Formal Semantics Review Problems

CSE505

November 8, 2001

In this exercise, you are going to add lists to the call-by-value simply-typed lambda calculus (with booleans). A list is defined recursively as either the value nil or the value (cons v_1 v_2), where v_2 is itself a list. For example, the list consisting of the booleans true, false, and true (in that order) would be represented by the following list value:

(cons true (cons false (cons true nil[Bool])))

The above list value has type Bool List. The value nil is annotated with the element type expected for this occurrence of the empty list. This annotation is necessary so that the type system doesn't have to "guess" the correct type of an empty list. The new primitives head and tail access the first and second components of a cons cell, respectively.

The new cases of the language's syntax are as follows:

e	::=	...
		wrong: T
		nil[T]
		cons e_1 e_2
		head e
		tail e
T	::=	...
		T List

The expression denoted wrong: T is a new value that is used to represent errors that will not be prevented by the static type system and can therefore not be considered stuck. In particular, the type system will not ensure that head and tail are always invoked on a non-empty list. The expression wrong is annotated with a type so that the type system does not need to "guess" it (see the typing rule T-Wrong below).

1 Values

Show the new cases of the grammar for values. I've provided one of them.

I would accept this grammar:

v	::=	...
		wrong: T
		nil[T]
		cons v_1 v_2

I would also accept this one:

v	::=	...
		wrong: T
		l
l	::=	nil[T]
		cons v l

2 Operational Semantics

Show the new rules of the operational semantics. I've provided the rules for head; you should add the other rules.

$$\frac{e_1 \longrightarrow e_1'}{\text{cons } e_1\ e_2 \longrightarrow \text{cons } e_1'\ e_2}\ \text{(E-Cons1)}$$

$$\frac{e_2 \longrightarrow e_2'}{\text{cons } v_1\ e_2 \longrightarrow \text{cons } v_1\ e_2'}\ \text{(E-Cons2)}$$

$$\frac{e \longrightarrow e'}{\text{head } e \longrightarrow \text{head } e'}\ \text{(E-Head)}$$

$$\frac{}{\text{head nil}[T] \longrightarrow \text{wrong} : T}\ \text{(E-HeadNil)}$$

$$\frac{}{\text{head } (\text{cons } v_1 v_2) \longrightarrow v_1}\ \text{(E-HeadCons)}$$

$$\frac{e \longrightarrow e'}{\text{tail } e \longrightarrow \text{tail } e'}\ \text{(E-Tail)}$$

$$\frac{}{\text{tail nil}[T] \longrightarrow \text{wrong} : T\ \text{List}}\ \text{(E-TailNil)}$$

$$\frac{}{\text{tail } (\text{cons } v_1 v_2) \longrightarrow v_2}\ \text{(E-TailCons)}$$

Technically, we also need the following rules, to handle reductions involving the "wrong" value:

$$\frac{}{\text{head } (\text{wrong} : T\ \text{List}) \longrightarrow \text{wrong} : T}\ \text{(E-HeadWrong)}$$

$$\frac{}{\text{tail } (\text{wrong} : T\ \text{List}) \longrightarrow \text{wrong} : T\ \text{List}}\ \text{(E-TailWrong)}$$

In addition, we should also add rules to handle the appearance of "wrong" in other parts of the syntax. For example, we should add an E-AppWrong rule to handle the case when the function value in a function call ends up being the "wrong" value. This is annoying, because it means we also have to revisit the cases of the proof involving function applications, to make sure this new rule is safe. A similar thing would be required for the if expression. I ignore these extra rules here for simplicity.

3 Typing Rules

Show the new typing rules. I've provided the rules for wrong and head; you should add the rules for nil, cons and tail.

$$\frac{}{\Gamma \vdash (\text{wrong} : T) : T}\ \text{(T-Wrong)}$$

$$\frac{}{\Gamma \vdash \text{nil}[T] : T\ \text{List}}\ \text{(T-Nil)}$$

$$\frac{\Gamma \vdash e_1 : T \quad \Gamma \vdash e_2 : T \text{ List}}{\Gamma \vdash \text{cons } e_1\ e_2 : T \text{ List}} \text{ (T-Cons)}$$

$$\frac{\Gamma \vdash e : T \text{ List}}{\Gamma \vdash \text{head } e : T} \text{ (T-Head)}$$

$$\frac{\Gamma \vdash e : T \text{ List}}{\Gamma \vdash \text{tail } e : T \text{ List}} \text{ (T-Tail)}$$

4 Type Soundness

a. State (no proof necessary) any new parts of the Canonical Forms lemma that are necessary in the proofs below.

