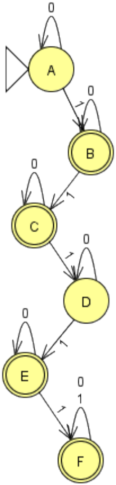


Teoria de Linguagens e Compiladores

Decidibilidade

Luiz Eduardo da Silva

Universidade Federal de Alfenas

1 de Março de 2021

Agenda

Agenda

Decibilidade

- Existem problemas que podem ser resolvidos algoritmicamente e outros não.
- Por que estudar insolubilidade de problemas?
  - Um algoritmo insolúvel é útil.
  - A computação tem limites e capacidade que precisam ser reconhecidas.
  - O vislumbre do insolúvel pode estimular a imaginação é ajudar a computação.

Agenda

Linguagens Regulares

Seja a linguagem A_{AFD} que contém as codificações de todos os AFDs juntamente com as cadeias que os AFDs aceitam:

$$A_{AFD} = \{< B, w > \mid \text{B é um AFD que aceita w } \}$$

Teorema

A_{AFD} é uma linguagem decidível

Ideia de Prova:
$M_1 =$ "Sobre a entrada $< B, w >$, onde B é a codificação do AFD e w, uma cadeia:

1. Simule B sobre a entrada w.
2. Se a simulação termina num estado de aceitação, aceite. Senão, rejeite."

Linguagens Regulares

Seja a linguagem A_{AFN} que contém as codificações de todos os *AFN*s juntamente com as cadeias que os *AFN*s aceitam:

$$A_{AFN} = \{< B, w > \mid \text{B é um AFN que aceita w } \}$$

Teorema

A_{AFN} é uma linguagem decidível

Ideia de Prova:
$M_2 =$ "Sobre a entrada $< B, w >$, onde B é a codificação do *AFD* e w, uma cadeia:

1. Converta o *AFN* B para um *AFD* equivalente C, usando a função-lambda.
2. Rode a MT M_1 anterior sobre a entrada $< C, w >$
3. Se a simulação termina num estado de aceitação, aceite. Senão, rejeite."

Linguagens Regulares

Seja a linguagem A_{EXR} que contém as codificações de todos as expressões regulares juntamente com as cadeias aceitas:

$$A_{EXR} = \{< R, w > \mid \text{R é um expressão regular que aceita w } \}$$

Teorema

A_{EXR} é uma linguagem decidível

Ideia de Prova:
$M_3 =$ "Sobre a entrada $< R, w >$, onde R é a codificação da expressão regular e w, uma cadeia:

1. Converta a expressão regular R para um AFN equivalente C, usando os modelos de união, concatenação e fecho de kleene.
2. Rode a MT M_2 anterior sobre a entrada $< C, w >$
3. Se a simulação termina num estado de aceitação, aceite. Senão, rejeite."

Linguagens Regulares

Para o *"teste de vacuidade"*, seja V_{AFD}

$$V_{AFD} = \{< A > \mid \text{A é um AFD e } L(A) = \emptyset\}$$

Teorema

V_{AFD} é uma linguagem decidível

Ideia de Prova:

M_4 = "Sobre a entrada $< A >$, onde A é um AFD:

1. Marque o estado inicial de A
2. Repita até que nenhum novo estado seja marcado:
   1. Marque qualquer estado que tenha uma transição chegando de um estado já marcado.
3. Se nenhum estado de aceitação estiver marcado, aceite. Senão, rejeite."

Linguagens Regulares

$$EQ_{AFD} = \{< A, B > \mid \text{A e B são AFD's e } L(A) = L(B)\}$$

Teorema

EQ_{AFD} é uma linguagem decidível

Ideia de Prova:

M_5 = "Sobre a entrada $< A, B >$, onde A e B são AFD:

1. Construa C para $L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$
2. Rode a MT M_4 sobre a entrada $< C >$
3. Se M_4 aceitar, aceite. Senão, rejeite."

Linguagens Livres de Contexto

$$A_{GLC} = \{< G, w > \mid \text{G é uma GLC que gera w }\}$$

Teorema

A_{GLC} é uma linguagem decidível

Ideia de Prova:

$M_6 =$ "Sobre a entrada $< G, w >$, onde G é uma GLC e w, uma cadeia:

1. Converta G para uma gramática na forma normal de Chomsky.
2. Liste todas as derivações com $2n - 1$ passos, onde n é o comprimento de w, exceto $n = 0$; neste último caso liste todas as derivações com um passo.
3. Se alguma das derivações gera w, aceite. Senão, rejeite."

