case rule for associativity

Johan G. F. Belinfante
2011 December 27

```
In[1]:= SetDirectory["l:"]; << goedel.11dec22a

:Package Title: goedel.11dec22a              2011 December 22 at 4:00 a.m.

Loading takes about thirteen minutes, half that time due to builtin pauses.

It is now:  2011 Dec 27 at 2:52

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2011 Dec 27 at 3:5
```

summary

The concept of **associative relation** was introduced 2003 July 1 in the notebook **assoc.nb** via the following **class**-wrapped definition:

> **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]]]]**.

Wrapping the definition of the predicate **associative** with **class** prevents the definition from being automatically expanded into its constituent four literals. After all, the whole purpose of definitions is for them to serve as abbreviations for complex expressions. Nonetheless, since one does once in a while need to be able to access the definition, unwrapped rewrite rules for statements about associative relations were derived using **AssertTest**. Breaking into **class**-wrapped definitions with **AssertTest** is agonizingly slow. Here an equivalent **case**-wrapped characterization is derived that can function as an alternative to the **class**-wrapped rule.

subclass rules for case

Theorem.

```
In[2]:= subclass[case[p], x] // AssertTest

Out[2]= subclass[case[p], x] == or[equal[V, x], not[p]]
```

```
In[3]:= subclass[case[p_], x_] := or[equal[V, x], not[p]]
```

Theorem.

```
In[4]:= subclass[x, case[p]] // AssertTest

Out[4]= subclass[x, case[p]] == or[p, equal[0, x]]

In[5]:= subclass[x_, case[p_]] := or[p, equal[0, x]]
```

case-wrapped characterization

The following case-wrapped characterization for the predicate **associative[x]** can serve as a replacement for the removed **class**-wrapped definition.

Theorem. A rewrite rule for **case[associative[x]]**.

```
In[6]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> case[associative[x]],
           v → case[and[subclass[x, cart[cart[V, V], V]],           equal[
              composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]]]]}] // MapNotNot

Out[6]= equal[case[and[equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]],
            subclass[x, cart[cart[V, V], V]]]], case[associative[x]]] == True

In[7]:= case[associative[x_]] :=
         case[and[equal[composite[x, cross[x, Id]], composite[x, cross[Id, x], ASSOC]],
           subclass[x, cart[cart[V, V], V]]]]
```

removing the old class-wrapped rule

The old **class** rule interferes with the new **case** rule. The use of **AssertTest** remains slow if both rules are retained. The following example illustrates this.

```
In[8]:= TimeConstrained[associative[MIXMUL] // AssertTest // Timing, 30]

Out[8]= $Aborted
```

The old **class** rewrite rule will now be removed.

```
In[9]:= class[w_, associative[x_]] =.
```

When the same example is repeated, the new **case** rule comes into play, and the execution time is cut down to something quite reasonable.

```
In[10]:= TimeConstrained[associative[MIXMUL] // AssertTest // Timing, 30]

Out[10]= {0.671 Second, associative[MIXMUL] == False}
```

The following alternative to **AssertTest** also works here.

```
In[11]:= Timing[SubstTest[equal, V, case[p], p -> associative[MIXMUL]]]

Out[11]= {0.688 Second, associative[MIXMUL] == False}
```

Such brute force derivations rarely succeed even when one speeds things up. For example:

```
In[15]:= Timing[SubstTest[equal, V, case[p],
           p -> associative[composite[NATADD, id[cart[even, even]]]]]]

Out[15]= {0.765 Second,
          associative[composite[NATADD, id[cart[even, even]]]] == and[subclass[composite[NATADD,
              cross[composite[NATADD, id[cart[even, even]]], id[even]], inverse[ASSOC], cross[
               id[even], composite[id[cart[even, even]], inverse[NATADD]]]], NATADD], subclass[
            composite[NATADD, cross[id[even], composite[NATADD, id[cart[even, even]]]]], ASSOC,
              cross[composite[id[cart[even, even]], inverse[NATADD]], id[even]]], NATADD]]}
```

Better results are obtained by using relevant theorems. In this case, the following is much better.

Theorem. The restriction of natural number addition to even numbers is associative.

```
In[25]:= Timing[
          (SubstTest[implies, member[t, MONOIDS], associative[t], t -> composite[NATADD, id[
                cartsq[image[DIV, set[nat[x]]]]]]]] // Reverse) /. x → succ[set[0]]]

Out[25]= {0.031 Second, associative[composite[NATADD, id[cart[even, even]]]] == True}

In[26]:= associative[composite[NATADD, id[cart[even, even]]]] := True
```