Implementação de Linguagens de Programação Lógica

Tabelação

Ricardo Rocha
ricroc@ncc.up.pt
DCC-FCUP

Tópicos Avançados de Informática
Mestrado em Informática 2003/04

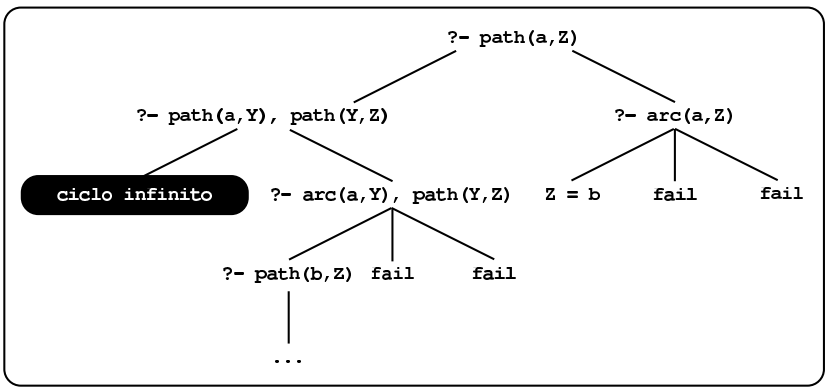
Porquê Tabelação?

- Apesar do poder, da flexibilidade e dos bons resultados que o Prolog tem demonstrado desde o aparecimento da WAM, a estratégia de **resolução SLD** na qual o Prolog se baseia é limitadora do potencial inerente ao paradigma da Programação Lógica.

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

arc(a,b).
arc(b,c).
arc(c,b).

          ?- path(a,Z).
```



Porquê Tabelação?

- A resolução SLD não consegue tratar devidamente:
  - **Ciclos positivos infinitos** (expressividade insuficiente)
  - **Ciclos negativos infinitos** (inconsistência)
  - **Computações redundantes** (ineficiência)

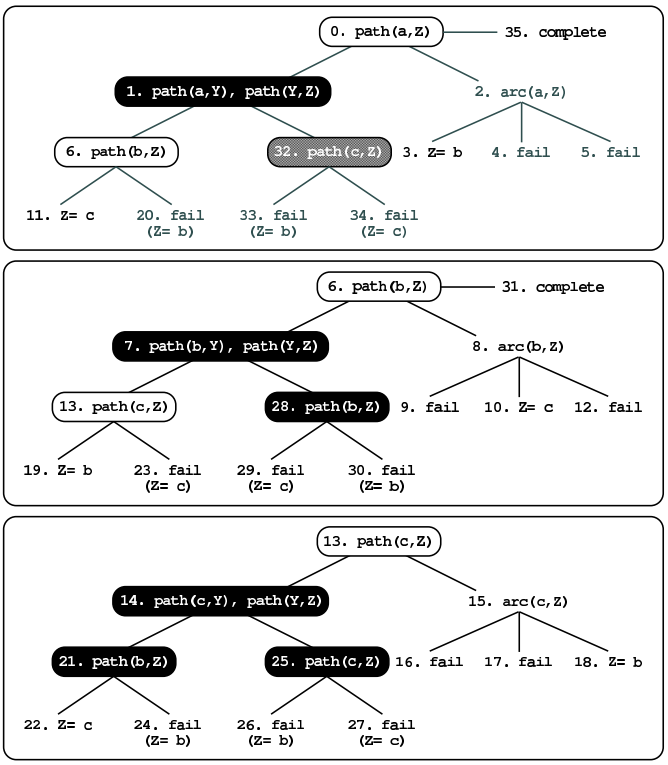
É necessário uma estratégia diferente!

- Uma das mais bem sucedidas técnicas para solucionar a incapacidade da resolução SLD no que respeita a ciclos infinitos e computações redundantes é a **Tabelação**.
- A ideia básica da tabelação consiste em ir guardando numa área auxiliar, a **tabela**, as soluções intermédias encontradas para determinados objectivos, para que estas possam ser reutilizadas assim que surgirem chamadas repetidas a esses mesmos objectivos.

Modelo Básico de Execução

- Predicados **não tabelados** são avaliados tal como em Prolog aplicando resolução SLD.
- Sempre que um objectivo de um predicado **tabelado** é chamado pela **primeira vez**, adiciona-se à tabela uma nova entrada relativa a esse objectivo, e a execução prossegue aplicando resolução SLD.
- Sempre que se obtém uma **nova solução** para um objectivo tabelado, esta é guardada na tabela.
- Chamadas repetidas de objectivos tabelados são resolvidas **consumindo** as soluções previamente encontradas e guardadas na tabela.
- Assim que todas as soluções forem consumidas, o estado da computação é **suspenso** e a execução falha. Entretanto, se durante a exploração do objectivo em causa surgirem novas soluções, então a computação suspensa é **retomada** para consumir as novas soluções. Caso contrário, é considerada **completa**.
- A computação **termina** assim que todas as soluções tiverem sido consumidas e não for possível continuar a execução por aplicação da resolução SLD.

Exemplo: Tabelação




