Universidade de Aveiro

Departamento de Electrónica, Telecomunicações e Informática

Linguagens Formais e Autómatos

Exame (Ano Lectivo de 2013/14) 19 de Junho de 2014

1. Sobre o alfabeto $T_1 = \{\texttt{a b c d e}\}$ considere a gramática G_1 dada a seguir e seja L_1 a linguagem por ela descrita.

$$\begin{aligned} S &\to A\ B\ C \\ A &\to \lambda \mid \texttt{a}\ A \\ B &\to \texttt{b}\ B\ \texttt{e} \mid \texttt{b}\ C\ \texttt{e} \\ C &\to A \mid \texttt{c}\ C\ \texttt{d} \\ D &\to \texttt{d}\ D \mid \texttt{d}\ E \\ E &\to \lambda \mid \texttt{e}\ E \end{aligned}$$

Avalie a veracidade das seguintes afirmações, justificando adequadamente cada uma das suas respostas.

[1,0] (a) $\texttt{abecd} \in L_1$.

[1,0] (b) $\lambda \in L_1$.

[1,0] (c) Todos os símbolos não terminais são produtivos.

[1,0] (d) O símbolo não terminal E é acessível.

[1,0] (e) $\texttt{b} \in \texttt{first}(A\,B\,C)$.

[1,0] (f) $\texttt{c} \in \texttt{first}(A\,B\,C)$.

[1,0] (g) $\texttt{c} \in \texttt{predict}(B \to \texttt{b}\ C\ \texttt{e})$.

[1,0] (h) $\$ \in \texttt{predict}(C \to A)$.

[1,0] (i) G_1 é uma gramática independente do contexto, mas não é regular.

[1,0] (j) G_1 é inadequada à implementação direta de um reconhecedor descendente.

---

2. Considere a linguagem $L_2 = \{\texttt{b}^n\texttt{c}^m\texttt{d}^m\texttt{e}^n\ :\ n > 0\ \wedge\ m \geq 0\}$.

[1,0] (a) Mostre que $L_2 \subset L_1$, sendo L_1 a linguagem da pergunta anterior.

[2,0] (b) Projete um autómato de pilha que reconheça a linguagem L_2.

---

3. Sobre o alfabeto $T_3 = \{\texttt{m}, \texttt{t}, \texttt{s}, \texttt{v}, \texttt{=}, \texttt{e}\}$ considere a gramática G_3 dada a seguir e seja L_3 a linguagem por ela descrita.

$$\begin{aligned} S &\to D \\ D &\to \texttt{t}\ L \mid \texttt{m}\ \texttt{t}\ L \\ L &\to V \mid L\ \texttt{s}\ V \\ V &\to \texttt{v} \mid \texttt{v}\ \texttt{=}\ \texttt{e} \end{aligned}$$

Considere ainda a coleção canónica de conjuntos de itens, usada na construção de um reconhecedor (*parser*) ascendente, parcialmente apresentada a seguir e onde a função $\delta(Z_i, a)$ representa a transição de estado.

$Z_0 = \{ S \to \bullet D , D \to \bullet \texttt{t} \; L , D \to \bullet \texttt{m} \; \texttt{t} \; L \}$

$Z_1 = \delta(Z_0, D) = \{ S \to D \bullet \}$

$Z_2 = \delta(Z_0, \texttt{t}) = \{ D \to \texttt{t} \bullet L , L \to \bullet V , L \to \bullet L \; \texttt{s} \; V , V \to \bullet \texttt{v} , V \to \bullet \texttt{v} \; \texttt{=} \; \texttt{e} \}$

$Z_3 = \delta(Z_0, \texttt{m}) = \{ D \to \texttt{m} \bullet \texttt{t} \; L \}$

$Z_4 = \delta(Z_2, L) = \{ D \to \texttt{t} \; L \bullet , L \to L \bullet \texttt{s} \; V \}$

$Z_5 = \delta(Z_2, V) = \{ L \to V \bullet \}$

$Z_6 = \delta(Z_2, \texttt{v}) = \{ \dots \}$

$Z_7 = \delta(Z_3, \texttt{t}) = \{ \dots \}$

$Z_8 = \delta(Z_4, \texttt{s}) = \{ \dots \}$

$\dots$

[1,0] (a) Trace as árvores de derivação das palavras “`m t v = e`” e “`t v s v`”.

[2,0] (b) Preencha as linhas da tabela de reconhecimento (*parsing*) para um reconhecedor ascendente relativamente aos estados Z_0 a Z_5.

[2,0] (c) Determine o valor dos estados (conjuntos) Z_6 a Z_8 e de mais dois à sua escolha.

[2,0] (d) A gramática G_3 representa uma abstração de uma declaração de variáveis. O terminal `t` representa o tipo; o terminal `m` representa um modificador. Considere que:

- o símbolo terminal `t` possui um atributo chamado `type` que representa o tipo específico que lhe está associado.
- o símbolo terminal `v` tem um atributo chamado `name` que representa o nome da variável que lhe está associado.
- o símbolo terminal `e` tem um atributo chamado `value` que representa uma grandeza númerica.
- se dispõe de uma função de manipulação de uma tabela de símbolos para inserções de novas entradas, com a assinatura `addsym(n, t, v, m)`, onde
  - `n` representa o nome da variável;
  - `t` representa o tipo específico;
  - `v` representa o valor a atribuir à variável;
  - `m` é um valor booleano, que indica se o modificador está presente ou não.

Construa uma gramática de atributos que permita invocar a função `addsym` de forma adequada por cada constante ou variável declarada.

ALGORITMO do `first`:

```
first(α) {
    if (α == λ) then
        return {λ}
    else if (α == a and a ∈ T) then
        return {a}
    else if (α == B and B ∈ N) then
        M = {}
        foreach (B → γ) ∈ P
            M = M ∪ first(γ)
        return M
    else        /* |α| > 1 */
        x = head(α)        /* the first symbol */
        β = tail(α)        /* all but the first symbol */
        M = first(x)
        if λ ∉ M then
            return M
        else
            return (M − {λ}) ∪ first(β)
}
```

ALGORITMO do `follow`:

1. $\$ \in \texttt{follow}(S)$.
2. se $(A \to \alpha\, B) \in P$, então $\texttt{follow}(B) \supseteq \texttt{follow}(A)$.
3. se $(A \to \alpha\, B\, \beta) \in P$ e $\lambda \notin \texttt{first}(\beta)$, então $\texttt{follow}(B) \supseteq \texttt{first}(\beta)$.
4. se $(A \to \alpha B \beta) \in P$ e $\lambda \in \texttt{first}(\beta)$, então $\texttt{follow}(B) \supseteq ((\texttt{first}(\beta) - \{\lambda\}) \cup \texttt{follow}(A))$.

ALGORITMO do `predict`:

$$\texttt{predict}(A \to \alpha) = \begin{cases} \texttt{first}(\alpha) & \lambda \notin \texttt{first}(\alpha) \\ (\texttt{first}(\alpha) - \{\lambda\}) \cup \texttt{follow}(A) & \lambda \in \texttt{first}(\alpha) \end{cases}$$