TIMES

Johan G. F. Belinfante
2006 September 23

```
In[1]:= SetDirectory["l:"]; << goedel85.21a; << tools.m

        :Package Title: goedel85.21a          2006 September 21 at 11:00 a.m.

        It is now:  2006 Sep 23 at 13:5

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 August 22

        weightlimit = 40
```

summary

The function **TIMES = lambda[x, times[x]]** is defined, and some of its properties are derived.

definition of TIMES

A membership rule is used to define **TIMES**:

```
In[2]:= member[x_, TIMES] := and[equal[times[first[x]], second[x]], member[first[x], omega]]
```

An explicit formula for **TIMES** can be derived using **Normality** and **symdif**:

```
In[3]:= (symdif[TIMES, composite[IMAGE[SWAP], VERTSECT[inverse[rotate[NATMUL]]], id[omega]]] //
          Normality) /. Equal → SetDelayed

In[4]:= SubstTest[empty, symdif[u, v],
         {u -> TIMES, v -> composite[IMAGE[SWAP], VERTSECT[inverse[rotate[NATMUL]]], id[omega]]}]

Out[4]= True ==
         equal[TIMES, composite[IMAGE[SWAP], VERTSECT[inverse[rotate[NATMUL]]], id[omega]]]

In[5]:= composite[IMAGE[SWAP], VERTSECT[inverse[rotate[NATMUL]]], id[omega]] := TIMES
```

basic properties

The formula for **TIMES** can be used to derive many of its properties. It is a function:

```
In[6]:= SubstTest[FUNCTION,
          composite[IMAGE[SWAP], VERTSECT[inverse[rotate[NATMUL]]], id[w]], w → omega]
Out[6]= FUNCTION[TIMES] == True

In[7]:= FUNCTION[TIMES] := True
```

The domain of **TIMES** is the set of natural numbers:

```
In[8]:= IminComp[composite[IMAGE[SWAP], VERTSECT[inverse[rotate[NATMUL]]]], id[omega], V]
Out[8]= domain[TIMES] == omega

In[9]:= domain[TIMES] := omega
```

Lemma. The set **binhom[NATADD,NATADD]** is the set of all **times[x]** functions.

```
In[10]:= ImageComp[composite[IMAGE[SWAP], VERTSECT[inverse[rotate[NATMUL]]]], id[omega], V]
Out[10]= range[TIMES] == binhom[NATADD, NATADD]

In[11]:= range[TIMES] := binhom[NATADD, NATADD]
```

Corollary. The function **TIMES** is a set.

```
In[12]:= member[TIMES, V] // AssertTest
Out[12]= member[TIMES, V] == True

In[13]:= member[TIMES, V] := True
```

The above facts can be combined into a succinct statement:

```
In[14]:= member[TIMES, map[omega, binhom[NATADD, NATADD]]] // AssertTest
Out[14]= member[TIMES, map[omega, binhom[NATADD, NATADD]]] == True

In[15]:= member[TIMES, map[omega, binhom[NATADD, NATADD]]] := True
```

The following fact can be reformulated without variables:

```
In[16]:= equal[times[nat[x]], times[nat[y]]]
Out[16]= equal[nat[x], nat[y]]
```

Theorem. The function **TIMES** is one-to-one:

```
In[17]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[x, y],
           implies[and[member[x, t], member[y, t], equal[second[x], second[y]]],
            equal[first[x], first[y]]], t → TIMES]] // Reverse
Out[17]= FUNCTION[inverse[TIMES]] == True

In[18]:= FUNCTION[inverse[TIMES]] := True
```

APPLY rule

To facilitate the use of **VSNormality** for deriving facts about **TIMES**, the following vertical section rule may be useful.

```
In[19]:= ImageComp[
          composite[IMAGE[SWAP], VERTSECT[inverse[rotate[NATMUL]]]], id[omega], set[x]]

Out[19]= image[TIMES, set[x]] ==
          intersection[image[V, intersection[omega, set[x]]], set[times[x]]]

In[20]:= image[TIMES, set[x_]] :=
          intersection[image[V, intersection[omega, set[x]]], set[times[x]]]
```

A closely related formula is this **APPLY** rule:

```
In[21]:= SubstTest[A, image[w, set[x]], w → TIMES] // Reverse

Out[21]= APPLY[TIMES, x] == union[complement[image[V, intersection[omega, set[x]]]], times[x]]

In[22]:= APPLY[TIMES, x_] := union[complement[image[V, intersection[omega, set[x]]]], times[x]]
```

In practice, the use of **nat** wrappers helps to simplify this result:

```
In[23]:= APPLY[TIMES, nat[x]]

Out[23]= times[nat[x]]
```

TIMES as a binary homomorphism

Lemma.

```
In[24]:= dif[composite[TIMES, NATMUL],
           composite[COMPOSE, cross[TIMES, TIMES]]] // VSTriNormality

Out[24]= composite[intersection[composite[TIMES, NATMUL],
            composite[complement[COMPOSE], cross[TIMES, TIMES]]], id[cart[V, V]]] == 0

In[25]:= % /. Equal → SetDelayed
```

This yields an inclusion.

```
In[26]:= SubstTest[empty, composite[dif[u, v], id[cart[V, V]]],
           {v -> composite[COMPOSE, cross[TIMES, TIMES]],
            u -> composite[TIMES, NATMUL]}] // Reverse

Out[26]= subclass[composite[TIMES, NATMUL], composite[COMPOSE, cross[TIMES, TIMES]]] == True

In[27]:= % /. Equal → SetDelayed
```

Since any subclass of a function is a restriction, an equation can be derived:

```
In[28]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
          equal[u, composite[v, id[domain[u]]]],
          {u -> composite[TIMES, NATMUL], v -> composite[COMPOSE, cross[TIMES, TIMES]]}]

Out[28]= equal[composite[COMPOSE, cross[TIMES, TIMES]], composite[TIMES, NATMUL]] == True

In[29]:= composite[TIMES, NATMUL] := composite[COMPOSE, cross[TIMES, TIMES]]
```

Reformulation: The function **TIMES** is a binary homomorphism from **NATMUL** to **COMPOSE**.

```
In[30]:= member[TIMES, binhom[NATMUL, COMPOSE]] // AssertTest

Out[30]= member[TIMES, binhom[NATMUL, COMPOSE]] == True

In[31]:= member[TIMES, binhom[NATMUL, COMPOSE]] := True
```