Planejamento como satisfatibilidade: uma abordagem não-clausal

Razer Montaño[1], Fabiano Silva[1], Marcos Castilho[1], Luis Künzle[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Centro Politécnico – Jardim das Américas
Caixa Postal 19.081 – 81.531-980 – Curitiba – PR – Brazil

{razer, fabiano, marcos, kunzle}@inf.ufpr.br

Abstract. *The proposed approach consists in resolve a planning problem as reachability in Petri nets. Initially, the problem is translated in an acyclic Petri net. A non-clausal SAT formula is obtained from mutex among actions, explicited in the net structure. The SAT formula resolution is made with Tableaux KE and corresponds to decide all mutex in the net. Finally, to resolve reachability in this acyclic conflict-free Petri net is polinomial.*

Resumo. *A abordagem proposta consiste em resolver um problema de planejamento como alcançabilidade em redes de Petri. Inicialmente, traduz-se o problema em uma rede Petri acíclica. Uma fórmula SAT não-clausal é obtida a partir dos* mutex *entre as ações, explícitas na estrutura da rede. A resolução da fórmula SAT é feita através de Tableaux KE e corresponde a decidir todos os* mutex *na rede. Finalmente, resolver alcançabilidade nesta rede acíclica e sem conflitos é polinomial.*

1. Introdução

Um problema de planejamento clássico em inteligência artificial é dado por um estado inicial, um objetivo e um conjunto de ações. Resolver este problema é encontrar uma seqüência de ações que, ao serem executadas, transformam o estado inicial no estado objetivo. Os programas que resolvem estes problemas são os planejadores.

O relacionamento entre planejamento e alcançabilidade em Redes de Petri foi verificado em [Silva et al. 2000] através do algoritmo PETRIPLAN, possibilitando que um grafo de planos possa ser transformado facilmente em uma RdP ordinária e limitada [Murata 1989]. Isto é vantajoso em termos de representação, visto que se consegue mapear, através da dinâmica da própria rede, as pré-condições e efeitos da execução de uma determinada ação, bem como seus conflitos (*mutex*) sem o uso de estruturas auxiliares.

Contudo, resolver o problema de planejamento nesta modelagem é resolver o problema de alcançabilidade em RdP que em geral é um processo exponencial [Esparza 1996]. Não se conhece ainda algoritmo eficiente para este problema, em especial se for considerado o tamanho das redes que modelam um problema de planejamento, que normalmente são matrizes com milhares/milhões de nodos, os algoritmos se mostram ineficientes tornando impraticável qualquer solução computacional efetiva.

Através da análise de uma RdP gerada a partir de um problema são obtidas as informações sobre todos os conflitos que incidem sobre a execução de ações. Conflitos

são pontos de escolha, onde duas ações são mutuamente exclusivas em sua execução. Caso se obtenha uma RdP sem conflitos, então não há escolhas a serem feitas e a solução é trivial e rápida de ser determinada, bastando uma varredura na rede. Em RdP sem conflitos e acíclicas o problema de alcançabilidade é polinomial [Esparza 1996].

Aqui mostra-se um novo método para resolver o problema de planejamento modelado como uma RdP, baseado na eliminação de conflitos desta rede [Montaño 2006]. Para isto, a partir da RdP gerada pelo PETRIPLAN, gera-se uma fórmula lógica na *Negation Normal Form* (NNF) que representa a relação entre o estado objetivo e os conflitos na rede. A resolução destes conflitos na rede se dá a partir de um resolvedor SAT para fórmulas não-clausais, que dará valores aos literais desta fórmula. Este resolvedor SAT é baseado no procedimento de Tableau KE [R. Hähnle and N. Murray and E. Rosenthal 2005], usado para transformar uma fórmula lógica de NNF para *Factored Negation Normal Form* (FNNF). Na estrutura da RdP, a valoração retornada pelo procedimento de SAT não-clausal indica a eliminação dos conflitos através da imposição de uma escolha.

Em contraste com o `Blackbox` [Kautz and Selman 1999], a abordagem aqui apresentada descarta o passos de propagação de *mutex*, visto que a própria dinâmica da rede de Petri representa a propagação das restrições. Usa-se também, como representação de conflitos na rede, os *mutex* de ações pois estes são naturalmente representados pela rede de Petri sem que haja necessidade de propagação.

