Lemma IM−IN

Johan G. F. Belinfante and Tiffany Goble
2004 February 28

```
In[1]:= << goedel54.28a; << tools.m

        :Package Title: goedel54.28a     2004 February 28 at 5:40 a.m.

        It is now:  2004 Mar 5 at 13:14

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 February 21

        weightlimit = 40
```

summary

Lemma **IM−IN** is a technical lemma proved 1997 November 15 using McCune's automated theorem proving program **Otter**, needed to prove monotonicity properties of **IMAGE**. Because this lemma is currently not automatically recognized as true by the **GOEDEL** program, it is rederived here, and will be added as a new rewrite rule.

statement of the theorem

Lemma.

```
In[2]:= SubstTest[implies, subclass[u, v], subclass[image[u, x], image[v, x]],
            {u -> id[image[y, z]], v -> composite[y, id[z], inverse[y]]}]

Out[2]= subclass[intersection[x, image[y, z]],
          image[y, intersection[z, image[inverse[y], x]]]] == True

In[3]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

The derivation of Lemma **IM−IN** is completed as follows:

```
In[4]:= SubstTest[implies, and[subclass[x, u], subclass[u, v]], subclass[x, v],
            {u -> intersection[x, image[y, z]],
          v -> image[y, intersection[z, image[inverse[y], x]]]}]

Out[4]= or[not[subclass[x, image[y, z]]],
          subclass[x, image[y, intersection[z, image[inverse[y], x]]]]] == True

In[5]:= or[not[subclass[x_, image[y_, z_]]],
          subclass[x_, image[y_, intersection[z_, image[inverse[y_], x_]]]]] := True
```

The following restatement can be given:

```
In[6]:= implies[subclass[x, image[y, z]], subvariant[composite[y, id[z], inverse[y]], x]]

Out[6]= True
```

The following special case requires a separate rewrite rule:

```
In[7]:= SubstTest[implies, subclass[x, image[y, z]],
          subvariant[composite[y, id[z], inverse[y]], x], z -> V]

Out[7]= or[not[subclass[x, range[y]]], subclass[x, image[y, image[inverse[y], x]]]] == True
```

```
In[8]:= or[not[subclass[x_, range[y_]]], subclass[x_, image[y_, image[inverse[y_], x_]]]] := True
```

For the special case that **x** is a set, the variable **x** can be eliminated from this expression. This is done in the next section.

variable−elimination for the case of sets

For the special case of sets, one can formulate the result using fewer variables. The basic case is this:

```
In[9]:= SubstTest[subclass, P[fix[w]], subvar[w], w -> composite[x, inverse[x]]]

Out[9]= subclass[P[range[x]], subvar[composite[x, inverse[x]]]] == True
```

```
In[10]:= subclass[P[range[x_]], subvar[composite[x_, inverse[x_]]]] := True
```

The more general case is done in a similar fashion:

```
In[11]:= SubstTest[subclass, P[fix[w]], subvar[w], w -> composite[x, id[y], inverse[x]]]

Out[11]= subclass[P[image[x, y]], subvar[composite[x, id[y], inverse[x]]]] == True
```

```
In[12]:= subclass[P[image[x_, y_]], subvar[composite[x_, id[y_], inverse[x_]]]] := True
```

totally variable−free

It is possible to formulate versions of the theorem with no variables at all. The basic idea is this:

```
In[13]:= Map[equal[0, #] &, SubstTest[reify, x, dif[P[f[x]], subvar[x]], f -> fix]] // Reverse

Out[13]= subclass[composite[SUBVAR, S, IMAGE[DUP]], E] == True
```

```
In[14]:= subclass[composite[SUBVAR, S, IMAGE[DUP]], E] := True
```