```
:- table path/2.

path(X,Z):- path(X,Y),path(Y,Z).
path(X,Z):- arc(X,Z).

arc(a,b).
arc(b,c).
arc(c,b).

        ?- path(a,Z).
```

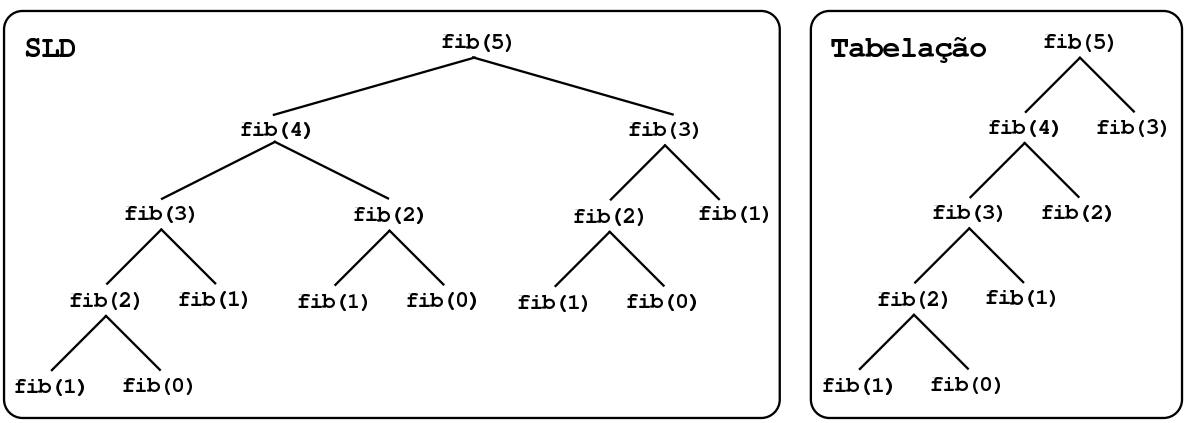
objectivo	soluções
0. path(a,Z)	3. Z= b 11. Z= c 35. complete
6. path(b,Z)	10. Z= c 19. Z= b 31. complete
13. path(c,Z)	18. Z= b 22. Z= c 31. complete

Exemplo: Reduzir a Complexidade

```
:- table fib/2.

fib(0,1).
fib(1,1).
fib(N,Z) :- P is N - 1, Q is N - 2,
            fib(P,X), fib(Q,Y), Z is X + Y.


