fix[HULL[binclosed[x]]]

Johan G. F. Belinfante
2003 September 14

```
In[1]:= << goedel52.s93; << tools.m

        :Package Title: goedel52.s93      2003 September 13 at 2:30 p.m.

        It is now:  2003 Sep 15 at 9:18

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 August 9

        weightlimit = 40
```

summary

Rewrite rules of the form **fix[HULL[x]] = x** are derived for several classes, including the class **TOPS** of all topologies.

a general lemma

```
In[2]:= hull[x_, y_] := A[intersection[x, image[S, singleton[y]]]]

        General::spell1 :
         Possible spelling error: new symbol name "hull" is similar to existing symbol "full". More...
```

The lemma proved in this section is motivated by the following observation:

```
In[3]:= assert[forall[y, implies[member[y, domain[HULL[x]]], member[hull[x, y], x]]]]

Out[3]= subclass[image[inverse[S], x], image[inverse[HULL[x]], x]]

In[4]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
            {u -> image[inverse[S], x], v -> image[inverse[HULL[x]], x], w -> HULL[x]}]

Out[4]= or[not[subclass[image[inverse[S], x], image[inverse[HULL[x]], x]]],
          subclass[fix[HULL[x]], x]] == True

In[5]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The following version is an improvement:

```
In[6]:= implies[subclass[image[inverse[S], x], image[inverse[HULL[x]], x]],
          equal[fix[HULL[x]], x]] // AssertTest

Out[6]= or[equal[x, fix[HULL[x]]],
          not[subclass[image[inverse[S], x], image[inverse[HULL[x]], x]]]] == True

In[7]:= or[equal[x_, fix[HULL[x_]]],
          not[subclass[image[inverse[S], x_], image[inverse[HULL[x_]], x_]]]] := True
```

binclosed

The key fact used in this section is this:

```
In[8]:= implies[subclass[y, binclosed[x]], or[equal[0, y], member[A[y], binclosed[x]]]]

Out[8]= True
```

The first lemma is a special case of this:

```
In[9]:= SubstTest[implies, subclass[w, binclosed[x]],
         or[equal[0, w], member[A[w], binclosed[x]]],
        w -> intersection[binclosed[x], image[S, singleton[y]]]]

Out[9]= or[not[member[y, image[inverse[S], binclosed[x]]]],
          subclass[image[x, cart[A[intersection[binclosed[x], image[S, singleton[y]]]],
             A[intersection[binclosed[x], image[S, singleton[y]]]]]],
           A[intersection[binclosed[x], image[S, singleton[y]]]]] == True

In[10]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[11]:= implies[member[y, domain[HULL[binclosed[x]]]],
          member[hull[binclosed[x], y], binclosed[x]]]

Out[11]= True
```

The variable **y** is eliminated as follows:

```
In[12]:= Map[equal[V, #] &, SubstTest[class, y, implies[
             member[y, domain[HULL[z]]], member[hull[z, y], z]], z -> binclosed[x]]] // Reverse

Out[12]= subclass[image[inverse[S], binclosed[x]],
           image[inverse[HULL[binclosed[x]]], binclosed[x]]] == True

In[13]:= subclass[image[inverse[S], binclosed[x_]],
           image[inverse[HULL[binclosed[x_]]], binclosed[x_]]] := True
```

The general lemma derived in the preceding section can now be applied:

```
In[14]:= SubstTest[implies, subclass[image[inverse[S], y], image[inverse[HULL[y]], y]],
          equal[fix[HULL[y]], y], y -> binclosed[x]]

Out[14]= equal[binclosed[x], fix[HULL[binclosed[x]]]] == True

In[15]:= fix[HULL[binclosed[x_]]] := binclosed[x]
```

the case of invar

The rule for **invar[x]** follows as a special case of the **binclosed[x]** rule.

```
In[16]:= SubstTest[fix, HULL[binclosed[y]], y -> composite[x, inverse[DUP]]]

Out[16]= fix[HULL[invar[x]]] == invar[x]

In[17]:= fix[HULL[invar[x_]]] := invar[x]
```

The following is also of interest:

```
In[18]:= SubstTest[implies, subclass[x, binclosed[z]],
           subclass[image[z, cart[A[x], A[x]]], A[x]], z -> composite[y, inverse[DUP]]]

Out[18]= or[not[subclass[x, invar[y]]], subclass[image[y, A[x]], A[x]]] == True

In[19]:= or[not[subclass[x_, invar[y_]]], subclass[image[y_, A[x_]], A[x_]]] := True
```

TOPS rule

The rule for **fix[HULL[TOPS]]** is now an easy corollary:

```
In[20]:= SubstTest[implies, and[equal[fix[HULL[x]], x], equal[fix[HULL[y]], y]],
           equal[fix[HULL[intersection[x, y]]], intersection[x, y]],
             {x -> fix[UCLOSURE], y -> binclosed[CAP]}]

Out[20]= equal[TOPS, fix[HULL[TOPS]]] == True

In[21]:= fix[HULL[TOPS]] := TOPS
```