Teoria de Linguagens – *overview*

(Prof. Marco Rodrigo Costa)

Conceitos básicos

- Alfabeto (Σ): conjunto finito de símbolos
- Símbolo (ou caractere): entidade abstrata, não definida formalmente, como números, letras e outros
- Palavra (ou cadeia, ou sentença) (w): sequência finita de símbolos do alfabeto justapostos
  - Palavra vazia (ε): palavra sem símbolo
  - Σ^*: conjunto de todas as palavras possíveis sobre Σ
  - Σ^+: conjunto de todas as palavras possíveis sobre Σ excetuando-se a palavra vazia $\Rightarrow \Sigma^+ = \Sigma^* - \varepsilon$

Linguagem Formal

- Linguagem formal: é um conjunto de palavras sobre um alfabeto
  - Exemplo: considere o alfabeto $\Sigma = \{a, b\}$. São exemplos de linguagens sobre Σ:
    - Conjunto vazio: = { }
    - Conjunto formado pela palavra vazia = { ε }
    - Conjunto de palíndromos (palavras que têm a mesma leitura da esquerda para a direita e da direita para a esquerda): {ε, a, b, aa, bb, aba, aaaa,...} $\Rightarrow$ linguagem infinita

Gramática

- Gramática: é uma quádrupla ordenada G = (V, T, P, S), onde:
  - V é um conjunto finito de símbolos **variáveis** ou **não-terminais**
  - T é um conjunto finito de símbolos **terminais** disjunto de V
  - P é um conjunto finito de pares, denominados **regras de produção**, onde a primeira componente é palavra de $(V \cup T)^+$ e a segunda componente é palavra de $(V \cup T)^*$
  - S é um elemento de V denominado **variável inicial**

- Uma regra de produção (α, β) é representada por $\alpha \rightarrow \beta$
- As regras de produção definem as condições de geração das palavras da linguagem $\Rightarrow$ gramática é formalismo de geração
- $\alpha \rightarrow \beta_1, \alpha \rightarrow \beta_2, ..., \alpha \rightarrow \beta_n \equiv \alpha \rightarrow \beta_1 \mid \beta_2 \mid ... \mid \beta_n$

- **Exemplo aplicado:** Mini Pascal

Gramática...

- Derivação: seja G = (V, T, P, S), uma gramática. Uma derivação é um par da relação denotada por $\Rightarrow$ com domínio em $(V \cup T)^+$ e contra-domínio em $(V \cup T)^*$. Logo, uma derivação (α, β) é representada por $\alpha \Rightarrow \beta$
  - A derivação é a substituição de uma subpalavra de acordo com uma regra de produção

Gramática...

- Linguagem gerada: seja G = (V, T, P, S), uma gramática. A linguagem gerada pela gramática G, denotada por L(G), é composta por todas as palavras de símbolos terminais deriváveis a partir do símbolo inicial S, ou seja:

$$L(G) = \{w \in T^* \mid S \Rightarrow^+ w\}$$

- Gramáticas equivalentes: duas gramáticas G_1 e G_2 são ditas equivalentes se, e somente se, $L(G_1) = L(G_2)$.

Gramática...

- Convenções: utiliza-se, para:
  - Símbolos variáveis: A, B,..., S, T
  - Símbolos terminais: a, b,..., s, t
  - Palavras de símbolos terminais: u, v, w, x, y, z
  - Palavras de símbolos terminais ou variáveis: α, β...
- Exemplos (gramática, derivação e linguagem gerada):
  - $V = \{S, A, B\}$, $T = \{a, b\}$, $P = \{S \rightarrow AB, A \rightarrow a, B \rightarrow b\}$
  - <Exemplo 22, pgs. 24 e 25, ref. Menezes>

Sistema de Estados Finitos

- Sistema de Estados Finitos: é um modelo matemático de sistema com entradas e saídas discretas
- Pode assumir um número finito e pré-definido de estados
- Cada estado resume somente as informações do passado necessárias para determinar as ações para a próxima entrada
- Exemplo clássico: elevador. Não memoriza as requisições anteriores. Cada “estado” sumariza as informações “andar corrente” e “direção de movimento”. As entradas são as requisições pendentes
- Exemplos: autômatos finitos (determinísticos ou não), autômatos com pilha e máquinas de Turing

Linguagens Formais – Contextos

- Teoria de linguagens
  - Projetos de Linguagens de Programação
  - Reconhecedores e tradutores de Linguagens de Programação. Especificamente, por exemplo, nas fases de análise léxica e sintática
- Implementação de linguagens
  - Empregada em técnicas de desenvolvimento de compiladores
- Problemas computacionais
  - Útil no estudo de viabilidade computacional de soluções de problemas
- Conceitos de linguagens de programação
  - Útil para melhor programação e projetos de linguagens

Exemplo aplicado da definição de uma LP – *Syntax of Mini-Pascal (Welsh & McKeag, 1980)* – https://www.cs.helsinki.fi/u/vihavain/k10/okk/minipascal/minipascalsyntax.html

Syntax in recursive descent order

<program> ::= program <identifier> ; <block> .

<block> ::= <variable declaration part>

<procedure declaration part>

<statement part>

<variable declaration part> ::= <empty> | var <variable declaration> ; { <variable declaration> ; }

<variable declaration> ::= <identifier > { , <identifier> } : <type>

<type> ::= <simple type> | <array type>

<array type> ::= array [<index range>] of <simple type>

<index range> ::= <integer constant> .. <integer constant>

<simple type> ::= <type identifier>

<type identifier> ::= <identifier>

...

Voltar

Exemplo aplicado da definição de uma LP – *Syntax of Mini-Pascal (Welsh & McKeag, 1980)* – https://www.cs.helsinki.fi/u/vihavain/k10/okk/minipascal/minipascalsyntax.html

(continuação)

Lexical grammar

<constant> ::= <integer constant> | <character constant> | <constant identifier>

<constant identifier> ::= <identifier>

<identifier> ::= <letter> { <letter or digit> }

<letter or digit> ::= <letter> | <digit>

<integer constant> ::= <digit> { <digit> }

<character constant> ::= < any character other than ' >' | ''''

<letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<special symbol> ::= + | - | * | = | <> | < | > | <= | >= | (|) | [|] | := | . | , | ; | : | .. | div | or | and | not | if | then | else | of | while | do | begin | end | read | write | var | array | procedure | program

<predefined identifier> ::= integer | Boolean | true | false

Voltar