Linguagens Formais e Autômatos - Notas de Aula 08

Prof. Marco A. Alvarez - Engenharia de Computação - UCDB

marco@ec.ucdb.br http://www.ec.ucdb.br/~marco

1 Propriedades de Linguagens Regulares

Uma das principais características das Linguagens Regulares é o fato de serem representadas por formalismos de pouca complexidade, grande eficiência e fácil implementação. Entretanto, por ser uma classe relativamente simples, é restrita e limitada, sendo fácil definir Linguagens Não-Regulares. Assim, algumas questões sobre Linguagens Regulares necessitam ser analisadas:

1. Como determinar se uma linguagem é Regular?
2. Como verificar se uma Linguagem Regular é infinita, finita ou vazia?
3. É possível analisar duas Linguagens Regulares e concluir se são iguais ou diferentes?
4. A classe de Linguagens Regulares é fechada para operações de união, concatenação e intersecção?

Nas próximas aulas, a análise de cada propriedade será desenvolvida para um dos formalismos estudados. Para os demais formalismos, é suficiente traduzi-los utilizando os algoritmos apresentados nos teoremas correspondentes, como ilustrado na Figura **??**.

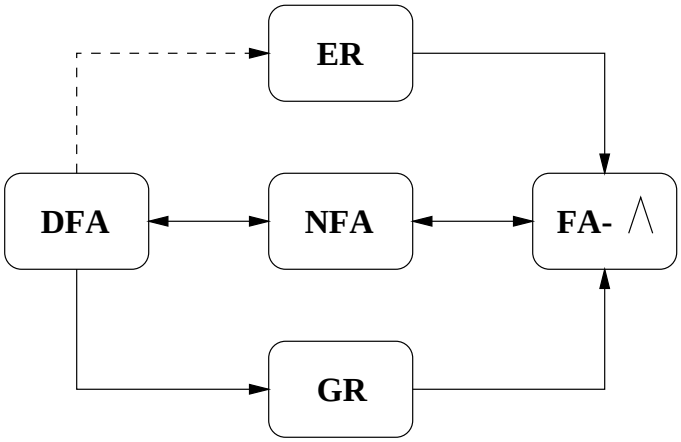

Figura 1: Tradução dos Formalismos das Linguagens Regulares

2 Bombeamento para Linguagens Regulares

O *Pumping Lemma* prova que certas linguagens infinitas não são regulares e, ao mesmo tempo, não pode ser usado para provar que uma dada linguagem é regular.

Se uma linguagem infinita é regular, então ela pode ser definida por um DFA, e este possui um número finito de estados (n por exemplo). Uma vez que a linguagem é infinita, alguns cadeias da linguagem devem ser de tamanho $> n$. Para que uma cadeia de tamanho $> n$ seja aceita pelo DFA, o "caminho" pelo DFA deve conter um ciclo. Se repetirmos o ciclo um número arbitrário de vezes, deveremos obter uma outra cadeia aceita pelo DFA.

A prova é sempre por contradição, e da seguinte forma:

- Assumir que a linguagem L é regular;
- Qualquer cadeia suficientemente longa em L deve repetir algum estado do DFA. Por isso, o caminho deve conter um ciclo;
- Mostrar que ao repetir algumas vezes o ciclo (*"pumping"* o ciclo), encontramos cadeias que não pertencem a L;
- Conclui-se que L não é regular.

É difícil utilizar este lema porque não conhecemos o DFA (se fosse conhecido, a linguagem seria regular). Por isso, é preciso fazer a prova para um DFA arbitrário que aceita L. Consequentemente, como o DFA não é conhecido, nada se conhece sobre o ciclo.

Definição 2.1 *Define-se formalmente o Pumping Lemma da seguinte forma: Se L é uma linguagem regular infinita, então existe um número inteiro positivo m tal que, qualquer cadeia $w \in L$, com $|w| \geq m$ pode ser decomposta em três partes, xyz, onde:*

- $|xy| \leq m$
- $|y| > 0$
- $w_i = xy^iz$ *está também em L para todo* $i = 0, 1, 2, 3 \ldots$

Da definição acima podemos retirar as seguintes observações:

- m é um número (finito) escolhido de tal forma que as cadeias de tamanho $\geq m$ devem possuir ciclos. Por isso, m deve ser maior ou igual ao número de estados do DFA. É bom lembrar que o DFA não é conhecido, então m não pode ser realmente estabelecido. Apenas sabe-se que deve existir um m;

