Technische Universität München
Fakultät für Informatik
Prof. Tobias Nipkow, Ph.D.
Fabian Huch

WS 2020/21
21.12.2020

Semantics of Programming Languages

Exercise Sheet 08

Exercise 8.1 Knaster-Tarski Fixed Point Theorem

The Knaster-Tarski theorem tells us that for each set P of fixed points of a monotone function f we have a fixpoint of f which is a greatest lower bound of P. In this exercise, we want to prove the Knaster-Tarski theorem.

First we give a construction of the greatest lower bound of all fixed points P of the function f. This is the union of all sets u smaller than P and $f\ u$. Then the task is to show that this is a fixed point, and that it is the greatest lower bound of all sets in P.

Let us define *Inf_fixp*:

definition *Inf_fixp* :: "$('a\ set \Rightarrow 'a\ set) \Rightarrow 'a\ set\ set \Rightarrow 'a\ set$" **where**
"$Inf\_fixp\ f\ P = \bigcup \{u.\ u \subseteq \bigcap P \cap f\ u\ \}$"

To work directly with this definition is a little cumbersome, we propose to use the following two theorems:

lemma *Inf_fixp_upperbound*: "$X \subseteq \bigcap P \Longrightarrow X \subseteq f\ X \Longrightarrow X \subseteq Inf\_fixp\ f\ P$"
by (*auto simp*: *Inf_fixp_def*)

lemma *Inf_fixp_least*: "$(\bigwedge u.\ u \subseteq \bigcap P \Longrightarrow u \subseteq f\ u \Longrightarrow u \subseteq X) \Longrightarrow Inf\_fixp\ f\ P \subseteq X$"
by (*auto simp*: *Inf_fixp_def*)

Now prove, that *Inf_fixp* is acually a fixed point of *f*.

Hint: First prove $Inf\_fixp\ f\ P \subseteq f\ (Inf\_fixp\ f\ P)$, this will be used for the other direction. It may be helpful to first think about the structure of your proof using pen-and-paper and then translate it into Isar.

lemma *Inf_fixp*:
assumes f: "*mono f*"
assumes P: "$\bigwedge p.\ p \in P \Longrightarrow f\ p = p$"
shows "$Inf\_fixp\ f\ P = f\ (Inf\_fixp\ f\ P)$"

Now we prove that it is a lower bound:

lemma *Inf_fixp_lower*: "$Inf\_fixp\ f\ P \subseteq \bigcap P$"

And that it is the greatest lower bound:

lemma *Inf_fixp_greatest*:
assumes "$f\ q = q$" "$q \subseteq \bigcap P$" **shows** "$q \subseteq Inf\_fixp\ f\ P$"

Exercise 8.2 Denotational Semantics

Define a denotational semantics for REPEAT-loops, and show its equivalence to the bigstep semantics.

datatype *com* = *SKIP*
    | *Assign vname aexp* (*"_ ::= _" [1000, 61] 61*)
    | *Seq com com* (*"_;;/ _" [60, 61] 60*)
    | *If bexp com com* (*"(IF _/ THEN _/ ELSE _)" [0, 0, 61] 61*)
    | *While bexp com* (*"(WHILE _/ DO _)" [0, 61] 61*)
    | *Repeat com bexp* (*"(REPEAT _/ UNTIL _)" [0, 61] 61*)

inductive
  big_step :: *"com × state ⇒ state ⇒ bool"* (**infix** *"⇒" 55*)
where
Skip: *"(SKIP,s) ⇒ s"* |
Assign: *"(x ::= a,s) ⇒ s(x := aval a s)"* |
Seq: "⟦ $(c_1,s_1) \Rightarrow s_2$; $(c_2,s_2) \Rightarrow s_3$ ⟧ ⟹ $(c_1;;c_2,\ s_1) \Rightarrow s_3$" |
IfTrue: "⟦ *bval b s*; $(c_1,s) \Rightarrow t$ ⟧ ⟹ (*IF b THEN* c_1 *ELSE* c_2, *s*) ⇒ *t*" |
IfFalse: "⟦ ¬*bval b s*; $(c_2,s) \Rightarrow t$ ⟧ ⟹ (*IF b THEN* c_1 *ELSE* c_2, *s*) ⇒ *t*" |
WhileFalse: *"¬bval b s ⟹ (WHILE b DO c,s) ⇒ s"* |
WhileTrue:
  "⟦ *bval b* s_1; $(c,s_1) \Rightarrow s_2$; (*WHILE b DO c*, s_2) ⇒ s_3 ⟧
  ⟹ (*WHILE b DO c*, s_1) ⇒ s_3"

