o que é absurdo, porque ter-se-ia $x \in \mathsf{Vars}(\mathcal{M}_1) \cap \mathsf{Vars}(\mathcal{M}_2)$.
- análoga à anterior, para $\mathcal{M}_2$;

Assim, sucessivas aplicações das regras R4 e R6 não podem conduzir à produção de $\perp$, e só podem ser em número finito. Concluímos que $\mathcal{M} \wedge \mathcal{M}_1 \wedge \mathcal{M}_2$ é satisfazível. 2

Na proposição anterior, é essencial que as restrições $\mathcal{M}_1$ e $\mathcal{M}_2$ estejam fechadas para $\mathsf{S}_{\mathcal{M}}^-$. Consideremos o seguinte exemplo:

$$(\mathsf{c}_{11}) \qquad \begin{array}{lcl} \mathcal{M} & \equiv & x.f^2 \wedge x.f_1^2 \doteq w \wedge y.f^2 \wedge y.f_1^2 = z \wedge u = s \\ \mathcal{M}_1 & \equiv & x = y \wedge s = a \\ \mathcal{M}_2 & \equiv & w = a \wedge z = b \wedge u = c \end{array}$$

Claramente $\mathcal{M} \wedge \mathcal{M}_1$ e $\mathcal{M} \wedge \mathcal{M}_2$ são satisfazíveis em $\mathcal{RT}$ e $\mathsf{Vars}(\mathcal{M}_1) \cap \mathsf{Vars}(\mathcal{M}_2) = \emptyset$, mas $\mathcal{M} \wedge \mathcal{M}_1 \wedge \mathcal{M}_2$ não o é. No entanto, se fecharmos $\mathcal{M}_1$ e $\mathcal{M}_2$ por $\mathsf{S}_{\mathcal{M}}^-$, obtemos, respectivamente

$$(\mathsf{c}_{12}) \qquad \begin{array}{lcl} \mathcal{M}_1 & \equiv & x = y \wedge z = w \wedge s = a \\ \mathcal{M}_2 & \equiv & w = a \wedge z = b \wedge s = c \end{array}$$

e $\mathsf{Vars}(\mathcal{M}_1) \cap \mathsf{Vars}(\mathcal{M}_2) = \{s, w, z\} \neq \emptyset$.

Se uma restrição fechada para $\mathsf{S}_{\mathcal{M}}^-$ contiver negações a condição da Proposição 6.17 já não é suficiente para a $\mathcal{M}$-independência.

Sejam

$$(\mathsf{c}_{13}) \qquad \begin{array}{lcl} \mathcal{M} & \equiv & z.f^1 \wedge z.f_1^1 \doteq w \\ \mathcal{C}_1 & \equiv & x.f^1 \wedge x.f_1^1 \doteq z \wedge y.f^1 \wedge y.f_1^1 \doteq u \wedge u.f^1 \wedge u.f_1^1 \doteq a \wedge x \neq y \\ \mathcal{C}_2 & \equiv & w = a \end{array}$$

temos que $\mathcal{C}_1$ e $\mathcal{C}_2$ são fechadas para $\mathsf{S}^-_{\mathcal{M}}$, $\mathsf{Vars}(\mathcal{C}_1) \cap \mathsf{Vars}(\mathcal{C}_2) = \emptyset$ e $\mathcal{M} \wedge \mathcal{C}_1 \wedge \mathcal{C}_2$ não é satisfazível porque

$$\mathcal{M} \wedge \mathcal{C}_p^1 \wedge \mathcal{C}_p^2 \models_{\mathcal{RT}} x = y \quad \text{e} \quad x \neq y \in \mathcal{C}_n^1 \wedge \mathcal{C}_n^2$$

Mas, $\mathsf{Var}_{\mathcal{M}}(\mathcal{C}_1) = \{x, y, z, u, w\}$ e $\mathsf{Var}_{\mathcal{M}}(\mathcal{C}_2) = \{w\}$, isto é, a intersecção dos dois conjuntos não é o conjunto vazio. Neste caso, as variáveis de $\mathsf{Var}_{\mathcal{M}}(\mathcal{C}_1)$ correspondem às que ocorreriam se expandíssemos em $\mathcal{M} \wedge \mathcal{C}_1$ a negação $x \neq y$, usando a regra $\mathsf{NP1}$, para redução a forma primária normal.

Para restrições em geral, temos a seguinte condição suficiente de $\mathcal{M}$-independência:

Proposição 6.18 *Sendo $\mathcal{M}$ um modelo e $\mathcal{C}_1$, $\mathcal{C}_2 \in \mathcal{LU}$ restrições fechadas para $\mathsf{S}^-_{\mathcal{M}}$, $\mathcal{C}_1$ e $\mathcal{C}_2$ são $\mathcal{M}$-independentes se $\mathsf{Var}_{\mathcal{M}}(\mathcal{C}_1) \cap \mathsf{Var}_{\mathcal{M}}(\mathcal{C}_2) = \emptyset$.*

Dem. Se $\mathcal{C}_1$ e $\mathcal{C}_2$ são formas resolvidas normais é óbvio, pela Proposição 6.17 e porque $\mathsf{Vars}(\mathcal{C}_i) \subseteq \mathsf{Var}_{\mathcal{M}}(\mathcal{C}_i)$, para $1 \leq i \leq 2$.

Suponhamos que $\mathcal{C}_1 \equiv \mathcal{C}_p^1 \wedge \mathcal{C}_n^1$ e $\mathcal{C}_2 \equiv \mathcal{C}_p^2 \wedge \mathcal{C}_n^2$ são formas primárias. Neste caso, $\mathcal{C}_1$ e $\mathcal{C}_2$ estão fechadas para o sistema de reescrita $\mathsf{IP}_{\mathcal{M}}$. Pela Proposição 6.17, $\mathcal{M} \wedge \mathcal{C}_p^1 \wedge \mathcal{C}_p^2$ é satisfazível mas pode não ser uma forma normal. Em particular, poderíamos ter

$$\mathcal{M} \wedge \mathcal{C}_p^1 \wedge \mathcal{C}_p^2 \models_{\mathcal{RT}} x = y \quad \text{e} \quad x \neq y \in \mathcal{C}_n^1 \wedge \mathcal{C}_n^2$$

Suponhamos que $x \neq y \in \mathcal{C}_n^1$. Então, x e y são distinguíveis em $\mathcal{M} \wedge \mathcal{C}_p^1$. Pelo Corolário 4.16, se $x = y$ fosse vinculado por $\mathcal{M} \wedge \mathcal{C}_p^1 \wedge \mathcal{C}_p^2$, então os caminhos de x e de y em $\mathcal{M} \wedge \mathcal{C}_p^1 \wedge \mathcal{C}_p^2$ coincidem.

Tal só seria possível se existissem variáveis $z \in \mathsf{Vars}(\mathcal{C}_1)$ e $w \in \mathsf{Vars}(\mathcal{C}_2)$ tal que $z.f_i^n \doteq w \in \mathcal{C}_1$ ou $z.f_i^n \doteq w \in \mathcal{M}$. Mas por hipótese, tal não é possível. Concluímos que $\mathcal{M} \wedge \mathcal{C}_1 \wedge \mathcal{C}_2$ é satisfazível.

Se $\mathcal{C}_1$ e $\mathcal{C}_2$ não forem formas primárias, pelo Corolário 6.13 sabemos que as conjunções de restrições atómicas que ocorrem em $\mathcal{C}_1$ e $\mathcal{C}_2$ são $\mathcal{M}$-satisfazíveis e a(s) regra(s) de redução a forma normal disjuntiva (ver Teorema 4.58) preservam a $\mathcal{M}$-satisfazibilidade. Analogamente, as regras de redução a forma primária preservam a $\mathcal{M}$-satisfazibilidade e não alteram as variáveis $\mathcal{M}$-ligadas. Para cada $1 \leq i \leq 2$, $\mathcal{C}_i$ é equivalente a uma disjunção

$$\mathcal{P}_1^i \vee \ldots \vee \mathcal{P}_{n_i}^i$$

onde para $1 \leq l \leq n_i$, $\mathcal{P}_l^i$ ou é $\perp$ ou uma forma primária $\mathcal{M}$-satisfazível. Se $\mathcal{C}_1$ e $\mathcal{C}_2$ forem satisfazíveis então existe um $1 \leq k \leq n_1$ e um $1 \leq j \leq n_2$ tal que $\mathcal{M} \wedge \mathcal{P}_k^1$

e $\mathcal{M} \wedge \mathcal{P}_j^2$ são satisfazíveis o que implica, por $\mathcal{P}_k^1$ e $\mathcal{P}_j^2$ serem formas primárias fechadas para $\mathsf{S^e}_{\mathcal{P}}^-$ e sem variáveis em comum, que $\mathcal{M} \wedge \mathcal{P}_k^1 \wedge \mathcal{P}_j^2$ é satisfazível, logo, $\mathcal{M} \wedge \mathcal{C}_1 \wedge \mathcal{C}_2$ também o é. 2

Para formas resolvidas, vamos ainda considerar o seguinte critério de $\mathcal{M}$-independência:

Proposição 6.19 *Sendo $\mathcal{M}$ um modelo e $\mathcal{M}_1$, $\mathcal{M}_2$ formas resolvidas normais fechadas para $\mathsf{S}_{\mathcal{M}}^-$, $\mathcal{M}_1$ e $\mathcal{M}_2$ são $\mathcal{M}$-independentes se para todo $x \in \mathsf{Vars}(\mathcal{M}_1) \cap \mathsf{Vars}(\mathcal{M}_2)$, $x \notin \mathsf{Vars}(\mathcal{M})$, as classes $[x]_{\mathcal{M}_1}$ e $[x]_{\mathcal{M}_2}$ são unitárias e uma das seguintes condições se verifica:*

(i) *x é livre em $\mathcal{M}_1$ ou em $\mathcal{M}_2$.*

(ii) *x é funcional em ambas, com o mesmo functor f^n, e para todo $1 \leq i \leq n$, as classes $[x]_{\mathcal{M}_1}$ e $[x]_{\mathcal{M}_2}$ não estão simultaneamente definidas para i.*

Dem. Considerando a demonstração da Proposição 6.17, se as condições (i) ou (ii) se verificarem, $\mathcal{M} \wedge \mathcal{M}_1 \wedge \mathcal{M}_2$ também é satisfazível. A condição das classes serem unitárias garante que não existe um elemento dessas classes que ocorra em $\mathcal{M}$ ou na outra restrição em classes distintas. 2

Na proposição anterior, se as classes não forem unitárias a $\mathcal{M}$-independência não é garantida.

Considerem-se os seguintes exemplos

$$(\mathsf{C}_{14}) \qquad \begin{array}{lcl} \mathcal{M} & \equiv & x.f^3 \wedge x.f_1^3 \doteq a \wedge y.f^3 \wedge y.f_1^3 \doteq b \\ \mathcal{M}_1 & \equiv & z.f^3 \wedge z.f_2^3 \doteq a \wedge x = z \\ \mathcal{M}_2 & \equiv & z.f^3 \wedge z.f_3^3 \doteq w \wedge y = z \end{array}$$

$$(\mathsf{C}_{15}) \qquad \begin{array}{lcl} \mathcal{M} & \equiv & y.f^3 \wedge y.f_1^3 \doteq a \\ \mathcal{M}_1 & \equiv & x.f^3 \wedge x.f_2^3 \doteq a \wedge x = z \\ \mathcal{M}_2 & \equiv & x.f^3 \wedge x.f_3^3 \doteq b \wedge z.f^3 \wedge z.f_3^3 \doteq a \end{array}$$

No primeiro exemplo, a igualdade entre duas variáveis incompatíveis em $\mathcal{M}$ seria vinculada pela conjunção $\mathcal{M} \wedge \mathcal{M}_1 \wedge \mathcal{M}_2$, e no segundo exemplo, o mesmo sucederia com duas variáveis incompatíveis em $\mathcal{M}_2$.

Se $\mathcal{C}_1$ e $\mathcal{C}_2$ forem formas primárias as condições da Proposição 6.19 já não garantem a $\mathcal{M}$-independência, a menos que as variáveis envolvidas não ocorram em negações nem em $\mathcal{C}_1$ nem em $\mathcal{C}_2$. Em particular, a noção de variável *livre* tem ser modificada de modo a impor que a variável não ocorra em negações. Consideremos os seguintes exemplos:

(C16) $\mathcal{C}_1 \equiv x.f^2 \wedge x.f_1^2 \doteq a \wedge y.f^2 \wedge y.f_1^2 \doteq a \wedge y.f_2^2 \doteq b \wedge x \neq y$
$\mathcal{C}_2 \equiv x.f^2 \wedge x.f_2^2 \doteq b$

(C17) $\mathcal{C}_1 \equiv x.f^2 \wedge x.f_2^2 \doteq y \wedge y \neq b$
$\mathcal{C}_2 \equiv y = b$

Nestes exemplos, $\mathcal{C}_1$ e $\mathcal{C}_2$ não são $\top$-independentes, embora as respectivas formas resolvidas normais verifiquem as condições da Proposição 6.19.

Na Proposição 6.19, a condição (ii) pode ser generalizada a caminhos. No entanto, na prática, essa procura pode tornar este critério pouco eficiente.

As condições das Proposição 6.17, 6.18 e 6.19 podem ser verificadas em tempo polinomial, quadrático no máximo, antes da aplicação da regra S8 do sistema de reescrita $\mathsf{S}_{\mathcal{M}}$.

A investigação de outros critérios que garantam a independência de restrições em relação a um modelo, embora com variáveis comuns, será do maior interesse, pois evitariam a aplicação daquela regra.

Teorema 6.20 *O sistema de reescrita $\mathsf{S}_{\mathcal{M}}$ é correcto (em $\mathcal{RT}$) e termina.*

Dem. O sistema $\mathsf{S}_{\mathcal{M}}^-$ é correcto e a regra S8, sem controlo, é correcta em qualquer álgebra de $\mathcal{L}$. O controlo que é introduzido não altera a correcção da regra, apenas limita as condições de aplicabilidade. Analisemos a terminação. Dado que a regra S8 só se aplica se nenhuma das anteriores se aplicar, e as restantes constituem um sistema de reescrita que termina ($\mathsf{S}_{\mathcal{M}}^-$), basta ver se poderá existir uma derivação infinita que envolva esta regra. Omitindo o teste de $\mathcal{M}$-independência esta regra é a regra da distributividade, usada na demonstração do Teorema 4.58. Seja γ a seguinte função de $\mathcal{LU}$ em $\mathbb{N}$:

$\gamma(\mathcal{C}) = 2$ se $\mathcal{C}$ é uma restrição atómica negativa ou uma conjunção de restrições atómicas positivas

$\gamma(\mathcal{C}_1 \vee \mathcal{C}_2) = \gamma(\mathcal{C}_1) + \gamma(\mathcal{C}_2) + 1$

$\gamma(\mathcal{C}_1 \wedge \mathcal{C}_2) = \gamma(\mathcal{C}_1) \times \gamma(\mathcal{C}_2)$

O valor de γ não aumenta por aplicação de nenhuma regra de $\mathsf{S}_{\mathcal{M}}^-$ e decresce estritamente por aplicação da regra S8. Considerando a análise feita na demonstração da Proposição 6.9, sendo Υ a medida associada ao sistema de reescrita $\mathsf{S}_{\mathcal{M}}^-$, temos que o par (γ, Υ), ordenado lexicograficamente, é estritamente decrescente por aplicação das regras de $\mathsf{S}_{\mathcal{M}}$. 2

Teorema 6.21 [4] *O sistema de reescrita $\mathsf{S}_{\mathcal{M}}$ é completo, isto é, sendo $\mathcal{M}$ um modelo e $\mathcal{C} \in \mathcal{LU}$ irredutível para $\mathsf{S}_{\mathcal{M}}$, então $\mathcal{M} \wedge \mathcal{C}$ é satisfazível em $\mathcal{RT}$ se e só se $\mathcal{C} \not\equiv \bot$.*

Dem. Se $\mathcal{C} \equiv \bot$ é óbvio que $\mathcal{M} \wedge \mathcal{C}$ não é satisfazível. Suponhamos, então, que $\mathcal{C} \not\equiv \bot$. Provamos por indução na estrutura de $\mathcal{C}$, que $\mathcal{C}$ é $\mathcal{M}$-satisfazível. Supondo a forma geral (6.1), consideramos $\mathcal{C}$ como conjunções e disjunções de restrições atómicas negativas e de conjunções de restrições atómicas positivas.

Base. Se $\mathcal{C}$ é uma conjunção de restrições atómicas o resultado é válido pelo Corolário 6.13.

Indução. Temos dois casos a considerar:

Caso $\mathcal{C} \equiv \mathcal{C}_1 \wedge \mathcal{C}_2$***.*** Se $\mathcal{C}_1$ é uma conjunção de restrições atómicas positivas e $\mathcal{C}_2$ não o é , pela hipótese de indução e pela regra $\mathsf{S7}$, $\mathcal{C}$ é $\mathcal{M}$-satisfazível. Se $\mathcal{C}_1$ (ou $\mathcal{C}_2$) é da forma $\mathcal{C}_0 \vee \mathcal{C}_3$, pela regra $\mathsf{S8}$ ou $\mathcal{C}_0$ ou $\mathcal{C}_3$ é $\mathcal{M}$-independente de $\mathcal{C}_2$. Logo pela hipótese de indução e pelo Lema 6.15, $\mathcal{C}$ é $\mathcal{M}$-satisfazível. Se $\mathcal{C}_1$ e $\mathcal{C}_2$ são restrições atómicas negativas (ou suas conjunções), usamos o caso base. E, neste caso, mais nada há a considerar.

Caso $\mathcal{C} \equiv \mathcal{C}_1 \vee \mathcal{C}_2$***.*** Por hipótese de indução, temos que $\mathcal{M} \wedge \mathcal{C}_1$ e $\mathcal{M} \wedge \mathcal{C}_2$ são satisfazíveis, caso contrário pelo menos uma delas era $\bot$ (regra $\mathsf{S5}$), e portanto $\mathcal{M} \wedge \mathcal{C}$ também o é (basta considerar uma das soluções).

2

Corolário 6.22 *Dado $\mathcal{C}_0 \in \mathcal{LU}$, $\langle \top, \mathcal{C}_0 \rangle \Rightarrow^{\star} \langle \mathcal{M}, \mathcal{C} \rangle$ tal que:*

(i) $\mathcal{M} \equiv \bot$ *se e só se* $\mathcal{RT} \models \mathcal{C}_0 \Leftrightarrow \bot$

(ii) $\mathcal{RT} \models \mathcal{C}_0 \Leftrightarrow \mathcal{M} \wedge \mathcal{C}$ e $\mathcal{C} \not\equiv \bot$, *caso contrário.*

As proposições seguintes caracterizam as soluções de $\mathcal{C}_0$ em termos das soluções de $\mathcal{M}$ e $\mathcal{C}$. Em particular, mostram em que medida as soluções de $\mathcal{C}$ são "independentes" das de $\mathcal{M}$.

Proposição 6.23 *Seja $\mathcal{C}_0 \in \mathcal{LU}$ e suponhamos que $\langle \top, \mathcal{C}_0 \rangle \Rightarrow^{\star} \langle \mathcal{M}, \mathcal{C} \rangle$ com $\mathcal{C} \not\equiv \bot$. Então existem formas primárias $\mathcal{P}'_i$ e $\mathcal{P}_i$, para $1 \leq i \leq n$, tal que*

$$\begin{aligned} \mathcal{C}_0 &\approx_{\mathcal{RT}} \mathcal{P}'_1 \vee \ldots \vee \mathcal{P}'_n \\ \mathcal{C} &\approx_{\mathcal{RT}} \mathcal{P}_1 \vee \ldots \vee \mathcal{P}_n \end{aligned}$$

e $\mathcal{P}'_i$ coincide com $\mathcal{M} \wedge \mathcal{P}_i$ a menos da presença de restrições atómicas equivalentes.

[4] *Um teorema análogo para linguagens de estruturas de atributos foi apresentado em [Bro92].*

Dem. Pelo Teorema 4.58, a restrição $\mathcal{C}$ é equivalente em $\mathcal{RT}$ a uma disjunção de formas primárias, $\mathcal{P}_1 \vee \ldots \vee \mathcal{P}_n$. Pelo Corolário 6.22(ii),

$$\mathcal{C}_0 \approx_{\mathcal{RT}} (\mathcal{M} \wedge \mathcal{P}_1) \vee \ldots \vee (\mathcal{M} \wedge \mathcal{P}_n)$$

Pelo Corolário 6.13, todas as conjunções de restrições atómicas que ocorrem em $\mathcal{C}$ são $\mathcal{M}$-satisfazíveis e a(s) regra(s) de redução a forma normal disjuntiva (ver Teorema 4.58) preservam a $\mathcal{M}$-satisfazibilidade. Analogamente, as regras de redução a forma primária, a menos que produzam $\perp$, preservam a $\mathcal{M}$-satisfazibilidade. Portanto, se cada $\mathcal{P}_i$, para $1 \leq i \leq n$, for satisfazível então é $\mathcal{M}$-satisfazível. Por outro lado, cada $\mathcal{P}_i$, para $1 \leq i \leq n$, é fechada para $\mathsf{S}_{\mathcal{M}}$, caso contrário, $\mathcal{C}$ também não o seria. Este facto garante que $\mathcal{M} \wedge \mathcal{P}_i$ está em forma primária, excepto, a presença de restrições atómicas equivalentes, como na Proposição 6.2. 2

Proposição 6.24 *Dado $\mathcal{C}_0 \in \mathcal{LU}$ se $\langle \top, \mathcal{C}_0 \rangle \Rightarrow^\star \langle \mathcal{M}, \mathcal{C} \rangle$ e $\mathcal{C} \not\equiv \perp$, qualquer modelo minimal de $\mathcal{C}_0$ pode ser obtido pela conjunção de $\mathcal{M}$ com um modelo minimal de $\mathcal{C}$.*

Dem. Seja $\mathcal{P}_0$ um modelo minimal de $\mathcal{C}_0$. Temos que

$$\begin{array}{lll} \mathcal{P}_0 & \models_{\mathcal{RT}} & \mathcal{C}_0 \\ \mathcal{P}_0 & \models_{\mathcal{RT}} & \mathcal{M} \wedge \mathcal{C} \\ \mathcal{P}_0 & \approx_{\mathcal{RT}} & \mathcal{M} \wedge \mathcal{P}_0 \end{array}$$

Consideremos $\mathcal{P}_0 \overset{\mathsf{IP}}{\mapsto}{}^\star_{\mathcal{M}} \mathcal{P}_c$, com $\mathcal{P}_0 \doteqdot \mathcal{M} \wedge \mathcal{P}_c$. Vamos ver que $\mathcal{P}_c$ é um modelo minimal de $\mathcal{C}$.

- $\mathcal{P}_c$ é modelo de $\mathcal{C}$. Suponhamos que não, então existia $\rho \in \mathsf{Sol}(\mathcal{P}_c, \mathcal{RT})$ e $\rho \notin \mathsf{Sol}(\mathcal{C}, \mathcal{RT})$. Se $\rho \in \mathsf{Sol}(\mathcal{M}, \mathcal{RT})$ então $\rho \in \mathcal{P}_0$, o que é um absurdo. Se $\rho \notin \mathsf{Sol}(\mathcal{M}, \mathcal{RT})$ então existe uma restrição c_i que é vinculada por $\mathcal{C}$ e que não é determinada em $\mathcal{P}_c$. Se c_i for determinada em $\mathcal{M}$, então $\mathcal{M}$ vincula $\neg c_i$. Senão, existe uma atribuição de $\mathsf{Sol}(\mathcal{M} \wedge \mathcal{P}_c, \mathcal{RT})$ que não o é de $\mathsf{Sol}(\mathcal{M} \wedge \mathcal{C}, \mathcal{RT})$, nomeadamente, uma atribuição ρ' tal que $\mathcal{RT} \models_{\rho'} \neg c_i$. Em ambos os casos temos um absurdo.
- Se $\mathcal{P}_c$ não fosse minimal existia um modelo $\mathcal{P}'$ de $\mathcal{C}$ que era vinculado por $\mathcal{P}_c$ e $\mathcal{M} \wedge \mathcal{P}'$ era um modelo de $\mathcal{C}_0$ que vincularia $\mathcal{P}_0$.

2

O recíproco da proposição anterior pode não ser verdade, isto é, nas condições da proposição a conjunção de um modelo minimal de $\mathcal{C}$ com $\mathcal{M}$ pode não ser um modelo de $\mathcal{C}_0$. Consideremos o seguinte exemplo:

$$(\mathsf{c}_{18}) \qquad y.f^n \ \wedge \ (\neg x.f^n \ \vee \ x \neq y)$$

Temos que

$$\langle \top, \mathsf{c}_{18} \rangle \Rightarrow^\star \langle y.f^n, \neg x.f^n \ \vee \ x \neq y \rangle$$

e $\neg y.f^n$ é modelo minimal de $\neg x.f^n \ \vee \ x \neq y$. Mas $y.f^n \ \wedge \ \neg y.f^n$ não é satisfazível.

No entanto, se $\langle \top, \mathcal{C}_0 \rangle \Rightarrow^\star \langle \mathcal{M}, \mathcal{C} \rangle$ e não existirem negações não redundantes em $\mathcal{C}$ o recíproco também é verdadeiro.

Proposição 6.25 *Seja $\mathcal{C}_0 \in \mathcal{LU}$ e $\langle \top, \mathcal{C}_0 \rangle \Rightarrow^\star \langle \mathcal{M}, \mathcal{C} \rangle$. Se $\mathcal{C} \not\equiv \bot$ e se $\mathcal{C}$ não contiver negações (não redundantes), qualquer modelo minimal $\mathcal{M}_c$ de $\mathcal{C}$ é $\mathcal{M}$-satisfazível e a conjunção de $\mathcal{M}_c$ com $\mathcal{M}$ é um modelo minimal de $\mathcal{C}_0$.*

Dem. Pelo Corolário 5.3, seja

$$\mathcal{C} \approx \mathcal{M}_1 \ \vee \ \ldots \ \vee \ \mathcal{M}_n$$

a disjunção de todos os modelos minimais de $\mathcal{C}$. Então

$$\mathcal{C}_0 \approx_{\mathcal{RT}} (\mathcal{M} \ \wedge \ \mathcal{M}_1) \ \vee \ \ldots \ \vee \ (\mathcal{M} \ \wedge \ \mathcal{M}_n) \qquad (6.2)$$

e para $1 \leq i \leq n$, basta ver que $\mathcal{M} \ \wedge \ \mathcal{M}_i$ são modelos minimais de $\mathcal{C}_0$.

Cada $\mathcal{M} \ \wedge \ \mathcal{M}_i$, para $1 \leq i \leq n$, é satisfazível e em forma resolvida, excepto repetição de restrições atómicas, por um argumento idêntico ao usado na demonstração da Proposição 6.23. Que são modelos de $\mathcal{C}_0$ é evidente por (6.2). Suponhamos que existia $\mathcal{M}_0$ modelo minimal de $\mathcal{C}_0$ que era distinto de todos os $\mathcal{M} \wedge \mathcal{M}_i$, para $1 \leq i \leq n$. Como $\mathcal{M}_0 \models_{\mathcal{RT}} \mathcal{C}_0$, então, pela Proposição 4.27, existe i tal que $\mathcal{M}_0 \models_{\mathcal{RT}} \mathcal{M} \ \wedge \ \mathcal{M}_i$. Mas dado que $\mathcal{M} \ \wedge \ \mathcal{M}_i \models_{\mathcal{RT}} \mathcal{C}_0$, concluíamos que $\mathcal{M}_0$ não era minimal. Então, como já se tinha visto, na Proposição 6.24, qualquer modelo minimal de $\mathcal{C}_0$ é um dos $\mathcal{M} \ \wedge \ \mathcal{M}_i$.

Suponhamos que para algum $1 \leq j \leq n$, $\mathcal{M} \ \wedge \ \mathcal{M}_j$ não era modelo minimal de $\mathcal{C}_0$. Então, existia um $1 \leq i \leq n$, tal que $\mathcal{M} \ \wedge \ \mathcal{M}_j \models_{\mathcal{RT}} \mathcal{M} \ \wedge \ \mathcal{M}_i$ e necessariamente $\mathcal{M}_j \models_{\mathcal{RT}} \mathcal{M}_i$, porque, pelo Corolário 6.3(ii)-(iii), $\mathcal{M}$ e $\mathcal{M}_k$, para $1 \leq k \leq n$ não vinculam as mesmas restrições atómicas positivas. Então, mesmo que $\rho \notin \mathsf{Sol}(\mathcal{M}, \mathcal{RT})$ se $\rho \in \mathsf{Sol}(\mathcal{M}_i, \mathcal{RT})$ então $\rho \in \mathsf{Sol}(\mathcal{M}_j, \mathcal{RT})$. $\Box$

6.3 Classes de Restrições Completas para $\mathsf{S}^-_{\mathcal{M}}$

Dada a importância do sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$, apesar de incompleto, é interessante analisar classes de restrições para as quais esse sistema é completo.

Iremos definir a classe $\mathcal{LU}^w \subset \mathcal{LU}$ que possui as seguintes propriedades:

- a classe é interna para o sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$, isto é, se $c \in \mathcal{LU}^w$ e $c \overset{\mathsf{S}}{\longmapsto}_{\mathcal{M}} c'$ então $c' \in \mathcal{LU}^w$.
- se $c \in \mathcal{LU}^w$ é uma restrição irredutível para $\mathsf{S}^-_{\mathcal{M}}$ então c é satisfazível se e só se é diferente de $\perp$ (completude de $\mathsf{S}^-_{\mathcal{M}}$ para $\mathcal{LU}^w$).
- para qualquer restrição $c \in \mathcal{LU}$ existe uma restrição $c' \in \mathcal{LU}^w$ tal que $c \approx_{\mathcal{RT}} c'$.
- é fechada para disjunções mas não o é para conjunções.

Esta classe foi apresentada em [DMV94], sem serem provadas as propriedades acima referidas.

A classe $\mathcal{LU}^w$ define-se como o subconjunto de $\mathcal{LU}$ que não contém restrições que envolvem conjunções de disjunções e de negações. Formalmente, sendo $\mathcal{M}$ uma conjunção de restrições atómicas positivas (forma normal ou $\perp$) e $\mathcal{P}$ uma conjunção de restrições atómicas (forma primária ou $\perp$), as restrições de $\mathcal{W} \in \mathcal{LU}^w$ podem ser descritas pela seguinte gramática em BNF:

$$\begin{array}{rcl} \mathcal{W} & ::= & \mathcal{M} \wedge (\mathcal{D}) \\ & | & \mathcal{P} \\ \mathcal{D} & ::= & \mathcal{W} \vee \mathcal{W} \\ & | & \mathcal{W} \vee \mathcal{D} \end{array}$$

Lema 6.26 *Qualquer restrição de $\mathcal{LU}$ pode ser transformada numa restrição de $\mathcal{LU}^w$ equivalente em $\mathcal{RT}$, pela aplicação da seguinte regra de reescrita, a menos da relação $\equiv_{\mathcal{B}}$:*

W1. $c \wedge (c_1 \vee c_2) \mapsto (c \wedge c_1) \vee (c \wedge c_2)$ *se c é uma restrição atómica negativa ou uma disjunção.*

Como é evidente, a transformação apresentada no lema anterior corresponde, no pior caso, à transformação em forma normal disjuntiva. Contudo, se a restrição estiver inicialmente nessa forma, ela é preservada pelo sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$:

Lema 6.27 *A classe de restrições $\mathcal{LU}^w$ é interna para o sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$.*

Dem. Seja $\mathcal{M}$ um modelo e $\mathcal{C} \in \mathcal{LU}^w$. Basta verificar que nenhuma das regras de $\mathsf{S}^-_{\mathcal{M}}$ (e $\mathsf{I}_{\mathcal{M}}$) produz uma conjunção de disjunções ou negações. Na realidade, todas as regras de $\mathsf{S}^-_{\mathcal{M}}$ fazem diminuir ou não alteram o número de conectivas em $\mathcal{C}$ (não considerando as as conectivas das conjunções de restrições atómicas). 2

Proposição 6.28 *O sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$ é completo para $\mathcal{LU}^w$.*

Dem. A demonstração é análoga à do Teorema 6.21. Seja $\mathcal{M}$ um modelo e $\mathcal{C} \in \mathcal{LU}^w$ uma restrição fechada para $\mathsf{S}^-_{\mathcal{M}}$. Se $\mathcal{C} \equiv \perp$ então $\mathcal{M} \wedge \mathcal{C}$ não é satisfazível. Suponhamos, então, que $\mathcal{C} \not\equiv \perp$. Provamos por indução na estrutura de $\mathcal{C}$, que $\mathcal{C}$ é $\mathcal{M}$-satisfazível.

Base. Se $\mathcal{C}$ é uma conjunção de restrições atómicas o resultado é válido pelo Corolário 6.13.

Indução. Temos dois casos a considerar:

Caso $\mathcal{C} \equiv \mathcal{C}_1 \wedge \mathcal{C}_2$**.** Então $\mathcal{C}_1$ é uma conjunção de restrições atómicas positivas e $\mathcal{C}_2$ uma disjunção. Pela hipótese de indução e pela regra $\mathsf{S7}$, a restrição $\mathcal{C}$ é $\mathcal{M}$-satisfazível.

Caso $\mathcal{C} \equiv \mathcal{C}_1 \vee \mathcal{C}_2$**.** Por hipótese de indução temos que $\mathcal{M} \wedge \mathcal{C}_1$ e $\mathcal{M} \wedge \mathcal{C}_2$ são satisfazíveis, caso contrário, pelo menos uma delas era $\perp$ (regra $\mathsf{S5}$), e portanto, $\mathcal{M} \wedge \mathcal{C}$ também o seria (basta considerar uma das soluções).

2

Considere-se a seguinte restrição em $\mathcal{LU}^w$

$$(\mathsf{c}_{19}) \qquad \begin{array}{l} x.f_2^2 \doteq a \wedge y = c \wedge ((x.f_2^2 \doteq z \wedge w = a \wedge z \neq c) \\ \quad \vee (z.f_2^2 \doteq w \wedge (w.f^2 \vee x.f_2^2 \doteq b)) \vee (y = w \wedge w = z)) \end{array}$$

Aplicando o sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$ obtemos a restrição

$$(\mathsf{c}_{20}) \qquad \begin{array}{l} x.f^2 \wedge x.f_2^2 \doteq a \wedge y = c \wedge ((x.f_2^2 \doteq a \wedge w = a \wedge z = a) \\ \quad \vee (z.f^2 \wedge z.f_2^2 \doteq w \wedge w.f^2) \vee (w = c \wedge z = c)) \end{array}$$

que é satisfazível.

No entanto, tal não é verdade para a restrição

$$(\mathsf{c}_{21}) \qquad x \neq a \wedge x \neq b \wedge (x = a \vee x = b)$$

embora seja fechada para o sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$.

Em aplicações práticas, esta classe de restrições pode ser obtida antes de serem usados os resolutores de restrições, por exemplo, durante a "compilação", e se houver a garantia que as restrições que são construídas durante a execução são de $\mathcal{LU}^w$.

6.4 Factorização

Se uma restrição $c \in \mathcal{LU}$ contém conjunções de disjunções, podemos reduzir o tamanho dessas disjunções utilizando a propriedade distributiva para factorizar informação comum. A factorização de disjunções vai permitir extrair informação determinística comum e diminuir o tamanho das restrições às quais ter-se-á de aplicar a regra S8 do sistema de reescrita completo $\mathsf{S}_{\mathcal{M}}$.

Considere-se a restrição

$$(\mathsf{c}_{22}) \qquad ((x = a \land y = b) \lor (x = a \land y.f_2^2 \doteq b)) \land ((y = b \land z = c) \lor (y = b \land z = y))$$

Factorizando $x = a$ na primeira disjunção e $y = b$ na segunda, obtemos a restrição

$$(\mathsf{c}_{23}) \qquad x = a \land (y = b \lor y.f_2^2 \doteq b) \land y = b \land (z = c \lor z = y)$$

Considerando a forma geral (6.1) vem

$$(\mathsf{c}_{24}) \qquad x = a \land y = b \land (y = b \lor y.f_2^2 \doteq b) \land (z = c \lor z = y)$$

Aplicando apenas o sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$ a restrição c_{24} simplifica para

$$(\mathsf{c}_{25}) \qquad x = a \land y = b \land (z = c \lor z = b)$$

6.4.1 Factorização de Formas Normais

Sendo $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{LU}$ restrições satisfazíveis, vamos caracterizar a informação positiva comum a $\mathcal{C}_1$ e $\mathcal{C}_2$.

Definição 6.29 *Sejam $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{LU}$ restrições satisfazíveis em $\mathcal{RT}$. Uma forma normal $\mathcal{M}$, diferente de $\top$, é um* factor positivo comum *a $\mathcal{C}_1$ e a $\mathcal{C}_2$ se $\mathcal{C}_1 \models_{\mathcal{RT}} \mathcal{M}$ e $\mathcal{C}_2 \models_{\mathcal{RT}} \mathcal{M}$, isto é, se $\mathcal{M}$ é um modelo parcial de ambas. Esta noção generaliza-se para qualquer número de restrições de $\mathcal{LU}$.*

Como na definição de redundâncias de restrições, ver página 117, as restrições da forma $x.f^n$ não serão, em geral, consideradas *factores positivos comuns*.

Começamos por estudar o caso em que as restrições $\mathcal{C}_1$ e $\mathcal{C}_2$ são formas normais. Uma primeira versão da proposição seguinte foi apresentada em [DMB93].