- Já que a cadeia possui tamanho maior ou igual a m, é possível dividi-la em duas partes, xy e z, de tal modo que xy deva conter um ciclo. Como o DFA não é conhecido, não é possível determinar exatamente onde realizar essa separação, mas é sabido que $|xy|$ pode ser menor ou igual a m;
- Considera-se x como a parte anterior ao ciclo, y como o ciclo e z como a parte depois do ciclo. (é possível que x e z também possuam ciclos, mas isso não é considerado). Mais uma vez, não é preciso saber exatamente onde ocorre essa divisão;
- Uma vez que o ciclo que estamos interessados está em y, devemos ter $|y| > 0$, senão não existiriam ciclos;
- Pela repetição de y um número arbitrário de vezes (xy^*z), deve ser possível obter outras cadeias em L;
- Se, apesar de todas as incertezas acima pudermos mostrar que o DFA pode aceitar uma cadeia que não está em L, então pode-se concluir que a linguagem não é regular.

Para utilizar o *Pumping Lema* é necessário mostrar:

1. Para qualquer escolha de m;
2. Para algum $w \in L$, onde $|w| \geq m$;
3. Para qualquer forma de decompor w em xyz, desde que $|xy|$ não seja maior que m e $|y| > 0$;
4. Podemos escolher um i tal que $xy^iz \notin L$

Exemplo 2.1 Prove que $L = \{a^nb^n | n \geq 0\}$ não é regular

1. Não conhecemos o valor de m, mas assumimos que existe um;
2. Escolhemos uma cadeia $w = a^nb^n$ onde $n > m$, de modo que o prefixo de tamanho m consiste inteiramente de a's;
3. Não sabemos como é feita a decomposição de w em xyz, mas, uma vez que $|xy| \leq m$, xy deve consistir inteiramente de a's. Além disso, y não pode ser vazio;
4. Escolhemos $i = 0$. Isso tem o efeito de retirar $|y|$ a's da cadeia, sem afetar o número de b's. A cadeia resultante passa a ter menos a's do que b's, e por isso não pode pertencer a L. Portanto, L não é regular.

Exemplo 2.2 Prove que $L = \{a^nb^k | n > k \text{ e } n > 0\}$ não é regular

1. Não conhecemos o valor de m, mas assumimos que existe um;
2. Escolhemos uma cadeia $w = a^n b^k$ onde $n > m$, de modo que o prefixo de tamanho m consiste inteiramente de a's e $k = n - 1$, de modo que exista apenas 1 a a mais do que b's;
3. Não sabemos como é feita a decomposição de w em xyz, mas, uma vez que $|xy| \leq m$, xy deve consistir inteiramente de a's. Além disso, y não pode ser vazio;
4. Escolhemos $i = 0$. Isso tem o efeito de retirar $|y|$ a's da cadeia, sem afetar o número de b's. A cadeia resultante passa a ter menos a's do que antes, desse modo, passa a ter um número menor de a's do que b's, ou um número igual. De qualquer modo, a cadeia não pode pertencer a L. Portanto, L não é regular.

Exemplo 2.3 Prove que $L = \{a^n | n \text{ é um número primo}\}$ não é regular

1. Não conhecemos o valor de m, mas assumimos que existe um;
2. Escolhemos uma cadeia $w = a^n$ onde n é um número primo e $|xyz| = n > m + 1$ (isto sempre pode ser feito porque existem infinitos números primos). Desse modo, qualquer prefixo de w consiste inteiramente de a's;
3. Não sabemos como é feita a decomposição de w em xyz, mas, uma vez que $|xy| \leq m$, sabemos que $|z| > 1$. Além disso, y não pode ser vazio ($|y| > 0$);
4. Uma vez que $|z| > 1$, $|xz| > 1$, escolhemos $i = |xz|$. Então, $|xy^i z| = |xz| + |y| \cdot |xz| = (1 + |y|) \cdot |xz|$. Já que $(1 + |y|)$ e $|xz|$ são ambos maiores do que 1, o produto deve ser um número composto. Por isso, $|xy^i z|$ não é primo e L não é regular.

3 Operações Fechadas sobre Linguagens Regulares

Um conjunto é fechado sob uma operação se, toda vez que essa operação é aplicada a membros do conjunto, o resultado é também um membro do conjunto. Por exemplo, o conjunto de inteiros é fechado sob a operação de adição, pois a soma de dois inteiros resulta noutro inteiro. O conjunto de inteiros não é fechado sob a operação de divisão. Estamos interessados em operações que aplicadas sobre conjuntos regulares, resultem em conjuntos regulares.

3.1 União

Decorre trivialmente da definição de Expressões Regulares

3.2 Concatenação

Decorre trivialmente da definição de Expressões Regulares

3.3 Fechamento de Kleene

Decorre trivialmente da definição de Expressões Regulares

3.4 Complemento

Seja a linguagem regular L_1, reconhecida pelo DFA M_1, onde

$$L_1 = \{w | w \notin L_1, w \in \Sigma^*\}$$

O DFA que aceite a linguagem $L_2 = L_1'$, será obtido transformando todos os estados finais em não finais e vice-versa. É importante observar também que o autômato inicial deve ser modificado para garantir que somente irá parar seu processamento ao terminar de ler toda a entrada. (criar estado q_{rej} se for necessário).