Na seção 2 descreve-se a teoria de redes de Petri. Na seção 3 trata-se dos métodos para SAT não-clausal, enquanto que na seção 4 estes resultados são aplicados ao problema de alcançabilidade em RdP.

2. Redes de Petri, Alcançabilidade e PETRIPLAN

As Redes de Petri (RdP) são um modelo matemático de especificação formal de sistemas discretos, possibilitando a representação e análise matemática de problemas baseados em estados e eventos, permitindo a verificação de propriedades, por exemplo conflitos, paralelismo, concorrência e corretude dos sistemas.

Definição 1 (Rede de Petri) *Uma RdP é uma quíntupla* $N =< P, T, Pre, Pos, M_0 >$*, onde* P *e* T *são conjuntos finitos e disjuntos de lugares e transições,* Pre *é uma função de incidência de entrada nas transições,* Pos *uma função de incidência de saída das transições e* M_0 *é uma marcação inicial.*

O vetor $Pre(.,\ t)$ representa todos os arcos de entrada da transição t, com seus pesos. De forma análoga, o vetor $Pos(.,\ t)$ representa todos arcos de saída da transição t, com seus pesos.

A dinâmica da RdP é dada pelos disparos de transições habilitadas. Uma transição está habilitada se todas as suas pré-condições forem atendidas, isto é, numa rede marcada, todos os lugares que possuem arcos para esta determinada transição devem possuir um número suficiente de marcas, conforme o peso do arco. Esta dinâmica é dada por $M' = M + Pos(.,t) - Pre(.,t)$. Esta equação é chamada *equação fundamental* de N.

O comportamento dinâmico de uma RdP N é definido pelo conjunto de marcações acessíveis. O problema de alcançabilidade em RdPs, baseado numa marcação inicial M_0, é verificar se uma determinada marcação é alcançável a partir de M_0.

Uma marcação M_g é dita alcançável a partir de M, se, e somente se, existe uma seqüência de transições s tal que seu disparo leva o estado da rede de M a M_g.

O PETRIPLAN [Silva et al. 2000] é um algoritmo de planejamento que utiliza RdP como ferramenta de modelagem. A partir da linguagem descritiva de problemas de planejamento PDDL, o PETRIPLAN gera uma RdP acíclica, na qual se torna possível a aplicação de métodos de alcançabilidade, em especial buscas e métodos de programação inteira (baseada na equação fundamental). Quando a solução é encontrada, é convertida para a representação de planos, resolvendo assim o problema de planejamento.

3. SAT Não-Clausal

Originalmente SAT é definido para fórmulas em CNF e a grande maioria das aplicações se baseia nesta forma normal. Entretanto, muitos problemas, quando são analisados como satisfatibilidade, geram fórmulas que não estão em CNF, sendo necessária uma conversão prévia para que se trate o problema como SAT tradicional.

Converter uma fórmula qualquer para CNF se faz pela aplicação sucessiva da operação distributiva e leis de De Morgan sobre os conectivos lógicos. Esta transformação é especialmente cara computacionalmente. O processo como um todo, conversão para CNF e SAT para CNF eleva drasticamente o tempo de execução e consumo de memória do resolvedor, sendo uma alternativa o estudo de resolvedores SAT para fórmulas não-clausais.

Resolver um problema SAT para um conjunto de cláusulas é computacionalmente intratável sem o uso de heurísticas ou métodos mais elaborados (SAT é NP-Completo). Existem procedimentos para solução de problemas SAT para fórmulas não-clausais muito eficientes. Por exemplo, segundo [Darwiche and Marquis 2002], se a fórmula estiver na *Deterministic Decomposable Negation Normal Form* (d-DNNF), pode-se fazer o teste de satisfatibilidade, contagem de modelos (*Model Counting*) e enumeração de modelos (*Model Enumeration*) em tempo polinomial.

Fórmulas na d-DNNF são um caso particular de fórmulas na NNF. Uma sentença está na NNF se é construída a partir de literais usando-se somente disjunções e conjunções. Uma representação prática de sentenças na NNF usa um grafo acíclico dirigido (DAG), que possui um nodo raiz, onde cada nodo folha é rotulado com valor lógico V (*Verdadeiro*), ou F (*Falso*) ou um literal, e cada nodo interno é rotulado com disjunção ($\vee$) ou conjunção ($\wedge$), e pode ter qualquer quantidade de filhos.

