IMP in HOLCF

Tobias Nipkow and Robert Sandner

March 13, 2025

Contents

1 Denotational Semantics of Commands in HOLCF

```
theory Denotational imports HOLCF "HOL-IMP.Big_Step" begin
```

1.1 Definition

```
definition
  dlift :: "(('a::type) discr -> 'b::pcpo) => ('a lift -> 'b)" where
  "dlift f = (LAM x. case x of UU ⇒ UU | Def y ⇒ f·(Discr y))"

primrec D :: "com ⇒ state discr → state lift"
where
  "D(SKIP) = (LAM s. Def(undiscr s))"
| "D(X ::= a) = (LAM s. Def((undiscr s)(X := aval a (undiscr s))))"
| "D(c0 ;; c1) = (dlift(D c1) oo (D c0))"
| "D(IF b THEN c1 ELSE c2) =
        (LAM s. if bval b (undiscr s) then (D c1)·s else (D c2)·s)"
| "D(WHILE b DO c) =
        fix·(LAM w s. if bval b (undiscr s) then (dlift w)·((D c)·s)
                      else Def(undiscr s))"
```

1.2 Equivalence of Denotational Semantics in HOLCF and Evaluation Semantics in HOL

```
lemma dlift_Def [simp]: "dlift f·(Def x) = f·(Discr x)"
  by (simp add: dlift_def)
```

```
lemma cont_dlift [iff]: "cont (%f. dlift f)"
  by (simp add: dlift_def)

lemma dlift_is_Def [simp]:
    "(dlift f·l = Def y) = (∃x. l = Def x ∧ f·(Discr x) = Def y)"
  by (simp add: dlift_def split: lift.split)

lemma eval_implies_D: "(c,s) ⇒ t ⟹ D c·(Discr s) = (Def t)"
apply (induct rule: big_step_induct)
      apply (auto)
 apply (subst fix_eq)
 apply simp
apply (subst fix_eq)
apply simp
done

lemma D_implies_eval: "∀s t. D c·(Discr s) = (Def t) ⟶ (c,s) ⇒ t"
apply (induct c)
    apply fastforce
   apply fastforce
  apply force
 apply (simp (no_asm))
 apply force
apply (simp (no_asm))
apply (rule fix_ind)
  apply (fast intro!: adm_lemmas adm_chfindom ax_flat)
 apply (simp (no_asm))
apply (simp (no_asm))
apply force
done

theorem D_is_eval: "(D c·(Discr s) = (Def t)) = ((c,s) ⇒ t)"
by (fast elim!: D_implies_eval [rule_format] eval_implies_D)

end
```

2 Correctness of Hoare by Fixpoint Reasoning

```
theory HoareEx imports Denotational begin
```

An example from the HOLCF paper by Müller, Nipkow, Oheimb, Slotosch [1]. It demonstrates fixpoint reasoning by showing the correctness of the Hoare rule for while-loops.

```
type_synonym assn = "state ⇒ bool"

definition
  hoare_valid :: "[assn, com, assn] ⇒ bool"  (‹|= {(1_)}/ (_)/ {(1_)}› 50) where
  "|= {P} c {Q} = (∀s t. P s ∧ D c·(Discr s) = Def t ⟶ Q t)"

lemma WHILE_rule_sound:
```

```
    "|= {A} c {A} ⟹ |= {A} WHILE b DO c {λs. A s ∧ ¬ bval b s}"
  apply (unfold hoare_valid_def)
  apply (simp (no_asm))
  apply (rule fix_ind)
    apply (simp (no_asm)) — simplifier with enhanced adm-tactic
   apply (simp (no_asm))
  apply (simp (no_asm))
  apply blast
  done

end
```

References

[1] O. Müller, T. Nipkow, D. v. Oheimb, and O. Slotosch. HOLCF = HOL + LCF. *J. Functional Programming*, 9:191–223, 1999.