Proposição 6.30 *Sejam $\mathcal{M}_1$ e $\mathcal{M}_2$ formas normais. Então*

$$\mathcal{RT} \models (\mathcal{M}_1 \vee \mathcal{M}_2) \Leftrightarrow \mathcal{F} \wedge (\mathcal{M}_1' \vee \mathcal{M}_2') \tag{6.3}$$

tal que

$$\langle \mathcal{M}_1, \mathcal{M}_2, \top \rangle \overset{\mathsf{F}}{\Rightarrow}^{\star} \langle \mathcal{M}_1', \mathcal{M}_2', \mathcal{F} \rangle$$

onde o sistema de reescrita F *para ternos* $\langle \mathcal{M}, \mathcal{M}', \mathcal{F} \rangle$ *é definido por:*

F1.	$\langle \{x_1 = v\} \cup \mathcal{M}, \{x_1 = v'\} \cup \mathcal{M}', \mathcal{F} \rangle$	$\Rightarrow$	$\langle \mathcal{M}, \mathcal{M}', \mathcal{F} \cup \{x_1 = v_1\} \rangle$ *se* $v_1 \in [v]_{\mathcal{M}}$, $v_1 \in [v']_{\mathcal{M}'}$ *e* $x_1 \not\equiv v_1$;
F2.	$\langle \{x.f_i^n \doteq v\} \cup \mathcal{M}, \{y.f_i^n \doteq v'\} \cup \mathcal{M}', \mathcal{F} \rangle$	$\Rightarrow$	$\langle \mathcal{M}, \mathcal{M}', \mathcal{F} \cup \{x_1.f_i^n \doteq v_1\} \rangle$ *se* $x_1 \in [x]_{\mathcal{M}}$, $x_1 \in [y]_{\mathcal{M}'}$, $v_1 \in [v]_{\mathcal{M}}$ *e* $v_1 \in [v']_{\mathcal{M}'}$;
F3.	$\langle \{x.f^n\} \cup \mathcal{M}, \{y.f^n\} \cup \mathcal{M}', \mathcal{F} \rangle$	$\Rightarrow$	$\langle \mathcal{M}, \mathcal{M}', \mathcal{F} \cup \{x_1.f^n\} \rangle$ *se* $x_1 \in [x]_{\mathcal{M}}$ *e* $x_1 \in [y]_{\mathcal{M}'}$ *e nem* x *nem* y *são definidas para nenhum* $1 \leq i \leq n$, *em* $\mathcal{M}$ *ou* $\mathcal{M}'$;

O sistema de reescrita F *termina e se o terno* $\langle \mathcal{M}_1', \mathcal{M}_2', \mathcal{F} \rangle$ *é irredutível para* F, *então* $\mathcal{M}_1'$ *e* $\mathcal{M}_2'$ *não têm factores positivos comuns.*

Dem. Cada regra do sistema de reescrita F é correcta se $\langle \mathcal{M}_1, \mathcal{M}_2, \mathcal{F} \rangle \overset{\mathsf{F}}{\Rightarrow} \langle \mathcal{M}_1', \mathcal{M}_2', \mathcal{F}' \rangle$ implicar que

$$\mathcal{RT} \models \mathcal{F} \wedge (\mathcal{M}_1 \vee \mathcal{M}_2) \Leftrightarrow \mathcal{F}' \wedge (\mathcal{M}_1' \vee \mathcal{M}_2')$$

A correcção das regras F1–F3 é imediata, em particular, a relação (6.3) verifica-se. A terminação resulta de ser finito o número de variáveis e restrições atómicas em $\mathcal{M}_1$ e em $\mathcal{M}_2$. Como $\mathcal{M}_1$ e $\mathcal{M}_2$ são formas normais, a regra F1 aplica-se no máximo uma vez para cada $x_1 \in \mathsf{Vars}(\mathcal{M}_1) \cap \mathsf{Vars}(\mathcal{M}_2)$. Pela mesma razão, as regras F2 e F3 aplicam-se no máximo uma vez para cada $x \in \mathsf{Vars}(\mathcal{M}_i)$, para $1 \leq i \leq 2$. Falta provar que se o terno $\langle \mathcal{M}_1', \mathcal{M}_2', \mathcal{F} \rangle$ é irredutível para o sistema de reescrita F, então $\mathcal{M}_1'$ e $\mathcal{M}_2'$

não têm factores positivos comuns. Note-se que, com a eventual excepção da aplicação da regra F3, $\mathcal{M}'_1$ e $\mathcal{M}'_2$ são formas normais e $\mathcal{F}$ é satisfazível, logo, podemos supor que está em forma normal[5]. Basta verificar que não existem restrições atómicas que sejam factores positivos comuns a $\mathcal{M}_1$ e a $\mathcal{M}_2$. Suponhamos que $\mathcal{M}'_i \models_{\mathcal{RT}} v' = v$, $v \not\equiv v'$, para $1 \leq i \leq 2$. Então, v e v' pertenceriam à mesma classe de equivalência de $=_{\mathcal{M}'_1}$ e de $=_{\mathcal{M}'_2}$ e F1 aplicar-se-ia. Analogamente, se $\mathcal{M}'_i \models_{\mathcal{RT}} x.f^n_1 \doteq v$, para $1 \leq i \leq 2$, também se concluiria que F2 se aplicaria. Finalmente, se $\mathcal{M}'_i \models_{\mathcal{RT}} x.f^n$ para $1 \leq i \leq 2$, $x.f^n$ é só considerado factor comum para $\mathcal{M}'_1$ e $\mathcal{M}'_2$ se não ocorrerem em $\mathcal{M}_i$ restrições da forma $u_i.f^n_k \doteq v'$ para $u_i \in [x]_{\mathcal{M}_i}$, $1 \leq i \leq 2$, caso em que F3 se aplicaria.
2

Sejam $\mathcal{M}_1$ e $\mathcal{M}_2$ formas normais e N e K dados por

$$\begin{array}{rcl} N & = & \max\{\mid \mathsf{Vars}(\mathcal{M}_1) \mid, \mid \mathsf{Vars}(\mathcal{M}_2) \mid\} \\ K & = & \max\{n \mid f^n \in \Sigma \text{ e é o functor de alguma classe } [x]_{\mathcal{M}_i}, 1 \leq i \leq 2\} \end{array}$$

Considerando que determinar se a intersecção de duas classes de equivalência é não vazia, é da ordem $O(N^2)$ ($O(N \log N)$ se os elementos estiverem ordenados), o algoritmo da proposição anterior tem complexidade da ordem $O(N^3)$.

Consideremos as formas normais

$$\text{(C26)} \quad \begin{array}{rcl} \mathcal{M}_1 & \equiv & x_2 = x_1 \ \wedge \ x_1.f^2_2 \doteq x_4 \ \wedge \ x_1.f^2 \ \wedge \ x_5 = x_4 \ \wedge \ x_6 = x_4 \ \wedge \ x_4.g^1 \\ \mathcal{M}_2 & \equiv & x_2 = x_3 \ \wedge \ x_3.f^2_2 \doteq x_7 \ \wedge \ x_3.f^2 \ \wedge \ x_5 = x_7 \ \wedge \ x_6 = x_7 \ \wedge \ x_7.g^1 \end{array}$$

cuja representação gráfica é, respectivamente

$$\begin{tikzcd} \mathcal{M}_1 & \mathcal{M}_2 \\ {[x_1, x_2]} \arrow[d, "f^2_2"] & {[x_2, x_3]} \arrow[d, "f^2_2"] \\ {[x_4, x_5, x_6]} \arrow[d, "g^1_1"] & {[x_5, x_6, x_7]} \arrow[d, "g^1_1"] \\ {} & {} \end{tikzcd}$$

Aplicando o sistema de reescrita F temos que $\mathcal{M}_1 \vee \mathcal{M}_2$ é equivalente em $\mathcal{RT}$ a

$$\text{(C27)} \quad \begin{array}{l} x_2.f^2 \ \wedge \ x_2.f^2_2 \doteq x_5 \ \wedge \ x_6 = x_5 \ \wedge \ x_5.g^1 \ \wedge \\ \quad ((x_1 = x_2 \ \wedge \ x_4 = x_5) \ \vee \ (x_3 = x_2 \ \wedge \ x_7 = x_5)) \end{array}$$

[5]Em cada passo podemos reduzir $\mathcal{F}$ a tal forma.

cuja representação do factor comum é:

$$\begin{array}{c} [x_2] \\ \big\downarrow{\scriptstyle f_2^2} \\ [x_5, x_6] \\ \big\downarrow{\scriptstyle g_1^1} \end{array}$$

Note-se que as condições de aplicabilidade da regra $\mathsf{F3}$ podiam não ser tão exigentes, omitindo-se a definibilidade das variáveis, o que permitiria a factorização de mais informação comum às duas restrições. No entanto, as restrições finais $\mathcal{M}'_1$ e $\mathcal{M}'_2$ não se encontrariam em forma normal.

Se $\mathcal{M}_1$ e $\mathcal{M}_2$ forem formas normais e $\mathcal{M}_1$ vincular $\mathcal{M}_2$, isto é, $\mathcal{M}_1 \models_{\mathcal{RT}} \mathcal{M}_2$, então $\mathcal{RT} \models (\mathcal{M}_1 \lor \mathcal{M}_2) \Leftrightarrow \mathcal{M}_2$. Neste caso, o sistema de reescrita F produz $\mathcal{M}'_2 \equiv \top$ e dizemos que a restrição $\mathcal{M}_2$ é *redundante* na disjunção $\mathcal{M}_1 \lor \mathcal{M}_2$.

Sendo $\mathcal{M}_1 \lor \ldots \lor \mathcal{M}_m$ uma disjunção de formas normais podemos eliminar as formas normais que são redundantes na disjunção usando o método anterior. Este método pode ser usado para a obtenção de modelos minimais, ver Capítulo 5, Corolário 5.3.

Podemos, ainda, utilizar o sistema de reescrita F, da Proposição 6.30, para obter factores positivos comuns a uma disjunção de várias formas normais.

Corolário 6.31 *Sejam $\mathcal{M}_1, \ldots, \mathcal{M}_m$ formas normais. Então*

$$\mathcal{M}_1 \lor \ldots \lor \mathcal{M}_m \approx_{\mathcal{RT}} \mathcal{F}_m \land \mathcal{D}_m \tag{6.4}$$

onde para $1 \leq i \leq m$, $\mathcal{F}_i$ e $\mathcal{D}_i$ são definidos por:

a) $\mathcal{F}_1 \equiv \mathcal{M}_1$
$\mathcal{D}_1 \equiv \top$

b) Para $1 \leq i \leq m-1$,

$$\langle \mathcal{F}_i, \mathcal{M}_{i+1}, \top \rangle \overset{\mathsf{F}}{\Rightarrow}^{\star} \langle \mathcal{F}'_i, \mathcal{M}'_{i+1}, \mathcal{F}_{i+1} \rangle$$
$$\mathcal{D}_{i+1} \equiv (\mathcal{F}'_i \land \mathcal{D}_i) \lor \mathcal{M}'_{i+1}$$

e para cada $1 \leq i \leq m-1$, $\mathcal{F}'_i \land \mathcal{D}_i$ e $\mathcal{M}'_{i+1}$ não têm factores positivos comuns. Mais ainda, $\mathcal{D}_m$ é equivalente a uma disjunção de formas normais que não têm factores positivos comuns.

Em particular, se $\mathcal{F}_i \equiv \top$, para algum $1 < i < m$, então

$$\mathcal{M}_1 \vee \ldots \vee \mathcal{M}_m \approx_{\mathcal{RT}} \mathcal{D}_i \vee \mathcal{M}_{i+1} \vee \ldots \vee \mathcal{M}_m \tag{6.5}$$

Dem. Sejam $\mathcal{M}_1$ e $\mathcal{M}_2$ formas normais e $\mathcal{C} \in \mathcal{LU}$. Se

$$\langle \mathcal{M}_1, \mathcal{M}_2, \top \rangle \overset{\mathsf{F}}{\Rightarrow}^{\star} \langle \mathcal{M}_1', \mathcal{M}_2', \mathcal{F} \rangle$$

então

$$\mathcal{RT} \models (\mathcal{M}_1 \wedge \mathcal{C}) \vee \mathcal{M}_2 \Leftrightarrow \mathcal{F} \wedge ((\mathcal{M}_1' \wedge \mathcal{C}) \vee \mathcal{M}_2')$$

Prova-se por indução sobre m que para todo $m > 1$, a relação (6.4) é verificada. Analogamente, prova-se por indução sobre m que para todo $1 \leq i \leq m-1$, $\mathcal{F}_i' \wedge \mathcal{D}_i$ e $\mathcal{M}_{i+1}'$ não têm factores positivos comuns. Para $m = 2$, é verdade pela Proposição 6.30. Supondo que é verdade para $m-1$, basta provar que $\mathcal{F}_{m-1}' \wedge \mathcal{D}_{m-1}$ e $\mathcal{M}_m'$ não têm factores comuns. Mas $\mathcal{D}_{m-1}$, por hipótese de indução, é uma disjunção de restrições que não têm factores comuns. Por outro lado, $\mathcal{F}_{m-1}'$ e $\mathcal{M}_m'$ também não têm factores positivos comuns.

Então, em particular, se reduzirmos $\mathcal{D}_m$ a forma normal disjuntiva as formas normais resultantes não têm factores positivos comuns.

Se $\mathcal{F}_i \equiv \top$, para algum $1 < i < m$ então, para todo $i+1 \leq j \leq m$, $\mathcal{F}_j \equiv \top$ e $\mathcal{D}_j \equiv \mathcal{D}_{j-1} \vee \mathcal{M}_j$. 2

Na prática, antes de determinar factores positivos comuns a uma disjunção de formas normais será conveniente separar as formas normais em grupos independentes para a satisfazibilidade em $\mathcal{RT}$. Essa separação pode ser feita usando um dos critérios de $\mathcal{M}$-independência, atrás apresentados, considerando $\mathcal{M} \equiv \top$.

Se $\mathcal{C}_1$ e $\mathcal{C}_2$ forem formas primárias podemos aplicar o sistema de reescrita F às respectivas formas normais.

Corolário 6.32 *Sejam $\mathcal{P}_1$ e $\mathcal{P}_2$ formas primárias. Então*

$$\mathcal{RT} \models (\mathcal{P}_1 \vee \mathcal{P}_2) \Leftrightarrow \mathcal{F} \wedge (\mathcal{P}_1' \vee \mathcal{P}_2') \tag{6.6}$$

onde $\mathcal{F}$ é uma forma normal e $\mathcal{P}_1'$ e $\mathcal{P}_2'$ são formas primárias sem factores positivos comuns, obtidas por aplicação do sistema de reescrita F às formas normais de $\mathcal{P}_1$ e $\mathcal{P}_2$.

Dem. Sejam $\mathcal{P}_1 \equiv \mathcal{P}_p^1 \wedge \mathcal{P}_n^1$ e $\mathcal{P}_2 \equiv \mathcal{P}_p^2 \wedge \mathcal{P}_n^2$. Se

$$\langle \mathcal{P}_p^1, \mathcal{P}_p^2, \top \rangle \overset{\mathsf{F}}{\Rightarrow}^{\star} \langle \mathcal{P}_p^3, \mathcal{P}_p^4, \mathcal{F} \rangle$$

e $\langle \mathcal{P}_p^3, \mathcal{P}_p^4, \mathcal{F}\rangle$ é irredutível para F então

$$\mathcal{RT} \models (\mathcal{P}_p^1 \land \mathcal{P}_n^1) \lor (\mathcal{P}_p^2 \land \mathcal{P}_n^2) \Leftrightarrow \mathcal{F} \land ((\mathcal{P}_p^3 \land \mathcal{P}_n^1) \lor (\mathcal{P}_p^4 \land \mathcal{P}_n^2))$$

com $\mathcal{P}_1' \equiv \mathcal{P}_p^3 \land \mathcal{P}_n^1$ e $\mathcal{P}_2' \equiv \mathcal{P}_p^4 \land \mathcal{P}_n^2$. 2

Corolário 6.33 *Sejam $\mathcal{P}_1, \ldots, \mathcal{P}_m$ formas primárias tais que para cada $1 \leq i \leq m$, $\mathcal{P}_i \equiv \mathcal{M}_i \land \mathcal{N}_i$. Então*

$$\mathcal{P}_1 \lor \ldots \lor \mathcal{P}_m \approx_{\mathcal{RT}} \mathcal{F}_m \land \mathcal{D}_m \tag{6.7}$$

onde $\mathcal{F}_i$ e $\mathcal{D}_i$, para $1 \leq i \leq m$, são definidos por:

a) $\mathcal{F}_1 \equiv \mathcal{M}_1$
$\mathcal{D}_1 \equiv \mathcal{N}_1$

b) Para $1 \leq i \leq m-1$,
$\langle \mathcal{F}_i, \mathcal{M}_{i+1}, \top\rangle \overset{\mathsf{F}}{\Rightarrow}^{\star} \langle \mathcal{F}_i', \mathcal{M}_{i+1}', \mathcal{F}_{i+1}\rangle$
$\mathcal{D}_{i+1} \equiv (\mathcal{F}_i' \land \mathcal{D}_i) \lor (\mathcal{M}_{i+1}' \land \mathcal{N}_{i+1})$

e para cada $1 \leq i \leq m-1$, as restrições $\mathcal{F}_i' \land \mathcal{D}_i$ e $\mathcal{M}_{i+1}' \land \mathcal{N}_{i+1}$ não têm factores positivos comuns.

6.4.2 Factorização de Formas Primárias

Para generalizar os resultados anteriores a formas primárias, alguns cuidados serão necessários. Neste caso, uma restrição pode ser consequência de duas restrições sem ser considerada *factor comum.*

Seja a restrição

$$(x = a \land y = b) \lor (x = c \land y = d) \lor (x = a \land y \neq a) \tag{c_{28}}$$

Obviamente, $x \neq y$ é consequência de todos os elementos da disjunção, mas factorizar essa informação aumenta o tamanho da restrição, e portanto, não é em geral desejável. Por outro lado, em

$$x \neq y \lor (x = a \land y = b) \tag{c_{29}}$$

$$x \neq a \lor x = b \tag{c_{30}}$$

a mesma factorização permite reduzir c_{29} a $x \neq y$ e a factorização de $x \neq a$ permite reduzir c_{30} a $x \neq a$. Nestes dois casos um dos elementos da disjunção vincula o outro, isto é, subsume o outro. Mais geralmente, dada uma disjunção de formas primárias podemos eliminar as formas primárias que são subsumidas por outras, usando o facto de que:

Sendo $\mathcal{P}_1$ e $\mathcal{P}_2$ formas primárias,

$$\mathcal{P}_1 \models \mathcal{P}_2 \text{ se e só se } \mathcal{P}_1 \wedge \mathcal{P}_2 \approx_{\mathcal{RT}} \mathcal{P}_1 \text{ se e só se } \mathcal{P}_1 \vee \mathcal{P}_2 \approx_{\mathcal{RT}} \mathcal{P}_2$$

Para determinar se uma forma primária subsume outra podemos usar o sistema de reescrita P ou o método que iremos apresentar na Secção 6.5.1, Corolário 6.40. Ao contrário do que sucedia para formas normais, neste caso não será possível usar para este fim, o processo de factorização de elementos comuns, a seguir apresentado.

Assim, para obter factores comuns a duas formas primárias vamos supor que nenhuma delas subsume (vincula) a outra e apenas iremos factorizar informação comum não redundante a ambas as formas primárias.

Antes, porém, vamos relacionar a factorizações duma disjunção de formas primárias com as disjunções de modelos minimais dessa disjunção.

Idealmente, em vez de apenas exigirmos que uma forma primária na disjunção não subsuma outra, pretenderíamos considerar uma disjunção de modelos minimais sem redundâncias. Contudo, como visto no Capítulo 5, a obtenção de tal disjunção parece obrigar à determinação de todos os modelos minimais o que é obviamente extremamente ineficiente.

Deste modo, embora uma forma primária não vincule a outra, pode acontecer que uma restrição atómica duma seja redundante na disjunção das duas. E, assim, de algum modo exista informação redundante "comum". Considere-se o seguinte exemplo:

(c_{31}) $\qquad x = a \wedge y = b$

(c_{32}) $\qquad x = a \wedge y \neq b \wedge z = c$

Nenhuma das restrições subsume a outra, mas a restrição $y \neq b$ é redundante na disjunção $\mathsf{c}_{31} \vee \mathsf{c}_{32}$, isto é,

$$\mathsf{c}_{31} \vee \mathsf{c}_{32} \approx_{\mathcal{RT}} (x = a \wedge y = b) \vee (x = a \wedge z = c)$$

Esta equivalência pode-se obter considerando uma resolvente entre os dois elementos da disjunção, isto é, obtendo os modelos minimais da disjunção.

A detecção, necessariamente parcial se eficiente, deste tipo de informação não será contudo estudada neste trabalho.

Definição 6.34 *Sejam $\mathcal{P}_1$, $\mathcal{P}_2$ e $\mathcal{P}$ formas primárias. A restrição $\mathcal{P}$ é um* factor comum estrito *a $\mathcal{P}_1$ e a $\mathcal{P}_2$ se $\mathcal{P}_1 \models_{\mathcal{RT}} \mathcal{P}$ e $\mathcal{P}_2 \models_{\mathcal{RT}} \mathcal{P}$, isto é, $\mathcal{P}$ é um modelo parcial de ambos, e $\mathcal{P}$ não é redundante nem em $\mathcal{P}_1$ nem em $\mathcal{P}_2$. Esta noção generaliza-se para qualquer número de restrições em $\mathcal{LU}$.*

No exemplo c_{28}, a restrição $x \neq y$ é redundante em todos os elementos da disjunção, logo não é um factor comum estrito.

Lema 6.35 *Sejam $\mathcal{P}_1$ e $\mathcal{P}_2$ formas primárias. Uma forma normal $\mathcal{M}$ é um factor positivo comum a $\mathcal{P}_1$ e a $\mathcal{P}_2$ se e só se $\mathcal{M}$ é um factor comum estrito a $\mathcal{P}_1$ e a $\mathcal{P}_2$.*

Dem. ($\Rightarrow$)

Para $1 \leq i \leq n$, sejam $\mathcal{P}_i \equiv \mathcal{P}_p^i \ \wedge \ \mathcal{P}_n^i$ e $\mathcal{P}_n^i \equiv n_1^i \ \wedge \ \ldots \ \wedge \ n_{m_i}^i$, com $m_i \geq 1$. Se $\mathcal{P}_i \models_{\mathcal{RT}} \mathcal{M}$, então

$$\mathcal{P}_p^i \models_{\mathcal{RT}} \mathcal{M} \ \vee \ (\mathcal{P}_p^i \ \wedge \ c_1^i) \ \vee \ \ldots \ \vee \ (\mathcal{P}_p^i \ \wedge \ c_{m_i}^i)$$

com $c_j^i \equiv \neg n_j^i$, para $1 \leq j \leq m_i$ e $1 \leq i \leq 2$. Então, pela Proposição 4.27, como $\mathcal{P}_p^i \not\models_{\mathcal{RT}} \mathcal{P}_p^i \ \wedge \ c_1^i$, concluímos que $\mathcal{M}$ é um factor positivo comum a $\mathcal{P}_p^1$ e a $\mathcal{P}_p^2$, logo não é redundante nem em $\mathcal{P}_1$ nem em $\mathcal{P}_2$.

($\Leftarrow$)
Óbvio. 2

O lema seguinte caracteriza as restrições atómicas redundantes numa forma primária, que correspondem àquelas que não serão factores comuns estritos.

Lema 6.36 *Seja $\mathcal{P} \equiv \mathcal{P}_p \ \wedge \ \mathcal{P}_n$ uma forma primária sem redundâncias e c_1 uma restrição atómica, tal que c_1 é um modelo parcial de $\mathcal{P}$. A restrição c_1 não trivial é redundante em $\mathcal{P}$ se e só se verifica uma das seguintes condições:*

(i) *c_1 é do tipo $x \neq a$ e a classe $[x]_{\mathcal{P}_p}$ é constante diferente de a ou é funcional;*

(ii) *c_1 é do tipo $\neg x.f^n$ e a classe $[x]_{\mathcal{P}_p}$ é constante ou é funcional com functor diferente de f^n;*

(iii) c_1 *é do tipo* $x \neq y$ e $x_1 \neq y_1 \notin \mathcal{P}_n$ *com* $x \in [x_1]_{\mathcal{P}_p}$, $y \in [y_1]_{\mathcal{P}_p}$ *e as classes* $[x_1]_{\mathcal{P}_p}$ *e* $[y_1]_{\mathcal{P}_p}$ *são incompatíveis em* $\mathcal{P}$.

Dem. ($\Leftarrow$)

Se c_1 verifica uma das condições é fácil ver que c_1 é redundante em $\mathcal{P}$.

($\Rightarrow$)

Suponhamos que $\mathcal{P} \models c_1$. Se c_1 fosse uma restrição atómica positiva então $\mathcal{P}_p \models c_1$ e, pelo Lema 4.35, c_1 ocorre em $\mathcal{P}_p$, a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{P}_p}$. Se c_1 é do tipo $\neg x.\tau$, onde se τ é uma constante a, $c_1 \equiv x \neq a$, e as condições (i) ou (ii) não se verificam então ou $x_1.\tau \in \mathcal{P}_p$ para $x \in [x_1]_{\mathcal{P}_p}$ ou não é verdade que $\mathcal{P} \models c_1$. Se c_1 é do tipo $x \neq y$ e não verifica as condições então, também, é fácil ver que $x \neq y \in \mathcal{P}$, a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{P}_p}$, ou não é verdade que $\mathcal{P} \models c_1$. 2

No exemplo c_{30} a restrição $x \neq a$ é redundante em $x = b$ mas não é para o outro elemento da disjunção. Como já foi dito, no exemplo c_{28}, a restrição $x \neq y$ é redundante em todos os elementos da disjunção.

Para a factorização estrita de formas primárias, basta adaptar o sistema de reescrita F a formas primárias e adicionar regras para as restrições atómicas negativas.

Proposição 6.37 *Sejam* $\mathcal{P}_1$ e $\mathcal{P}_2$ *duas formas primárias. Então*

$$\mathcal{RT} \models (\mathcal{P}_1 \vee \mathcal{P}_2) \Leftrightarrow \mathcal{F} \wedge (\mathcal{P}'_1 \vee \mathcal{P}'_2) \tag{6.8}$$

tal que

$$\langle \mathcal{P}_1, \mathcal{P}_2, \top \rangle \overset{\mathsf{F^e}}{\Rightarrow^\star} \langle \mathcal{P}'_1, \mathcal{P}'_2, \mathcal{F} \rangle$$

onde o sistema de reescrita $\mathsf{F^e}$ *para ternos* $\langle \mathcal{C}, \mathcal{C}', \mathcal{F} \rangle$, *com* $\mathcal{C} \equiv \mathcal{C}_p \wedge \mathcal{C}_n$, $\mathcal{C}' \equiv \mathcal{C}'_p \wedge \mathcal{C}'_n$ e $\mathcal{F} \equiv \mathcal{F}_p \wedge \mathcal{F}_n$, *é constituído pelas regras* $\mathsf{F1}$–$\mathsf{F3}$ *do sistema de reescrita* F *da Proposição 6.30, que são aplicadas primeiro a* $\mathcal{C}_p$ *e* $\mathcal{C}'_p$, *e pelas regras seguintes:*

$$\mathsf{F^e4.} \quad \langle \{x \neq v\} \cup \mathcal{C}, \{y \neq v'\} \cup \mathcal{C}', \mathcal{F}\rangle \Rightarrow \langle \mathcal{C}, \mathcal{C}', \mathcal{F} \cup \{x_1 \neq v_1\}\rangle$$

se

$$\begin{array}{lcl} x_1 & \in & [x]_{\mathcal{C}_p \cup \mathcal{F}_p} \\ x_1 & \in & [y]_{\mathcal{C}'_p \cup \mathcal{F}_p} \\ v_1 & \in & [v]_{\mathcal{C}_p \cup \mathcal{F}_p} \\ v_1 & \in & [v']_{\mathcal{C}'_p \cup \mathcal{F}_p} \end{array}$$

$$\mathsf{F^e5.} \quad \langle \{\neg x.f^n\} \cup \mathcal{C}, \{\neg y.f^n\} \cup \mathcal{C}', \mathcal{F}\rangle \Rightarrow \langle \mathcal{C}, \mathcal{C}', \mathcal{F} \cup \{\neg x_1.f^n\}\rangle$$

se

$$\begin{array}{lcl} x_1 & \in & [x]_{\mathcal{C}_p \cup \mathcal{F}_p} \\ x_1 & \in & [y]_{\mathcal{C}'_p \cup \mathcal{F}_p} \end{array}$$

O sistema de reescrita $\mathsf{F^e}$ *termina e se o terno* $\langle \mathcal{P}'_1, \mathcal{P}'_2, \mathcal{F}\rangle$ *é irredutível para o sistema de reescrita* $\mathsf{F^e}$*, então* $\mathcal{P}'_1$ e $\mathcal{P}'_2$ *não têm factores comuns estritos.*

Dem. A correcção e a terminação do sistema de reescrita $\mathsf{F^e}$ são imediatas. Apenas é necessário provar que se $\langle \mathcal{P}'_1, \mathcal{P}'_2, \mathcal{F}\rangle$ é irredutível para o sistema de reescrita $\mathsf{F^e}$, então $\mathcal{P}'_1$ e $\mathcal{P}'_2$ não têm factores comuns estritos. Pelo Lema 6.35 e pela Proposição 6.30, $\mathcal{P}'_1$ e $\mathcal{P}'_2$ não têm factores positivos comuns estritos. Por outro lado, se c_1 for um factor negativo comum estrito a $\mathcal{P}'_1$ e $\mathcal{P}'_2$, usando o Lema 6.36 e argumentos análogos aos da demonstração da Proposição 6.30, concluímos que uma das regras $\mathsf{F^e4}$ ou $\mathsf{F^e5}$ se aplica.
2

Podemos utilizar o sistema de reescrita $\mathsf{F^e}$ para obter factores comuns estritos a uma disjunção de formas primárias.

Corolário 6.38 *Sejam* $\mathcal{P}_1, \ldots, \mathcal{P}_m$ *formas primárias. Então*

$$\mathcal{P}_1 \vee \ldots \vee \mathcal{P}_m \approx_{\mathcal{RT}} \mathcal{F}_m \wedge \mathcal{D}_m \tag{6.9}$$

onde $\mathcal{F}_i$ e $\mathcal{D}_i$*, para* $1 \leq i \leq m$*, são definidos por:*

a) $\mathcal{F}_1 \equiv \mathcal{P}_1$
$\mathcal{D}_1 \equiv \top$

b) Para $1 \leq i \leq m-1$, $\langle \mathcal{F}_i, \mathcal{P}_{i+1}, \top\rangle \overset{\mathsf{F^e}}{\Rightarrow^\star} \langle \mathcal{F}'_i, \mathcal{P}'_{i+1}, \mathcal{F}_{i+1}\rangle$
$\mathcal{D}_{i+1} \equiv (\mathcal{F}'_i \wedge \mathcal{D}_i) \vee \mathcal{P}'_{i+1}$

e para cada $1 \leq i \leq m-1$, $\mathcal{F}_i' \wedge \mathcal{D}_i$ *e* $\mathcal{P}_{i+1}'$ *não têm factores comuns estritos. Mais ainda,* $\mathcal{D}_m$ *é equivalente a uma disjunção de formas primárias que não têm factores comuns estritos.*

Em particular, se $\mathcal{F}_i \equiv \top$*, para algum* $1 < i < m$*, então*

$$\mathcal{P}_1 \vee \ldots \vee \mathcal{P}_m \approx_{\mathcal{RT}} \mathcal{D}_i \vee \mathcal{P}_{i+1} \vee \ldots \vee \mathcal{P}_m \tag{6.10}$$

Dem. Como a demonstração do Corolário 6.31. 2

Finalmente, se $\mathcal{C} \in \mathcal{LU}$ for uma restrição na forma geral (6.1) podemos factorizar informação comum a elementos duma disjunção, supondo as conjunções de restrições atómicas em forma primária. Em particular, podemos supor que $\mathcal{C}$ é fechada para o sistema de reescrita $\mathsf{S}_\top^-$, e aplicar a seguinte regra:

FG1. $(\mathcal{P}_1 \wedge \mathcal{C}_1) \vee (\mathcal{P}_2 \wedge \mathcal{C}_2) \longmapsto \mathcal{F} \wedge ((\mathcal{P}_1' \wedge \mathcal{C}_1') \vee (\mathcal{P}_2' \wedge \mathcal{C}_2'))$

para $1 \leq i \leq 2$, $\mathcal{P}_i$ são formas primárias, $\mathcal{C}_i$ são conjunções, possivelmente vazias, de restrições não primárias e

$$\langle \mathcal{P}_1, \mathcal{P}_2, \top \rangle \overset{\mathsf{F^e}}{\Rightarrow^\star} \langle \mathcal{F}, \mathcal{P}_1', \mathcal{P}_2' \rangle$$
$$\mathcal{C}_i \overset{\mathsf{FG}}{\longmapsto^\star} \mathcal{C}_i'$$

Esta regra generaliza as regras do Corolário 6.38. Se $\mathcal{C}$ é irredutível para esta regra, então $\mathcal{C}$ será mais adequada para a aplicação da regra S8 do sistema de reescrita completo $\mathsf{S}_\mathcal{M}$.

6.5 Formas Primárias como Modelos Parciais

O facto de se factorizar uma forma primária "comum" a uma disjunção de restrições, e não apenas, uma forma normal, é uma das motivações para a extensão, no sistema de reescrita $\mathsf{S}_\mathcal{M}$, do modelo parcial $\mathcal{M}$ a formas primárias. No entanto, o facto da informação negativa poder ser redundante cria, mais uma vez, problemas adicionais.

Sendo $\mathcal{C}$ uma restrição fechada para o sistema de reescrita $\mathsf{S}_\mathcal{M}$, enquanto que a forma normal $\mathcal{M}$ é independente de $\mathcal{C}$, a informação negativa contida no modelo pode ser redundante em relação a $\mathcal{C}$. Consideremos os seguintes exemplos:

(C33) $x \neq a \wedge (x = b \vee (y = a \wedge (x = c \vee x.f^n)))$

(c_{34}) $\quad x \neq a \ \wedge \ (x = b \ \vee \ (y = a \ \wedge \ z = b))$

A restrição $x \neq a$ é um modelo parcial de c_{33} e de c_{34}. Contudo, na restrição c_{33}, é redundante e, na restrição c_{34}, é redundante no primeiro elemento da disjunção. Assim, enquanto que em c_{33} pode ser eliminado, em c_{34} deve "passar para dentro". A restrição c_{34} é equivalente em $\mathcal{RT}$ a

(c_{35}) $\quad x = b \ \vee \ (x \neq a \ \wedge \ y = a \ \wedge \ z = b)$

O mesmo resultado é obtido usando a regra S8 do sistema de reescrita completo $\mathsf{S}_{\mathcal{M}}$.

No entanto, se o número de elementos da disjunção for grande, pode ser preferível manter factorizada (no modelo parcial) informação redundante. Neste caso, os sistemas de reescrita a seguir apresentados serão preferíveis.

6.5.1 Satisfazibilidade Incremental para Formas Primárias

Nesta secção vamos generalizar o sistema de reescrita $\mathsf{I}_{\mathcal{M}}$, apresentado na Secção 6.1, para o caso do modelo ser uma forma primária. Dada uma conjunção $\mathcal{C}$ de restrições atómicas e uma forma primária $\mathcal{P}$, se $\mathcal{C}$ é $\mathcal{M}$-satisfazível pretendemos obter as "diferenças" entre $\mathcal{P}$ e a forma primária de $\mathcal{P} \ \wedge \ \mathcal{C}$. Sem falta de generalidade podemos supor que $\mathcal{C}$ é uma forma primária.

Proposição 6.39 *Sejam $\mathcal{P} \equiv \mathcal{P}_p \ \wedge \ \mathcal{P}_n$ e $\mathcal{C} \equiv \mathcal{C}_p \ \wedge \ \mathcal{C}_n$ formas primárias. Se $\mathcal{C}$ é $\mathcal{P}$-satisfazível, então $\mathcal{P} \models_{\mathcal{RT}} \mathcal{C} \ \Leftrightarrow \ \mathcal{C}_1$, onde $\mathcal{C}_1 \equiv \mathcal{C}_p^1 \ \wedge \ \mathcal{C}_n^1$ e $\mathcal{C} \overset{\mathsf{I}^e}{\mapsto}{}^{\star}_{\mathcal{P}} \mathcal{C}_1$. O sistema de reescrita $\mathsf{I^e}_{\mathcal{P}}$ é constituído pelas regras do sistema $\mathsf{I}_{\mathcal{P}_p}$ aplicado a $\mathcal{C}$ e pelas regras a seguir apresentadas. O sistema de reescrita $\mathsf{I}_{\mathcal{P}_p}$ aplica-se primeiro para obter $\mathcal{C}_p^1 \wedge \mathcal{C}_n^0$ fechada para $\mathsf{I}_{\mathcal{P}_p}$. As restantes regras do sistema $\mathsf{I^e}_{\mathcal{P}}$ são as seguintes:*

$\mathsf{I^e}13$. $\{a \neq x\} \ \cup \ \mathcal{C} \mapsto_{\mathcal{P}} \{x \neq a\} \ \cup \ \mathcal{C}$

$\mathsf{I^e}14$. $\{a \neq b\} \ \cup \ \mathcal{C} \mapsto_{\mathcal{P}} \mathcal{C}$ *se a e b são constantes distintas;*

$\mathsf{I^e}15$. $\{\neg a.f^n\} \ \cup \ \mathcal{C} \mapsto_{\mathcal{P}} \mathcal{C}$

$\mathsf{I^e}16$. $\{x \neq a\} \ \cup \ \mathcal{C} \mapsto_{\mathcal{P}} \mathcal{C}$ *se para algum $f^n \in \Sigma$, $x.f^n \in \mathcal{C}_p^1$ ou ou $x \in [y]_{\mathcal{C}_p^1}$ e $y.f^n \in \mathcal{P}_p$ ou $y \neq a \in \mathcal{P}_n$;*

$\mathsf{I^e}17$. $\{\neg x.g^m\} \ \cup \ \mathcal{C} \mapsto_{\mathcal{P}} \mathcal{C}$ *se $x.f^n \in \mathcal{C}_p^1$ ou $x \in [y]_{\mathcal{C}_p^1}$, $y.f^n \in \mathcal{P}_p$ ou $\neg y.g^m \in \mathcal{P}_n$, para algum $f^n \neq g^m$;*

$\mathsf{I}^{\mathsf{e}}18.$ $\{x \neq y\} \cup \mathcal{C} \mapsto_{\mathcal{P}} \mathcal{C}$ se $x_1 \neq y_1 \in \mathcal{P}_n$ e $x_1 \in [x]_{\mathcal{C}_p^1}$ e $y_1 \in [y]_{\mathcal{C}_p^1}$;

$\mathsf{I}^{\mathsf{e}}19.$ $\{x \neq y, y \neq x\} \cup \mathcal{C} \mapsto_{\mathcal{P}} \{x \neq y\} \cup \mathcal{C}$

$\mathsf{I}^{\mathsf{e}}20.$ $\{x \neq v, x \neq v\} \cup \mathcal{C} \mapsto \{x \neq v\} \cup \mathcal{C}$

$\mathsf{I}^{\mathsf{e}}21.$ $\{\neg x.f^n, \neg x.f^n\} \cup \mathcal{C} \mapsto \{\neg x.f^n\} \cup \mathcal{C}$

O sistema de reescrita $\mathsf{I}^{\mathsf{e}}{}_{\mathcal{P}}$ termina e se $\mathcal{C}_1$ é irredutível para $\mathsf{I}^{\mathsf{e}}{}_{\mathcal{P}}$, então $\mathcal{C}_1$ é uma forma primária e $\mathcal{P} \wedge \mathcal{C}_1$ coincide com uma forma primária $\mathcal{C}' \equiv \mathcal{C}'_p \wedge \mathcal{C}'_n$ de $\mathcal{P} \wedge \mathcal{C}$ a menos de permutação de valores na mesma classe de equivalência de $=_{\mathcal{C}'_p}$ e a presença de restrições atómicas equivalentes. Neste caso, escreve-se $\mathcal{C}' \doteqdot \mathcal{P} \wedge \mathcal{C}_1$.