As definições seguintes mostram as propriedades que devem ser atendidas para uma fórmula na NNF estar também na d-DNNF.

Definição 2 (Decomponibilidade) *Dada uma fórmula na NNF, para cada conjunção $\alpha_1 \wedge \ldots \wedge \alpha_n$, se uma variável não aparece em mais de um termo α_i, isto é, os átomos encontrados em um termo desta conjunção não são os mesmos encontrados em outros termos, então ela é decomponível.*

Uma fórmula está na *Decomposable Negation Normal Form* (DNNF) se está na NNF e satisfaz a propriedade de decomponibilidade.

Uma fórmula decomponível permite que certos tipos de problemas computacionais sejam decompostos em problemas menores, dado que suas subsentenças não com-

partilham átomos. Sempre que um nodo and for encontrado, cada um de seus ramos terá, seguramente, informações diferentes, podendo ser computado de forma independente.

Decomponibilidade é a propriedade que torna DNNF computacionalmente tratável para o teste de satisfatibilidade. Este teste é como segue:

- se α é um literal, então α é satisfatível;
- se $\alpha = \alpha_1 \vee ... \vee \alpha_n$ (também representado por $\vee_i \alpha_i$), então α é satisfatível se, e somente se, *algum* α_i o for;
- se $\alpha = \alpha_1 \wedge ... \wedge \alpha_n$ (também representado por $\wedge_i \alpha_i$), então α é satisfatível se, e somente se, *todos* os α_i forem.

Analisando-se a definição acima, percebe-se que esta operação é $\mathcal{O}(n)$, onde n é o número de literais da fórmula.

Definição 3 (Determinismo) *Dado uma fórmula na NNF, se para toda disjunção $\vee_i \alpha_i$, todo par de termos α_i e α_j, onde $i \neq j$, $\alpha_i \wedge \alpha_j$ é uma contradição, isto é, são mutuamente exclusivos, então a fórmula é determinística.*

Uma fórmula está na *Deterministic Negation Normal Form* (d-NNF) se está na NNF e satisfaz a propriedade de determinismo.

É a propriedade de determinismo que torna a contagem de modelos e sua enumeração tratável, isto é, o número de modelos de uma d-DNNF $\wedge_i \alpha_i$ é o produto do número de modelos de cada termo α_i e o número de modelos de uma d-DNNF $\vee_i \alpha_i$ é a soma do número de modelos de cada termo α_i.

Uma fórmula que satisfaz as propriedades de determinismo e decomponibilidade está na *Deterministic Decomposible Negation Normal Form* (d-DNNF).

Da mesma forma, os modelos de uma fórmula na d-DNNF é a união dos modelos dos termos $\vee_i \alpha_i$ e o produto cartesiano entre os modelos dos termos $\wedge_i \alpha_i$. A enumeração de modelos é tratada detalhadamente mais adiante.

Para se transformar uma teoria proposicional em d-DNNF deve-se aplicar, recursivamente, uma operação chamada *condicionamento*[1] [Palacios et al. 2005]. Esta operação é derivada da identidade conhecida como expansão de Shannon dada por:

$$\mathcal{F} = (\mathcal{F}[V/p] \wedge p) \vee (\mathcal{F}[F/\neg p] \wedge \neg p)$$

onde $\mathcal{F}[x/p]$ indica a troca de todas as ocorrências de p em $\mathcal{F}$ por x.

O condicionamento de Δ sobre o literal α, escrito $\Delta \mid \alpha$, é obtido pela troca de cada folha α no DAG por V e cada folha $\neg\alpha$ por F. Ex.: se $\Delta = (p \vee q) \wedge (\neg p \vee r)$, então $\Delta \mid p$ resulta em $(V \vee q) \wedge (F \vee r)$. A identidade que representa a expansão de Shannon pode ser vista em termos de condicionamento como:

$$\mathcal{F} = (\mathcal{F} \mid p \ \wedge \ p) \vee (\mathcal{F} \mid \neg p \ \wedge \ \neg p).$$

Portanto, para transformar uma teoria proposicional Δ em d-DNNF, basta:

[1] Do inglês *conditioning*

- enumerar todas as variáveis de Δ, sendo elas $x_1, x_2, ..., x_n$;
- aplicar o condicionamento em x_1, obtendo $(\Delta \,|\, x_1 \ \wedge \ x_1) \vee (\Delta \,|\, \neg x_1 \ \wedge \ \neg x_1)$;
- aplicar condicionamento em $x_2, x_3, ..., x_n$ para $\Delta \mid x_1$ e $\Delta \mid \neg x_1$.

