On Applying Linear Tabling to Logic Programs

Miguel Areias
CRACS & INESC-Porto LA
Faculdade de Ciências
Univerdade do Porto, Portugal

FCUP-DCC & CRACS Porto, Portugal, Setembro 2010

Prolog e a Resolução SLD

- Os sistemas de Prolog são conhecidos por terem boas performances e boa flexibilidade, no entanto são baseados na resolução SLD, que limita todo o potencial do paradigma de programação em lógica.
- A resolução SLD não consegue lidar bem com as seguintes situações:
  - **Ciclos Infinitos Positivos** (falta de expressividade)
  - **Ciclos Infinitos Negativos** (Inconsistência)
  - **Computações Redundantes** (Ineficiência)

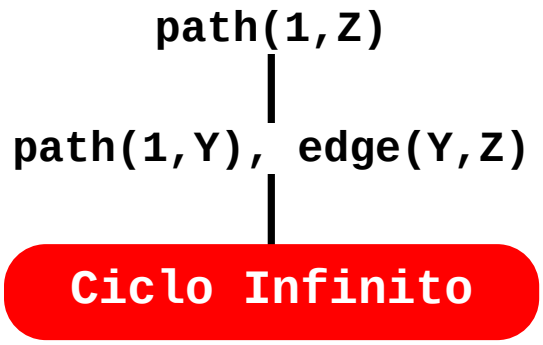
Resolução SLD : Ciclos Infinitos

```
path(X,Z) :- path(X,Y), edge(Y,Z).
path(X,Z) :- edge(X,Z).

edge(1,2).
edge(2,1).
```

```
path(1,Z)
```

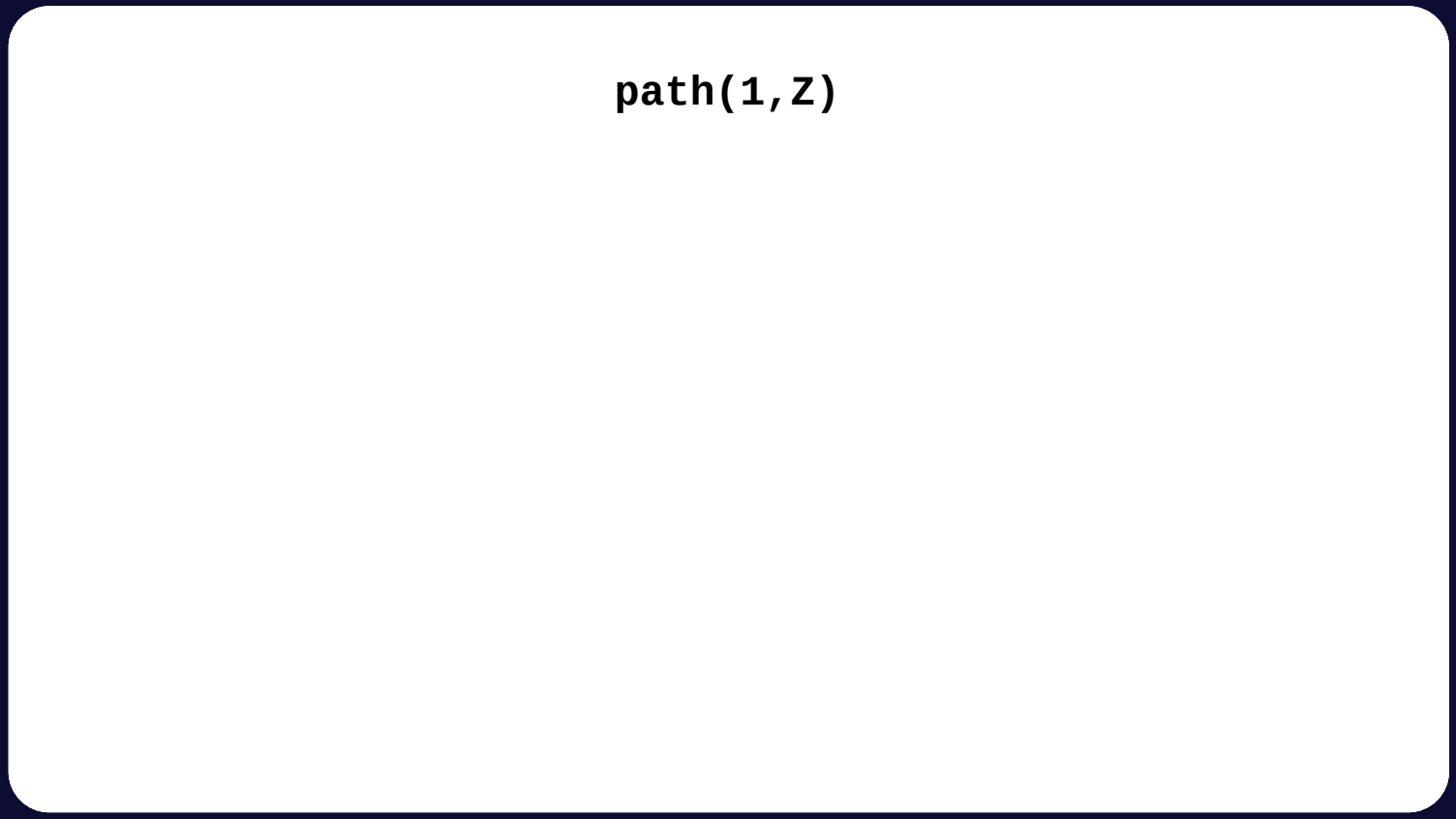
Resolução SLD : Ciclos Infinitos

```
path(X,Z) :- path(X,Y), edge(Y,Z).
path(X,Z) :- edge(X,Z).

edge(1,2).
edge(2,1).
```



Resolução SLD : Ciclos Infinitos

```
path(X,Z) :- edge(X,Y), path(Y,Z).
path(X,Z) :- edge(X,Z).

edge(1,2).
edge(2,1).
```



Resolução SLD : Ciclos Infinitos

```
path(X,Z) :- edge(X,Y), path(Y,Z).
path(X,Z) :- edge(X,Z).

edge(1,2).
edge(2,1).
```