Dem. Cada regra do sistema de reescrita $\mathsf{I}^{\mathsf{e}}{}_{\mathcal{P}}$ é correcta em relação a $\mathcal{P}$, se $\mathcal{C} \mapsto_{\mathcal{P}} \mathcal{C}'$ implicar $\mathcal{P} \models_{\mathcal{RT}} \mathcal{C} \Leftrightarrow \mathcal{C}'$. As regras do sistema de reescrita $\mathsf{I}_{\mathcal{P}_p}$ são correctas e as regras $\mathsf{I}^{\mathsf{e}}13$–$\mathsf{I}^{\mathsf{e}}21$ coincidem com regras do sistema de reescrita P, para redução a forma primária, considerando a forma normal $\mathcal{C}'_p \doteqdot \mathcal{P}_p \wedge \mathcal{C}_p^1$. A terminação resulta da terminação dos dois sistemas referidos, $\mathsf{I}_{\mathcal{P}_p}$ e P, e da sua ordem de aplicação. Resulta ainda que $\mathcal{C}_p^1$ está em forma normal e se $\mathcal{C}_1$ é irredutível para $\mathsf{I}^{\mathsf{e}}{}_{\mathcal{P}}$ então é uma forma primária. Para além da repetição de restrições atómicas positivas (a menos de permutação de valores na mesma classe de equivalência de $=_{\mathcal{C}'_p}$), pode haver repetição de restrições atómicas negativas em $\mathcal{P}_n$ e $\mathcal{C}_n^1$. 2

Note-se que não é necessário que a conjunção $\mathcal{C}$ de restrições atómicas esteja em forma primária para que o sistema de reescrita $\mathsf{I}^{\mathsf{e}}{}_{\mathcal{P}}$ seja aplicado. Na realidade, no sistema de reescrita $\mathsf{I}_{\mathcal{M}}$ não se supõe que a conjunção de restrições atómicas positivas esteja em forma normal e as regras $\mathsf{I}^{\mathsf{e}}13$–$\mathsf{I}^{\mathsf{e}}21$ coincidem essencialmente com as usadas para a redução a forma primária.

Se a forma primária $\mathcal{P}$ não contiver restrições redundantes e $\mathcal{C} \overset{\mathsf{I}^{\mathsf{e}}}{\mapsto}{}^{\star}_{\mathcal{P}} \mathcal{C}_1$, a restrição $\mathcal{C}_1 \equiv \mathcal{C}_p^1 \wedge \mathcal{C}_n^1$ só pode conter restrições redundantes da forma $x \neq y$. Estas redundâncias, podem ser apenas em $\mathcal{C}_1$ ou em $\mathcal{P} \wedge \mathcal{C}_1$. Para verificar se uma restrição $x \neq y \in \mathcal{C}_n^1$ é redundante em $\mathcal{P} \wedge \mathcal{C}_1$, podemos usar o método do Corolário 4.38 considerando a respectiva forma primária. Claro que este teste pode ser feito incrementalmente, quando se aplica a regra $\mathsf{I}^{\mathsf{e}}18$.

Corolário 6.40 *Sejam $\mathcal{P}$ uma forma primária, $\mathcal{C}$ uma conjunção de restrições atómicas e $\mathcal{P}_1$ uma forma primária obtida de $\mathcal{C}$, irredutível para o sistema de reescrita $\mathsf{I}^{\mathsf{e}}{}_{\mathcal{P}}$ e tal que $\mathcal{P}_1$ não contém restrições redundantes em $\mathcal{P} \wedge \mathcal{P}_1$. Então*

(i) $\mathcal{P}_1 \equiv \mathcal{P}_p^1 \wedge \mathcal{P}_n^1$ *é única a menos de permutação de variáveis na mesma classe de equivalência de $=_{\mathcal{P}_p^1}$.*

(ii) $\mathcal{P}_1 \equiv \top$ *se e só se* $\mathcal{P} \models_{\mathcal{RT}} \mathcal{C}$.

A Proposição 6.39 pode-se generalizar para uma conjunção de formas primárias, correspondendo a um processo incremental de satisfazibilidade análogo ao do Corolário 6.6.

Corolário 6.41 *Sendo* $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_m$ *conjunções de restrições atómicas, se*

$$\mathcal{C}_0 \ \wedge \ \mathcal{C}_1 \ \wedge \ \ldots \ \wedge \ \mathcal{C}_m$$

é satisfazível em $\mathcal{RT}$*então, a forma primária* $\mathcal{P}'$ *de* $\mathcal{C}_0 \ \wedge \ \mathcal{C}_1 \ \wedge \ \ldots \ \wedge \ \mathcal{C}_m$ *coincide a menos de permutação de valores na mesma classe de equivalência de* $=_{\mathcal{P}'}$ *e repetição de restrições atómicas, com* $\mathcal{P}_0 \ \wedge \ \mathcal{P}_1 \ \wedge \ \ldots \ \wedge \ \mathcal{P}_m$*, onde* $\mathcal{P}_i$ *e* $\mathcal{P}'_i$*, para* $0 \leq i \leq m$*, são definidos por:*

a) $\mathcal{P}'_0 \equiv \mathcal{P}_0 \quad$ *se* $\quad \mathcal{C}_0 \overset{\mathsf{P}}{\mapsto^\star} \mathcal{P}_0$

b) $\mathcal{P}'_{i+1} \doteq \mathcal{P}'_i \ \wedge \ \mathcal{P}_{i+1} \quad$ *se* $\quad \mathcal{C}_{i+1} \overset{\mathsf{I^e}}{\mapsto^\star_{\mathcal{P}'_i}} \mathcal{P}_{i+1}, \quad$ *para* $0 \leq i \leq m-1$

6.5.2 O Sistema de Reescrita Completo $\mathsf{S^e}_\mathcal{P}$

O sistema de reescrita $\mathsf{S}_\mathcal{M}$ pode ser generalizado para o caso do modelo ser uma forma primária. Sendo $\mathcal{P}$ uma forma primária, o conjunto de regras de reescrita do sistema $\mathsf{S^e}_\mathcal{P}$ para um modelo $\mathcal{P}$ ou $\perp$ é apresentado na Tabela 6.4.

Teorema 6.42 *O sistema de reescrita* $\mathsf{S^e}_\mathcal{P}$ *é correcto (em* $\mathcal{RT}$*), termina e é completo para a satisfazibilidade em* $\mathcal{RT}$*.*

Dem. A correcção do sistema de reescrita $\mathsf{S^e}_\mathcal{P}$ é imediata pela correcção do sistema de reescrita $\mathsf{I^e}_\mathcal{P}$ e do sistema de reescrita $\mathsf{S}_\mathcal{M}$. Como para o sistema de reescrita $\mathsf{S}_\mathcal{M}$, a não terminação apenas pode ser causada pela aplicação das regras $\mathsf{S^e7}$ ou $\mathsf{S^e8}$. Suponhamos que a regra $\mathsf{S^e8}$ não se aplica e para $\mathcal{C} \in \mathcal{LU}$ seja $\gamma'(\mathcal{C})$ o número de conjunções de restrições atómicas (incluindo $\top$ e $\perp$). Se mais nenhuma regra se aplica, a regra $\mathsf{S^e7}$ aplica-se uma vez para cada conjunção de restrições atómicas, transformando-as, duma maneira única, em formas primárias (ou $\perp$). Cada uma das restantes regras diminui o valor de γ'. Assim por cada aplicação duma destas regras o valor de γ' ou diminui ou mantém-se constante para um número finito de aplicações. Podemos então concluir que não pode haver uma derivação infinita de regras de $\mathsf{S^e}_\mathcal{P}$, excluindo a regra $\mathsf{S^e8}$. Para esta regra, consideremos a função γ, análoga à usada na demonstração do Teorema 6.20,

$\gamma(\mathcal{C}) = 2$	se $\mathcal{C}$ é uma conjunção de restrições atómicas
$\gamma(\mathcal{C}_1 \ \vee \ \mathcal{C}_2) = \gamma(\mathcal{C}_1) + \gamma(\mathcal{C}_2) + 1$	
$\gamma(\mathcal{C}_1 \ \wedge \ \mathcal{C}_2) = \gamma(\mathcal{C}_1) \times \gamma(\mathcal{C}_2)$	

$\mathsf{S^e}1.$	$\mathcal{C}$	$\longmapsto_{\perp}$	$\perp$
$\mathsf{S^e}2.$	$\mathcal{C} \wedge \top$	$\longmapsto_{\mathcal{P}}$	$\mathcal{C}$
$\mathsf{S^e}3.$	$\mathcal{C} \wedge \perp$	$\longmapsto_{\mathcal{P}}$	$\perp$
$\mathsf{S^e}4.$	$\mathcal{C} \vee \top$	$\longmapsto_{\mathcal{P}}$	$\top$
$\mathsf{S^e}5.$	$\mathcal{C} \vee \perp$	$\longmapsto_{\mathcal{P}}$	$\mathcal{C}$
$\mathsf{S^e}6.$	$\mathcal{C}_1 \wedge \mathcal{C}$	$\longmapsto_{\mathcal{P}}$	$\perp$ se $\mathcal{C}_1$ é uma conjunção, não vazia, de restrições atómicas, $\mathcal{C}$ é uma conjunção, possivelmente vazia, de restrições não atómicas e $\mathcal{P} \wedge \mathcal{C}_1 \overset{\mathsf{P}}{\mapsto}^{\star} \perp$;
$\mathsf{S^e}7.$	$\mathcal{C}_1 \wedge \mathcal{C}$	$\longmapsto_{\mathcal{P}}$	$\mathcal{C}_1' \wedge \mathcal{C}'$ se $\mathcal{C}_1$ é uma conjunção, não vazia, de restrições atómicas, $\mathcal{C}$ é uma conjunção, possivelmente vazia, de restrições não atómicas $\mathcal{P} \wedge \mathcal{C}_1 \overset{\mathsf{P}}{\mapsto}^{\star} \mathcal{P}'$ $\mathcal{C} \overset{\mathsf{S^e}}{\longmapsto}^{\star}_{\mathcal{P}'} \mathcal{C}'$ $\mathcal{P}' \doteqdot \mathcal{P} \wedge \mathcal{C}_1'$ e $\mathcal{C} \not\equiv \mathcal{C}'$ ou $\mathcal{C}_1' \not\equiv \mathcal{C}_1$;
$\mathsf{S^e}8.$	$(\mathcal{C}_1 \vee \mathcal{C}_2) \wedge \mathcal{C}_3$	$\longmapsto_{\mathcal{P}}$	$(\mathcal{C}_1 \wedge \mathcal{C}_3) \vee (\mathcal{C}_2 \wedge \mathcal{C}_3)$ se nenhuma das regras anteriores se aplica, $\mathcal{C}_3$ não é uma conjunção de restrições atómicas e nem $\mathcal{C}_1$ nem $\mathcal{C}_2$ são $\mathcal{P}$-independentes com $\mathcal{C}_3$.

Tabela 6.4: Sistema de reescrita $\mathsf{S^e}_{\mathcal{P}}$

O valor de γ não aumenta por aplicação de nenhuma regra de $\mathsf{S^e}_{\mathcal{P}}$ e decresce estritamente por aplicação da regra $\mathsf{S^e}8$. Considerando as propriedades da função γ' e o controlo da regra $\mathsf{S^e}8$, temos que o par (γ, γ'), ordenado lexicograficamente, é decrescente por aplicação das regras de $\mathsf{S^e}_{\mathcal{P}}$. A demonstração da completude de $\mathsf{S^e}_{\mathcal{P}}$ é análoga à do Teorema 6.21 excepto que no passo de indução para $\mathcal{C} \equiv \mathcal{C}_1 \ \wedge \ \mathcal{C}_2$, considera-se $\mathcal{C}_1$ uma forma primária e a regra $\mathsf{S^e}7$. Note-se que o Corolário 6.13 e o Lema 6.15 são válidos para $\mathsf{S^e}_{\mathcal{P}}$ e $\mathcal{P}$-independência. 2

Denotamos por $\mathsf{S^e}^-_{\mathcal{P}}$ o sistema de reescrita que resulta de $\mathsf{S^e}_{\mathcal{P}}$ excluindo a regra $\mathsf{S^e}8$. Como o sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$, o sistema de reescrita $\mathsf{S^e}^-_{\mathcal{P}}$ tem complexidade polinomial mas é incompleto para a satisfazibilidade. Pelo Corolário 6.13, adaptado para $\mathsf{S^e}^-_{\mathcal{P}}$, o sistema de reescrita $\mathsf{S^e}^-_{\mathcal{P}}$ é completo para conjunções de restrições atómicas.

Os critérios de $\mathcal{P}$-independência não podem ser directamente adaptados dos obtidos para a $\mathcal{M}$-independência e o sistema de reescrita $\mathsf{S}^-_{\mathcal{M}}$.

Dada a presença de negações no modelo, a Proposição 6.17 já não é válida, *i.e*, para duas formas resolvidas não basta que o seus conjuntos de variáveis sejam disjuntos. O seguinte exemplo, trivial, ilustra este facto:

(C36)
$$\begin{array}{rcl} \mathcal{P} & \equiv & x \neq y \\ \mathcal{C}_1 & \equiv & x = a \\ \mathcal{C}_2 & \equiv & y = a \end{array}$$

A noção de variáveis $\mathcal{P}$-*ligadas* tem de ser alterada para incluir o caso de que se $x \in \mathsf{Vars}(\mathcal{C})$ e $x \neq y \in \mathcal{P}$ então $y \in \mathsf{Var}_{\mathcal{P}}(\mathcal{C})$. Deste modo, as variáveis que dependam de y em $\mathcal{M}$, também pertencerão a $\mathsf{Var}_{\mathcal{P}}(\mathcal{C})$.

Com esta modificação a Proposição 6.18 verifica-se para $\mathcal{P}$-independência e o sistema de reescrita $\mathsf{S^e}^-_{\mathcal{P}}$.

Os métodos de factorização de formas primárias apresentados na Secção 6.4.2 podem ser usados antes da aplicação da regra $\mathsf{S^e}8$, reduzindo, eventualmente, o tamanho das restrições a que essa regra se aplica.

Finalmente,

Corolário 6.43 *Sendo* $\mathcal{C}_0 \in \mathcal{LU}$, $\mathcal{C}_0 \overset{\mathsf{S^e}}{\longmapsto}{}^{\star}_{\top} \ \mathcal{P} \ \wedge \ \mathcal{C}$ *tal que:*

(i) $\mathcal{P} \equiv \bot$ *se e só se* $\mathcal{RT} \models \mathcal{C}_0 \ \Leftrightarrow \ \bot$

(ii) $\mathcal{P}$ é uma forma primária e $\mathcal{RT} \models \mathcal{C}_0 \Leftrightarrow \mathcal{P} \wedge \mathcal{C}$, caso contrário.

Como foi dito, no início desta secção, os modelos parciais $\mathcal{P}$ obtidos pelo sistema de reescrita $\mathsf{S^e}_{\mathcal{P}}$, podem conter redundâncias que seriam eliminadas pelo sistema de reescrita $\mathsf{S}_{\mathcal{M}}$. Esse facto foi ilustrado nos exemplos c_{33} e c_{34}. No caso geral, para que num modelo parcial $\mathcal{P}$ não se mantenham redundâncias será necessário "marcar", para cada restrição atómica negativa, quais as restrições em que é redundante. Se $\mathcal{C}$ for uma restrição fechada para $\mathsf{S^e}_{\mathcal{P}}$ e $n_i \in \mathcal{P}$ for redundante em $\mathcal{C}$, para eliminar n_i podemos usar o Lema 6.36 e substituir a regra $\mathsf{S^e7}$ do sistema de reescrita $\mathsf{S^e}_{\mathcal{P}}$, pela regra seguinte:

$\mathsf{S^e7a.}$ $\quad \mathcal{C}_1 \wedge \mathcal{C} \quad \longmapsto_{\mathcal{P}} \quad \mathcal{C}_1'' \wedge \mathcal{C}'$

se $\mathcal{C}_1$ é uma conjunção, não vazia, de restrições atómicas, $\mathcal{C}$ é uma conjunção, possivelmente vazia, de restrições não atómicas,

$$\mathcal{P} \wedge \mathcal{C}_1 \overset{\mathsf{P}}{\mapsto^\star} \mathcal{P}'$$
$$\mathcal{C} \overset{\mathsf{S^e}}{\longmapsto^\star_{\mathcal{P}'}} \mathcal{C}'$$
$$\mathcal{P}' \doteqdot \mathcal{P} \wedge \mathcal{C}_1'$$

e $\mathcal{C} \not\equiv \mathcal{C}'$ ou $\mathcal{C}_1' \not\equiv \mathcal{C}_1$ e $\mathcal{C}_1''$ resulta de $\mathcal{C}_1'$ eliminando as restrições redundantes em $\mathcal{C}'$;

Se $\mathcal{C}$ for uma disjunção e $n_i \in \mathcal{P}$ for redundante só em alguns elementos da disjunção, a regra anterior já não é aplicável. Neste caso, podemos usar uma variante da regra $\mathsf{S^e8}$ modificando o próprio modelo parcial $\mathcal{P}$. Se, como para o sistema de reescrita SP, o par $\langle \mathcal{P}, \mathcal{C} \rangle$ representar a conjunção $\mathcal{P} \wedge \mathcal{C}$ e $\mathcal{C}$ for fechada para $\mathsf{S^e}_{\mathcal{P}}$, temos a seguinte regra:

$$\langle \mathcal{P} \cup \{n_i\}, \mathcal{C}_1 \vee \mathcal{C}_2 \rangle \quad \Rightarrow \quad \langle \mathcal{P}, \mathcal{C}_1 \vee (n_i \wedge \mathcal{C}_2) \rangle$$

se n_i é redundante em $\mathcal{C}_1$ e não o é em $\mathcal{C}_2$

6.6 Sumário

Neste capítulo, apresentamos resolutores para $\mathcal{LU}$ baseados na factorização de informação determinística, retardando e evitando a multiplicação de disjunções, e reduzindo o tamanho das restrições envolvidas nessa multiplicação. Deste modo, pretende-se reduzir o custo computacional da determinação da satisfazibilidade em $\mathcal{LU}$.

Os sistemas de reescrita constroem (extraem) um modelo parcial duma restrição que é usado para simplificar a restrição resultante. Em cada etapa as conjunções de restrições atómicas são reduzidas a formas primárias ou $\perp$. Este processo tem complexidade polinomial.

Uma propriedade importante dos resolutores é que qualquer modelo minimal da restrição inicial pode ser obtido pela reunião do modelo parcial com um modelo minimal da restrição resultante.

Foram apresentados métodos para a detecção de informação comum nas disjunções que permitem também a redução do tamanho das restrições.

Para a obtenção de um resolutor completo para a satisfazibilidade, é necessário ainda multiplicar as disjunções, que restarem na restrição, e que não forem independentes para a satisfazibilidade. Para tal, foram analisados alguns critérios para a independência de restrições em relação a um modelo.

Consideramos e comparamos os casos do modelo parcial ser uma forma normal ou uma forma primária. Enquanto que no primeiro caso, o modelo é independente da restrição resultante, no segundo não o é e pode conter redundâncias. Assim, a informação negativa introduz problemas adicionais que diminuem a vantagem da sua utilização.

Capítulo 7

Implementação dos Resolutores

Neste capítulo vamos fazer algumas considerações sobre a implementação dos resolutores de restrições apresentados no Capítulo 6. A eficiência destas implementações depende da unificação de árvores racionais e da reescrita de restrições complexas dado um modelo parcial. Não iremos analisar essa eficiência em função de tempos de execução e comparação com outros resolutores para árvores racionais (ou de estruturas de atributos). Nenhum dos resolutores recentes de que temos conhecimento manipulam restrições complexas gerais. Os resolutores para estruturas de atributos das linguagens de programação por restrições LIFE e Oz [PR94, ST94], determinam a satisfazibilidade e o vínculo de conjunções de restrições atómicas. Em [RvH93] é descrito um algoritmo incremental para satisfazibilidade e o vínculo, sobre árvores racionais. A resolução de equações e desigualdades em árvores finitas é também apresentada em [Smi91]. Mais semelhante, é o trabalho de T. Keisu [Kei94], onde se apresentam resolutores completos para $\mathcal{T}$ e $\mathcal{RT}$, com assinaturas infinitas, desenhados no âmbito das linguagens de programação concorrente por restrições [Sar93]. No entanto, as formas normais utilizadas reflectem fortemente os requisitos daquele paradigma e em particular da linguagem AKL, não sendo possível uma imediata (e desejável) comparação entre os dois sistemas.

Começamos por apresentar um algoritmo que, dado um conjunto de restrições atómicas positivas tem como resultado uma forma normal, como um conjunto de normalizado de classes, ou `false`, se o conjunto não é satisfazível. Os aspectos fundamentais do algoritmo são implementados em linguagem C mas é fornecido um interface em linguagem Prolog através dum conjunto de predicados básicos. Os sistemas de reescrita completos foram escritos em Yap Prolog [DCAR89] usando estes predicados. A componente do resolutor em C implementa essencialmente a unificação de formas resolvidas dum modo que é muito similar ao da unificação de termos em Prolog. A diferença desta implementação reside no uso duma trilha que permite não só a obtenção duma forma resolvida mas também a implementação dos sistemas de reescrita de satisfazibilidade incremental. Estas implementações, sem a referência a restrições negativas, foram descritas em [DMB93].

A implementação dos resolutores de restrições sobre árvores (racionais, finitas ou de atributos) com interface em Prolog corresponde a uma extensão desta linguagem a diversas instâncias do esquema $\mathcal{CLP}$.

Na Secção 7.1 é definida uma representação canónica para formas normais. Na Secção 7.2 são descritas as estruturas de dados usadas para as árvores racionais e para os modelos parciais. O algoritmo que implementa o sistema de reescrita para formas normais é apresentado na Secção 7.3. A sua generalização para implementar os sistemas de reescrita I de satisfazibilidade incremental, e $\mathsf{S}_{\mathcal{M}}$ para restrições complexas, é apresentada na Secção 7.4. Nesta secção são também discutidos os modos de implementar as restrições negativas. Na Secção 7.5, é descrita uma implementação dos sistemas de reescrita que admitem formas primárias como modelos parciais. Modificações das implementações anteriores para as linguagens de árvores de atributos são apresentadas na Secção 7.6. Finalmente, na Secção 7.7, são referidas diversas extensões que foram ou poderão ser implementadas.

7.1 Conjunto Normalizado de Classes

No Capítulo 4 vimos que uma forma normal $\mathcal{M}$ é única a menos de permutações de variáveis, da mesma classe de equivalência da relação $=_{\mathcal{M}}$. Para obter uma representação canónica basta introduzirmos uma relação de ordem nessas classes. Para tal, estendemos a relação $=_{\mathcal{M}}$ definida em $\mathsf{Val}(\mathcal{M})$ a expressões do tipo $x.f_i^n$ considerando que $x.f_i^n =_{\mathcal{M}} v$ se $x.f_i^n \doteq v \in \mathcal{M}$. A relação $=_{\mathcal{M}}$ é a menor relação de equivalência que verifica esta condição e as restantes condições para elementos de $\mathsf{Val}(\mathcal{M})$.

Designamos as expressões do tipo $x.f_i^n$ por *projecções*, uma vez que representam a *projecção* (sub-termo) i do termo (racional) cujo functor é f^n. As variáveis x, y, $\dots$, as constantes a, b, $\dots$, os functores f^n, g^k, $\dots$ e projecções $x.f_i^n$, $\dots$ serão designadas por *entradas* e o seu conjunto denotado por $\mathcal{E}$.

Seja $\preceq$ uma relação de ordem definida no conjunto de variáveis X e na assinatura Σ, respectivamente. Essas relações induzem, naturalmente, uma relação de ordem $\preceq_{\mathcal{E}}$ em $\mathcal{E}$ a seguir definida.

Para $e_1,\ e_2 \in \mathcal{E}$, $e_1 \preceq_{\mathcal{E}} e_2$ se:

- e_1 é uma constante e e_2 é uma variável;
- e_1 é uma variável e e_2 é uma projecção;
- e_1 e e_2 são ambas constantes, functores ou variáveis e $e_1 \preceq e_2$;
- e_1 e e_2 são projecções $x.f_i^n$ e $y.g_j^k$, respectivamente e:
  - $x \prec y$;
  - $x \equiv y$ e $f^n \prec g^k$;
  - $f^n \equiv g^k$ e $i \leq j$.

Dada uma forma normal $\mathcal{M}$ e a relação de ordem $\preceq_{\mathcal{E}}$, $\mathcal{M}$ é representada por um conjunto normalizado de classes[1] $\mathcal{N} = \{l_1, \ldots, l_m\}$ tal que para $1 \leq i \leq m$:

(i) l_i é do tipo $v_1 = v_2 = \ldots = v_k$ (*classe de igualdades*) ou do tipo $x.f^n$ (*funcional*); no primeiro caso, cada v_l é uma variável, uma constante ou uma projecção, $v_l =_{\mathcal{M}} v_j$ e $v_l \prec_{\mathcal{E}} v_j$, para $1 \leq l < j \leq k$; neste caso, representa-se a classe por uma lista $[v_1, v2, \ldots, v_k]$.

(ii) l_i tem no máximo uma constante e, nesse caso, é o primeiro elemento de l_i;

(iii) se l_i é uma classe de igualdades e $v \in l_i$ então v ocorre exactamente uma vez em $\mathcal{N}$;

(iv) se l_i é do tipo $x.f^n$, $x.f^n \in \mathcal{M}$ e x não ocorre noutra l_j deste tipo;

(v) se l_i é uma classe de igualdades, $x \in l_i$ e l_i contém uma constante então não existe em $\mathcal{N}$ nenhum l_j igual a $x.f^n$, para nenhum $f^n \in \Sigma$;

(vi) se l_i é uma classe de igualdades e $x.f_k^n \in l_i$ então existe um l_j em $\mathcal{N}$ igual a $x.f^n$;

(vii) se $x \sim_{\mathcal{M}} y$ então x e y pertencem a uma mesma classe de igualdades l_i de $\mathcal{N}$.

Considerando a relação $\preceq_{\mathcal{E}}$, podemos ainda generalizá-la para uma relação de ordem em $\mathcal{N}$ supondo que as classes de igualdades são "menores" que as funcionais. No entanto, essa relação de ordem só será utilizada nos algoritmos de factorização e será omitida nos algoritmos que vamos descrever.

[1]Note-se que esta noção de classe não corresponde à de classes de equivalência.

7.2 Representação de Árvores Racionais

A representação de árvores racionais que será usada, permite não só uma implementação eficiente da unificação mas também fornece um modo incremental de obter modelos parciais, que é adequado para implementar o sistema de reescrita $\mathsf{S}_{\mathcal{M}}$, *i.e*, a reescrita de restrições complexas com "contexto".

Dado um conjunto de restrições atómicas positivas, vamos utilizar um algoritmo de unificação destrutivo, produzindo uma trilha. Usando essa trilha é possível obter um *modelo parcial* representado por um conjunto normalizado.

Cada *entrada* é representada por uma estrutura, de tipo `Slot`, numa tabela denominada `SlotChain`. Cada uma dessas estruturas tem os seguintes campos:

`kind` pode tomar os valores $AtomS$, $VarS$, $FuncS$ ou $SlotS$ indicando que a entrada é, respectivamente, uma *constante*, uma *variável*, uma *functor* ou uma *projecção*;

`name` se a entrada corresponde a uma constante, um functor ou uma variável este campo corresponde a um *identificador* (um átomo Prolog na implementação actual); se a entrada é uma projecção este campo corresponde ao *inteiro* associado a essa projecção;

`value` é um apontador na cadeia de entradas na mesma classe de igualdades; caso contrário o seu valor é `NIL`;

`daughters` se uma entrada é uma variável (funcional) ou uma projecção este campo é um apontador para a cadeia das suas projecções (sub-árvores) ou restantes projecções (irmãs), respectivamente; caso contrário o seu valor é `NIL`;

`arity` se a entrada é um functor este campo é um inteiro correspondente à sua aridade; caso contrário o seu valor é `NIL`;

`base` se a entrada é uma projecção $x.f_i^n$ este campo é um apontador para a entrada correspondente à variável x;

se a entrada é uma variável, corresponde a um eventual apontador para o functor associado a x;

`next` é um apontador para a entrada seguinte na tabela, se não corresponder a uma sub-árvore; caso contrário o seu valor é `NIL`;

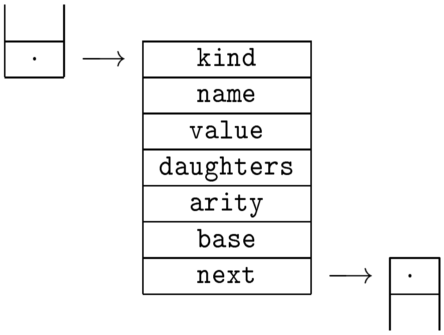


Figura 7.1: Tabela de entradas

7.2.1 Representação de um Modelo Parcial

Os modelos parciais vão ser representados por uma `trilha`. A `trilha` é uma pilha que contém apontadores para as entradas cujo campo `value` mudou durante o processo. Em cada momento, apenas é necessário representar um único modelo parcial, o *modelo activo*. Dois apontadores `TrailOld` e `TrailPtr` marcam o início e o fim da porção da `trilha` que está activa. O apontador `TrailPtr` indica sempre o topo da pilha.

7.3 Algoritmo para Formas Normais

O algoritmo para a obtenção de formas normais permitirá implementar o sistema de reescrita I apresentado no Capítulo 6, Secção 6.1. Contudo, nesta secção, vamos apenas descrever um algoritmo que, dado um conjunto de restrições atómicas, tem como resultado uma forma normal, como um conjunto de normalizado de classes, ou `false` se o conjunto for não satisfazível. Baseia-se na implementação feita por L. Damas dum algoritmo para a obtenção de formas resolvidas para a lógica de estruturas de atributos.

O algoritmo é constituído por 3 fases.

Construção da Trilha Inicialmente, a unificação de um modelo parcial com o modelo activo é reduzida a uma série de operações elementares em que o modelo activo é estendido com uma restrição atómica. Embora cada tipo de restrição atómica tenha um tratamento específico podemos, para simplificar a exposição, englobá-las na função `Unify` apresentada na Figura 7.3.

Como habitualmente, a substituição de termos é implementada por um mecanismo de asso-

```
DerefSlot(Slot v)
{   while(v->value) v=v->value;
    return v;
}
Bind(Slot v1,Slot v2)
{   v1->value=v2;
    *TrailPtr++=v1;
}
```

Figura 7.2: Desreferenciação de uma entrada e associação de duas entradas

ciação/desreferenciação. Cada entrada, depois de localizada ou criada a sua estrutura na `tabela de entradas`, é desreferenciada utilizando-se a função `DerefSlot` (Figura 7.2).

Para restrições do tipo $v = v'$ ou $x.f_i^n \doteq v$, o algoritmo de unificação é aplicado tendo como argumentos cada entrada que ocorre nas igualdades. As restrições do tipo $x.f^n$ serão analisadas posteriormente. No algoritmo de unificação sempre que duas entradas, não constantes, são associadas é chamada a função `Bind` (Figura 7.2), em que o campo `value` duma entrada é associado a outra entrada ("unificado") e um apontador para essa entrada é colocado na `trilha`. Deste modo, o modelo activo é estendido. Sempre que duas variáveis são unificadas é necessário assegurar que as suas sub-árvores também podem ser unificadas. A função `UnifySubTrees` (Figura 7.4) realiza esta tarefa. Se a entrada unificada não tiver as mesmas sub-árvores que a entrada unificadora, uma nova entrada é criada. Isto pode gerar algumas redundâncias que serão eliminadas quando a forma normal for extraída.

Os predicados Prolog seguintes permitem adicionar restrições atómicas positivas a um modelo parcial:

`add_ac_va/2` adiciona uma restrição da forma $x = a$

`add_ac_vv/2` adiciona uma restrição da forma $x = y$

`add_ac_fa/5` adiciona uma restrição da forma $x.f_i^n \doteq a$

`add_ac_fv/5` adiciona uma restrição da forma $x.f_i^n \doteq x$

`add_ac_f/3` adiciona uma restrição da forma $x.f^n$

Para os dois primeiros predicados, os argumentos são os argumentos das igualdades e para os restantes também cada elemento da projecção (ou do functor): variável, functor, aridade e projecção. Como já se disse, para cada argumento a entrada correspondente é procurada na

```
Unify(Slot v1,Slot v2)
{   if(v1->kind>v2->kind) { /* troca v1 com v2 */
        Slot t=v1; v1=v2; v2=t; }
    if(v1->kind==AtomS) {
        if(v2->kind==AtomS) return v1==v2;
        if(v2->kind==VarS) {
            /* testa se v2 não é funcional */
            if(IsFunctor(v2)) return 0;
                Bind(v2,v1);
                return 1;
            }
    }
    /* v2 é uma projecção */
    if(v2->kind==SlotS) {
        /* testa se v2 não é da forma a.f_i^n */
        if(!IsProper(v2)) return 0;
        Bind(v2,v1);
        return 1;
    }
    /* v1 e v2 são variáveis*/
    if(v1==v2) return 1;
    if(!SameFunctor(v1,v2)) return 0;
    /* ordena variáveis */
    if(Compare(v2,v1)) { /* troca v1 com v2 */
        Slot t=v1; v1=v2; v2=t; }
    Bind(v2,v1);
    return UnifySubTrees(v1,v2);
}
```

Figura 7.3: Algoritmo de unificação

```
UnifySubTrees(Slot u, Slot v)
{   Slot du ,dv;
    if(!SameFunctor(v,u)) return 0;
    /* assegura que as sub-árvores de v unificam com as de u */
    du=u;
    dv=v->daughters;
    while(dv!=NIL) {
        while(du->daughters!=NIL && du->daughters->name < dv->name)
            du=du->daughters;
        /* se u não tem esta sub-árvore ela é criada */
        /* contudo, poderá ser elimimada se não for necessária */
        if(du->daughters==NIL || du->daughters->name!=dv->name) {
            Slot t = (Slot) tmp_alloc(sizeof(*t));
            t->kind=SlotS;
            t->name=dv->name;
            t->arity=dv->arity;
            t->base=u;
            t->value=NIL;
            t->daughters=du->daughters;
            du->daughters=t;
        }
        /* unifica as sub-árvores correspondentes de u e v */
        if(!Unify(DerefSlot(du->daughters),DerefSlot(dv)) return 0;
        dv=dv->daughters;
    }
    return 1;
    }
}
```

Figura 7.4: Unificação de sub-árvores `tabela de entradas` e se não for encontrada é criada e acrescentada à tabela. Neste passo podem ser detectadas algumas incompatibilidades entre os argumentos que asseguram que a unificação irá falhar. Em seguida é chamada a função `Unify` (ou uma função equivalente, que trate casos mais específicos) e se falhar o predicado correspondente também falha.

Podemos ilustrar o funcionamento destes predicados com o objectivo Prolog

```
add_ac_va(x,a).
```

Primeiro são pesquisadas na `tabela de entradas` as estruturas referentes à variável `x` e à constante `a`. Seguidamente a variável `x` é desreferenciada seguindo a cadeia de campos `value`. Se a cadeia termina numa constante diferente de `a`, o predicado falha. Se a cadeia termina numa variável `y` então

- se o campo `base` de `y` é um apontador para um functor $\mathtt{f}^{\mathtt{n}}$, o predicado falha;
- senão
  - o campo `value` de `y` passa a ser o apontador para `a`;
  - o apontador para `y` é adicionado à `trilha`.

A adição duma restrição da forma $x.f^n$ é feita de modo diferente.

Considere-se o objectivo Prolog

```
add_ac_f(x,f,n)
```

Neste caso, não há a unificação de duas entradas, mas apenas a associação do functor $\mathtt{f}^{\mathtt{n}}$ à variável `x`. Para isso associa-se ao campo `base` da entrada `x` um apontador para a entrada correspondente ao functor, se tal for possível. Para que esta informação possa ser usada na extracção do modelo parcial (forma normal) o apontador para `x` é acrescentado também à `trilha`. Na implementação actual existem de facto duas trilhas uma para guardar entradas cujo campo `value` foi alterado e outra para guardar as entradas cujo campo `base`, correspondente a um functor, foi alterado.

Normalização Antes de se obter a forma normal é ainda necessário determinar se existem variáveis cuja igualdade é vinculada pelo modelo parcial. A função `Check_eq_vars` (Figura 7.5) implementa o algoritmo para identificar classes distinguíveis numa forma resolvida, apresentado no Capítulo 4 (Figura 4.1), e acrescenta ao modelo as necessárias restrições atómicas do tipo $x = y$ (sistema de reescrita N). Como apenas as variáveis funcionais podem ainda ser não distinguíveis, a `trilha` de functores é percorrida e são marcadas as variáveis distinguíveis. Num segundo passo, as variáveis não marcadas e com o mesmo functor, são associadas pela função `Bind`.

Obtenção de um Modelo Parcial Quando não existirem mais restrições atómicas para adicionar e nenhuma falha ocorreu, o predicado `undo_ac` permite obter uma forma normal como um conjunto normalizado de classes. Na Figura 7.6 apresenta-se uma versão simplificada deste predicado, implementado pela função `Undo_ac`. A estrutura `SolvForm` que representa a forma normal é implementada como uma lista de listas de termos Prolog. Cada classe é uma lista constituída por termos Prolog das seguintes formas:

`atom(v)` representa uma constante e `v` é seu identificador;

`var(v)` representa uma variável e `v` é seu identificador;

`feat(v,f,n,i)` representa uma projecção cuja variável é identificada por `v`, o functor por `f`, a aridade é o inteiro `n` e a projecção é o inteiro `i`;

`func(v,f,n)` representa uma restrição do tipo $v.f^n$, onde `v`, `f` e `n` são como no caso anterior.

As funções `StartClass`, `AddtoClass` e `MkClasses` constroem as classes e asseguram uma representação canónica única.

Começando no topo da `trilha` cada entrada é desreferenciada e as entradas que referem um mesmo valor são agrupadas numa classe. Os apontadores para essas entradas passam temporariamente para o fim da trilha (que é, apontado por `AuxTrailPtr`), e a sua posição é apagada da trilha. Se uma entrada é uma projecção a sua variável, valor do campo `base`, tem de ser desreferenciada e, para isso, é chamada a função `DerefFeatS`. A projecção correspondente é adicionada à classe. Este passo é necessário, porque essa entrada (projecção) podia ter sido adicionada à trilha antes da variável que lhe está associada. Como iremos exemplificar, permite, também, eliminar as redundâncias que tenham sido criadas pela função `UnifySubTrees`. Quando já não houver nenhuma entrada na trilha são construídas as restrições funcionais do tipo $x.f^n$. O processo é idêntico ao anterior usando-se uma trilha própria, onde foram guardadas as variáveis funcionais que ocorreram na primeira fase.

