subclass[image[complement[x],y],z]

Johan G. F. Belinfante
2004 February 18

```
In[1]:= << goedel54.17a; << tools.m

        :Package Title: goedel54.17a      2004 February 17 at 8:30 a.m.

        It is now:  2004 Feb 18 at 13:25

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 February 11

        weightlimit = 40
```

summary

Existing rewrite rules for statements of the form **subclass[cart[x,y],z]** and **subclass[image[x,y],z]** in the **GOEDEL** program have been designed to eliminate occurrences of **complement** and **inverse** whenever possible. A new rule is derived in this notebook which does not immediately eliminate **inverse**, but does so under certain circumstances. Adding this rewrite rule has the advantage that it helps to automatically recognize the equivalence of various statements of this kind by converting them to a common standard form. A drawback of the proposed rule is that it requires adding additional rewrite rules if it is desired that the truth of various standard facts about upper and lower bounds be automatically recognized.

derivation

The new rule, derived using **AssertTest**, does not appear to eliminate anything,, but only converts **image** into **cart**.

```
In[2]:= subclass[cart[y, complement[z]], x] // AssertTest // Reverse

Out[2]= subclass[image[complement[x], y], z] == subclass[cart[y, complement[z]], x]

In[3]:= subclass[image[complement[x_], y_], z_] := subclass[cart[y, complement[z]], x]
```

However, if one applies it to the following special case, it eliminates **inverse**.

```
In[4]:= subclass[image[complement[inverse[x]], y], z]

Out[4]= subclass[cart[complement[z], y], x]
```

An experiment in which the rule for this special case was turned around led to looping in evaluating the following expression:

```
In[5]:= union[cart[singleton[0], V], composite[complement[S], BIGCAP]] // complement

Out[5]= union[complement[cart[V, V]], composite[S, BIGCAP]]
```

antitone property of upper and lower bounds

The proposed orientation of the new rewrite rule is not without drawbacks. The original inclination was to orient the rule in the opposite direction to avoid special rules for certain theorems about upper and lower bounds. For convenience the following definitions are introduced for classes of upper and lower bounds:

```
In[6]:= ub[x_, y_] := complement[image[complement[x], y]]
```

```
In[7]:= lb[x_, y_] := complement[image[complement[inverse[x]], y]]
```

The antitone properties of **lb** and **ub** require special rules on account of the orientation of the new rewrite rule.

```
In[8]:= SubstTest[implies, subclass[y, z],
         subclass[ub[complement[w], z], ub[complement[w], y]], w -> complement[x]]
Out[8]= or[not[subclass[y, z]],
          subclass[cart[y, complement[image[complement[x], z]]], x]] == True
```

```
In[9]:= or[not[subclass[y_, z_]],
          subclass[cart[y_, complement[image[complement[x_], z_]]], x_]] := True
```

Restatement:

```
In[10]:= implies[subclass[y, z], subclass[ub[x, z], ub[x, y]]]
Out[10]= True
```

A similar story holds for lower bounds.

```
In[11]:= SubstTest[implies, subclass[y, z],
          subclass[lb[complement[w], z], lb[complement[w], y]], w -> complement[x]]
Out[11]= or[not[subclass[y, z]],
           subclass[cart[complement[image[complement[inverse[x]], z]], y], x]] == True
```

```
In[12]:= or[not[subclass[y_, z_]],
           subclass[cart[complement[image[complement[inverse[x_]], z_]], y_], x_]] := True
```

```
In[13]:= implies[subclass[y, z], subclass[lb[x, z], lb[x, y]]]
Out[13]= True
```

ub−lb and lb−ub rules

On account of the new rewrite rule, the following inclusions are not recognized automatically, but must be added as separate rules:

```
In[14]:= SubstTest[subclass, y, lb[complement[z], ub[complement[z], y]], z -> complement[x]]
Out[14]= subclass[cart[y, complement[image[complement[x], y]]], x] == True
```

```
In[15]:= subclass[cart[y_, complement[image[complement[x_], y_]]], x_] := True
```

```
In[16]:= SubstTest[subclass, y, ub[complement[z], lb[complement[z], y]], z -> complement[x]]

Out[16]= subclass[cart[complement[image[complement[inverse[x]], y]], y], x] == True

In[17]:= subclass[cart[complement[image[complement[inverse[x_]], y_]], y_], x_] := True
```