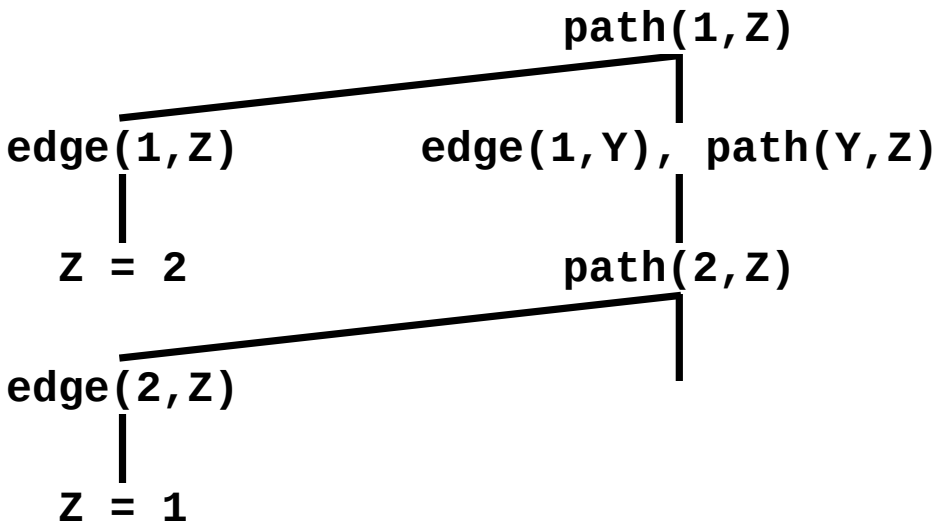
Resolução SLD : Ciclos Infinitos

```
path(X,Z) :- edge(X,Z).
path(X,Z) :- edge(X,Y), path(Y,Z).

edge(1,2).
edge(2,1).
```

```
path(1,Z)
```

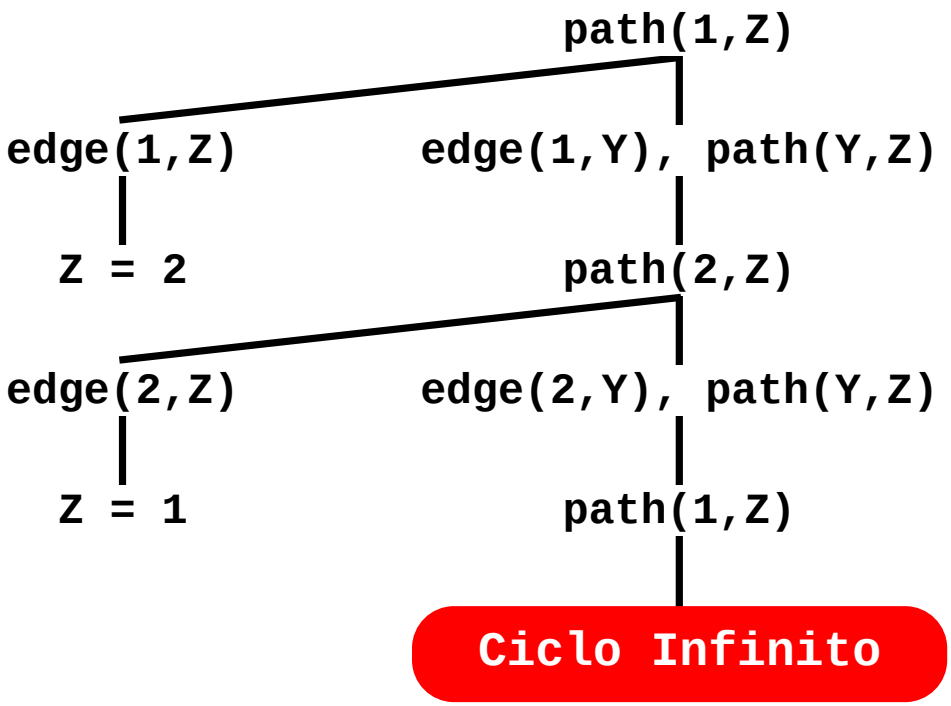
Resolução SLD : Ciclos Infinitos

```
path(X,Z) :- edge(X,Z).
path(X,Z) :- edge(X,Y), path(Y,Z).

edge(1,2).
edge(2,1).
```



Resolução SLD : Ciclos Infinitos

```
path(X,Z) :- edge(X,Z).
path(X,Z) :- edge(X,Y), path(Y,Z).

edge(1,2).
edge(2,1).
```



Tabulação na Programação em Lógica

- A tabulação é uma técnica de implementação que supera algumas das limitações da resolução SLD.
- As implementações de tabulação estão correntemente disponíveis em vários sistemas, tais como, o XSB Prolog, Yap Prolog, B-Prolog, ALS-Prolog, Mercury e mais recentemente no Ciao Prolog.

Tabulação na Programação em Lógica

- A tabulação é uma técnica de implementação que supera algumas das limitações da resolução SLD.
- As implementações de tabulação estão correntemente disponíveis em vários sistemas, tais como, o XSB Prolog, Yap Prolog, B-Prolog, ALS-Prolog, Mercury e mais recentemente no Ciao Prolog.
- Podemos distinguir as implementações de tabulação em duas grandes categorias:
  - **Por Suspensão de Execução**: pode ser visto como uma sequência de sub-computações que são suspensas e mais tarde retomadas, quando necessário, até que seja encontrado o ponto fixo da execução (XSB Prolog, Yap Prolog, Mercury e Ciao Prolog).
  - **Por Execução Linear**: pode ser visto como uma única árvore de execução em que os sub-golos tabelados são iterativamente computados até que seja encontrado o ponto fixo da computação, não recorrendo a quaisquer mecanismos de suspensão ou retoma da computação (B-Prolog e ALS Prolog).

Estratégias de Escalonamento

Ambas as categorias de tabulação podem usar dois diferentes tipos de escalonamento: **Local** e **Batched**. A diferença situa-se no comportamento do mecanismo quando este encontra uma **resposta nova** para um sub-golo tabelado.

Escalonamento	**Resposta Nova**	**Após a exploração de todas as alternativas que emparelhavam com a chamada ao sub-golo**
Local	Falha a computação para o ponto de escolha em uso.	Consome todas as respostas, propagando-as para o contexto da chamada anterior.
Batched	Consome a resposta, propagando-a imediatamente para o contexto da chamada anterior.	Falha a computação para o ponto de escolha anterior, uma vez que o actual já foi totalmente explorado.

Motivação

- Neste trabalho, iremos estender a actual plataforma **Yap Prolog**, para suportar uma implementação eficiente de **tabulação linear**, de forma a que esta seja a primeira plataforma a permitir a execução de ambos os mecanismos de tabulação.
- Para a nossa implementação linear ser o mais eficiente possível, ela irá reunir e inovar as optimizações lineares mais bem sucedidas até agora implementadas.