Finalmente, as trilhas são esvaziadas e os campos `value` ou `base`, de cada uma das entradas que esteve numa `trilha` é apagado.

Para ilustrar o funcionamento da trilha considere-se a restrição

```
Check_eq_vars()
{  Slot fun, f1, f2, f11, f21, *p2, *p1=FTrailPtr,*AuxTM=AuxTrailPtr;
   Int n;
   /* procura na trilha variáveis funcionais */
   while(p1!=FTrailBase) {
      f1=*--p1;
      if(!f1) continue;
      f1=DerefSlot(f1); fun=f1->base; p2=p1;
      while(p2!=FTrailBase) {
         f2=*--p2;
         if(!f2) continue;
         if(f2->base !=fun) continue;
         f2=DerefSlot(f2);
         if(f1==f2) continue;
         n=fun->arity; f11=f1->daughters; f21=f2->daughters;
      /* verifica se f11 e f21 já são distinguíveis */
         while(n && f11 && f21 && f11->name==f21->name
            && !check_marked(f11,f21)) {
            f11=f11->daughters; f21=f21->daugthres; n--;}
      /* marca f1 e f2 como distinguíveis, e os ascendentes não marcados */
         if(n) mark_div(f1,f2);
   }  }
   p1=FTrailPtr;
   /* as variáveis funcionais não marcadas são unificadas */
   while(p1!=FTrailBase) {
      f1=*--p1;
      if(!f1) continue;
      f1=DerefSlot(f1); fun=f1->base; p2=p1;
      while(p2!=FTrailBase) {
         f2=*--p2;
         if(!f2) continue;
         f2=DerefSlot(f2);
         if(f1==f2 || f2->base!=fun) continue;
         if(!Marked(f1,f2)) {
            if(f1->kind > f2->kind) {Slot f=f1;f1=f2;f2=f;}
            if(Compare(&f1,&f2)>0) {Slot f=f1; f1=f2; f2=f;}
            Bind(f2,f1);}}
   }/* apaga as marcas */
   while(AuxTrailPtr!=AuxTM) Clean_Mark(*AuxTrailPtr++);
}
```

Figura 7.5: Verificar a existência de variáveis iguais

```
Undo_ac()
{   Slot *p,*q,r0,r1;
    SolvForm classes=NIL;
    q=AuxTrailPtr;
    while(1){
        *p=TrailPtr;
        r0=*--p;
        /* procura uma entrada na trilha; se uma entrada foi removida
        da trilha, o valor correspondente na trilha é 0 */
        while(p!=TrailOld && !r0)) r0=*--p;
        if(p==TrailOld) break; /* a trilha está vazia */
        r0=DerefSlot(r0);
        ++p;
        /* inicio da construção duma classe */
        StartClass();
        AddToClass(r0);
        /* procura de outra entrada na mesma classe */
        while(p!=TrailOld){
            r1=*--p;
            if(!r1) continue;
        /* se não pertence à mesma classe, procura outra entrada */
            if(DerefSlot(r1)!=r0) continue;
            *--q=r1; *p=0; /* remove r1 da trilha */
            /* se é uma projecção, desreferencia a variável */
            r1=DerefFeatS(r1);
            AddToClass(r1);
        }
        classes=MkClasses(MkAtomicClass(),classes);
    }
    /* constrói restrições func(x,f,n) */
    Build_Functor();
    classes=MkClasses(MkFunctor(),classes);
    /* limpa as entradas e inicializa os limites na trilha
    do novo modelo activo */
    while(q!=AuxTrailPtr)(*q++)->value=NIL;
    while(FTrailPtr!=FTrailOld)(*--FTrailPtr)->base=NIL;
    TrailPtr=TrailOld;
    FTrailPtr=FTrailOld;
    if (ChoicePtr!=ChoicePointBase){
        TrailOld=*--ChoicePtr;
        FTrailOld=*--ChoicePtr;
    }
}
```

Figura 7.6: Obtenção duma forma normal (c$_1$) $\quad z.f_1^4 \doteq b \;\wedge\; z = y \;\wedge\; z = x \;\wedge\; w = b \;\wedge\; x.f_1^4 \doteq u \;\wedge\; x.f_1^4 \doteq b$

Depois de adicionar estas restrições atómicas, a trilha contém apontadores para as entradas descritas na tabela seguinte

`Slot s`	`DerefSlot(s)`
$z.f_1^4$	b
z	y
$y.f_1^4$	b
y	x
$x.f_1^4$	b
w	b
u	b

Nesta tabela, o terceiro e quinto elementos são criados pela unificação de sub-árvores e o último é obtido por desreferenciação de $\mathtt{x.f}_1^4$. A última restrição atómica $x.f_1^4 \doteq b$ é trivialmente verdadeira, e portanto nada é acrescentado na trilha. Começando pela entrada `u` no topo da trilha, é construída a classe $[\mathtt{b} = \mathtt{u} = \mathtt{w} = \mathtt{x.f}_1^4]$. Em seguida é construída a classe $[\mathtt{x} = \mathtt{y} = \mathtt{z}]$ e, finalmente, é construído o elemento $[\mathtt{x.f}^4]$. O conjunto normalizado de classes correspondente a c$_1$ é $[[\mathtt{x.f}^4], [\mathtt{x} = \mathtt{y} = \mathtt{z}], [\mathtt{b} = \mathtt{u} = \mathtt{w} = \mathtt{x.f}_1^4]]$.

pode ser implementado pelo predicado `solve/2` definido pela cláusula

```
solve(C,M):-
      clean_ac,
      (add_constraints(C) -> check_eq_vars, undo_ac(M);
         M=false
      ).
```

onde `C` é uma lista de restrições atómicas positivas e `M` um conjunto normalizado de classes ou `false`. O predicado `clean_ac`, que será definido na secção seguinte, inicializa a `tabela de entradas` e a `trilha`, considerando o modelo activo inicial como o modelo vazio. O predicado `add_constraints/1` chama para cada restrição atómica o predicado correspondente e se algum deles falhar o predicado `add_constraints/1` falha.

Contudo, o predicado `solve` não implementa ainda o sistema de reescrita I, *i.e*, a satisfazibilidade incremental duma conjunção de restrições atómicas. Essa implementação será descrita na secção seguinte, juntamente com a implementação do sistema de reescrita $\mathsf{S}_{\mathcal{M}}$.

7.4 Implementação do Sistema de Reescrita $\mathsf{S}_{\mathcal{M}}$

O algoritmo descrito na secção anterior pode ser eficientemente generalizado para restrições complexas e implementar o sistema de reescrita completo $\mathsf{S}_{\mathcal{M}}$. No entanto, nesta secção vamos omitir, ainda, o tratamento de restrições atómicas negativas e considerar apenas disjunções e conjunções de restrições atómicas positivas. As restrições negativas serão objecto das Secções 7.4.1 e 7.5.

Para implementar a satisfazibilidade incremental, a ideia básica é marcar a `trilha` (modelo activo) antes de se considerar outra conjunção de restrições atómicas, nomeadamente quando ocorre uma disjunção. Deste modo todas as conjunções de restrições atómicas podem ser tratadas de modo similar. Após o processamento de um elemento duma disjunção o seu *factor determinístico*, se for satisfazível, é extraído como um conjunto normalizado de classes. Se não for satisfazível, o modelo parcial construído até à última marca é apagado, uma nova marca é colocada e é produzido o valor `false`.

Juntamente com a função `Undo_ac`, as funções `Clean_ac`, `Mark_ac` e `Fail_ac` (Figura 7.7) implementam este processo. Para tal, é usada uma `pilha de pontos de escolha` que é representada por `ChoicePointBase[MAXCHOICE]`. Cada `ponto de escolha` é um apontador para a trilha que corresponde ao início dum modelo parcial. Para colocar um `ponto de escolha`, o apontador para o início actual da `trilha` activa, indicado por `TrailOld`, é colocado no topo da `pilha de pontos de escolha` e o novo início da `trilha` activa passa a ser o actual topo da `trilha`. Esta operação é efectuada pela função `Mark_ac`. A operação inversa é efectuada quando o modelo é extraído. A função `Undo_ac` extrai um modelo entre os limites activos da trilha, o topo da `trilha` passa a ser o ínicio do modelo activo anterior (`TrailOld`) e o valor guardado no topo da `pilha de pontos de escolha`, se existir, passa a ser o início do novo modelo activo. Se ocorrer uma falha a função `Fail_ac` "apaga" o modelo activo e restabelece o início do novo modelo activo. Em todas estas funções, a `trilha` de functores é manipulada de modo análogo à `trilha` de entradas.

Os predicados Prolog correspondentes às funções descritas no parágrafo anterior e que permitem manipular os pontos de escolha são:

`clean_ac/0` inicializa o modelo activo com a trilha vazia

`mark_ac/0` coloca um ponto de escolha no modelo activo

`undo_ac/1` constrói uma forma normal considerando o modelo activo entre o último ponto de escolha e o topo da trilha.

`fail_ac/0` restabelece o último modelo activo

Usando estes predicados, o sistema de reescrita $\mathsf{S}_{\mathcal{M}}$ é facilmente implementado em Prolog.

Uma conjunção de restrições atómicas positivas é representada por uma lista de listas de termos Prolog, cada uma correspondendo à igualdade dos seus elementos ou, se a lista tiver um só elemento, a uma restrição do tipo $x.f^n$. Os termos podem ser (como na página 215) das seguintes formas: `atom(v)`, `var(v)`, `feat(v,f,n,i)` ou `func(v,f,n)`.

Os termos `and/2` e `or/2` representam, respectivamente, conjunções e disjunções de restrições de $\mathcal{LU}$. Na Figura 7.8 apresentamos a implementação parcial do sistema de reescrita $\mathsf{S}_{\mathcal{M}}$.

O predicado `rewrite_atomic/2` procura restrições atómicas numa conjunção e usa o predicado `add_constraints/1` para as adicionar ao modelo parcial, implementando as seis primeiras regras do sistema de reescrita SP, ou equivalentemente, as regras $\mathsf{S6}$ e $\mathsf{S7}$ do sistema de reescrita $\mathsf{S}_{\mathcal{M}}$.

O predicado `rewrite_m/2` implementa as restantes regras usando os predicados para manipular os pontos de escolha. Os predicados `and/3` e `or/3` constroem os termos correspondentes a cada uma das conectivas e simplificam-os utilizando as propriedades booleanas. O predicado `rewrite_dist/2` implementa a regra de reescrita $\mathsf{S8}$, testando a $\mathcal{M}$-independência das restrições. Antes da aplicação desta regra, são factorizadas as conjunções de disjunções utilizando-se os sistemas de reescrita descritos no Capítulo 6, Secção 6.4. Na implementação actual todos estes predicados estão escritos em Prolog.

Na Figura 7.9 ilustramos o funcionamento da `pilha de pontos de escolha` e da `trilha` durante o processamento da restrição

$$(\mathsf{c}_2) \qquad x = y \ \wedge \ ((x = a \ \wedge \ ((z = c \ \wedge \ x = b) \ \vee \ (w = b \ \wedge \ t = c))) \ \vee \\ (y = b \ \wedge \ (x = z \ \vee \ x = a)))$$

Na figura, as setas indicam os pontos de escolha que são colocados para os elementos das disjunções. No primeiro elemento mais interno, quando falha a adição da restrição $x = b$, a restrição resultante é $\perp$. No segundo elemento, o modelo extraído é $w = b \ \wedge \ t = c$. Da disjunção destes dois elementos e da extracção do modelo até ao próximo ponto de escolha, resulta a restrição $x = a \ \wedge \ w = b \ \wedge \ t = c$. Analogamente, processa-se o outro elemento da disjunção resultando $z = x \ \wedge \ x = b$. Finalmente, é extraído o modelo $y = x$ obtendo-se a restrição final

$$(\mathsf{c}_3) \qquad y = x \ \wedge \ ((x = a \ \wedge \ \ \wedge w = b \ \wedge \ t = c) \ \vee \ (z = x \ \wedge \ x = b))$$

```
Clean_ac()
{   SlotsChain = NIL;
    if(TrailBase==0)
       TrailBase = (Slot *) malloc(sizeof(Slot)*TRAIL_SIZE);
    if(FTrailBase==0)
       FTrailBase = (Slot *) malloc(sizeof(Slot)*TRAIL_SIZE);
    AuxTrailPtr = TrailBase + TRAIL_SIZE - 1;
    TrailPtr = TrailOld = TrailBase;
    ChoicePtr = ChoicePointBase ;
    return 1;
}

Mark_ac()
{   *ChoicePtr++ = TrailOld;
    TrailOld = TrailPtr;
    *ChoicePtr++ = FTrailOld;
    FTrailOld = FTrailPtr;
    return 1;
}

Fail_ac()
{   while(FTrailPtr!=FTrailOld) {
       Slot s;
       if(s=(*--FTrailPtr)) s->base= NIL;
    }
    FTrailOld=*--ChoicePtr;
    while(TrailPtr!=TrailOld) (*--TrailPtr)->value = NIL;
    TrailOld = *--ChoicePtr;
    return 1;
}
```

Figura 7.7: Colocar e retirar pontos de escolha

```
solve(C,C1):-
       clean_ac,
       rewrite(C,C0),
       (C0==C-> C1=C0; solve(C0,C1)).
rewrite(C,C3):-
       rewrite_atomic(C,C0),
       check_eq_vars,
       rewrite_m(C0,C1),
       rewrite_dist(C1,C2),
       undo_ac(M), and(M,C2,C3).
rewrite_atomic(and(and(B,C),A),C1):- !,
       rewrite_atomic(and(B,and(C,A)),C1).
rewrite_atomic(and(A,B),R) :- !, rewrite_atomic(A,R1),
       (R1=false -> R=false; rewrite_atomic(B,R2), and(R1,R2,R)).
rewrite_atomic(C,R) :- conj_atomic(C), !,
       (add_constraints(C) -> R=true; fail_ac, mark_ac, R=false).
rewrite_atomic(A,A).
rewrite_m(and(A,B),C):-
       rewrite_m(Ap,A1),
       rewrite_m(B,B1).
       and(A1,B1,C).
rewrite_m(or(or(A,B),C),R):- !,
       rewrite_m(or(A,or(B,C)),R).
rewrite_m(or(A,B),C):-
       mark_ac,
       rewrite(A,A1),
       rewrite_tail_or(B,B1),
       or(A1,B1,C).
rew_tail_or(or(A,B),R):- !,
       mark_ac,
       rewrite(A,A1),
       (A1==true -> R=true; rew_tail_or(B,B1), or(A1,B1,R)).
rew_tail_or(A,R):-
       mark_ac,
       rewrite(A,R).
```

Figura 7.8: Implementação do sistema de reescrita $\mathsf{S}_{\mathcal{M}}$

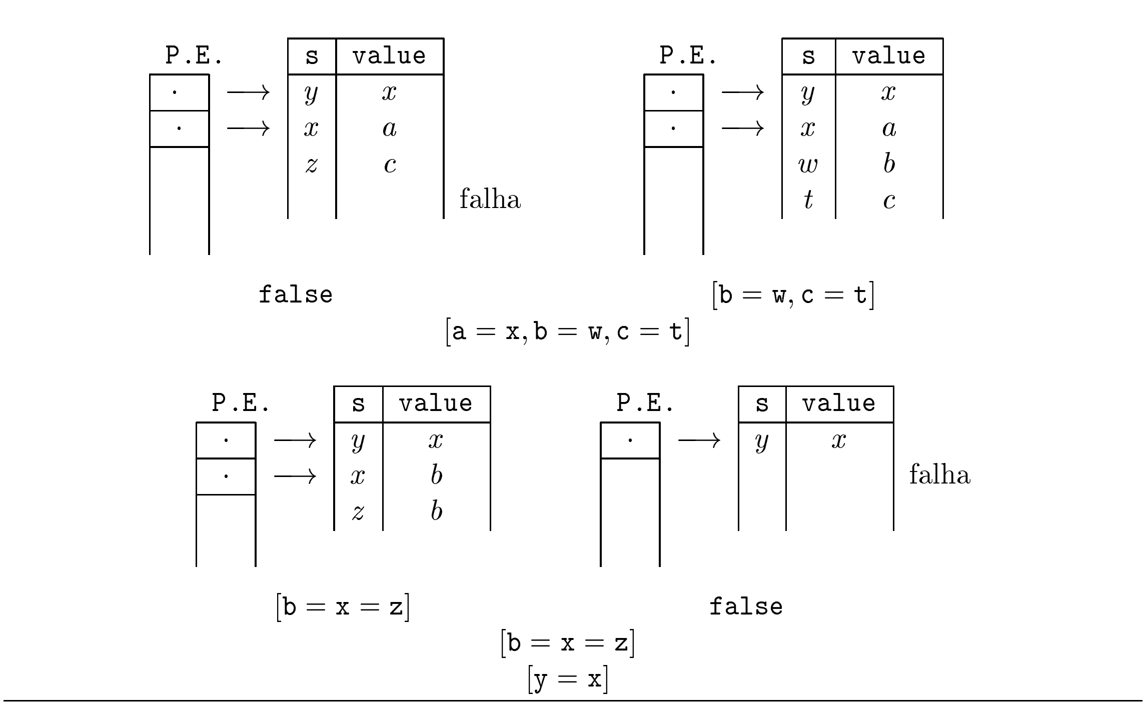


Figura 7.9: Uso de pontos de escolha

7.4.1 Implementação de Restrições Negativas

Para a implementação do sistema de reescrita P, ou mais propriamente IP, apenas faltam considerar as regras $\mathsf{P8}$–$\mathsf{P15}$. Todas elas se implementam facilmente utilizando a informação na `trilha`.

Os predicados Prolog que permitem processar as restrições atómicas negativas são:

`rew_ac_nva/3` processa uma restrição da forma $x \neq a$

`rew_ac_nvv/3` processa uma restrição da forma $x \neq y$

`rew_ac_nf/4` processa uma restrição da forma $\neg x.f^n$

Os primeiros argumentos destes predicados são os argumentos das desigualdades ou a variável, o functor e a aridade, para uma restrição do tipo $\neg x.f^n$. O último argumento destes predicados é unificado com `false` se a restrição não for satisfazível no modelo activo, com `true` se for vinculada pelo modelo activo ou com a própria restrição, com os argumentos desreferenciados, nos restantes casos.

Por exemplo, para o objectivo Prolog

```
rew_ac_nva(x,a,C).
```

a variável x é desreferenciada e

- se é uma constante distinta de a, C é unificada com true;
- se for a constante a ou uma variável funcional, C é unificada com false;
- nos restantes casos, C corresponde a $x \neq a$, com a variável x desreferenciada.

Para o predicado rew_ac_nvv/3 é ainda necessário verificar se x e y referem a mesma variável.

O objectivo Prolog

```
rew_ac_nf(x,f,n,C).
```

unifica C com true se x refere uma constante ou uma variável funcional com functor distinto de f^n. Se o functor é f^n, C unifica com false.

Para obter a implementação completa do sistema de reescrita $\mathsf{S}_{\mathcal{M}}$, excluindo casos triviais, devem ser acrescentadas as seguintes cláusulas

```
rewrite_m(not([var(X),var(Y)]),C):- !, rew_ac_nvv(X,Y,C).
rewrite_m(not([atom(A),var(X)]),C):- !, rew_ac_nva(X,A,C).
rewrite_m(not(func(X,F,N)),C):- !, rew_ac_nf(X,F,N,C).
```

7.4.1.1 Implementação usando Pontos de Escolha

Considerando o Corolário 6.5 e, no caso da assinatura ser infinita, as restrições negativas podiam ser implementadas de modo análogo à implementação de disjunções:

```
rewrite_m(not(A),C):-
mark_ac,
rewrite(A,A1),
not(A1,C).
```

No entanto, neste caso, não são obtidas formas primárias e a unificação de entradas e a extracção do modelo não são evitadas. Por outro lado, como foi discutido na apresentação das formas primárias, a verificação da igualdade de variáveis – Check_eq_vars – seria desnecessária.

7.5 Restrições Negativas no Modelo Parcial

Para a implementação dos sistemas de reescrita $\mathsf{I^e}_{\mathcal{P}}$ e $\mathsf{S^e}_{\mathcal{P}}$, em que o modelo parcial é uma forma primária, modificamos a estrutura associada a cada entrada na `tabela de entradas` de modo permitir guardar informação negativa. Numa forma primária, essa informação indica que duas entradas são incompatíveis e esse facto não é vinculado pelas restrições positivas. Assim, a cada entrada `e` será associada uma lista de entradas que são incompatíveis com `e`.

Cada estrutura do tipo `Slot` passa a ter mais um campo, designado por `div`, do tipo `Slist` cujo valor é uma `lista de incompatíveis`. Cada elemento dessa lista contém os seguintes campos:

`svalue` é um apontador para uma entrada;

`snext` é um apontador para a entrada seguinte na lista;

`slevel` é o ponto de escolha que indica o modelo parcial em que a restrição negativa ocorreu; é necessário para a extracção do modelo.

O algoritmo para formas normais apresentado na Secção 7.3 é modificado de modo a serem adicionadas ao modelo activo as restrições atómicas negativas. Sempre que uma restrição atómica negativa não é vinculada pelo modelo já construído nem é incompatível como esse modelo, é chamada a função `DisBind` (Figura 7.10), que coloca uma entrada na `lista de incompatíveis` de outra, cujo apontador é colocado na `trilha`.

No algoritmo de unificação, é necessário verificar se duas entradas a unificar não são incompatíveis. Caso não sejam, ter-se-á de juntar as `listas de incompatíveis` de cada uma. Estas operações são feitas, respectivamente, pelas funções `Disagree` e `MergeDisag` (Figura 7.11).

Os predicados Prolog que permitem adicionar restrições atómicas negativas a um modelo parcial são:

`add_div_va/2` adiciona uma restrição da forma $x \neq a$

`add_div_vv/2` adiciona uma restrição da forma $x \neq y$

`add_div_f/3` adiciona uma restrição da forma $\neg x.f^n$

Excluindo a chamada à função `DisBind`, a implementação destes predicados é idêntica à dos predicados para restrições atómicas positivas. Apenas no predicado `add_div_f/3` a

```
DisBind(Slot v1,Slot v2)
{   Slist ld1,ld=v1->div;
    Slot s0;
    while(ld && (s0=DerefSlot(ld->svalue))!=v2)
        ld1=ld, ld=ld->snext;
    if(ld==NIL || s0!=v2) {
        Slist vl=(Slist) tmp_alloc(sizeof(*vl));
        if(ld!=v1->div) ld1->snext=vl; else v1->div=vl;
        vl->svalue = v2;
        vl->snext = ld;
        vl->slevel = TrailOld;
        *TrailPtr++= v1;
    }
}
```

Figura 7.10: Colocação de uma entrada na lista de incompatíveis de outra

entrada f^n é tratada como uma constante ou variável e adicionada, se necessário, à lista de incompatíveis da variável `x`.

As restrições negativas só devem ser adicionadas ao modelo depois de todas as restrições positivas terem sido adicionadas e a função `Check_eq_vars` ter sido chamada. Deste modo, se os predicados acima referidos sucederem é garantido que a restrição inicial é satisfazível. Contudo, na satisfazibilidade incremental para formas primárias, restrições positivas podem ser adicionadas após a existência de restrições negativas no modelo. Assim, se existirem restrições do tipo $x \neq y$, sempre que se pretenda adicionar uma restrição positiva tem de ser chamada a função `Check_eq_vars`.

Consideremos a restrição

$$(\mathsf{C4}) \qquad x.f_1^1 \doteq a \ \wedge \ x \neq y \ \wedge \ (y.f_1^1 \doteq a \ \vee \ y = a)$$

Para o modelo $\mathcal{M} \equiv x.f_1^1 \doteq a \ \wedge \ x \neq y$, a restrição $y.f_1^1 \doteq a$ não é $\mathcal{M}$-satisfazível.

No entanto, aplicando o sistema de reescrita I^{e} obtém-se a forma normal e só depois são verificadas as restrições negativas. Seguindo este processo, na implementação deste sistema de reescrita a função `Check_eq_vars` só é chamada quando todas as restrições atómicas foram adicionadas ao modelo corrente, o qual pode conter incompatibilidades. Na função `Check_eq_vars` antes de se unificar duas entradas não marcadas como distinguíveis será necessário verificar se elas não são incompatíveis, usando-se para tal a função `Disagree`. Caso sejam incompatíveis, o predicado correspondente falha e a restrição é equivalente a

```
Disagree(Slot r0,Slot r)
{   Slist r1=r0->div;
    while(r1 && DerefSlot(r1->svalue)!=r) r1=r1->snext;
    if(r1) return 1;
    r1=r->div;
    while(r1 && DerefSlot(r1->svalue)!=r0) r1=r1->snext;
    if(r1) return 1;
    return 0;
}

MergeDisag(Slot u, Slot v)
{           Slist su1,su,sv; Slot su0,sv0;
    su=u->div; sv=v->div;
    while(sv){
        sv0=DerefSlot(sv->svalue);
        while(su && (su0=DerefSlot(su->svalue))!=sv0)
            su1=su; su=su->snext;
        if(su==NIL || su0!=sv0){
            Slist s;
            s=(Slist) tmp_alloc(sizeof(*s));
            s->svalue=sv0;
            s->slevel=sv->slevel;
            s->snext=su;
            if(u->div==su) u->div=s; else su1->snext=s;
            su=s;
        }
        sv=sv->snext;
    }
}
```

Figura 7.11: Verificar e juntar listas de incompatíveis `false`. A marcação de entradas distinguíveis é feita usando uma estrutura análoga à `lista de incompatíveis`.

A restante implementação do sistema de reescrita $\mathsf{S^e}_{\mathcal{P}}$ é idêntica à implementação do sistema $\mathsf{S}_{\mathcal{M}}$.

A eliminação de redundâncias nos modelos, discutida na Secção 6.5.2, não está actualmente implementada. Apesar das redundâncias, em certos casos, com o sistema de reescrita $\mathsf{S^e}_{\mathcal{P}}$, e não com $\mathsf{S}_{\mathcal{M}}$, é possível obter a satisfazibilidade duma restrição sem a aplicação da regra $\mathsf{S^e 8}$, como é ilustrado a seguir.

Consideremos a restrição

$$(\mathsf{c}_5) \qquad (x = a \ \vee \ x = b \ \vee \ x = c) \ \wedge \ x \neq a \ \wedge \ y \neq b$$

Pelo sistema de reescrita $\mathsf{S^e}_{\mathcal{P}}$ obtém-se a restrição equivalente

$$(\mathsf{c}_6) \qquad x \neq a \ \wedge \ y \neq b \ \wedge \ (x = a \ \vee \ x = c)$$

enquanto que pelo sistema de reescrita $\mathsf{S}_{\mathcal{M}}$ obtém-se

$$(\mathsf{c}_7) \qquad (x = b \ \wedge \ y \neq b) \ \vee \ (x = c \ \wedge \ y \neq b)$$

7.6 Implementação para Árvores de Atributos

A implementação dos sistemas de reescrita $\mathsf{S}_{\mathcal{M}}$ e $\mathsf{S^e}_{\mathcal{P}}$ adaptados para as linguagens de árvores de atributos, $\mathcal{FT}$ e $\mathcal{CFT}$, é obtida fazendo pequenas modificações nas estruturas e funções definidas nas secções precedentes. Como já se disse, a primeira implementação dos resolutores foi efectuada para subconjuntos destas linguagens.

A representação de árvores de atributos é idêntica à usada para árvores racionais. As restrições do tipo xft são tratadas como restrições da forma $x.f \doteq t$. As variáveis e as constantes são representadas como para as árvores racionais. Em vez de projecções temos entradas do tipo $x.f$. Uma entrada $x.f$ é representada por uma estrutura do tipo `Slot`, que é uma sub-árvore da entrada x, cujo campo `kind` é `SlotS`, o campo `name` é o identificador do atributo f e o campo `base` é um apontador para a entrada x.

Na linguagem $\mathcal{FT}$, na estrutura `Slot` não é necessário o campo `arity` nem o valor `FuncS` para o campo `kind`. Na linguagem $\mathcal{CFT}$, o valor do campo `arity`, para uma variável, é uma lista de atributos que representa a sua *aridade*. Para uma constante, este campo tem o valor `NIL`.

Na linguagem $\mathcal{FT}$, supondo uma assinatura infinita, a função `Check_eq_vars` não é necessária pois os atributos não caracterizam unicamente um objecto. Assim, é sempre possível que duas variáveis sejam distinguíveis, mesmo que tenham os mesmos valores para os atributos em que estão definidas. Na linguagem $\mathcal{CFT}$, a função `Check_eq_vars` tem de ser alterada de maneira óbvia, para utilizar as *aridades* em vez de *functores*.

7.7 Extensões, Melhoramentos e Trabalhos Futuros

Sem abandonar o interface em linguagem Prolog será conveniente que os resolutores sejam integralmente implementados em linguagem C. Isso será essencial para uma implementação eficiente dos testes de independência e factorização de restrições. Em particular, relacionado com o problema da factorização, é o da eliminação de redundâncias no modelo para formas primárias, em que a sugestão apresentada no final da Secção 6.5.2, poderá ser implementada usando os pontos de escolha.

Pelo Teorema 2.34 do Capítulo 2, resolutores para as linguagens $\mathcal{LC}_0$ e $\mathcal{LC}$ são obtidos facilmente transformando as restrições daquelas linguagens em restrições de $\mathcal{LU}$ pela introdução de variáveis quantificadas existencialmente. Esses resolutores foram implementados em Prolog, assim como um algoritmo de decisão, apresentado no Apêndice A, para fórmulas de $\mathcal{L}$ em $\mathcal{RT}$ e em $\mathcal{T}$ [DM91].

No caso de não ocorrerem negações, as restrições com `caminhos` – linguagem $\mathcal{LC}$ – podem ser implementadas directamente sem a introdução de variáveis novas. As alterações na representação das árvores são triviais e foram implementadas. Contudo, com a presença de negações a sua utilidade é menor. Será interessante analisar as situações em que este tipo de representação é apropriada.

Se a assinatura é finita, no algoritmo para obter formas primárias, descrito na Secção 4.3.2, é necessário considerar para cada variável x a restrição $\mathsf{DCA}(x)$. Como vimos, neste caso, o problema da satisfazibilidade de conjunções de restrições atómicas é **NP**-completo e uma implementação eficiente poderá usar técnicas de satisfazibilidade de restrições em domínios finitos. Será ainda interessante explorar técnicas específicas para o domínio das árvores (racionais ou de atributos). Deste modo, será também possível obter uma implementação eficiente do algoritmo da eliminação de negações apresentado na Secção 4.3.2.1.

7.8 Sumário

Neste capítulo descrevemos o modo como pode ser implementado eficientemente o algoritmo de satisfazibilidade incremental para formas normais e que permite a implementação de resolutores de restrições complexas em álgebras de árvores. Vimos como o mecanismo básico pode ser estendido para incorporar o tratamento de restrições atómicas negativas. Com a implementação a baixo-nível destes mecanismos centrais, as restantes componentes dos resolutores podem ser facilmente implementadas em Prolog, permitindo a sua integração em sistemas de processamento de Linguagem Natural, nomeadamente no formalismo $\mathcal{CCLG}$ que apresentaremos no capítulo seguinte.

Capítulo 8

O *Calculus* Λ_{LR}

Neste capítulo apresentamos uma linguagem lógica de ordem superior que é uma combinação dum λ-*calculus* tipificado (simples) e uma linguagem de restrições. Consideraremos, inicialmente, a linguagem de restrições $\mathcal{LU}$ interpretada em $\mathcal{RT}$, mas o *Calculus* que iremos definir generaliza-se para a linguagem $\mathcal{LC}$, interpretada em $\mathcal{RT}$ ou $\mathcal{T}$, ou subconjuntos das linguagens de árvores de atributos $\mathcal{FT}$ ou $\mathcal{CFT}$, apresentadas no Capítulo 3.

A motivação para a introdução desta linguagem, que designaremos genericamente[1] por Λ_{LR}, é a inadequação dos formalismos linguísticos baseados em unificação, para o tratamento da semântica da coordenação ou do âmbito da quantificação. É bem conhecido da literatura [DSP91, Per90, Ste90, Kul95], que este tipo de fenómenos são correctamente interpretados se se utilizar alguma forma de λ-*calculus*. De facto, a unificação de termos de 1ª ordem (ou estruturas de atributos) apenas permite a implementação duma forma fraca de redução β. Para a construção correcta e clara das representações semânticas, para além da redução β (aplicação funcional), o uso explícito da abstracção funcional é (também) essencial [Per90, Per91].

Na Secção 8.4 será descrito um formalismo linguístico, designado por $\mathcal{CCLG}$ e baseado em gramáticas de categorias, que foi desenhado para testar a praticabilidade do *Calculus* que iremos apresentar nas próximas secções, e onde aqueles fenómenos são tratados adequadamente.

Tanto o *Calculus* Λ_{LR} como o formalismo linguístico $\mathcal{CCLG}$ foram apresentados em [DM93, DM95a], usando-se uma linguagem de descrições de estruturas de atributos. O sistema de reescrita descrito em [DM95a] é reformulado neste capítulo, de modo a simplificar a

[1] Do Inglês, *feature descriptions.*

demonstração do Teorema 8.11.

Em linguagens lógicas de ordem superior, como a que está base da linguagem de programação λProlog [NM88], a noção de termo é generalizada a λ-termos, para incluir abstracção e aplicação, e a definição de unificação é estendida a λ-termos. Para que estas lógicas sejam consistentes uma das soluções é considerar os λ-termos tipificados. Por outro lado, para que sejam base de um método de computação é necessário ainda restringir a um subconjunto destas lógicas, cujos conjuntos de fórmulas válidas seja recursivamente enumeráveis. Um tal sistema é a *Teoria Simples de Tipos* de A. Church [Chu40], que está na base do λProlog, e que será a base da linguagem que iremos apresentar. Em vez de usarmos *termos*, iremos considerar fórmulas duma linguagem de restrições (em particular, $\mathcal{LU}$) e a noção de unificação será substituída pela de satisfazibilidade.

Uma característica essencial desta linguagem é a decidibilidade do problema da satisfazibilidade e que é uma consequência da decidibilidade da linguagem de restrições e da existência de forma normal por redução β.

Esta linguagem, como as anteriormente apresentadas neste trabalho, permite a manipulação de descrições (parciais) de objectos (termos, estruturas de atributos, árvores de atributos) e não os próprios objectos. Por esse motivo, esta linguagem é designada por *Calculus* de descrições.

Na Secção 8.1 apresentamos a sintaxe e semântica da linguagem de descrições $\Lambda_{\mathcal{LU}}$ e definimos a noção de descrição satisfazível. Na Secção 8.2 é definida a noção de descrição básica e, com base nos sistemas do Capítulo 6, é definido o sistema de reescrita Λ sendo demonstrada a sua correcção, terminação e a sua completude para a satisfazibilidade de descrições básicas. Na Secção 8.3, os resultados das secções anteriores são estendidos para a linguagem $\mathcal{LC}$. Na Secção 8.4, o *Calculus* de descrições é usado na definição e implementação do formalismo $\mathcal{CCLG}$, das gramáticas de categorias com restrições.

8.1 O *Calculus* $\Lambda_{\mathcal{LU}}$

Neste capítulo vamos supor alguma familiaridade com o λ-*calculus* tipificado, embora sejam definidas as noções básicas e referida literatura onde encontrar as demonstrações dos principais resultados utilizados.

8.1.1 Sintaxe

Consideramos como tipos básicos bool para os valores lógicos e fs para os termos[2] de $\mathcal{L}$. O conjunto de *tipos de descrições* τ, τ', τ_0, ... é definido pela seguinte gramática em BNF:

$$\tau \ ::= \ \mathsf{bool} \mid \mathsf{fs} \to \tau \mid \tau \to \tau$$

Consideramos o operador $\to$ associativo à direita e uma expressão $\sigma_1 \to \sigma_2 \to \cdots \to \sigma_n$ é um tipo onde, para $1 \leq i \leq n$, σ_i é um tipo ou fs. Note-se que excluímos fs do conjunto de tipos das descrições, reflectindo-se assim o nosso empenho em processar descrições parciais de objectos e não os objectos em si. Este facto, obriga-nos a ter algum cuidado adicional na definição da linguagem $\Lambda_{\mathcal{LU}}$.

A cada expressão de $\Lambda_{\mathcal{LU}}$ será associado um tipo τ. Sejam x, x', y, ... variáveis de $\mathcal{LU}$, v, v', ... valores de $\mathcal{LU}$, c, c', c_0, ... restrições de $\mathcal{LU}$ e para cada tipo τ considere-se um conjunto de variáveis x_τ, y_τ, ... Para cada tipo τ, o conjunto de *descrições* de $\Lambda_{\mathcal{LU}}$ do tipo τ, e_τ, d_τ, e'_τ,..., e^0_τ, ... é definido pela seguinte gramática em BNF:

$$
\begin{array}{lcll}
e_\tau & ::= & \mathsf{true} & \\
 & \mid & \mathsf{false} & \\
 & \mid & x_\tau & \\
 & \mid & e_\tau \ \wedge \ e_\tau & (\mathit{conjunção}) \\
 & \mid & e_\tau \ \vee \ e_\tau & (\mathit{disjunção}) \\
 & \mid & e_{\mathsf{fs}\to\tau} v & (\mathit{aplicação}) \\
 & \mid & e_{\tau'\to\tau} e_{\tau'} & \quad '' \\
e_{\mathsf{bool}} & ::= & c & (c \in \mathcal{LU}) \\
e_{\mathsf{fs}\to\tau} & ::= & \lambda x.e_\tau & (\mathit{abstracção}) \\
e_{\tau'\to\tau} & ::= & \lambda x_{\tau'}.e_\tau & \quad ''
\end{array}
$$

onde true, false $\wedge$ e $\vee$ são definidos para cada tipo τ, o qual é omitido sempre que não haja ambiguidade. Para o tipo bool estas expressões correspondem respectivamente aos valores lógicos $\top$ e $\perp$ e conectivas lógicas $\wedge$ e $\vee$, havendo nestes casos redundância na definição[3] das descrições. Supomos que a aplicação é associativa à esquerda e a abstracção à direita, considerando $\lambda\bar{x}_{\bar{\sigma}}.e_\tau$ abreviatura de $\lambda x^1_{\sigma_1}.\ldots\lambda x^n_{\sigma_n}.e_\tau$, onde os $x^i_{\sigma_i}$ são variáveis tipificadas ou variáveis de $\mathcal{LU}$. As conectivas lógicas têm maior precedência que a abstracção e menor que a aplicação. Sempre que necessário, usaremos parêntesis para evitar ambiguidade.

[2] A designação fs corresponde a estruturas de atributos, do inglês, *feature structures*.

[3] Alternativamente, as conectivas podiam ser consideradas constantes de tipo $\mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$, generalizando-se para os restantes tipos.

