quasigroup wrapper

Johan G. F. Belinfante
2008 October 9

```
In[1]:= SetDirectory["l:"]; << goedel.08oct08a;<< tools.m

        :Package Title: goedel.08oct08a          2008 October 8 at 7:50 p.m.

        It is now:  2008 Oct 9 at 14:53

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 October 7

        weightlimit = 40
```

summary

A quasigroup wrapper **quasigp[x]** is defined by analogy with the **binop[x]** wrapper for binary operations, and many of its basic properties are derived as corollaries of results for **binop[x]**. In a few cases sharper results are obtained for the **quasigp[x]** wrapper because its domain is the cartesian square of its range. Due to looping problems, one cannot rewrite the domain as the cartesian square of the range, but one must instead orient this rewrite rule in the opposite direction.

definition

The following rewrite rule serves to define the **quasigp[x]** wrapper:

```
In[2]:= image[V, intersection[quasigp[x_], set[y_]]] :=
         intersection[image[V, intersection[BINOPS, set[x]]],
          image[V, intersection[BINOPS, set[rotate[x]]]],
          image[V, intersection[BINOPS, set[rotate[composite[x, SWAP]]]]],
          image[V, intersection[x, set[y]]]]
```

normalization

Theorem.

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

Out[3]= intersection[binop[x], image[V, intersection[BINOPS, set[rotate[x]]]],
          image[V, intersection[BINOPS, set[rotate[composite[x, SWAP]]]]]] == quasigp[x]
```

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

wrapper introduction and removal rules

Lemma. (Simplification rule.)

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

Out[5]= True

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

Theorem. (Wrapper removal rule.)

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

Out[7]= equal[x, quasigp[x]] == member[x, QUASIGPS]

In[8]:= equal[x_, quasigp[x_]] := member[x, QUASIGPS]
```

Theorem. (Automatic removal.)

```
In[9]:= implies[member[x, QUASIGPS], equal[quasigp[x], x]]

Out[9]= True

In[10]:= quasigp[x_] := x /; member[x, QUASIGPS]
```

Theorem. (Wrapper introduction rule.)

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

Out[11]= member[quasigp[x], QUASIGPS] == True

In[12]:= member[quasigp[x_], QUASIGPS] := True
```

reify rule

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

Out[13]= reify[x, quasigp[f[x]]] ==
          composite[reify[x, f[x]], id[image[inverse[VERTSECT[reify[x, f[x]]], QUASIGPS]]]

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

properties of the quasigroup wrapper

Theorem.

```
In[15]:= SubstTest[implies, member[t, QUASIGPS], member[t, BINOPS], t → quasigp[x]] // Reverse
Out[15]= member[quasigp[x], BINOPS] == True

In[16]:= member[quasigp[x_], BINOPS] := True
```

Corollary.

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

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

Corollary.

```
In[19]:= SubstTest[composite, binop[t], id[cart[V, V]], t → quasigp[x]] // Reverse
Out[19]= composite[quasigp[x], id[cart[V, V]]] == quasigp[x]

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

Theorem.

```
In[21]:= SubstTest[member, binop[t], V, t → quasigp[x]] // Reverse
Out[21]= member[quasigp[x], V] == True

In[22]:= member[quasigp[x_], V] := True
```

Theorem

```
In[23]:= SubstTest[FUNCTION, binop[t], t → quasigp[x]] // Reverse
Out[23]= FUNCTION[quasigp[x]] == True

In[24]:= FUNCTION[quasigp[x_]] := True
```

Corollary.

```
In[25]:= SubstTest[implies, equal[x, quasigp[t]], FUNCTION[x], t → x] // Reverse
Out[25]= or[FUNCTION[x], not[member[x, QUASIGPS]]] == True

In[26]:= or[FUNCTION[x_], not[member[x_, QUASIGPS]]] := True
```

flip and rotate rules

Theorem.

```
In[27]:= SubstTest[implies, member[t, QUASIGPS],
           member[rotate[t], BINOPS], t → quasigp[x]] // Reverse

Out[27]= member[rotate[quasigp[x]], BINOPS] == True

In[28]:= member[rotate[quasigp[x_]], BINOPS] := True
```

Corollary.

```
In[29]:= SubstTest[FUNCTION, binop[t], t → rotate[quasigp[x]]] // Reverse

Out[29]= FUNCTION[rotate[quasigp[x]]] == True

In[30]:= FUNCTION[rotate[quasigp[x_]]] := True
```

Theorem.

```
In[31]:= SubstTest[implies, member[t, QUASIGPS],
           member[rotate[flip[t]], BINOPS], t → quasigp[x]] // Reverse

Out[31]= member[rotate[composite[quasigp[x], SWAP]], BINOPS] == True

In[32]:= member[rotate[composite[quasigp[x_], SWAP]], BINOPS] := True
```

