inttimes[int[x]] is the range of a subgroup

Johan G. F. Belinfante
2011 December 17

```
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 17 at 2:16

        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 17 at 2:29
```

summary

The function **inttimes[int[x]]** and its inverse are both ranges of subgroups of the group **direct[INTADD, INTADD]**.

derivation

Theorem. A general result about **binhom** specialized to **INTADD**.

```
In[2]:= SubstTest[subclass, binhom[funpart[x], y],
          binclosed[direct[funpart[x], y]], {x → INTADD, y → INTADD}] // Reverse

Out[2]= subclass[binhom[INTADD, INTADD],
          binclosed[composite[cross[INTADD, INTADD], TWIST]]] == True

In[3]:= subclass[binhom[INTADD, INTADD],
          binclosed[composite[cross[INTADD, INTADD], TWIST]]] := True
```

Lemma. The function **inttimes[int[x]]** is binary closed under **direct[INTADD, INTADD]**.

```
In[4]:= SubstTest[implies, and[member[u, v], subclass[v, w]],
          member[u, w], {u → inttimes[int[x]], v → binhom[INTADD, INTADD],
           w → binclosed[direct[INTADD, INTADD]]}] // Reverse

Out[4]= subclass[composite[INTADD, cross[inttimes[int[x]], inttimes[int[x]]], inverse[INTADD]],
          inttimes[int[x]]] == True
```

```
In[5]:= (% /. x → x_) /. Equal → SetDelayed
```

The above inclusion can be strengthened to an equation and made into a rewrite rule. One can also eliminate the **int** wrapper.

Lemma. An equation for **inttimes[int[x]]**.

```
In[6]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
          equal[u, composite[v, id[domain[u]]]],
          {u -> composite[INTADD, cross[inttimes[int[x]], inttimes[int[x]]], inverse[INTADD]],
           v -> inttimes[int[x]]}] // Reverse

Out[6]= equal[composite[INTADD, cross[inttimes[int[x]], inttimes[int[x]]], inverse[INTADD]],
          inttimes[int[x]]] == True

In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. (Eliminate the **int** wrapper.)

```
In[8]:= SubstTest[implies, equal[x, int[t]],
          equal[composite[INTADD, cross[inttimes[x], inttimes[x]], inverse[INTADD]],
           inttimes[x]], t → x] // Reverse

Out[8]= or[equal[composite[INTADD, cross[inttimes[x], inttimes[x]], inverse[INTADD]],
           inttimes[x]], not[member[x, Z]]] == True

In[9]:= (% /. x → x_) /. Equal → SetDelayed
```

This still has a redundant literal which will now be removed.

Lemma.

```
In[10]:= SubstTest[implies, equal[t, 0],
           equal[composite[INTADD, cross[t, t], inverse[INTADD]], t], t → inttimes[x]] // Reverse

Out[10]= or[equal[composite[INTADD, cross[inttimes[x], inttimes[x]], inverse[INTADD]],
            inttimes[x]], member[x, Z]] == True

In[11]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. A better rewrite rule.

```
In[12]:= SubstTest[and, implies[p, q], or[p, q],
          {p → member[x, Z], q -> equal[composite[INTADD, cross[inttimes[x], inttimes[x]],
              inverse[INTADD]], inttimes[x]]}]

Out[12]= equal[composite[INTADD, cross[inttimes[x], inttimes[x]], inverse[INTADD]],
           inttimes[x]] == True

In[13]:= composite[INTADD, cross[inttimes[x_], inttimes[x_]], inverse[INTADD]] := inttimes[x]
```

Corollary. A similar result holds for **inverse[inttimes[x]]**.

```
In[14]:= composite[INTADD, inverse[cross[inttimes[x], inttimes[x]]], inverse[INTADD]] //
          DoubleInverse

Out[14]= composite[INTADD, cross[inverse[inttimes[x]], inverse[inttimes[x]]],
           inverse[INTADD]] == inverse[inttimes[x]]

In[15]:= composite[INTADD, cross[inverse[inttimes[x_]], inverse[inttimes[x_]]],
           inverse[INTADD]] := inverse[inttimes[x]]
```

Lemma.

```
In[16]:= SubstTest[intersection, binclosed[gp[t]],
           fix[IMAGE[inv[gp[t]]], t → direct[INTADD, INTADD]] // Reverse

Out[16]= intersection[binclosed[composite[cross[INTADD, INTADD], TWIST]],
           fix[IMAGE[cross[composite[id[Z], INVERSE], composite[id[Z], INVERSE]]]]] ==
          union[image[IMAGE[SECOND],
            intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]], set[0]]

In[17]:= intersection[binclosed[composite[cross[INTADD, INTADD], TWIST]],
           fix[IMAGE[cross[composite[id[Z], INVERSE], composite[id[Z], INVERSE]]]]] :=
          union[image[IMAGE[SECOND], intersection[GROUPS,
            P[composite[cross[INTADD, INTADD], TWIST]]], set[0]]
```