Exemplo 3.1 Seja $L_1 = (0+1)^*00$, aceita por um DFA, obtenha o autômato que aceite L_1'

Exemplo 3.2 Seja $L_1 = (a+b)^*ba$, aceita por um DFA, obtenha o autômato que aceite L_1'

3.5 Inversão

Seja a linguagem regular $L1$, reconhecida pelo DFA M_1, contendo apenas um estado final. Define-se L_1^R como sendo o conjunto de cadeias de L_1 em ordem reversa. O autômato que aceite a linguagem L_1^R, será assim construído:

- Transforme o estado final antigo em novo estado inicial, e vice-versa. Quando existir mais de um estado final antigo, crie um estado inicial e ligue aos estados finais antigos através de transições vazias;
- Reverta a direção de todos os arcos do autômato.

Exemplo 3.3 Seja $L_1 = a(a+b)^*c$, aceita por um DFA, obtenha o autômato que aceite L_1^R

3.6 Interseção

Sejam duas linguagens regulares L_1 e L_2, definidas através dos DFA's M_1 e M_2, respectivamente. Define-se a interseção das duas linguagens como sendo:

$$L_1 \cap L_2 = \{w | w \in L_1, w \in L_2\}$$

O autômato que aceite a linguagem $L_3 = L_1 \cap L_2$, consistirá de estados rotulados com um par de nomes de estados: o primeiro elemento de cada par é um estado de M_1 e o segundo elemento de cada par é um estado de M_2 (Geralmente não é preciso um estado para cada par, apenas para alguns deles). O novo autômato será obtido assim:

1. Comece criando um estado inicial cujo rótulo é: (estado inicial de M_1 e estado inicial de M_2);
2. Repita os passos seguintes até que não haja arcos a serem incluídos:
   (a) Encontre um estado (A, B) que possua uma transição para algum $x \in \Sigma$;
   (b) Adicione uma transição rotulada por x partindo do estado (A, B) para o estado $(\delta(A, x), \delta(B, x))$, caso esse estado não exista, crie-o.
3. Marque o estado (A, B) como estado final, se A for estado final em M_1 e B for estado final em M_2.

Portanto, esta operação aplicada a conjuntos regulares também produz um conjunto regular, pois conseguimos obter um autômato para a nova linguagem.

Exemplo 3.4 Seja $L_1 = 11(0+1)^*$ e $L_2 = (0+1)^*00$, obtenha $L_3 = L_1 \cap L_2$

Exemplo 3.5 Seja $L_1 = a(a+b)^*c$ e $L_2 = a(a+b+c)^*$, obtenha $L_3 = L_1 \cap L_2$

4 Algoritmos de Decisão para Conjuntos Regulares

É importante ter algoritmos para decidir algumas questões:

- Dada uma linguagem regular, ela é vazia, finita, ou infinita?
- Um conjunto regular é equivalente a outro?

O teorema a seguir mostra que existe um algoritmo para verificar se uma linguagem regular representada por um DFA é vazia, finita ou infinita.

Teorema 1 (Linguagem Regular Vazia, Finita ou Infinita) *Se L é uma linguagem regular aceita por um autômato $M = \langle \Sigma, Q, \delta, q_0, F \rangle$ com n estados, então L é:*

1. **Vazia** *se, e somente se, M não aceita qualquer palavra w tal que $|w| < n$*
2. **Finita** *se, e somente se, M não aceita uma palavra w tal que $n \leq |w| < 2n$*
3. **Infinita** *se, e somente se, M aceita uma palavra w tal que $n \leq |w| < 2n$*

O seguinte teorema mostra que existe um algoritmo para verificar se dois DFA's são equivalentes, ou seja, se reconhecem a mesma linguagem.

Teorema 2 (Igualdade de Linguagens Regulares) *Se M_1 e M_2 são DFA's, então existe um algoritmo para determinar se $L(M_1) = L(M_2)$.*

PROVA:
Sejam M_1 e M_2 autômatos finitos tais que $L(M_1) = L_1$ e $L(M_2) = L_2$. É possível construir um autômato finito M_3 tal que $L(M_3) = L_3$, onde: $L_3 = (L_1 \cap L_2') \cup (L_1' \cap L_2)$. Claramente, $L_1 = L_2$ se, e somente se, L_3 é vazia. Já vimos anteriormente que existe um algoritmo para verificar se uma linguagem regular é vazia.