Aveiro University

Department of Electronics, Telecommunications and Informatics

Compilers

Theory and practice exam, part 2 (School Year 2022-2023) Sample Exam

. . . .

Course: **Name:** **MecNo:**

`predict` ALGORITHM:

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

ALGORITHM of `first`:

```
first( α ){
    if (α = ε) then
        return {ε}
    h = head (α)        # with |h| = 1
    ω = tail (α)        # such that α = h ω
    if ( h ∈ T ) then
        return {h}
    else
        return   ⋃_{(h→βi) ∈ P}   first(βi ω)
}
```

ALGORITHM of `follow`:

1. $\$ \in \texttt{follow}(S)$
2. if $(A \to \alpha B \in P)$ then
   $\texttt{follow}(B) \supseteq \texttt{follow}(A)$
3. if $(A \to \alpha B \beta \in P) \wedge (\varepsilon \notin \texttt{first}(\beta))$ then
   $\texttt{follow}(B) \supseteq \texttt{first}(\beta)$
4. if $(A \to \alpha B \beta \in P) \wedge (\varepsilon \in \texttt{first}(\beta))$ then
   $\texttt{follow}(B) \supseteq ((\texttt{first}(\beta) - \{\varepsilon\}) \cup \texttt{follow}(A))$

---

1. About the alphabet $T_1 = \{\texttt{t b z w a o v n}\}$ consider the grammar G_1 given below and let L_1 be the language it describes.

```
P → o | X I t P | X b P z P
X → o | w C
I → ε | a
C → T | C o T
T → v | n T
```

[1,5] (a) Show that a t w n v b z $\in L_1$.

[1,5] (b) Evaluate the truth of the statement $\{\texttt{w, t}\} \subset \texttt{first}(\texttt{X I t P})$.
Present the appropriate intermediate steps and/or reasoning to support your answer.

[1,5] (c) Evaluate the truth of the statement $\texttt{t} \in \texttt{follow}(\texttt{T})$.
Present the appropriate intermediate steps and/or reasoning to support your answer.

[2,0] (d) Compute the set $\texttt{predict}(\texttt{P} \to \texttt{X I t P})$.
Present the appropriate intermediate steps and/or reasoning to support your answer.

[2,0] (e) The productions started by P and C make the grammar G_1 unsuitable for the implementation of a top-down re-cognizer with *lookahead* of 1. Change it in order to obtain an equivalent one that allows it.

---

2. Consider the alphabet $A = \{\texttt{a, b, c}\}$ and let L_2 be the set of all regular expressions definable over the

continues on the next page

alphabet A. L_2 is a context-independent language defined over the alphabet $T_2 = A \cup \{(,\ ),\ *,\ +\}$, where * represents the closure operator and + the choice operator; concatenation operation has the ope- cation. implicit rator. In terms of precedence, from highest to lowest, are the operations closure, concatenation, and choice. Parentheses can be used to change the default precedence.

[3,0] (.) Construct a context-independent grammar that represents the language L_2 .

continues on the next page

3. About the alphabet T_3 = {NUM, BOX, CIRCLE, THICKNESS, COLOR, '{', '}}, consider the grammar G_3 given below and let
L_3 the language she described.

draw → *seq*
seq → ε
| *seq item*
item → COLOR
NUM
| THICKNESS NUM
| CIRCLE *point* NUM
| BOX *point* '{' *seq* '}'
point → NUM NUM NUM

Consider further the set of states (itemsets) used in the construction of a partially ascending recognizer shown below, where $\delta(Z_i , a)$ represents the state transition function.

Z0 = {*draw* → - *seq* , *seq* → - , *seq* → - *seq* item}
Z1 = δ(Z0, *seq*) = {*draw* → *seq* - , *seq* → seq - *item* , *item* → - COLOR NUM , *item* → - THICKNESS NUM ,
item → - CIRCLE *point* NUM , *item* → - BOX *point* '{' *seq* '}'}
Z2 = δ(Z1, *item*) = {*seq* → *seq item* -}
Z3 = δ(Z1, COLOR) = {*item* → COLOR - NUM}
Z4 = δ(Z1, THICKNESS) = {*item* → THICKNESS - NUM}
$Z_5 = \delta(Z_1$, CIRCLE) = {- - - }
$Z_6 = \delta(Z_1$, BOX) = {- - - }
Z7 = δ(Z3, NUM) = {*item* → COLOR NUM -}
Z8 = δ(Z4, NUM) = {*item* → THICKNESS NUM -}

[2,0] (a) Fill in the rows of the *parsing* table for an ascending recognizer for states Z_0 to Z_4 .

	NUM	BOX	CIRCLE	THICKNESS	COLOR	{	}	$	*draw*	*seq*	*item*	*point*
Z_0												
Z_1												
Z_2												
Z_3												
Z_4												

[2,0] (b) Determine the item sets defining the states Z_5 , Z_6 and three more, in addition to those shown.

continues on the next page

4. Consider again the grammar G_3 given in the previous exercise. A word in the language given by G_3 describes a drawing defined by a sequence of the following graphic operations (*item*):

- `COLOR NUM`, which allows you to change the color of the drawing pen to that given by `NUM`.
- `THICKNESS NUM`, which allows you to change the thickness of the drawing pen to that given by `NUM`.
- `CIRCLE` *point* `NUM`, which draws a circle centered at the point given by *point* and with radius given by `NUM`, using the active drawing pen.
- `BOX` *point* `'{'` *seq* `'}'`, which creates a subdrawing with an *offset* given by *point* relative to the drawing in which it lies. The point (0,0) of the sub-drawing is the *point of* the drawing in which it is included.

Only the terminal symbol `NUM` has an associated attribute, called *v*, which represents a number. The non-terminal symbol *point* represents the X and Y coordinates of a point. The initial configuration of the system is characterized by color 0, thickness 1, and *offset* (0,0). Finally, consider that you have the function `drawCircle(x, y, r, c, t)` which draws a circle centered at the point `(x,y)`, with radius `r`, using a drawing pen with color `c` and thickness `t`.

[1,5] (a) Trace the derivation tree of the word

`COLOR NUM CIRCLE NUM NUM NUM BOX NUM NUM NUM '{' THICKNESS NUM CIRCLE NUM NUM NUM NUM '}'`

If you like, when tracing the tree, you can abbreviate the name of the symbols, using `N`, `CI`, `CO`, `T`, `B`, *s*, *i*, and *p* instead of `NUM`, `CIRCLE`, `COLOR`, `THICKNESS`, `BOX`, *seq, item*, and *point*, respectively.

[3,0] (b) Complete the attribute grammar below such that it properly invokes the `drawCircle` function for each circle included in a description in L_3.

Production	**Semantic rule**
draw → *seq*	
seq → ε	
seq → *seq item*	
item → `COLOR NUM`	
item → `THICKNESS NUM`	
item → `CIRCLE` *point* `NUM`	`drawCircle(`*point*`.x, `*point*`.y, NUM.v, ...`
item → `BOX` *point* { *seq* }	
point → `NUM NUM`	