Segundo [Palacios et al. 2005], esta transformação é correta e completa, gerando fórmulas equivalentes.

Se na geração da d-DNNF, a cada passo de condicionamento, forem aplicadas as seguintes identidades:

- $p \vee p$ troca-se por p
- $p \wedge p$ troca-se por p
- $p \vee F$ troca-se por p
- $p \wedge V$ troca-se por p
- $p \vee V$ troca-se por V
- $p \wedge F$ troca-se por F
- $p \vee \neg p$ troca-se por V
- $p \wedge \neg p$ troca-se por F

o que se obtém é uma fórmula que está na FNNF (*Factored Negation Normal Form*). Uma fórmula na FNNF também está na d-DNNF, visto que estas simplicações somente diminuem seu tamanho, gerando fórmulas equivalentes.

A FNNF [R. Hähnle and N. Murray and E. Rosenthal 2005] foi introduzida para expressar a aplicação de simplificações no processo de condicionamento. Estas simplificações promovem vários cortes na fórmula, durante o processo de geração da d-DNNF, o que leva à geração de uma fórmula menor e equivalente. Caso a fórmula seja uma contradição, será reduzida a F. Analogamente, sendo uma tautologia, será reduzida a V. Esta representação em FNNF é canônica.

Segundo [Darwiche and Marquis 2002], a operação de enumeração de modelos para uma fórmula na d-DNNF pode ser efetuada em tempo polinomial. Por causa das propriedades de decomponibilidade e determinismo, basta varrer a fórmula uma vez em busca de todos os modelos. Convém ressaltar que este procedimento é o mesmo para uma fórmula na FNNF, visto que uma fórmula na FNNF também está na d-DNNF.

Seja um modelo representado como um conjunto de literais e suponha que N_i e M_i sejam modelos para uma fórmula (ou subfórmula). O produto cartesiano ($\times$) entre modelos define-se como:

$$\{N_1, ..., N_n\} \times \{M_1, ..., M_m\} \equiv N_1 \cup M_1, N_1 \cup M_2, ..., N_n \cup M_m$$

O conjunto $Models(\Delta)$, conjunto de modelos de uma teoria proposicional Δ, é definido como:

$$Models(\Delta = literal) = \Delta$$
$$Models(\Delta = \bigvee_i \alpha_i) = Models(\alpha_1) \cup Models(\alpha_2) \cup ... \cup Models(\alpha_n)$$
$$Models(\Delta = \bigwedge_i \alpha_i) = Models(\alpha_1) \times Models(\alpha_2) \times ... \times Models(\alpha_n)$$

Se Δ está na d-DNNF e w é uma atribuição de verdade sobre os átomos de Δ, então $w \models \Delta$ se, e somente se, $w \in Models(\Delta)$. A complexidade desta operação é

$\mathcal{O}(mn)$, onde m é o tamanho de Δ e $n = |\, Models(\Delta)\,|^2$ [Darwiche 1998]. Portanto, se o número de modelos é pequeno o suficiente para ser visto como uma constante, o processo de enumeração leva um tempo linear sobre o tamanho da fórmula.

Um algoritmo que percorre uma fórmula na d-DNNF pode ser construído através do caminhamento no DAG correspondente. Parte-se do nó raiz, retornando o modelo para a fórmula lógica que contém o nó visitado como raiz. Se o nó visitado é literal, este é retornado como modelo. Se o nó é conjunção, o modelo é o produto cartesiano entre os modelos de seus filhos, e sendo disjunção, retorna a união dos modelos dos filhos.

4. Alcançabilidade em Redes de Petri via SAT para Formas Não-Clausais

A estrutura de representação usada aqui para modelar o problema de planejamento é uma rede de Petri Lugar/Transição, obtida pelo algoritmo PETRIPLAN [Silva et al. 2000], cuja geração é similar à do grafo de planos. Esta rede gerada é acíclica, cada transição pode disparar no máximo uma vez e todos os lugares possuem no máximo dois arcos de saída. Resolver o problema de planejamento clássico é equivalente a se resolver o problema de alcançabilidade de submarcação nesta rede.

