ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Dipl.-Inf. Achim D. Brucker
Dr. Burkhart Wolff

Computer-supported Modeling and Reasoning

http://www.infsec.ethz.ch/education/permanent/csmr/

(rev. 16814)

Submission date: –

HOL: Inductive Data Types

In this exercise, we will study the concept of the least fix-point operator lfp, its main theorems knaster_tarski and lfp_induct and its major application: providing semantics for inductive definitions. lfp

The importance of the concept of inductive definition will be revealed by applying it in three examples, ranging from closures, finite sets to natural numbers.

1 More on Isabelle/HOL

1.1 Inductive Definitions

The general syntactic scheme of an inductive definition is: **inductive**

```
inductive "expr"
  intros
  thmname_1: "H_1 ∈ expr"
  ...
  thmname_m: "⟦ Cond_1(expr); ...; Cond_n(expr)⟧ ⟹ H_m ∈ expr"
```

where expr must be a set of the form C var_1 ...var_k and where C is a previously declared, but not yet defined constant, and the list of variables var_i may be empty. After the keyword **intros**, introduction rules for the inductive set may be inserted, either with assumptions or not (both forms can be arbitrarily mixed). The conditions Cond_i may depend on expr or not.

Isabelle will process such statements and compile it to

C.defs

1. a constant definition for C which can be referenced by C.**defs**

2. proofs for the introduction rules in the form given in the inductive statement; the theorems can be referenced by their given name thmname_i, and

C.induct

3. proofs for the induction rules which can be referenced by C.induct

declare

Note that introducing theorems via the **declare** statement (see the ISAR Reference Manual[1]) allows to insert such rules once and for all into the appropriate "slots" of the proof engine; there are more syntactic variants in the inductive statement that have the same effect.

1.2 Constant Specifications

constant specification

There is an alternative conservative extension scheme supported by Isabelle, namely the *constant specification*. In contrast to the constant definition used so far, a "fresh" constant c may be specified by a syntacticly unlimited predicate P in an axiom $P\,x$. Of course, this axiom must be justified by the proof of the semantic side-condition $\exists x. P\,x$.

specification

The overall syntactic scheme of a constant specification in the ISAR language is:

```
specification (C)
  thmname: "P C"
  ...
  done
```

where C is a previously declared, but not yet defined constant, P a characterizing predicate that can be referenced by thmname, followed by a proof for the side-condition.

2 Exercises

2.1 Exercise 36

Prove the Knaster-Tarski theorem

$$mono\ f \Longrightarrow lfp\ f = f(lfp\ f)$$

[1] http://isabelle.in.tum.de/dist/Isabelle2004/doc/isar-ref.pdf

using the presentation given in the lecture "HOL: Fixpoints", i.e., first prove the claims 1–4. Use whatever proof methods you like, but you should no use any theorem from the HOL library.

2.2 Exercise 37

1. Define inductively the function "Fin:: 'a set ⇒ 'a set set" that produces the set of all finite subsets.

2. Prove the following properties over set of all finite subsets:

   a) **lemma** "{1,2}∈Fin{1,2,3}"

   b) **lemma** "⟦a∈Fin A; b∈Fin A⟧⟹(a∪b) ∈ Fin A"

   c) **lemma** "⟦(A ∈Fin X) ∨ (A ∈ Fin Y)⟧ ⟹A ∈ Fin (X ∪Y)"

   d) **lemma** finite_Inl : "⟦ b∈ Fin A⟧ ⟹ (a∩b) ∈ Fin A"

   e) **lemma** "⟦A ∈Fin X⟧⟹Pow(A)∈ Pow(Fin X)"

Remark: The elements 1, 2, etc. do not imply that we have already numbers; they are constants in syntactic classes predefined in the library. As a result, Fin{1,2,3} has the type ('a::{one,zero,number})set and not nat set.

2.3 Exercise 38

1. Define the concept of a reflexive transitive closure as an inductive definition over the constant

   consts
     rtc :: "('a × 'a) set ⇒ ('a × 'a) set" ("(_^**)" [1000] 999)

2. Prove the following properties, using the derived induction scheme (The last two are optional.):

   a) **lemma** rtc: "⋀p. p ∈ r ⇒ p ∈ r^**"

   b) **lemma** rtc_induct_pointwise:
      assumes a: "(a:: 'a, b) ∈ r^**"
      assumes base: "P a"
      assumes step: "⋀ y z. ⟦(a, y) ∈ r^**; (y, z) ∈ r; P y⟧ ⟹P z"
      shows "P b"

   c) **lemma** ctr_trans: "⟦ (a,b)∈ r^**;(b,c)∈ r^** ⟧ ⟹ (a,c)∈ r^**"

d) **lemma** rtc_is_closure: "(r^**)^** = r^**"

e) **lemma** rtc_un_distr: "(R^** ∪S^**)^** = (R ∪S)^**"

f) **lemma** rtc_un_distr: "R^** O R^** = R^**"

Hints:

1. Prove the lemmas in the given order.
2. You may unfold variables denoting pairs with the method: **apply**(simp only: split_tupled_all)
3. The crucial alternative induction scheme needs an additional assumption a = a ⟶P(b). You should add this assumption (using subgoal_tac) and prove it using the derived induction scheme with the instance P = λx y. x = a ⟶P y.

2.4 Exercise 39

State the axiom of infinity

axioms infinity : "∃ f::ind ⇒ ind. inj f ∧ ¬ surj f"

and build a conservative theory extension deriving the core of the natural number theory, the Peano Axioms:

1. Declare the constants ZERO::ind and SUC::ind ⇒ind,
2. Use a constant specifications to specify ZERO and SUC appropriately, i.e., such that you can derive ZERO ≠SUC X and SUC X = SUC Y ⟹X = Y,
3. Define NAT as the inductive set built over ZERO and SUC
4. Show the "induction" theorem on NAT.