binary operation wrapper

Johan G. F. Belinfante
2008 October 7

```
In[1]:= SetDirectory["l:"]; << goedel.08oct07a;<< tools.m

        :Package Title: goedel.08oct07a           2008 October 7 at 11:45 a.m.

        It is now:  2008 Oct 7 at 19:19

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 October 7

        weightlimit = 40
```

summary

A binary operation wrapper **binop[x]** is introduced via an **image[V, -]** rule.

definition

```
In[2]:= image[V, intersection[binop[x_], set[y_]]] :=
         intersection[image[V, intersection[BINOPS, set[x]]], image[V, intersection[x, set[y]]]]
```

Theorem.

```
In[3]:= Map[fix, SubstTest[reify, y, image[V, intersection[t, set[y]]], t → binop[x]]] // Reverse

Out[3]= intersection[x, image[V, intersection[BINOPS, set[x]]]] == binop[x]
```

```
In[4]:= intersection[x_, image[V, intersection[BINOPS, set[x_]]]] := binop[x]
```

Lemma. (Simplification rule.)

```
In[5]:= equiv[or[equal[0, x], member[x, BINOPS]], member[x, BINOPS]]

Out[5]= True
```

```
In[6]:= or[equal[0, x_], member[x_, BINOPS]] := member[x, BINOPS]
```

Theorem. (Wrapper removal rule.)

```
In[7]:= SubstTest[equal, x,
          intersection[x, image[V, intersection[t, set[x]]]], t → BINOPS] // Reverse

Out[7]= equal[x, binop[x]] == member[x, BINOPS]

In[8]:= equal[x_, binop[x_]] := member[x, BINOPS]
```

Theorem. Automatic wrapper removal rule.

```
In[9]:= implies[member[x, BINOPS], equal[binop[x], x]]

Out[9]= True

In[10]:= binop[x_] := x /; member[x, BINOPS]
```

Theorem. (Wrapper introduction rule.)

```
In[11]:= SubstTest[member,
           intersection[x, image[V, intersection[t, set[x]]]], t, t → BINOPS] // Reverse

Out[11]= member[binop[x], BINOPS] == True

In[12]:= member[binop[x_], BINOPS] := True
```

reify rule

```
In[13]:= SubstTest[reify, x,
           intersection[f[x], image[V, intersection[t, set[f[x]]]]], t → BINOPS] // Reverse

Out[13]= reify[x, binop[f[x]]] ==
          composite[reify[x, f[x]], id[image[inverse[VERTSECT[reify[x, f[x]]], BINOPS]]]

In[14]:= reify[x_, binop[y_]] :=
          composite[reify[x, y], id[image[inverse[VERTSECT[reify[x, y]]], BINOPS]]]
```

properties

Theorem.

```
In[15]:= SubstTest[implies, member[t, BINOPS], member[t, V], t → binop[x]] // Reverse

Out[15]= member[binop[x], V] == True

In[16]:= member[binop[x_], V] := True
```

Theorem.

```
In[17]:= SubstTest[implies, member[t, BINOPS],
           subclass[t, cart[cart[V, V], V]], t → binop[x]] // Reverse
Out[17]= subclass[binop[x], cart[cart[V, V], V]] == True

In[18]:= subclass[binop[x_], cart[cart[V, V], V]] := True
```

Corollary.

```
In[19]:= equal[composite[binop[x], id[cart[V, V]]], binop[x]]
Out[19]= True

In[20]:= composite[binop[x_], id[cart[V, V]]] := binop[x]
```

Theorem.

```
In[21]:= SubstTest[implies, member[t, BINOPS], FUNCTION[t], t → binop[x]] // Reverse
Out[21]= FUNCTION[binop[x]] == True

In[22]:= FUNCTION[binop[x_]] := True
```

Theorem. (The domain is a square.)

```
In[23]:= SubstTest[implies, member[t, BINOPS],
           member[domain[t], image[CART, Id]], t → binop[x]] // Reverse
Out[23]= equal[cart[fix[domain[binop[x]]], fix[domain[binop[x]]]], domain[binop[x]]] == True

In[24]:= cart[fix[domain[binop[x_]]], fix[domain[binop[x_]]]] := domain[binop[x]]
```

Corollary.

```
In[25]:= SubstTest[domain, cartsq[t], t → fix[domain[binop[x]]]] // Reverse
Out[25]= domain[domain[binop[x]]] == fix[domain[binop[x]]]

In[26]:= domain[domain[binop[x_]]] := fix[domain[binop[x]]]
```

Corollary.

```
In[27]:= SubstTest[range, cartsq[t], t → fix[domain[binop[x]]]] // Reverse
Out[27]= range[domain[binop[x]]] == fix[domain[binop[x]]]

In[28]:= range[domain[binop[x_]]] := fix[domain[binop[x]]]
```

Corollary.

```
In[29]:= SubstTest[inverse, cartsq[t], t → fix[domain[binop[x]]]] // Reverse
Out[29]= inverse[domain[binop[x]]] == domain[binop[x]]

In[30]:= inverse[domain[binop[x_]]] := domain[binop[x]]
```

