certain restricted divisibility relations are transitive

Johan G. F. Belinfante
2004 December 10

```
In[1]:= SetDirectory["i:"]; << goedel64.07a; << tools.m

        :Package Title: goedel64.07a      2004 December 7 at 12:10 noon

        It is now:  2004 Dec 10 at 18:15

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 November 17

        weightlimit = 40
```

summary

If **x** is an associative relation, and **y** is binary-closed under **x**, then certain divisibility relations of **x** restricted to **y** are transitive.

FIRST rule

```
In[2]:= SubstTest[implies, equal[u, v], equal[composite[u, w], composite[v, w]],
          {u -> composite[x, cross[x, Id]], v → composite[x, cross[Id, x], ASSOC], w →
            composite[id[cart[V, z]], inverse[FIRST], id[cart[V, y]], inverse[FIRST]]}]

Out[2]= or[equal[composite[x, id[cart[V, image[x, cart[y, z]]]], inverse[FIRST]],
            composite[x, id[cart[V, z]], inverse[FIRST],
             x, id[cart[V, y]], inverse[FIRST]]], not[
            equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]] == True

In[3]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed

In[4]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], not[implies[p1, p3]], {p1 → associative[x],
            p2 -> equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]],
            p3 -> equal[composite[x, id[cart[V, image[x, cart[y, z]]]], inverse[FIRST]],
              composite[x, id[cart[V, z]], inverse[FIRST],
               x, id[cart[V, y]], inverse[FIRST]]]}]]

Out[4]= or[equal[composite[x, id[cart[V, image[x, cart[y, z]]]], inverse[FIRST]],
            composite[x, id[cart[V, z]], inverse[FIRST], x,
             id[cart[V, y]], inverse[FIRST]]], not[associative[x]]] == True
```

```
In[5]:= or[equal[composite[x_, id[cart[V, image[x_, cart[y_, z_]]]], inverse[FIRST]],
           composite[x_, id[cart[V, z_]], inverse[FIRST], x_,
            id[cart[V, y_]], inverse[FIRST]]], not[associative[x_]]] := True
```

Lemma.

```
In[6]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
         w -> composite[intersection[composite[inverse[FIRST], inverse[y]],
            composite[inverse[SECOND], x]], inverse[SECOND]]]

Out[6]= or[not[subclass[u, v]], subclass[
           composite[x, id[cart[V, u]], y], composite[x, id[cart[V, v]], y]]] == True

In[7]:= or[not[subclass[u_, v_]], subclass[composite[x_, id[cart[V, u_]], y_],
           composite[x_, id[cart[V, v_]], y_]]] := True
```

Main result.

```
In[8]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
          implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
          {p1 -> associative[x], p2 -> subclass[image[x, cart[y, y]], y],
           p3 -> equal[composite[x, id[cart[V, image[x, cart[y, y]]]], inverse[FIRST]],
             composite[x, id[cart[V, y]], inverse[FIRST],
              x, id[cart[V, y]], inverse[FIRST]]],
           p4 → subclass[composite[x, id[cart[V, image[x, cart[y, y]]]],
              inverse[FIRST]], composite[x, id[cart[V, y]], inverse[FIRST]]],
           p5 -> TRANSITIVE[composite[x, id[cart[V, y]], inverse[FIRST]]]}]]

Out[8]= or[not[associative[x]], not[subclass[image[x, cart[y, y]], y]],
          TRANSITIVE[composite[x, id[cart[V, y]], inverse[FIRST]]]] == True

In[9]:= or[not[associative[x_]], not[subclass[image[x_, cart[y_, y_]], y_]],
          TRANSITIVE[composite[x_, id[cart[V, y_]], inverse[FIRST]]]] := True
```

---

example: FIRST rule for COMPOSE

Some examples are derived in this section. It is easy enough to derive the examples directly without using the rules developed in the preceding section. In fact, the examples had been derived first, and the general theory was developed later.

```
In[10]:= Assoc[composite[COMPOSE, cross[Id, COMPOSE]], ASSOC, composite[id[cart[V, x]],
           inverse[FIRST], id[cart[V, y]], inverse[FIRST]]] // Reverse

Out[10]= composite[COMPOSE, id[cart[V, x]],
           inverse[FIRST], COMPOSE, id[cart[V, y]], inverse[FIRST]] ==
         composite[COMPOSE, id[cart[V, image[COMPOSE, cart[y, x]]]], inverse[FIRST]]
```

```
In[11]:= composite[COMPOSE, id[cart[V, x_]],
           inverse[FIRST], COMPOSE, id[cart[V, y_]], inverse[FIRST]] :=
         composite[COMPOSE, id[cart[V, image[COMPOSE, cart[y, x]]]], inverse[FIRST]]

In[12]:= SubstTest[subclass, composite[w, w], w,
           w -> composite[COMPOSE, id[cart[V, FUNS]], inverse[FIRST]]] // Reverse

Out[12]= TRANSITIVE[composite[COMPOSE, id[cart[V, FUNS]], inverse[FIRST]]] == True

In[13]:= TRANSITIVE[composite[COMPOSE, id[cart[V, FUNS]], inverse[FIRST]]] := True

In[14]:= SubstTest[subclass, composite[w, w], w,
           w -> composite[COMPOSE, id[cart[V, BIJ]], inverse[FIRST]]] // Reverse

Out[14]= TRANSITIVE[composite[COMPOSE, id[cart[V, BIJ]], inverse[FIRST]]] == True

In[15]:= TRANSITIVE[composite[COMPOSE, id[cart[V, BIJ]], inverse[FIRST]]] := True
```

