multiplication distributes over divisibility

Johan G. F. Belinfante
2005 June 29

```
In[1]:= SetDirectory["i:"]; << goedel70.28a; << tools.m

        :Package Title: goedel70.28a       2005 June 28 at 10:30 p.m.

        It is now:  2005 Jun 29 at 15:59

        Loading Simplification Rules

        TOOLS.M                      Revised 2005 June 17

        weightlimit = 40
```

summary

It is shown in this notebook that multiplication of natural numbers distributes over divisibility: if **y | z**, then **(x y) | (x z)**. This result is derived as a corollary of a general commutativity theorem valid for an arbitrary associative relation.

associative relations

A ternary relation is **associative** if it satisfies **equal[composite[x,cross[Id,x],ASSOC], composite[x,cross[x,Id]]]**.

```
In[2]:= equiv[associative[x], and[subclass[x, cart[cart[V, V], V]], equal[composite[x,
              cross[Id, x], ASSOC], composite[x, cross[x, Id]]]]] // not // not

Out[2]= True
```

Affiliated with each associative relation are two transitive relations of left and right divisibility.

```
In[3]:= implies[associative[x], TRANSITIVE[composite[x, inverse[FIRST]]]]

Out[3]= True

In[4]:= implies[associative[x], TRANSITIVE[composite[x, inverse[SECOND]]]]

Out[4]= True
```

It will be shown below that, for any associative relation, left-composition commutes with right composition. Also, the two affiliated left and right divisibility relations commute with each other, and each divisibility relation commutes with one-sided composition.

---

a basic commutativity theorem

The associative law can be viewed as a statement that every left-composition commutes with every right-composition.

```
In[5]:= 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 -> commute[composite[x, LEFT[y]], composite[x, RIGHT[z]]]}]]

Out[5]= or[equal[composite[x, LEFT[y], x, RIGHT[z]],
           composite[x, RIGHT[z], x, LEFT[y]]], not[associative[x]]] == True

In[6]:= or[equal[composite[x_, LEFT[y_], x_, RIGHT[z_]],
           composite[x_, RIGHT[z_], x_, LEFT[y_]]], not[associative[x_]]] := True
```

Restatement:

```
In[26]:= implies[associative[x],
          commute[composite[x, LEFT[y]], composite[x, RIGHT[y]]]]

Out[26]= True
```

---

left-divisibility commutes with right-divisibility

For any associative relation, left-divisibility commutes with right-divisibility.

```
In[7]:= SubstTest[implies, equal[u, v], equal[composite[u, w], composite[v, w]],
         {u -> composite[x, cross[Id, x], ASSOC], v -> composite[x, cross[x, Id]],
          w → composite[inverse[FIRST], inverse[SECOND]]}]

Out[7]= or[equal[composite[x, inverse[FIRST], x, inverse[SECOND]],
           composite[x, inverse[SECOND], x, inverse[FIRST]]], not[
           equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]] == True

In[8]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[9]:= 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 ->
            commute[composite[x, inverse[FIRST]], composite[x, inverse[SECOND]]]}]]

Out[9]= or[equal[composite[x, inverse[FIRST], x, inverse[SECOND]], composite[x,
            inverse[SECOND], x, inverse[FIRST]]], not[associative[x]]] == True

In[10]:= or[equal[composite[x_, inverse[FIRST], x_, inverse[SECOND]], composite[x_,
             inverse[SECOND], x_, inverse[FIRST]]], not[associative[x_]]] := True
```

Restatement.

```
In[11]:= implies[associative[x],
          commute[composite[x, inverse[FIRST]], composite[x, inverse[SECOND]]]]

Out[11]= True
```

