rules about domain, range and image

Johan G. F. Belinfante
2004 February 8

```
In[1]:= << goedel54.08a; << tools.m

        :Package Title: goedel54.08a      2004 February 8 at 12:35 midnite

        It is now:  2004 Feb 9 at 13:37

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 January 3

        weightlimit = 40
```

summary

The results presented in this notebook are part of an ongoing effort to assure that the **GOEDEL** program can easily recognize theorems that have been proved using **Otter,** and whenever possible to improve upon those theorems. All the theorems in this notebook have in common the feature that the hypotheses include a membership statement involving ordered pairs. To facilitate pattern matching, the new rules purposefully have variables that appear in the hypotheses, but not in the conclusion. Replacements for two rules about **domain** and **range** are derived, and two new ones are added. The two old rules, which apply only to the special case **u = v = V**, can be removed after the new ones are added to the **GOEDEL** program. In addition, some analogous rules about **image** are derived.

domain rules

The following uses an existing rule to derive a more general one. The existing rule, which is the special case **u = v = V**, is akin to Theorem **DO−1C** which was proved 1994 May 8 using **Otter**.

```
In[2]:= Map[not, SubstTest[and, implies[p1, p4], implies[p2, p5],
          implies[and[p3, p4, p5], p6], not[implies[and[p1, p2, p3], p6]],
          {p1 -> member[x, u], p2 -> member[y, v], p3 -> member[pair[x, y], z],
           p4 -> member[x, V], p5 -> member[y, V], p6 -> member[x, domain[z]]}]]

Out[2]= or[member[x, domain[z]], not[member[x, u]],
          not[member[y, v]], not[member[pair[x, y], z]]] == True

In[3]:= or[member[x_, domain[z_]], not[member[x_, u_]],
          not[member[y_, v_]], not[member[pair[x_, y_], z_]]] := True
```

The following lemma will be used to derive a variant.

```
In[4]:= SubstTest[implies, and[member[s, z], subclass[z, t]], member[s, t],
            {s -> pair[x, y], t -> cart[u, v]}]

Out[4]= or[and[member[x, u], member[y, v]],
          not[member[pair[x, y], z]], not[subclass[z, cart[u, v]]]] == True

In[5]:= (% /. {u -> u_, v -> v_, x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

Corollary **DO−1C'**, proved 1997 April 20 using **Otter**, is the special case **u = v = V** of the following:

```
In[6]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
          implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
            {p1 -> subclass[z, cart[u, v]], p2 -> member[pair[x, y], z],
           p3 -> and[member[x, u], member[y, v]], p4 -> member[x, domain[z]]}]]

Out[6]= or[member[x, domain[z]], not[member[pair[x, y], z]], not[subclass[z, cart[u, v]]]] == True

In[7]:= or[member[x_, domain[z_]],
          not[member[pair[x_, y_], z_]], not[subclass[z_, cart[u_, v_]]]] := True
```

This new rule does not replace any existing rule in the **GOEDEL** program, so nothing needs to be removed.

range rules

In this section, the **range** counterparts of the results in the preceding section are derived. The first result replaces an existing rule which should be removed.

```
In[8]:= Map[not, SubstTest[and, implies[p1, p4], implies[p2, p5],
          implies[and[p3, p4, p5], p6], not[implies[and[p1, p2, p3], p6]],
          {p1 -> member[x, u], p2 -> member[y, v], p3 -> member[pair[x, y], z],
           p4 -> member[x, V], p5 -> member[y, V], p6 -> member[y, range[z]]}]]

Out[8]= or[member[y, range[z]], not[member[x, u]],
          not[member[y, v]], not[member[pair[x, y], z]]] == True

In[9]:= or[member[y_, range[z_]], not[member[x_, u_]],
          not[member[y_, v_]], not[member[pair[x_, y_], z_]]] := True
```

The same lemma that was used in the preceding section suffices to derive the following corollary, which does not replace any existing rule.

```
In[10]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
             {p1 -> subclass[z, cart[u, v]], p2 -> member[pair[x, y], z],
            p3 -> and[member[x, u], member[y, v]], p4 -> member[y, range[z]]}]]

Out[10]= or[member[y, range[z]], not[member[pair[x, y], z]], not[subclass[z, cart[u, v]]]] == True

In[11]:= or[member[y_, range[z_]],
           not[member[pair[x_, y_], z_]], not[subclass[z_, cart[u_, v_]]]] := True
```

images and inverse images

The rules for domain and range can be generalized to the case of images and inverse images. The image formula follows as a corollary of the one for ranges:

```
In[12]:= SubstTest[implies, member[pair[u, v], composite[id[y], w, id[x]]],
          member[v, range[w]], w -> composite[z, id[x]]]

Out[12]= or[member[v, image[z, x]], not[member[u, x]],
           not[member[v, y]], not[member[pair[u, v], z]]] == True

In[13]:= or[member[v_, image[z_, x_]], not[member[u_, x_]],
           not[member[v_, y_]], not[member[pair[u_, v_], z_]]] := True
```

Similarly, the formula for inverse images follows from the one for domains:

```
In[14]:= SubstTest[implies, member[pair[u, v], composite[id[y], w, id[x]]],
          member[u, domain[w]], w -> composite[id[y], z]]

Out[14]= or[member[u, image[inverse[z], y]], not[member[u, x]],
           not[member[v, y]], not[member[pair[u, v], z]]] == True

In[15]:= or[member[u_, image[inverse[z_], y_]], not[member[u_, x_]],
           not[member[v_, y_]], not[member[pair[u_, v_], z_]]] := True
```

two more image rules that required some reasoning

In the following rule, the variables **u** and **y** occur in the hypotheses, but not in the conclusion.

```
In[16]:= Map[not,
          SubstTest[and, implies[and[p1, p2], p3], implies[p1, p4], implies[and[p3, p4], p5],
             implies[and[p1, p2], p6], implies[and[p2, p5, p6], p7],
           not[implies[and[p1, p2], p7]], {p1 -> subclass[z, cart[x, y]],
            p2 -> member[pair[u, v], z], p3 -> member[u, domain[z]], p4 -> subclass[domain[z], x],
            p5 -> member[u, x], p6 -> member[v, range[z]], p7 -> member[v, image[z, x]]}]]

Out[16]= or[member[v, image[z, x]],
           not[member[pair[u, v], z]], not[subclass[z, cart[x, y]]]] == True

In[17]:= or[member[v_, image[z_, x_]],
           not[member[pair[u_, v_], z_]], not[subclass[z_, cart[x_, y_]]]] := True
```

There is an analog for inverse images:

```
In[18]:= Map[not,
          SubstTest[and, implies[and[p1, p2], p3], implies[p1, p4], implies[and[p3, p4], p5],
             implies[and[p1, p2], p6], implies[and[p2, p5, p6], p7],
           not[implies[and[p1, p2], p7]],
           {p1 -> subclass[z, cart[x, y]], p2 -> member[pair[u, v], z],
            p3 -> member[v, range[z]], p4 -> subclass[range[z], y], p5 -> member[v, y],
            p6 -> member[u, domain[z]], p7 -> member[u, image[inverse[z], y]]}]]

Out[18]= or[member[u, image[inverse[z], y]],
           not[member[pair[u, v], z]], not[subclass[z, cart[x, y]]]] == True

In[19]:= or[member[u_, image[inverse[z_], y_]],
           not[member[pair[u_, v_], z_]], not[subclass[z_, cart[x_, y_]]]] := True
```