binclosed[direct[x,y]]

Johan G. F. Belinfante
2011 December 16

```
In[1]:= SetDirectory["l:"]; << goedel.11dec15a

        :Package Title: goedel.11dec15a              2011 December 15 at 5:30 p.m.

        Loading takes about thirteen minutes, half that time due to builtin pauses.

        It is now:  2011 Dec 16 at 0:24

        Loading Simplification Rules

        TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

        weightlimit = 40

        Loading completed.

        It is now:  2011 Dec 16 at 0:37
```

summary

Rewrite rules for **binclosed[direct[x, y]]** are derived. No conditions are placed on the classes **x** and **y**. When this derivation was first attempted a day ago, slow execution times encountered due to conditional rewrite rules involving a test for thin-ness, which have now been removed (commented out).

derivation

The basic tool for the derivation is a normality test. To expedite execution, two flags will be temporarily cleared.

```
In[2]:= cond = False; simplify = False;
```

Lemma. A temporary rewrite rule.

```
In[3]:= TimeConstrained[binclosed[direct[x, y]] // Normality // Reverse // Timing, 60]

Out[3]= {32.422 Second,
         complement[fix[composite[complement[E], fix[composite[inverse[SECOND], inverse[FIRST],
               fix[composite[inverse[FIRST], inverse[SECOND], inverse[SECOND],
                 fix[composite[inverse[SECOND], y, fix[composite[inverse[SECOND],
                     inverse[FIRST], SECOND, intersection[composite[inverse[FIRST], FIRST,
                       intersection[composite[inverse[SECOND], FIRST, intersection[composite[
                           inverse[SECOND], SECOND, SECOND], composite[inverse[E], FIRST, FIRST,
                           FIRST, FIRST]]], composite[inverse[x], SECOND, FIRST, FIRST]]],
                      composite[inverse[E], FIRST, FIRST, FIRST, FIRST]]]]]]]]]]]]]] ==
          binclosed[composite[cross[x, y], TWIST]]}

In[4]:= complement[fix[composite[complement[E],
           fix[composite[inverse[SECOND], inverse[FIRST], fix[composite[inverse[FIRST],
               inverse[SECOND], inverse[SECOND], fix[composite[inverse[SECOND], y_,
                 fix[composite[inverse[SECOND], inverse[FIRST], SECOND, intersection[
                   composite[inverse[FIRST], FIRST, intersection[composite[inverse[SECOND],
                     FIRST, intersection[composite[inverse[SECOND], SECOND, SECOND],
                       composite[inverse[E], FIRST, FIRST, FIRST, FIRST]]],
                     composite[inverse[x_], SECOND, FIRST, FIRST]]],
                   composite[inverse[E], FIRST, FIRST, FIRST, FIRST]]]]]]]]]]]]] :=
         binclosed[composite[cross[x, y], TWIST]]
```

Each of the next two derivations also takes about half a minute.

Theorem. A simplification rule.

```
In[5]:= image[inverse[IMAGE[id[cart[V, V]]]], binclosed[direct[x, y]]] // Normality

Out[5]= image[inverse[IMAGE[id[cart[V, V]]]], binclosed[composite[cross[x, y], TWIST]]] ==
         binclosed[composite[cross[x, y], TWIST]]

In[6]:= image[inverse[IMAGE[id[cart[V, V]]]], binclosed[composite[cross[x_, y_], TWIST]]] :=
         binclosed[composite[cross[x, y], TWIST]]
```

Theorem. A simplification rule.

```
In[7]:= image[inverse[IMAGE[SWAP]], binclosed[direct[x, y]]] // Normality

Out[7]= image[inverse[IMAGE[SWAP]], binclosed[composite[cross[x, y], TWIST]]] ==
         binclosed[composite[cross[y, x], TWIST]]

In[8]:= image[inverse[IMAGE[SWAP]], binclosed[composite[cross[x_, y_], TWIST]]] :=
         binclosed[composite[cross[y, x], TWIST]]
```

At this point both cleared flags can be reset.

```
In[9]:= cond = True; simplify = True;
```

Theorem.

```
In[10]:= ImageComp[IMAGE[id[cart[V, V]]], inverse[IMAGE[id[cart[V, V]]]],
           binclosed[composite[cross[x, y], TWIST]]] // Reverse

Out[10]= image[IMAGE[id[cart[V, V]]], binclosed[composite[cross[x, y], TWIST]]] ==
          intersection[binclosed[composite[cross[x, y], TWIST]], P[cart[V, V]]]

In[11]:= image[IMAGE[id[cart[V, V]]], binclosed[composite[cross[x_, y_], TWIST]]] :=
          intersection[binclosed[composite[cross[x, y], TWIST]], P[cart[V, V]]]
```

Theorem.

```
In[12]:= ImageComp[IMAGE[SWAP], inverse[IMAGE[SWAP]],
           binclosed[composite[cross[y, x], TWIST]]] // Reverse

Out[12]= image[IMAGE[SWAP], binclosed[composite[cross[x, y], TWIST]]] ==
          intersection[binclosed[composite[cross[y, x], TWIST]], P[cart[V, V]]]

In[13]:= image[IMAGE[SWAP], binclosed[composite[cross[x_, y_], TWIST]]] :=
          intersection[binclosed[composite[cross[y, x], TWIST]], P[cart[V, V]]]
```

---

an application to group theory

In the **GOEDEL** program, the terms **group** and **subgroup** both refer to composition laws. The underlying sets of these composition laws are their ranges. A basic fact in group theory is that a nonempty set is the range of a subgroup if it is binary closed and fixed under imaging with inversion. The first lemma applies this to the case of a direct product group. The constructor **gp[x]** can be either empty or a group.