Lemma (Canonical Forms): If $\Gamma \vdash v : T$ List, then v has the form nil$[T]$, (cons $v_1\ v_2$), or (wrong:T List).

b. Add the relevant cases to the proof of the Progress theorem. I've provided the cases for wrong and head; you should add the cases for nil, cons and tail.

Theorem (Progress): If $\vdash e : T$, then either e is a value or there exists e' such that $e \longrightarrow e'$.

- Case T-Wrong: Then $e =$ wrong: T, so e is a value.
- Case T-Nil: Then $e =$ nil$[T']$, so e is a value.
- Case T-Cons: Then $e =$ cons $e_1\ e_2$ and $T = T'$ List and $\vdash e_1 : T'$ and $\Gamma \vdash e_2 : T'$ List. By the inductive hypothesis, we have that either e_1 is a value or there exists e_1' such that $e_1 \longrightarrow e_1'$. Similarly, either e_2 is a value or there exists e_2' such that $e_2 \longrightarrow e_2'$. We perform a case analysis on these possibilities:
  - Case there exists e_1' such that $e_1 \longrightarrow e_1'$: Then by E-Cons1 we have $(\text{cons } e_1\ e_2) \longrightarrow (\text{cons } e_1'\ e_2)$.
  - Case e_1 is a value v_1: There are two sub-cases.
    - Case there exists e_2' such that $e_2 \longrightarrow e_2'$: Then by E-Cons2 we have (cons $v_1\ e_2$) $\longrightarrow$ (cons $v_1\ e_2'$).
    - Case e_2 is a value v_2: Then $e = (\text{cons } v_1\ v_2)$, so e is a value.
- Case T-Head: Then $e =$ (head e_1) and $\vdash e_1 : T$ List. By the inductive hypothesis, e_1 is either a value or there exists e_1' such that $e_1 \longrightarrow e_1'$. In the latter case, by E-Head we have (head e_1) $\longrightarrow$ (head e_1'). In the former case, by the Canonical Forms Lemma (assuming you wrote it properly) we have that e_1 is either nil$[T]$, (cons $v_1\ v_2$), or (wrong:T List). If e_1 is nil$[T]$, then by E-HeadNil we have $e \longrightarrow$ wrong : T. If e_1 has the form (cons $v_1\ v_2$), then by E-HeadCons we have $e \longrightarrow v_1$. If e_1 has the form (wrong:T List), then by E-HeadWrong we have $e \longrightarrow$ wrong : T.
- Case T-Tail: Then $e =$ (tail e_1) and $T = T'$ List and $\vdash e_1 : T'$ List. By the inductive hypothesis, e_1 is either a value or there exists e_1' such that $e_1 \longrightarrow e_1'$. In the latter case, by E-Tail we have (tail e_1) $\longrightarrow$ (tail e_1'). In the former case, by the Canonical Forms Lemma we have that e_1 is either nil$[T']$, (cons $v_1\ v_2$), or (wrong:T' List). If e_1 is nil$[T']$, then by E-TailNil we have $e \longrightarrow$ wrong : T' List. If e_1 has the form (cons $v_1\ v_2$), then by E-TailCons we have $e \longrightarrow v_2$. If e_1 has the form (wrong:T' List), then by E-HeadWrong we have $e \longrightarrow$ wrong : T' List.

c. Add the relevant cases to the proof of the Type Preservation theorem. I've provided the cases for wrong and head; you should add the cases for nil, cons and tail.

Theorem (Type Preservation): If $\Gamma \vdash e : T$ and $e \longrightarrow e'$, then $\Gamma \vdash e' : T$.