Corollary.

```
In[31]:= SubstTest[subclass, cartsq[t], cartsq[V], t -> fix[domain[binop[x]]]] // Reverse
Out[31]= subclass[domain[binop[x]], cart[V, V]] == True

In[32]:= subclass[domain[binop[x_]], cart[V, V]] := True
```

Corollary.

```
In[33]:= SubstTest[inverse, inverse[t], t → domain[binop[x]]]
Out[33]= composite[Id, domain[binop[x]]] == domain[binop[x]]

In[34]:= composite[Id, domain[binop[x_]]] := domain[binop[x]]
```

Theorem.

```
In[35]:= SubstTest[implies, member[t, BINOPS],
           subclass[range[t], fix[domain[t]]], t → binop[x]] // Reverse
Out[35]= subclass[range[binop[x]], fix[domain[binop[x]]]] == True

In[36]:= subclass[range[binop[x_]], fix[domain[binop[x_]]]] := True
```

variable-free formulations

Theorem.

```
In[37]:= Map[composite[VERTSECT[#], id[BINOPS]] &,
          SubstTest[reify, x, composite[f[x], id[cart[V, V]]], f → binop]]
Out[37]= composite[IMAGE[id[cart[cart[V, V], V]]], id[BINOPS]] == id[BINOPS]

In[38]:= composite[IMAGE[id[cart[cart[V, V], V]]], id[BINOPS]] := id[BINOPS]
```

Theorem.

```
In[39]:= Map[composite[VERTSECT[#], id[BINOPS]] &, SubstTest[reify, x, funpart[f[x]], f → binop]]
Out[39]= composite[FUNPART, id[BINOPS]] == id[BINOPS]

In[40]:= composite[FUNPART, id[BINOPS]] := id[BINOPS]
```

Theorem.

```
In[41]:= Map[composite[VERTSECT[#], id[BINOPS]] &,
          SubstTest[reify, x, cartsq[fix[domain[f[x]]]], f → binop]]
Out[41]= composite[CART, DUP, IMAGE[inverse[DUP]], IMAGE[FIRST], id[BINOPS]] ==
          composite[IMAGE[FIRST], id[BINOPS]]
```

```
In[42]:= composite[CART, DUP, IMAGE[inverse[DUP]], IMAGE[FIRST], id[BINOPS]] :=
          composite[IMAGE[FIRST], id[BINOPS]]
```

Theorem.

```
In[43]:= Map[composite[VERTSECT[#], id[BINOPS]] &,
          SubstTest[reify, x, domain[domain[f[x]]], f → binop]]

Out[43]= composite[IMAGE[FIRST], IMAGE[FIRST], id[BINOPS]] ==
          composite[IMAGE[inverse[DUP]], IMAGE[FIRST], id[BINOPS]]
```

```
In[44]:= composite[IMAGE[FIRST], IMAGE[FIRST], id[BINOPS]] :=
          composite[IMAGE[inverse[DUP]], IMAGE[FIRST], id[BINOPS]]
```

Theorem.

```
In[45]:= Map[composite[VERTSECT[#], id[BINOPS]] &,
          SubstTest[reify, x, range[domain[f[x]]], f → binop]]

Out[45]= composite[IMAGE[SECOND], IMAGE[FIRST], id[BINOPS]] ==
          composite[IMAGE[inverse[DUP]], IMAGE[FIRST], id[BINOPS]]
```

```
In[46]:= composite[IMAGE[SECOND], IMAGE[FIRST], id[BINOPS]] :=
          composite[IMAGE[inverse[DUP]], IMAGE[FIRST], id[BINOPS]]
```

Theorem.

```
In[47]:= Map[composite[VERTSECT[#], id[BINOPS]] &,
          SubstTest[reify, x, inverse[domain[f[x]]], f → binop]]

Out[47]= composite[IMAGE[SWAP], IMAGE[FIRST], id[BINOPS]] == composite[IMAGE[FIRST], id[BINOPS]]
```

```
In[48]:= composite[IMAGE[SWAP], IMAGE[FIRST], id[BINOPS]] := composite[IMAGE[FIRST], id[BINOPS]]
```

Corollary.

```
In[49]:= Assoc[INVERSE, IMAGE[SWAP], composite[IMAGE[FIRST], id[BINOPS]]]

Out[49]= composite[INVERSE, IMAGE[FIRST], id[BINOPS]] == composite[IMAGE[FIRST], id[BINOPS]]
```

```
In[50]:= composite[INVERSE, IMAGE[FIRST], id[BINOPS]] := composite[IMAGE[FIRST], id[BINOPS]]
```

Corollary.

```
In[51]:= Assoc[id[P[cart[V, V]]], IMAGE[SWAP], composite[IMAGE[FIRST], id[BINOPS]]]

Out[51]= composite[id[P[cart[V, V]]], IMAGE[FIRST], id[BINOPS]] ==
          composite[IMAGE[FIRST], id[BINOPS]]
```

```
In[52]:= composite[id[P[cart[V, V]]], IMAGE[FIRST], id[BINOPS]] :=
          composite[IMAGE[FIRST], id[BINOPS]]
```