Illustrations of (Co-)Inductive Definitions

Roger Bishop Jones

Date: 2012-08-11 21:01:53

Abstract

This document provides examples of the use of the facilities provided in t007.doc.
## Contents

1 INTRODUCTION 3
   1.1 Formalities .......................................................... 3

2 FIXED POINTS 3

3 INDUCTIVE DEFINITIONS OF SETS 3
   3.1 Hereditarily Over a Function .................................... 3
   3.2 Hereditarily Over a Relation .................................... 5
   3.3 Hereditarily Over a Property .................................... 5

4 INDUCTIVE DEFINITIONS OF SETS 5
   4.1 Sets Defined Using CCP’s .......................................... 5
      4.1.1 Hereditarily Pure Functions ................................ 5
      4.1.2 Hereditarily Pure Functors and Categories .............. 5

5 CODING CONSTRUCTIONS 5
   5.1 HOL Types and Terms .............................................. 5

6 MAKING NEW TYPES 8

7 The Theory fixp-egs 9
   7.1 Parents .............................................................. 9
   7.2 Constants .......................................................... 9
   7.3 Aliases .............................................................. 9
   7.4 Definitions ......................................................... 10
   7.5 Theorems ............................................................ 10

8 INDEX 11
1 INTRODUCTION

The document follows, as far as is reasonable, the structure of [1] in providing material illustrating (and testing) the facilities for (co-)inductive definition of sets and types provided in that document.

1.1 Formalities

Create new theory “fixp_egs”.

SML

|open_theory "fixp";
|force_new_theory "fixp_egs";
|set_merge pcs["hol", "savedthm_cs_∃_proof"];

2 FIXED POINTS

I probably won’t supply any direct examples of this material. It is used indirectly in all that follows.

3 INDUCTIVE DEFINITIONS OF SETS

3.1 Hereditarily Over a Function

The example I had in mind which made me think this kind of definition useful is the inductive definition of the theorems of some deductive system. This belongs here because taking the rules of a deductive system as constructors gives an inductive definition in which the constructor is not one-one. It also raises a question about inductive definition of functions over the type, since a naive interpretation of the ‘content’ relation would not be well-founded.

Another example would be the definition of recursive function.

These things are easy to do without any special machinery so it might be interesting to try one which has already been done that way, for example my definition of a first order property in [2].

SML

|new_parent "membership";

The inductive definition is that of fof. How would that look if formulated as the fixed point of a function?
HOL Constant

\[ \text{FofCSet: } ((\text{STRING}, 'a) \text{ PPMS SET} \times (\text{STRING}, 'a) \text{ PPMS SET}) \]

\[ \text{FofCSet} = \{ (\text{ppmss}, \text{ppms1}) | \]
\[ \text{ppmss} = \{ \} \land (\exists s1 s2 \bullet \text{ppms1} = \text{s1} =_p s2 \lor \text{ppms1} = \text{s1} \in_p s2) \]
\[ \lor (\exists p \bullet \text{ppmss} = \{ p \} \land \text{ppms1} = (\exists p \ s \ p) \lor \text{ppms1} = \lnot_p \ p) \]
\[ \lor (\exists p1 p2 \bullet \text{ppmss} = \{ p1; p2 \} \land \text{ppms1} = p1 \land_p p2) \}

This is turned into a set of formulae thus:

HOL Constant

\[ \text{Fof: } (\text{STRING}, 'a) \text{ PPMS SET} \]

\[ \text{Fof} = \text{HeredFun} (\text{Rules2Fun} \ \text{FofCSet}) \]

From this we should be able to get an induction principle by instantiating and expanding \text{HeredFun\_induction\_thm2}.

However, this looks quite hard to massage into a nicely formulated induction property.

By comparison coding up the required closure property directly gives:

HOL Constant

\[ \text{FofCProp: } (\text{STRING}, 'a) \text{ PPMS SET} \rightarrow \text{BOOL} \]

\[ \forall \text{ppmss} \bullet \text{FofCProp ppmss} \Leftrightarrow \]
\[ (\forall s1 s2 \bullet s1 =_p s2 \in \text{ppmss} \land s1 \in_p s2 \in \text{ppmss}) \]
\[ \land (\forall p \bullet p \in \text{ppmss} \Rightarrow \lnot_p p \in \text{ppmss} \land \forall s \bullet (\exists p s \ p) \in \text{ppmss}) \]
\[ \land (\forall p1 p2 \bullet p1 \in \text{ppmss} \land p2 \in \text{ppmss} \Rightarrow p1 \land_p p2 \in \text{ppmss}) \]