---

divisibility commutes with composition on one side

Lemma.

```
In[12]:= SubstTest[implies, equal[u, v], equal[composite[u, w], composite[v, w]],
          {u -> composite[x, cross[Id, x], ASSOC],
           v -> composite[x, cross[x, Id]], w → composite[RIGHT[y], inverse[SECOND]]}]

Out[12]= or[equal[composite[x, inverse[SECOND], x, RIGHT[y]],
            composite[x, RIGHT[y], x, inverse[SECOND]]], not[equal[
             composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]] == True

In[13]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[14]:= 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 -> commute[composite[x, RIGHT[y]], composite[x, inverse[SECOND]]]}]]

Out[14]= or[equal[composite[x, inverse[SECOND], x, RIGHT[y]],
            composite[x, RIGHT[y], x, inverse[SECOND]]], not[associative[x]]] == True

In[15]:= or[equal[composite[x_, inverse[SECOND], x_, RIGHT[y_]], composite[x_,
             RIGHT[y_], x, inverse[SECOND]]], not[associative[x_]]] := True
```

Similarly:

```
In[16]:= SubstTest[implies, equal[u, v], equal[composite[u, w], composite[v, w]],
          {u -> composite[x, cross[Id, x], ASSOC],
           v -> composite[x, cross[x, Id]], w → composite[inverse[FIRST], LEFT[y]]}]

Out[16]= or[equal[composite[x, inverse[FIRST], x, LEFT[y]],
            composite[x, LEFT[y], x, inverse[FIRST]]], not[equal[
             composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]] == True

In[17]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
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]], composite[x, cross[Id, x], ASSOC]],
            p3 → commute[composite[x, inverse[FIRST]], composite[x, LEFT[y]]]}]]

Out[18]= or[equal[composite[x, inverse[FIRST], x, LEFT[y]],
            composite[x, LEFT[y], x, inverse[FIRST]]], not[associative[x]]] == True

In[19]:= or[equal[composite[x_, inverse[FIRST], x_, LEFT[y_]],
            composite[x_, LEFT[y_], x_, inverse[FIRST]]], not[associative[x_]]] := True
```

---

multiplication distributes over divisibility

As an application of the commutative laws about associative relations, one finds the following rule about divisibility.

```
In[20]:= SubstTest[implies, associative[y],
          commute[composite[y, inverse[FIRST]], composite[y, LEFT[x]]], y → NATMUL]

Out[20]= equal[composite[DIV, NATMUL, LEFT[x]], composite[NATMUL, LEFT[x], DIV]] == True
```

This fact can be formulated as a rewrite rule. The orientation of the rewrite rule is suggested by applying the rule of thumb that the composite of a relation and a function behaves better when the function occurs on the right. Strictly speaking, this rule of thumb is not compelling when the function is one-to-one. The function **composite[NATMUL, LEFT[x]]** is one-to-one unless **x** is zero. The chosen orientation has the advantage that it does not make an exception of the case **x = 0**.

```
In[21]:= composite[NATMUL, LEFT[x_], DIV] := composite[DIV, NATMUL, LEFT[x]]
```

Corollary.

```
In[22]:= SubstTest[implies, subclass[u, v],
           subclass[image[w, u], image[w, v]], {u → cart[set[x], set[y]],
            v → DIV, w -> cross[composite[NATMUL, LEFT[nat[z]]],
              composite[NATMUL, LEFT[nat[z]]]]}] // MapNotNot

Out[22]= or[member[pair[natmul[x, nat[z]], natmul[y, nat[z]]], DIV],
           not[member[pair[x, y], DIV]]] == True

In[23]:= or[member[pair[natmul[x_, nat[z_]], natmul[y_, nat[z_]]], DIV],
           not[member[pair[x_, y_], DIV]]] := True
```

The **nat** wrapper can be removed.

```
In[24]:= SubstTest[implies, and[equal[w, nat[z]], member[pair[x, y], DIV]],
          member[pair[natmul[w, x], natmul[w, y]], DIV], w → z]

Out[24]= or[member[pair[natmul[x, z], natmul[y, z]], DIV],
           not[member[z, omega]], not[member[pair[x, y], DIV]]] == True

In[25]:= or[member[pair[natmul[x_, z_], natmul[y_, z_]], DIV],
           not[member[z_, omega]], not[member[pair[x_, y_], DIV]]] := True
```