Restatement:

```
In[18]:= subclass[y, lb[x, ub[x, y]]]

Out[18]= True

In[19]:= subclass[y, ub[x, lb[x, y]]]

Out[19]= True
```

the third image theorem

Lemma.

```
In[20]:= SubstTest[implies, subclass[y, z], subclass[image[x, y], image[x, z]],
          z -> complement[image[inverse[x], complement[image[x, y]]]]]

Out[20]= subclass[image[x, y],
           image[x, complement[image[inverse[x], complement[image[x, y]]]]]] == True

In[21]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[22]:= SubstTest[subclass, z,
          complement[image[x, complement[image[inverse[x], z]]]], z -> complement[image[x, y]]]

Out[22]= subclass[
           image[x, complement[image[inverse[x], complement[image[x, y]]]]], image[x, y]] == True

In[23]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

These two inclusions can be combined into an equation which reduces a certain combination of three images to a single image:

```
In[24]:= SubstTest[and, subclass[u, v], subclass[v, u],
             {u -> image[x, complement[image[inverse[x], complement[image[x, y]]]]],
           v -> image[x, y]}]

Out[24]= True ==
          equal[image[x, y], image[x, complement[image[inverse[x], complement[image[x, y]]]]]]

In[25]:= image[x_, complement[image[inverse[x_], complement[image[x_, y_]]]]] := image[x, y]
```

analogous rules

The third image theorem in the preceding section has three counterparts obtained by replacing x with its inverse, its complement, or the complement of the inverse. All three of these rules are derived here.

```
In[26]:= SubstTest[image, w,
          complement[image[inverse[w], complement[image[w, y]]]], w -> inverse[x]]

Out[26]= image[inverse[x], complement[image[x, complement[image[inverse[x], y]]]]] ==
          image[inverse[x], y]

In[27]:= image[inverse[x_], complement[image[x_, complement[image[inverse[x_], y_]]]]] :=
          image[inverse[x], y]

In[28]:= SubstTest[image, w,
          complement[image[inverse[w], complement[image[w, y]]]], w -> complement[x]]

Out[28]= image[complement[x],
           complement[image[complement[inverse[x]], complement[image[complement[x], y]]]]] ==
          image[complement[x], y]

In[29]:= image[complement[x_], complement[image[complement[inverse[x_]],
             complement[image[complement[x_], y_]]]]] := image[complement[x], y]

In[30]:= SubstTest[image, w, complement[image[inverse[w], complement[image[w, y]]]],
          w -> complement[inverse[x]]]

Out[30]= image[complement[inverse[x]],
           complement[image[complement[x], complement[image[complement[inverse[x]], y]]]]] ==
          image[complement[inverse[x]], y]

In[31]:= image[complement[inverse[x_]], complement[
            image[complement[x_], complement[image[complement[inverse[x_]], y_]]]] :=
          image[complement[inverse[x]], y]
```

The third image theorems look better when restated as follows:

```
In[32]:= lb[x, ub[x, lb[x, y]]] == lb[x, y]

Out[32]= True

In[33]:= ub[x, lb[x, ub[x, y]]] == ub[x, y]

Out[33]= True
```

relational versions

In this section, two relational versions of the third image theorems are derived in which the variable **y** has been eliminated.

```
In[34]:= Map[composite[Id, complement[#]] &,
           SubstTest[reify, y, ub[x, lb[x, ub[z, y]]], z -> x]] // Reverse

Out[34]= composite[complement[x], complement[composite[complement[inverse[x]], UB[x]]]] ==
          composite[complement[x], inverse[E]]

In[35]:= composite[complement[x_], complement[composite[complement[inverse[x_]], UB[x_]]]] :=
          composite[complement[x], inverse[E]]

In[36]:= Map[composite[Id, complement[#]] &,
           SubstTest[reify, y, lb[x, ub[x, lb[z, y]]], z -> x]] // Reverse

Out[36]= composite[complement[inverse[x]], complement[composite[complement[x], LB[x]]]] ==
          composite[complement[inverse[x]], inverse[E]]
```

```
In[37]:= composite[complement[inverse[x_]], complement[composite[complement[x_], LB[x_]]]] :=
          composite[complement[inverse[x]], inverse[E]]
```