                ?- fib(5,Z).
```



Conceitos Básicos

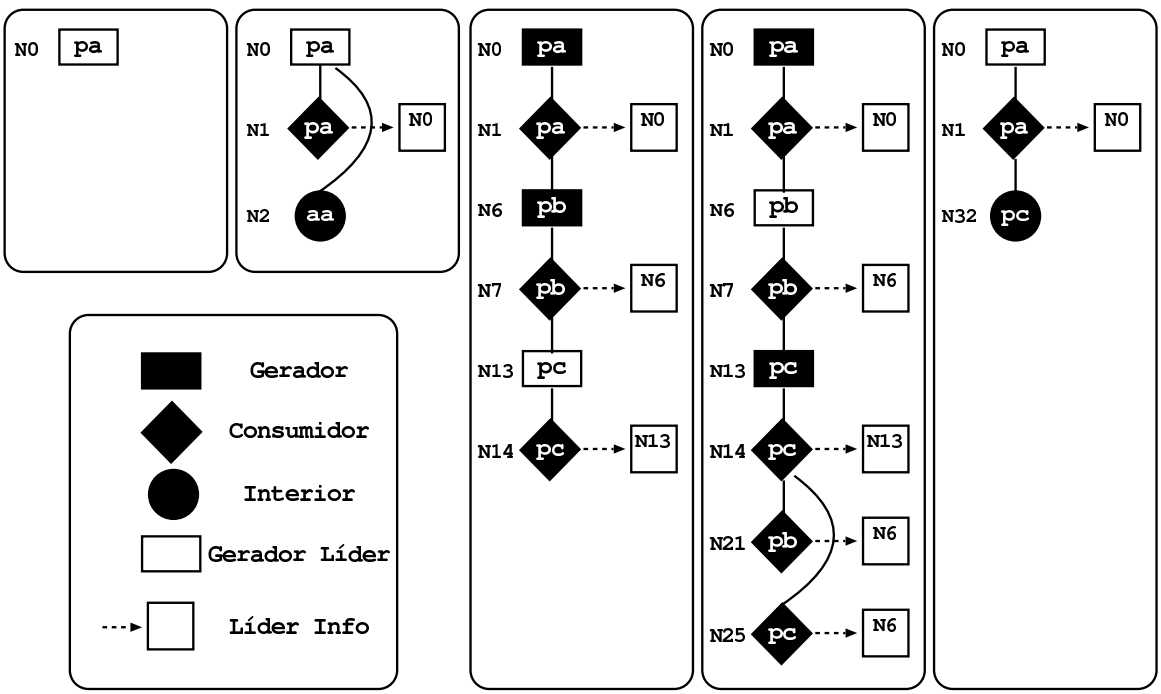
Classificação dos nós da árvore de procura

- **Interiores:** representam os objectivos não tabelados
- **Geradores:** representam as primeiras chamadas a objectivos tabelados
- **Consumidores:** representam as chamadas repetidas a objectivos tabelados

Completude e líderes

- Um objectivo tabelado está **completamente avaliado** quando:
  - Não podem ser geradas mais soluções por resolução SLD
  - Todas as chamadas ao objectivo consumiram todas as soluções encontradas
- Um conjunto de objectivos pode ser mutuamente dependente, **strongly connected component (SCC)**, e nesse caso só pode ser completo em simultâneo.
- Um SCC está completamente avaliado quando cada objectivo do SCC está completamente avaliado. A completude de um SCC deve ser realizada pelo **líder** do SCC, isto é, o objectivo mais antigo do SCC.

Calcular o Líder



Extensões de Suporte

