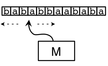
O autômato bidirecional

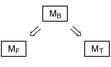
O autômato bidirecional, como nós já sabemos, é um autômato que pode mover a sua cabeça de leitura para frente e para trás



O controle da movimentação da cabeça de leitura é feito por meio de um parâmetro adicional associado às transições

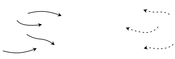
$$q_i \xrightarrow{a,F} q_j \qquad\qquad q_i \xrightarrow{a,T} q_j$$

Em princípio, nós podemos imaginar a coleção de transições para frente e para trás como dois autômatos independentes: um autômato M_F que só anda para a frente, e outro autômato M_T que só anda para trás.



Desse ponto de vista, uma computação em M_B se desenrola como uma sequência alternada de computações em M_F e M_T.

A ideia da simulação consiste em realizar uma série de computações simultâneas em M_F, e simular em paralelo uma série de computações em M_T (sempre andando com a cabeça de leitura para frente)



E de alguma maneira combinar esses segmentos de computação para determinar o resultado da computação bidirecional de M_B

Mais especificamente, a simulação vai acontecer da seguinte maneira

Quer dizer, nós vamos ter uma linha de execução principal que realiza a trajetória indicada acima (com saltos).

E nós também vamos ter outras computações para frente e simulações de computações para trás acontecendo em paralelo, que vão fornecer as informações sobre como os saltos devem ocorrer.

Vejamos.

Os saltos na linha de execução principal acontecem no momento em que a cabeça de leitura começa a andar para trás — i.e., no momento em que uma transição do tipo $q_i \xrightarrow{a,T} q_j$ está para ser realizada em M_B.

Quer dizer, os saltos acontecem no momento em que um segmento de computação para frente (em M_F) está se encerrando, o que corresponde também ao início de um segmento de computação para trás (em M_T).

A ideia é que esse segmento de computação para trás já vem sendo simulado (com a cabeça de leitura andando para frente), e que ele pode ser identificado pelo estado q_i onde ele começa.

Esse segmento de computação para trás se encerra no momento em que a cabeça de leitura começa a andar para frente novamente — i.e., no momento em que uma transição do tipo $q_i \xrightarrow{a,F} q_j$ está para ser realizada em M_B.

E, como no caso anterior, esse ponto corresponde também ao início de um segmento de computação para frente (em M_F).

De fato, é essa linha de execução para frente que vai indicar o destino do salto que acontece na linha de execução principal

Certo.

Vejamos agora os detalhes de como como a coisa vai funcionar.

A primeira observação importante é que, andando com a cabeça de leitura para frente, nós encontramos primeiro o ponto em que o segmento de computação para trás termina, e só mais adiante o ponto em que esse segmento de computação começa.

Nós já sabemos que um segmento de computação para trás termina no momento em que uma transição do tipo $q_i \xrightarrow{a,F} q_j$ está para ser realizada em M_B.

Mas, nós não temos como consultar a computação em M_B (ela não está acontecendo) para saber qual é a transição que será realizada em seguida.

Quer dizer, nós até sabemos qual é o símbolo da palavra de entrada nesse ponto, a ou b, mas nós não sabemos qual é o estado de M_B.

A ideia, então, é considerar todas as possibilidades.

Suponha que o símbolo da palavra de entrada nesse ponto é b (como na figura).

Daí, nós examinamos todos os estados de M_B que possuem uma transição

$$q_k \xrightarrow{b,F}$$

Cada um desses estados q_k é um lugar onde, nesse ponto da palavra de entrada, pode estar se encerrando um segmento de computação para trás.

Mas, se isso é o caso, então cada um desses estados q_k deve ser o lugar onde nós devemos estar iniciando a simulação de uma computação para trás.

Quer dizer, agora nós já sabemos o momento em que as simulações de computações para trás são iniciadas.

Mas, em princípio, não há como saber se essa simulação será útil mais adiante ou não — i.e., se ela vai se conectar com a linha de execução principal.

Em todo caso, nós vamos realizar todas elas em paralelo.

Nesse ponto, surge a questão sobre o número de simulações que estarão sendo realizadas em paralelo.

E aqui nós fazemos uma segunda observação importante:

- no momento em que nós estamos iniciando a simulação de uma computação para trás a partir de q_k, nenhuma outra simulação pode estar no estado q_k

A razão é fácil de entender.

Qualquer outra simulação de uma computação para trás teve início em um ponto anterior da palavra de entrada.

E esse ponto anterior onde a simulação começa é o lugar onde a computação para trás termina.

Mas, uma computação para trás que esteja em q_k no ponto atual não pode terminar mais atrás na palavra de entrada, pois q_k tem a transição

$$q_k \xrightarrow{b,F}$$

e logo não pode ter a transição

$$q_k \xrightarrow{b,T}$$

(pois M_B é determinístico).

Da mesma maneira, duas simulações de computações para trás não podem convergir para o mesmo estado, pois isso corresponde ao seguinte par de transições

o que não pode acontecer, pois M_B é determinístico.

Por outro lado, a simulação de uma computação para trás pode bifurcar em simulações que continuam em direções diferentes

o que corresponde a duas linhas de execução para trás diferentes que convergem para o mesmo estado.

Nesse ponto, a simulação se divide em duas simulações diferentes, uma para cada possível continuação.

Finalmente, pode acontecer da simulação de uma computação para trás não ter como continuar, porque não há uma transição para trás, associada ao símbolo atual da palavra de entrada, chegando no estado em que ela se encontra

Nesse ponto, a simulação é encerrada.

A figura abaixo ilustra a situação que nós descrevemos até o momento

— Figura —

Quer dizer

- nós temos uma linha de execução principal que foi interrompida em um certo ponto, e que está prestes a realizar um salto
- e nós temos uma série de simulações de computações para trás, que podem ter começado em pontos diferentes da palavra de entrada, e que se encontram em estados diferentes