HOL Constant

\[ \text{Fof2: } (\text{STRING}, 'a) \text{ PPMS SET} \]

\[ \text{Fof2} = \bigcap \{ s | \text{FofCProp s} \} \]

\[ \text{fof\_induction\_thm} \vdash \forall s \bullet \text{FofCProp s} \Rightarrow \text{Fof2} \subseteq s \]

\[ \text{fof\_induction\_thm2} \vdash \forall s \]
\[ \bullet (\forall s1 s2 \bullet s1 =_p s2 \in s \land s1 \in_p s2 \in s) \]
\[ \land (\forall p \bullet p \in s \Rightarrow \lnot_p p \in s \land (\forall s' \bullet (\exists p s' p) \in s)) \]
\[ \land (\forall p1 p2 \bullet p1 \in s \land p2 \in s \Rightarrow p1 \land_p p2 \in s) \]
\[ \Rightarrow \text{Fof2} \subseteq s \]
3.2 Hereditarily Over a Relation

3.3 Hereditarily Over a Property

4 INDUCTIVE DEFINITIONS OF SETS

The examples here are examples which do not use the machinery for generating constructor functions.

4.1 Sets Defined Using CCP’s

4.1.1 Hereditarily Pure Functions

4.1.2 Hereditarily Pure Functors and Categories

5 CODING CONSTRUCTIONS

5.1 HOL Types and Terms

Here is a simple type description to work from, the types and terms of HOL: Note that for each constructor is supplied the type of the constructor and a predicate over the domain of the constructor. In this example the predicates enforce constraints on the names of constants and variables requiring that names beginning with ‘ are variable names.

The type variables ‘TYPE and ‘TERM give the names of the sets or types which are being defined.

```
val constructor_types = [
  (⌜MkVarType: STRING → TYPE⌝, ⌜λx:STRING• Hd x = x⌝),
  (⌜MkCType: STRING → TYPE LIST → TYPE⌝, ⌜λ(x:STRING, y:TYPE LIST)• ¬ Hd x = x⌝),
  (⌜MkVarTerm: STRING → TYPE → TERM⌝, ⌜λ(x:STRING, y:TYPE)• Hd x = x⌝),
  (⌜MkCTerm: STRING → TYPE → TERM⌝, ⌜λ(x:STRING, y:TYPE)• ¬ Hd x = x⌝),
  (⌜MkLamTerm: STRING → TYPE → TERM → TERM⌝, ⌜λ(x:STRING, y:TYPE, z:TERM)• Hd x = x⌝),
  (⌜MkAppTerm: TERM → TERM → TERM⌝, ⌜λ(x:TERM, y:TERM)• T⌝)
];
```

This system of type specifications can be translated into an expression for a fixedpoint as follows:

```
set_flag ("pp_use_alias", true);
declare_ctk_aliases ntree_ctk ctk_aliases;
val fixp_expression = translate_sig ntree_ctk constructor_types;
undeclarer_ctk_aliases ntree_ctk ctk_aliases;
fixp_expression;
```
Which yields:

\[
\text{val fixp_expression =}
\begin{array}{c}
\Gamma \Theta \\
[[\Xi (\phi \gamma) (\phi \rho) (\lambda x \bullet \text{Head } x = "m") ; \\
\Xi (\phi \gamma \times \phi I) (\phi \rho \times (\omega I)) (\lambda (x, y) \bullet \neg \text{Head } x = "m") ] ; \\
\Xi (\phi \gamma \times I) (\phi \rho \times (\omega I)) (\lambda (x, y) \bullet \text{Head } x = "m") ; \\
\Xi (I \times I) (\omega \times (\omega I)) (\lambda (x, y) \bullet T) ] ] ] \triangleright: \text{TERM}
\end{array}
\]

