identity functions

Johan G. F. Belinfante and Tiffany D. Goble
2004 April 9

```
In[1]:= << goedel56.07b << tools.m;

        :Package Title: goedel56.07b           2004 April 7 at 9:00 p.m.

        It is now:  2004 Apr 9 at 13:23

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 April 6

        weightlimit = 40
```

summary

The term ***identity function*** refers to any subclass of **Id**. All such functions can of course be written in the form **id[x]**, but for certain reasoning processes about identity functions it is desirable to avoid the **id** wrapper, which triggers numerous rewrite rules. This notebook contains derivations of some clauses involving the literal **subclass[x, Id]** that have previously been proved using **Otter**.

a theorem in the ID−1 group

Corollary **ID−8−COR** in the **ID−1** group is quickly obtained using equality substitution:

```
In[2]:= SubstTest[implies, and[equal[x, y], SYMMETRIC[y]], SYMMETRIC[x], y -> id[fix[x]]]

Out[2]= or[equal[x, inverse[x]], not[subclass[x, Id]]] == True

In[3]:= or[equal[x_, inverse[x_]], not[subclass[x_, Id]]] := True
```

generalizing a theorem in the ID−CO group

The following rewrite rule generalizes Theorem **ID−CP−V** in the **ID−CO** group:

```
In[4]:= equal[Id, cart[x, y]] // AssertTest

Out[4]= equal[Id, cart[x, y]] == False

In[5]:= equal[Id, cart[x_, y_]] := False
```

theorems in the ID−2 group

Corollary **COR−ID−E** in the **ID−2** group is easily derived as follows:

```
In[6]:= Map[not, SubstTest[implies, subclass[x, Id], FUNCTION[x], x -> E]]

Out[6]= subclass[E, Id] == False

In[7]:= subclass[E, Id] := False
```

The following procedure derives Theorem **ID−5E** directly, bypassing Lemma **ID−5D**.

```
In[8]:= SubstTest[implies, and[subclass[x, y], FUNCTION[y]],
          equal[x, composite[y, id[domain[x]]]], y -> Id]

Out[8]= or[equal[x, id[domain[x]]], not[subclass[x, Id]]] == True

In[9]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The converse also holds:

```
In[10]:= SubstTest[implies, and[equal[x, w], subclass[w, z]],
           subclass[x, z], {w -> id[y], z -> Id}]

Out[10]= or[not[equal[x, id[y]]], subclass[x, Id]] == True

In[11]:= (% /. {x -> x_, y → y_}) /. Equal -> SetDelayed
```

Consequently, one has:

```
In[12]:= equiv[equal[x, id[domain[x]]], subclass[x, Id]]

Out[12]= True
```

It is useful to make this a rewrite rule:

```
In[13]:= equal[x_, id[domain[x_]]] := subclass[x, Id]
```

As an immediate corollary, one obtains **ID−3−COR** in the **ID−2** group:

```
In[14]:= SubstTest[implies, and[member[u, v], equal[v, y]],
           member[u, y], {u → pair[x, x], v → id[domain[y]]}]

Out[14]= or[member[pair[x, x], y], not[member[x, domain[y]]], not[subclass[y, Id]]] == True

In[15]:= or[member[pair[x_, x_], y_], not[member[x_, domain[y_]]], not[subclass[y_, Id]]] := True
```

a similar result

In this section a result analogous to Theorem **ID−5E** is derived, but with **range** in place of **domain**.

```
In[16]:= SubstTest[implies, and[equal[x, y], equal[y, z]],
          equal[x, z], {y → id[domain[x]], z → id[range[x]]}]

Out[16]= or[equal[x, id[range[x]]],
           not[equal[domain[x], range[x]]], not[subclass[x, Id]]] == True

In[17]:= (% /. x → x_) /. Equal → SetDelayed

In[18]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 → subclass[x, Id], p2 → equal[domain[x], range[x]], p3 → equal[x, id[range[x]]]}]]

Out[18]= or[equal[x, id[range[x]]], not[subclass[x, Id]]] == True

In[19]:= (% /. x → x_) /. Equal → SetDelayed
```

Again, one can combine this result with its converse to obtain a simple rewrite rule.

```
In[20]:= equiv[equal[x, id[range[x]]], subclass[x, Id]]

Out[20]= True

In[21]:= equal[x_, id[range[x_]]] := subclass[x, Id]
```