A negação não é definida para descrições de ordem superior, dado que a sua semântica não é clara para a aplicação deste *Calculus* em gramáticas de Linguagem Natural, como veremos na Secção 8.4.

O lema seguinte caracteriza os tipos e as descrições de $\Lambda_{\mathcal{LU}}$ em relação ao tipo base bool.

Lema 8.1 *(i) Cada tipo τ pode ser escrito de maneira única como ou* bool *ou*

$$\sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \mathsf{bool}$$

onde $n \geq 1$ e para todo $1 \leq i \leq n$, σ_i é um tipo ou fs.

(ii) Para qualquer descrição e_τ de tipo $\tau \equiv \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow$ bool *e quaisquer descrições ou valores de $\mathcal{LU}$, caso $\sigma_i \equiv$* fs, *$e^1_{\sigma_1}, \ldots, e^n_{\sigma_n}$ tem-se que*

$$e_\tau e^1_{\sigma_1} \ldots e^n_{\sigma_n}$$

é uma descrição de tipo bool.

Dem. (i) Por indução na estrutura do tipo τ.

(ii) Por indução em n e pela definição das descrições de $\Lambda_{\mathcal{LU}}$.

2

As noções de substituição, ocorrência livre de variáveis e ocorrência de variáveis ligadas são definidas, como habitualmente, considerando λ um quantificador. Em particular, uma *descrição fechada* é uma descrição sem variáveis livres. Denotamos por $e_\tau[v/x]$ (respectivamente por $e_\tau[e'_\tau/x_\tau]$) a substituição das ocorrências livres de x (x_τ) em e_τ por v (e'_τ). As noções de reduções α e β e formas normais são generalizadas do λ-*calculus*.

As reduções α e β são os fechos reflexivos e transitivos dos sistemas de reescrita $\mapsto_\alpha$ e $\mapsto_\beta$ em $\Lambda_{\mathcal{LU}}$, definidos por:

(α_1) $\quad \lambda x.e_\tau \;\mapsto_\alpha\; \lambda y.e_\tau[y/x]$ se y não ocorre livre em e_τ

(α_2) $\quad \lambda x_{\tau'}.e_\tau \;\mapsto_\alpha\; \lambda y_{\tau'}.e_\tau[y_{\tau'}/x_{\tau'}]$ se $y_{\tau'}$ não ocorre livre em e_τ

(β_1) $\quad (\lambda x.e_\tau)v \;\mapsto_\beta\; e_\tau[v/x]$

(β_2) $\quad (\lambda x_{\tau'}.e_\tau)e_{\tau'} \;\mapsto_\beta\; e_\tau[e_{\tau'}/x_{\tau'}]$

O facto de $\mapsto_\alpha$ e $\mapsto_\beta$ serem sistemas de reescrita garante que as regras podem ser aplicadas a sub-descrições de $\Lambda_{\mathcal{LU}}$. Uma descrição e_τ está em *forma normal* β se é irredutível para o sistema de reescrita $\mapsto_\beta$. Durante a redução β, assumimos que a redução α é aplicada sempre que necessária para evitar que variáveis livres passem a ligadas. As propriedades a seguir apresentadas para $\Lambda_{\mathcal{LU}}$ resultam de propriedades análogas para o λ-*calculus* tipificado simples. Para qualquer descrição e_τ existe uma descrição e'_τ em forma normal β tal que $e_\tau \mapsto^\star_\beta e'_\tau$. Este resultado é consequência da tipificação (simples) das descrições (ver, por exemplo, [Bar92]).

Diz-se que uma relação binária $\mathcal{R}$ sobre um conjunto $\mathcal{E}$ é Church-Rosser (ou tem a propriedade de) se

$$\forall e, e_1, e_2 \in \mathcal{E} \;\; e\mathcal{R}e_1 \wedge e\mathcal{R}e_2 \;\Rightarrow\; \exists\, e_3 \in \mathcal{E} \;\; e_1\mathcal{R}e_3 \wedge e_2\mathcal{R}e_3 \tag{8.1}$$

Como a redução β tem a propriedade de Church-Rosser, e'_τ é única a menos de redução α e é denominada a *forma normal* (β) de e_τ. Mais ainda qualquer descrição é *fortemente normalizável*, *i.e*, qualquer sequência de aplicações das regras β terminam (ver, por exemplo, [Bar92, HS86]).

8.1.2 Semântica

A semântica de $\Lambda_{\mathcal{LU}}$ vai ser definida como habitualmente para o λ-*calculus* tipificado e linguagens lógicas de ordem superior [Bar84, And86].

Interpretando a linguagem $\mathcal{LU}$ em $\mathcal{RT}$, associamos a cada tipo τ um domínio semântico $\mathcal{D}_\tau$ da seguinte forma:

$$\begin{aligned} \mathcal{D}_{\mathsf{bool}} &= \{0,1\} \\ \mathcal{D}_{\mathsf{fs}\to\tau} &= \mathcal{RT} \longrightarrow \mathcal{D}_\tau \\ \mathcal{D}_{\tau'\to\tau} &= \mathcal{D}_{\tau'} \longrightarrow \mathcal{D}_\tau \end{aligned}$$

onde $A \longrightarrow B$ representa o conjunto de funções de A em B e $\{0,1\}$ representa o conjunto dos valores lógicos, munido das operações booleanas habituais. Supomos que se $f,\, g \in A \longrightarrow B$, $f = g$ se e só se $\forall x \in A,\;\; f(x) = g(x)$. Sendo $A \times B$ é o produto cartesiano de A por B, identificamos $\mathcal{D}_{\sigma_1\to\sigma_2\to\cdots\to\sigma_n\to\tau}$ com $\mathcal{D}_{\sigma_1} \times \mathcal{D}_{\sigma_2} \times \cdots \times \mathcal{D}_{\sigma_n} \longrightarrow \mathcal{D}_\tau$, onde se σ_i é fs então $\mathcal{D}_{\sigma_i}$ é $\mathcal{RT}$.

A semântica das descrições é definida de modo análogo à dos λ-termos do λ-*calculus* tipificado (simples), com extensão às conectivas e valores lógicos. Para tal notemos que as operações booleanas $\wedge$ e $\vee$ podem ser estendidas a todos os domínios semânticos, componente a componente, definindo-se genericamente:

$$(8.2) \qquad \begin{array}{rcl} (\boldsymbol{\lambda} s.d) \ \wedge \ (\boldsymbol{\lambda} s.d') & \stackrel{def}{=} & (\boldsymbol{\lambda} s.d \ \wedge \ d') \\ (\boldsymbol{\lambda} s.d) \ \vee \ (\boldsymbol{\lambda} s.d') & \stackrel{def}{=} & (\boldsymbol{\lambda} s.d \ \vee \ d') \end{array}$$

onde $\boldsymbol{\lambda} s$ denota uma função nos domínios semânticos.

Sejam $\bigwedge_{\mathsf{bool}}$ e $\bigvee_{\mathsf{bool}}$ são as funções binárias[4] booleanas correspondentes à conjunção e disjunção, respectivamente. Podemos definir indutivamente no tipo τ as funções binárias $\bigwedge_\tau \in \mathcal{D}_\tau \longrightarrow \mathcal{D}_\tau \longrightarrow \mathcal{D}_\tau$ por

$$\begin{array}{lcll} d \ \bigwedge_{\mathsf{bool}} \ d' & = & \begin{cases} 1 & \text{se } d = 1 \text{ e } d' = 1 \\ 0 & \text{caso contrário} \end{cases} & \\ (\boldsymbol{\lambda} s.d) \ \bigwedge_{\mathsf{fs} \rightarrow \tau} (\boldsymbol{\lambda} s.d') & = & \boldsymbol{\lambda} s.d \ \bigwedge_\tau \ d' & (s \ \in \ \mathcal{RT}) \\ (\boldsymbol{\lambda} s.d) \ \bigwedge_{\tau' \rightarrow \tau} (\boldsymbol{\lambda} s.d') & = & \boldsymbol{\lambda} s.d \ \bigwedge_\tau \ d' & (s \ \in \ \mathcal{D}_{\tau'}) \end{array}$$

onde $(s \in \mathcal{D})$ significa que s é um elemento de $\mathcal{D}$. De modo análogo definimos as funções $\bigvee_\tau$.

Para qualquer tipo $\tau \equiv \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \mathsf{bool}$, $s_i \in \mathcal{D}_{\sigma_i}$ e d, $d' \in \mathcal{D}_\tau$ temos que

$$(d \bigwedge_{\sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \mathsf{bool}} d')(s_1, \ldots, s_n) = d(s_1, \ldots, s_n) \bigwedge_{\mathsf{bool}} d'(s_1, \ldots, s_n)$$

e analogamente para $\bigvee_{\sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \mathsf{bool}}$.

Do mesmo modo, podemos definir funções nos domínios semânticos para os valores lógicos true e false.

Definimos 1_τ indutivamente no tipo τ por:

$$\begin{array}{lcll} 1_{\mathsf{bool}} & = & 1 & (1 \ \in \ \{0,1\}) \\ 1_{\mathsf{fs} \rightarrow \tau} & = & \boldsymbol{\lambda} s.1_\tau & (s \ \in \ \mathcal{RT}) \\ 1_{\tau' \rightarrow \tau} & = & \boldsymbol{\lambda} s.1_\tau & (s \ \in \ \mathcal{D}_{\tau'}) \end{array}$$

e, de modo análogo, definimos 0_τ.

Para qualquer tipo $\tau \equiv \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \mathsf{bool}$ temos que

$$1_\tau = \boldsymbol{\lambda} s_1. \ldots \boldsymbol{\lambda} s_n.1$$

e analogamente para 0_τ.

Estamos agora em condições de definir a semântica das descrições de $\Lambda_{\mathcal{LU}}$. Seja ρ uma atribuição de variáveis, tal que $\rho(x) \in \mathcal{RT}$ e $\rho(x_\tau) \in \mathcal{D}_\tau$, para cada tipo τ. Como usualmente,

[4] Usando notação infixa.

$\rho(v) = v$ se v é uma constante de $\mathcal{LU}$ e $\rho[d/x]$ ($\rho[d_\tau/x_\tau]$) denota a função obtida de ρ atribuindo d (d_τ) a x (x_τ). Para qualquer atribuição de variáveis ρ, a semântica das descrições de $\Lambda_{\mathcal{LU}}$ é definida indutivamente na estrutura das descrições por:

$$
\begin{array}{lcll}
[\![x_\tau]\!]\rho & = & \rho(x_\tau) & \\
[\![c]\!]\rho & = & \begin{cases} 1 & \text{se } \mathcal{RT} \models_\rho c \\ 0 & \text{caso contrário} \end{cases} & \\
[\![\lambda x.e_\tau]\!]\rho & = & \boldsymbol{\lambda} s.[\![e_\tau]\!]\rho[s/x] & (s \in \mathcal{RT}) \\
[\![\lambda x_{\tau'}.e_\tau]\!]\rho & = & \boldsymbol{\lambda} s.[\![e_\tau]\!]\rho[s/x_{\tau'}] & (s \in \mathcal{D}_{\tau'}) \\
[\![e_{\mathsf{fs}\to\tau} v]\!]\rho & = & ([\![e_{\mathsf{fs}\to\tau}]\!]\rho)\rho(v) & \\
[\![e_{\tau'\to\tau} e_{\tau'}]\!]\rho & = & ([\![e_{\tau'\to\tau}]\!]\rho)([\![e_{\tau'}]\!]\rho) & \\
[\![e_\tau \wedge e'_\tau]\!]\rho & = & [\![e_\tau]\!]\rho \bigwedge_\tau [\![e'_\tau]\!]\rho & \\
[\![e_\tau \vee e'_\tau]\!]\rho & = & [\![e_\tau]\!]\rho \bigvee_\tau [\![e'_\tau]\!]\rho & \\
[\![\mathsf{true}_\tau]\!]\rho & = & 1_\tau & \\
[\![\mathsf{false}_\tau]\!]\rho & = & 0_\tau &
\end{array}
$$

Definição 8.2 *Para qualquer tipo τ, duas descrições e_τ e e'_τ* são equivalentes *se para qualquer atribuição de variáveis ρ tem-se que $[\![e_\tau]\!]\rho = [\![e'_\tau]\!]\rho$ e escreve-se $e_\tau \approx e'_\tau$. Uma descrição e_τ é* satisfazível *se não é equivalente* a false.

O *Calculus* $\Lambda_{\mathcal{LU}}$ é consistente no sentido em que true não é equivalente a false.

A comutatividade e a associatividade da conjunção e disjunção preservam a equivalência de descrições em $\Lambda_{\mathcal{LU}}$. A relação $\equiv_{\mathcal{B}}$, definida para $\mathcal{LU}$ na página 95, generaliza-se para $\Lambda_{\mathcal{LU}}$ e, portanto, podemos considerar descrições do tipo τ a menos da relação $\equiv_{\mathcal{B}}$.

Para qualquer tipo τ e descrições e_τ, e'_τ, e e''_τ, verificam-se as propriedades booleanas:

$$
\begin{array}{rcl}
e_\tau \wedge \mathsf{true} & \approx & e_\tau \\
e_\tau \wedge \mathsf{false} & \approx & \mathsf{false} \\
e_\tau \vee \mathsf{false} & \approx & e_\tau \\
e_\tau \vee \mathsf{true} & \approx & \mathsf{true} \\
(e_\tau \wedge e'_\tau) \vee e''_\tau & \approx & (e_\tau \vee e''_\tau) \wedge (e'_\tau \vee e''_\tau) \\
(e_\tau \vee e'_\tau) \wedge e''_\tau & \approx & (e_\tau \wedge e''_\tau) \vee (e'_\tau \wedge e''_\tau)
\end{array}
\tag{8.3}
$$

É verdade também que

$$
\begin{array}{rcl}
\mathsf{true}_{\mathsf{bool}} & \approx & \top \\
\mathsf{true}_{\mathsf{fs}\to\tau} & \approx & \lambda x.\mathsf{true}_\tau \\
\mathsf{true}_{\tau'\to\tau} & \approx & \lambda x_{\tau'}.\mathsf{true}_\tau
\end{array}
\tag{8.4}
$$

e, analogamente, para false.

É ainda uma consequência da semântica do λ-*calculus* tipificado que se $e_\tau \mapsto_\beta e'_\tau$ então e_τ e e'_τ são equivalentes (ver, por exemplo, [Sto77]). Em particular, se e'_τ é a forma normal β de e_τ, as duas descrições são equivalentes.

Na secção seguinte iremos ver que é sempre possível decidir se uma descrição fechada é equivalente a false. Esta propriedade é essencialmente uma generalização do problema da satisfazibilidade para a álgebra das árvores racionais.

O seguinte lema irá ser usado para relacionar descrições fechadas e satisfazibilidade em $\Lambda_{\mathcal{LU}}$.

Lema 8.3 *Uma descrição e_τ é satisfazível se e só se o seu fecho $\lambda\bar{x}_{\bar{\sigma}}.e_\tau$ é satisfazível, onde* $\mathsf{Vars}(e_\tau) = \{x^1_{\sigma_1}, \ldots, x^n_{\sigma_n}\}$.

Dem. ($\Rightarrow$)

Suponhamos que para a atribuição de variáveis ρ_1, $[\![e_\tau]\!]\rho_1 \neq 0_\tau$ e $\rho_1(\bar{x}_{\bar{\sigma}}) = \bar{r}$. Então, $[\![\lambda\bar{x}_{\bar{\sigma}}.e_\tau]\!]\rho_1 = \boldsymbol{\lambda}\bar{s}.[\![e_\tau]\!]\rho_1[\bar{s}/\bar{x}_{\bar{\sigma}}]$. Para $\bar{s} = \bar{r}$ temos que $[\![e_\tau]\!]\rho_1[\bar{s}/\bar{x}_{\bar{\sigma}}] = [\![e_\tau]\!]\rho_1 \neq 0_\tau$. Logo $[\![\lambda\bar{x}_{\bar{\sigma}}.e_\tau]\!]\rho_1 \neq 0_{\bar{\sigma}\to\tau}$.

($\Leftarrow$)

Se $[\![\lambda\bar{x}_{\bar{\sigma}}.e_\tau]\!]\rho = \boldsymbol{\lambda}\bar{s}.[\![e_\tau]\!]\rho[\bar{s}/\bar{x}_{\bar{\sigma}}] \neq 0_{\bar{\sigma}\to\tau}$ então existe $\bar{r}$ tal que $[\![e_\tau]\!]\rho[\bar{r}/\bar{x}_{\bar{\sigma}}] \neq 0_\tau$. 2

8.2 Um Resolutor para $\Lambda_{\mathcal{LU}}$

A relação $\equiv_{\mathcal{B}}$ em $\Lambda_{\mathcal{LU}}$ permite-nos considerar descrições em que as conjunções e disjunções são associativas à direita e com as abstracções no "início". Como para as restrições em $\mathcal{LU}$, qualquer descrição e_τ pode ser transformada em tempo polinomial numa descrição equivalente dessa forma. Para tal basta modificar o sistema de reescrita G, substituindo as restrições atómicas por abstracções e acrescentar regras para a conectiva $\vee$.

Nas regras de reescrita que se seguem, a conectiva Θ designa ou $\wedge$ ou $\vee$.

$\mathsf{G}_\Lambda 1.$	$(e^1_\tau \;\Theta\; e^2_\tau) \;\Theta\; e^3_\tau$	$\mapsto$	$e^1_\tau \;\Theta\; (e^2_\tau \;\Theta\; e^3_\tau)$
$\mathsf{G}_\Lambda 2.$	$e^1_\tau \;\Theta\; e^2_\tau$	$\mapsto$	$e^2_\tau \;\Theta\; e^1_\tau$ se e^2_τ é uma abstracção e e^1_τ não é uma abstracção;
$\mathsf{G}_\Lambda 3.$	$e^1_\tau \;\Theta\; (e^2_\tau \;\Theta\; e^3_\tau)$	$\mapsto$	$e^2_\tau \;\Theta\; (e^1_\tau \;\Theta\; e^3_\tau)$

se e^2_τ é uma abstracção e e^1_τ não é uma abstracção.

Começamos por definir uma redução em $\Lambda_{\mathcal{LU}}$ que preserva a equivalência das descrições e traduz a operacionalidade das conectivas e valores lógicos. Como acima, a conectiva Θ designa ou $\wedge$ ou $\vee$.

A redução δ é o fecho reflexivo e transitivo do sistema de reescrita $\mapsto_\delta$ em $\Lambda_{\mathcal{LU}}$ definido por:

(δ_1) $\quad \mathsf{true}_{\mathsf{fs}\to\tau} v \ \mapsto_\delta \ \mathsf{true}_\tau$

(δ_2) $\quad \mathsf{true}_{\tau'\to\tau} e_{\tau'} \ \mapsto_\delta \ \mathsf{true}_\tau$

(δ_3) $\quad \mathsf{false}_{\mathsf{fs}\to\tau} v \ \mapsto_\delta \ \mathsf{false}_\tau$

(δ_4) $\quad \mathsf{false}_{\tau'\to\tau} e_{\tau'} \ \mapsto_\delta \ \mathsf{false}_\tau$

(δ_5) $\quad (\lambda x.e_\tau) \ \Theta \ (\lambda x.e'_\tau) \ \mapsto_\delta \ \lambda x.e_\tau \ \Theta \ e'_\tau$

(δ_6) $\quad (\lambda x_{\tau'}.e_\tau) \ \Theta \ (\lambda x_{\tau'}.e'_\tau) \ \mapsto_\delta \ \lambda x_{\tau'}.e_\tau \ \Theta \ e'_\tau$

(δ_7) $\quad (e''_{\mathsf{fs}\to\tau} \ \Theta \ e_{\mathsf{fs}\to\tau}) v \ \mapsto_\delta \ e''_{\mathsf{fs}\to\tau} v \ \Theta \ e_{\mathsf{fs}\to\tau} v$ se $e''_{\mathsf{fs}\to\tau}$ ou $e_{\mathsf{fs}\to\tau}$ não é uma abstracção

(δ_8) $\quad (e''_{\tau'\to\tau} \ \Theta \ e'_{\tau'\to\tau}) e_{\tau'} \ \mapsto_\delta \ e''_{\tau'\to\tau} e_{\tau'} \ \Theta \ e'_{\tau'\to\tau} e_{\tau'}$ se $e''_{\tau'\to\tau}$ ou $e'_{\tau'\to\tau}$ não é uma abstracção

Como para a redução β, supomos que a redução α é aplicada sempre que necessário. Resulta das definições da semântica das conectivas e dos valores lógicos que a redução δ preserva a equivalência das descrições. Pelas equivalências (8.4), da página 237, as regras (δ_1)-(δ_4) são instâncias da redução β e, portanto, serão omitidas no que se segue.

Designamos por *redução $\beta\delta$* o fecho reflexivo e transitivo da reunião dos sistemas de reescrita β e δ. A noção de forma normal generaliza-se para forma normal $\beta\delta$ (a menos das relações α e $\equiv_{\mathcal{B}}$).

Proposição 8.4

1. *Para todo tipo τ, uma descrição $e_\tau \in \Lambda_{\mathcal{LU}}$ admite forma normal $\beta\delta$ e se e'_τ é essa forma normal, então $e_\tau \approx e'_\tau$.*

2. *A redução $\beta\delta$ satisfaz a propriedade de Church-Rosser.*

3. *Qualquer descrição e_τ é fortemente normalizável em relação a $\beta\delta$.*

As demonstrações da proposição anterior são análogas às dos correspondentes resultados para a redução β no λ-*calculus* tipificado simples, e são apresentadas no Apêndice B.

Lema 8.5 *O conjunto das descrições em forma normal $\beta\delta$ é o menor conjunto tal que*

(i) *$c \in \mathcal{LU}$ é uma forma normal $\beta\delta$ de tipo* bool*;*

(ii) true, false *e x_τ são formas normais $\beta\delta$;*

(iii) *se $e^1_{\sigma_1}, \ldots, e^n_{\sigma_n}$ são formas normais $\beta\delta$ ou valores de $\mathcal{LU}$ e $\tau' \equiv \sigma_1 \to \cdots \to \sigma_n \to \tau$, então $x_{\tau'} e^1_{\sigma_1} \ldots e^n_{\sigma_n}$ é uma forma normal $\beta\delta$ de tipo τ;*

(iv) *Se e^1_τ e e^2_τ são formas normais $\beta\delta$ e não simultaneamente conjunções que contenham uma abstracção, então $e^1_\tau \wedge e^2_\tau$ é uma forma normal $\beta\delta$;*

(v) *Se e^1_τ e e^2_τ são formas normais $\beta\delta$ e não simultaneamente disjunções que contenham uma abstracção, então $e^1_\tau \vee e^2_\tau$ é uma forma normal $\beta\delta$;*

(vi) *Se e_τ é forma normal $\beta\delta$, então $\lambda x_{\tau'}.e_\tau$ e $\lambda x.e_\tau$ são formas normais $\beta\delta$.*

Dem. Se e_τ verifica (i)-(vi) então é irredutível para $\beta\delta$. Se e_τ é irredutível para $\beta\delta$ então, mostra-se por indução na estrutura de e_τ que e_τ satisfaz (i)-(vi). 2

Definição 8.6 *Uma descrição fechada $e_\tau \in \Lambda_{\mathcal{LU}}$ em forma normal $\beta\delta$ designa-se por* básica.

No que se segue vamos considerar essencialmente descrições *básicas.*

O resolutor para descrições básicas que vamos apresentar, generaliza os resolutores para $\mathcal{LU}$, descritos no Capítulo 6. Iremos tomar como base o sistema de reescrita $\mathsf{S}_{\mathcal{M}}$, mas as factorizações e o sistema de reescrita $\mathsf{S^e}_{\mathcal{P}}$ podem ser igualmente generalizados.

A cada restrição $c \in \mathcal{LU}$, com $\mathsf{Vars}(c) = \{x_1, \ldots, x_n\}$, associamos o seu *fecho*, *i.e*, a descrição básica $\lambda x_1. \ldots \lambda x_n.c$.

Se $\mathsf{Vars}(c) = \emptyset$ consideramos apenas a descrição c de tipo bool.

Proposição 8.7 *Seja $e_\tau \equiv \lambda x_1. \ldots \lambda x_n.c \in \Lambda_{\mathcal{LU}}$, com $\tau \equiv \mathsf{fs} \to \cdots \to \mathsf{fs} \to \mathsf{bool}$, $c \in \mathcal{LU}$ e $\mathsf{Vars}(c) = \{x_1, \ldots, x_n\}$. A descrição e_τ é satisfazível se e só se $c \overset{\mathsf{S}}{\longmapsto}{}^{\star}_{\top} c'$, c' é irredutível para o sistema de reescrita S e $c' \not\equiv \bot$.*

Dem. Pela completude do sistema de reescrita S e pela semântica das descrições. 2

Na Tabela 8.6 é apresentado o sistema de reescrita Λ, que é uma generalização para $\Lambda_{\mathcal{LU}}$ do sistema de reescrita S. Dada uma descrição básica e_τ, o sistema de reescrita Λ produz uma descrição e'_τ tal que $e_\tau \approx e'_\tau$ e $e'_\tau \equiv \mathsf{false}$ se e só se e_τ é não satisfazível. Para descrições do tipo e_{bool}, a noção de modelo parcial é a mesma que para $\mathcal{LU}$. Supondo inicialmente que $\mathcal{M} \equiv \top$, o sistema de reescrita $\mathsf{S}_\mathcal{M}$ para descrições do tipo bool coincide com o de $\mathcal{LU}$, uma vez que para garantir a satisfazibilidade ou a $\mathcal{M}$-independência não é necessário reescrever descrições da forma $x_{\tau'} e^1_{\tau_1} \dots e^n_{\tau_n}$, com $\tau' \equiv \tau_1 \to \cdots \to \tau_n \to \mathsf{bool}$. Descrições dessa forma são sempre satisfazíveis independentemente da semântica dos argumentos $e^i_{\tau_i}$. Pelo mesmo motivo, para qualquer tipo, no sistema de reescrita Λ não se reescrevem descrições da forma $x_{\tau'} e^1_{\tau_1} \dots e^n_{\tau_n}$. Nestes casos, dizemos que os argumentos $e^i_{\tau_i}$ estão *protegidos*. Comportamento diferente será considerado, na Secção 8.3, para a semântica da aplicação de valores na linguagem $\mathcal{LC}$. Estabelecendo um paralelo com a semântica da passagem de argumentos em funções ou procedimentos em linguagens de programação, podemos dizer que a semântica definida para $\Lambda_{\mathcal{LU}}$ é com "chamada por nome" para descrições de ordem superior e será com "chamada por valor" para valores, *i.e*, termos de $\mathcal{L}$. Tendo em conta estas considerações, cada regra de reescrita de Λ pode ser aplicada a uma sub-descrição, que satisfaça as condições de aplicabilidade, apresentadas na Tabela 8.5.

Salvo indicação em contrário, vamos ainda supor que para um sistema de reescrita $\mapsto$, se $c \mapsto^\star c'$, então c' é irredutível para esse sistema de reescrita. Como no sistema de reescrita S, a aplicação da regra $\Lambda 10$ é retardada, procurando-se diminuir o tamanho das descrições a que se aplique. Em particular, se as descrições forem independentes não será necessária.

$$\frac{e_\tau \overset{\Lambda}{\longmapsto} e'_\tau}{\lambda x.e_\tau \overset{\Lambda}{\longmapsto} \lambda x.e'_\tau} \qquad \frac{e_\tau \overset{\Lambda}{\longmapsto} e'_\tau}{\lambda x_{\tau'}.e_\tau \overset{\Lambda}{\longmapsto} \lambda x_{\tau'}.e'_\tau}$$

$$\frac{e_\tau \overset{\Lambda}{\longmapsto} e'_\tau}{e_\tau \wedge e''_\tau \overset{\Lambda}{\longmapsto} e'_\tau \wedge e''_\tau} \qquad \frac{e_\tau \overset{\Lambda}{\longmapsto} e'_\tau}{e_\tau \vee e''_\tau \overset{\Lambda}{\longmapsto} e'_\tau \vee e''_\tau}$$

Tabela 8.5: Aplicabilidade do sistema de reescrita Λ

Como habitualmente,

Definição 8.8 *Duas descrições e_τ e e'_τ são independentes se a satisfazibilidade de e_τ e e'_τ implica a satisfazibilidade de $e_\tau \wedge e'_\tau$.*

$\Lambda 1.$	$\lambda x.\mathsf{false}_\tau$	$\longmapsto$	$\mathsf{false}_{\mathsf{fs}\to\tau}$
$\Lambda 2.$	$\lambda x_{\tau'}.\mathsf{false}_\tau$	$\longmapsto$	$\mathsf{false}_{\tau'\to\tau}$
$\Lambda 3.$	$\lambda x.\mathsf{true}_\tau$	$\longmapsto$	$\mathsf{true}_{\mathsf{fs}\to\tau}$
$\Lambda 4.$	$\lambda x_{\tau'}.\mathsf{true}_\tau$	$\longmapsto$	$\mathsf{true}_{\tau'\to\tau}$
$\Lambda 5.$	$e_\tau \wedge \mathsf{true}_\tau$	$\longmapsto$	e_τ
$\Lambda 6.$	$e_\tau \wedge \mathsf{false}_\tau$	$\longmapsto$	false_τ
$\Lambda 7.$	$e_\tau \vee \mathsf{true}_\tau$	$\longmapsto$	true_τ
$\Lambda 8.$	$e_\tau \vee \mathsf{false}_\tau$	$\longmapsto$	e_τ
$\Lambda 9.$	e_{bool}	$\longmapsto$	e'_{bool} se $e_{\mathsf{bool}} \overset{\mathsf{S}}{\longmapsto}{}^{\star}_{\top}\ e'_{\mathsf{bool}}$ e $e_{\mathsf{bool}} \not\equiv e'_{\mathsf{bool}}$
$\Lambda 10.$	$(e_\tau \vee e'_\tau) \wedge e''_\tau$	$\longmapsto$	$(e_\tau \wedge e''_\tau) \vee (e'_\tau \wedge e''_\tau)$ se nenhuma das regras anteriores se aplica, $\tau \not\equiv \mathsf{bool}$ e nem e_τ nem e'_τ são independentes com e''_τ; a descrição resultante é reduzida a forma normal $\beta\delta$.

Tabela 8.6: Sistema de reescrita Λ

Lema 8.9 *Para qualquer descrição $e_\tau \equiv x_{\tau'} e^1_{\sigma_1} \ldots e^n_{\sigma_n}$, onde $e^i_{\sigma_i}$ são formas normais $\beta\delta$ ou valores de $\mathcal{LU}$ e $\tau' \equiv \sigma_1 \to \cdots \to \sigma_n \to \tau$, tem-se que*

(i) e_τ é satisfazível.

(ii) qualquer descrição e'_τ em forma normal $\beta\delta$ é independente de e_τ.

Dem. (i) Para qualquer atribuição de variáveis ρ, temos que

$$[\![x_{\tau'} e^1_{\sigma_1} \ldots e^n_{\sigma_n}]\!]\rho \quad = \quad ((([\![x_{\tau'}]\!]\rho([\![e^1_{\sigma_1}]\!]\rho))\ldots)[\![e^n_{\sigma_n}]\!]\rho)$$

Uma vez que para qualquer tipo τ', $\mathcal{D}_{\tau'}$ tem pelo menos um elemento s diferente de $0_{\tau'}$ (nomeadamente $1_{\tau'}$) se $\rho(x_{\tau'}) = s$, então $[\![e_\tau]\!]\rho \neq 0_\tau$ (para $s \equiv 1_{\tau'}$, $[\![e_\tau]\!]\rho = 1_\tau$).

(ii) Suponhamos que e'_τ é satisfazível, isto é, que existe uma atribuição de variáveis ρ tal que $[\![e'_\tau]\!]\rho \neq 0_\tau$. Por indução na estrutura de e'_τ, usando o Lema 8.5, mostra-se que $[\![e'_\tau]\!]\rho[1_{\tau'}/x_{\tau'}] \neq 0_\tau$.

2

O lema seguinte estabelece uma condição suficiente de independência.

Lema 8.10 *Duas descrições em forma normal $\beta\delta$ sem disjunções são independentes se não contêm simultaneamente abstracções.*

Dem. Resulta do Lema 8.9(ii). 2

Como no caso de $\mathcal{LU}$, uma disjunção de descrições $e_\tau \equiv e^1_\tau \,\vee\, \ldots \,\vee\, e^n_\tau$ é independente duma descrição e'_τ se e só se alguma das descrições e^i_τ o for.

Teorema 8.11 *Para qualquer descrição básica e_τ, o sistema de reescrita Λ é correcto, termina e é completo no sentido em que e_τ é satisfazível a menos que* false *seja produzido.*

Dem. Para a correcção do sistema de reescrita Λ temos que provar que se $e_\tau \stackrel{\Lambda}{\longmapsto} e'_\tau$ então $e_\tau \approx e'_\tau$. Isso é imediato pela definição da semântica das restrições, das equivalências (8.3) e (8.4) e da correcção do sistema de reescrita S. Analisemos a terminação. Uma sequência infinita de aplicações de regras teria necessariamente de envolver a regra $\Lambda10$. Como para o sistema de reescrita S, seja γ a função de $\Lambda_{\mathcal{LU}}$ em $\mathbb{N}$:

$\gamma(e_\tau) = 2$ se e_τ é uma aplicação, true, false ou $c \in \mathcal{LU}$
$\gamma(\lambda x.e_\tau) = \gamma(e_\tau)$
$\gamma(\lambda x_{\tau'}.e_\tau) = \gamma(e_\tau)$
$\gamma(e^1_\tau \,\vee\, e^2_\tau) = \gamma(e^1_\tau) + \gamma(e^2_\tau) + 1$
$\gamma(e^1_\tau \,\wedge\, e^2_\tau) = \gamma(e^1_\tau) \times \gamma(e^2_\tau)$

A função γ é estritamente decrescente por aplicação da regra $\Lambda10$ e as restantes regras não aumentam o seu valor. A redução duma descrição a forma $\beta\delta$, na aplicação da regra $\Lambda10$, também não aumenta o valor de γ uma vez que a regra β não se aplica e as únicas regras δ que se podem aplicar são (δ_5)-(δ_6).

Analisemos a completude. Se false é produzido então, pela correcção, e_τ não é satisfazível. A aplicação das regras de Λ transforma descrições básicas em descrições básicas. Seja e'_τ uma descrição básica irredutível para Λ e diferente de false. Se for true o resultado está provado. Caso contrário, e'_τ é uma descrição básica da forma $\lambda\bar{x}_{\bar{\sigma}}.e_{\tau'}$, onde $e_{\tau'}$ é uma forma normal $\beta\delta$ com variáveis livres em $\{x^1_{\sigma_1}, \ldots, x^n_{\sigma_n}\}$. Pelo Lema 8.3 e porque e'_τ é irredutível para Λ, apenas é necessário provar que $e_{\tau'}$ é satisfazível. Procedemos por indução na estrutura de $e_{\tau'}$.

Caso $e_{\tau'} \equiv c$. Neste caso, $c \not\equiv \bot$. Pela completude do sistema de reescrita S, e_τ é satisfazível.

Caso $e_{\tau'} \equiv x_{\tau''} e^1_{\sigma_1} \ldots e^n_{\sigma_n}$, onde $e^i_{\sigma_i}$ são formas normais $\beta\delta$. Pelo Lema 8.9 (ii), $e_{\tau'}$ é satisfazível.

Caso $e_{\tau'} \equiv \lambda x.e^1_{\tau_1}$ ***ou*** $e_{\tau'} \equiv \lambda x_{\tau''}.e^1_{\tau_1}$. Pela hipótese de indução e por um argumento análogo ao usado na demonstração do Lema 8.3.

Caso $e_{\tau'} \equiv e^1_{\tau'} \wedge e^2_{\tau'}$. onde $e^1_{\tau'}$ e $e^2_{\tau'}$ são formas normais $\beta\delta$ satisfazíveis e verificam a condição (iv) do Lema 8.5. Se $\tau' \equiv \mathsf{bool}$ temos o primeiro caso. Senão, considerando a relação $\equiv_{\mathcal{B}}$ temos os seguintes sub-casos:

Caso $e^1_{\tau'} \equiv x_{\tau''} e^1_{\sigma_1} \ldots e^n_{\sigma_n}$. Pelo Lema 8.9(ii), $e^1_{\tau'} \wedge e^2_{\tau'}$ é satisfazível.

Caso $e^1_{\tau'} \equiv \lambda x.e^1_{\tau''}$ ***ou*** $e^1_{\tau'} \equiv \lambda x_{\tau^1}.e^1_{\tau''}$. Se $e^2_{\tau'}$ é uma conjunção de aplicações então $e^1_{\tau'} \wedge e^2_{\tau'}$ é satisfazível, pelo Lema 8.9 (ii). Se $e^2_{\tau'}$ contém uma disjunção, como a regra $\Lambda10$ não se aplica, concluímos também que $e^1_{\tau'} \wedge e^2_{\tau'}$ é satisfazível

Caso $e^1_{\tau'} \equiv e^3_{\tau'} \vee e^4_{\tau'}$. Como a regra $\Lambda10$ não se aplica, $e^1_{\tau'} \wedge e^2_{\tau'}$ é satisfazível.

Caso $e_{\tau'} \equiv e^1_{\tau'} \vee e^2_{\tau'}$. Por hipótese de indução $e^1_{\tau'}$ e $e^2_{\tau'}$ são satisfazíveis e como $e_{\tau'}$ é irredutível para Λ, pela definição de $\bigvee_\tau$, $e_{\tau'}$ é satisfazível.

$\Box$

8.3 O *Calculus* $\Lambda_{\mathcal{LC}}$

Recordemos que a linguagem $\mathcal{LC}$ difere da linguagem $\mathcal{LU}$ pela existência de caminhos de comprimento superior a 1 e admitir a negação de qualquer restrição. Como se viu, pelo

Teorema 2.34, para qualquer restrição $c \in \mathcal{LC}$ é possível construir uma restrição $c' \in \mathcal{LU}$ tal que c é satisfazível se e só se c' for satisfazível.

Para estender o *Calculus* $\Lambda_{\mathcal{LU}}$ ao *Calculus* $\Lambda_{\mathcal{LC}}$, bastaria modificar a definição da sintaxe, ver página 233, para que as restrições c pertencessem a $\mathcal{LC}$. A semântica para este tipo de descrições não é alterada e para testar a satisfazibilidade de uma restrição em $\mathcal{LC}$, pelo Teorema 2.34, basta testar a satisfazibilidade da correspondente restrição de $\mathcal{LU}$. Assim, o *Calculus* para $\Lambda_{\mathcal{LC}}$ coincidiria com $\Lambda_{\mathcal{LU}}$, considerando antes da aplicação da regra $\Lambda 9$ a transformação da restrição numa restrição equivalente em $\mathcal{LU}$, introduzindo variáveis novas quantificadas existencialmente, e depois eliminando essas variáveis.

