various rewrite rules

Johan G. F. Belinfante and Tiffany D. Goble
2004 March 31

```
In[1]:= << goedel55.29a; << tools.m

        :Package Title: goedel55.29a           2004 March 29 at 3:25 p.m.

        It is now:  2004 Mar 31 at 13:44

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 March 14

        weightlimit = 40
```

summary

This notebook contains a diverse collection of rewrite rules discovered in the course of re−examining various theorems in the **CO**, **E**, **SR** and **ID** groups proved using **Otter**.

formulas for LB[x] and UB[x]

A variable−free formulation was derived for what Quaife called Boyer's lemma 18. The following lemma proved useful:

```
In[2]:= ImageComp[LB[x], S, y] // Reverse

Out[2]= image[LB[x], image[S, y]] == image[LB[x], y]

In[3]:= image[LB[x_], image[S, y_]] := image[LB[x], y]
```

The elimination of variables from Boyer's lemma yields

```
In[4]:= Map[equal[0, complement[#]] &,
          SubstTest[class, y, implies[subclass[range[x], domain[y]],
            equal[domain[composite[y, x]], z]], z -> domain[x]]] // Reverse

Out[4]= subclass[intersection[image[S, singleton[range[x]]],
           image[inverse[LB[complement[x]]], domain[x]]], 0] == True
```

With the lemma in place, the **GOEDEL** program recognizes this equivalent formulation:

```
In[5]:= equal[intersection[image[S, singleton[range[y]]],
          image[inverse[LB[complement[y]]], domain[y]]], 0]

Out[5]= True
```

This justifies the following rewrite rule:

```
In[6]:= intersection[image[S, singleton[range[y_]]],
          image[inverse[LB[complement[y_]]], domain[y_]]] := 0
```

One can now eliminate the remaining variable, but the final result seems too complicated to be worth saving:

```
In[7]:= SubstTest[reify, y, intersection[image[S, singleton[f[y]]],
           image[inverse[LB[complement[y]]], domain[y]]], f → range] // Reverse

Out[7]= intersection[composite[S, IMAGE[SECOND]],
          composite[SECOND, intersection[composite[inverse[FIRST], inverse[E], IMAGE[FIRST]],
            UB[complement[cross[Id, E]]]]]] == 0
```

The verdict: perhaps the most useful result obtained is the lemma. A companion result:

```
In[8]:= ImageComp[UB[x], S, y] // Reverse

Out[8]= image[UB[x], image[S, y]] == image[UB[x], y]

In[9]:= image[UB[x_], image[S, y_]] := image[UB[x], y]
```

Theorem CO−E5

Theorem **CO−E5** in the **CO4** group of theorems proved using **Otter** is recognized to be true by the **GOEDEL** program once one adds this rewrite rule:

```
In[10]:= member[pair[x, y], composite[complement[S], IMAGE[z]]] // AssertTest

Out[10]= member[pair[x, y], composite[complement[S], IMAGE[z]]] ==
           and[member[x, V], member[y, V], member[image[z, x], V], not[subclass[image[z, x], y]]]

In[11]:= member[pair[x_, y_], composite[complement[S], IMAGE[z_]]] :=
           and[member[x, V], member[y, V], member[image[z, x], V], not[subclass[image[z, x], y]]]
```

The following formula was also discovered in the course of this study:

```
In[12]:= fix[composite[E, complement[inverse[x]], UB[x]]] // Normality

Out[12]= fix[composite[E, complement[inverse[x]], UB[x]]] == 0

In[13]:= fix[composite[E, complement[inverse[x]], UB[x]]] := 0
```

This result is analogous to an existing rewrite rule:

```
In[14]:= fix[composite[E, complement[x], LB[x]]]

Out[14]= 0
```

Theorem CO−DJ−RO

The following result is related to the Schroeder rotation theorem:

```
In[15]:= or[equal[0, intersection[composite[y, z], inverse[x]]],
            not[equal[0, intersection[composite[x, y], inverse[z]]]]] // AssertTest

Out[15]= or[equal[0, intersection[composite[y, z], inverse[x]]],
            not[equal[0, intersection[composite[x, y], inverse[z]]]]] == True

In[16]:= or[equal[0, intersection[composite[y_, z_], inverse[x_]]],
            not[equal[0, intersection[composite[x_, y_], inverse[z_]]]]] := True
```

