



BU CAS CS 520 (FALL SEMESTER, 2009)
PRINCIPLES OF PROGRAMMING LANGUAGES

Solution 3

Total: 150 points + 50 extra points

The following is the syntax of the language STLC0 and its static and dynamic semantics.

Syntax

constants	$c ::= \mathbf{true} \mid \mathbf{false} \mid 0 \mid 1 \mid -1 \mid \dots$
operators	$op ::= + \mid - \mid * \mid / \mid \dots$
terms	$t ::= c \mid x \mid \mathbf{if } t_0 \mathbf{ then } t_1 \mathbf{ else } t_2 \mid op(t) \mid \lambda x : T.t \mid t_1(t_2)$
values	$v ::= c \mid \lambda x : T.t$
types	$T ::= Bool \mid Int \mid T_1 \rightarrow T_2$
contexts	$\Gamma ::= \emptyset \mid \Gamma, x : T$

Static Semantics

$$\frac{c \in \{\mathbf{true}, \mathbf{false}\}}{\Gamma \vdash c : Bool} \text{ (T-Bool)}$$
$$\frac{c \in \{0, 1, -1, \dots\}}{\Gamma \vdash c : Int} \text{ (T-Int)}$$
$$\frac{\Gamma \vdash t_0 : Bool \quad \Gamma \vdash t_1 : T \quad \Gamma \vdash t_2 : T}{\Gamma \vdash \mathbf{if } t_0 \mathbf{ then } t_1 \mathbf{ else } t_2 : T} \text{ (T-If)}$$
$$\frac{\Sigma(op) = T_1 \rightarrow T_2 \quad \Gamma \vdash t : T_1}{\Gamma \vdash op(t) : T_2} \text{ (T-Op)}$$
$$\frac{\Gamma(x) = T}{\Gamma \vdash x : T} \text{ (T-Var)}$$
$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \lambda x : T_1.t : T_1 \rightarrow T_2} \text{ (T-Abs)}$$
$$\frac{\Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1(t_2) : T_2} \text{ (T-App)}$$

Dynamic Semantics

$$\begin{array}{c}
\frac{t_0 \rightarrow t'_0}{\text{if } t_0 \text{ then } t_1 \text{ else } t_2 \rightarrow \text{if } t'_0 \text{ then } t_1 \text{ else } t_2} \text{ (E-If)} \\
\frac{}{\text{if true then } t_1 \text{ else } t_2 \rightarrow t_1} \text{ (E-IfTrue)} \\
\frac{}{\text{if false then } t_1 \text{ else } t_2 \rightarrow t_2} \text{ (E-IfFalse)} \\
\frac{t \rightarrow t'}{op(t) \rightarrow op(t')} \text{ (E-Op)} \\
\frac{op(v) = v'}{op(v) \rightarrow v'} \text{ (E-OpVal)} \\
\frac{t_1 \rightarrow t'_1}{t_1(t_2) \rightarrow t'_1(t_2)} \text{ (E-App1)} \\
\frac{t_2 \rightarrow t'_2}{v_1(t_2) \rightarrow v_1(t'_2)} \text{ (E-App2)} \\
\frac{}{(\lambda x.t_1)(v_2) \mapsto t_1[x \mapsto v_2]} \text{ (E-AppAbs)}
\end{array}$$

Theorem (Progress) Suppose that t is a closed and well-typed term in STLC0, that is, there exists a derivation $\mathcal{D} :: \emptyset \vdash t : T$. Then either t is a value or $t \rightarrow t'$ holds for some term t' .

Exercise 1 (30 pts) Please present a **complete** proof of the progress theorem by structural induction on the derivation $\mathcal{D} :: \emptyset \vdash t : T$.

Proof (sketch): By structural induction on \mathcal{D} . We present the case where the last rule applied in \mathcal{D} is **(T-App)**:

$$\frac{\mathcal{D}_1 :: \Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \mathcal{D}_2 :: \Gamma \vdash t_2 : T_1}{\mathcal{D} :: \Gamma \vdash t_1(t_2) : T_2} \text{ (T-App)}$$

Note that we have $T = T_2$ and $t = t_1(t_2)$ in this case.