No entanto, vamos também permitir que na sintaxe de $\Lambda_{\mathcal{LC}}$ os *valores* sejam de $\mathcal{LC}$, isto é, para além de constantes, possam ser expressões do tipo $x.p$, onde p é um caminho. Assim, serão admitidas aplicações da forma $e_{\mathsf{fs}\to\tau}x.p$. Esta extensão não aumenta a expressividade da linguagem, em relação à satisfazibilidade, apenas evita o uso desnecessário de descrições de tipos superiores. Por exemplo, considerem-se as seguintes descrições

(c_1) $\quad (\lambda y.y = a)x.f_1^2$

(c_2) $\quad \lambda z.(\lambda y.y = a)z \ \wedge \ x.f_1^2 \doteq z$

(c_3) $\quad (\lambda y.y = a)x.f_1^2 \ \wedge \ x.f^2$

Pela semântica, que irá ser definida, a descrição c_1 é satisfazível se e só se a descrição c_2 o é. No entanto, a descrição c_1 será, em aplicações práticas, mais "natural". Note-se, ainda, que a descrição c_3 é "intuitivamente" equivalente a c_1.

A noção de substituição é generalizada a valores da forma $x.p$, considerando a concatenação de caminhos. As reduções α e β são definidas como para $\Lambda_{\mathcal{LU}}$.

A semântica das descrições de $\Lambda_{\mathcal{LC}}$ coincide com a semântica das descrições de $\Lambda_{\mathcal{LU}}$, substituindo valores de $\mathcal{LU}$ por valores de $\mathcal{LC}$ e acrescentando a definição para a semântica de aplicações da forma $e_{\mathsf{fs}\to\tau}x.p$. Recordemos que um valor $x.p$ é o sub-termo t de x na "posição" p, isto é, corresponde ao termo t tal que a expansão $\pi(p,t)$ de p em t é (unificável) a x. Temos então, pela Definição 2.26 das abreviaturas da linguagem $\mathcal{L}$, que

$$(8.5)\quad [\![e_{\mathsf{fs}\to\tau}x.p]\!]\rho \;=\; \begin{cases} ([\![e_{\mathsf{fs}\to\tau}]\!]\rho)s_1 & \text{se } \mathcal{RT} \models_\rho \exists\bar{z}\, x = \pi(p,z_1), \text{ isto é, se existe} \\ & \bar{s} \in \mathcal{RT} \text{ tal que } \mathcal{RT} \models_{\rho[\bar{s}/\bar{z}]} x = \pi(p,z_1) \\ 0_\tau & \text{caso contrário} \end{cases}$$

onde para cada solução de x, s_1 é único.

Para $p \equiv \epsilon$, vem $\mathcal{RT} \models_{\rho[s_1/z_1]} x = z_1$, isto é, $s_1 = \rho(x)$, o que coincide com a semântica das aplicações de valores em $\Lambda_{\mathcal{LU}}$.

Para qualquer tipo τ, descrição e_τ e valor $x.p$ temos que

$$(\lambda y.e_\tau)x.p \quad \approx \quad e_\tau[x.p/y]$$

o que garante que a redução $\beta\delta$ preserva a equivalência das descrições em $\Lambda_{\mathcal{LC}}$.

A Proposição 8.4 e o Lema 8.5 generalizam-se para descrições e_τ de $\Lambda_{\mathcal{LC}}$. A Proposição 8.7 generaliza-se para $\Lambda_{\mathcal{LC}}$, considerando para cada restrição $c \in \mathcal{LC}$, uma restrição $c_u \in \mathcal{LU}$ equivalente para a satisfazibilidade a c. No entanto, o Lema 8.9 não é válido para $\Lambda_{\mathcal{LC}}$, uma vez que a semântica duma aplicação pode depender da semântica dos argumentos, no caso destes serem valores de $\mathcal{LC}$. Contudo, como se mostra nos exemplos c_1 e c_3, é possível extrair a "informação" dos argumentos de forma a obter uma descrição equivalente onde as aplicações, em descrições básicas, possam, para o problema da satisfazibilidade, ser "ignoradas". Para qualquer tipo $\tau \equiv \sigma_1 \to \cdots \to \sigma_n \to \mathsf{bool}$, descrição $e_{\mathsf{fs}\to\tau}$ e valor $x.pf_i^m$, temos que

$$e_{\mathsf{fs}\to\tau} x.pf_i^m \quad \approx \quad e_{\mathsf{fs}\to\tau} x.pf_i^m \ \wedge \ \lambda\bar{x}_{\bar{\sigma}}.\, x.pf^m$$

onde $x \notin \{x^1_{\sigma_1}, \ldots, x^n_{\sigma_n}\}$. De facto, para qualquer atribuição de variáveis ρ, $[\![\lambda\bar{x}_{\bar{\sigma}}.\, x.pf^m]\!]\rho$ é igual ou a 1_τ ou a 0_τ, e neste último caso, $[\![e_{\mathsf{fs}\to\tau} x.pf_i^m]\!]\rho$ também é igual a 0_τ.

Esta equivalência garante a correcção da seguinte regra de reescrita para descrições básicas de tipo $\tau \equiv \sigma_1 \to \cdots \to \sigma_n \to \mathsf{bool}$,

$$\begin{array}{lll} x_{\tau'} e^1_{\sigma'_1} \ldots e^n_{\sigma'_n} \ \wedge \ e_\tau & \longmapsto & x_{\tau'} e^1_{\sigma'_1} \ldots e^n_{\sigma'_n} \ \wedge \ e_\tau \ \wedge \ \lambda\bar{x}_{\bar{\sigma}}.x.pf^m \\ & & \text{se } e^k_{\sigma'_k} \equiv x.pf_i^m \text{ para algum } 1 \le k \le n \text{ e } 1 \le i \le m; \end{array}$$

Para garantir a terminação do sistema de reescrita Λ, acrescentando esta regra (e a modificação da regra $\Lambda 9$ para restrições de $\mathcal{LC}$), é necessário assegurar que a informação da descrição introduzida não é "vinculada" por e_τ.

Definição 8.12 *Sendo $c \in \mathcal{LC}$ e e_τ uma forma normal $\beta\delta$ a relação $c \mathrel{\dot\in} e_\tau$ é definida indutivamente na estrutura de e_τ por*

(i) $c \mathrel{\dot\in} c'$ *se* $c \models_{\mathcal{RT}} c'$;

(ii) $c \mathrel{\dot\in} e^1_\tau \ \wedge \ e^2_\tau$ *se* $c \mathrel{\dot\in} e^1_\tau$ *ou* $c \mathrel{\dot\in} e^2_\tau$;

(iii) $c \mathrel{\dot\in} \lambda x.e_{\tau'}$ *se* $c \mathrel{\dot\in} e_{\tau'}$.

Podemos, então, acrescentar a seguinte regra ao sistema de reescrita Λ, preservando a correcção, a terminação e a completude:

$$\Lambda 11. \quad x_{\tau'} e^1_{\sigma'_1} \ldots e^n_{\sigma'_n} \wedge e_\tau \longmapsto x_{\tau'} e^1_{\sigma'_1} \ldots e^n_{\sigma'_n} \wedge e'_\tau$$

se nenhuma das regras Λ1-Λ9 se aplica, se para algum $1 \leq k \leq n$, $e^k_{\sigma'_k} \equiv x.pf^n_i$, $x.pf^n_i \dot{\notin} e_\tau$ e

$$e_\tau \wedge \lambda \bar{x}_{\bar{\sigma}}.x.pf^n \overset{\Lambda}{\longmapsto}{}^\star e'_\tau$$

onde $e_\tau \not\equiv e'_\tau$.

O controlo da regra Λ10 deve ser alterado de modo a que a regra apenas se aplique caso nenhuma das regras Λ1-Λ9 ou Λ11 se apliquem. Note-se que a regra Λ11 não retira a vantagem do uso de valores de $\mathcal{LC}$ em aplicações, uma vez que essa regra só se aplica depois de todas as reduções β serem efectuadas.

8.4 $\mathcal{CCLG}$, Gramáticas de Categorias com Restrições

Nesta secção vamos mostrar como a expressividade das gramáticas de categorias pode ser aumentada usando o *Calculus* de descrições Λ_{LR}, permitindo, ao contrário, de outros formalismos de gramáticas de categorias baseadas em unificação [ZKC87, Usz86], um tratamento adequado de fenómenos como a semântica da coordenação e tratamento do âmbito da quantificação.

A escolha destas gramáticas, em detrimento dum formalismo baseado em gramáticas independentes de contexto, deve-se, para além da sua simplicidade, à sua estreita relação com formalismos de semântica de LN baseados em λ-*calculus* [Mon74, Cur61, vB83, vB91].

8.4.1 Gramáticas de Categorias e Gramáticas de Unificação

Começamos por fazer uma introdução às gramáticas de categorias e mostrar como versões destas gramáticas baseadas em unificação levantam problemas no tratamento de representações semânticas.

Com base nos trabalhos de Ajdukiewicz [Ajd35], as gramáticas de categorias ($\mathcal{CG}$) foram introduzidas nos fins dos anos cinquenta por Bar-Hillel e seus colegas [YBHS60]. Uma gramática de categorias básica (rígida bi-direccional) é constituída por um conjunto de categorias (ou tipos sintácticos), um léxico, que é uma aplicação de um conjunto finito de palavras L num conjunto de conjuntos finitos de categorias, e um *Calculus* que determina o conjunto de combinações de categorias admissíveis. Dado um conjunto de categorias básicas $\mathcal{C}at_0$, o conjunto das categorias $\mathcal{C}at$ é definido recursivamente por: os elementos de $\mathcal{C}at_0$ são categorias; se A e B são categorias então A/B e $B\backslash A$ são categorias. Nas categorias complexas, denominadas *functores*, A é a categoria *valor* e B a categoria *argumento*. Como nas gramáticas de estrutura de frase, considera-se uma categoria básica distinta A_1 que representa o símbolo inicial. O *Calculus* básico inclui as seguintes regras de inferência, ditas de *de aplicação funcional*:

$$\text{(8.6)} \qquad \begin{array}{lccc} (app/) & A/B \quad B & \to & A \\ (app\backslash) & B \quad B\backslash A & \to & A \end{array}$$

Uma sequência de palavras $x_1 \ldots x_n$ *reduz-se* a uma categoria A se, atribuindo categorias A_i a x_i para $1 \leq i \leq n$, A pode-se derivar de $A_1 \ \ldots \ A_n$, por aplicação das regras de inferência. A linguagem gerada por uma $\mathcal{CG}$ é definida como o conjunto de todos os elementos de $L^\star$ que se reduzem à categoria A_1. Bar-Hillel e os seus colegas [YBHS60] provaram que estas gramáticas são fracamente equivalentes a gramáticas independentes de contexto, isto é, para uma gramática de categorias existe uma gramática independente de contexto que gera a mesma linguagem, e vice-versa.

Tradicionalmente, paralelamente ao *Calculus* das categorias sintácticas, a interpretação semântica duma frase é obtida por composicionalidade dos significados das suas componentes. Associa-se a cada palavra do léxico uma interpretação semântica e a interpretação semântica das regras de aplicação funcional é a aplicação da interpretação semântica da categoria functor à interpretação semântica do seu argumento. Dado o carácter iminentemente funcional destas representações é natural o uso dum λ-*calculus* para a obter os significados das frases. As categorias correspondem essencialmente a tipos semânticos dos λ-termos, obtendo-se assim, um λ-*calculus* tipificado. A existência de forma normal β garante a unicidade da representação semântica. Considerando uma aplicação Υ de Cat_0 num sistema de tipos aplicativos, Υ generaliza-se a Cat definindo

$$\Upsilon(A/B) = \Upsilon(B\backslash A) = \Upsilon(B) \to \Upsilon(A)$$

Considerando M e N λ-termos com tipos $\Upsilon(B) \to \Upsilon(A)$ e $\Upsilon(B)$, respectivamente, as regras (8.6) correspondem à seguinte regra de inferência de tipos para aplicações

$$\text{(8.7)} \qquad (app) \qquad \frac{M : \Upsilon(B) \to \Upsilon(A) \qquad\qquad N : \Upsilon(A)}{(MN) : \Upsilon(B)}$$

Habitualmente, para a semântica (categorial) das linguagens naturais, usam-se os tipos básicos `e` para os indivíduos e `t` para as *proposições*.

Para ilustrar, vamos considerar um fragmento duma gramática de categorias para o Inglês, capaz de analisar a coordenação de frases nominais.

Sejam

- s e iv as categorias básicas, associadas respectivamente a frases e verbos intransitivos, com $\Upsilon(s) = \mathtt{t}$ e $\Upsilon(iv) = \mathtt{e} \rightarrow \mathtt{t}$;
- s/iv a categoria complexa associada a frases nominais[5], que abreviamos para np;
- $(np\backslash(np/np))$ a categoria associada a conectivas de coordenação[6].

Supondo que $john'$, $mary'$, run' e and' são λ-termos dos tipos apropriados, e que, após aplicação duma regra, os λ-termos são reduzidos a forma normal β, a derivação para a interpretação da frase "`John and Mary run`" é apresentada na Figura 8.1. Na figura, por baixo de cada palavra (ou sub-frase) encontra-se a sua categoria e o λ-termo correspondente à sua interpretação semântica.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{\texttt{John}}{\begin{array}{c} s/iv \\ \lambda P.P\ john' \end{array}}
      \quad
      \dfrac{\texttt{and}}{\begin{array}{c} (s/iv)\backslash((s/iv)/(s/iv)) \\ \lambda X.\lambda Y.\lambda Z.and'(XZ)(YZ) \end{array}}
    }{\begin{array}{c} ((s/iv)/(s/iv)) \\ \lambda Y\lambda Z.and'(Z\ john')\ (YZ) \end{array}}\ \backslash
    \quad
    \dfrac{\texttt{Mary}}{\begin{array}{c} s/iv \\ \lambda P.P\ mary' \end{array}}
  }{\begin{array}{c} s/iv \\ \lambda Z.and'(Z\ john')\ (Z\ mary') \end{array}}\ /
  \quad
  \dfrac{\texttt{run}}{\begin{array}{c} iv \\ \lambda x.run'\ x \end{array}}
}{\begin{array}{c} s \\ and'(run'\ john')\ (run'\ mary') \end{array}}\ /
$$

Figura 8.1: Derivação da frase "`John and Mary run`"

Essencialmente temos que a interpretação de `run` é distribuída por cada elemento da conjunção.

[5]Tradicionalmente associa-se a frases nominais a categoria básica np com tipo `e` e a categoria apresentada é obtida por uma transformação de mudança de tipo, ver Secção 8.4.2.

[6]Mais geralmente, podíamos associar tipos polimórficos de modo a que, neste caso, teríamos uma categoria da forma $X\backslash(X/X)$, com X uma variável de tipo.

Por outro lado, nas gramáticas lógicas, como as DCGs, ou genericamente em gramáticas baseadas em unificação, o tratamento directo deste tipo de estruturas levanta problemas técnicos. Desde as gramáticas de metamorfoses, nestes formalismos associam-se às representações semânticas termos com variáveis não instanciadas e a combinação de significados é realizada pela unificação apropriada dessas variáveis [PS87a, Per91]. Por exemplo, se a semântica da palavra "`runs`" for representada pelo termo $run(X)$, o significado de "`John runs`" é obtido unificando X com $john$. Na Figura 8.2 apresenta-se um fragmento duma gramática baseada em unificação[7], que codifica a regra de aplicação funcional ($app/$) usando estruturas de atributos e tem duas entradas lexicais, para `John` e `runs`, respectivamente. Esse fragmento permite derivar a interpretação para a frase "`John runs`", cuja representação é a estrutura de atributos indexada por $X0$.

No entanto, por mera unificação, não é possível obter o significado da frase "`John and Mary run`", uma vez que, para se obter algo como $and(run(john), run(mary))$, a variável X tem de unificar simultaneamente com as constantes $john$ e $mary$. O problema reside no facto de não ser possível distinguir duma forma conveniente variáveis livres e variáveis ligadas. Pode ser parcialmente resolvido, introduzindo operações extra-lógicas, como cópia de termos, ou pela utilização explícita de abstracção e aplicação de λ-termos e pela codificação da redução β [Moo89]. No entanto, ao obrigar o utilizador a lidar explicitamente com estas operações, a interpretação semântica torna-se demasiado complicada de implementar. O mesmo acontece em casos em que é essencial a noção do âmbito das variáveis, como no tratamento da semântica de quantificadores e anáforas [Per90, DSP91, Per91, GH90].

Várias soluções usando abstracção e aplicações funcionais, foram apresentadas e implementadas usando linguagens de ordem superior, nomeadamente o λProlog [Per91, Kul95].

8.4.2 Gramáticas de Categorias Flexíveis

As gramáticas de categorias básicas, apenas com as regras de inferência de aplicação funcional, não permitem o tratamento conveniente de fenómenos linguísticos em que há movimento de constituintes ou dependências de longa distância. Para tal, nas chamadas gramáticas de categorias flexíveis, introduzem-se novas regras de inferência.

Exemplos dessas regras, anotadas com os λ-termos correspondentes, são as regras de *composição*

$$\text{(8.8)}\quad \begin{array}{lllll} (fc/) & A/B : M & B/C : N & \rightarrow & A/C : \lambda x.M(Nx) \\ (fc\backslash) & A\backslash B : M & B\backslash C : N & \rightarrow & A\backslash C : \lambda x.M(Nx) \end{array}$$

[7] Na notação matricial, as variáveis não instanciadas correspondem a índices.

$$X0 \;\rightarrow\; X1\,X2$$

$$\begin{array}{l}
X0.phon = concat(X1.phon, X2.phon) \\
\quad \& \quad X1.cat.dir = right \\
\quad \& \quad X0.cat = X1.cat.val.cat \;\;\&\;\; X2.cat = X1.cat.arg.cat \\
\quad \& \quad X0.var = X1.var \;\;\&\;\; X1.var = X2.var \\
\quad \& \quad X0.rel = X1.rel \;\;\&\;\; X1.rel = X2.rel
\end{array}$$

$$X1\begin{bmatrix} phon & \langle \texttt{John} \rangle \\ cat & \begin{bmatrix} val & \begin{bmatrix} cat & s \end{bmatrix} \\ dir & right \\ arg & \begin{bmatrix} cat & iv \end{bmatrix} \end{bmatrix} \\ rel & \boxed{1} \\ var & \begin{bmatrix} arg & john' \\ num & sg \\ pers & 3 \end{bmatrix} \end{bmatrix} \qquad X2\begin{bmatrix} phon & \langle \texttt{runs} \rangle \\ cat & iv \\ rel & run' \\ var & \begin{bmatrix} arg & \boxed{1} \\ num & sg \\ pers & 3 \end{bmatrix} \end{bmatrix}$$

$$X0\begin{bmatrix} phon & \langle \texttt{John runs} \rangle \\ cat & s \\ rel & run' \\ var & \begin{bmatrix} arg & john' \\ num & sg \\ pers & 3 \end{bmatrix} \end{bmatrix}$$

Figura 8.2: Fragmento duma gramática de unificação categorial com a regra ($app/$)

e as de *mudança de tipo*[8]

$$(8.9) \qquad \begin{array}{llcl} (tr/) & A : M & \rightarrow & B/(B\backslash A) : \lambda x.xM \\ (tr\backslash) & A : M & \rightarrow & (B/A)\backslash B : \lambda x.xM \end{array}$$

Já em 1958, Lambek [Lam58] tinha apresentado um *Calculus*, que corresponde a um fragmento implicacional da lógica (linear) proposicional intuicionista, em que essas regras, entre outras, também habitualmente utilizadas, são teoremas. Nesse *Calculus* , para além das regras de aplicação funcional, existem duas regras de *abstracção funcional* que correspondem a regras de inferência de tipos para abstracções: supondo que x tem tipo τ e M tem tipo τ' concluir que $\lambda x.M$ tem tipo $\tau' \rightarrow \tau$. Recentemente, este *Calculus* tem sido estudado e estendido por diversos autores [vB91, Moo88, Mor89a, Mor89b, Hep90]. Implementações computacionais foram descritas em [Kön89, Kön94, Kön95, Hep92, Mor94, Car94b]. Um dos problemas que a implementação destes *Calculi* levantam, ao contrário do das gramáticas de categorias básicas, é como lidar com a existência de várias demonstrações equivalentes para as mesmas expressões. Uma das técnicas usadas para o resolver, reside na definição de formas normais para as demonstrações, à semelhança da forma normal β [Kön89, Kön94, Hep92].

Curiosamente, Pentus [Pen93] mostrou que as gramáticas associadas ao *Calculus* de Lambek são, como as gramáticas de categorias básicas, fracamente equivalentes às gramáticas independentes de contexto. Esta equivalência tinha sido conjecturada por Chomsky nos anos sessenta [Cho63], mas apenas tinha sido provado que as linguagens independentes de contexto eram reconhecidas por uma gramática de Lambek [Coh67].

Outro tipo de gramáticas de categorias flexíveis são as *gramáticas combinatórias de categorias* (CCG) [AS82, Ste85, Sza87, Ste88, Ste90]. Têm como base o paralelo entre as novas regras introduzidas para resolver problemas linguísticos, como as regras de composição e de mudança de tipo, e os combinadores (λ-termos fechados) usados em sistemas de lógica combinatória [CF58, CHS72]. Consoante o conjunto de combinadores que se considerem como primitivos obtém-se diferentes fragmentos do λ-*calculus*. Do ponto de vista computacional, estes sistemas têm sobre o λ-*calculus* , a vantagem de não usarem variáveis ligadas nem abstracção, o que os torna mais adequados para base de implementações de linguagens de programação funcionais.

Advogando que *Calculus* de Lambek poderá ser demasiado geral para o tratamento dos fenómenos linguísticos, nas CCGs tenta-se restringir o conjunto de regras de inferência. No entanto, por vezes, as regras utilizadas não são independentes ou não são suficientemente expressivas. Por sua vez, o *Calculus* de Lambek pode ser axiomatizado adicionando às

[8] Em Inglês, normalmente, denominam-se de *type-raising*.

regras de aplicação funcional, um conjunto de regras unárias, entre elas as regras (8.6) de *mudança de tipo* [Zie81, Mor89b].

8.4.3 $\mathcal{CCLG}$: Gramáticas de Categorias com Restrições

As gramáticas de categorias com restrições ($\mathcal{CCLG}$) generalizam as gramáticas de categorias associando a cada constituinte duma frase não só uma categoria, mas também uma descrição de Λ_{LR}. O tipo de cada descrição é determinado pela categoria associada. Considera-se uma gramática de categorias básica à qual se adicionam um conjunto de regras unárias que podem ser especificadas para cada gramática e para categorias específicas. Estas regra unárias tornam as gramáticas flexíveis, permitindo o tratamento de diversos fenómenos de descontinuidade. Semanticamente estas regras permitem a abstracção funcional sobre constituintes deslocados ou que faltam.

Formalmente, uma gramática de categorias com restrições $\mathcal{CCLG}$ é um quadruplo

$$< Cat_0, \Upsilon, Lexicon, Rules >$$

onde

(i) Cat_0 é um conjunto de categorias básicas que determina o conjunto de categorias Cat;

(ii) Υ é uma aplicação que associa a cada categoria $C \in Cat$ um tipo $\Upsilon(C)$ e satisfaz

$$\Upsilon(A/B) = \Upsilon(B\backslash A) = \Upsilon(B) \rightarrow \Upsilon(A)$$

(iii) $Lexicon$ é um conjunto de triplos $< w, C, c >$, onde w é uma palavra, C é uma categoria e $c \in \Lambda_{LR}$ é uma descrição de tipo $\Upsilon(A)$;

(iv) $Rules$ é um conjunto de regras de inferência que combinam pares $C - c$ de categorias sintácticas $C \in Cat$ e descrições $c \in \Lambda_{LR}$ (representação semântica).

As regras de inferência usadas nas gramáticas actuais são

$$(app/) \quad \frac{A/B - c_f \quad B - c_b}{A - (c_f c_b)} \quad \text{se } c_f c_b \text{ é satisfazível}$$

$$(app\backslash) \quad \frac{B - c_b \quad B\backslash A - c_f}{A - (c_f c_b)} \quad \text{se } c_f c_b \text{ é satisfazível}$$

mais um conjunto de regras unárias (ou regras de transformação de categorias).

Na Figura 8.3 apresentamos um fragmento duma gramática $\mathcal{CCLG}$. A linguagem de restrições utilizada é uma linguagem de árvores de atributos $\mathcal{FT}$. O construtor `let` permite a definição de *macros* com parâmetros para *descrições* que podem ser usadas para a especificação do léxico. O construtor `transformation` será usado para definir cada uma das regras unárias (Figura 8.5). O construtor `lex` é usado para definir cada entrada lexical. Se associarmos a uma frase uma estrutura de atributos, uma descrição sua será do tipo $\mathsf{fs} \to \tau \to \mathsf{bool}$. Como apenas lidámos com as descrições das estruturas de atributos, a categoria básica `s`, associada a uma frase, terá esse tipo. As representações semânticas que utilizamos nas gramáticas $\mathcal{CCLG}$, baseiam-se nas apresentadas em [PS87a].

A gramática da Figura 8.3 permite analisar a frase "`John and Mary run`", cuja descrição é representada na Figura 8.4. A descrição da coordenação sem concordância, definida pela *macro* AND1, corresponde directamente ao λ-termo usado na gramática de categorias definida na Secção 8.4.1. A *macro* AND2 apenas garante a correcta interpretação da concordância sujeito-predicado.

As definições das macros para as concordâncias, 3RD_SG e NOT_3RD_SG, são um exemplo de que a negação de descrições de ordem superior não é, pelo menos em certos casos, linguisticamente justificada. Se se estender a negação aos domínios semânticos, como foi feito para as conectivas lógicas, temos

$$\neg(\boldsymbol{\lambda} s.d) \stackrel{def}{=} \boldsymbol{\lambda} s.\neg d$$

a que corresponde a regra δ

$$\neg\ \lambda x.e_\tau \quad \mapsto_\delta \quad \lambda x.\neg e_\tau$$

Para a descrição

$$\text{3RD\_SG} \equiv \lambda x.x.pers \doteq p3\ \wedge\ x.nb \doteq sg$$

a sua negação seria

$$\begin{array}{llll}
(8.10) & \neg\text{3RD_SG} & \equiv & \lambda x.\neg(x.pers \doteq p3\ \wedge\ x.nb \doteq sg) \\
(8.11) & & \equiv & \lambda x.\neg(x.pers \doteq p3)\ \vee\ \neg(x.nb \doteq sg) \\
(8.12) & & \equiv & \lambda x.(x.pers\uparrow\ \vee (x.pers \neq p3))\ \vee\ (x.nb\uparrow\ \vee (x.nb \neq sg))
\end{array}$$

que tem um significado diferente do pretendido na macro NOT_3RD_SG.

```
Base_Categories          % Define o conjunto categorias básicas e os seus tipos
      s = fs → bool,
      iv = fs → fs → bool,
      np = s/iv;

%%%% Definição de macros %%%%
% Especificações de concordância
let 3RD_SG = \x. x.pers=p3 & x.nb=sg;

let NOT_3RD_SG = \x. x.pers\=p3 | x.nb\=sg;

% Nomes próprios
let PN(W) = \P.\s. s.reln=naming & s.arg=W & P s.var s.pred & 3RD_SG s.var;

%Verbos intransitivos
let IV(V,AGR)=\x.\p.p.reln=V & p.arg1=x & AGR x;

% Descrição da coordenação nominal sem concordância
let AND1 = \NP1.\NP2.\VT.\s.s.type=coord & NP1 VT s.arg1 & NP2 VT s.arg2

% Descrição da coordenação nominal com concordância
let AND2 = \NP1\NP2\VT\s. s.type=coord &
      NP1 (\x\p p.reln=s.coord.pred.reln& p.arg1=s.arg1.var) s.arg1 &
      NP2 (\x\p p.reln=s.coord.pred.reln & p.arg1=s.arg2.var) s.arg2
      & NOT_3RD_SG s.coord.var & VT s.coord.var s.coord.pred;

%%%% Léxico %%%%
lex john, np, PN(john);
lex mary, np, PN(mary);
lex run, iv, IV(run,NOT_3RD_SG);
lex runs, iv, IV(run,3RD_SG);
lex and, np\(np/np), AND2;
```

Figura 8.3: Fragmento duma gramática $\mathcal{CCLG}$:I

$$
X0 \left[\begin{array}{ll}
type & coord \\
arg1 & \left[\begin{array}{ll}
var\boxed{1} & \left[\begin{array}{ll} nb & sg \\ pers & p3 \end{array}\right] \\
reln & naming \\
pred & \left[\begin{array}{ll} reln & run \\ arg1 & \boxed{1} \end{array}\right] \\
arg & john
\end{array}\right] \\
arg2 & \left[\begin{array}{ll}
var\boxed{1} & \left[\begin{array}{ll} nb & sg \\ pers & p3 \end{array}\right] \\
reln & naming \\
pred & \left[\begin{array}{ll} reln & run \\ arg1 & \boxed{1} \end{array}\right] \\
arg & mary
\end{array}\right]
\end{array}\right]
$$

Figura 8.4: Descrição de “`John and Mary run`”

8.4.3.1 Processamento

O formalismo $\mathcal{CCLG}$ foi implementado em Yap Prolog aumentado com um resolutor de Λ_{LR}, neste caso, $\Lambda_{\mathcal{FT}}$. Cada gramática é processada antes de ser usada para a análise de frases. Embora as descrições usadas nas gramáticas sejam não tipificadas é usado um algoritmo de inferência[9] de tipos, que permite tipificar cada expressão. Para cada entrada lexical o tipo da descrição é comparado com o tipo da sua categoria e, sempre que possível, é calculada uma forma normal para a descrição.

Análise de frases. É utilizado um analisador gramatical ascendente com tabulação, baseado no algoritmo de Cocke-Younger-Kasami [HU79], considerando apenas as categorias sintácticas e com as regras de inferência de aplicação funcional codificadas no analisador. O analisador, constrói uma tabela em que cada entrada é representada por um termo Prolog da forma $arc(Begin, End, Cat, Sref)$. Cat é a categoria que abrange a sub-frase entre as palavras nas posições $Begin$ e End. $Sref$ contém informação que permitirá obter a representação semântica de cada sub-frase e reflecte o modo como essa entrada foi construída: se corresponder a uma entrada lexical $Sref$ é uma referência para a sua representação; se resultar da regra de aplicação funcional esquerda ou direita, é um par de referências para as entradas de cada um dos antecessores; se resultar duma regra de transformação

[9] Note-se que para este sistema de tipos este problema é decidível.

unária, é um par de referências para a entrada inicial (sem transformação) e para a regra unária utilizada. Neste passo, são eliminadas redundâncias que as regras de transformação possam introduzir, isto é, que conduzam a árvores de derivação distintas com representações semânticas equivalentes.

Depois das árvores de derivação serem construídas, as descrições (com a representação semântica) são extraídas e o resolutor de descrições é aplicado. Estas duas componentes podem ser intercaladas para cortar, o mais cedo possível, arcos inconsistentes. No entanto, ao se construir a tabela apenas com categorias sintácticas, evitam-se os problemas que os métodos de análise sintáctica baseados em tabelas apresentam, para as gramáticas de unificação. Na ausência de regras unárias, o problema do reconhecimento universal para estas gramáticas é claramente decidível. Resulta do facto de poder ser executado [10] em duas fases, cada uma delas decidível: reconhecimento universal para gramáticas de categorias básicas e a satisfazibilidade em Λ_{LR}. Na implementação actual não é permita a aplicação sucessiva de regras unárias, portanto, o número de árvores de derivação é também limitado pelo número de palavras da frase. O mesmo resultado ainda é válido se se garantir a ausência de ciclos na aplicação sucessiva dessas regras.

Utilizando uma extensão Tcl/Tk ao Yap Prolog, o $\mathcal{CCLG}$ possui um interface gráfic que permite editar as gramáticas e o léxico, visualizar as árvores (e florestas) de derivação e as descrições de Λ_{LR}.

Outro exemplo. Usando a gramática da Figura 8.5, vamos analisar o processamento da frase

“`a man said that john read a book and mary died`”

Esta frase admite duas árvores de derivação, uma em que a coordenação está no âmbito da frase relativa e outra em que aquela tem âmbito maior. A representação desta frase corresponde a uma descrição da forma `\x_1.X1` $\vee$ `X2`, onde `X1` e `X2` são descrições representadas parcialmente nas Figuras 8.7 e 8.8, respectivamente. As Figuras 8.9 e 8.10 correspondem à representação das frases “`john read a book`” e “`mary died`”, respectivamente. Na descrição da Figura 8.8, o valor de `x_1.arg1.scope.arg1` é identificado com a primeira e o valor de `x_1.arg2` com a segunda. Na descrição da Figura 8.7, o valor de `x_1.scope.arg2` é a descrição da coordenação daquelas duas descrições.

[10]Note-se que esta divisão não é possível em geral se se usar gramáticas independentes de contexto que admitam produções unárias.

```
Base_Categories            % Define o conjunto de categorias básicas
      s = fs → bool,       % e os seus tipos
      iv = fs → fs → bool,
      np = s/iv, tv = iv/np,
      dv = tv/np, n = fs→ fs→bool,
      det = np/n, pp= fs→ bool;
transformation             % Definem regras de mudança de tipo
      np = (s/np)/(iv/np) : \S \Vt \C. S (Vt C);
transformation
      np = (s/(s/np)) : \ S\C. C S;
%%%% Definição de macros %%%%
% Especificações de concordância
let 3RD_SG = \X. X.pers=p3 & X.nb=sg;
let NOT_3RD_SG = \X. X.pers\=p3 | X.nb\=sg; let ANY = \X. X=X;
% Nomes próprios
let PN(W) = \P.\s. s.quant=exists_one & s.arg.reln=naming
      & s.arg.arg1=W & 3RD_SG(s.arg) & P s.arg s.pred ;
% Nomes comuns
let CN(W,AGR) = \s. s.reln=W & s.arg1=x & AGR s;
% Artigos
let DET(Q,AGR) = \N. \P. \s. s.quant=Q & AGR s.var &
      N s.var s.range & P s.var s.scope;
% Verbos intransitivos
let IV(W,AGR) = \s.\p. p.reln=W & p.arg1=s & AGR s;
% Verbos transitivos (Obj é a semântica para o objecto directo)
let TV(W,AGR) = \Obj. \su.\p. Obj (\o \q. q.reln=W & q.arg1=su & q.arg2=o) p;
let V_PP(W,AGR) = \SS. \su \ s. SS s.arg2 & s.reln=W & s.arg1=su & AGR su;
% Verbos di-transitivos
let DV(W,AGR) = \Ci. \Cs. \subj. \si. Cs (\ind. \s. Ci (\obj \p. p.reln=W &
      p.arg1=subj & p.arg2=obj & p.arg3=ind s)) si & AGR subj;
%%%% Léxico %%%%
lex a, det, DET(exists_one,3RD_SG); lex every, det, DET(all,ANY);
lex book, n, CN(book,3RD_SG); lex man, n, CN(man,3RD_SG);
lex john, np, PN(john); lex mary, np, PN(mary);
lex died, iv, IV(die,3RD_SG); lex loves, tv, TV(love,3RD_SG);
lex read, tv, TV(read,ANY); lex said, iv/pp, V_PP(say,ANY);
lex gave, dv, DV(give,ANY); lex that, pp/s, \s.s;
% Coordenação
lex and, s\(s/s), \S1\S2\s. s.type=coord & S1 s.arg1 & S2 s.arg2;
lex and, np\((tv\iv)/np), \NP1\NP2\VT. \subj\s. s.type=coord &
      VT NP1 subj s.arg1 & VT NP2 subj s.arg2;
lex and, iv\(iv/iv), \V1\V2. \subj.\s.s.type=coord & V1 subj s.arg1 & V2 subj s.arg2;
```

Figura 8.5: Fragmento duma gramática $\mathcal{CCLG}$: II

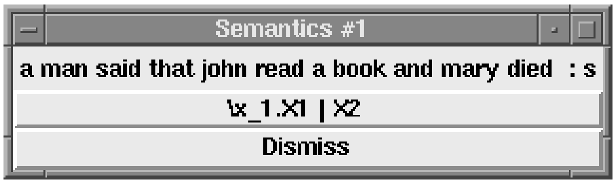


Figura 8.6: Descrição duma frase

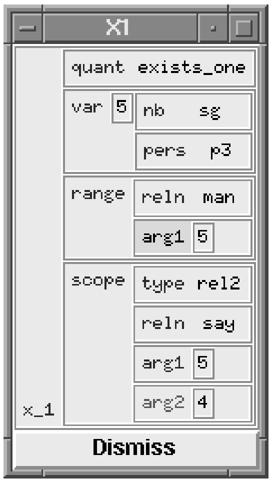


Figura 8.7: Coordenação na frase relativa

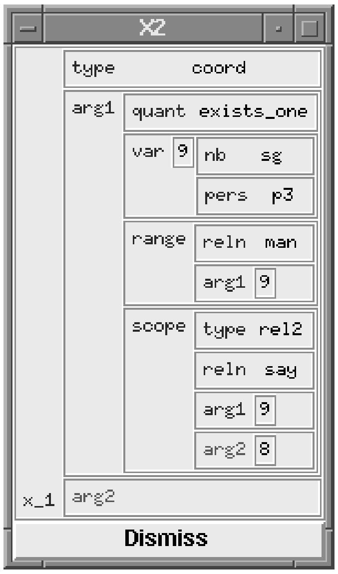


Figura 8.8: Âmbito da coordenação maior

Como foi referido acima, o analisador gramatical constrói primeiro uma floresta de derivação, usando apenas as categorias das entradas lexicais e as regras de inferência. Na Figura 8.11 apresentamos a floresta de derivação da frase "`john read a book`". Na primeira linha, encontram-se as categorias de cada entrada lexical (dada no léxico ou derivada por uma regra de transformação). Nas linhas seguintes, cada entrada corresponde a um dos possíveis modos de derivar uma categoria que abranja uma porção da frase inicial. Por exemplo, a categoria `iv`, na terceira linha da Figura 8.11, pode ser derivada das categorias `iv/(s/iv)` e `s/iv`, que cobrem a sub-frase "`read a book`".

Cada árvore sintáctica que abranja toda a frase, pode ser visualizada separadamente. Por exemplo, na Figura 8.12 apresenta-se uma (única) árvore sintáctica para a frase `John and Mary run`, obtida usando a gramática da Figura 8.3.

