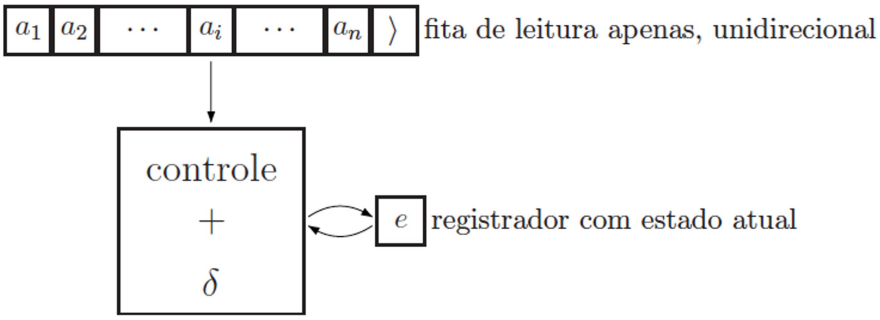
Linguagens livre de contexto

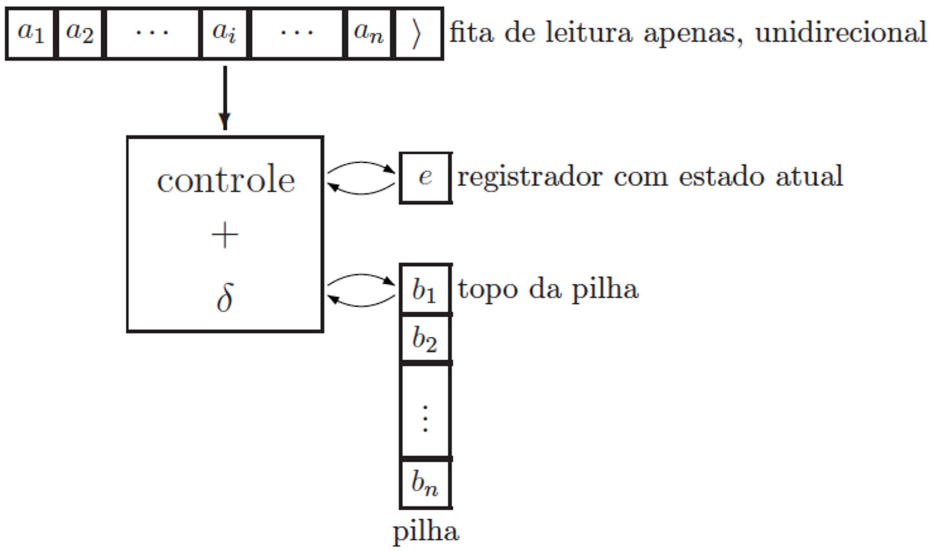
Como vimos na aula anterior, existem linguagens que não podem ser reconhecidas por um autômato finito. Do ponto de vista prático, essa é uma limitação significativa visto que não poderíamos, por exemplo, reconhecer a linguagem dos parêntesis bem balanceados (útil no reconhecimento de expressões aritméticas ou na verificação de blocos de comando bem formados em linguagens de programação).

Essencialmente, a maior dificuldade dos AFs está na sua memória limitada. Os autômatos possuem a seguinte arquitetura:

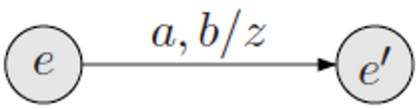


Eles possuem uma fita com o conteúdo da entrada da qual a palavra é lida da esquerda para direita; uma unidade de controle (função de transição); e um único registrador (memória) para armazenar o estado atual da máquina. Logo, linguagens como a citada acima representam um obstáculo para essas máquinas, uma vez que é necessário contar a quantidade de abre parêntesis para reconhecer depois reconhecer a mesma quantidade de fecha parêntesis. Como vimos nos diversos exemplos, os estados eram usados como forma de contar símbolos. Contudo, agora não podemos usá-los para esse fim, já que temos uma quantidade infinita de palavras com diferentes tamanhos. Precisamos, dessa forma, de um novo formalismo para reconhecer estender a capacidade dos AFs. Nessa aula veremos os autômatos de (com) pilha que abordam exatamente esse problema.

- Os **autômatos de pilha** (AP) estendem os AF acrescentando exatamente o que os faltam: memória. Como o nome sugere, uma nova unidade de memória é adicionada à máquina. Essa unidade é uma pilha tal como conhecemos em programação. Essa pilha possui um alfabeto próprio e o AP pode armazenar quantos símbolos quiser nessa estrutura (capacidade ilimitada). Contudo, como em toda pilha, só pode acessar o último símbolo armazenado na estrutura. Logo, a estrutura dos AP é a seguinte:



Vemos que a arquitetura é praticamente igual à de um AF, exceto que ele possui essa memória adicional. A fita continua contendo a palavra de entrada e só pode ser lida da esquerda para direita. O registrador de estado armazena o estado atual da máquina. A pilha contém símbolos armazenados pela unidade de controle (função de transição). Ela é uma memória de leitura e escrita, contudo, somente o símbolo mais recente pode ser lido. As transições agora podem consumir tanto o símbolo da entrada quanto o topo da pilha, além de armazenar algum dado adicional. Ou seja, as transições agora são da seguinte forma:



Note que foi dito que a transição *pode* consumir um símbolo da entrada ou topo da pilha. Isso porque podemos realizar uma transição desprezando o símbolo da entrada ($a = \lambda$) ou desprezando o topo da pilha ($b = \lambda$). Por fim, podemos ou não armazenar algo na pilha, o que é representado pela palavra z (mais à frente definiremos formalmente sua composição).