The result can best be understood in light of the following observations:

```
In[17]:= equal[0, intersection[composite[y, z], inverse[x]]] // AssertTest

Out[17]= equal[0, intersection[composite[y, z], inverse[x]]] == equal[0, fix[composite[z, x, y]]]

In[18]:= implies[equal[0, fix[composite[x, y]]], equal[0, fix[composite[y, x]]]]

Out[18]= True
```

Theorem E−C−A−1

Theorem **E−C−A−1** in the **E2** group of theorems proved with **Otter** is recognized by the **GOEDEL** program once the following rewrite rule is added.

```
In[19]:= member[pair[x, y], composite[inverse[BIGCAP], complement[E]]] // AssertTest

Out[19]= member[pair[x, y], composite[inverse[BIGCAP], complement[E]]] ==
          and[member[x, V], member[y, V], not[member[x, A[y]]]]

In[20]:= member[pair[x_, y_], composite[inverse[BIGCAP], complement[E]]] :=
          and[member[x, V], member[y, V], not[member[x, A[y]]]]
```

Theorems E−CO4 and E−CO5

The following rewrite rules are analogous to theorems **E−CO4** and **E−CO5** in the **E2** group.

```
In[21]:= subclass[composite[E, x], composite[E, y]] // AssertTest

Out[21]= subclass[composite[E, x], composite[E, y]] == subclass[composite[Id, x], y]

In[22]:= subclass[composite[E, x_], composite[E, y_]] := subclass[composite[Id, x], y]
```

The following corollary is immediate:

```
In[23]:= equal[composite[E, x], composite[E, y]] // AssertTest

Out[23]= equal[composite[E, x], composite[E, y]] == equal[composite[Id, x], composite[Id, y]]

In[24]:= equal[composite[E, x_], composite[E, y_]] := equal[composite[Id, x], composite[Id, y]]
```

The following companion rules are derived in the same way:

```
In[25]:= subclass[composite[x, inverse[E]], composite[y, inverse[E]]] // AssertTest

Out[25]= subclass[composite[x, inverse[E]], composite[y, inverse[E]]] ==
          subclass[composite[Id, x], y]

In[26]:= subclass[composite[x_, inverse[E]], composite[y_, inverse[E]]] :=
          subclass[composite[Id, x], y]

In[27]:= equal[composite[x, inverse[E]], composite[y, inverse[E]]] // AssertTest

Out[27]= equal[composite[x, inverse[E]], composite[y, inverse[E]]] ==
          equal[composite[Id, x], composite[Id, y]]

In[28]:= equal[composite[x_, inverse[E]], composite[y_, inverse[E]]] :=
          equal[composite[Id, x], composite[Id, y]]
```

other theorems in the E2 group

The following corollary of monotonicity is needed to recognize another theorem in the **E2** group:

```
In[29]:= SubstTest[implies, subclass[u, v], subclass[composite[z, u], composite[z, v]],
          {u → composite[Id, x], v → composite[Id, y]}]

Out[29]= or[not[subclass[composite[Id, x], y]],
           subclass[composite[z, x], composite[z, y]]] == True

In[30]:= or[not[subclass[composite[Id, x_], y_]],
           subclass[composite[z_, x_], composite[z_, y_]]] := True
```

This is similar, but with the factors reversed:

```
In[31]:= SubstTest[implies, subclass[u, v],
          subclass[composite[u, z], composite[v, z]], {u → composite[Id, x], v → y}]

Out[31]= or[not[subclass[composite[Id, x], y]],
           subclass[composite[x, z], composite[y, z]]] == True

In[32]:= or[not[subclass[composite[Id, x_], y_]],
           subclass[composite[x_, z_], composite[y_, z_]]] := True
```

Lemma:

```
In[33]:= implies[equal[composite[Id, x], composite[Id, y]],
           subclass[composite[z, x], composite[z, y]]] // AssertTest

Out[33]= or[not[equal[composite[Id, x], composite[Id, y]]],
           subclass[composite[z, x], composite[z, y]]] == True

In[34]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem:

```
In[35]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
           not[implies[p1, p4]], {p1 → equal[composite[E, x], composite[E, y]],
            p2 → subclass[composite[z, x], composite[z, y]],
            p3 → subclass[composite[z, y], composite[z, x]],
            p4 → equal[composite[z, y], composite[z, x]]}]]

