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

Syntax

\[
\begin{align*}
\text{constants} & \quad c ::= \text{true} | \text{false} | 0 | 1 | -1 | \cdots \\
\text{operators} & \quad op ::= + | - | \ast | / | \cdots \\
\text{terms} & \quad t ::= c \mid x \mid \text{if } t_0 \text{ then } t_1 \text{ else } t_2 \mid \text{op}(t) \mid \lambda x : T.t \mid t_1(t_2) \\
\text{values} & \quad v ::= c \mid \lambda x : T.t \\
\text{types} & \quad T ::= \text{Bool} \mid \text{Int} \mid T_1 \to T_2 \\
\text{contexts} & \quad \Gamma ::= \emptyset \mid \Gamma, x : T
\end{align*}
\]

Static Semantics

\[
\begin{align*}
\Gamma \vdash c : \text{Bool} & \quad \text{(T-Bool)} \\
\Gamma \vdash c : \text{Int} & \quad \text{(T-Int)} \\
\Gamma \vdash t_0 : \text{Bool} \quad \Gamma \vdash t_1 : T \quad \Gamma \vdash t_2 : T & \quad \text{(T-If)} \\
\Sigma(op) = T_1 \to T_2 \quad \Gamma \vdash t : T_1 & \quad \text{(T-Op)} \\
\Gamma(x) = T & \quad \text{(T-Var)} \\
\Gamma, x : T_1 \vdash t : T_2 & \quad \text{(T-Abs)} \\
\Gamma \vdash t_1 : T_1 \to T_2 \quad \Gamma \vdash t_2 : T_1 & \quad \text{(T-App)}
\end{align*}
\]
Dynamic Semantics

\[
\frac{t_0 \rightarrow t'_0}{\text{if } t_0 \text{ then } t_1 \text{ else } t_2 \rightarrow \text{if } t'_0 \text{ then } t_1 \text{ else } t'_2} \quad (\text{E-If})
\]

\[
\frac{\text{if true then } t_1 \text{ else } t_2 \rightarrow t_1}{(\text{E-IfTrue})}
\]

\[
\frac{\text{if false then } t_1 \text{ else } t_2 \rightarrow t_2}{(\text{E-IfFalse})}
\]

\[
\frac{t \rightarrow t'}{\text{op}(t) \rightarrow \text{op}(t') \quad (\text{E-Op})}
\]

\[
\frac{\text{op}(v) = v'}{\text{op}(v) \rightarrow v' \quad (\text{E-OpVal})}
\]

\[
\frac{t_1 \rightarrow t'_1}{t_1(t_2) \rightarrow t'_1(t_2) \quad (\text{E-App1})}
\]

\[
\frac{t_2 \rightarrow t'_2}{v_1(t_2) \rightarrow v_1(t'_2) \quad (\text{E-App2})}
\]

\[
(\lambda x.t_1)(v_2) \vdash t_1[x \mapsto v_2] \quad (\text{E-AppAbs})
\]

**Theorem** (Progress) Suppose that \( t \) is a closed and well-typed term in STLC0, that is, there exists a derivation \( D :: \emptyset \vdash t : T \). 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 \( D :: \emptyset \vdash t : T \).

**Theorem** (Subject Reduction) Suppose that we have \( D :: \Gamma \vdash t : T \) 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 \( D :: \Gamma \vdash t : T \). 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).

**Exercise 3** (20 pts) The following declared dataprop F91 encodes the MacCarthy’s 91-function:

\[
\text{dataprop F91 (int, int) =}
\]

\[
| \text{F91def1 (91, 91)}
| \{i:int | i <= 100; i <> 91\} \{r1,r2:int\}
\quad \text{F91def2 (i, r2) of (F91 (i+11, r1), F91 (r1, r2))}
| \{i:int | 101 <= i\} \{r:int\}
\quad \text{F91def3 (i, r) of F91 (i-10, r)}
\]

*Given integers \( i \) and \( r \), if a proof value can be assigned the type \( F91(i, r) \), then \( f91(i) = r \), where \( f91 \) is formally defined in Assignment 1. Please construct a proof function \( f91_lemma \) in ATS that is declared as follows:

\[
\text{prfun f91_lemma \{i,r:int\} (pf: F91 (i, r)): \[r==91\] void}
\]

\[
\text{Given integers} i \text{ and } r, \text{ if a proof value can be assigned the type } F91(i, r), \text{ then } f91(i) = r, \text{ where } f91 \text{ is formally defined in Assignment 1. Please construct a proof function } f91_lemma \text{ in ATS that is declared as follows:}
\]

\[
\text{prfun f91_lemma \{i,r:int\} (pf: F91 (i, r)): \[r==91\] void}
\]
Exercise 4 (50 extra pts) The definition of Braun trees is encoded into the following declared dataprop isBraun:

datasort bt = B of (bt, bt) | E of ()

dataprop isBraun (bt) =
  | {t1,t2:bt} {s1,s2:nat | s2 <= s1; s1 <= 1 + s2}
    isBraun_B (B (t1, t2)) of (isBraun t1, isBraun t2, btsz (t1, s1), btsz (t2, s2))
  | isBraun_E (E ()) of ()

Please construct a proof function brauntree_height_lemma in ATS that is declared as follows:

prfun brauntree_height_lemma {t1,t2:bt} {h,h1:nat}
  (pf0: isBraun (B (t1, t2)), pf1: btht (B (t1, t2), h), pf2: btht (t1, h1))
  : [h == h1+1] void

Please also construct a proof function brauntree_size_height_lemma in ATS that is declared as follows:

prfun brauntree_size_height_lemma {t:bt} {s,h,n:nat}
  (pf0: isBraun (t), pf1: btsz (t, s), pf2: btht (t, h), pf3: POW2 (h, n))
  : [n <= s + s + 1] void

Note that the dataprops btsz, btht and POW2 are all declared in a file available on-line.