Tabulação Linear

- As duas estratégias lineares mais bem sucedidas até ao momento são:
  - **DRA (Dynamic Reordering of Alternatives)**: as tabelas guardam usadas para guardar as respostas dos sub-golos, passam a guardar também as alternativas que resultam em chamadas repetidas para os sub-golos. Estas cláusulas são denominadas de **looping alternatives** e são encontradas na primeira ronda de execução do sub-golo. Nas rondas seguintes essas alternativas são executadas até que seja encontrado o ponto fixo (ALS Prolog).
  - **DRE (Dynamic Reordering of Execution)**: as chamadas repetidas a um sub-golo são denominadas de **followers**. Estas chamadas executam as alternativas do programa que emparelham com o sub-golo que ainda não foram executadas, em vez de consumirem as respostas das tabelas conforme o mecanismo tradicional. Um **follower** é então re-executado sucessivamente até que o ponto fixo da computação seja encontrado, ou seja até todas as respostas e cláusulas terem sido exploradas (B-Prolog).

A Nossa Proposta

- Apresentamos então uma nova estratégia:
  - **DRS (Dynamic Reordering of Solutions)**: pode ser vista como uma variante da estratégia DRA, mas aplicada ao consumo de soluções. A ideia principal é memorizar as soluções de levam a chamadas consumidoras, denominadas de **looping solutions**, e usá-las como a estratégia DRA usa as **looping alternatives** [CoRTA 2010].
- Inovar as estratégias existentes:
  - Todas as estratégias anteriores irão escalonar as rondas de re-avaliação de chamadas tabeladas, de uma forma similar as estratégias baseadas em suspensão do Yap. Significa que em cada ronda de avaliação de um sub-golo, este apenas executará as alternativas do programa na primeira chamada, as seguintes consumiram apenas as soluções nas tabelas [PADL 2010].

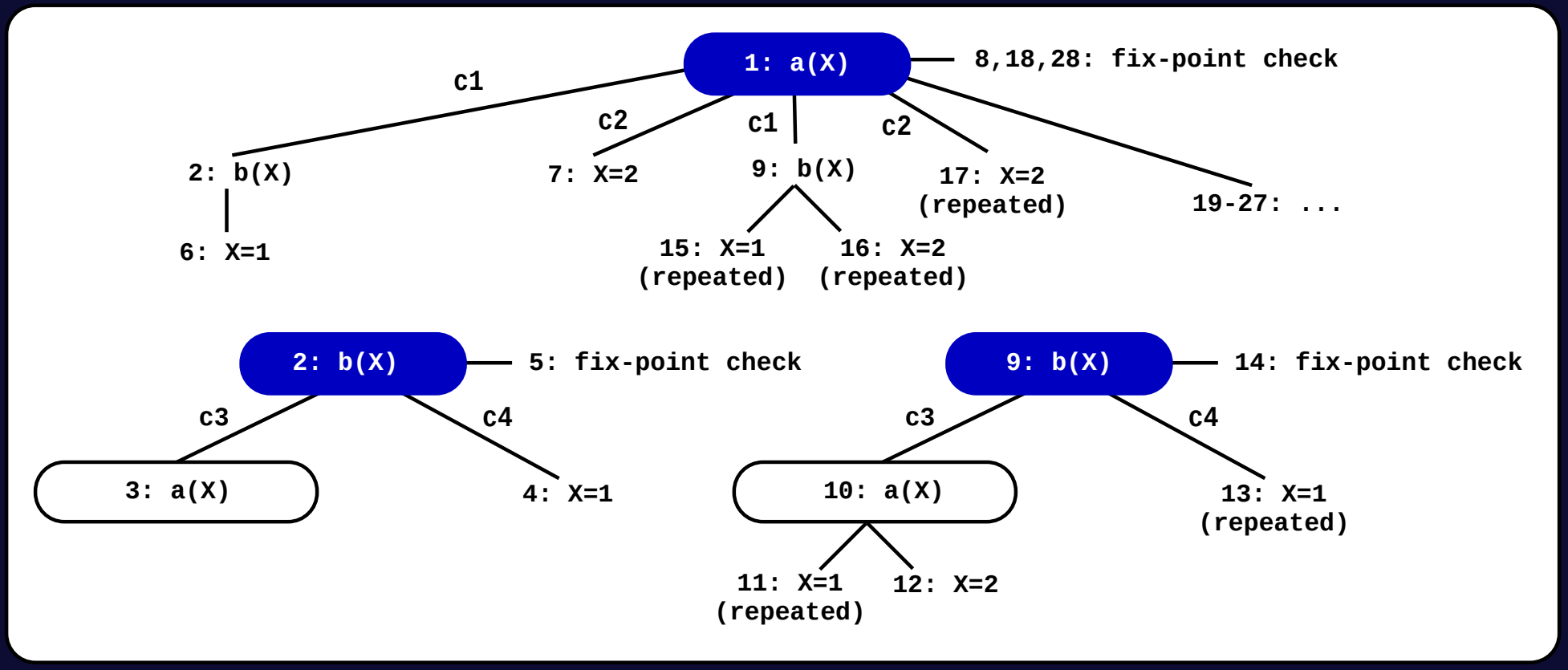
A Nossa Proposta

- Implementar uma plataforma que trabalha sobre o sistema **Yap Prolog** que permitirá suportar a **combinação das três estratégias**.
- Desta forma, poderemos analisar as **vantagens e desvantagens** de cada estratégia, quando usadas **sozinhas ou combinadas** com outras.
- Iremos também comparar a **performance** do sistema linear com o sistema baseado em suspensão.

Exemplo de Execução Linear

```
:- table a/1, b/1.

a(X):- b(X).          (c1)
a(2).                 (c2)
b(X):- a(X).          (c3)
b(1).                 (c4)
```