Out[35]= or[equal[composite[z, x], composite[z, y]],
           not[equal[composite[Id, x], composite[Id, y]]]] == True
```

```
In[36]:= or[equal[composite[z_, x_], composite[z_, y_]],
            not[equal[composite[Id, x_], composite[Id, y_]]]] := True
```

Similarly, with the factors reversed:

```
In[37]:= implies[equal[composite[Id, x], composite[Id, y]],
            subclass[composite[x, z], composite[y, z]]] // AssertTest

Out[37]= or[not[equal[composite[Id, x], composite[Id, y]]],
            subclass[composite[x, z], composite[y, z]]] == True

In[38]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed

In[39]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
            not[implies[p1, p4]], {p1 → equal[composite[Id, x], composite[Id, y]],
             p2 → subclass[composite[x, z], composite[y, z]],
             p3 → subclass[composite[y, z], composite[x, z]],
             p4 → equal[composite[y, z], composite[x, z]]}]]

Out[39]= or[equal[composite[x, z], composite[y, z]],
            not[equal[composite[Id, x], composite[Id, y]]]] == True

In[40]:= or[equal[composite[x_, z_], composite[y_, z_]],
            not[equal[composite[Id, x_], composite[Id, y_]]]] := True
```

Theorems in the SR groups

Theorem **SR−CO3**:

```
In[41]:= SubstTest[implies, and[subclass[x, v], subclass[v, w]],
             subclass[x, w], {v → composite[x, Id], w → composite[x, S]}]

Out[41]= or[not[subclass[x, cart[V, V]]], subclass[x, composite[x, S]]] == True

In[42]:= or[not[subclass[x_, cart[V, V]]], subclass[x_, composite[x_, S]]] := True
```

Lemma:

```
In[43]:= SubstTest[implies, equal[z, V], subclass[x, z], z → A[y]]

Out[43]= or[not[equal[0, y]], subclass[x, A[y]]] == True

In[44]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[45]:= equiv[and[not[equal[0, y]], not[subclass[x, A[y]]]], not[subclass[x, A[y]]]]

Out[45]= True

In[46]:= and[not[equal[0, y_]], not[subclass[x_, A[y_]]]] := not[subclass[x, A[y]]]
```

The following rule suffices for the **GOEDEL** program to recognize Theorems **SR−C−A1** and **SR−C−A2**.

```
In[47]:= member[pair[x, y], composite[inverse[BIGCAP], complement[S]]] // AssertTest

Out[47]= member[pair[x, y], composite[inverse[BIGCAP], complement[S]]] ==
          and[member[x, V], member[y, V], not[subclass[x, A[y]]]]
```

```
In[48]:= member[pair[x_, y_], composite[inverse[BIGCAP], complement[S]]] :=
          and[member[x, V], member[y, V], not[subclass[x, A[y]]]]
```

Theorem **SR−IM−2D**:

```
In[49]:= SubstTest[implies, equal[u, v], equal[U[u], U[v]], {u → V, v → image[inverse[S], x]}]

Out[49]= or[equal[V, U[x]], not[equal[V, image[inverse[S], x]]]] == True

In[50]:= or[equal[V, U[x_]], not[equal[V, image[inverse[S], x_]]]] := True
```

Theorems in the ID−CO group

Theorem **ID−IN−E**:

```
In[51]:= SubstTest[subclass, composite[x, inverse[x]], Id, x → inverse[E]] // Reverse

Out[51]= FUNCTION[inverse[E]] == False

In[52]:= FUNCTION[inverse[E]] := False
```

Theorem **ID−IN−SR**:

```
In[53]:= FUNCTION[inverse[S]] // AssertTest

Out[53]= FUNCTION[inverse[S]] == False

In[54]:= FUNCTION[inverse[S]] := False
```

Theorem **ID−SU−CO**:

```
In[55]:= SubstTest[implies, and[subclass[x, v], subclass[y, v]],
          subclass[composite[x, y], composite[v, v]], v → Id]

Out[55]= or[not[subclass[x, Id]], not[subclass[y, Id]], subclass[composite[x, y], Id]] == True

In[56]:= or[not[subclass[x_, Id]],
            not[subclass[y_, Id]], subclass[composite[x_, y_], Id]] := True
```

Comment: The following related result is already in the **GOEDEL** program:

