relations similar to a unary operation

Johan G. F. Belinfante
2010 May 9

```
In[1]:= SetDirectory["l:"]; << goedel.10may07a; << tools.m

        :Package Title: goedel.10may07a                2010 May 7 at 3:35 p.m.

        It is now:  2010 May 9 at 8:26

        Loading Simplification Rules

        TOOLS.M                                       Revised 2010 February 26

        weightlimit = 40
```

summary

Any relation similar to a unary operation is a unary operation. This is proved by showing that the condition **range[x] ⊂ domain[x]** is preserved by similarity transformations. As a corollary, it follows that a relation similar to a permutation is a permutation.

image rules for SIMILAR

Some simplification rules about images of the relation **SIMILAR** are derived in this section.

```
In[2]:= ImageComp[IMG,
          composite[id[composite[inverse[S], IMAGE[FIRST], id[image[CROSS, id[BIJ]]]]],
           inverse[SECOND]], x] // Reverse

Out[2]= image[IMG, composite[id[x], inverse[S], IMAGE[FIRST], id[image[CROSS, id[BIJ]]]]] ==
         image[SIMILAR, x]

In[3]:= image[IMG, composite[id[x_], inverse[S], IMAGE[FIRST], id[image[CROSS, id[BIJ]]]]] :=
         image[SIMILAR, x]

In[4]:= ImageComp[SIMILAR, id[P[cart[V, V]]], x] // Reverse

Out[4]= image[SIMILAR, intersection[x, P[cart[V, V]]]] == image[SIMILAR, x]

In[5]:= image[SIMILAR, intersection[x_, P[cart[V, V]]]] := image[SIMILAR, x]

In[6]:= ImageComp[id[P[cart[V, V]]], SIMILAR, x] // Reverse

Out[6]= intersection[image[SIMILAR, x], P[cart[V, V]]] == image[SIMILAR, x]
```

```
In[7]:= intersection[image[SIMILAR, x_], P[cart[V, V]]] := image[SIMILAR, x]
```

Lemma.

```
In[8]:= SubstTest[subclass, t, image[SIMILAR, t], t → intersection[x, P[cart[V, V]]]] // Reverse

Out[8]= subclass[intersection[x, P[cart[V, V]]], image[SIMILAR, x]] == True

In[9]:= subclass[intersection[x_, P[cart[V, V]]], image[SIMILAR, x_]] := True
```

Lemma.

```
In[10]:= SubstTest[subclass, image[t, x], range[t], t → SIMILAR] // Reverse

Out[10]= subclass[U[image[SIMILAR, x]], cart[V, V]] == True

In[11]:= subclass[U[image[SIMILAR, x_]], cart[V, V]] := True
```

---

derivation

Lemma.

```
In[12]:= Map[not, SubstTest[and, implies[p1, p3], implies[p3, p4],
            implies[and[p0, p2, p4], p5], not[implies[and[p0, p1, p2], p5]],
            {p0 → equal[y, composite[oopart[t], x, inverse[oopart[t]]]],
             p1 -> subclass[x, cartsq[domain[oopart[t]]]],
             p2 -> subclass[range[x], domain[x]], p3 -> subclass[range[x], domain[oopart[t]]],
             p4 → equal[domain[x], image[inverse[x], domain[oopart[t]]]],
             p5 -> subclass[range[y], domain[y]]}]] // Reverse

Out[12]= or[not[equal[y, composite[oopart[t], x, inverse[oopart[t]]]]], not[
            subclass[x, cart[domain[oopart[t]], intersection[domain[x], domain[oopart[t]]]]]],
           subclass[range[y], domain[y]]] == True

In[13]:= (% /. {x → x_, y → y_, t → t_}) /. Equal → SetDelayed
```

The next lemma helps cope with the following rewrite rule.

```
In[14]:= and[subclass[x, cart[y, y]], subclass[range[x], z]]

Out[14]= subclass[x, cart[y, intersection[y, z]]]
```

Lemma.

```
In[15]:= or[not[equal[y, composite[oopart[t], x, inverse[oopart[t]]]]],
           not[subclass[x, cart[domain[oopart[t]], domain[oopart[t]]]]],
           not[subclass[range[x], domain[x]]], subclass[range[y], domain[y]]] // NotNotTest

Out[15]= or[not[equal[y, composite[oopart[t], x, inverse[oopart[t]]]]],
           not[subclass[x, cart[domain[oopart[t]], domain[oopart[t]]]]],
           not[subclass[range[x], domain[x]]], subclass[range[y], domain[y]]] == True
```

```
In[16]:= (% /. {x → x_, y → y_, t → t_}) /. Equal → SetDelayed
```

Lemma. (Eliminate the wrapper **oopart**.)

```
In[17]:= SubstTest[implies, equal[t, oopart[z]], or[not[equal[y, composite[t, x, inverse[t]]]],
           not[subclass[x, cart[domain[t], domain[t]]]], not[subclass[range[x], domain[x]]],
           subclass[range[y], domain[y]]], z → t] // Reverse

Out[17]= or[not[equal[y, composite[t, x, inverse[t]]]], not[FUNCTION[t]],
           not[FUNCTION[inverse[t]]], not[subclass[x, cart[domain[t], domain[t]]]],
           not[subclass[range[x], domain[x]]], subclass[range[y], domain[y]]] == True
```

```
In[18]:= (% /. {x → x_, y → y_, t → t_}) /. Equal → SetDelayed
```

