Assignment 2 – COMP 523
Language-based security

Brigitte Pientka

Winter 2008
Due Feb 7th 2008

1 Lazy evaluation (60 pts)

Consider the following extension of Mini-ML, which supports lazy evaluation via a new type $\mathsf{susp}\ \tau$ and the two new expressions $\mathsf{delay}\ e$ and $\mathsf{let\ delay}\ x = e_1\ \mathsf{in}\ e_2$. $\mathsf{delay}\ e$ suspends the evaluation of e. $\mathsf{let\ delay}\ x = e_1\ \mathsf{in}\ e_2$ allows us to continue evaluating an expression e_1 which has been suspended.

Typing Rules:

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{delay}\ e : \mathsf{susp}\ \tau} \qquad \frac{\Gamma \vdash e_1 : \mathsf{susp}\ \tau' \quad \Gamma, x{:}\tau' \vdash e_2 : \tau}{\Gamma \vdash \mathsf{let\ delay}\ x = e_1\ \mathsf{in}\ e_2 : \tau}$$

Evaluation Rules (Big-step):

$$\frac{}{\mathsf{delay}\ e \Downarrow \mathsf{delay}\ e} \qquad \frac{e_1 \Downarrow \mathsf{delay}\ e' \qquad [e'/x]e_2 \Downarrow v}{\mathsf{let\ delay}\ x = e_1\ \mathsf{in}\ e_2 \Downarrow v}$$

1. (10 pts) Define a function force which has type $\mathsf{susp}\ \alpha \rightarrow \alpha$ for a type variable α. $\mathsf{force}\ e$ forces the evaluation of e, i.e. e will be evaluated.

2. (10 pts) Prove that $\mathsf{force}(\mathsf{delay}\ (e))$ evaluates to v if and only if e evaluates to v according to our new operational semantics.

3. (10pts) Give the type preservation proof for the rules above.

4. (10 pts) Show how type preservation breaks down when we choose the following typing rule:

$$\frac{\Gamma \vdash e_1 : \mathsf{susp}\ \tau' \quad \Gamma \vdash [e_1/x]e_2 : \tau}{\Gamma \vdash \mathsf{let\ delay}\ x = e_1\ \mathsf{in}\ e_2 : \tau}$$

5. (10 pts) Extend the values for Mini-ML and prove value-soundness for the new constructs, i.e. if $e \Downarrow v$ then v is a value.

6. (10 pts) Another choice of primitives to model suspension are $\mathsf{delay}\ e$ and $\mathsf{force}\ e$. State the appropriate evaluation rule for $\mathsf{force}\ e$ and compare this to the primitives $\mathsf{delay}\ e$ and $\mathsf{let\ delay}\ x = e_1\ \mathsf{in}\ e_2$ used above. Do you see any advantages or disadvantages?

2 Case-statement(40 points)

An alternative definition for numbers is as follows:

$$
\begin{array}{lll}
\text{Terms } t & ::= & x \mid \mathsf{z} \mid \mathsf{succ}\ t \mid (\mathsf{case}\ t\ \mathsf{of}\ \mathsf{z} \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2) \\
\text{Types } T & ::= & \textsc{nat}
\end{array}
$$

Here we can analyze numbers using a case-expression where we pattern match against the possible shapes of numbers. So, if the subject t of the case-expression $\mathsf{case}\ t\ \mathsf{of}\ \mathsf{z} \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2$ evaluates to z then we choose the first branch t_1. Otherwise t must evaluate to some value of the form $\mathsf{succ}\ v$. In this case we match $\mathsf{succ}\ x$ against $\mathsf{succ}\ v$ which will yield the instantiation of x to v. We then proceed to evaluate the second branch t_2 under this instantiation by applying the substitution $[v/x]$ to t_2. The evaluation for these terms can be then defined as follows:

$$
\frac{}{\mathsf{z} \Downarrow \mathsf{z}} \qquad \frac{t \Downarrow v}{\mathsf{succ}\ t \Downarrow \mathsf{succ}\ v}
$$

$$
\frac{t \Downarrow \mathsf{z} \quad t_1 \Downarrow v}{\mathsf{case}\ t\ \mathsf{of}\ \mathsf{z} \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2 \Downarrow v} \qquad \frac{t \Downarrow \mathsf{succ}\ v_2 \quad [v_2/x]t_2 \Downarrow v}{\mathsf{case}\ t\ \mathsf{of}\ \mathsf{z} \Rightarrow t_1 \mid \mathsf{succ}\ x \Rightarrow t_2 \Downarrow v}
$$

1. (5pts) Assuming we also have functions, function application, and booleans, show how we can define functions for predecessor and iszero as abbreviations.

2. (5pts) Define the appropriate typing rule for the case-expression.

3. (10pts) Show that type preservation holds for this rule.

4. (10pts) Give the corresponding small-step evaluation rules.
5. (10 pts) Show progress holds for the small step semantics you propose.