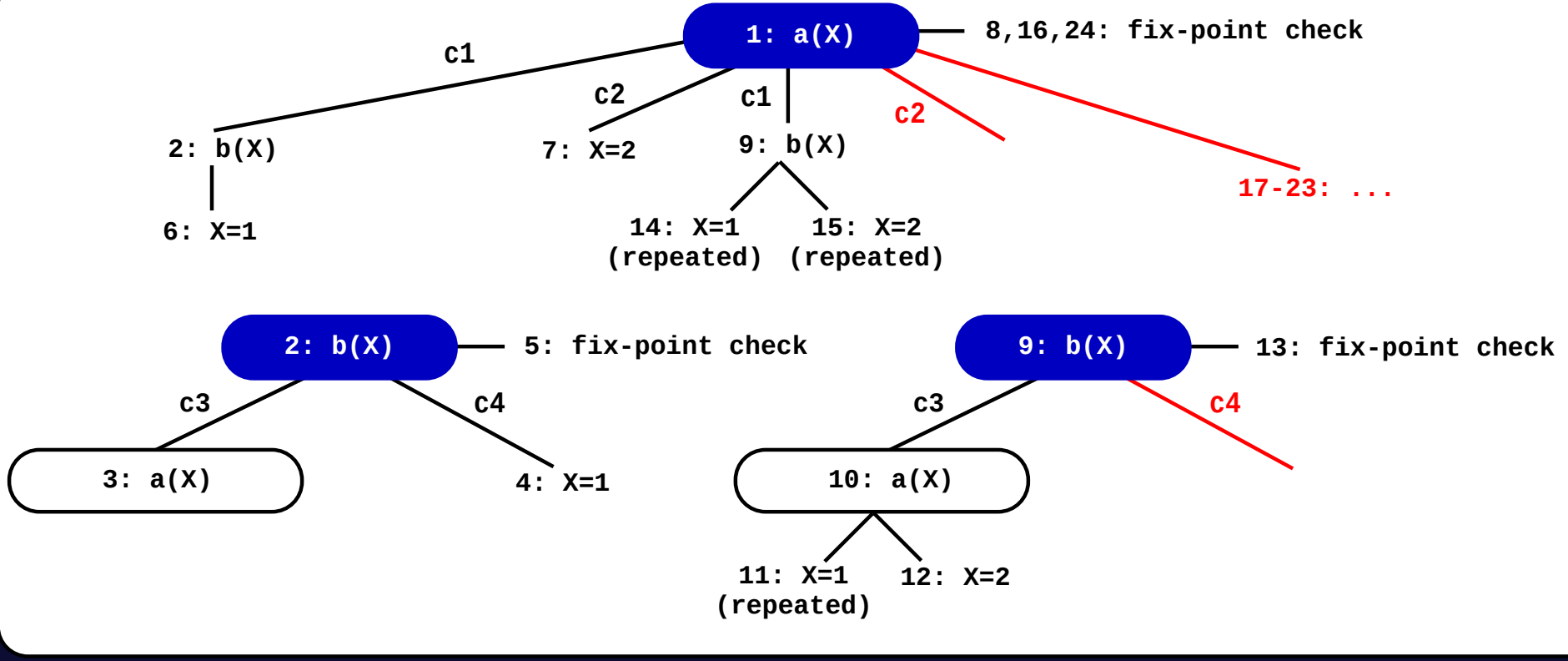
Call	Solutions
1: a(X)	6: X=1 7: X=2
2: b(X)	4: X=1 12: X=2



Exemplo de Execução com DRA

```
:- table a/1, b/1.

a(X):- b(X).          (c1)
a(2).                 (c2)
b(X):- a(X).          (c3)
b(1).                 (c4)
```

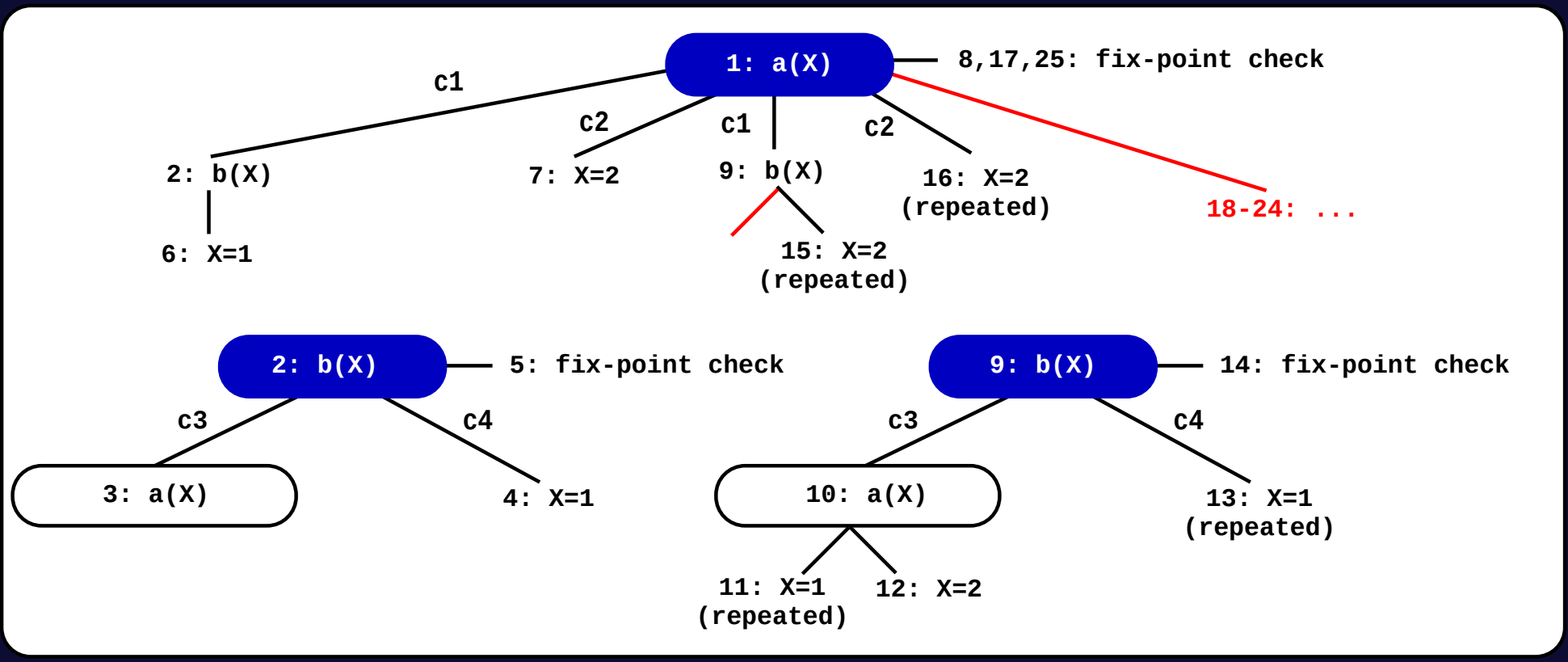
Call	Solutions	Looping Alternatives
1: a(X)	6: X=1 7: X=2	3: c1
2: b(X)	4: X=1 12: X=2	3: c3



Exemplo de Execução com DRS

```
:- table a/1, b/1.

a(X):- b(X).          (c1)
a(2).                 (c2)
b(X):- a(X).          (c3)
b(1).                 (c4)
```

