

## Compiler Construction WS11/12

### Exercise Sheet 6

Please hand in the solutions to the theoretical exercises until the beginning of the lecture next Friday 2011-12-02, 12:00. Please write the number of your tutorial group or the name of your tutor on the first sheet of your solution.

#### Exercise 6.1 Attribute Grammar *BoolExp* (Points: 4+2+8+2+2)

Consider the attribute grammar *BoolExp* for the short circuit evaluation of boolean expressions as presented in the lecture and specified on the slides about attribute grammars (pages 20 to 22).

- a) For each grammar production, list all attribute occurrences and classify them as defining or applied occurrences. Note: Remember that some identity semantic rules may be omitted in the specification of the grammar, like for example for productions from non-terminal to non-terminal.

- b) Draw the parse tree for the following word.

**if *a* and ( *b* or *c* ) or *d* and *e* then *yesStats* else *noStats* fi**

Add to each node in the tree all the attribute instances calculated for that node according to the attribute grammar *BoolExp*.

- c) Give the equation system on the attribute instances of the tree nodes in exercise b).

Assume that *a*, *b*, *c*, *d* and *e* are *id* tokens and each of them has got its name as value of the *identifier* field. Further assume that *yesStats* and *noStats* are instances of *STATS* and each of them has got its name as value of the *code* field. All these values are statically known before the evaluation of the attribute instances according to *BoolExp*.

Evaluate the attribute instances.

- d) Explain why the attribute grammar *BoolExp* is non-circular.

- e) Explain why a single top-down pass followed by a single bottom-up pass of an evaluator on a syntax tree is sufficient to determine the values of all attribute instances of the tree nodes according to the attribute grammar *BoolExp*.