Corollary.

```
In[33]:= SubstTest[FUNCTION, binop[t], t → rotate[flip[quasigp[x]]]] // Reverse

Out[33]= FUNCTION[rotate[composite[quasigp[x], SWAP]]] == True

In[34]:= FUNCTION[rotate[composite[quasigp[x_], SWAP]]] := True
```

domain and range rules

In this section it will be shown that the domain of **quasigp[x]** is the cartesian square of its range. The orientation of the rewrite rule for this fact is dictated by the following result:

Theorem.

```
In[35]:= SubstTest[empty, domain[funpart[t]], t → quasigp[x]] // Reverse

Out[35]= equal[0, domain[quasigp[x]]] == equal[0, quasigp[x]]

In[36]:= equal[0, domain[quasigp[x_]]] := equal[0, quasigp[x]]
```

The above rewrite rule would unfortunately lead to looping if one were to introduce a rewrite rule that eliminates the domain in favor of the range.

Lemma.

```
In[37]:= ImageComp[quasigp[x], id[cart[V, V]], V] // Reverse
Out[37]= image[quasigp[x], cart[V, V]] == range[quasigp[x]]

In[38]:= image[quasigp[x_], cart[V, V]] := range[quasigp[x]]
```

Temporary lemma.

```
In[39]:= Map[domain, SubstTest[cartsq, fix[domain[binop[t]]], t → rotate[quasigp[x]]]] // Reverse
Out[39]= fix[composite[FIRST, inverse[quasigp[x]]]] == range[quasigp[x]]

In[40]:= fix[composite[FIRST, inverse[quasigp[x_]]]] := range[quasigp[x]]
```

Theorem.

```
In[41]:= Map[range, SubstTest[cartsq, fix[domain[binop[t]]], t → rotate[quasigp[x]]]]
Out[41]= domain[domain[quasigp[x]]] == range[quasigp[x]]

In[42]:= domain[domain[quasigp[x_]]] := range[quasigp[x]]
```

Theorem.

```
In[43]:= SubstTest[domain, domain[binop[t]], t → quasigp[x]]
Out[43]= fix[domain[quasigp[x]]] == range[quasigp[x]]

In[44]:= fix[domain[quasigp[x_]]] := range[quasigp[x]]
```

Theorem. (As indcated above, the orientation of this rule is dictated by the desire to avoid looping.)

```
In[45]:= SubstTest[cartsq, fix[domain[binop[t]]], t → quasigp[x]] // Reverse
Out[45]= cart[range[quasigp[x]], range[quasigp[x]]] == domain[quasigp[x]]

In[46]:= cart[range[quasigp[x_]], range[quasigp[x_]]] := domain[quasigp[x]]
```

Corollary.

```
In[47]:= SubstTest[range, cart[t, t], t → range[quasigp[x]]] // Reverse
Out[47]= range[domain[quasigp[x]]] == range[quasigp[x]]

In[48]:= range[domain[quasigp[x_]]] := range[quasigp[x]]
```

Corollary.

In[49]:= **SubstTest[inverse, cartsq[t], t → range[quasigp[x]]] // Reverse**

Out[49]= inverse[domain[quasigp[x]]] == domain[quasigp[x]]

In[50]:= **inverse[domain[quasigp[x_]]] := domain[quasigp[x]]**

Corollary.

In[51]:= **SubstTest[inverse, inverse[t], t → domain[quasigp[x]]]**

Out[51]= composite[Id, domain[quasigp[x]]] == domain[quasigp[x]]

In[52]:= **composite[Id, domain[quasigp[x_]]] := domain[quasigp[x]]**

---

rotated domains

Theorem.

In[53]:= **SubstTest[cartsq, fix[domain[binop[t]]], t → rotate[quasigp[x]]]**

Out[53]= composite[FIRST, inverse[quasigp[x]]] == domain[quasigp[x]]

In[54]:= **composite[FIRST, inverse[quasigp[x_]]] := domain[quasigp[x]]**

Corollary.

In[55]:= **composite[quasigp[x], inverse[FIRST]] // DoubleInverse**

Out[55]= composite[quasigp[x], inverse[FIRST]] == domain[quasigp[x]]

In[56]:= **composite[quasigp[x_], inverse[FIRST]] := domain[quasigp[x]]**

Theorem.

In[57]:= **SubstTest[composite, quasigp[t], inverse[FIRST], t → rotate[quasigp[x]]] // Reverse**

Out[57]= composite[SECOND, inverse[quasigp[x]]] == domain[quasigp[x]]

In[58]:= **composite[SECOND, inverse[quasigp[x_]]] := domain[quasigp[x]]**

Corollary.

In[59]:= **composite[quasigp[x], inverse[SECOND]] // DoubleInverse**

Out[59]= composite[quasigp[x], inverse[SECOND]] == domain[quasigp[x]]

In[60]:= **composite[quasigp[x_], inverse[SECOND]] := domain[quasigp[x]]**