SECOND rule

This section is exactly analogous to the section for the **FIRST** rule.

```
In[16]:= SubstTest[implies, equal[u, v], equal[composite[u, w], composite[v, w]],
          {u -> composite[x, cross[Id, x]],
           v → composite[x, cross[x, Id], inverse[ASSOC]], w → composite[
             id[cart[y, V]], inverse[SECOND], id[cart[z, V]], inverse[SECOND]]}]

Out[16]= or[equal[composite[x, id[cart[image[x, cart[y, z]], V]], inverse[SECOND]],
             composite[x, id[cart[y, V]], inverse[SECOND], x, id[cart[z, V]],
              inverse[SECOND]]], not[equal[composite[x, cross[Id, x]],
              composite[x, cross[x, Id], inverse[ASSOC]]]]] == True

In[17]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed

In[18]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → associative[x], p2 -> equal[
              composite[x, cross[x, Id], inverse[ASSOC]], composite[x, cross[Id, x]]],
            p3 -> equal[composite[x, id[cart[image[x, cart[y, z]], V]],
               inverse[SECOND]], composite[x, id[cart[y, V]],
               inverse[SECOND], x, id[cart[z, V]], inverse[SECOND]]]}]]

Out[18]= or[equal[composite[x, id[cart[image[x, cart[y, z]], V]], inverse[SECOND]],
             composite[x, id[cart[y, V]], inverse[SECOND], x,
              id[cart[z, V]], inverse[SECOND]]], not[associative[x]]] == True

In[19]:= or[equal[composite[x_, id[cart[image[x_, cart[y_, z_]], V]], inverse[SECOND]],
             composite[x_, id[cart[y_, V]], inverse[SECOND], x_,
              id[cart[z_, V]], inverse[SECOND]]], not[associative[x_]]] := True
```

Lemma.

```
In[20]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          w -> composite[intersection[composite[inverse[FIRST], inverse[y]],
             composite[inverse[SECOND], x]], inverse[FIRST]]]

Out[20]= or[not[subclass[u, v]], subclass[
            composite[x, id[cart[u, V]], y], composite[x, id[cart[v, V]], y]]] == True

In[21]:= or[not[subclass[u_, v_]], subclass[composite[x_, id[cart[u_, V]], y_],
            composite[x_, id[cart[v_, V]], y_]]] := True

In[22]:= Map[not, SubstTest[and, implies[p1, p3], implies[p2, p4],
           implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 -> associative[x], p2 -> subclass[image[x, cart[y, y]], y],
            p3 -> equal[composite[x, id[cart[image[x, cart[y, y]], V]],
               inverse[SECOND]], composite[x, id[cart[y, V]],
               inverse[SECOND], x, id[cart[y, V]], inverse[SECOND]]],
            p4 → subclass[composite[x, id[cart[image[x, cart[y, y]], V]],
               inverse[SECOND]], composite[x, id[cart[y, V]], inverse[SECOND]]],
            p5 -> TRANSITIVE[composite[x, id[cart[y, V]], inverse[SECOND]]]}]]

Out[22]= or[not[associative[x]], not[subclass[image[x, cart[y, y]], y]],
           TRANSITIVE[composite[x, id[cart[y, V]], inverse[SECOND]]]] == True

In[23]:= or[not[associative[x_]], not[subclass[image[x_, cart[y_, y_]], y_]],
           TRANSITIVE[composite[x_, id[cart[y_, V]], inverse[SECOND]]]] := True
```

---

example: SECOND rule for COMPOSE

```
In[24]:= Assoc[composite[COMPOSE, cross[Id, COMPOSE]],
           ASSOC, inverse[ASSOC]] // Reverse

Out[24]= composite[COMPOSE, cross[COMPOSE, Id], inverse[ASSOC]] ==
          composite[COMPOSE, cross[Id, COMPOSE]]

In[25]:= composite[COMPOSE, cross[COMPOSE, Id], inverse[ASSOC]] :=
          composite[COMPOSE, cross[Id, COMPOSE]]

In[26]:= Assoc[composite[COMPOSE, cross[COMPOSE, Id]],
           inverse[ASSOC], composite[id[cart[x, V]], inverse[SECOND],
            id[cart[y, V]], inverse[SECOND]]] // Reverse

Out[26]= composite[COMPOSE, id[cart[x, V]],
           inverse[SECOND], COMPOSE, id[cart[y, V]], inverse[SECOND]] ==
          composite[COMPOSE, id[cart[image[COMPOSE, cart[x, y]], V]], inverse[SECOND]]
```

```
In[27]:= composite[COMPOSE, id[cart[x_, V]],
           inverse[SECOND], COMPOSE, id[cart[y_, V]], inverse[SECOND]] :=
         composite[COMPOSE, id[cart[image[COMPOSE, cart[x, y]], V]], inverse[SECOND]]

In[28]:= SubstTest[subclass, composite[w, w], w,
           w -> composite[COMPOSE, id[cart[FUNS, V]], inverse[SECOND]]] // Reverse

Out[28]= TRANSITIVE[composite[COMPOSE, id[cart[FUNS, V]], inverse[SECOND]]] == True

In[29]:= TRANSITIVE[composite[COMPOSE, id[cart[FUNS, V]], inverse[SECOND]]] := True

In[30]:= SubstTest[subclass, composite[w, w], w,
           w -> composite[COMPOSE, id[cart[BIJ, V]], inverse[SECOND]]] // Reverse

Out[30]= TRANSITIVE[composite[COMPOSE, id[cart[BIJ, V]], inverse[SECOND]]] == True

In[31]:= TRANSITIVE[composite[COMPOSE, id[cart[BIJ, V]], inverse[SECOND]]] := True
```