Main Theorem. The function **inttimes[int[x]]** is the range of a subgroup of **direct[INTADD, INTADD]**.

```
In[18]:= SubstTest[member, inttimes[int[x]], intersection[u, v],
           {u -> binclosed[composite[cross[INTADD, INTADD], TWIST]], v -> fix[
              IMAGE[cross[composite[id[Z], INVERSE], composite[id[Z], INVERSE]]]]}] // Reverse

Out[18]= member[inttimes[int[x]], image[IMAGE[SECOND],
            intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] == True

In[19]:= member[inttimes[int[x_]], image[IMAGE[SECOND],
            intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] := True
```

Lemma. The inverse of **inttimes[x]** is fixed by imaging with the inversion function for **direct[INTADD, INTADD]**.

```
In[21]:= composite[INVERSE, inverse[inttimes[x]], INVERSE] // DoubleInverse

Out[21]= composite[INVERSE, inverse[inttimes[x]], INVERSE] == inverse[inttimes[x]]

In[22]:= composite[INVERSE, inverse[inttimes[x_]], INVERSE] := inverse[inttimes[x]]
```

Theorem. The function **inverse[inttimes[int[x]]]** is the range of a subgroup of **direct[INTADD, INTADD]**.

```
In[23]:= SubstTest[member, inverse[inttimes[int[x]]], intersection[u, v],
           {u -> binclosed[composite[cross[INTADD, INTADD], TWIST]], v -> fix[
              IMAGE[cross[composite[id[Z], INVERSE], composite[id[Z], INVERSE]]]]}] // Reverse

Out[23]= member[inverse[inttimes[int[x]]], image[IMAGE[SECOND],
            intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] == True
```

```
In[24]:= member[inverse[inttimes[int[x_]]], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] := True
```

wrapper-free and variable-free restatements

Lemma. Eliminating the **int** wrapper.

```
In[25]:= SubstTest[implies, equal[x, int[t]],
           member[inttimes[x], image[IMAGE[SECOND], intersection[GROUPS,
             P[composite[cross[INTADD, INTADD], TWIST]]]]], t → x] // Reverse

Out[25]= or[member[inttimes[x], image[IMAGE[SECOND], intersection[GROUPS,
             P[composite[cross[INTADD, INTADD], TWIST]]]]], not[member[x, Z]]] == True

In[26]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma. A converse result.

```
In[27]:= SubstTest[implies, member[t, image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]],
           not[empty[t]], t → inttimes[x]] // Reverse

Out[27]= or[member[x, Z], not[member[inttimes[x], image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]]]] == True

In[28]:= (% /. x → x_) /. Equal→ SetDelayed
```

The above two results can be combined into a single rewrite rule.

Theorem. A wrapper-free restatement of the main theorem.

```
In[29]:= equiv[member[inttimes[x], image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]], member[x, Z]]

Out[29]= True

In[30]:= member[inttimes[x_], image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] := member[x, Z]
```

The following is a variable-free restatement of the main theorem.

Theorem. The class **binhom[INTADD, INTADD]** is a subclass of the class of ranges of subgroups of **direct[INTADD, INTADD]**.

```
In[31]:= Map[equal[V, domain[#]] &,
           SubstTest[reify, x, case[implies[member[x, Z], member[inttimes[x], t]]], t -> image[
             IMAGE[SECOND], intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]]]

Out[31]= subclass[binhom[INTADD, INTADD], image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] == True
```

```
In[32]:= subclass[binhom[INTADD, INTADD], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] := True
```

Lemma.

```
In[33]:= SubstTest[implies, equal[x, int[t]],
           member[inverse[inttimes[x]], image[IMAGE[SECOND], intersection[
             GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]], t → x] // Reverse
```

```
Out[33]= or[member[inverse[inttimes[x]], image[IMAGE[SECOND], intersection[GROUPS,
             P[composite[cross[INTADD, INTADD], TWIST]]]], not[member[x, Z]]] == True
```

```
In[34]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. The same holds for **image[INVERSE, binhom[INTADD,INTADD]]**.

```
In[35]:= Map[equal[V, domain[#]] &, SubstTest[reify, x,
           case[implies[member[x, Z], member[inverse[inttimes[x]], t]], t -> image[
             IMAGE[SECOND], intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]]]
```

```
Out[35]= subclass[image[INVERSE, binhom[INTADD, INTADD]], image[IMAGE[SECOND],
             intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] == True
```

```
In[36]:= subclass[image[INVERSE, binhom[INTADD, INTADD]], image[IMAGE[SECOND],
           intersection[GROUPS, P[composite[cross[INTADD, INTADD], TWIST]]]]] := True
```