Figura 8.9: Representação de "`john read a book`"

Figura 8.10: Representação de "`mary died`"

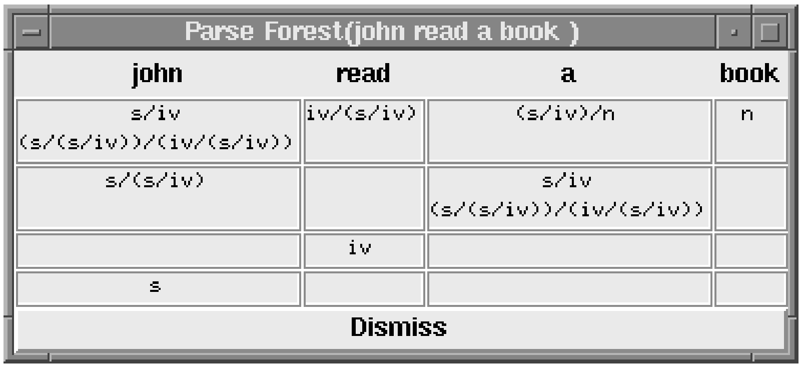


Figura 8.11: Uma floresta de derivação

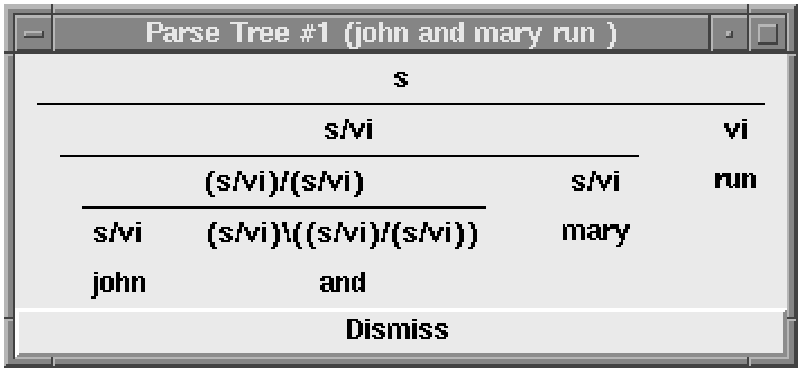


Figura 8.12: Uma árvore sintáctica

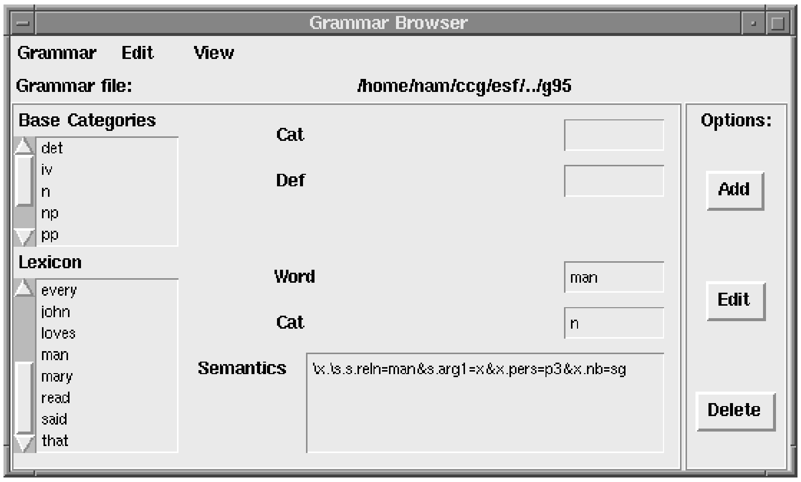


Figura 8.13: Editor de gramáticas $\mathcal{CCLG}$

8.4.4 Extensões e Trabalhos Relacionados

A implementação actual do formalismo $\mathcal{CCLG}$ mostra que a integração de mecanismos de ordem superior com linguagens de descrições de estruturas de atributos é computacionalmente exequível e permite, ganhando expressividade, o tratamento adequado das representações de semânticas de LN. No entanto, do ponto vista linguístico, será conveniente enriquecer o sistema de tipos das descrições, por exemplo, para um ao estilo do formalismo HPSG e que foi utilizado no $\mathcal{CLG}(2)$. Isso permitiria, em primeiro lugar, a distinção entre descrições sintácticas e semânticas.

Por outro lado, outro modo de obter esta distinção é considerar as categorias básicas estruturas complexas munidas duma hierarquia de tipos. Nomeadamente em [BJ95, DM95b], as gramáticas de categorias são generalizadas associando ao conjunto de categorias básicas uma relação de ordem. Em [BJ95] as categorias básicas são fórmulas duma lógica proposicional e a relação de ordem a relação de vínculo entre fórmulas. Em [DM95b] consideram-se, como exemplo, conjunções de fórmulas atómicas da lógica de atributos de Smolka, com a mesma relação de ordem. Em ambos os casos, as categorias básicas são usadas para definir as categorias sintácticas das frases (nominais, verbais, adverbiais, etc) e concordâncias. As gramáticas de Lambek assim generalizadas são ainda (fracamente) equivalentes a gramáticas independentes de contexto [DM95b]. Neste último trabalho são apenas sugeridas formas de estender essas gramáticas de modo a permitir o tratamento de representações semânticas, embora nenhuma na linha do *Calculus* aqui apresentado.

Será interessante investigar a integração este tipo de gramáticas com o *Calculus* de descrições Λ_{LR}, eventualmente, tendo dois níveis de descrições.

8.5 Sumário

Neste capítulo vimos como as linguagens de restrições sobre árvores podem ser generalizadas a linguagens de ordem superior, mantendo-se as características computacionais essenciais e aumentando a expressividade. Em particular, as novas linguagens permitem um tratamento convenientemente da representação semântica de vários fenómenos linguísticos. A novidade do formalismo reside em estender a noção de λ-termo à de restrições complexas de ordem superior.

Capítulo 9

Conclusão

9.1 Sumário

Nesta dissertação apresentamos uma teoria formal e computacional para as linguagens de restrições complexas sobre as álgebras das árvores finitas e racionais. As formas normais utilizadas e os algoritmos desenvolvidos foram motivados pela sua aplicação em formalismos de processamento de Linguagem Natural (PLN) *baseados em unificação*.

A clara distinção entre *estruturas de atributos* e suas descrições, e uma formalização baseada em lógica de 1ª com igualdade, permitiram ainda que as teorias anteriores se aplicassem directamente a essas estruturas de dados, utilizadas extensivamente nos formalismos de PLN referidos. Em particular, vimos em que condições as álgebras de árvores racionais podem ser consideradas os modelos canónicos daquelas estruturas.

O modelo de processamento apresentado, foi utilizado e testado no desenho e desenvolvimento das *gramáticas lógicas com restrições* ($\mathcal{CLG}$).

Os estudos formais das linguagens de restrições desenvolvidos, levaram à obtenção de algoritmos mais eficientes para os resolutores de restrições complexas e que foram, na sua grande parte, implementados e testados.

A implementação dos resolutores de restrições complexas com interface em linguagem Prolog, permite o seu uso generalizado como uma linguagem de programação em lógica por restrições do esquema $\mathcal{CLP}$, tendo como característica específica o tratamento de restrições complexas (disjunções e negações) e, não apenas, conjunções de restrições atómicas.

Finalmente, o *Calculus* Λ_{LR} e o formalismo $\mathcal{CCLG}$ permitiram testar a integrabilidade e eficiência dos resolutores de restrições complexas e aumentar a expressividade das linguagens, sem uma perda significativa de eficiência computacional.

9.2 Trabalho Futuro

Discutimos a seguir alguns direções de trabalho futuro ou questões em aberto, directamente relacionados com o objectivos desta dissertação e que nos parecem mais interessantes e promissores de resultados positivos.

Como vimos no Capítulo 1, Secção 1.3.2.5, as linguagens de restrições que estudamos, não são suficientemente expressivas para a descrição total duma gramática para PLN. O aumento da expressividade, conduziram a linguagens indecidíveis, e portanto, a adopção de formalismos híbridos será o mais aconselhável. Contudo, por exemplo, o desenvolvimento de resolutores para fragmentos de linguagens que incorporem expressões regulares, parece computacionalmente exequível e, como foi visto, tem várias aplicações no tratamento de LN. Por outro lado, será interessante relacionar essas linguagens com o *Calculus* de descrições Λ_{LR} e caracterizar, se possível, a que tipo de extensões ele corresponde.

As teorias de 1ª ordem das álgebras quociente de termos $\mathcal{T}/_{=_E}$, onde E é uma conjunção de equações, são em geral indecidíveis, mas considerando fragmentos ou impondo (fortes) condições sobre E foram obtidos vários resultados positivos de decidibilidade [Com91b, Com91a]. A adaptação das linguagens de restrições utilizadas para $\mathcal{T}$, pelo menos para alguns conjuntos E, e das estratégias usadas nos resolutores de restrições complexas a estas álgebras, é uma extensão natural e merecedora de investigação futura.

Esperamos poder confirmar a Conjectura do Capítulo 5, página 158, referente à obtenção de modelos minimais. Embora, seja um resultado plausível, seria desejável, primeiro, obter uma caracterização mais simples do enunciado do Lema 5.12. Para além das questões de implementação, referidas naquele capítulo, será interessante investigar a aplicabilidade destes resultados nas áreas de Inteligência Artificial que actualmente utilizam algoritmos de obtenção de *implicantes primos* de fórmulas proposicionais ou outras *generalizações* de restrições.

A implementação dos resolutores completos de restrições complexas deve ser melhorada em vários aspectos. Com os protótipos desenvolvidos pretendeu-se testar a eficiência dos algoritmos de satisfazibilidade incremental e facilitar a experimentação de várias estratégias de factorização de restrições complexas.

Para aumentar a eficiência, deve ser desenvolvida uma implementação integral em baixo-nível, que evite a tradução entre as representações internas e externas das restrições. Isto é especialmente relevante para a implementação eficiente dos critérios de independência e métodos de factorização. Outro aspecto importante é a investigação de representações da informação negativa e métodos de detecção de redundâncias.

O interface com o utilizador deve ser melhorado de modo a facilitar a

- edição e visualização de restrições
- parametrização do tratamento da informação negativa
- combinação de resolutores para as várias linguagens de restrições

Como se disse anteriormente, para a implementação dos resolutores para assinaturas finitas será conveniente investigar a incorporação e a combinação de técnicas de satisfazibilidade de restrições em domínios finitos.

Finalmente, devem ser objecto de um estudo mais aprofundado a representação e transformação de fórmulas de lógica de 1ª ordem com igualdade (linguagem $\mathcal{L}$) em restrições de $\mathcal{LU}$ (ou de $\mathcal{LC}$). Isso será importante, para outras aplicações de restrições nestes domínios e para a eventual classificação de fórmulas cuja resolução seja polinomial. Referimos, em particular, a sua aplicação no tratamento da *negação construtiva* em programação em lógica [Cha88].

Apêndice A

Decidibilidade de $\mathcal{RT}$

Teorema A.1 *Sendo $\mathcal{A} = \mathcal{RT}$, para toda a fórmula $\phi \in \mathcal{L}$ existe uma restrição $c \in \mathcal{LC}$ tal que $\mathcal{A} \models \phi \Leftrightarrow c$.*

Dem. A demonstração a seguir apresentada é baseada em [Mah88]. Qualquer fórmula $\phi \in \mathcal{L}$ é equivalente, em qualquer álgebra, a uma fórmula em forma prenex normal. Podemos, então, supor que $\phi \equiv Q_n x_n \dots Q_1 x_1.\psi$, onde ψ é uma fórmula sem quantificadores e Q_i é ou $\forall$ ou $\exists$, para $1 \leq i \leq n$. Pelos lemas do Capítulo 2, a fórmula ψ é equivalente em $\mathcal{RT}$ a uma restrição $c' \in \mathcal{LC}$. Se ϕ não contiver quantificadores o resultado está demonstrado. Caso contrário, seja Y o conjunto de variáveis que ocorrem na forma prenex de ϕ. Vamos demonstrar que

$$\text{(A.1)} \quad \mathcal{RT} \quad \models \quad \phi \Leftrightarrow Q_n x_n \dots Q_1 x_1 c' \Leftrightarrow Q_n x_n \dots Q_1 x_1 \exists \bar{z} c_u \Leftrightarrow \exists \bar{z} c'_u \Leftrightarrow c$$

onde, $c, c' \in \mathcal{LC}$ e $c_u, c'_u \in \mathcal{LU}$. A segunda e quarta equivalências resultam do Teorema 2.34. Em particular, se c'_u for satisfazível e $\exists \bar{z} c'_u$ for uma fórmula fechada então $c \equiv \top$. Falta apenas provar a terceira equivalência, que corresponde à eliminação de quantificadores.

Podemos supor que em c_u as variáveis de Y não ocorrem do lado direito duma equação ou duma restrição do tipo $x.f_i^n \doteq v$. Se tais restrições ocorrerem, podemos substituí-las pelas conjunções $x = z \ \wedge \ v = z$ ou $x.f_i^n \doteq z \ \wedge \ v = z$, onde $v \in Y$ e z é uma variável nova, quantificada existencialmente. Por outro lado, pelas regras de reescrita do Teorema 2.34, todas as variáveis de $\bar{z}$ ocorrem do lado direito duma restrição daquele tipo. Finalmente, as regras de redução a forma resolvida ou a forma primária mantém estas características. Na redução a forma normal, a introdução de uma restrição da forma $z = y$ pode ser substituída pela introdução da restrição $y = z$ e a introdução de restrições da forma $x = y$ com $x, y \in Y$, embora garanta que x é uma variável resolvida na forma normal, pode ser substituída também por uma conjunção $x = z \ \wedge \ y = z$, com z uma variável nova quantificada existencialmente.

Vamos eliminar sucessivamente cada um dos quantificadores Q_i, para $1 \leq i \leq n$. Suponhamos que $Q_1 \equiv \exists$.

Pelo Teorema 4.58, a restrição c_u é equivalente a uma disjunção de formas primárias $\mathcal{P}_1 \ \vee \ \ldots \ \vee \ \mathcal{P}_m$ ou a $\perp$. Neste último caso, vem $\mathcal{RT} \models \phi \ \Leftrightarrow \ \perp$. No primeiro caso, podemos distribuir o quantificador pela disjunção e provar que $\mathcal{RT} \models \exists x_1 \exists \bar{z} P_i \ \Leftrightarrow \exists \bar{z} P'_i$, onde $\mathcal{P}'_i$ é uma forma primária, para $1 \leq i \leq m$. Temos os seguintes casos:

(i) $x_1 \notin \mathsf{Vars}(\mathcal{P}_i)$ então $\mathcal{P}'_i \equiv \mathcal{P}_i$

(ii) $x_1 = v \in \mathcal{P}_i$, então x_1 é resolvida em $\mathcal{P}_i$ e portanto $\mathcal{P}'$ é idêntica a $\mathcal{P}_i$ excluindo apenas a restrição $x_1 = v$.

(iii) $x_1.f^n \in \mathcal{P}_i$ então $\mathcal{P}'_i$ resulta de $\mathcal{P}_i$ excluindo todas as restrições que envolvam x_1. Estas restrições só podem ser do tipo $x_1.f_i^n \doteq z$, $x_1.f_i^n \doteq a$ ou $x_1 \neq v$.

(iv) $\neg x_1.f^n \in \mathcal{P}_i$ então $\mathcal{P}'_i$ resulta de $\mathcal{P}_i$ excluindo todas as restrições que envolvam x_1. Estas restrições só podem ser do tipo $\neg x_1.f^n$ ou $x_1 \neq v$.

(v) $x_1 \neq v \in \mathcal{P}_i$ então $\mathcal{P}'_i$ é obtido como no caso anterior.

A equivalência resulta imediatamente da satisfazibilidade de $\mathcal{P}_i$ e o quantificador Q_1 é, assim, eliminado em cada elemento da disjunção. Note-se que cada $\exists \bar{z} \mathcal{P}'_i$ pode ser transformado numa restrição $c' \in \mathcal{LC}$, envolvendo apenas variáveis de Y.

Se $Q_1 \equiv \forall$ temos

$$\begin{aligned} \mathcal{RT} \ \models \ & Q_n x_n \ldots Q_2 x_2 \forall x_1 c' \ \Leftrightarrow \ Q_n x_n \ldots Q_2 x_2 \neg \exists x_1 \neg c' \\ & \Leftrightarrow \ Q_n x_n \ldots Q_2 x_2 \neg \exists x_1 \exists \bar{z} c_u \ \Leftrightarrow \ Q_n x_n \ldots Q_2 x_2 \neg \exists \bar{z} c'_u \ \Leftrightarrow \ Q_n x_n \ldots Q_2 x_2 c'' \end{aligned}$$

onde $c', c'' \in \mathcal{LC}$ e $c_u, c'_u \in \mathcal{LU}$.

Repetindo este processo podemos eliminar todos os Q_i, para $1 \leq i \leq n$. 2

A decidibilidade de $\mathcal{RT}$ resulta do teorema anterior, uma vez que se ϕ é uma fórmula fechada, então c é $\top$ ou $\perp$.

Analogamente podemos obter a decidibilidade para $\mathcal{T}$ considerando as formas primárias adequadas.

Apêndice B

Normalização Forte $\beta\delta$ de $\Lambda_{\mathcal{LU}}$

Repetimos aqui o enunciado da Proposição 8.4.

Proposição B.1

1. *Para todo tipo τ, $e_\tau \in \Lambda_{\mathcal{LU}}$ admite forma normal $\beta\delta$ e se e'_τ é essa forma, então $e_\tau \approx e'_\tau$.*
2. *A redução $\beta\delta$ satisfaz a propriedade de Church-Rosser.*
3. *Qualquer descrição e_τ é fortemente normalizável em relação a $\beta\delta$.*

A demonstração da Proposição B.1.3. vai ser baseada no método original de W. Tait [Tai67], que é apresentado em [HS86], Apêndice 2, para várias extensões do λ-*calculus* puro tipificado, e também em [Bar92].

Seja $\mathsf{SN} = \{e_\tau \in \Lambda_{\mathcal{LU}} \mid e_\tau$ é fortemente normalizável $\beta\delta\}$.

Qualquer sub-descrição duma descrição em SN pertence a SN.

No que se segue a conectiva Θ designa ou $\wedge$ ou $\vee$.

Definição B.2 *O conjunto SC de descrições* computáveis *de tipo τ é definido por indução na estrutura do tipo τ:*

(i) $e_{\mathsf{bool}} \in \mathsf{SC}$ *se e só se* $e_{\mathsf{bool}} \in \mathsf{SN}$;

(ii) $e_{\mathsf{fs}\to\tau} \in \mathsf{SC}$ *se e só se para todo o valor v de $\mathcal{LU}$,* $e_{\mathsf{fs}\to\tau}v \in \mathsf{SC}$;

(iii) $e_{\tau'\to\tau} \in \mathsf{SC}$ *se e só se para todo o* $e_{\tau'} \in \mathsf{SC}$, $e_{\tau'\to\tau}e_{\tau'} \in \mathsf{SC}$.

Por abuso de notação, se v é um valor de $\mathcal{LU}$, escrevemos $v \in \mathsf{SC}$ e $v \in \mathsf{SN}$.

Seja $\tau \equiv \sigma_1 \to \cdots \to \sigma_n \to \mathsf{bool}$ e e_τ uma descrição. Temos as seguintes equivalências

(B.1) $e_\tau \in \mathsf{SC}$ $\Leftrightarrow$ para todas as descrições $e^1_{\sigma_1}, \ldots, e^n_{\sigma_n} \in \mathsf{SC}$, $e_\tau e^1_{\sigma_1} \ldots e^n_{\sigma_n} \in \mathsf{SC}$
$\Leftrightarrow$ para todas as descrições $e^1_{\sigma_1}, \ldots, e^n_{\sigma_n} \in \mathsf{SC}$, $e_\tau e^1_{\sigma_1} \ldots e^n_{\sigma_n} \in \mathsf{SN}$

onde, a segunda equivalência resulta de $e_\tau e^1_{\sigma_1}, \ldots, e^n_{\sigma_n}$ ser uma descrição do tipo bool.

O lema seguinte coincide com o Lema 1 do Apêndice 2 em [HS86], restrito a descrições de $\Lambda_{\mathcal{LU}}$.

Lema B.3 *Para qualquer tipo* τ

(i) Seja $\tau' \equiv \sigma_1 \cdots \to \sigma_n \to \tau$. *Para toda a variável* $x_{\tau'}$ *e* $e^1_{\sigma_1}, \ldots, e^n_{\sigma_n} \in \mathsf{SN}$, *tem-se que* $x_{\tau'} e^1_{\sigma_1} \ldots e^n_{\sigma_n} \in \mathsf{SC}$.

(ii) Se $e_\tau \in \mathsf{SC}$ *então* $e_\tau \in \mathsf{SN}$.

Dem. Por indução na estrutura do tipo τ.

Base. τ é bool.

(i) Como $e^1_{\sigma_1}, \ldots, e^n_{\sigma_n}$ pertencem a SN e $x_{\tau'}$ é uma variável, então

$$x_{\tau'} e^1_{\sigma_1} \ldots e^n_{\sigma_n} \in \mathsf{SN}$$

mas como é do tipo bool, pertence a SC, pela Definição B.2(i).

(ii) Pela Definição B.2(i).

Indução. a) τ é $\mathsf{fs} \to \tau''$.

(i) Para qualquer valor v, $x_{\tau'} e^1_{\sigma_1} \ldots e^n_{\sigma_n} v$ tem tipo τ''. Por hipótese de indução, $x_{\tau'} e^1_{\sigma_1} \ldots e^n_{\sigma_n} v \in \mathsf{SC}$. Pela Definição B.2(ii) temos $x_{\tau'} e^1_{\sigma_1} \ldots e^n_{\sigma_n} \in \mathsf{SC}$.

(ii) Para qualquer valor v, $e_{\mathsf{fs}\to\tau''} v \in \mathsf{SC}$, e, por hipótese de indução, $e_{\mathsf{fs}\to\tau''} v \in \mathsf{SN}$. Logo, porque é uma sub-descrição, $e_{\mathsf{fs}\to\tau''} \in \mathsf{SN}$.

b) τ é $\tau_1 \to \tau_2$. Análogo ao caso anterior, notando para o caso (ii) que, por hipótese de indução, qualquer variável x_{τ_1} pertence a SC.

No lema seguinte a aliena a) corresponde ao Lema 2 do Apêndice 2 em [HS86] e as restantes são uma generalização para a redução δ.

Lema B.4

a) *(i)* *Se* $e_\tau[v/x] \in \mathsf{SC}$, *então* $(\lambda x.e_\tau)v \in \mathsf{SC}$.

(ii) *Se* $e_\tau[e_{\tau'}/x_{\tau'}] \in \mathsf{SC}$, *então* $(\lambda x_{\tau'}.e_\tau)e_{\tau'} \in \mathsf{SC}$ *desde que* $e_{\tau'} \in \mathsf{SC}$, *se* $x_{\tau'}$ *não ocorre livre em* e_τ.

b) *(i)* *Se* $\mathsf{true}_\tau \in \mathsf{SC}$ *e* $\mathsf{false}_\tau \in \mathsf{SC}$ *então* $\mathsf{true}_{\mathsf{fs}\to\tau}v \in \mathsf{SC}$ *e* $\mathsf{false}_{\mathsf{fs}\to\tau}v \in \mathsf{SC}$.

(ii) *Se* $\mathsf{true}_\tau \in \mathsf{SC}$ *e* $\mathsf{false}_\tau \in \mathsf{SC}$ *então* $\mathsf{true}_{\tau'\to\tau}e_{\tau'} \in \mathsf{SC}$ *e* $\mathsf{false}_{\tau'\to\tau}e_{\tau'} \in \mathsf{SC}$.

c) *Para qualquer dos tipos abaixo, suponha-se que* e^1, $e^2, e^3 \in \mathsf{SC}$ *e pelo menos um dos* e^1 *ou* e^2 *não se reduz* $\beta\delta$ *a uma abstracção.*

(i) *Se* $e^1_{\mathsf{fs}\to\tau}v \ \Theta\ e^2_{\mathsf{fs}\to\tau}v \in \mathsf{SC}$ *então* $(e^1_{\mathsf{fs}\to\tau} \ \Theta\ e^2_{\mathsf{fs}\to\tau})v \in \mathsf{SC}$

(ii) *Se* $e^1_{\tau'\to\tau}e^3_{\tau'} \ \Theta\ e^2_{\tau'\to\tau}e^3_{\tau'} \in \mathsf{SC}$ *então* $(e^1_{\tau'\to\tau} \ \Theta\ e^2_{\tau'\to\tau})e^3_{\tau'} \in \mathsf{SC}$

d) *Para qualquer dos tipos abaixo, suponha-se que* e^1, $e^2 \in \mathsf{SC}$ *e* $e^1 \mapsto^\star_{\beta\delta} \lambda x_\sigma.e^3$ *e* $e^2 \mapsto^\star_{\beta\delta} \lambda x_\sigma.e^4$.

(i) *Se* $\lambda x.e^3_\tau \ \Theta\ e^4_\tau \in \mathsf{SC}$ *então* $e^1_{\mathsf{fs}\to\tau} \ \Theta\ e^2_{\mathsf{fs}\to\tau} \in \mathsf{SC}$

(ii) *Se* $\lambda x_{\tau'}.e^3_\tau \ \Theta\ e^4_\tau \in \mathsf{SC}$ *então* $e^1_{\tau'\to\tau} \ \Theta\ e^2_{\tau'\to\tau} \in \mathsf{SC}$.

Dem. a) Os casos a)(i) e a)(ii) são idênticos. Consideramos apenas o caso a)(ii).

Seja $\tau \equiv \sigma_1 \to \cdots \to \sigma_n \to \mathsf{bool}$. Como $e_\tau[e_{\tau'}/x_{\tau'}] \in \mathsf{SC}$, para todas as descrições $d^1_{\sigma_1}, \ldots d^n_{\sigma_n} \in \mathsf{SC}$, temos que:

$$(e_\tau[e_{\tau'}/x_{\tau'}])d^1_{\sigma_1} \ldots d^n_{\sigma_n} \in \mathsf{SN} \tag{B.2}$$

e $e_\tau[e_{\tau'}/x_{\tau'}]$, $d^1_{\sigma_1}, \ldots, d^n_{\sigma_n} \in \mathsf{SN}$. Vamos provar que

$$(\lambda x_{\tau'}.e_\tau)e_{\tau'}d^1_{\sigma_1} \ldots d^n_{\sigma_n} \in \mathsf{SN}$$

Uma sequência infinita de reduções desta descrição não pode envolver apenas reduções de e_τ, $e_{\tau'}$, $d^1_{\sigma_1}, \ldots, d^n_{\sigma_n}$. Então tem de ser da forma:

$$\begin{array}{rll} (\lambda x_{\tau'}.e_\tau)e_{\tau'}d^1_{\sigma_1} \ldots d^n_{\sigma_n} & \mapsto^\star_{\beta\delta} & (\lambda x_{\tau'}.e'_\tau)e'_{\tau'}d^{11}_{\sigma_1} \ldots d^{n1}_{\sigma_n} \\ & \mapsto_\beta & (e'_\tau[e'_{\tau'}/x_{\tau'}])d^{11}_{\sigma_1} \ldots d^{n1}_{\sigma_n} \\ & \mapsto^\star_{\beta\delta} & \cdots \end{array}$$

onde $e_\tau \mapsto^\star_{\beta\delta} e'_\tau, \ldots$.

Mas então existiria uma derivação infinita para (B.2) o que é um absurdo. Por (B.1) vem $(\lambda x_{\tau'}.e_\tau)e_{\tau'} \in \mathsf{SC}$.

b) Como visto na definição da redução δ, pode ser considerado um caso particular de a). Ou prova-se directamente como no caso a).

c) Os casos c)(i) e c)(ii) são idênticos. Consideramos apenas o caso c)(ii). Seja $\tau \equiv \sigma_1 \to \cdots \to \sigma_n \to \mathsf{bool}$. Como $e^1_{\tau'\to\tau}e^3_{\tau'}\ \Theta\ e^2_{\tau'\to\tau}e^3_{\tau'} \in \mathsf{SC}$, para todas as descrições $d^1_{\sigma_1}, \ldots d^n_{\sigma_n} \in \mathsf{SC}$, temos que:

$$(e^1_{\tau'\to\tau}e^3_{\tau'}\ \Theta\ e^2_{\tau'\to\tau}e^3_{\tau'})d^1_{\sigma_1}\ldots d^n_{\sigma_n} \in \mathsf{SN} \tag{B.3}$$

e $e^1_{\tau'\to\tau}$, $e^2_{\tau'\to\tau}$, $e^3_{\tau'}$, $d^1_{\sigma_1}$, $\ldots$, $d^n_{\sigma_n} \in \mathsf{SN}$.
Vamos provar que

$$(e^1_{\tau'\to\tau}\ \Theta\ e^2_{\tau'\to\tau})e^3_{\tau}d^1_{\sigma_1}\ldots d^n_{\sigma_n} \in \mathsf{SN}$$

Para legibilidade, no que se segue omitimos os tipos das descrições. Se aquela descrição admitir uma sequência infinita de reduções, dado que as sub-expressões pertencem todas a SN, essa sequência tem de ser da seguinte forma:

$$\begin{array}{rll} (e^1\ \Theta\ e^2)e^3d^1\ldots d^n & \mapsto^\star_{\beta\delta} & (e^{11}\ \Theta\ e^{21})e^{31}d^{11}\ldots d^{n1} \\ & \mapsto_{\beta\delta} & (e^{11}e^{31}\ \Theta\ e^{21}e^{31})d^{11}\ldots d^{n1} \\ & \mapsto^\star_{\beta\delta} & \ldots \end{array}$$

onde $e^1 \mapsto^\star_{\beta\delta} e^{11}, \ldots$.
Mas então existiria uma derivação infinita para (B.3) o que é um absurdo. Por (B.1) vem $(e^1_{\tau'\to\tau}\ \Theta\ e^2_{\tau'\to\tau})e^3_\tau \in \mathsf{SC}$.

d) Os casos d)(i) e d)(ii) são idênticos. Consideramos apenas o caso d)(ii). Seja $\tau \equiv \sigma_1 \to \cdots \to \sigma_n \to \mathsf{bool}$. Como nos casos anteriores, para todas as descrições $e'_{\tau'}$, $d^1_{\sigma_1}, \ldots d^n_{\sigma_n} \in \mathsf{SC}$, temos que:

$$(\lambda x_{\tau'}.e^3_\tau\ \Theta\ e^4_\tau)e'_{\tau'}d^1_{\sigma_1}\ldots d^n_{\sigma_n} \in \mathsf{SN} \tag{B.4}$$

e e^3_τ, e^4_τ, $e'_{\tau'}$, $d^1_{\sigma_1}$, $\ldots$, $d^n_{\sigma_n} \in \mathsf{SN}$.
Vamos provar que

$$(e^1_{\tau'\to\tau}\ \Theta\ e^2_{\tau'\to\tau})e'_{\tau'}d^1_{\sigma_1}\ldots d^n_{\sigma_n} \in \mathsf{SN}$$

Como no caso anterior, no que se segue omitimos os tipos das descrições. Uma sequência infinita de reduções desta descrição pode ser da forma:

$$\begin{array}{rll} (e^1\ \Theta\ e^2)e'd^1\ldots d^n & \mapsto^\star_{\beta\delta} & ((\lambda x.e^{31})\ \Theta\ (\lambda x.e^{41}))e''d^{11}\ldots d^{n1} \\ & \mapsto_{\beta\delta} & (\lambda x.e^{31}\ \Theta\ e^{41})e''d^{11}\ldots d^{n1} \\ & \mapsto^\star_{\beta\delta} & \ldots \end{array}$$

onde $e^1 \mapsto^\star_{\beta\delta} (\lambda x.e^{31})$, $e' \mapsto^\star_{\beta\delta} e''$, $\ldots$.

Ou da forma

$$\begin{array}{lll}(e^1 \ \Theta \ e^2)e'd^1 \ldots d^n & \mapsto^\star_{\beta\delta} & ((\lambda x.e^{31})e''d^{11} \ldots d^{i1} \ \Theta \ (\lambda x.e^{41})e''d^{11} \ldots d^{i1})d^{(i+1)1} \ldots d^{n1} \\ & \mapsto^\star_{\beta\delta} & ((\lambda y.e^{32}) \ \Theta \ (\lambda y.e^{42}))d^{(i+1)1} \ldots d^{n1} \\ & \mapsto_{\beta\delta} & (\lambda y.e^{32} \ \Theta \ e^{42})d^{(i+1)1} \ldots d^{n1} \\ & \mapsto^\star_{\beta\delta} & \ldots\end{array}$$

E, em ambos os casos, existiria uma sequência infinita para (B.4) o que é um absurdo. Por (B.1), vem $(e^1_{\tau'\to\tau} \ \Theta \ e^2_{\tau'\to\tau}) \in \mathsf{SC}$, isto é, para todo o $e^3_{\tau'} \in \mathsf{SC}$, $(e^1_{\tau'\to\tau} \ \Theta \ e^2_{\tau'\to\tau})e^3_{\tau'} \in \mathsf{SC}$.

2

Lema B.5 *Para qualquer descrição e_τ:*

(i) $e_\tau \in \mathsf{SC}$

(ii) Para todas as variáveis $x^1_{\sigma_1}, \ldots, x^n_{\sigma_n}$ e descrições $e^1_{\sigma_1}, \ldots, e^n_{\sigma_n} \in \mathsf{SC}$,

$$e^\star_\tau \equiv e_\tau[e^1_{\sigma_1}/x^1_{\sigma_1}] \ldots [e^n_{\sigma_n}/x^n_{\sigma_n}] \in \mathsf{SC}$$

Dem. Provamos (ii) por indução na estrutura de e_τ. O resultado (i) obtém-se de (ii) fazendo $e^i_{\sigma_i} \equiv x^i_{\sigma_i}$, para $1 \le i \le n$. Temos os seguintes casos:

***Caso** $e_\tau \equiv x_\tau$.* Se $x_\tau \equiv x^i_{\sigma_i}$ então $e^\star_\tau \equiv e^i_{\sigma_i} \in \mathsf{SC}$. Senão $e^\star_\tau \equiv x_\tau \in \mathsf{SC}$ pelo Lema B.3.

***Caso** $e_\tau \equiv \mathsf{true}$ **ou** $e_\tau \equiv \mathsf{false}$.* Nestes casos $e^\star_\tau \equiv e_\tau$. Por indução na estrutura do tipo τ.

Base. Se $\tau \equiv \mathsf{bool}$, pela Definição B.2(i).

Indução. Temos dois casos: $\tau \equiv \mathsf{fs} \to \tau'$ e $\tau \equiv \tau' \to \tau''$. Suponhamos que $\mathsf{true}_{\tau'} \in \mathsf{SC}$, então $\mathsf{true}_{\mathsf{fs}\to\tau'} \in \mathsf{SC}$ se e só se para todo o v, $\mathsf{true}_{\mathsf{fs}\to\tau'}v \in \mathsf{SC}$. Mas isso resulta do Lema B.4b). Análogo para o tipo $\tau' \to \tau''$ e para false.

***Caso** $e_\tau \equiv e^1_\tau \ \Theta \ e^2_\tau$.* Suponhamos que $e^{1\star}_\tau$, $e^{2\star}_\tau \in \mathsf{SC}$. Por indução na estrutura do tipo τ.

Base. Se $\tau \equiv \mathsf{bool}$, pela Definição B.2(i).

Indução. Temos dois casos: $\tau \equiv \mathsf{fs} \to \tau'$ e $\tau \equiv \tau' \to \tau''$. Suponhamos que o resultado é verdadeiro para o tipo τ'. Então $e^{1\star}_{\mathsf{fs}\to\tau'} \ \Theta \ e^{2\star}_{\mathsf{fs}\to\tau'} \in \mathsf{SC}$ se e só se para todo o v, $(e^{1\star}_{\mathsf{fs}\to\tau'} \ \Theta \ e^{2\star}_{\mathsf{fs}\to\tau'})v \in \mathsf{SC}$. Temos dois casos a considerar:

(i) $e^{1\star}_{\mathsf{fs}\to\tau'}$ ou $e^{2\star}_{\mathsf{fs}\to\tau'}$ não se reduzem a uma abstracção. Então,

$$(e^{1\star}_{\mathsf{fs}\to\tau'} \ \Theta \ e^{2\star}_{\mathsf{fs}\to\tau'})v \quad \mapsto_{\beta\delta} \quad (e^{1\star}_{\mathsf{fs}\to\tau'}v \ \Theta \ e^{2\star}_{\mathsf{fs}\to\tau'}v) \in \mathsf{SC}$$

porque $(e^{1\star}_{\mathsf{fs}\to\tau'}v \ \Theta \ e^{2\star}_{\mathsf{fs}\to\tau'}v)$ é do tipo τ'. Pelo Lema B.4c),

$$(e^{1\star}_{\mathsf{fs}\to\tau'} \ \Theta \ e^{2\star}_{\mathsf{fs}\to\tau'})v \in \mathsf{SC}$$

(ii) $e^{1\star}_{\mathsf{fs}\to\tau'}$ ou $e^{2\star}_{\mathsf{fs}\to\tau'}$ reduzem-se a uma abstracção.

$$
\begin{array}{lrcl}
(\mathrm{B.5}) & (e^{1\star}_{\mathsf{fs}\to\tau'}\ \Theta\ e^{2\star}_{\mathsf{fs}\to\tau'})v & \mapsto^{\star}_{\beta\delta} & ((\lambda x.e^{3\star}_{\mathsf{fs}\to\tau'})\ \Theta\ (\lambda x.e^{4\star}_{\mathsf{fs}\to\tau'}))v \\
(\mathrm{B.6}) & & \mapsto^{\star}_{\beta\delta} & (e^{3\star}_{\mathsf{fs}\to\tau'}\ \Theta\ e^{4\star}_{\mathsf{fs}\to\tau'})[v/x] \\
(\mathrm{B.7}) & & \equiv & (e^{3\star}_{\mathsf{fs}\to\tau'}[v/x]\ \Theta\ e^{4\star}_{\mathsf{fs}\to\tau'}[v/x])
\end{array}
$$

onde (B.7) $\in \mathsf{SC}$ por hipótese de indução, (B.5) $\in \mathsf{SC}$ pelo Lema B.4a) e, pelo Lema B.4d), $(e^{1\star}_{\mathsf{fs}\to\tau'}\ \Theta\ e^{2\star}_{\mathsf{fs}\to\tau'})v \in \mathsf{SC}$

Análogo para o tipo $\tau' \to \tau''$.

Caso $e_\tau \equiv e^1_{\tau'\to\tau''}e^2_{\tau'}$ ***ou*** $e_\tau \equiv e^1_{\mathsf{fs}\to\tau'}v$. Seja $e^\star_\tau \equiv e^{1\star}_{\tau'\to\tau''}e^{2\star}_{\tau'}$. Por hipótese de indução, $e^{1\star}_{\tau'\to\tau''}$, $e^{2\star}_{\tau'} \in \mathsf{SC}$ logo, pela Definição B.2(iii), $e^\star_\tau \in \mathsf{SC}$. Análogo para $e^1_{\mathsf{fs}\to\tau'}v$.

