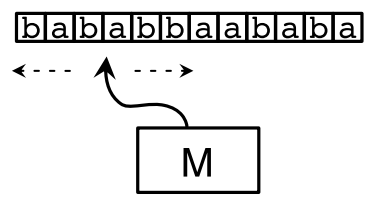
## Controlando a movimentação da cabeça de leitura

O *autômato bidirecional* é um autômato que pode mover a cabeça de leitura para frente e para trás




Do ponto de vista do controle, a primeira novidade é que agora a movimentação da cabeça de leitura não é mais autômatica.

Quer dizer, como agora existem duas opções de movimento, é o próprio autômato quem deve determinar, a cada passo, se ele quer mover a cabeça de leitura para frente ou para trás.

Na prática, isso é feito adicionando um novo parâmetro às transições para indicar a direção do movimento.

Por exemplo,

$$q_i \xrightarrow{a,\,>} q_j \qquad\qquad q_i \xrightarrow{b,\,<} q_k$$

E do ponto de vista computacional, essa pequena mudança tem uma consequência muito significativa, porque

→ *os autômatos bidirecionais podem entrar em um* loop infinito*,*
*e quando isso acontece a sua computação não acaba nunca*

Por exemplo, o par de transições

$$q_i \xrightarrow{a,\,>} q_j \qquad\qquad q_j \xrightarrow{a,\,<} q_i$$

pode levar a um loop infinito, na situação em que a palavra de entrada tem 2 a's consecutivos.

Na prática, isso significa que é o autômato quem determina o momento em que a computação acaba.

Por conveniência, nós vamos adotar as seguintes convenções para os autômatos bidirecionais

1. Além da palavra de entrada, a fita também contém os símbolos delimitadores ⊢ e ⊣

   ⊢ a b a b b a a b ⊣

2. A cabeça de leitura não se move quando nós tentamos andar para trás sobre o símbolo ⊢,
   ou quando nós tentamos andar para frente sobre o símbolo ⊣.
3. A computação sempre começa com a cabeça de leitura posicionada sobre sobre
   o símbolo inicial ⊢
4. O autômato sempre deve posicionar a cabeça de leitura sobre o símbolo final ⊣
   antes de encerrar a sua computação.
5. O autômato possui apenas um estado final $q_F$, e apenas um estado de rejeição $q_R$,
   e a computação sempre deve terminar em um desses dois estados.

A seguir, nós vamos ver alguns exemplos de autômatos bidirecionais.

**Exemplos**

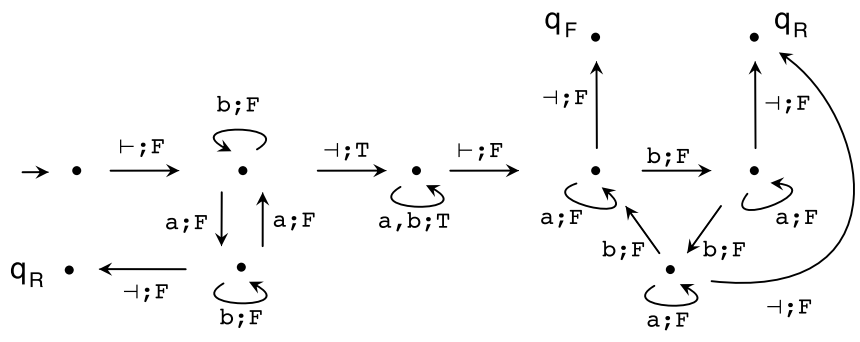
*a.* Duas tarefas

Considere a propriedade

→ palavras que possuem uma quantidade par de a's e uma quantidade de b's múltipla de 3

Reconhecer essa propriedade é uma tarefa fácil para os autômatos bidirecionais:

- primeiro, ele percorre a palavra verificando a condição associada aos a's
- depois, ele retorna a cabeça de leitura para o início da palavra
- finalmente, ele percorre a palavra mais uma vez, verificando a condição associada aos b's



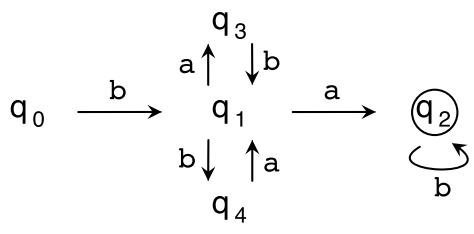

◇

*b.* Backtracking

Considere a expressão regular

$$b(ab \cup ba)^* ab^*$$

Abaixo nós temos um autômato não-determinístico que aceita as palavras descritas por essa expressão regular



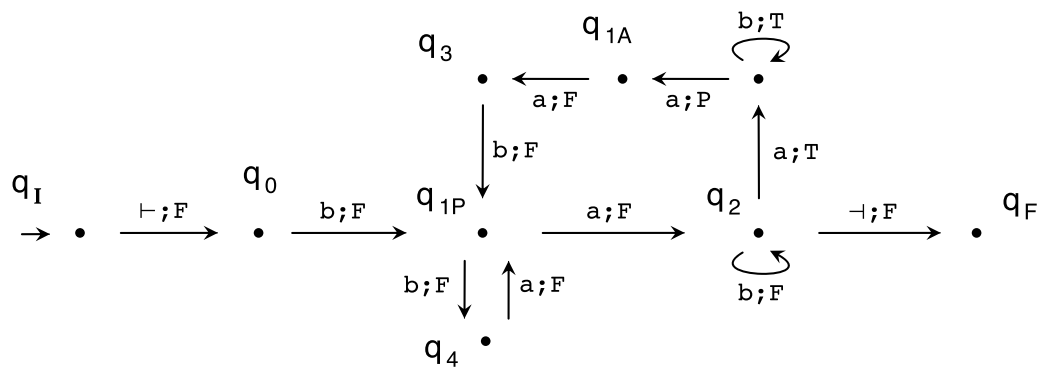

E agora, com os autômatos bidirecionais, nós podemos implementar a estratégia de backtracking para eliminar o não-determinismo.

A ideia é a seguinte

- Quando nós estamos no estado $q_1$ e aparece um símbolo a, nós fazemos a opção
  de seguir para o estado $q_2$
- Mais adiante, caso apareça um outro símbolo a, nós descobrimos que a decisão
  foi equivocada, e voltamos para o estado $q_1$, retornando a cabeça de leitura
  para o símbolo a anterior

Para implementar essa ideia, nós vamos utilizar duas cópias do estado $q_1$: uma associada ao comportamento padrão (i.e., seguir para $q_2$ quando aparece um a), e a outra associada ao comportamento alternativo (i.e., seguir para $q_3$ quando aparece um a).

Abaixo nós temos o diagrama que implementa essa ideia.




Nesse diagrama, nós utilizamos o parâmetro P (na transição chegando em $q_{1A}$) que mantém a cabeça de leitura parada no mesmo lugar, para deixar a solução mais elegante.

◇