A principal diferença entre o grafo de planos e a rede de Petri está no fato dos conflitos (*mutex*) serem representados como meta-informações no grafo, enquanto que na rede de Petri eles são parte da rede. Ainda, o grafo de planos é uma estrutura estática e os métodos de busca devem percorrer a estrutura até encontrar um caminho livre de conflitos. Na rede de Petri existe uma dinâmica associada, fato que permite reduzir sensivelmente o número de conflitos na rede em relação ao grafo de planos.

Esta rede representa o comportamento dinâmico da aplicação de ações ao longo do tempo, partindo-se de uma situação inicial em busca de um estado final. Resolver um problema de planejamento usando esta RdP é encontrar todas as transições que devem ser disparadas, na ordem correta, para que, a partir de uma marcação inicial da RdP se consiga outra que contenha o estado objetivo. Esta seqüência de ações é chamada *plano*.

Para se chegar a um plano pela análise de alcançabilidade em RdP, deve-se tratar os conflitos desta rede. Conflitos em uma RdP indicam que existem transições que não podem ser disparadas para a obtenção do mesmo plano, isto é, são mutuamente exclusivas. Assim, uma decisão consistente destes conflitos é dada pelo conjunto de escolhas feitas sobre quais transições deveriam ser disparadas.

O objetivo é resolver os conflitos de uma dada RdP baseando-se em SAT para se obter uma RdP livre de conflitos. Se isto ocorrer, para se encontrar as transições necessárias para alcançar o estado final a partir do estado inicial, bastaria simular seus disparos na rede sem que haja nenhuma escolha a ser feita.

O procedimento proposto pode ser visualizado através do Algoritmo 1, abaixo.

4.1. Geração de fórmulas em NNF a partir de uma Rede de Petri

A técnica proposta aqui é, a partir da RdP original, partindo de cada lugar pertencente ao estado objetivo, varre-se a rede até os lugares iniciais construindo uma fórmula lógica onde seus átomos representam unicamente os conflitos. A retirada dos conflitos da rede se dá a partir de um resolvedor SAT para fórmulas não-clausais, que dará valores

Algoritmo 1 Planejador

```
Carrega o PDDL
repeat
  Expande a rede até que os objetivos sejam encontrados consistentemente
  Gera a fórmula na NNF
  Transforma a fórmula para FNNF
  Aplica resolvedor SAT para FNNF
  if resolvedor SAT encontrou valoração para a fórmula then
    Constrói o plano a partir da valoração
  end if
until encontrar plano
```

aos literais desta fórmula. Na estrutura da RdP, esta valoração indica a eliminação dos conflitos através da imposição de uma escolha.

Nas RdP geradas pelo PETRIPLAN usado nas implementações, os conflitos são sempre binários (um lugar possuindo no máximo dois arcos de saída), portanto pode-se usar uma variável proposicional para representar cada conflito. Por exemplo, na Figura 1 $\neg p$ indica o disparo da transição t_1 e p indica o disparo da transição t_2.

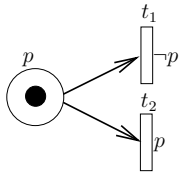

Figure 1. Conflito com a proposição p associada

Analisando-se a estrutura da RdP, percebe-se que os lugares pertencentes ao estado final podem ser descritos como uma fórmula baseada nas variáveis proposicionais associadas aos conflitos. Caso cada um dos lugares que chegam a t tenham fórmulas associadas ($\Delta_1, ..., \Delta_n$), como mostra a Figura 2, a fórmula associada a t é $\Delta_1 \wedge ... \wedge \Delta_n$. Caso cada uma das transições que chegam a p tenham fórmulas associadas ($\Delta_1, ..., \Delta_n$), como mostra a Figura 3, a fórmula associada a p é $\Delta_1 \vee ... \vee \Delta_n$.