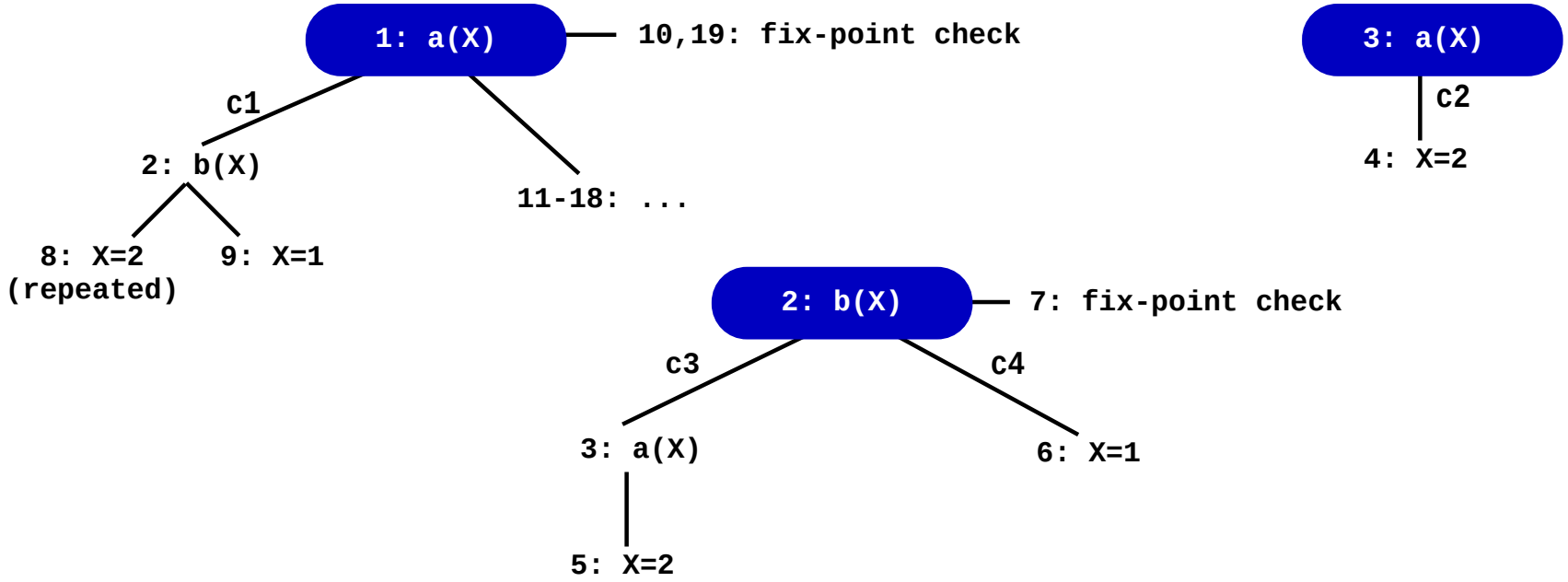
Call	Solutions	Looping Solutions
1: a(X)	6: X=1 7: X=2	
2: b(X)	4: X=1 12: X=2	



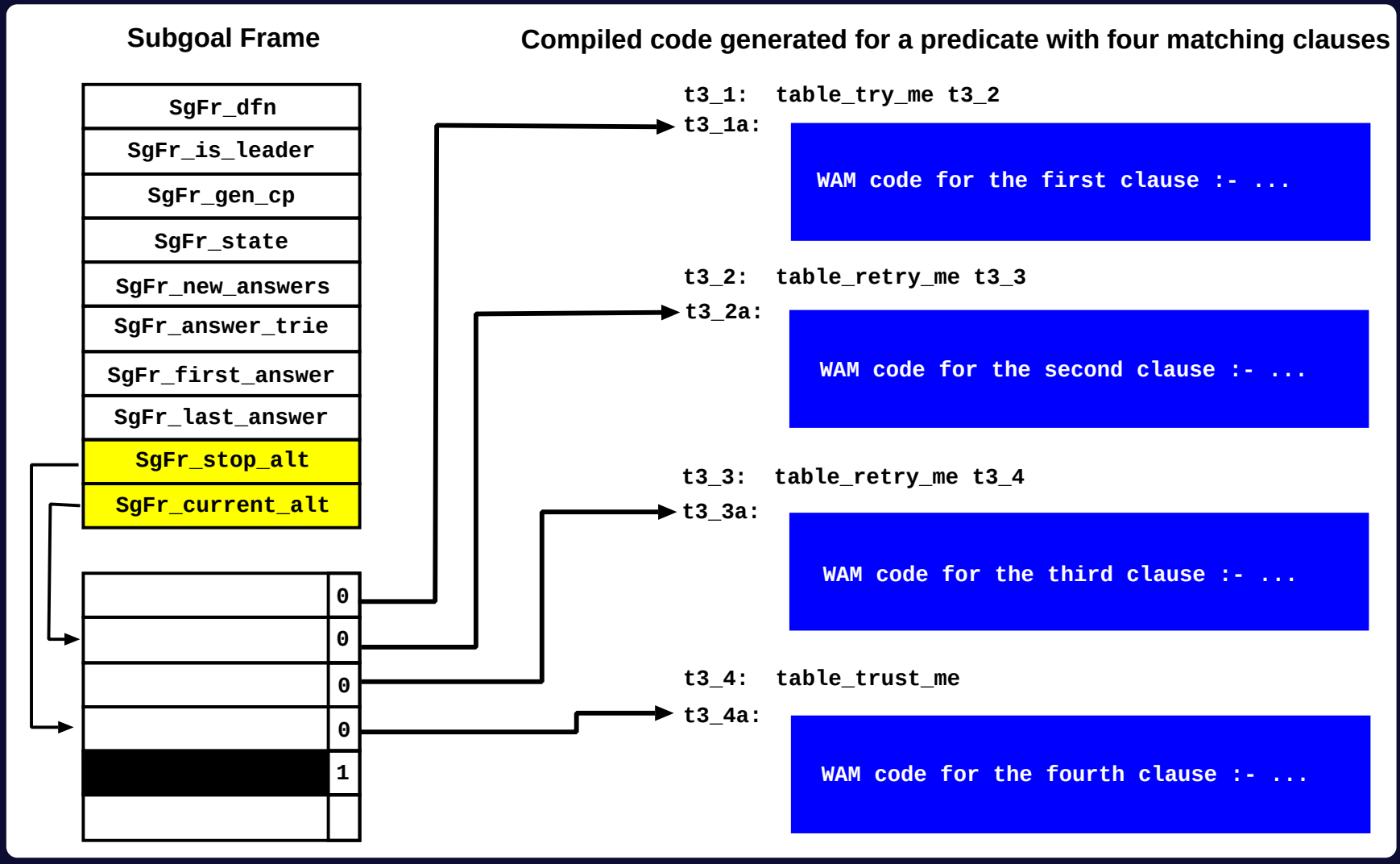
Exemplo de Execução com DRE

```
:- table a/1, b/1.

a(X):- b(X).          (c1)
a(2).                 (c2)
b(X):- a(X).          (c3)
b(1).                 (c4)
```

