Department of Information and Computing Sciences
Utrecht University

# INFOB3TC – Final Exam

Alejandro Serrano, Trevor L. McDonell

Tuesday, 29 January 2019, 11:00–13:00

- Do **not** turn the page until instructed.
- Write your **name** and **student number** in every page you hand in.
- Give **clear** and **concise** answers. Use a blue or black pen.
- Answer questions in English or Dutch.

**Important.** In every grammar in this exam, the **start symbol** is the **first** nonterminal appearing in its description.

**Exercise 1 (1.5 points).** Give **both** a deterministic finite state automaton (DFA) **and** a regular expression which accepts the language described by the following grammar.

$S \to 0S$
$S \to 1T$
$S \to \varepsilon$
$T \to 0T$
$T \to 1S$

**Exercise 2 (1.5 points).** Given the following grammar:

$S \to 0S0$
$S \to 1T1$
$S \to \varepsilon$
$T \to 0T0$
$T \to 1S1$

(a) Is this grammar a regular grammar? Explain why or why not.

(b) Prove that it defines a non-regular language by using the **pumping lemma**.
Hint: use the word $10^n110^n1$.

**Exercise 3 (3 points).** Given the following grammar:

$T \to \mathtt{a}T\mathtt{c}$
$T \to B$
$B \to \mathtt{b}B$
$B \to \varepsilon$

(a) Execute the Expand-Match algorithm without lookahead over the string aabcc so that it leads to acceptance.
   - Describe at each step the state of the stack and the input string.
   - Mark those places where the machine performs a non-deterministic choice.

(b) Compute *empty*, *first*, and *follow* for each nonterminal, and the lookahead sets for each rule. Is this grammar LL(1)?

(c) Describe how the parsing of aabcc proceeds with this new information.

**Exercise 4 (2 points).** Given the following grammar:

$S \to \mathtt{a}A$
$S \to AB$
$A \to \mathtt{a}A\mathtt{b}$
$A \to \mathtt{b}$
$B \to \mathtt{ab}$
$B \to \mathtt{b}$

(a) Construct the LR(0) automaton.

(b) Classify each state as shift state, reduce state, shift-reduce conflict, or reduce-reduce conflict.

**Exercise 5 (2 points).** Give constructions for the following closure properties:

(a) Regular languages are closed by *reversal*, where the reverse of a word $s_1 s_2 \ldots s_{n-1} s_n$ is defined as $s_n s_{n-1} \ldots s_2 s_1$.

Hint: consider a finite state automaton for a language $L$, and build the corresponding finite state automaton for its reversal. Another possibility is to consider a regular expression describing $L$ and then building the corresponding regular expression for its reversal.

(b) Given a context-free language $X$ and a regular language $Y$, its intersection $X \cap Y$ is context-free.

Hint: consider the pushdown automaton for $X$ and the finite state automaton for $Y$, and build a pushdown automaton which recognizes $X \cap Y$.

In both cases, you do **not** need to formally prove that the automata or expression you have constructed accepts the language upon consideration. However, you must give an **intuitive** account of why it works.

**Exercise 6 (bonus, 1 point).** Consider the following Haskell data type which describes regular expressions over sequences of symbols of type $a$:

```
data RegExp a = Empty
              | Symbol   a
              | Choice   (RegExp a) (RegExp a)
              | Sequence (RegExp a) (RegExp a)
              | Star     (RegExp a)
```

Write a function which transforms a value of *RegExp a* into a parser using parser combinators. The result of a successful parse should be the string itself.

```
toParser :: RegExp a → Parser a [a]
```