TEC0001 – Teoria da Computação
Videoaula 06
Linguagens Decidíveis

Karina Girardi Roggia
karina.roggia@udesc.br

Departamento de Ciência da Computação
Centro de Ciências Tecnológicas
Universidade do Estado de Santa Catarina

2020

Como provar que uma linguagem é decidível?

- É necessário haver uma Máquina de Turing decisora que a aceite
- Apresenta-se o algoritmo que decida a linguagem
- Com a análise de que ele para para qualquer entrada

Algumas Linguagens Decidíveis

- $A_{AFD} = \{\langle B, w\rangle \mid B \text{ é um AFD e } w \in L(B)\}$
- $A_{AFN} = \{\langle C, w\rangle \mid C \text{ é um AFN e } w \in L(C)\}$
- $A_{EXR} = \{\langle R, w\rangle \mid B \text{ é uma expr. regular e } w \in L(R)\}$
- $V_{AFD} = \{\langle B\rangle \mid B \text{ é um AFD e } w \in L(B) = \varnothing\}$
- $EQ_{AFD} = \{\langle A, B\rangle \mid A \text{ e } B \text{ são AFDs } L(A) = L(B)\}$
- $A_{GLC} = \{\langle G, w\rangle \mid G \text{ é uma GLC e } w \in L(G)\}$
- $V_{GLC} = \{\langle G\rangle \mid G \text{ é uma GLC e } L(G) = \varnothing\}$
- Qualquer linguagem livre de contexto

- Dada uma entrada qualquer para um AFD, seu processamento **sempre termina**
- Uma Máquina de Turing tem capacidade para receber o código de um AFD e o simular sobre qualquer entrada
- Portanto, o algoritmo decisor verifica o código $\langle B \rangle$, caso seja de fato um AFD, simula o autômato com entrada w e responde o que B responder

- Qualquer autômato finito não determinístico pode ser convertido em um AFD equivalente
- Um algoritmo para isso, mesmo tendo problema de explosão de estados, é o de ter um AFD com todas as combinações de estados do AFN. Ainda assim, o número de estados resultantes, apesar de grande, é finito.
- Portanto, a Máquina de Turing decisora verifica se $\langle C \rangle$ é de fato um código de AFN, caso seja, aplica o algoritmo de conversão para AFD equivalente. A partir disto, envia o código do AFD equivalente, juntamente com w, para o decisor de A_{AFD} e responde o que ele responder.

- Assim como existe algoritmo de conversão de AFN para AFD, existe algoritmo que, dada uma expressão regular, obtém-se AFN que reconhece a linguagem gerada pela expressão dada.
- Tal algoritmo passa por autômatos finitos com movimentos vazios, que depois são convertidos em AFN. A tradução de expressão regular para $AF\varepsilon$ é dada indutivamente pela estrutura da expressão regular.
- O algoritmo de tradução de $AF\varepsilon$ para AFN é baseado no fecho dos movimentos vazios de cada estado, o que é finito, portanto sempre para.
- A decisora para esta linguagem, portanto, verifica se o código $\langle R \rangle$ é uma expressão regular, em caso positivo, faz a conversão para AFN e passa o resultado juntamente com w para o decisor A_{AFN} respondendo o mesmo que receber dele.

- Para um AFD não aceitar nenhuma palavra, não pode haver caminho do estado inicial para nenhum estado final do autômato.
- O decisor verifica se $\langle B \rangle$ é um código de AFD, caso positivo, coloca o estado inicial em um conjunto de estados marcados S.
- Para cada $s \in S$ e para cada $\sigma \in \Sigma$, o decisor consulta $\delta(s, \gamma)$ e inclui o resultado da função em S.
- A cada iteração S pode aumentar a cardinalidade ou permanecer inalterado. Quando a segunda opção ocorrer, o decisor procura em S se existe algum estado pertencente ao conjunto de estados finais.
- Se não existir nenhum $q_f \in F \cap S$, o decisor aceita. Caso contrário, rejeita.
- Note que o algoritmo sempre para, uma vez que o número de estados em S é, no máximo, o número de estados de B, que é finito.

- Vamos usar algumas propriedades de álgebra de conjuntos para chegar ao decisor de EQ_{AFD}.
  - Dados conjuntos X e Y, temos que a *diferença simétrica* $X \ominus Y = (X - Y) \cup (Y - X)$ de conjuntos é vazia se, e somente se, $X = Y$.
  - Relembrando que $X - Y = X \cap \overline{Y}$ e $X \cap Y = \overline{\overline{X} \cup \overline{Y}}$
- Linguagens regulares são fechadas para as operações de complemento e união.
- Decisor constrói AFD que aceita $L(A) \ominus L(B)$ e manda para o decisor de V_{AFD}. Caso receba resposta positiva, aceita. Caso contrário, rejeita.

- Toda GLC pode ser convertida para uma gramática na forma normal de Chomsky equivalente
- Lembrando: gramática na forma normal de Chomsky possui somente dois tipos de derivação. $A \rightarrow t$ ou $A \rightarrow BC$ onde $t \in \Sigma$ e A, B e C são variáveis da gramática.
- Portanto, uma palavra de tamanho n somente pode ser derivada em $2n - 1$ passos.
- O decisor converte a gramática recebida para uma GLC na FNC, em seguida, testa todas as possíveis derivações com $2n - 1$ passos. Se encontrar w, aceita. Caso contrário, rejeita.

- Abordagem semelhante à do decisor de V_{AFD}, porém trabalha dos símbolos terminais para a variável inicial
- Utiliza conjunto de símbolos marcados, inicialmente somente com os símbolos terminais.
- A cada ciclo, procura símbolos que geram palavras compostas somente por símbolos já marcados e os marca também.
- Quando o conjunto de símbolos marcados não aumenta mais de uma iteração para a seguinte, verifica se o símbolo inicial da gramática está no conjunto de símbolos marcados.
- Se estiver, rejeita. Se não estiver, aceita.

- Toda linguagem livre de contexto possui uma gramática que a gere.
- A partir da gramática G e do decisor de A_{GLC}, tem-se um decisor para qualquer linguagem livre de contexto.

Considerações

- Demonstração de decidibilidade: apresentar um algoritmo que sempre pare para a linguagem desejada.
- Não há uma receita pronta para achar um algoritmo! Depende da linguagem desejada e das propriedades possíveis dos objetos da linguagem.