rotated associative relation

Johan G. F. Belinfante
2005 July 30

```
In[1]:= SetDirectory["i:"]; << goedel71.25a; << tools.m

        :Package Title: goedel71.24a          2005 July 24 at 6:35 p.m.

        It is now:  2005 Jul 30 at 9:48

        Loading Simplification Rules

        TOOLS.M                      Revised 2005 July 18

        weightlimit = 40
```

summary

Addition is related to subtraction by rotation. The associative law for addition implies a corresponding law for subtraction: **(x – y) – z = x – (y + z)**. In this notebook it is shown that a similar result holds for any associative relation.

rotate and ASSOC

Lemma.

```
In[12]:= Assoc[rotate[inverse[rotate[x]]],
           id[composite[FIRST, rotate[x]]], id[cart[cart[V, V], V]]] // Reverse

Out[12]= composite[rotate[inverse[rotate[x]]], id[cart[cart[V, V], V]]] ==
          rotate[inverse[rotate[x]]]

In[14]:= composite[rotate[inverse[rotate[x_]]], id[cart[cart[V, V], V]]] :=
          rotate[inverse[rotate[x]]]
```

Theorem.

```
In[15]:= composite[rotate[composite[x, ASSOC]], ASSOC] // VSTerNormality

Out[15]= composite[rotate[composite[x, ASSOC]], ASSOC] == rotate[inverse[rotate[x]]]

In[16]:= composite[rotate[composite[x_, ASSOC]], ASSOC] := rotate[inverse[rotate[x]]]
```

main theorem

Lemma.

```
In[23]:= composite[rotate[y], cross[x, Id]] // TripleRotate // Reverse

Out[23]= rotate[composite[inverse[x], y]] == composite[rotate[y], cross[x, Id]]

In[24]:= rotate[composite[inverse[x_], y_]] := composite[rotate[y], cross[x, Id]]
```

The following observation is used in the next step.

```
In[17]:= abstract[x, composite[rotate[x], ASSOC]]

Out[17]= composite[cross[inverse[ASSOC], Id], ROT]
```

Theorem.

```
In[25]:= SubstTest[implies, equal[u, v], equal[image[w, u], image[w, v]],
          {u → composite[x, cross[x, Id]], v → composite[x, cross[Id, x], ASSOC],
           w -> composite[cross[inverse[ASSOC], Id], ROT]}]

Out[25]= or[equal[composite[rotate[x], cross[rotate[x], Id]],
            composite[rotate[x], cross[Id, x], ASSOC]], not[equal[
             composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]] == True

In[26]:= or[equal[composite[rotate[x_], cross[rotate[x_], Id]],
            composite[rotate[x_], cross[Id, x_], ASSOC]],
          not[equal[composite[x_, cross[x_, Id]],
             composite[x_, cross[Id, x_], ASSOC]]]] := True
```

Corollary.

```
In[27]:= 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[rotate[x], cross[rotate[x], Id]],
             composite[rotate[x], cross[Id, x], ASSOC]]}]]

Out[27]= or[equal[composite[rotate[x], cross[rotate[x], Id]],
            composite[rotate[x], cross[Id, x], ASSOC]], not[associative[x]]] == True

In[28]:= or[equal[composite[rotate[x_], cross[rotate[x_], Id]],
            composite[rotate[x_], cross[Id, x_], ASSOC]], not[associative[x_]]] := True
```

serendipity

An amusing application is presented in this section.

```
In[29]:= SubstTest[implies, associative[x],
          equal[composite[rotate[x], cross[rotate[x], Id]],
           composite[rotate[x], cross[Id, x], ASSOC]], x → inverse[DUP]]

Out[29]= equal[composite[SECOND, id[inverse[DUP]]],
           composite[FIRST, FIRST, id[inverse[DUP]]]] == True

In[30]:= composite[FIRST, FIRST, id[inverse[DUP]]] :=
          composite[SECOND, id[inverse[DUP]]]
```

The following formula is analogous.

```
In[31]:= composite[SECOND, id[inverse[DUP]]] // TripleRotate // Reverse

Out[31]= composite[SECOND, FIRST, id[inverse[DUP]]] ==
          composite[SECOND, id[inverse[DUP]]]

In[32]:= composite[SECOND, FIRST, id[inverse[DUP]]] :=
          composite[SECOND, id[inverse[DUP]]]
```

Two further corollaries:

```
In[33]:= Assoc[composite[FIRST, FIRST], id[inverse[DUP]], SWAP]

Out[33]= composite[FIRST, SECOND, id[DUP]] == composite[FIRST, id[DUP]]

In[34]:= composite[FIRST, SECOND, id[DUP]] := composite[FIRST, id[DUP]]

In[35]:= Assoc[composite[SECOND, FIRST], id[inverse[DUP]], SWAP]

Out[35]= composite[SECOND, SECOND, id[DUP]] == composite[FIRST, id[DUP]]

In[36]:= composite[SECOND, SECOND, id[DUP]] := composite[FIRST, id[DUP]]
```