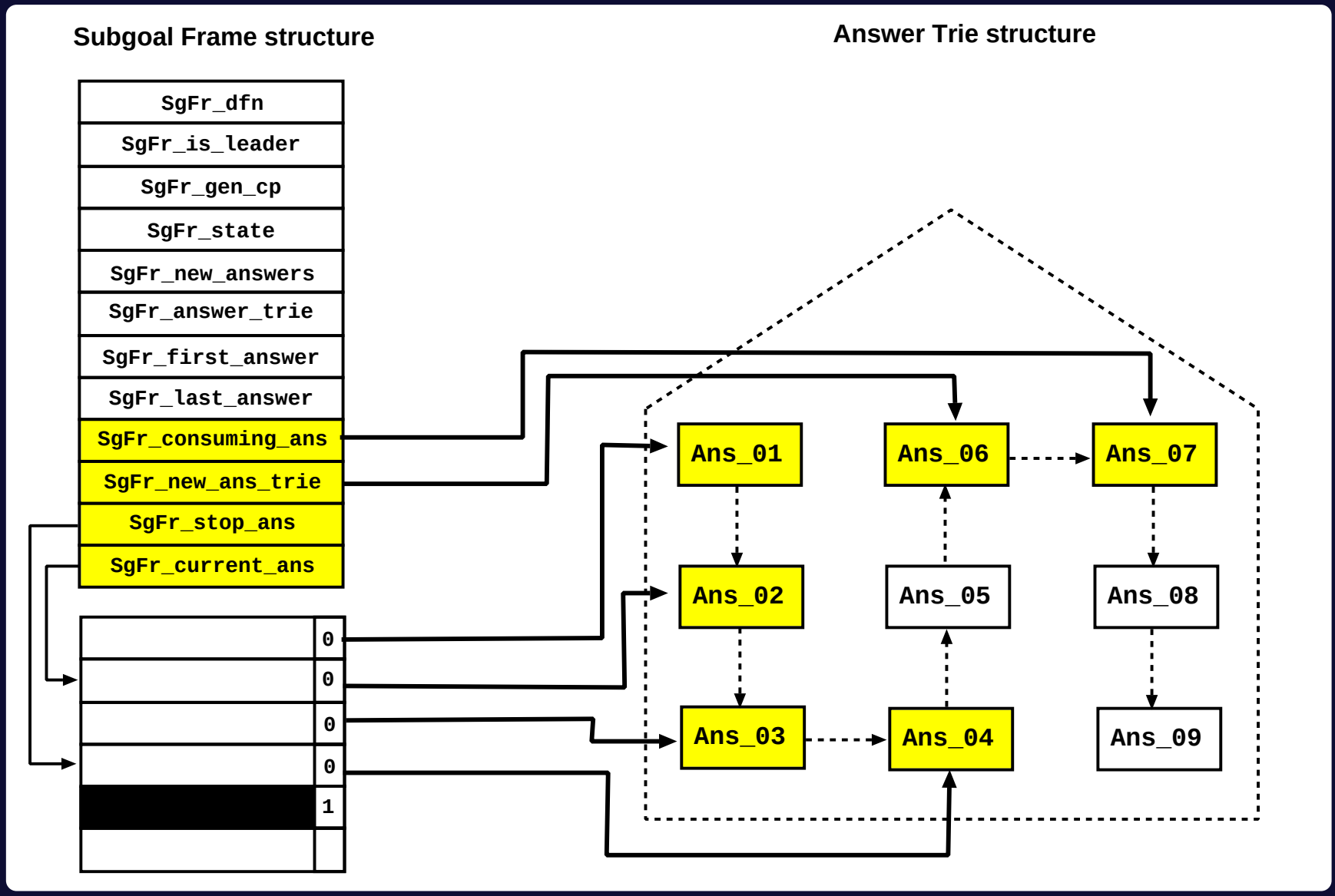
Call	Solutions
1: a(X)	4: X=2 9: X=1
2: b(X)	5: X=2 6: X=1