- **Novas instruções:** instruções de suporte às 4 operações básicas
  - **Tabled Subgoal Call:** verifica se um objectivo já existe na tabela, e senão insere-o e cria um novo nó gerador. Caso contrário, cria um nó consumidor e começa a consumir as soluções disponíveis
  - **New Answer:** verifica se uma nova solução já existe na tabela, e senão insere-a
  - **Answer Return:** consume a próxima solução disponível, se alguma
  - **Completion:** determina se um objectivo tabelado está completamente avaliado, e senão calcula uma possível resolução para continuar a execução
- **Suspensão/Recuperação:** mecanismos que permitam suspender e recuperar o estado da computação
- **Tabela:** espaço para guardar os objectivos tabelados e respectivas soluções

Completion

É executada sempre que a computação atinge um **nó gerador** e este já explorou todas as suas alternativas.

O nó gerador é líder?

- **Não** → Falha (backtracking)
- **Sim** → Existe algum nó consumidor CN mais novo com soluções por consumir?
  - **Sim** → Retoma a computação em CN
  - **Não** → Completa o SCC corrente

Answer Resolution

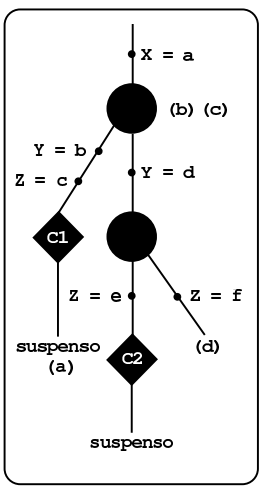
É executada sempre que a computação atinge um **nó consumidor**.

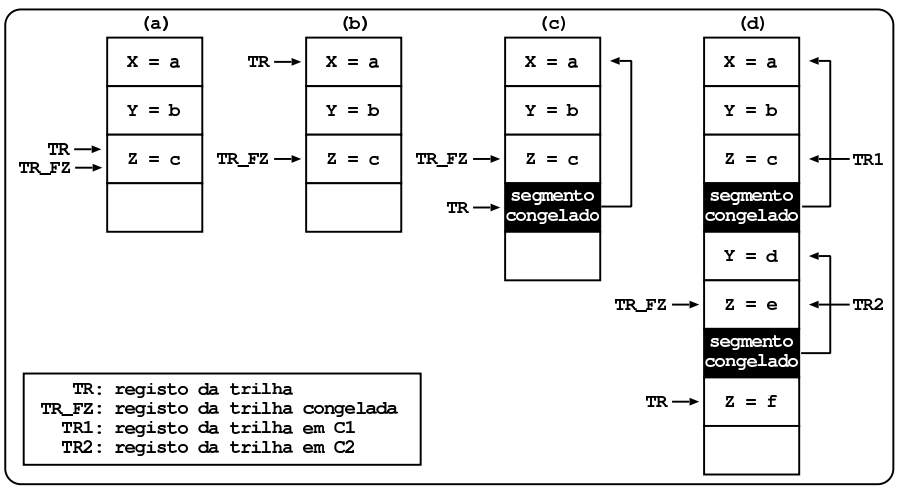
O nó consumidor tem soluções por consumir?

- **Sim** → Carrega a próxima solução e prossegue a execução
- **Não** → Retoma a computação no mais recente dos seguintes nós
  - Nó consumidor mais antigo com soluções por consumir
  - Líder onde foi executada a última operação de completion sem sucesso

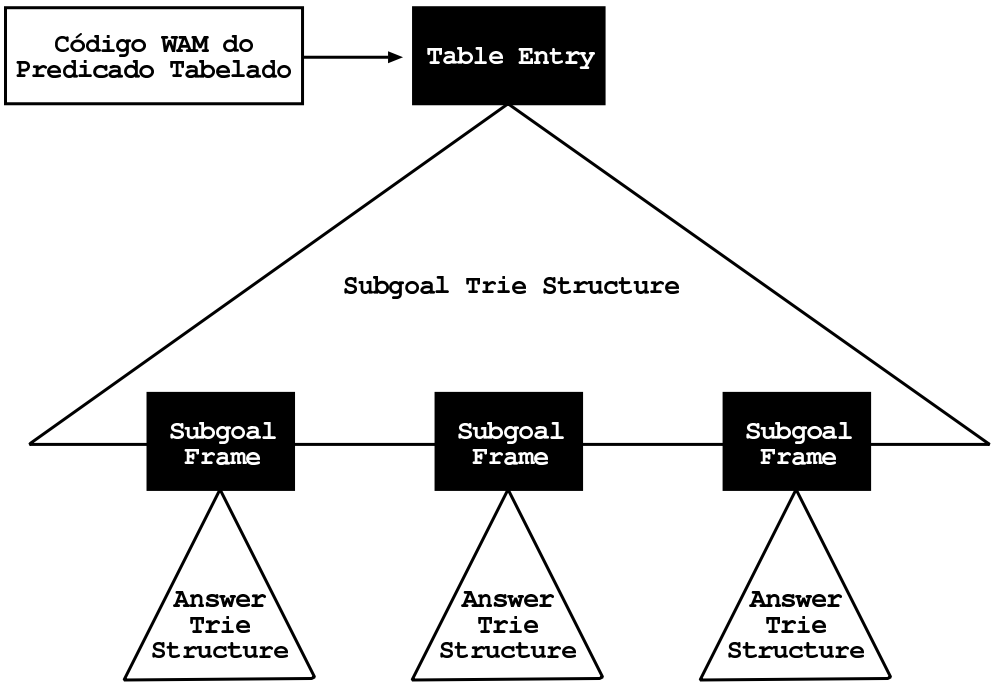
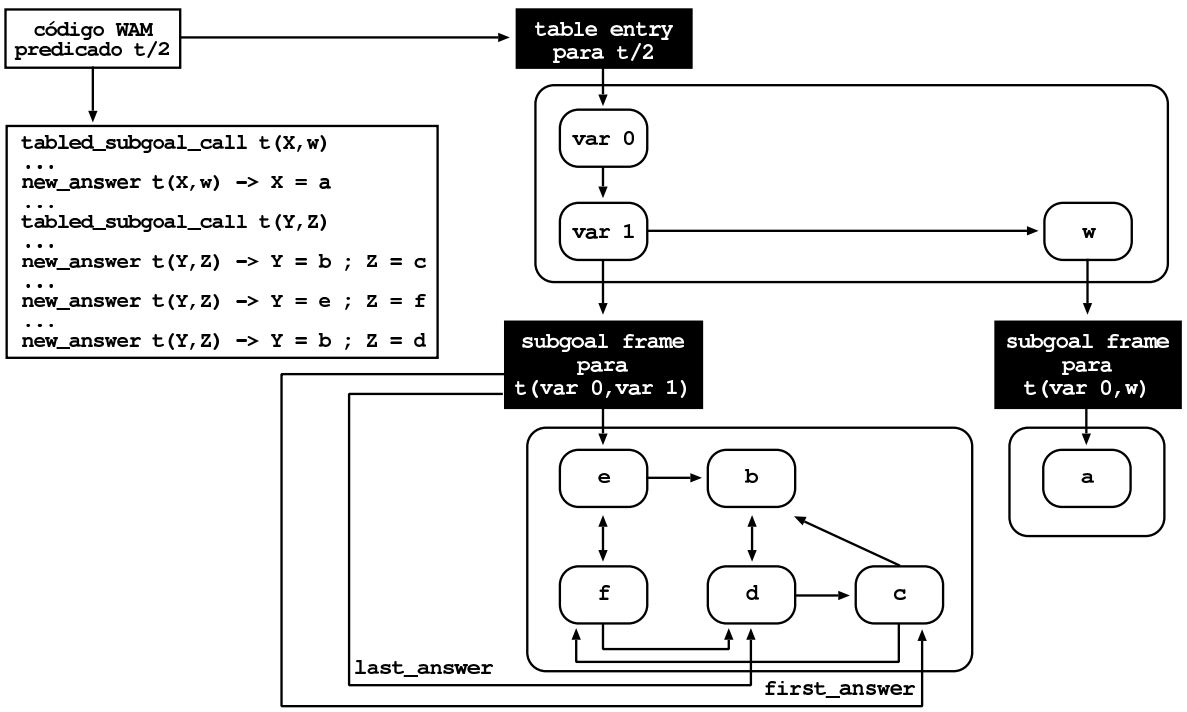
Suspensão/Recuperação

- **Suspensão**
  - Copiar as pilhas de execução para uma área auxiliar
  - Congelar as pilhas de execução (**freeze registers**)
- **Recuperação**
  - Restaurar as pilhas de execução e as atribuições condicionais (**forward trail**)





Organização da Tabela



Exemplo: Organização da Tabela