Proof automation:

lemmas [*intro*] = *big_step.intros*
lemmas *big_step_induct* = *big_step.induct[split_format(complete)]*

inductive_cases *SkipE[elim!]*: *"(SKIP,s) ⇒ t"*
inductive_cases *AssignE[elim!]*: *"(x ::= a,s) ⇒ t"*
inductive_cases *SeqE[elim!]*: *"(c1;;c2,s1) ⇒ s3"*
inductive_cases *IfE[elim!]*: *"(IF b THEN c1 ELSE c2,s) ⇒ t"*
inductive_cases *WhileE[elim]*: *"(WHILE b DO c,s) ⇒ t"*

Execution is deterministic:

theorem *big_step_determ*: *"⟦ (c,s) ⇒ t; (c,s) ⇒ u ⟧ ⟹ u = t"*
  by (*induction arbitrary: u rule: big_step.induct*) *blast+*

type_synonym *com_den* = *"(state × state) set"*

definition *W* :: *"(state ⇒ bool) ⇒ com_den ⇒ (com_den ⇒ com_den)"* **where**
"W db dc = (λdw. {(s,t). if db s then (s,t) ∈ dc O dw else s=t})"

fun *D* :: *"com ⇒ com_den"* **where**
"D SKIP = Id" |
"D (x ::= a) = {(s,t). t = s(x := aval a s)}" |
"D (c1;;c2) = D(c1) O D(c2)" |
"D (IF b THEN c1 ELSE c2)

= {(s,t). *if bval b s then* (s,t) ∈ *D c1 else* (s,t) ∈ *D c2*}" |
"*D* (*WHILE b DO c*) = *lfp* (*W* (*bval b*) (*D c*))"
lemma *W_mono*: "*mono* (*W b r*)"
by (*unfold W_def mono_def*) *auto*

lemma *R_mono*: "*mono* (*R b r*)"
by (*unfold R_def mono_def*) *auto*

lemma *D_While_If*:
"*D*(*WHILE b DO c*) = *D*(*IF b THEN c*;;*WHILE b DO c ELSE SKIP*)"
proof−
let *?w* = "*WHILE b DO c*" **let** *?f* = "*W* (*bval b*) (*D c*)"
have "*D ?w* = *lfp ?f*" **by** *simp*
also have "... = *?f* (*lfp ?f*)" **by**(*rule lfp_unfold* [*OF W_mono*])
also have "... = *D*(*IF b THEN c*;;*?w ELSE SKIP*)" **by** (*simp add*: *W_def*)
finally show *?thesis* **.**
qed

Equivalence of denotational and big-step semantics:

lemma *D_if_big_step*: "(c,s) ⇒ t ⟹ (s,t) ∈ D(c)"
proof (*induction rule*: *big_step_induct*)
case *WhileFalse*
with *D_While_If* **show** *?case* **by** *auto*
next
case *WhileTrue*
show *?case* **unfolding** *D_While_If* **using** *WhileTrue* **by** *auto*
nextqed *auto*

abbreviation *Big_step* :: "*com* ⇒ *com_den*" **where**
"*Big_step c* ≡ {(s,t). (c,s) ⇒ t}"

lemma *Big_step_if_D*: "(s,t) ∈ D(c) ⟹ (s,t) : *Big_step c*"
proof (*induction c arbitrary*: *s t*)
case *Seq* **thus** *?case* **by** *fastforce*
next
case (*While b c*)
let *?B* = "*Big_step* (*WHILE b DO c*)" **let** *?f* = "*W* (*bval b*) (*D c*)"
have "*?f ?B* ⊆ *?B*" **using** *While.IH* **by** (*auto simp*: *W_def*)
from *lfp_lowerbound*[**where** *?f* = "*?f*", *OF this*] *While.prems*
show *?case* **by** *auto*
nextqed (*auto split*: *if_splits*)

theorem *denotational_is_big_step*:
"(s,t) ∈ D(c) = ((c,s) ⇒ t)"
by (*metis D_if_big_step Big_step_if_D*[*simplified*])

Homework 8.1 Be Original!

Submission until Sunday, Jan 10, 23:59. In total, this exercise is worth 15 points, plus bonus points for nice submissions.

You should now have a topic to formalize, for example:

- Prove some interesting result about algorithms/graphs/automata/formal language theory
- Formalize some results from mathematics
- Find interesting modifications of IMP material and prove interesting properties about them
- ...

Do the formalization! You can submit your work via the submission system or by email.

You should set yourself a time limit before starting your project. Also incomplete/unfinished formalizations are welcome and will be graded!

Please comment your formalization well, such that we can see what it does/is intended to do.

Merry Christmas!