



Automatos Finitos

AF com λ -movimentos
Equivalência entre GR e AF

(H&U, 1969),(H&U, 1979),
(H;M;U, 2001) e (Menezes, 2002)

AFND com λ -movimentos (movimentos vazios/silenciosos)

- Podemos estender nosso modelo de **AFND** para incluir transições com a entrada vazia (AF- λ).
- Esta facilidade **não** aumenta o poder de reconhecimento de linguagens, pois qq AF com λ -movimentos pode ser simulado por um AFND, MAS é muito utilizado nas provas de teoremas (por exemplo, para transformar uma GR em AF como veremos)
- Formalmente, só mudamos a função de transição δ que:
$$Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$$

$\delta(q,a) = p$ onde a é λ ou um símbolo de Σ .

Ex.

$L(M) = (w \in \{0,1,2\}^* \mid w \text{ contenha } \forall \text{ nro de 0's seguido por } \forall \text{ nro de 1's seguido por } \forall \text{ número de 2's})$

Input	Result
001122	Accept
012	Accept
12	Accept
21	Reject
0	Accept
1	Accept
2	Accept

Exemplo anterior com AFND

Input	Result
001122	Accept
012	Accept
12	Accept
21	Reject
0	Accept
1	Accept
2	Accept

Exemplo anterior com AFD

Input	Result
001122	Accept
012	Accept
12	Accept
21	Reject
0	Accept
1	Accept
2	Accept

Equivalência entre GR e AF

• Teo 2.18 (Menezes, 2002): Para mostrar que uma linguagem é Regular é suficiente construir um AF que a reconheça. Suponha $G = (V, T, P, S)$ uma GR, então o **AF- λ** $M = (Q, \Sigma, \delta, q_0, F)$ construído abaixo simula as derivações de G .

- $\Sigma = T$
- $Q = V \cup \{q_f\}$
- $F = \{q_f\}$
- $q_0 = S$

Produção

$A \rightarrow \lambda$

$A \rightarrow a$

$A \rightarrow aB$

Transição

$\delta(A, \lambda) = q_f$

$\delta(A, a) = q_f$

$\delta(A, a) = B$

Exemplo

$$L(G) = \{0^n 1^m \mid n, m \geq 0\}$$

$$G = (\{S, A, B\}, \{0, 1\}, P, S)$$

$$P = \{ S \rightarrow \lambda \mid 0 \mid 1 \mid 0A \mid 1B$$

$$A \rightarrow 0A \mid 0 \mid 1 \mid 1B$$

$$B \rightarrow 1B \mid 1 \}$$

O AF- λ construído de acordo com o Teo 2.18 é:

$$M = (Q, \Sigma, \delta, q_0, F)$$

- $\Sigma = T = \{0, 1\}$
- $Q = V \cup \{q_f\} = \{S, A, B, q_f\}$
- $F = \{q_f\}$
- $q_0 = S$
- δ :
 - $\delta(S, \lambda) = q_f$
 - $\delta(S, 0) = q_f$
 - $\delta(S, 1) = q_f$
 - $\delta(S, 0) = A$
 - $\delta(S, 1) = B$
 - $\delta(A, 0) = A$
 - $\delta(A, 0) = q_f$
 - $\delta(A, 1) = q_f$
 - $\delta(A, 1) = B$
 - $\delta(B, 1) = B$
 - $\delta(B, 1) = q_f$

Aplicações: Analisadores Léxicos

- **Analisadores léxicos (AL) de compiladores.** O AL é a interface entre o programa fonte e o resto do compilador. Ele é responsável principalmente por:
 - “empacotar” os caracteres do programa e lhes dar um rótulo que será usado pelo analisador sintático montar a árvore sintática.
 - Os rótulos são:
 - identificadores,
 - os nomes dos símbolos simples (<, =, [,), etc.),
 - os nomes dos símbolos compostos (:=, <>, <=, .., etc.),
 - constantes inteiras,
 - constantes reais,
 - constantes literais (cadeias e caracteres)
 - constantes lógicas (true/false),
 - o nome das palavras-chaves.

AL são modelados por AF para depois serem programados em uma linguagem de programação.

Outra opção é utilizar um gerador de analisadores léxicos como o Lex, Flex.

A entrada para esses geradores é uma **expressão regular**, assunto que veremos logo e a saída é um programa que gerencia uma enorme tabela de transição de estados.

- Já fizemos um AF para identificadores, inteiros do Pascal, e operadores relacionais
 - Como exercício façam para os outros elementos do Vt.
- Atentem que os números reais, por exemplo, **não** são iguais em todas as linguagens. Algol e Fortran permitem **5.** e **.5** quando Pascal, por exemplo, **não!!**
- Uniremos todos os AF's em um único que reconhecerá todo o Vt da linguagem escolhida. Nele também representaremos o tratamento dado aos símbolos br/CR/LF/tab e comentários
 - porém **NÃO** vamos reconhecer/aceitar esses elementos.

- Observem, entretanto que a modelagem com AF mostra o que o Analisador Léxico deve reconhecer **MAS** não mostra como.
 - Por exemplo, nada diz sobre o que fazer quando uma cadeia pode ter 2 análises como é o caso de:
2.3 (real **ou** inteiro seguido de ponto seguido de real)
Ou
<= (menor seguido de igual **ou** menor igual)
OU
Program (identificador **ou** palavra reservada program)

Regras de Desambiguação

- Assim, precisamos de regras para desambiguar esses casos.
- Usamos as regras:
 - escolha a maior cadeia
 - Dê preferência para a formação de:
 - palavras-reservadas em detrimento de identificadores