Linguagens Livres de Contexto

$$V_{GLC} = \{< G > \mid \text{G é uma GLC e } L(G) = \emptyset\}$$

Teorema

V_{GLC} é uma linguagem decidível

Ideia de Prova:

M_7 = "Sobre a entrada $< G >$, onde G é uma GLC:

1. Marque todos os símbolos terminais em G
2. Repita até que nenhuma variável venha a ser marcada:
   - Marque qualquer A onde G tem uma regra $A \rightarrow U_1 U_2 ... U_k$ e cada símbolo $U_1, U_2, ... U_k$ já tenha sido marcada
3. Se a variável inicial não está marcada, aceite. Senão, rejeite."

Linguagens Livres de Contexto

$$EQ_{GLC} = \{< G, H > \mid \text{G e H são GLC's e } L(G) = L(H)\}$$

Teorema

EQ_{GLC} **NÃO** é uma linguagem decidível

Linguagens Livres de Contexto

Teorema

Toda linguagem livre de contexto é decidível

Seja A uma LLC e seja G uma GLC para A. Construímos M_8 que decide A da seguinte forma:

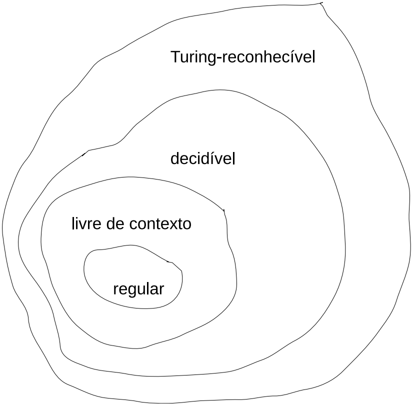
$M_8 =$ "Sobre a entrada w:

1. Rode a M_6 sobre a entrada $< G, w >$.
2. Se M_6 aceita, *aceite*. Senão, *rejeite*."

Linguagens Livres de Contexto

Teorema

Toda linguagem livre de contexto é decidível



Agenda

O Problema da Parada

- Um dos teoremas mais importantes da teoria de computação: **existe um problema que é computacionalmente insolúvel.**
- Em outras palavras, os computadores são limitados.
- Exemplo de um problema insolúvel:
    - Verificar (automaticamente) se um programa funciona conforme o especificado.

Máquina de Turing

$$A_{MT} = \{< M, w > \mid \text{M é uma MT e M aceita w }\}$$

Teorema

A_{MT} é indecidível

MT é Turing-reconhecível
$M_9 =$ "Sobre a entrada $< M, w >$, onde M é uma MT e w uma cadeia:

1. Simule M sobre a entrada w
2. Se M em algum momento entra no estado de aceitação, aceite. Se M em algum momento entra no estado de rejeição, rejeite."

O problema é que se M entra em loop, M_9 também entra em loop. Logo A_{MT} é, às vezes, denominada **problema da parada**. M_9 ou U é também chamada de MT **Universal**.

O problema da parada é indecidível

$$A_{MT} = \{< M, w > \mid \text{M é uma MT e M aceita w }\}$$

Supomos que A_{MT} é decidível e chegamos numa contradição. Seja H um decisor para A_{MT}:

$$H(< M, w >) = \begin{cases} \text{aceite} & \text{se M aceita w} \\ \text{rejeite} & \text{se M não aceita w} \end{cases}$$

Contruímos outra MT D que usa H como subrotina e retorna o contrário de H.

$D =$ "Sobre a entrada $< M >$, onde M é uma MT:

1. Rode H sobre a entrada $< M, < M >>$
2. Dê como saída o oposto do que H dá como saída; ou seja, se H aceita, *rejeite* e se H rejeita, *aceite*."

O problema da parada é indecidível

Em resumo:

$$D(< M >) = \begin{cases} \textit{aceite} & \text{se M não aceita } < M > \\ \textit{rejeite} & \text{se M aceita } < M > \end{cases}$$

O que acontece se rodamos D com sua própria descrição $< D >$?

$$D(< D >) = \begin{cases} \textit{aceite} & \text{se D não aceita } < D > \\ \textit{rejeite} & \text{se D aceita } < D > \end{cases}$$

O que D faz é forçada a fazer o contrário, o que é obviamente uma **contradição**. Obviamente, nem D, nem H devem existir.