Without the aliases that is:

\[
\text{val fixp_expression =}
\begin{array}{c}
\Gamma FR \\
[[CR (NTreeeMkList MkLeafTtreee) (NTrListC NTrLeafC) (\lambda x \bullet \text{Head } x = "m") ; \\
CR (NTreeeMkProd (NTreeeMkList MkLeafTtreee) (NTreeeMkList I)) (NTrProdC (NTrListC NTrLeafC) (NTrListC (NTreeeTagC 1))) (\lambda (x, y) \bullet \neg \text{Head } x = "m") ] ; \\
CR (NTreeeMkProd (NTreeeMkList MkLeafTtreee) I) (NTrProdC (NTrListC NTrLeafC) (NTrTagC 2)) (\lambda (x, y) \bullet \text{Head } x = "m") ; \\
CR (NTreeeMkProd (NTreeeMkList MkLeafTtreee) (NTreeeMkProd I I)) (NTrProdC (NTrListC NTrLeafC) (NTreeeTagC 1)) (\lambda (x, y, z) \bullet \text{Head } x = "m") ; \\
CR (NTreeeMkProd I I) (NTrProdC (NTreeeTagC 2) (NTreeeTagC 2)) (\lambda (x, y) \bullet T) ] ] \triangleright: \text{TERM}
\end{array}
\]

which has type:

\textbf{SML}

\textbf{type_of fixp_expression;}

\textbf{val it = }\Gamma:(\mathbb{N}, \text{CHAR}) \text{ TREEE SET}\triangleright: \text{TYPE}
This set contains the representatives of each type discriminated by tags, so to recover the two sets we need to filter the one as follows:

```sml
val type_rep = \{ x | \exists y \in fixp_expression \land IsTag 1 y \land x = UnTag y \}\;
```

```sml
val term_rep = \
\{ x \mid \exists y \in \Theta \; \bullet y \in \Theta \; \\
[[\exists (\phi \rho) (\lambda x \bullet Head x =\,''m)]]; \\
\exists (\phi \rho \times (\phi (\omega 1))) (\lambda (x, y) \bullet Head x =\,''m)]; \\
[[\exists (\phi \rho \times 1) (\phi \rho \times \omega 1) (\lambda (x, y) \bullet Head x =\,''m)]; \\
\exists (\phi \rho \times 1) (\phi \rho \times 1) (\lambda (x, y) \bullet Head x =\,''m)]; \\
\exists (\phi \rho \times 1) (\phi \rho \times 1) (\lambda (x, y) \bullet Head x =\,''m)]; \\
\exists (\phi \rho \times I \times I) \\
(\phi \rho \times \omega 1 \times \omega 2) \\
(\lambda (x, y, z) \bullet Head x =\,''m)]; \\
\exists (I \times I) (\omega 2 \times \omega 2) (\lambda (x, y) \bullet T)]] \\
\land IsTag 2 y \\
\land x = UnTag y\}; \text{T\text{ERM}}
```

```sml
val term_rep = \{ x | \exists y \in fixp_expression \land IsTag 2 y \land x = UnTag y \}\;
```

```sml
val term_rep = \
\{ x \mid \exists y \in \Theta \; \bullet y \in \Theta \; \\
[[\exists (\phi \rho) (\lambda x \bullet Head x =\,''m)]]; \\
\exists (\phi \rho \times (\phi (\omega 1))) (\lambda (x, y) \bullet Head x =\,''m)]; \\
[[\exists (\phi \rho \times 1) (\phi \rho \times \omega 1) (\lambda (x, y) \bullet Head x =\,''m)]; \\
\exists (\phi \rho \times 1) (\phi \rho \times 1) (\lambda (x, y) \bullet Head x =\,''m)]; \\
\exists (\phi \rho \times I \times I) \\
(\phi \rho \times \omega 1 \times \omega 2) \\
(\lambda (x, y, z) \bullet Head x =\,''m)]; \\
\exists (I \times I) (\omega 2 \times \omega 2) (\lambda (x, y) \bullet T)]] \\
\land IsTag 2 y \\
\land x = UnTag y\}; \text{T\text{ERM}}
```

Let’s check this out by trying a proof that the set of types is non-empty.

```sml
set_goal([], \gamma t \in \gamma \text{type_rep} \gamma); \
a (rewrite_tac (map get_spec [\gamma \Theta, \gamma \exists, \gamma NTrProdC, \gamma NTreeeMkProd]));
```
6 MAKING NEW TYPES

Here are some examples I would like to be able to handle.

algebras e.g. a type of groups, these are essentially subtypes of labelled product types in which the projections yield objects conforming to the signature of the relevant algebra. This is therefore a special case of the general inductive datatypes problem (if an implementation of that allowed predicates).

types from algebras e.g. given a type of groups, make a type out of a presentation of a particular group.