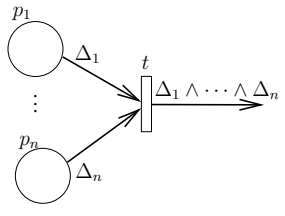

Figure 2. Conjunção na RdP com fórmula associada

Seja M_o a marcação final e M_i a marcação inicial da RdP. Seja $\theta(V)$ a quantidade de elementos de V, onde V é um vetor, diferentes de zero e $\mu(t)$, definido sobre uma transição t, indicando se t é uma transição de um conflito, isto é, $\mu(t)$ é V se $\exists p,\ Pre(p,t) > 0\ \wedge\ \theta(Pre(p,.)) > 1$ e F caso contrário. Assume-se que existe uma ordenação qualquer entre as transições, que pode ser a ordem de inserção na fórmula.

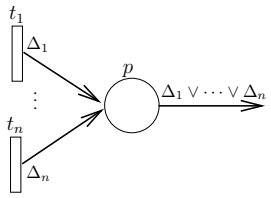

Figure 3. Disjunção na RdP com fórmula associada

A fórmula $\mathcal{F}$ que representa os conflitos de uma RdP é dada por:

$$\mathcal{F} \quad = \bigwedge_i L(p_i),\ p_i \in M_o$$

$$L(p) = \begin{cases} \bigvee_i T(t_i);\ \forall t_i,\ Pos(p,\ t_i)\ >\ 0 & se\ \theta(Pos(p,\ .))\ \geq\ 1 \\ V & se\ \theta(Pos(p,\ .))\ =\ 0\ e\ p\ \in\ M_i \\ F & se\ \theta(Pos(p,\ .))\ =\ 0\ e\ p\ \notin\ M_i \end{cases}$$

$$T(t) = \begin{cases} \bigwedge_i \alpha(p_i,\ t);\ \forall p_i,\ Pre(p_i,\ t)\ >\ 0 & se\ \theta(Pre(.,\ t))\ \geq\ 1 \\ F & se\ \theta(Pre(.,\ t))\ =\ 0 \end{cases}$$

$$\alpha(p,\ t) = \begin{cases} L(p) & se\ \neg\mu(t) \\ \beta(p,t)\ \wedge L(p) & se\ \mu(t) \end{cases}$$

$$\beta(p,\ t) = \begin{cases} p & se\ \exists t',\ Pre(p,\ t')\ >\ 0\ e\ t\ >\ t' \\ \neg p & se\ \exists t',\ Pre(p,\ t')\ >\ 0\ e\ t\ <\ t' \end{cases}$$

Como a fórmula gerada é composta somente por literais que representam os conflitos na rede, qualquer modelo (uma valoração que a satisfaça) é interpretado como uma seqüência de disparos não conflitantes. Caso um modelo não seja encontrado, então a disposição dos conflitos não permite que os objetivos sejam alcançados, indicando que não existe um plano para o problema de planejamento associado.

Em virtude da estrutura de geração da fórmula, percebe-se que os únicos conectivos presentes são disjunções e conjunções. Também percebe-se que as negações possuem seu escopo limitado aos átomos das fórmula. Portanto, a fórmula gerada está na NNF.

4.2. Geração da fórmula em FNNF através de Tableau KE

O tableau KE [D'Agostino and Mondadori 1994] é um procedimento de prova baseado em tableau, mas que inclui a regra de *cut*, que classicamente não é considerada. Conforme mostrado em [R. Hähnle and N. Murray and E. Rosenthal 2005] o procedimento de Tableau KE, pode ser usado para a geração da FNNF.

Este procedimento está intimamente ligado à expansão de Shannon, que é a transformação chave na obtenção da d-DNNF, segundo o método previamente apresentado. O tableau KE possui regras α e β, juntamente com a regra de *cut* que possui sua aplicação restrita às subfórmulas da fórmula original.

No procedimento de transformação para FNNF (Algoritmo 2), somente são usadas a regra *cut*, regras α e regras de transformação de tableau. As regras β são substituídas por um mecanismo de propagação de restrições, conhecido como *simplificação*

[Massacci 1998]. Este mecanismo tem o mesmo propósito que a subjugação para a resolução e que a regra de cláusulas unitárias para o procedimento de Davis-Putnam.

