associative

Johan G. F. Belinfante
2003 July 1

```
In[1]:= << goedel52.s27; << tools.m

:Package Title: goedel52.s27       2003 June 30 at 3:05 p.m.

It is now:  2003 Jul 7 at 9:19

Loading Simplification Rules

TOOLS.M                       Revised 2003 July 1

weightlimit = 40
```

■ summary

The predicate **associative** is defined by a rule wrapped inside **class**. The wrapping is done to prevent the definition from being expanded each time it is used. The definition of **associative[x]** does not require that **x** be a function.

■ definition and some examples

The wrapped definition of **associative[x]** is:

```
In[2]:= class[w_, associative[x_]] := class[w, and[subclass[x, cart[cart[V, V], V]],
          equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]]
```

In order ti reason effectively about associativity, it is useful also to have available these clauses:

```
In[3]:= implies[associative[x],
          equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]] //
         AssertTest

Out[3]= or[equal[composite[x, cross[x, Id]],
          composite[x, cross[Id, x], ASSOC]], not[associative[x]]] ==
         True

In[4]:= or[equal[composite[x_, cross[x_, Id]],
          composite[x_, cross[Id, x_], ASSOC]], not[associative[x_]]] :=
         True

In[5]:= implies[associative[x], subclass[x, cart[cart[V, V], V]]] // AssertTest

Out[5]= or[not[associative[x]], subclass[x, cart[cart[V, V], V]]] == True

In[6]:= or[not[associative[x_]], subclass[x_, cart[cart[V, V], V]]] := True
```

```
In[7]:= implies[and[subclass[x, cart[cart[V, V], V]], equal[
            composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]], associative[x]] //
         AssertTest

Out[7]= or[associative[x],
          not[equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]],
          not[subclass[x, cart[cart[V, V], V]]]] ==
         True

In[8]:= or[associative[x_],
          not[equal[composite[x_, cross[x_, Id]], composite[x_, cross[Id, x_], ASSOC]]],
          not[subclass[x_, cart[cart[V, V], V]]]] :=
         True
```

■ examples

To save writing, the corresponding unwrapped definition will be introduced:

```
In[9]:= ASSOCIATIVE[x_] := and[subclass[x, cart[cart[V, V], V]],
          equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]
```

The wrapped and unwrapped predicates are logically equivalent:

```
In[10]:= equiv[associative[x], ASSOCIATIVE[x]] // not // not

Out[10]= True
```

The unwrapped predicate makes it easy to find some examples of associative relations:

```
In[11]:= Select[NamedClasses, ASSOCIATIVE]

Out[11]= {0, CAP, COMPOSE, CUP, FIRST, NATADD, NATMUL, SECOND, SYMDIF}
```

The corresponding wrapped rules can be derived:

```
In[12]:= Map[AssertTest[associative[#]]&,
           {0, CAP, COMPOSE, CUP, FIRST, NATADD, NATMUL, SECOND, SYMDIF}] //
          TableForm

Out[12]//TableForm=
      associative[0] == True
      associative[CAP] == True
      associative[COMPOSE] == True
      associative[CUP] == True
      associative[FIRST] == True
      associative[NATADD] == True
      associative[NATMUL] == True
      associative[SECOND] == True
      associative[SYMDIF] == True
```

These are made into rewrite rules:

```
In[13]:= associative[0] := True
```

```
In[14]:= associative[CAP] := True
```

```
In[15]:= associative[COMPOSE] := True
```

```
In[16]:= associative[CUP] := True
```

```
In[17]:= associative[FIRST] := True
```

```
In[18]:= associative[NATADD] := True
```

```
In[19]:= associative[NATMUL] := True
```

```
In[20]:= associative[SECOND] := True
```

```
In[21]:= associative[SYMDIF] := True
```

This example is useful for providing a counterexample at the end of this notebook. It also shows that associative relations need not be functions.

```
In[22]:= associative[cart[cart[x, x], y]] // AssertTest

Out[22]= associative[cart[cart[x, x], y]] == True
```

```
In[23]:= associative[cart[cart[x_, x_], y_]] := True
```

To provide some simple counterexamples, the following may be useful:

```
In[24]:= associative[Id] // AssertTest

Out[24]= associative[Id] == False
```

```
In[25]:= associative[Id] := False
```

```
In[26]:= associative[V] // AssertTest

Out[26]= associative[V] == False
```

```
In[27]:= associative[V] := False
```

■ a theorem

In this section it will be shown that if **x** satisfies just the equational condition in the definition of associativity, then **composite[x,id[cart[V,V]]]** is associative. The following lemma is needed:

```
In[28]:= Assoc[cross[x, y], cross[Id, cross[Id, Id]], ASSOC] // Reverse

Out[28]= composite[cross[x, composite[y, id[cart[V, V]]]], ASSOC] ==
          composite[cross[x, y], ASSOC]
```

```
In[29]:= composite[cross[x_, composite[y_, id[cart[V, V]]]], ASSOC] :=
          composite[cross[x, y], ASSOC]
```

```
In[30]:= 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 -> cross[cross[Id, Id], Id]}]

Out[30]= or[equal[composite[x, cross[composite[x, id[cart[V, V]]], Id]],
            composite[x, cross[Id, x], ASSOC]],
           not[equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]] ==
          True
```

This is made into a temporary rewrite rule:

```
In[31]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The result may be rewritten as follows:

```
In[32]:= implies[equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]],
          ASSOCIATIVE[composite[x, id[cart[V, V]]]]]

Out[32]= True
```

To derive the corresponding wrapped version of this, a lemma is needed:

```
In[33]:= SubstTest[implies, ASSOCIATIVE[y], associative[y], y -> composite[x, id[cart[V, V]]]]

Out[33]= or[associative[composite[x, id[cart[V, V]]]],
           not[equal[composite[x, cross[composite[x, id[cart[V, V]]], Id]],
             composite[x, cross[Id, x], ASSOC]]]] ==
          True

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

The transfer from the unwrapped version to the wrapped version is done as follows:

```
In[35]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]],
            p2 -> ASSOCIATIVE[composite[x, id[cart[V, V]]]],
            p3 -> associative[composite[x, id[cart[V, V]]]]}]]

Out[35]= or[associative[composite[x, id[cart[V, V]]]],
           not[equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]] ==
          True

In[36]:= or[associative[composite[x_, id[cart[V, V]]]],
           not[equal[composite[x_, cross[x_, Id]], composite[x_, cross[Id, x_], ASSOC]]]] :=
          True
```

What about the converse? It is not true, as the following example shows:

```
In[37]:= implies[associative[composite[x, id[cart[V, V]]]],
           equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]] /.
          x -> V

Out[37]= False
```