Assignment 5 – COMP 523
Language-based security

Brigitte Pientka

Winter 2008
Due April 15, 2008

Exercise 1 (30 pts) Extend the Twelf implementation of the MiniML language together with its meta-theory with recursive types.

1. Extend the syntax for types and Mini-ML expressions, and values.
2. Extend the small-step evaluation rules.
3. Extend type-preservation proof.
4. Extend progress proof.

Exercise 2 (70 pts) : The big-step operational semantics we have discussed so far is not very efficient in the sense that it re-evaluates expressions although they are already values. Consider for example the expression: (fn x. (s x)) (s (s z)). Using the big-step semantics the expression (s (s z)) will be evaluated twice.

Explore an alternative operational semantics in which expressions that are known to be values (since they have been evaluated) are not evaluated again. State and prove in which way the new semantics is equivalent to the one given in the file `eval.elf`.

Hint: It may be necessary to extend the language of expressions or explicitly separate the language of values from the language of expressions).

1. Define the syntax of a language MiniML^v where expressions which are values are not re-evaluated. Define your syntax in Twelf.
2. Define the operational semantics for MiniML^v in Twelf.

3. Prove that the operational semantics for MiniML^v and the original semantics for MiniML are equivalent, and implement your proof in Twelf.