the class of all groups

Johan G. F. Belinfante
2008 October 18

```
In[1]:= SetDirectory["l:"]; << goedel.08oct18a;<< tools.m

        :Package Title: goedel.08oct18a          2008 October 18 at 4:50 a.m.

        It is now:  2008 Oct 18 at 4:52

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 October 17

        weightlimit = 40
```

summary

A group is a nonempty quasigroup binary operation that is also a semigroup binary operation. The following rewrite rule has been introduced to define the class **GROUPS**:

```
In[2]:= Begin["Goedel`Private`"];

In[3]:= FirstMatch[class[x_, member[y_, HoldPattern[GROUPS]]]]

Out[3]= class[x_, member[y_, GROUPS]] := class[x, and[
          member[y, QUASIGPS], member[y, SEMIGPS], not[equal[0, y]]]]
```

normalization

```
In[4]:= GROUPS // Normality // Reverse

Out[4]= intersection[QUASIGPS, SEMIGPS, complement[set[0]]] == GROUPS

In[5]:= intersection[QUASIGPS, SEMIGPS, complement[set[0]]] := GROUPS
```

The following facts are immediate consequences of the definition.

Corollary.

```
In[6]:= SubstTest[subclass, intersection[u, v],
          u, {u → QUASIGPS, v → dif[SEMIGPS, set[0]]}] // Reverse

Out[6]= subclass[GROUPS, QUASIGPS] == True

In[7]:= subclass[GROUPS, QUASIGPS] := True
```

Corollary.

```
In[8]:= SubstTest[subclass, intersection[u, v],
          v, {u → dif[QUASIGPS, set[0]], v → SEMIGPS}] // Reverse
```

```
Out[8]= subclass[GROUPS, SEMIGPS] == True
```

```
In[9]:= subclass[GROUPS, SEMIGPS] := True
```

Corollary.

```
In[11]:= Map[not, SubstTest[subclass, intersection[u, v], v,
            {u → intersection[QUASIGPS, SEMIGPS], v → complement[set[0]]}]] // Reverse
```

```
Out[11]= member[0, GROUPS] == False
```

```
In[12]:= member[0, GROUPS] := False
```

Corollary.

```
In[13]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
            subclass[u, w], {u → GROUPS, v -> QUASIGPS, w→ BINOPS}] // Reverse
```

```
Out[13]= subclass[GROUPS, BINOPS] == True
```

```
In[14]:= subclass[GROUPS, BINOPS] := True
```

Corollary.

```
In[15]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
            subclass[u, w], {u → GROUPS, v→ BINOPS, w→ FUNS}] // Reverse
```

```
Out[15]= subclass[GROUPS, FUNS] == True
```

```
In[16]:= subclass[GROUPS, FUNS] := True
```

Corollary. A group is a nonempty associative quasigroup.

```
In[17]:= AssInt[ASSOCIATIVE, BINOPS, dif[QUASIGPS, set[0]]]
```

```
Out[17]= intersection[ASSOCIATIVE, QUASIGPS, complement[set[0]]] == GROUPS
```

```
In[18]:= intersection[ASSOCIATIVE, QUASIGPS, complement[set[0]]] := GROUPS
```

examples

A basic example is the group of integers under addition.

```
In[19]:= SubstTest[member, INTADD, intersection[u, v],
            {u → SEMIGPS, v→ dif[QUASIGPS, set[0]]}] // Reverse
```

```
Out[19]= member[INTADD, GROUPS] == True
```

```
In[20]:= member[INTADD, GROUPS] := True
```

Another collection of groups is provided by the symmetric difference operation restricted to any power set.

```
In[21]:= member[composite[SYMDIF, id[cart[P[x], P[x]]]], GROUPS] // AssertTest

Out[21]= member[composite[SYMDIF, id[cart[P[x], P[x]]]], GROUPS] == member[x, V]

In[22]:= member[composite[SYMDIF, id[cart[P[x_], P[x_]]]], GROUPS] := member[x, V]
```

Lemma.

```
In[23]:= Map[equal[V, class[x, #]] &,
           SubstTest[implies, member[u, v], member[range[u], image[IMAGE[SECOND], v]],
            {u -> composite[SYMDIF, id[cart[P[x], P[x]]]], v -> GROUPS}]] // Reverse

Out[23]= subclass[range[POWER], image[IMAGE[SECOND], GROUPS]] == True

In[24]:= subclass[range[POWER], image[IMAGE[SECOND], GROUPS]] := True
```

Lemma.

```
In[25]:= SubstTest[implies, member[t, V], member[range[t], V], t → U[x]] // Reverse

Out[25]= or[member[range[U[x]], V], not[member[x, V]]] == True

In[26]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem. The class of groups is not a set.

```
In[27]:= Map[and[member[GROUPS, x], #] &, Map[not[implies[#, not[member[GROUPS, V]]]] &,
           SubstTest[implies, and[subclass[u, v], member[v, V]],
            member[u, V], {u → range[POWER], v → image[IMAGE[SECOND], GROUPS]}]]]

Out[27]= member[GROUPS, x] == False

In[28]:= member[GROUPS, x_] := False
```