```
In[57]:= image[COMPOSE, cart[P[Id], P[Id]]]

Out[57]= P[Id]
```

involutions

Theorem **ID−INVL1** about involutions:

```
In[58]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
           implies[and[p3, p4], p5], implies[and[p2, p5], p6], not[implies[and[p1, p2], p6]],
           {p1 → subclass[composite[y, y], Id], p2 → subclass[x, composite[y, x]],
            p3 → subclass[composite[y, y, x], composite[Id, x]],
            p4 → subclass[composite[y, x], composite[y, y, x]],
            p5 → subclass[composite[y, x], composite[Id, x]], p6 → equal[x, composite[y, x]]}]]

Out[58]= or[equal[x, composite[y, x]],
           not[subclass[x, composite[y, x]]], not[subclass[composite[y, y], Id]]] == True

In[59]:= or[equal[x_, composite[y_, x_]], not[subclass[x_, composite[y_, x_]]],
           not[subclass[composite[y_, y_], Id]]] := True
```

Eliminating the variable **x**:

```
In[60]:= Map[or[subclass[subvar[cross[Id, y]], fix[IMAGE[cross[Id, y]]]], equal[V, #]] &,
           SubstTest[class, x, or[equal[x, composite[y, x]], not[subclass[x, composite[y, x]]],
             not[subclass[w, Id]]], w -> composite[y, y]]] // Reverse

Out[60]= or[not[subclass[composite[y, y], Id]],
           subclass[subvar[cross[Id, y]], fix[IMAGE[cross[Id, y]]]]] == True

In[61]:= (% /. y → y_) /. Equal → SetDelayed
```

This result can be improved:

```
In[62]:= SubstTest[and, implies[p, subclass[u, v]],
           subclass[v, u], {p → subclass[composite[x, x], Id],
            u → subvar[cross[Id, x]], v → fix[IMAGE[cross[Id, x]]]}] // Reverse

Out[62]= or[equal[fix[IMAGE[cross[Id, x]]], subvar[cross[Id, x]]],
           not[subclass[composite[x, x], Id]]] == True

In[63]:= or[equal[fix[IMAGE[cross[Id, x_]]], subvar[cross[Id, x_]]],
           not[subclass[composite[x_, x_], Id]]] := True
```

Similarly:

```
In[64]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
           implies[and[p3, p4], p5], implies[and[p2, p5], p6], not[implies[and[p1, p2], p6]],
           {p1 → subclass[composite[y, y], Id], p2 → subclass[x, composite[x, y]],
            p3 → subclass[composite[x, y, y], composite[x, Id]],
            p4 → subclass[composite[x, y], composite[x, y, y]],
            p5 → subclass[composite[x, y], composite[x, Id]], p6 → equal[x, composite[x, y]]}]]

Out[64]= or[equal[x, composite[x, y]],
           not[subclass[x, composite[x, y]]], not[subclass[composite[y, y], Id]]] == True

In[65]:= or[equal[x_, composite[x_, y_]], not[subclass[x_, composite[x_, y_]]],
           not[subclass[composite[y_, y_], Id]]] := True
```

identity functions

Theorem **ID−4−COR**:

```
In[66]:= SubstTest[implies, and[FUNCTION[v], subclass[x, v]], FUNCTION[x], v → Id]

Out[66]= or[FUNCTION[x], not[subclass[x, Id]]] == True

In[67]:= or[FUNCTION[x_], not[subclass[x_, Id]]] := True
```

Variable−free form is already in the **GOEDEL** program:

```
In[68]:= subclass[P[Id], FUNS]

Out[68]= True
```

Lemma.

```
In[69]:= implies[subclass[x, Id], subclass[inverse[x], Id]] // AssertTest

Out[69]= or[not[subclass[x, Id]], subclass[inverse[x], Id]] == True

In[70]:= (% /. x -> x_) /. Equal -> SetDelayed

In[71]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> subclass[x, Id], p2 -> subclass[inverse[x], Id], p3 -> FUNCTION[inverse[x]]}]]

Out[71]= or[FUNCTION[inverse[x]], not[subclass[x, Id]]] == True

In[72]:= or[FUNCTION[inverse[x_]], not[subclass[x_, Id]]] := True
```

A variable−free version holds for the special case of sets:

```
In[73]:= subclass[P[Id], BIJ]

Out[73]= True
```