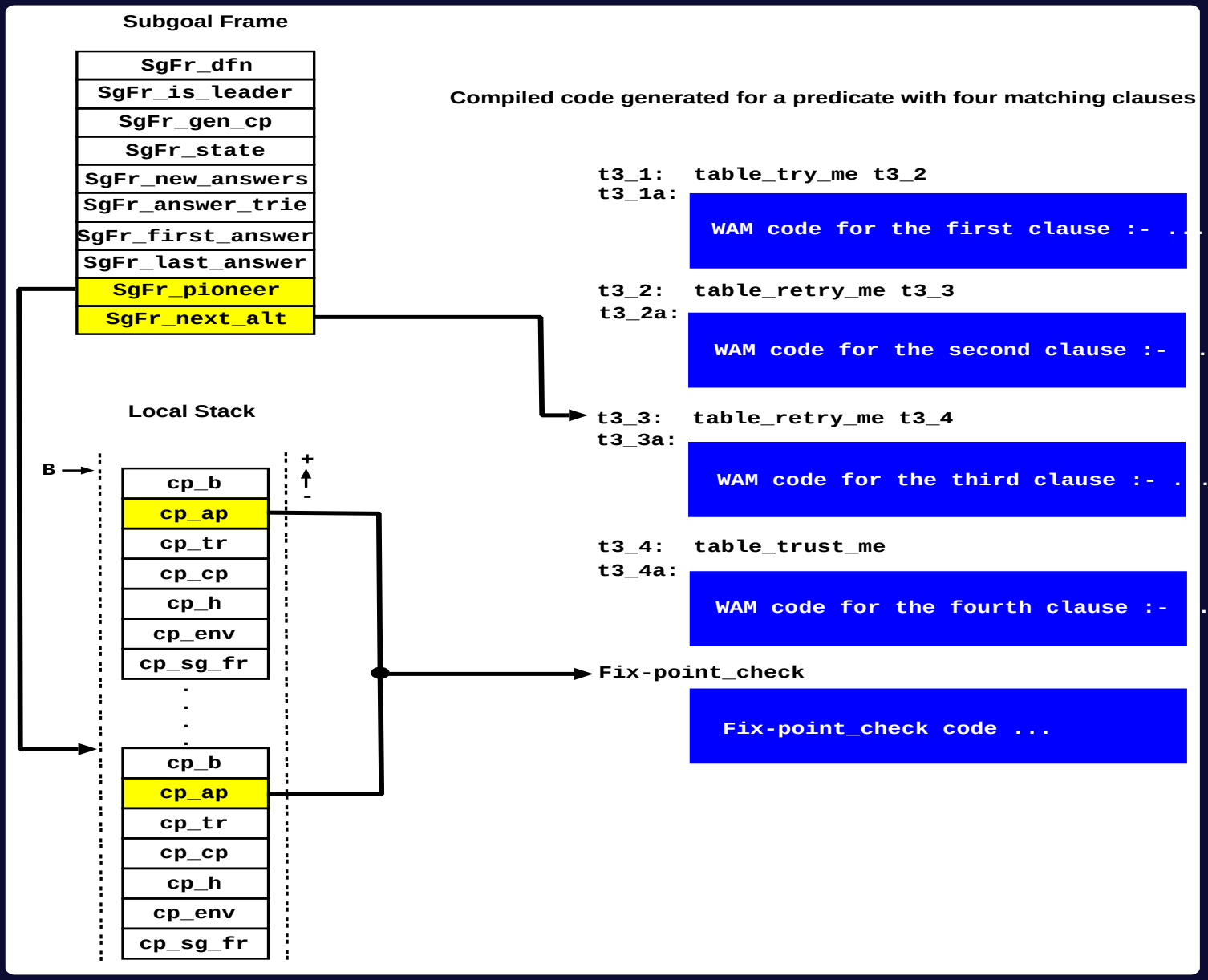
Detalhes de Implementação - DRA



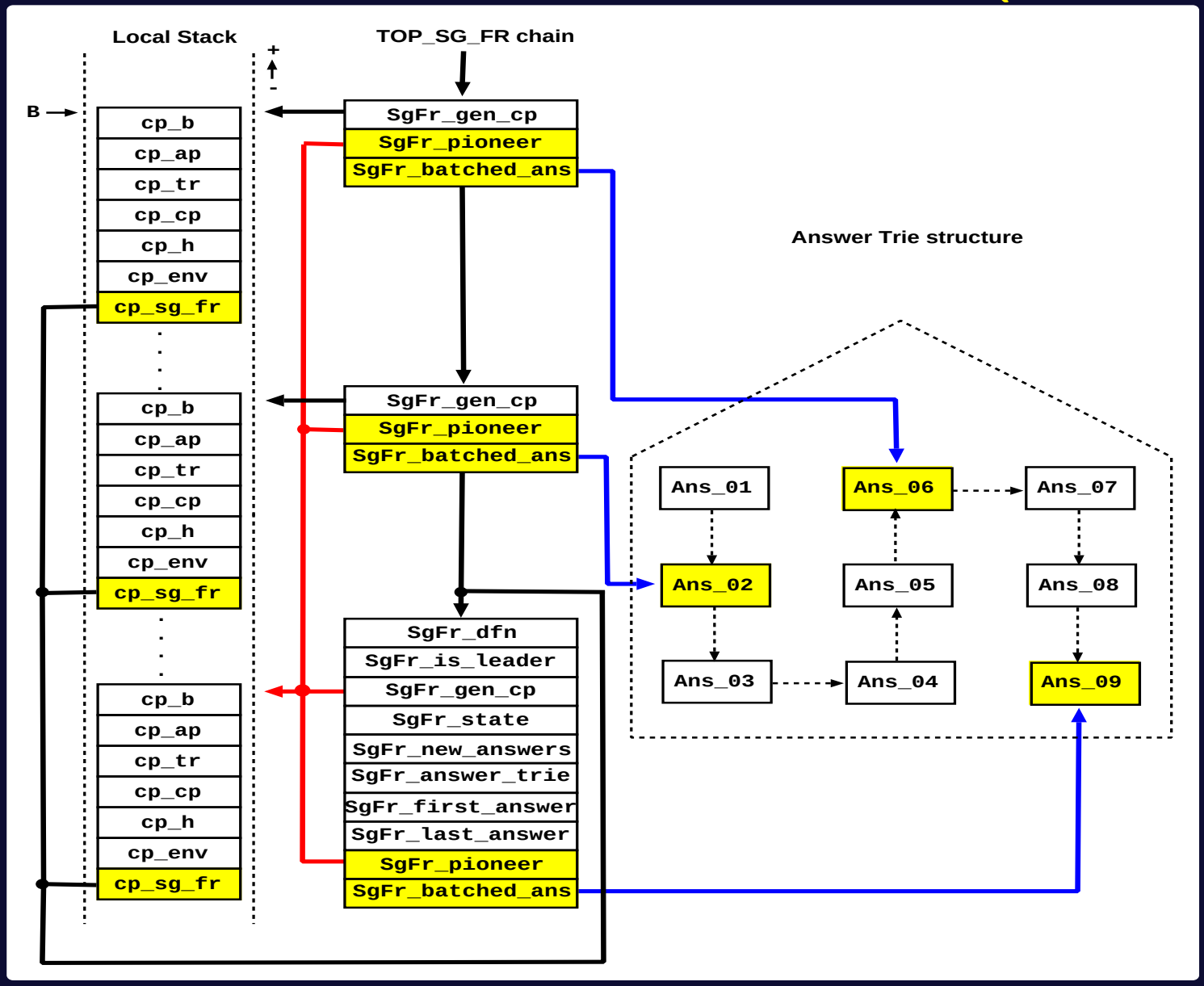
Detalhes de Implementação - DRS



Detalhes de Implementação - DRE (Local)



Detalhes de Implementação - DRE (Batched)



Resultados Experimentais - Path (Local)

Programs	DRA	DRE	DRS	DRA + DRE	DRA + DRS	DRE + DRS	All
Double First							
Cycle	**1.16**	**1.15**	**1.15**	1.16	1.14	1.15	**1.17**
Grid	**1.11**	**1.11**	**1.10**	1.10	1.09	1.09	1.09
Pyramid	**1.02**	**1.02**	1.00	1.02	1.02	1.01	1.02
Double Last							
. . .							
Right First							
Cycle	**1.18**	**1.02**	**1.28**	**1.22**	**1.64**	1.27	1.56
Grid	**1.12**	**1.06**	**1.27**	**1.15**	**1.42**	**1.30**	**1.49**
Pyramid	**1.71**	**1.08**	**1.10**	**1.72**	**1.72**	1.08	**1.74**
Right Last							
. . .							
Left First							
. . .							
Left Last							
. . .							
Average	**1.18**	**1.12**	**1.15**	**1.19**	**1.24**	**1.18**	**1.26**