Algoritmo 2 FNNF

```
while nem todos os nodos são literais do
  Aplicar simplificações baseadas em tabela verdade
  Aplicar regra α
  if não foi possível aplicar a regra α then
    Aplicar regras de simplificação de fórmulas: expansão de Shannon
    if não foi possível aplicar a simplificação then
      Aplicar cut
    end if
  end if
  Resolve próximo nodo
end while
```

O tableau resultante terá literais em todos os nodos ou a árvore inteira será reduzida a F ou V. Este tableau é chamado *Tableau FNNF saturado*. Se na aplicação do *cut* for utilizado sempre a mesma ordem de variáveis, será obtido um FNNF ordenado (OFNNF). O algoritmo apresentado, utilizando-se de busca em profundidade com *backtracking* é exponencial [R. Hähnle and N. Murray and E. Rosenthal 2005].

Aqui, não há necessidade de se obter um tableau saturado pois qualquer ramo saturado deste tableau já é um modelo para a fórmula que está sendo processada. Este processo de parada caracteriza um o procedimento como um SAT não-clausal, sendo que do ramo saturado do Tableau KE se consegue extrair um modelo para a fórmula em NNF original. Esta valoração é transformada em cortes na RdP original, sendo que estes cortes incidem sobre os conflitos, visto que a fórmula é formada por informações provenientes destes. Estes cortes resolvem todos os conflitos da rede e, a partir deste ponto, pode-se aplicar um procedimentos de busca para encontrar o plano de forma eficiente, pois não será efetuado nenhum *backtracking*.

5. Considerações

Este trabalho se situa na intersecção entre quatro grandes áreas de pesquisa: Planejamento, Redes de Petri, SAT e Compilação do Conhecimento. Um problema de planejamento, modelado como uma RdP, é resolvido como um problema de alcançabilidade em RdP. As técnicas para se resolver este problema são computacionalmente intratáveis, sobretudo considerando-se o tamanho das redes aqui geradas.

Esta abordagem se diferencia do `Blackbox` pois usa redes de Petri como base para a construção do problema SAT que representa o problema de planejamento. Com este novo modelo, consegue-se representar naturalmente a estrutura dos *mutex*, não necessitando de operações de propagação.

O uso do Tableau KE na tentativa de se resolver problemas de alcançabilidade em tempo razoável não apresentou resultados favoráveis, pois a conversão para FNNF se mostrou ineficiente. As causas da explosão no uso de memória se dá por causa da aplicação do condicionamento, que em sua essência, faz uma duplicação da fórmula com

alguns literais valorados. O uso de outras técnicas de transformação não fazem uso excessivo de memória, e poderiam levar à solução satisfatória da transformação de NNF para FNNF.

Mesmo assim, esta nova abordagem permite uma boa perspectiva ao problema de planejamento e alcançabilidade, levando ao estudo de novas técnicas para sua solução, uma vez que as redes de Petri têm um bom potencial para uso em problemas de planejamento envolvendo noções temporais e de recursos.

References

D'Agostino, M. and Mondadori, M. (1994). The taming of the cut: Classical refutations with analytic cut. *Journal of Logic and Computation*, 4(3):285–319.

Darwiche, A. (1998). Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222.

Darwiche, A. and Marquis, P. (2002). A Knowledge Compilation Map. *Journal of Artificial Intelligence Research*, 17:229–264.

Esparza, J. (1996). Decidability and complexity of petri net problems - an introduction. In *Petri Nets*, pages 374–428.

Kautz, H. and Selman, B. (1999). Unifying SAT-based and graph-based planning. In Minker, J., editor, *Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14–16, 1999*, College Park, Maryland. Computer Science Department, University of Maryland.

Massacci, F. (1998). Simplification: A general constraint propagation technique for propositional and modal tableaux. *Lecture Notes in Computer Science*, 1397:217–231.

Montaño, R. (2006). Aplicação de Fórmulas Não-Clausais em Planejamento com Redes de Petri. Master's thesis, Universidade Federal do Paraná, Curitiba PR, Brasil.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

Palacios, H., Bonet, B., Darwiche, A., and Geffner, H. (2005). Pruning conformant plans by counting models on compiled d-DNNF representations. In *ICAPS*, pages 141–150.

R. Hähnle and N. Murray and E. Rosenthal (2005). Normal forms for knowledge compilation. *Lecture Notes in Computer Science*, 3488:304–313.

Silva, F., Castilho, M., and Künzle, L. (2000). Petriplan: a new algorithm for plan generation (preliminary report). In *Lecture Notes in Artificial Intelligence*, volume 1952, pages 86–95. International Joint Conference IBERAMIA'2000 - SBIA'2000.