UNIVERSITÉ LIBRE DE BRUXELLES

INFO-F-403 – Introduction to language theory and compiling
First session examination

January, 22nd, 2018

Instructions

- This is a closed book test. You are not allowed to use any kind of reference.
- You can anwser in French or in English.
- Write your first and last names on each sheet that you hand in.
- Write clearly: you can use a pencil or a ballpen or even a quill as long as your answers are readable!
- Always provide full and rigourous justifications along with your answers.
- This test is worth 12 points out of 20. The weight of each questions is given as a reference.
- In your answers (diagrams representing automata, grammars,...), you can always use the conventions adopted in the course, without recalling them explicitely. If you deviate from these conventions, be sure to make it clear.

Question 1 — 4 points

1. Give the algorithm that transforms an ε-NFA into an equivalent DFA (accepting the same language).
2. We want to build a DFA for the following language L: 'The set of all words (of length ≥ 3) on the alphabet $\{0, 1\}$ in which every sequence of 3 characters contains at least one 1'. For example, the words 00100, 1111 and 0101 are in L, while 10001 and 000 are not. To obtain such a DFA, apply the following steps:
   (a) build an NFA that accepts the *complement* of L (it's much easier to build an NFA than a DFA to accept the complement of L, so try and exploit non-determinism);
   (b) turn this NFA into an equivalent DFA using the procedure described above; and
   (c) compute the complement of this DFA.

Question 2 — 2 points

1. Give the syntax and the semantics of *Pushdown Automata*. For the semantics, make sure to define both the final state accepted language $L(P)$ and the empty stack accepted language $N(P)$. Don't forget to define formally the notions of configuration and move (from one configuration to the other).

2. Give a language that is *not regular* and that is accepted by a PDA. Prove that the language you give is not regular and give a PDA that accepts it.

Question 3 — 3 points

Let us consider the following grammar to generate Boolean expressions:

(1)	E	$\rightarrow$	$E \vee E$
(2)		$\rightarrow$	$E \wedge E$
(3)		$\rightarrow$	$E \oplus E$
(4)		$\rightarrow$	(E)
(5)		$\rightarrow$	$\neg E$
(6)		$\rightarrow$	v

where the $\vee$, $\wedge$, $\oplus$ and $\neg$ operators are the logical *or*, *and*, *exclusive or* and *negation* respectively. v is a lexical unit that corresponds to the variables of the expression. Priority of the operators is as follows (from higher to lower priority):

— $()$ and $\neg$

— $\wedge$

— $\vee$ and $\oplus$

You are asked to:

1. Modify the grammar to let it take into account the priority and (left) associativity of the operators.
2. Modify further the resulting grammar to make it $\mathrm{LL}(1)$.
3. Give the action table of the $\mathrm{LL}(1)$ parser.

Question 4 — 3 points

Consider the following grammar, where the set of variables is $\{S', S, A, B, C\}$, and the set of terminals is $\{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}, \$\}$:

(1)	S'	$\rightarrow$	$S\$$
(2)	S	$\rightarrow$	$A\texttt{b}$
(3)	S	$\rightarrow$	$B\texttt{b}$
(4)	A	$\rightarrow$	$C\texttt{a}$
(5)	A	$\rightarrow$	$C\texttt{ac}$
(6)	A	$\rightarrow$	ε
(7)	C	$\rightarrow$	$\texttt{d}A$
(8)	B	$\rightarrow$	$\texttt{e}$
(9)	B	$\rightarrow$	$\texttt{f}$

1. Give the rightmost derivation of `ddacab$`, according to this grammar.
2. Give the $\mathrm{LR}(1)$ analyser for this grammar. To this end, give the CFSM and the $\mathrm{LR}(1)$ action table.
3. Is the grammar $\mathrm{LR}(0)$? Justify your answer.
4. Simulate the run of your analyser on the input `ddacab$`. For each step of the analysis, give the content of the stack, and the remaining input.