ASSOCIATIVE and divisibility

Johan G. F. Belinfante
2004 December 7

```
In[1]:= SetDirectory["i:"]; << goedel64.06a; << tools.m

        :Package Title: goedel64.06a      2004 December 65 at 12:20 noon

        It is now:  2004 Dec 7 at 12:5

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 November 17

        weightlimit = 40
```

summary

The membership rule for the class **ASSOCIATIVE** of all (small) associative relations is replaced with a simpler one. Several rewrite rules are derived, including a flip rule and a variable-free formulation of the transitivity of divisibility relations. A simple condition is derived for a restriction of **FIRST** or **SECOND** to be associative. In particular, their restrictions to equivalence relations are associative. This is used to show that any equivalence relation is a divisibility relation.

a new membership rule for ASSOCIATIVE

The existing membership rule for **ASSOCIATIVE** is equivalent to a simpler one:

```
In[2]:= equiv[member[x, ASSOCIATIVE], and[member[x, V], associative[x]]] // not // not
Out[2]= True
```

The old membership rule is removed.

```
In[3]:= member[x_, ASSOCIATIVE] =.
```

A new rule is installed in its place.

```
In[4]:= member[x_, ASSOCIATIVE] := and[associative[x], member[x, V]]
```

some rewrite rules for ASSOCIATIVE

```
In[5]:= equal[intersection[ASSOCIATIVE, P[cart[cart[V, V], V]]], ASSOCIATIVE]

Out[5]= True

In[6]:= intersection[ASSOCIATIVE, P[cart[cart[V, V], V]]] := ASSOCIATIVE

In[7]:= ImageComp[IMAGE[id[cart[cart[V, V], V]]],
          id[P[cart[cart[V, V], V]]], ASSOCIATIVE] // Reverse

Out[7]= image[IMAGE[id[cart[cart[V, V], V]]], ASSOCIATIVE] == ASSOCIATIVE

In[8]:= image[IMAGE[id[cart[cart[V, V], V]]], ASSOCIATIVE] := ASSOCIATIVE
```

flip rule

The flip of any associative relation is associative. Since any associative relation is the flip of its own flip, the class of flipped associative relations is the same as the class of all associative relations. A rewrite rule for this is derived here. We begin with a set-hood lemma.

```
In[9]:= or[and[associative[composite[x, id[cart[V, V]]]],
           member[composite[x, SWAP], V]],
          not[associative[x]], not[member[x, V]]] // NotNotTest

Out[9]= or[and[associative[composite[x, id[cart[V, V]]]],
           member[composite[x, SWAP], V]],
          not[associative[x]], not[member[x, V]]] == True

In[10]:= (% /. x → x_) /. Equal → SetDelayed
```

The inclusion in one direction follows:

```
In[11]:= Map[equal[V, #] &,
           SubstTest[class, x, implies[member[x, y], member[composite[x, z], y]],
            {y → ASSOCIATIVE, z → SWAP}]] // Reverse

Out[11]= subclass[image[IMAGE[cross[SWAP, Id]], ASSOCIATIVE], ASSOCIATIVE] == True

In[12]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[13]:= composite[IMAGE[cross[SWAP, Id]], IMAGE[cross[SWAP, Id]]] // VSNormality

Out[13]= composite[IMAGE[cross[SWAP, Id]], IMAGE[cross[SWAP, Id]]] ==
          IMAGE[id[cart[cart[V, V], V]]]

In[14]:= composite[IMAGE[cross[SWAP, Id]], IMAGE[cross[SWAP, Id]]] :=
          IMAGE[id[cart[cart[V, V], V]]]
```

Temporary lemma.

```
In[15]:= ImageComp[IMAGE[cross[SWAP, Id]],
           IMAGE[cross[SWAP, Id]], ASSOCIATIVE] // Reverse

Out[15]= image[IMAGE[cross[SWAP, Id]],
           image[IMAGE[cross[SWAP, Id]], ASSOCIATIVE]] == ASSOCIATIVE

In[16]:= (% /. x → x_) /. Equal → SetDelayed
```

The inclusion in the opposite direction is obtained:

```
In[17]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> image[IMAGE[cross[SWAP, Id]], ASSOCIATIVE],
           v -> ASSOCIATIVE, w -> IMAGE[cross[SWAP, Id]]}]

Out[17]= subclass[ASSOCIATIVE, image[IMAGE[cross[SWAP, Id]], ASSOCIATIVE]] == True

In[18]:= % /. Equal → SetDelayed
```

These two inclusions are combined into an equation and made into a rewrite rule.

```
In[19]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> ASSOCIATIVE, v -> image[IMAGE[cross[SWAP, Id]], ASSOCIATIVE]}]

Out[19]= True == equal[ASSOCIATIVE, image[IMAGE[cross[SWAP, Id]], ASSOCIATIVE]]

In[20]:= image[IMAGE[cross[SWAP, Id]], ASSOCIATIVE] := ASSOCIATIVE
```

---

left and right divisibility

Lemmas.

```
In[21]:= composite[IMAGE[cross[FIRST, Id]], IMAGE[cross[SWAP, Id]]] // VSNormality

Out[21]= composite[IMAGE[cross[FIRST, Id]], IMAGE[cross[SWAP, Id]]] ==
          IMAGE[cross[SECOND, Id]]

In[22]:= composite[IMAGE[cross[FIRST, Id]], IMAGE[cross[SWAP, Id]]] :=
          IMAGE[cross[SECOND, Id]]
```

```
In[23]:= composite[IMAGE[cross[SECOND, Id]], IMAGE[cross[SWAP, Id]]] // VSNormality

Out[23]= composite[IMAGE[cross[SECOND, Id]], IMAGE[cross[SWAP, Id]]] ==
          IMAGE[cross[FIRST, Id]]

In[24]:= composite[IMAGE[cross[SECOND, Id]], IMAGE[cross[SWAP, Id]]] :=
          IMAGE[cross[FIRST, Id]]
```