Resultados Experimentais - Path (Batched)

Programs	DRA	DRE	DRA + DRE
Double First			
Cycle	**1.02**	**2.04**	**2.05**
Grid	**1.01**	**1.80**	**1.81**
Pyramid	**1.01**	**2.02**	**2.07**
Double Last			
...			
Right First			
...			
Right Last			
...			
Left First			
Cycle	**1.13**	0.78	1.02
Grid	**1.07**	0.72	1.06
Pyramid	**1.17**	0.83	1.11
...			
Left Last			
...			
Average	**1.19**	**1.03**	**1.23**

Resultados Experimentais - Warren (Local e Batched)

Programs / Depth	DRA	DRE	DRS	DRA + DRE	DRA + DRS	DRE + DRS	All
3000	**1.35**	0.48	**1.23**	0.60	1.22	0.54	0.55
6000	**1.08**	0.37	0.99	0.42	0.98	0.41	0.43
9000	**1.11**	0.39	**1.02**	0.46	1.00	0.43	0.45
12000	**1.04**	0.36	0.94	0.43	0.89	0.39	0.41
Average	**1.15**	0.40	**1.05**	0.48	1.02	0.44	0.46

Programs / Depth	DRA	DRE	DRA + DRE
3000	**1.01**	**29.00**	**174.00**
6000	**1.01**	**50.57**	**354.00**
9000	1.00	**62.31**	**741.50**
12000	**1.01**	**74.15**	**810.00**
Average	**1.01**	**54.01**	**519.88**

Resultados Experimentais - Comp. com YapTab

Programs	Local Scheduling		Batched Scheduling	
	YapTab	Best Linear (Opt)	YapTab	Best Linear (Opt)
Double First				
Cycle	1.96	1.17 (All)	2.04	**2.05 (DRA+DRE)**
Grid	1.85	1.11 (DRA)	2.07	1.81 (DRA+DRE)
Pyramid	2.04	1.02 (DRA)	2.02	**2.07 (DRA+DRE)**
Double Last				
Cycle	1.92	1.14 (All)	2.04	1.04 (DRA+DRE)
Grid	1.93	1.18 (DRE)	2.07	1.08 (DRA+DRE)
Pyramid	2.98	1.14 (All)	2.16	1.08 (DRA+DRE)
Right First				
Cycle	1.43	**1.64 (DRA+DRS)**	2.09	1.30 (DRA)
Grid	5.25	1.49 (All)	2.10	1.40 (DRA)
Pyramid	**1.78**	**1.74 (All)**	1.99	**2.05 (DRA)**
Right Last				
Cycle	**1.77**	**1.76 (DRA+DRS)**	1.82	1.34 (DRA)
Grid	5.14	1.46 (All)	2.25	1.35 (DRA)
Pyramid	1.59	**1.62 (DRA+DRS)**	1.99	1.80 (DRA)

Resultados Experimentais - Comp. com YapTab

Programs	Local Scheduling		Batched Scheduling	
	YapTab	Best Linear (Opt)	YapTab	Best Linear (Opt)
Left First				
Cycle	1.92	1.16 (All)	2.09	1.13 (DRA)
Grid	2.80	1.35 (All)	2.14	1.07 (DRA)
Pyramid	1.99	1.19 (DRE)	2.26	1.17 (DRA)
Iproto	2.23	1.15 (DRE+DRS)	2.57	1.17 (DRA)
Leader	2.28	1.13 (DRE+DRS)	2.31	1.10 (DRA)
Sieve	2.26	1.10 (DRE+DRS)	2.34	1.12 (DRA)
Left Last				
Cycle	1.99	1.22 (DRA+DRE)	1.94	1.05 (DRA)
Grid	2.48	1.35 (All)	1.89	1.01 (DRA)
Pyramid	1.96	1.16 (All)	2.23	1.09 (DRA)
Iproto	2.30	1.19 (DRE+DRS)	2.33	1.09 (DRA)
Leader	2.34	1.16 (DRE+DRS)	2.13	1.02 (DRA)
Sieve	2.26	1.09 (DRE+DRS)	2.26	1.08 (DRA)
Warren				
3000	384.00	1.35 (DRA)	**348.00**	**174.00 (DRA+DRE)**
6000	1,264.00	1.08 (DRA)	**354.00**	**354.00 (DRA+DRE)**
9000	2,992.00	1.11 (DRA)	**810.00**	**741.50 (DRA+DRE)**
12000	1,288.00	1.04 (DRA)	**1,483.00**	**810.00 (DRA+DRE)**

Conclusões

- Foi apresentada uma nova plataforma que permite a avaliação de programas lógicos com recurso a tabulação por suspensão ou linear, permitindo desta forma **comparar** pela primeira vez de forma justa ambos os mecanismos de tabulação.
- A tabulação linear integra todas as estratégias de tabulação linear mais bem sucedidas actualmente existentes (**DRA**, **DRE** e a nova estratégia **DRS**).

Conclusões

- Foi apresentada uma nova plataforma que permite a avaliação de programas lógicos com recurso a tabulação por suspensão ou linear, permitindo desta forma **comparar** pela primeira vez de forma justa ambos os mecanismos de tabulação.
- A tabulação linear integra todas as estratégias de tabulação linear mais bem sucedidas actualmente existentes (**DRA**, **DRE** e a nova estratégia **DRS**).
- Os resultados experimentais mostram que:
  - **DRA**. Reduz os tempos de execução de programas com cláusulas cíclicas. Não tem impacto negativo nos restantes programas.
  - **DRE**. Tem bons e maus resultados. Provoca uma maior expansão das pilhas do sistema, logo torna sistema de tabulação mais propício ao custo adicional de operações de **garbage collection** e de expansão de memória.
  - **DRS**. Pode ser bastante eficaz para programas que possam beneficiar da optimização, tendo um custo na execução bastante insignificante para os restantes programas.

Trabalho Futuro

- Estudar eventuais formas automáticas de determinação de qual a estratégia linear mais adequada para um programa lógico antes de este ser avaliado.
- Explorar o impacto da aplicação da nossa plataforma em programas mais complexos e próximos do mundo real, de forma a consolidar ainda mais a implementação.
- Estender o motor de tabulação linear para suporte de execução com múltiplos processos e paralelismo.