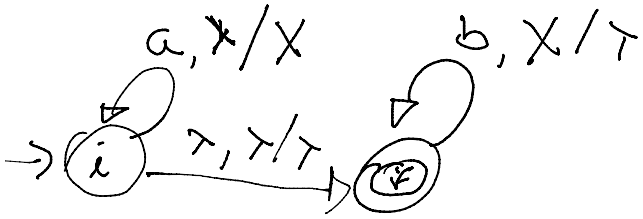
★ Formalmente, um autômato de pilha é uma sêxtupla $M = (Q, \Sigma, \Gamma, \delta, I, F)$ em que:

- Q é um conjunto finito, não vazio, de estados (como em AFs);
- Σ é o alfabeto da linguagem;
- Γ é o alfabeto da pilha (conjunto de símbolos que serão armazenados na pilha);
- $\delta: Q \times \Sigma_\lambda \times \Gamma_\lambda \to D$ é a função de transição ($\Sigma_\lambda = \Sigma \cup \{\lambda\}$; $\Gamma_\lambda = \Gamma \cup \{\lambda\}$ D é constituído pelos subconjuntos finitos de $Q \times \Gamma^*$);
- $I \subseteq$ Q é um conjunto de estados iniciais;
- $F \subseteq Q$ é um conjunto de estados finais.

Em relação aos AFs, a definição acima acrescenta um novo alfabeto, que representa os valores a serem armazenados na pilha, e a função de transição que acomoda a descrição informal dada anteriormente.

Veja que essa é a definição de um AP não determinístico. A principal característica é definida pela função de transição. Sob um mesmo símbolo de entrada e topo da pilha, podemos transitar para diferentes estados e empilhar diferentes conteúdos. A restrição que se faz é que esse conteúdo a ser empilhado seja finito. Pode-se demonstrar que os APs determinísticos são estritamente menos poderosos que os APN. Por essa razão, vamos focar somente no segundo.

Exemplo. Segue o exemplo de um APN para reconhecer a linguagem $L = \{a^n b^n | n \geq 0\}$.



De maneira similar ao que fizemos com autômatos finitos, podemos definir o conceito de **configuração instantânea** de um AP. Para seguir com as computações em um AP, precisamos saber o estado atual da máquina, o restante da palavra de entrada, e o conteúdo da pilha. Esses dados são suficientes para continuar a computação de um AP. Portanto, uma configuração instantânea de um AP é a tripla $[e, y, z]$ em que:

- e é um estado qualquer;
- $y \in \Sigma^*$ é o restante da palavra a ser consumida;
- $z \in \Gamma^*$ é o conteúdo da pilha.

Novamente, podemos definir a relação resulta em, $\vdash \subseteq (Q \times \Sigma^* \times \Gamma^*)^2$, da seguinte forma:

- $[e, ay, bz] \vdash [e', y, xz] \Leftrightarrow \delta(e, a, b) = [e', x]$ para todo $a \in \Sigma_\lambda, y \in \Sigma^*, b \in \Gamma_\lambda, z, x \in \Gamma^*$

Exemplo: O AP do exemplo anterior passaria pelas seguintes configurações instantâneas para computar a^2b^2:

$[i, aabb, \lambda] \vdash$ $[i, abb, X] \vdash$

$[i, bb, XX] \vdash$

$[f, bb, XX] \vdash$

$[f, b, X] \vdash [f, \lambda, \lambda]$

Ao contrário do que fizemos com os AFs, não é possível definir uma função de transição estendida para APs. Isso porque esses podem entrar em um laço infinito. Veja o AP abaixo e suponha que lhe seja apresentada a palavra vazia como entrada.

Existe uma computação que leva ao reconhecimento da palavra: o AP simplesmente para e aceita. Porém existe outra computação em que o AP empilha infinitamente o símbolo X.

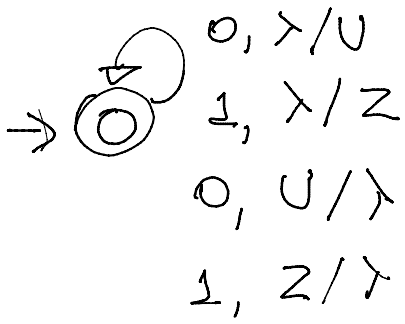
Assim, o critério de reconhecimento para um AP será definido pela relação resulta em. Especificamente, usaremos o fecho transitivo reflexivo da relação para definir a linguagem

aceita por um AP. Seja $M = (Q, \Sigma, \Gamma, \delta; I; F)$ um autômato de pilha não determinístico qualquer. A linguagem aceita por $M, L(M)$, é definida por:

$$L(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \vdash^* [f, \lambda, \lambda] \wedge i \in I \wedge f \in F\}$$

Ou seja, o AP somente reconhece uma palavra se atingir um estado final após consumir toda a palavra de entrada e a pilha estiver vazia.

- Exemplo: Construa um AP para reconhecer $L = \{w \in \{0,1\}^* \mid \eta_0(w) = \eta_1(w)\}$



- Exemplo: Construa um AP para reconhecer $L = \{w = w^R \mid w \in \{0,1\}^*\}$