Laurea Magistrale in INFORMATICA
Principi di Linguaggi di Programmazione
Compilatori

prof. M. Bellia
Appello III - june 4th, 2013

(Timing: 2 hours – Grading: (pts n-m) is the score range to be obtained in each exercise)

Exercise 1. (pts 3 – 5) Let E≡ a (a* | b*) b be a regular expression.

(a) Give the dotted automaton of E (show the computation and the set od items of each state);
(b) By using the minimization algorithm, prove the minimality of the automaton that has been obtained in (a).

Exercise 2. (pts 5 - 10)

(a) Compute the Canonical Collection, Coll(1), of the LR(1) parser of the grammar G below:
S::= aSS | Sb | c
(b) Give the Parsing Table of the LALR(1) parser of G
(c) Show the behaviour of the shift/reduce automaton during the analysis of: acacbc.

Exercise 3. (pts 7 - 15) Let G be an LR grammar for Boolean expressions with *disjunction*, _or_, *negation*, not_, *conditional*, _?_:_, grouping, variables and the literals *true*, e *false*. All the operators have left associativity and precedence as follow: ? > not > or.

(a) Define G and show that it recognize the expression: x or not y ? not not y : z
(b) Give an oblivious, ascendant, translation scheme for the generation of 3AC code with loc as the invariant.
(c) Give an oblivious, ascendant, translation scheme for the generation of 3AC code with target-uncomplete statement lists (short-circuit)
(d) Apply the scheme in (b) in order to provide the code generation of the expression:
x or not y ? not not y : z.
(e) Apply the scheme in (c) in order to provide the code generation of the expression:
x or not y ? not not y : z,
also showing the value of the list the two attributes of the root of the expression parse-tree.

Assume that the following operators and constants be available in 3AC: [or] for _or_, [not] for not_, #T for true and #F for false. Finally, assume the following association in symbol table: loc_x for x.loc, loc_y for y.loc, loc_z for z.loc.