Caso $e_\tau \equiv \lambda x_{\tau'}.e^1_{\tau''}$ ***ou*** $e_\tau \equiv \lambda x.e^1_{\tau''}$. Seja $e^\star_\tau \equiv \lambda x_{\tau'}.e^{1\star}_{\tau''}$. Para que $e^\star_\tau \in \mathsf{SC}$, para todos os $e^3_{\tau'}$, $e^\star_\tau e^3_{\tau'} \in \mathsf{SC}$. Mas

$$
\begin{array}{rcl}
e^\star_\tau e^3_{\tau'} & \equiv & (\lambda x_{\tau'}.e^{1\star}_{\tau''})e^3_{\tau'} \\
 & \mapsto_{\beta\delta} & e^{1\star}_{\tau''}[e^3_{\tau'}/x_{\tau'}]
\end{array}
$$

que pertence a SC, por hipótese de indução. Então, pelo Lema B.4a), $e^\star_\tau e^3_{\tau'} \in \mathsf{SC}$. Análogo para $\lambda x.e^1_{\tau''}$.

2

A condição 3. da Proposição B.1 resulta do Lema B.5(i) e do Lema B.3(ii).

Para a condição 2. basta provar que a redução δ satisfaz a propriedade de Church-Rosser e que a redução β comuta com a redução δ. As demonstrações são análogas às usadas para provar que $\beta\eta$ satisfaz a propriedade de Church-Rosser, [Bar84], Capítulo 3. A primeira parte da condição 1. é implicada pelas condições 3. e 2., e a demonstração da segunda é análoga à do mesmo resultado para a redução β, [Sto77].

Apêndice C

Terminologia

Apresentamos aqui traduções de expressões que foram usadas neste trabalho.

Inglês – Português

assignment	atribuição de variáveis
branch	ramo
constraint	restrição
definite clause grammar	gramática de cláusulas definidas
domain closure axiom	axioma do domínio fechado
dereference	desreferenciação
determinant	determinante
entails	vincula
entailment	vínculo
extensional	extensiva
to fail	falhar
failure	falha
feature	atributo
feature logic	lógica de estruturas de atributos
feature structure	estrutura de atributos
feature tree	árvore de atributos
feature term	termo de atributos
tree adjoining grammar	gramática de árvores adjacentes
functional unification grammar	gramática funcional de unificação

frame	enquadramento
goal	objectivo
head-driven phrase structure grammar	gramática de estrutura de frase guiada por cabeça
link	ligação
lexical functional grammar	gramática funcional léxica
normal form	forma normal
path	caminho
parser	analisador sintáctico
performance	desempenho
record	estrutura
satisfability	satisfazibilidade
satisfiable	satisfazível
slot	projecção
solved form	forma resolvida
stand alone	autónomo
strongly normalizing	fortemente normalizável
trail	trilha
unification categorial grammar	gramática categorial de unificação

Português – Inglês

analisador sintáctico	parser
árvore de atributos	feature tree
atribuição de variáveis	assignment
atributo	feature
autónomo	stand alone
axioma do domínio fechado	domain closure axiom
caminho	path
desempenho	performance
desreferenciação	dereference
determinante	determinant
estrutura de atributos	feature structure
estrutura	record
enquadramento	frame
extensiva	extensional

falhar	to fail
falha	failure
forma normal	normal form
forma resolvida	solved form
fortemente normalizável	strongly normalizing
gramática de cláusulas definidas	definite clause grammar
gramática funcional de unificação	functional unification grammar
gramática funcional léxica	lexical functional grammar
gramática categorial de unificação	unification categorial grammar
gramática de estrutura de frase guiada por cabeça	head-driven phrase structure grammar
gramática de árvores adjacentes	tree adjoining grammar
ligação	link
lógica de estruturas de atributos	feature logic
objectivo	goal
projecção	slot
ramo	branch
restrição	constraint
satisfazibilidade	satisfability
satisfazível	satisfiable
termo de atributos	feature term
trilha	trail
vincula	entails
vínculo	entailment

Referências

[Ajd35] K. Ajdukiewicz. Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27, 1935. Also in S. McCall (eds) *Polish Logic: 1920-1939*, Oxford University Press, 207–231.

[AK86] Hassan Aït-Kaci. An algebraic semantics approach to the effective resolution of type equations. *Theoretical Computer Science*, 45:293–351, 1986.

[AKP93a] Hassan Aït-Kaci and A. Podelski. Towards a meaning of LIFE. *Journal of Logic Programming*, 16:195–234, 1993.

[AKP93b] Hassan Aït-Kaci and Andreas Podelski. Entailment and disentailment for order-sorted feature constraints. In Andrei Voronkov, editor, *Proceedings of the Fourth International Conference on Logic Programming and Automated Reasoning.* Springer-Verlag, 1993.

[AKPS92] Hassan Aït-Kaci, Andreas Podelski, and Gert Smolka. A feature-based constraint system for logic programming with entailment. Technical Report RR-92-17, DFKI, 1992.

[AKPS94] Hassan Aït-Kaci, Andreas Podelski, and Gert Smolka. A feature-based constraint system for logic programming with entailment. *Theoretical Computer Science*, 122(1-2):263–283, January 1994.

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory:To Truth Through Proof.* Academic Press, Orlando, Florida, 1986.

[AS82] Anthony Ades and Mark. Steedman. On the order of words. *Linguistics and Philosophy*, 4:517–558, 1982.

[ASS+88] A. Aiba, K. Sakai, Y. Sato, D. J. Hawley, and R. Hasegawa. Constraint logic programming language CAL. In *Proceedings of the International Conference on*

Fifth Generation Computer Systems (FGCS-88), ICOT, Tokyo, pages 263–276, December 1988.

[Bac94a] Rolf Backofen. *Expressivity and Decidability of First Order Theories over Feature Trees*. PhD thesis, Technische Fakultät der Universität des Saarlandes, Germany, 1994.

[Bac94b] Rolf Backofen. Regular path expressions in feature logic. *Journal of Symbolic Computation*, 17:421–455, 1994.

[Bac96] Rolf Backofen. Controling functional uncertainty. In W. Wahlster, editor, *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI 96)*, pages 557–562. John Wiley and Sons, 1996.

[Bar84] H. P. Barendregt. *The Lambda Calculus*. Elsevier Science Publishers, 1984.

[Bar92] H. P. Barendregt. Lambda calculi with types. In S. Abramsky D. M. Gabbay and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1. Oxford University Press, 1992.

[BDMV90] Sergio Balari, Luís Damas, Nelma Moreira, and Giovanni B. Varile. $\mathcal{CLG}$: Constraint logic grammars. In H.Karlgren, editor, *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, volume 3, pages 7–12, Helsinki, 1990.

[BJ95] Sam Bayer and Mark Johnson. Features and agreement. In *33th Annual Meeting of the Association for Computational Linguistics*, Cambridge, Mass., 1995. Massachusetts Institute of Technology.

[BK93] Rolf Backofen and Hans-Ulrich Krieger. The $\mathcal{TDL}/\mathcal{UDINE}$ system. In Backofen et al. [BKSU93], pages 67–74.

[BKSU93] Rolf Backofen, Hans-Ulrich Krieger, S. Spackman, and Hans Uszkoreit, editors. *Report of the EAGLES Workshop on Implemented Formalisms at DFKI*, Saarbrücken, March 1993.

[Bla91] Patrick Blackburn. Modal logic and attribute value structures. In M. de Rijke, editor, *Diamonds and Defaults*, Studies in Logic, Language and Information. Kluwer, 1991.

[Bla94] Patrick Blackburn. Structures, languages and translations: the structural approach to feature logic. In C. Rupp, M. A. Rosner, and R. L. Johnson, editors, *Constraints, Language, and Computation*, Computation in Cognitive Science, pages 1–27. Academic Press, London, 1994.

[Bro92] Sabine Broda. Um sistema de reescrita completo para linguagens com um número finito de features. Technical report, Centro de Informática da Universidade do Porto, 1992.

[BS92] Rolf Backofen and Gert Smolka. A complete and recursive feature theory. Technical Report RR-92-30, DFKI, 1992.

[BS93a] Patrick Blackburn and Edith Spaan. Decidability and undecidability in stand-alone feature logics. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics*, pages 30–36, Utrecht, 1993.

[BS93b] Patrick Blackburn and Edith Spaan. A modal perspective on the computational complexity of attribute value grammar. *Journal of Logic, Language and Information*, 2:129–169, 1993.

[BS95] Rolf Backofen and Gert Smolka. A complete and recursive feature theory. *Theoretical Computer Science*, 1995.

[Büc60] J. R. Büchi. On a decision method in restricted second order arithmetic. In E. Nagel et. al., editor, *International Congr. on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1960.

[Car84] Luca Cardelli. Semantics of multiple inheritance. In Kahn, MacQueen, and Plotkin, editors, *Semantics of Data Types*, volume 173 of *Lecture Notes on Computer Science*. Springer-Verlag, 1984.

[Car90] Bob Carpenter. Typed feature structures: Inheritance, (in)equations and extensionality. In *Inheritance in Natural Language Processing*, pages 9–18, Institute for Language Technology and AI, Tilburg University, 1990.

[Car92] Bob Carpenter. *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge, 1992.

[Car94a] Bob Carpenter. Ale: the Attribute Logic Engine. User's guide. Technical report, Carnegie-Mellon University, Pittsburgh, 1994. Version 2.0.

[Car94b] Bob Carpenter. A natural deduction theorem prover for type-theoretic categorial grammars. Technical report, Carnegie-Mellon Laboratory for Computational Linguistics, Pittsburgh, 1994.

[CF58] H. B. Curry and R. Feys. *Combinatory Logic*, volume 1. North Holland, Amsterdam, 1958.

[CF92] Hubert Comon and M. Fernández. Negation elimination in equational formulae. In *Proceedings of the 17th Mathematical Foundations of Computer Science*, number 629 in Lecture Notes on Computer Science, Praha, 1992. Springer-Verlag.

[CH57] Noam Chomsky and M. Halle. *The Sound Pattern of English.* Harper and Row, 1957.

[Cha88] David Chan. Constructive negation based on the completed database. In *Logic Programming: Proceedings of the Fifth International Conference and Symposium*, pages 111–125. MIT Press, 1988.

[Cho63] Noam Chomsky. Formal proprieties of grammars. In R. D. Luce, editor, *Handbook of Mathematical Psychology*, volume 2, pages 323–418. John Wiley and Sons, 1963.

[Cho65] Noam Chomsky. *Aspects of the Theory of Syntax.* MIT Press, 1965.

[CHS72] H. B. Curry, J. R. Hindley, and J. P. Seldin. *Combinatory Logic*, volume 2. North Holland, Amsterdam, 1972.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[CKPR73] Alain Colmerauer, H. Kanoui, R. Pasero, and P. Roussel. Un système de communication homme-machine en Français. Technical report, Univertité d'Aix-Marseille II, 1973.

[CL89] Hubert Comon and Pierre Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:317–425, 1989.

[Cla78] K. L. Clark. Negation as failure. In H. Gallaire and J.Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, New York, 1978.

[CN95] Philippe Codognet and Giuseppe Nardiello. Enhancing the constraint-solving power of CLP(FD) by means of path-consistency methods. In Andreas Podelski, editor, *Constraint Programming: Basics and Trends*, number 910 in Lecture Notes on Computer Science, pages 39–61. Springer-Verlag, 1995.

[Coh67] J. Cohen. The equivalence of two concepts of categorial grammar. *Information ant Control*, 10:475–484, 1967.

[Col78] Alain Colmerauer. Metamorphosis grammars. In L. Bolc, editor, *Natural Language Communication with Computers*. Springer-Verlag, 1978.

[Col82] Alain Colmerauer. Prolog and infinite trees. In K. L. Clark and S. A. Tarnlund, editors, *Logic Programming*, pages 231–251. Academic Press, New York, 1982.

[Col90] Alain Colmerauer. An introduction to PROLOG-III. *Communications of the ACM*, 33(7):69–90, July 1990.

[Col94] Alain Colmerauer. Equations and inequations on finite and infinite trees. In *Proceedings of the 2nd International Conference on Fifth Generation Computer System*, pages 85–99, Tokyo, 1994.

[Com88] Hubert Comon. *Unification et Disunification: Théorie et Applications*. PhD thesis, I.N.P. de Grenoble, France, 1988.

[Com90] Commission of the European Community, Luxembourg. *The Eurotra Reference Manual*, 1990.

[Com91a] Hubert Comon. Complete axiomatizations of some quotient term algebras. In *Proceedings of ICALP*, number 510 in Lecture Notes on Computer Science, Madrid, 1991. Springer-Verlag.

[Com91b] Hubert Comon. Disunification: a survey. In J.-J. Lassez and Gordon Plötkin, editors, *Computational Logic:Essays in Honor of Alan Robinson*, pages 322–359. MIT Press, 1991.

[Cou83] Bruno Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, 2(25):95–169, March 1983.

[Cur61] H. B. Curry. Some logical aspects of grammatical structure. In R. Jakobson, editor, *Structure of Languege and its Mathematical Aspects*. AMS, Providence, 1961.

[Dah94] Veronica Dahl. Natural language processing and logic programming. *Journal of Logic Programming*, 19(20):681–714, 1994.

[Dam88] Luís Damas. A formal model for the ETS engine. Technical report, Universidade do Porto, 1988.

[Dam91] Luís Damas. $\mathcal{CLG}$ final report. Technical report, Commission of the European Communities, September 1991.

[DCAR89] Luís Damas, Vítor Santos Costa, R. Azevedo, and R. Reis. *Yap Reference Manual*. Centro de Informática da Universidade do Porto, 1989.

[DE90] Jochen Dörre and Andreas Eisele. Feature logic with disjunctive unification. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, pages 100–105, Helsinki, Finland, 1990.

[Der82] Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 6, pages 245–320. Elsevier Science Publishers, 1990.

[dK92] J. de Kleer. An improved incremental algorithm for computing prime implicants. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-92), San José, California*, pages 780–785, 1992.

[dKMR90] J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-90), Boston, Massachusetts*, pages 324–330, 1990.

[dKR87] J. de Kleer and R. Reiter. Foundations of assumption-based truth maintenance systems: preliminary report. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-87), Seattle*, pages 183–188, 1987.

[DM79] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, August 1979.

[DM91] Luís Damas and Nelma Moreira. A quantifier elimination algorithm for a first order logic with equality. Technical report, Univerdidade do Porto, July 1991.

[DM93] Luís Damas and Nelma Moreira. $\mathcal{CCLG}$: Constraint categorial grammars. In Backofen et al. [BKSU93], pages 26–28.

[DM95a] Luís Damas and Nelma Moreira. Constraint categorial grammars. In Nuno Mamede and Carlos Pinto-Fereira, editors, *Proceedings of the 7th Portuguese Conference on Artificial Intelligence (EPIA 95)*, volume 990 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1995.

[DM95b] Jochen Dörre and Suresh Manandhar. On constraint-based Lambek calculi. Hcrc/tr-69, University of Edinburgh, 1995.

[DMB93] Luís Damas, Nelma Moreira, and Sabine Broda. Resolution of constraints in algebras of rational trees. In Miguel Filgueiras and Luís Damas, editors, *Progress in Artificial Intelligence — 6th Portuguese Conference on Artificial Intelligence (EPIA 93)*, volume 727 of *Lecture Notes in Artificial Intelligence*, pages 61–76. Springer-Verlag, 1993.

[DMV91] Luís Damas, Nelma Moreira, and Giovanni B. Varile. The formal and processing models of $\mathcal{CLG}$. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics*, pages 173–178, Berlin, 1991.

[DMV94] Luís Damas, Nelma Moreira, and Giovanni B. Varile. The formal and computational theory of complex constraint solution. In C. Rupp, M. A. Rosner, and R. L. Johnson, editors, *Constraints, Language, and Computation*, Computation in Cognitive Science, pages 149–166. Academic Press, London, 1994.

[DSP91] Mary Dalrymple, Stuart M. Shieber, and Fernando C. N. Pereira. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14:399–452, 1991.

[DV89a] Luís Damas and Giovanni B. Varile. $\mathcal{CLG}$: A grammar formalism based on constraint resolution. In E.M.Morgado and J.P.Martins, editors, *EPIA 89*, volume 390 of *Lecture Notes in Artificial Intelligence*, pages 175–186. Springer-Verlag, 1989.

[DV89b] Luís Damas and Giovanni B. Varile. A guide to Constraint Logic Grammar. Technical report, Commission of the European Communities, 1989. MK2 document set.

[DV92] Luís Damas and Giovanni B. Varile. On the satisfiability of complex constraints. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, Nantes, France, 1992.

[DvS+88] M. Dincbas, P. van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier. The Constraint Logic Programming Language CHIP. In *Proceedings of the International Conference on Fifth Generation Computer Systems FGCS-88*, pages 693–702, Tokyo, Japan, December 1988.

[DVS90] Anuj Dawar and K. Vijay-Shanker. An interpretation of negation in feature structure descriptions. *Computational Linguistics*, 16(1):11–21, March 1990.

[ED88] Andreas Eisele and Jochen Dörre. Unification of disjunctive feature descriptions. In *26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, New York, 1988.

[Fer96] M. Fernández. AC complement problems: Satisfiability and negation elimination. *Journal of Symbolic Computation*, 22:49–82, 1996.

[FH86] François Fages and Gérard Huet. Complete sets of unifiers and matchers in equational theories. *Theoretical Computer Science*, 43:189–200, 1986.

[GH90] Dale Gerdemann and Erhard Hinrichs. Unification-based approach to quantifier scoping. In L. Aiello, editor, *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI 90)*, pages 292–297, 1990.

[GJ79] Michael Garey and David Johnson. *Computers and Intractability: a guide to the theory of NP-Completeness.* W.H. Freeman and Company, San Francisco, 1979.

[GJW86] Barbara Grosz, Karen Sparck Jones, and Bonny Lynn Webber, editors. *Readings in Natural Language Processing.* Morgan Kaufmann, 1986.

[GKPS85] Gerald Gadzar, E. Klein, G. Pullum, and I. Sag. *Generalized Phrase Structure Grammar.* Oxford: Basil Blackwell, 1985.

[GP85] Gerald Gazdar and Geoffrey K. Pullum. Computationally relevant properties of natural languages and their grammars. *New Generation Computing*, 3, 1985.

[GS90] C. Gunter and D. Scott. Semantic domains. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 1, chapter 12, pages 633–674. Elsevier Science Publishers, 1990.

[Han94] Hans-Ulrich Krieger and Ulrich Schäfer. $\mathcal{TDL}$ – a type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, pages 893–899, Kyoto, Japan, 1994.

[Har84] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 497–604. Reidel, 1984.

[Hep90] Mark Hepple. *The Grammar and Processing of Order and Dependency: a Categorial Approach.* PhD thesis, University of Edinburgh, Edinburgh, Scotland, 1990.

[Hep92] Mark Hepple. Chart parsing Lambek grammars. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 134–140, Nantes,France, 1992.

[Her30] Jacques Herbrand. *Recherches sur la theorie de la démonstration*. PhD thesis, Université de Paris, 1930. Ecrits logiques de Jacques Herbrand, Paris, PUF, 1968.

[HS86] J. Roger Hindley and J. P. Seldin. *Introduction to combinators and λ- Calculus*. Cambridge University Press, 1986.

[HSW93] Martin Henz, Gert Smolka, and Jörg Würt. Oz– a programming language for multi-agent systems. In Ruzena Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery, France*, volume 11, pages 404–409, Chambéry, 1993. Morgan Kaufmann Publishers.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 1979.

[Hue76] Gérard Huet. *Résolution d'equations dans les Langages d'ordre* $1, 2, \ldots, \omega$. PhD thesis, Univ. Paris 7, 1976.

[Jaf84] J. Jaffar. Efficient unification over infinite trees. *New Generation Computing*, 2(2):207–219, 1984.

[JK91] Jean-Pierre Jouannaud and Claude Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In J.-J. Lassez and Gordon Plötkin, editors, *Computational Logic:Essays in Honor of Alan Robinson*, pages 257–321. MIT Press, 1991.

[JL87] Joxan Jaffar and Jean-Louis Lassez. Constraint logic programming. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages (POPL), Munich, Germany*, pages 111–119. ACM, January 1987.

[JM94] Joxan Jaffar and Michael Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 1994.

[JMSY90] Joxan Jaffar, Spiro Michaylov, Peter Stuckey, and Roland Yap. The CLP($\mathcal{R}$) language and system. Technical Report RC 16292 (#72336) 11/15/90, IBM Research Division, November 1990.

[Joc93] Jochen Dörre and Michel Dorna. CUF: A formalism for linguistic knowledge representation. Technical report, DYANA Deliverable, 1993.

[Joh88] Mark Johnson. *Attribute Value Logic and the Theory of Grammar*. Center for the Study of Language and Information Stanford, 1988.

[Joh90] Mark Johnson. Expressing disjunctive and negative feature constraints. In *28th Annual Meeting of the Association for Computational Linguistics*, pages 173–179, Chicago, 1990.

[Joh91] Mark Johnson. Features and formulae. *Computational Linguistics*, 17(2):131–151, 1991.

[Joh94] Mark Johnson. Computing with features as formulae. *Computational Linguistics*, 20(1):1–25, 1994.

[JP90] Peter Jackson and John Pais. Computing prime implicants. In M. E. Stickel, editor, *10th International Conference on Automated Deduction*, number 449 in Lecture Notes on Computer Science, pages 543–557. Springer-Verlag, 1990.

[JRSL70] C. Chang J. R. Slagle and R. C. Lee. A new algorithm for generating prime implicants. *IEEE Transactions on Computers*, 19(4):304–310, 1970.

[Kar84] L. Karttunen. Features and values. In *Proceedings of the 10th International Conference on Computational Linguistics (COLING)*, Stanford, CA, 1984.

[Kas87] R. T. Kasper. Unification method for disjunctive feature descriptions. In *25th Annual Meeting of the Association for Computational Linguistics*, pages 235–242, Standford, CA, 1987.

[Kas88] R. T. Kasper. Conditional descriptions in functional unification grammar. In *26th Annual Meeting of the Association for Computational Linguistics*, pages 233–240, Buffalo, New York, 1988.

[Kay85a] Martin Kay. Parsing in functional unification grammar. In David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, editors, *Natural Language Parsing*. Ellis Horwood/Wiley, 1985. also appears in [GJW86].

[Kay85b] Martin Kay. Unification in grammar. In V. Dahl and P. Saint-Dizier, editors, *Natural Language Understanding and Logic Programming*, pages 233–240. Elsevier Science Publishers, 1985.

[KB82] R. Kaplan and J. Bresnan. Lexical functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*. MIT Press, 1982.

[Kei94] Torbjörn Keisu. *Trees Constraints*. PhD thesis, The Royal Institute of Technology, Stockholm, Sweden, 1994.

[Kel93] Bill Keller. *Feature Logics, Infinitary Descriptions and Grammar.* PhD thesis, University of Sussex, 1993.

[KI88] Ronald M. Kaplan and John T. Maxwell III. An algorithm for functional uncertainly. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, pages 297–302, Budapest, Hungary, 1988.

[Kön89] Esther König. Parsing as natural deduction. In *27th Annual Meeting of the Association for Computational Linguistics*, pages 272–279, Vancouver, British Columbia, 1989.

[Kön94] Esther König. A hypothetical reasoning algorithm for linguistic analysis. *Journal of Logic and Computation*, 1994. to appear.

[Kön95] Esther König. Lexgram– a practical categorial grammar formalism. In *Computational Logic for Natural Language Processing*, Edinburgh, 1995 1995.

[Kow79] Robert Kowalski. *Logic for problem solving.* Elsevier Science Publishers, 1979.

[KR86] R.T. Kasper and W.C. Rounds. A logical semantics for feature structures. In *24th Annual Meeting of the Association for Computational Linguistics*, pages 257–266, Columbia university, New York, 1986.

[KR90] R. T. Kasper and W. C. Rounds. The logic of unification in grammar. *Linguistics and Philosophy*, 13(1):35–58, 1990.

[Kri95] Hans-Ulrich Krieger. *$\mathcal{TDL}$ – A Type Description Language for Constraint-Based Grammars. Foundations, Implementation and Applications.* PhD thesis, Universität des Saarlandes, 1995.

[KT90] A. Kean and G. Tsiknis. An incremental method for generating prime implicants/implicates. *Journal of Symbolic Computation*, 9:185–206, 1990.

[Kul95] Seth Kulick. Using higher-order logic programming for semantic interpretation of coordinate constructs. In *33th Annual Meeting of the Association for Computational Linguistics*, 1995.

[Kun87] K. Kunen. Negation in logic programming. *Journal of the Logic Programming*, 4:289–308, 1987.

[KZ86] Ronald Kaplan and Annie Zaenen. Long-distance dependencies, constituent structure, and functional uncertainty. In M. Baltin and A. S. Kroch, editors, *Aternative Conceptions of Phrase Structure*, pages 17–42. University of Chicago Press, 1986.

[Lam58] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958. Reprinted in: Buszkowski, W., W. Marciszewski, and J. van Benthem (eds): Categorial Grammar. Amsterdam, 1988.

[Lan94] Tore Langholm. Describing sets with sets: remarks on the use and interpreattion of set-valued features structures. In C. Rupp, M. A. Rosner, and R. L. Johnson, editors, *Constraints, Language, and Computation*, Computation in Cognitive Science, pages 167–186. Academic Press, London, 1994.

[LM87] J.-L. Lassez and K. Marriott. Explicit representation of terms defined by counter examples. *Journal of Automated Reasoning*, 3:301–317, 1987.

[LM90] J.-L. Lassez and K. McAloon. A constraint sequent calculus. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 52–62. IEEE, Philadelphia, 1990.

[LMD91] José P. Leal, Nelma Moreira, and Luis Damas. An history based interface. In *ICLP 91 Workshop Procceedings*, 1991.

[LMM87] J.-L. Lassez, M.J. Maher, and K. Marriott. Unification revisited. In J.Minker, editor, *Foundations of Deductive Databases and Logic Programming*. Kauffman, 1987.

[LMM92] J.-L. Lassez, Michael Maher, and K. Marriott. Elimination of negation in term algebras. draft, 1992.

[Mah] Michael Maher. Complexity. draft.

[Mah88] Michael J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. Technical report, IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, U.S.A., 1988.

[Mah93] Michael Maher. A logic programming view of CLP. In *Proceedings of the 10th International Conference on Logic Programming*, pages 737–753, 1993.

[Mal71] A. I. Mal'cev. Axiomatizable classes of locally free algebras of various types. In *The Metamathematics of Algebraic Systems: Collected Papers, 1936-1967*, pages 262–289. North-Holland, 1971.

[Man93a] Suresh Manandhar. CUF in context. Technical report, DYANA Deliverable, 1993.

[Man93b] Suresh Manandhar. *Relational Extensions to Feature Logic: Applications to Constraint-Based Grammars.* PhD thesis, University of Edinburgh, 1993.

[Man94] Suresh Manandhar. An attributive logic of set descriptions and set operations. In *32th Annual Meeting of the Association for Computational Linguistics*, 1994.

[McC56] E. McCluskey. Minimization of boolean functions. *Bell Sys. Tech. Journal*, 35:1417–1444, 1956.

[MK91] Maxwell III, John T. and Ronald M. Kaplan. A method for disjunctive constraint satisfaction. In Massaru Tomita, editor, *Current Issues in Parsing Technology.* Kluwer Academic Publishers, 1991.

[MM82] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, April 1982.

[MN70] Z. Manna and S. Ness. On the termination of Markov algorithms. In *Procceedings of the 3rd Hawaii International Conference on System Science*, pages 789–792, 1970.

[Mon74] R. Montague. *Formal Philosophy.* Yale Universit Press, New Haven, 1974.

[Moo88] Michael Moortgat. *Categorial Investigations: Logical and Linguistic Aspects of Lambek Calculus.* Foris, Dordrecht, 1988.

[Moo89] R. C. Moore. Unification-based semantic interpretation. In *27th Annual Meeting of the Association for Computational Linguistics*, pages 33–41, Vancouver, British Columbia, 1989. Association for Computational Linguistics.

[Mor89a] Glyn Morrill. Grammar as logic. Technical report, University of Edinburgh, Centre for Congnitive Science, 1989.

[Mor89b] Glyn Morrill. Intensionality, boundness and modal logic. Technical report, University of Edinburgh, Centre for Congnitive Science, 1989.

[Mor94] Glyn Morrill. Higher-order linear logic programming of categorial deduction. Report de recerca lsi-94-42-r, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 1994.

[MR87] D. Moshier and W.C. Rounds. A logic for partially specified data structures. In *Fourteenth ACM Symposium on Principles of Programming Languages*, pages 156–167, 1987.

[Nga93] T. Ngair. A new algorithm for incremental prime implicate generation. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery, France*, 1993.

[NM88] Gopalan Nadathur and Dale Miller. An overview of λProlog. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Logic Programming: Proceedings of the Fifth International Conference and Symposium*, volume 1, pages 810–827, Seattle, WA, 1988. MIT Press.

[NO80] G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *JACM*, 27(2):245–257, 1980.

[Pen93] M. Pentus. Lambek grammars are context free. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*, pages 429–433, Los Alamitos, California, 1993. IEEE Computer Society Press.

[Per87] Fernando C.N. Pereira. Grammars and logics of partial information. In *Logic Programming: Proceedings of the 4th International n Conference*, pages 989–1013. MIT Press, 1987.

[Per90] Fernando C.N. Pereira. Categorial semantics and scoping. *Computational Linguistics*, 16(1), 1990.

[Per91] Fernando C.N. Pereira. Semantic interpretation as higher-order deduction. In J. van Eijck, editor, *Logic in AI: European Workshop JELIA'90*, volume 478, pages 78–96, Amsterdam, The Netherlands, 1991. Springer-Verlag.

[PG82] Geoffrey K. Pullum and Gerald Gazdar. Natural languages and context-free languages. *Linguistics and Philosophy*, 4, 1982.

[PM90] Carl Pollard and M. Moshier. Unifying partial descriptions on sets. In Philip P. Hanson, editor, *Information, Language and Cognition*, number 1 in Vancouver Studies in Cognitive Science. University of British Columbia Press, 1990.

[PR94] Andreas Podelski and Peter Van Roy. The beauty and beast algorithm: Quasi-linear incremental tests of entailment and disentailment over trees. In *Logic Programming: Proceedings of the 11th International Conference*, pages 359–374, 1994.

[Prz89] T. C. Przymusinski. An algorithm for circumscription. *AI Journal*, 38:49–73, 1989.

[PS84] Fernando C.N. Pereira and Stuart M. Shieber. The semantics of grammar formalisms seen as computer languages. In *Proceedings of the 10th International Conference on Computational Linguistics (COLING)*, Stanford, 1984.

[PS87a] Fernando C.N. Pereira and Stuart M. Shieber. *Prolog and Natural Language Analysis*. Center for the Study of Language and Information Stanford, 1987.

[PS87b] Carl Pollard and Ivan Sag. *Information Based Syntax and Semantics, Volume 1, Fundamentals*, volume 13. Center for the Study of Language and Information Stanford, 1987.

[PS94] Carl Pollard and Ivan Sag. *Information Based Syntax and Semantics, Volume 2*. Center for the Study of Language and Information Stanford, 1994.

[PW80] Fernando C.N. Pereira and David Warren. Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13, 1980. reprinted in [GJW86].

[PW83] Fernando C.N. Pereira and David Warren. Parsing as deduction. In *21st Annual Meeting of the Association for Computational Linguistics*, Cambridge Massachusetts, 1983.

[Qui52] W. V. Quine. The problem of simplifying truth functions. *American Math. Monthly*, 59(8):521–531, 1952.

[Qui55] W. V. Quine. A way to simplify truth functions. *American Math. Monthly*, 62:627–631, 1955.

[Qui59] W. V. Quine. On cores and prime implicants of truth functions. *American Math. Monthly*, 66:755–760, 1959.

[Rea91] Mike Reape. An introduction to the semantics of unification-based formalisms. Dyana, deliverable r3.2.a, University of Edinburgh, 1991.

[Rea94] Mike Reape. A feature value logic with intensionality, nonwellfoundedness and functional and relational dependencies. In C. Rupp, M. A. Rosner, and R. L. Johnson, editors, *Constraints, Language, and Computation*, Computation in Cognitive Science, pages 78–110. Academic Press, London, 1994.

[Rei87] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[RK86] W.C. Rounds and R.T. Kasper. A complete logical calculus for record structures representing linguistic information. In *Proceedings of the First Annual IEEE Symposium on Logic in Computer Science*. IEEE, 1986.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12, 1965.

[Rou88] W.C. Rounds. Set values for unification-based grammars. Technical report, Center for the Study of Language and Information Stanford, Standford, CA, 1988.

[Rou97] William C. Rounds. Feature logics. In J. van Benthem and Ter Meulen, editors, *Handbook of Logic and Language*, chapter 1. Elsevier Science Publishers, 1997.

[RvH93] Viswanath Ramachandran and P. van Hentenryck. Incremental algorithms for constraint solving and entailment over racional trees. In *Proceedings of the 13th conf. Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes on Computer Science*, pages 205–217, 1993.

[Rym94] Ron Rymon. On kernel rules and prime implicants. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-94)*, 1994.

[Sar93] V. Saraswat. *Concurrent Constraint Programming*. MIT Press, 1993.

[Sco82] Dana Scott. Domains for denotational semantics. In *ICALP 82*. Springer-Verlag, 1982.

[Sel85] Peter Sells. *Lectures on Contemporary Syntactic Theories*. CSLI, Stanford, 1985.

[Shi85] Stuart M. Shieber. Evidence against the non-context-freeness of natural language. *Linguistics and Philosophy*, 8, 1985.

[Shi86] Stuart M. Shieber. *Introduction to Unification-Based Approaches to Grammar*. Center for the Study of Language and Information Stanford, 1986.

[Shi88] Stuart M. Shieber. A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, Budapest, 1988.

[Sho67] J. Shoenfield. *Mathematical Logic*. Addison-Wesley, Reading, Mass., 1967.

[SM54] E. W. Samson and Burton. E. Mills. Circuit minimization: Algebra and algorithm for new boolean canonical expressions. Technical Report 54-21, Air Force Cambridge Res. Center, April 1954.

[Smi91] Donald Smith. Constraint operations for CLP($\mathcal{FT}$). In Koichi Furukawa, editor, *Logic Programming: Proceedings of the 8th International Conference*, pages 760–774. MIT Press, 1991.

[Smo89a] Gert Smolka. Feature constraint logics for unification grammars. Technical report, IBM Wissenschafliches Zentrum, Institut für Wissensbasierte Systeme, 1989. IWBS Report 93.

[Smo89b] Gert Smolka. Feature logic with subsorts. Technical report, IBM Wissenschafliches Zentrum, Institut für Wissensbasierte Systeme, 1989. LILOG Report 33.

[Smo92] Gert Smolka. Feature constraint logics for unification grammars. *Journal of Logic Programming*, 12:51–87, 1992.

[ST92] Gert Smolka and Ralf Treinen. Records for logic programming. In Krzysztof Apt, editor, *Logic Programming: Proceedings of the 9th International Conference*. MIT Press, 1992.

[ST94] Gert Smolka and Ralf Treinen. Records for logic programming. *Journal of Logic Programming*, 18(3):229–258, April 1994.

[Ste85] Marc Steedman. Dependency and coordination in the grammar of Dutch and English. *Language*, 61, 1985.

[Ste88] Mark Steedman. Combinators and grammars. In R. Oehrle et al., editor, *Categorial Grammars and Natural Language Structures*, pages 417–442. D.Reidel Publishing, 1988.

[Ste90] Mark Steedman. Gapping as constituent coordination. *Linguistics and Philosophy*, 13:207–263, 1990.

[Sto77] J. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. MIT Press, 1977.

[SUP+83] Stuart M. Shieber, Hans Uszkoreit, Fernando C.N. Pereira, J. Robinson, and M. Tyson. The formalism and implementation of PATR-II. In B. J. Grosz and M. E. Stickel, editors, *Research on Interactive Acquisition and Use of Knowledge*. SRI report, 1983.

[Sza87] Anna Szabolcsi. Bound variables in syntax. In *Proceedings of the Sixth Amsterdam Colloquium*. Institute for Language, Logic and Information, University of Amsterdam, 1987.

[Tai67] W. W. Tait. Intensional interpretation of functionals of finite type I. *Journal of Symbolic Logic*, 32:198–212, 1967.

[Tra95] Marten Trautwein. Assessing complexity results in feature theories. Technical report, University of Amsterdam, 1995.

[Tre93] Ralf Treinen. On feature constraints with variable feature access. Technical report, DFKI, 1993.

[Tre95] Ralf Treinen. Feature trees over arbitrary structures. Technical report, DFKI, 1995.

[Usz86] Hans Uszkoreit. Categorial unification grammar. In *Proceedings of the 11th International Conference on Computational Linguistics (COLING)*, Bonn, 1986.

[vB83] J. van Benthem. The semantics of variety in Categorial Grammar. In W. Buszkowski, W. Marciszewski, and J. van Benthem, editors, *Categorial Grammar*. Benjamins, Amsterdam, 1983.

[vB91] J. van Benthem. *Language in Action.Categories, Lambdas and Dynamic Logic*, volume 130 of *Studies in Logic*. North Holland, 1991.

[vHSD92a] P. van Hentenryck, V. Saraswat, and Y. Deville. Constraint logic programming over finite domains- the design, implementation and applications of cc(FD). Technical report, Brown University, 1992.

[vHSD92b] P. van Hentenryck, H. Simonis, and M. Dincbas. Constraint satisfaction using constraint logic programming. *Artificial Intelligence*, 58:113–159, 1992.

[VS87] K. Vijay-Shanker. *A Study in Tree Adjoining Grammars*. PhD thesis, University of Pennsylvania, Pennsylvania, 1987.

[VSJ88] K. Vijay-Shanker and A. K. Joshi. Feature structure based tree adjoining grammar. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, pages 714–719, Budapest, 1988.

[Woo70] W. A. Woods. Transition network grammars for natural language. *CACM*, 13(10), 1970.

[YBHS60] C. Gaifman Y. Bar-Hillel and E. Shamir. On categorial and phrase structure grammars. *Bull. Res. Council Israel, sec. F*, 9:1–16, 1960. Reprinted in: Bar-Hillel, Y. *Language and Information. Selected Essays on Their Theory and Application*, Addison-Wesley, Reading. Mass. 1964, 99-115.

[Zaj92] Rémi Zajac. Inheritance and constraint-based grammar formalisms. *Computational Linguistics*, 18(2):159–182, 1992.

[Zie81] Wojciech Zielonka. Axiomatizability of Ajdukiewicz-Lambek calculus by means of cancellation schemes. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 27:215–224, 1981.

[ZKC87] Henk Zeevat, Ewan Klein, and Jo Calder. Unification categorial grammar. In Nicholas Haddock, Ewan Klein, and Glyn Morrill, editors, *Categorial Grammar, Unification Grammar and Parsing.* Centre for Cognitive Science, University of Edinburgh, 1987. Volume 1 of Working Papers in Cognitive Science.

Índice Remissivo