- Case T-Wrong: Then $e = \text{wrong}: T$. By inspection, there is no e' such that $\text{wrong}: T \longrightarrow e'$ (assuming you didn't add such a rule), so this case is satisfied trivially.
- Case T-Nil: Then $e = \text{nil}[T']$. By inspection, there is no e' such that $\text{nil}[T'] \longrightarrow e'$, so this case is satisfied trivially.
- Case T-Cons: Then $e = \text{cons}\ e_1\ e_2$ and $T = T'\ \text{List}$ and $\Gamma \vdash e_1 : T'$ and $\Gamma \vdash e_2 : T'\ \text{List}$. We're given that $e \longrightarrow e'$. Case analysis of the last rule used in the derivation of this reduction step:
  - Case E-Cons1: Then $e' = \text{cons}\ e_1'\ e_2$ and $e_1 \longrightarrow e_1'$. By the inductive hypothesis we have that $\Gamma \vdash e_1' : T'$. Therefore, by T-Cons we have $\Gamma \vdash \text{cons}\ e_1'\ e_2 : T$.
  - Case E-Cons2: Then $e' = \text{cons}\ e_1\ e_2'$ and $e_2 \longrightarrow e_2'$. By the inductive hypothesis we have that $\Gamma \vdash e_2' : T'\ \text{List}$. Therefore, by T-Cons we have $\Gamma \vdash \text{cons}\ e_1\ e_2' : T$.
- Case T-Head: Then $e = (\text{head}\ e_1)$ and $\Gamma \vdash e_1 : T\ \text{List}$. We're given that $e \longrightarrow e'$. Case analysis of the last rule used in the derivation of this reduction step:
  - Case E-Head: Then $e' = \text{head}\ e_1'$ and $e_1 \longrightarrow e_1'$. By the inductive hypothesis we have that $\Gamma \vdash e_1' : T\ \text{List}$. Therefore, by T-Head we have $\Gamma \vdash e' : T$.
  - Case E-HeadNil: Then $e_1 = \text{nil}[T']$ and $e' = \text{wrong}: T'$. Since $\Gamma \vdash e_1 : T\ \text{List}$, by T-Nil (assuming you wrote it properly) we have that $T' = T$. Then by T-Wrong we have $\Gamma \vdash (\text{wrong}: T) : T$.
  - Case E-HeadCons: Then $e_1 = (\text{cons}\ v_1 v_2)$ and $e' = v_1$. Since $\Gamma \vdash e_1 : T\ \text{List}$, by T-Cons (assuming you wrote it properly) we have that $\Gamma \vdash v_1 : T$.
  - Case E-HeadWrong: Then $e_1 = \text{wrong}: T'\ \text{List}$ and $e' = \text{wrong}: T'$. Since $\Gamma \vdash e_1 : T\ \text{List}$, by T-Wrong we have $T' = T$, so again by T-Wrong we have $\Gamma \vdash (\text{wrong}: T) : T$.
- Case T-Tail: Then $e = (\text{tail}\ e_1)$ and $T = T'\ \text{List}$ and $\Gamma \vdash e_1 : T'\ \text{List}$. We're given that $e \longrightarrow e'$. Case analysis of the last rule used in the derivation of this reduction step:
  - Case E-Tail: Then $e' = \text{tail}\ e_1'$ and $e_1 \longrightarrow e_1'$. By the inductive hypothesis we have that $\Gamma \vdash e_1' : T'\ \text{List}$. Therefore, by T-Tail we have $\Gamma \vdash e' : T$.
  - Case E-TailNil: Then $e_1 = \text{nil}[T'']$ and $e' = \text{wrong}: T''\ \text{List}$. Since $\Gamma \vdash e_1 : T'\ \text{List}$, by T-Nil we have that $T'' = T'$. Then by T-Wrong we have $\Gamma \vdash (\text{wrong}: T'\ \text{List}) : T'\ \text{List}$.
  - Case E-TailCons: Then $e_1 = (\text{cons}\ v_1 v_2)$ and $e' = v_2$. Since $\Gamma \vdash e_1 : T'\ \text{List}$, by T-Cons we have that $\Gamma \vdash v_2 : T'\ \text{List}$.
  - Case E-TailWrong: Then $e_1 = \text{wrong}: T''\ \text{List}$ and $e' = \text{wrong}: T''\ \text{List}$. Since $\Gamma \vdash e_1 : T'\ \text{List}$, by T-Wrong we have that $T'' = T'$. Then by T-Wrong we have $\Gamma \vdash (\text{wrong}: T'\ \text{List}) : T'\ \text{List}$.