Theorem. Removing all three variables at the same time. (This takes a while.)

```
In[19]:= Map[empty,
          Map[composite[complement[#], id[cart[V, V]]] &, SubstTest[class, pair[pair[t, x], y],
            or[not[member[pair[pair[setpart[t], setpart[x]], setpart[y]], u]],
             not[member[setpart[x], v]], member[setpart[y], v]],
            {u -> composite[IMG, cross[composite[CROSS, DUP, id[BIJ]], Id],
               id[composite[inverse[S], CART, DUP, IMAGE[FIRST]]]],
             v -> image[inverse[DORA], inverse[S]]}]]]

Out[19]= subclass[image[DORA, image[SIMILAR, image[inverse[DORA], inverse[S]]]], inverse[S]] ==
          True
```

```
In[20]:= % /. Equal → SetDelayed
```

Theorem. Similarity preserves the condition **range ⊂ domain**.

```
In[21]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[SIMILAR, image[inverse[DORA], inverse[S]]],
           v → intersection[P[cart[V, V]], image[inverse[DORA], inverse[S]]]}]

Out[21]= equal[image[SIMILAR, image[inverse[DORA], inverse[S]]],
           intersection[image[inverse[DORA], inverse[S]], P[cart[V, V]]]] == True
```

```
In[22]:= image[SIMILAR, image[inverse[DORA], inverse[S]]] :=
          intersection[image[inverse[DORA], inverse[S]], P[cart[V, V]]]
```

Corollary. Similarity preserved the class of unary operations.

```
In[23]:= SubstTest[implies, and[equal[image[SIMILAR, x], x], equal[image[SIMILAR, y], y]],
           equal[image[SIMILAR, intersection[x, y]], intersection[x, y]], {x → FUNS,
            y -> intersection[image[inverse[DORA], inverse[S]], P[cart[V, V]]]}] // Reverse

Out[23]= equal[UNOPS, image[SIMILAR, UNOPS]] == True
```

```
In[24]:= image[SIMILAR, UNOPS] := UNOPS
```

permutations

In this section it is shown that a relation similar to a permutation is a permutation.

Lemma.

```
In[39]:= Map[image[SIMILAR, #] &, IminComp[DORA, INVERSE, S]] // Reverse

Out[39]= image[SIMILAR, image[INVERSE, image[inverse[DORA], S]]] ==
          intersection[image[inverse[DORA], inverse[S]], P[cart[V, V]]]

In[40]:= image[SIMILAR, image[INVERSE, image[inverse[DORA], S]]] :=
          intersection[image[inverse[DORA], inverse[S]], P[cart[V, V]]]
```

Lemma.

```
In[43]:= IminComp[DORA, INVERSE, inverse[S]] // Reverse

Out[43]= image[INVERSE, image[inverse[DORA], inverse[S]]] ==
          intersection[image[inverse[DORA], S], P[cart[V, V]]]

In[44]:= image[INVERSE, image[inverse[DORA], inverse[S]]] :=
          intersection[image[inverse[DORA], S], P[cart[V, V]]]
```

Theorem.

```
In[45]:= Map[image[INVERSE, #] &, ImageComp[SIMILAR, INVERSE, image[inverse[DORA], S]]]

Out[45]= image[SIMILAR, image[inverse[DORA], S]] ==
          intersection[image[inverse[DORA], S], P[cart[V, V]]]

In[46]:= image[SIMILAR, image[inverse[DORA], S]] :=
          intersection[image[inverse[DORA], S], P[cart[V, V]]]
```

Corollary.

```
In[47]:= SubstTest[implies, and[equal[image[SIMILAR, x], x], equal[image[SIMILAR, y], y]],
            equal[image[SIMILAR, intersection[x, y]], intersection[x, y]],
            {x → intersection[image[inverse[DORA], S], P[cart[V, V]]],
             y -> intersection[image[inverse[DORA], inverse[S]], P[cart[V, V]]]}] // Reverse

Out[47]= equal[image[SIMILAR, image[inverse[DORA], Id]],
            intersection[image[inverse[DORA], Id], P[cart[V, V]]]] == True

In[49]:= image[SIMILAR, image[inverse[DORA], Id]] :=
          intersection[image[inverse[DORA], Id], P[cart[V, V]]]
```

Corollary.

```
In[51]:= SubstTest[implies, and[equal[image[SIMILAR, x], x], equal[image[SIMILAR, y], y]],
           equal[image[SIMILAR, intersection[x, y]], intersection[x, y]],
           {x → intersection[image[inverse[DORA], Id], P[cart[V, V]]], y → BIJ}] // Reverse

Out[51]= equal[PERMS, image[SIMILAR, PERMS]] == True

In[53]:= image[SIMILAR, PERMS] := PERMS
```

The following could also have been used to derive this result.

Theorem.

```
In[56]:= intersection[UNOPS, image[INVERSE, UNOPS]] // Renormality

Out[56]= intersection[UNOPS, image[INVERSE, UNOPS]] == PERMS

In[57]:= intersection[UNOPS, image[INVERSE, UNOPS]] := PERMS
```