Left divisibility for one associative relation is right divisibility for the flipped relation, and conversely.

```
In[25]:= ImageComp[IMAGE[cross[FIRST, Id]], IMAGE[cross[SWAP, Id]], ASSOCIATIVE]

Out[25]= image[IMAGE[cross[SECOND, Id]], ASSOCIATIVE] ==
          image[IMAGE[cross[FIRST, Id]], ASSOCIATIVE]

In[26]:= image[IMAGE[cross[SECOND, Id]], ASSOCIATIVE] :=
          image[IMAGE[cross[FIRST, Id]], ASSOCIATIVE]
```

---

divisibility relations are transitive

Sethood lemma.

```
In[27]:= Map[implies[member[x, y], #] &,
           member[composite[x, inverse[FIRST]], V] // AssertTest] // MapNotNot

Out[27]= or[member[composite[x, inverse[FIRST]], V], not[member[x, y]]] == True

In[28]:= or[member[composite[x_, inverse[FIRST]], V], not[member[x_, y_]]] := True
```

Temporary lemma obtained by double negation.

```
In[29]:= implies[member[x, ASSOCIATIVE],
           member[composite[x, inverse[FIRST]], TRV]] // NotNotTest

Out[29]= or[and[member[composite[x, inverse[FIRST]], V],
             TRANSITIVE[composite[x, inverse[FIRST]]]],
           not[associative[x]], not[member[x, V]]] == True

In[30]:= (% /. x → x_) /. Equal → SetDelayed
```

It follows that every divisibility relation is transitive.

```
In[31]:= Map[equal[V, #] &, SubstTest[class, x,
            implies[member[x, u], member[composite[x, inverse[FIRST]], v]],
            {u → ASSOCIATIVE, v → TRV}]] // Reverse

Out[31]= subclass[image[IMAGE[cross[FIRST, Id]], ASSOCIATIVE], TRV] == True
```

```
In[32]:= subclass[image[IMAGE[cross[FIRST, Id]], ASSOCIATIVE], TRV] := True
```

Question: Does equality hold? Is every transitive relation a divisibility relation for some associative relation?

---

equivalence relations are divisibility relations

Here is a criterion for a restriction of **FIRST** to be associative.

```
In[33]:= associative[composite[FIRST, id[x]]] // AssertTest

Out[33]= associative[composite[FIRST, id[x]]] ==
          and[subclass[composite[x, inverse[x]], x], TRANSITIVE[composite[Id, x]]]

In[34]:= associative[composite[FIRST, id[x_]]] :=
          and[subclass[composite[x, inverse[x]], x], TRANSITIVE[composite[Id, x]]]
```

Flipping yields a similar result for restrictions of **SECOND**.

```
In[35]:= SubstTest[associative, flip[y], y → composite[FIRST, id[inverse[x]]]]

Out[35]= associative[composite[SECOND, id[x]]] ==
          and[subclass[composite[inverse[x], x], x], TRANSITIVE[composite[Id, x]]]

In[36]:= associative[composite[SECOND, id[x_]]] :=
          and[subclass[composite[inverse[x], x], x], TRANSITIVE[composite[Id, x]]]
```

In particular, this holds when **x** is an equivalence relation. It follows from this that every equivalence relation **x** is the divisibility relation of some associative relation, namely of the associative relation **composite[SECOND, id[x]]**. A variable-free statement of this fact will now be derived.

```
In[37]:= Map[equal[EQV, image[EQUIV, #]] &,
           SubstTest[class, x, member[composite[SECOND, id[eqv[setpart[x]]]], y],
            y → ASSOCIATIVE]] // Reverse

Out[37]= subclass[image[IMAGE[composite[id[SECOND], inverse[FIRST]]], EQV],
           ASSOCIATIVE] == True

In[38]:= subclass[image[IMAGE[composite[id[SECOND], inverse[FIRST]]], EQV],
           ASSOCIATIVE] := True
```

Lemma.

```
In[39]:= composite[IMAGE[cross[FIRST, Id]],
           IMAGE[composite[id[SECOND], inverse[FIRST]]]] // VSNormality

Out[39]= composite[IMAGE[cross[FIRST, Id]],
           IMAGE[composite[id[SECOND], inverse[FIRST]]]] == IMAGE[id[cart[V, V]]]

In[40]:= composite[IMAGE[cross[FIRST, Id]],
           IMAGE[composite[id[SECOND], inverse[FIRST]]]] := IMAGE[id[cart[V, V]]]
```

Lemma.

```
In[41]:= ImageComp[IMAGE[cross[FIRST, Id]],
           IMAGE[composite[id[SECOND], inverse[FIRST]]], EQV] // Reverse

Out[41]= image[IMAGE[cross[FIRST, Id]],
           image[IMAGE[composite[id[SECOND], inverse[FIRST]]], EQV]] == EQV

In[42]:= image[IMAGE[cross[FIRST, Id]],
           image[IMAGE[composite[id[SECOND], inverse[FIRST]]], EQV]] := EQV
```

This yields the promised variable-free formulation of the fact that any equivalence relation is a divisibility relation for some associative relation.

```
In[43]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> image[IMAGE[composite[id[SECOND], inverse[FIRST]]], EQV],
           v -> ASSOCIATIVE, w -> IMAGE[cross[FIRST, Id]]}]

Out[43]= subclass[EQV, image[IMAGE[cross[FIRST, Id]], ASSOCIATIVE]] == True

In[44]:= subclass[EQV, image[IMAGE[cross[FIRST, Id]], ASSOCIATIVE]] := True
```