Lemma.

```
In[14]:= SubstTest[intersection, binclosed[gp[t]],
           fix[IMAGE[inv[gp[t]]], t → direct[gp[x], gp[y]]] // Reverse

Out[14]= intersection[binclosed[composite[cross[gp[x], gp[y]], TWIST]],
           fix[IMAGE[cross[inv[gp[x]], inv[gp[y]]]]] == union[image[IMAGE[SECOND],
            intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]], set[0]]

In[15]:= intersection[binclosed[composite[cross[gp[x_], gp[y_]], TWIST]],
           fix[IMAGE[cross[inv[gp[x_]], inv[gp[y_]]]]] := union[image[IMAGE[SECOND],
            intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]], set[0]]
```

Lemma. Simplification rule.

```
In[18]:= SubstTest[intersection, complement[set[0]], image[IMAGE[SECOND],
             intersection[GROUPS, P[gp[t]]]], t → direct[gp[x], gp[y]]] // Reverse

Out[18]= intersection[complement[set[0]], image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]]] ==
          image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]]
```

```
In[19]:= intersection[complement[set[0]], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[gp[x_], gp[y_]], TWIST]]]]] :=
         image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]]
```

Theorem.

```
In[20]:= Map[intersection[P[cart[V, V]], #] &, SubstTest[intersection,
           image[inverse[IMAGE[SWAP]], u], image[inverse[IMAGE[SWAP]], v],
           image[inverse[IMAGE[SWAP]], w], {u → binclosed[direct[gp[x], gp[y]]],
            v → fix[IMAGE[inv[direct[gp[x], gp[y]]]]], w → complement[set[0]]}]]
```

```
Out[20]= image[INVERSE, image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]] ==
         image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[gp[y], gp[x]], TWIST]]]]
```

```
In[21]:= image[INVERSE, image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[gp[x_], gp[y_]], TWIST]]]] :=
         image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[gp[y], gp[x]], TWIST]]]]
```

Lemma. A simplification rule.

```
In[31]:= AssInt[dif[binclosed[direct[gp[x], gp[y]]], set[0]],
           fix[IMAGE[inv[direct[gp[x], gp[y]]]]], P[cart[V, V]]] // Reverse
```

```
Out[31]= intersection[image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]], P[cart[V, V]]] ==
         image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]]
```

```
In[32]:= intersection[image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[gp[x_], gp[y_]], TWIST]]]], P[cart[V, V]]] :=
         image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]]
```

Corollary. A similar result but with **IMAGE[SWAP]** in place of **INVERSE**.

```
In[37]:= ImageComp[IMAGE[SWAP], id[P[cart[V, V]]], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]]] // Reverse
```

```
Out[37]= image[IMAGE[SWAP], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[gp[x], gp[y]], TWIST]]]] ==
         image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[gp[y], gp[x]], TWIST]]]]
```

```
In[38]:= image[IMAGE[SWAP], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[gp[x_], gp[y_]], TWIST]]]] :=
         image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[gp[y], gp[x]], TWIST]]]]
```

A new variable **w** will now be introduced to help clarify the meaning of this result.

Theorem. If **w** is the range of a subgroup of a direct product of two groups, then **inverse[w]** is the range of a subgroup of the direct product of the same groups in the reverse order.

```
In[39]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → IMAGE[SWAP], u → set[w], v -> image[IMAGE[SECOND], intersection[GROUPS,
             P[composite[cross[gp[x], gp[y]], TWIST]]]]}] // Reverse // MapNotNot

Out[39]= or[member[inverse[w], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[gp[y], gp[x]], TWIST]]]],
         not[member[w, image[IMAGE[SECOND], intersection[GROUPS,
           P[composite[cross[gp[x], gp[y]], TWIST]]]]]] == True

In[40]:= or[member[inverse[w_], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[gp[y_], gp[x_]], TWIST]]]],
         not[member[w_, image[IMAGE[SECOND], intersection[GROUPS,
           P[composite[cross[gp[x_], gp[y_]], TWIST]]]]]] := True
```

Corollary. (Remove the **gp** wrappers.)

```
In[42]:= SubstTest[implies, and[equal[x, gp[u]], equal[y, gp[v]]],
           or[member[inverse[w], image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[y, x], TWIST]]]], not[member[w,
             image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[x, y], TWIST]]]]]],
           {u → x, v → y}] // Reverse // MapNotNot

Out[42]= or[member[inverse[w], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[y, x], TWIST]]]], not[member[w,
           image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[x, y], TWIST]]]]],
         not[member[x, GROUPS]], not[member[y, GROUPS]]] == True

In[44]:= or[member[inverse[w_], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[y_, x_], TWIST]]]], not[member[w_,
           image[IMAGE[SECOND], intersection[GROUPS, P[composite[cross[x_, y_], TWIST]]]]],
         not[member[x_, GROUPS]], not[member[y_, GROUPS]]] := True
```