from hereditarily sets e.g. a type of hereditarily pure functions, or a pair of types for the pure concrete categories and functors.
7 The Theory fixp-egs

7.1 Parents

\[ \text{membership } \text{fixp} \]

7.2 Constants

\[
\begin{align*}
FofCSet & \quad (\text{STRING}, 'a) \text{ PPMS } P \leftrightarrow (\text{STRING}, 'a) \text{ PPMS} \\
Fof & \quad (\text{STRING}, 'a) \text{ PPMS } P \\
FofCProp & \quad (\text{BOOL}, (\text{STRING}, 'a) \text{ PPMS } P) \text{ VA} \\
Fof_2 & \quad (\text{STRING}, 'a) \text{ PPMS } P
\end{align*}
\]

7.3 Aliases

\[
\begin{align*}
\nu & \quad \text{NTreeeTag} \\
\omega & \quad \text{NTreeeTagC} \\
\gamma & \quad \text{MkLeafTreee} : (('a, 'b) \text{ TREEE}, 'b) \text{ VA} \\
\times & \quad \text{NTreeeMkProd} \\
\phi & \quad \text{NTreeeMkSum} \\
\rho & \quad \text{NTreeeMkList} \\
\phi & \quad \text{NTyProdC} \\
\phi & \quad \text{NTySumC} \\
\phi & \quad \text{NTyListC} \\
\rho & \quad \text{NTyLeafC} : ((\text{N}, 'b) \text{ TREEE}, 'a) \text{ VA}
\end{align*}
\]

\[
\begin{align*}
\Xi & \quad : (((((\text{BOOL}, 'b) \text{ VA}, 'b) \text{ VA}, (\text{BOOL}, 'a) \text{ VA}) \text{ VA}, ('b, 'a) \text{ VA}) \text{ VA}, ('b, 'a) \text{ VA}) \text{ VA}
\end{align*}
\]
7.4 Definitions

\[ FofCSet \vdash FofCSet = \{(ppmss, ppms1) \mid \begin{align*}
ppmss &= \{} \\
(\exists s1 s2) &\quad (ppms1 = s1 =_p s2 \lor ppms1 = s1 \in_p s2) \\
\exists p &\quad (ppms1 = \{p\} \land ppms1 = \exists p s p) \\
\exists p1 p2 &\quad (ppms1 = \{p1; p2\} \land ppms1 = p1 \land p p2) \end{align*} \] \]

\[ Fof \vdash Fof = \text{HeredFun} (\text{Rules2Fun} FofCSet) \]

\[ FofCProp \vdash \forall ppmss \quad (\text{FofCProp ppmss} \iff \begin{align*}
(\forall s1 s2) &\quad (s1 =_p s2 \in ppmss \land s1 \in_p s2 \in ppmss) \\
\forall p &\quad (p \in ppmss \Rightarrow (p \in ppmss \land (\forall s (\exists p s p \in ppmss)))) \\
\forall p1 p2 &\quad (p1 \in ppmss \land p2 \in ppmss \Rightarrow p1 \land p p2 \in ppmss) \end{align*} \] \]

\[ Fof2 \vdash Fof2 = \bigcap \{s \mid \text{FofCProp s} \} \]

7.5 Theorems

\[ \text{fof induction thm} \quad \vdash \forall s \quad \text{FofCProp s} \Rightarrow Fof2 \subseteq s \]

\[ \text{fof induction thm2} \quad \vdash \forall s \quad \begin{align*}
(\forall s1 s2) &\quad (s1 =_p s2 \in s \land s1 \in_p s2 \in s) \\
\forall p &\quad (p \in s \Rightarrow (\exists p s p \in s)) \\
\forall p1 p2 &\quad (p1 \in s \land p2 \in s \Rightarrow p1 \land p p2 \in s) \end{align*} \]

\[ \Rightarrow Fof2 \subseteq s \]
INDEX

+ .......................................................... 9
Θ .......................................................... 9
Ξ .......................................................... 9
γ .......................................................... 9
ω .......................................................... 9
φ .......................................................... 9
ρ .......................................................... 9
× .......................................................... 9
υ .......................................................... 9
Fof .......................................................... 9, 10
fof_induction_thm .................................... 10
fof_induction_thm2 ................................... 10
FofCProp ............................................... 9, 10
FofCSet ................................................ 9, 10
Fof2 ..................................................... 9, 10