- t_1 is not a value. By induction hypothesis on \mathcal{D}_1 , we have $t_1 \rightarrow t'_1$ for some t'_1 . Hence, we can derive $t = t_1(t_2) \rightarrow t'_1(t_2)$ by **(E-App1)**.
- t_1 is a value but t_2 is not a value. By induction hypothesis on \mathcal{D}_2 , we have $t_2 \rightarrow t'_2$ for some t'_2 . Hence, we can derive $t = t_1(t_2) \rightarrow t_1(t'_2)$ by **(E-App2)**.
- t_1 and t_2 are values. By the Lemma of Canonical Forms, t_1 is of the form $\lambda x.t_{10}$. So we can derive $t \rightarrow t_{10}[x \mapsto t_2]$ by **(E-AppAbs)**.

All of the rest of the cases are omitted.

Theorem (Subject Reduction) Suppose that we have $\mathcal{D} :: \Gamma \vdash t : T$ and $t \rightarrow t'$ in STLC0. Then $\Gamma \vdash t' : T$ is derivable.

Exercise 2 (50 pts) Please present a **complete** proof of the subject reduction theorem by structural induction on the derivation $\mathcal{D} :: \Gamma \vdash t : T$. You need to spot the place where substitution lemma is needed and then prove it, which is worth 20 pts (out of the 50 total pts).

Proof (sketch): By structural induction on \mathcal{D} . We present the case where the last rule applied in \mathcal{D} is **(T-App)**:

$$\frac{\mathcal{D}_1 :: \Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \mathcal{D}_2 :: \Gamma \vdash t_2 : T_1}{\mathcal{D} :: \Gamma \vdash t_1(t_2) : T_2} \text{ (T-App)}$$

Note that we have $T = T_2$ and $t = t_1(t_2)$ in this case.

- $t' = t'_1(t_2)$ for some t'_1 such that $t' \rightarrow t'_1$ holds. By induction on \mathcal{D}_1 , $\Gamma \vdash t'_1 : T_1 \rightarrow T_2$ is derivable. With \mathcal{D}_2 , we can derive $\Gamma \vdash t' : T_2$.
- t_1 is a value, and $t' = t_1(t'_2)$ for some t'_2 such that $t_2 \rightarrow t'_2$ holds. By induction on \mathcal{D}_2 , $\Gamma \vdash t'_2 : T_1$ is derivable. With \mathcal{D}_1 , we can derive $\Gamma \vdash t' : T_2$.
- t_1 and t_2 are values. Since $t \rightarrow t'$ is derivable, t_1 must be of the form $\lambda x.t_{10}$. Hence, \mathcal{D}_1 must be of the following form:

$$\frac{\mathcal{D}_{10} :: \Gamma, x : T_1 \vdash t_{10} : T_2}{\mathcal{D}_1 :: \Gamma \vdash \lambda x.t_{10} : T_1 \rightarrow T_2} \text{ (T-Abs)}$$

Note that $t' = t_{10}[x \mapsto t_2]$. By the Substitution Lemma, we have that $\Gamma \vdash t' : T_2$ is derivable.

All of the rest of the cases are omitted.

Exercise 3 (20 points + 20 extra points) A function named `power` is declared in the file `power.dats`, which is available on-line. Please implement the function `power`. To receive full credit, your implementation of `power` needs to be of $O(\log(n))$ -time complexity. You will receive 20 extra points if your implementation is also tail-recursive.

Exercise 4 (50 points + 30 extra points) In contrast to the (ordinary) list data structure, which only supports $O(n)$ -time list subscripting operations, the random-access list data structure by Chris Okasaki can support $O(\log(n))$ -time list subscripting operations while keeping $O(1)$ -time list consing and unconsing operations. In the file `ralist.sats`, which is available on-line, there are several list operations. Please implement these operations in a file named `ralist.dats`.