transforms of quasiorders

Johan G. F. Belinfante
2013 December 7

```
In[1]:= SetDirectory["l:"]; << goedel.13dec06a

        :Package Title: goedel.13dec06a                 2013 December 6 at 5:45 p.m.

        Loading takes about seventeen minutes, half that time due to builtin pauses.

        It is now:  2013 Dec 7 at 5:51

        Loading Simplification Rules

        TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

        weightlimit = 40

        Loading completed.

        It is now:  2013 Dec 7 at 6:8
```

summary

If **q** is a quasiorder and if **f** is a function, then **inverse[f] ∘ q ∘ f** is a quasiorder. In particular, any relation similar to a quasiorder is a quasiorder.

derivation

Theorem. (Introduce wrappers.) The transform of a reflexive relation is reflexive.

```
In[5]:= SubstTest[implies, and[FUNCTION[u], REFLEXIVE[x]],
          REFLEXIVE[composite[inverse[u], x, u]], u → funpart[y]] // Reverse

Out[5]= or[not[REFLEXIVE[x]], REFLEXIVE[composite[inverse[funpart[y]], x, funpart[y]]]] == True

In[6]:= or[not[REFLEXIVE[x_]],
          REFLEXIVE[composite[inverse[funpart[y_]], x_, funpart[y_]]]] := True
```

Theorem. A special case. The transform of a quasiorder is reflexive.

```
In[7]:= SubstTest[implies, and[FUNCTION[u], REFLEXIVE[v]],
          REFLEXIVE[composite[inverse[u], v, u]], {u → funpart[x], v → trv[rfx[y]]}] // Reverse

Out[7]= REFLEXIVE[composite[inverse[funpart[x]], trv[rfx[y]], funpart[x]]] == True

In[8]:= REFLEXIVE[composite[inverse[funpart[x_]], trv[rfx[y_]], funpart[x_]]] := True
```

Corollary. (Remove wrappers.) The transform of a quasiorder is a quasiorder.

```
In[9]:= SubstTest[implies, and[equal[x, funpart[u]], equal[y, trv[rfx[v]]]],
          QUASIORDER[composite[inverse[x], y, x]], {u → x, v → y}] // Reverse

Out[9]= or[not[FUNCTION[x]], not[QUASIORDER[y]], QUASIORDER[composite[inverse[x], y, x]]] == True

In[10]:= or[not[FUNCTION[x_]], not[QUASIORDER[y_]],
           QUASIORDER[composite[inverse[x_], y_, x_]]] := True
```

variable-free rules

Lemma.

```
In[19]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t -> composite[IMG, id[cart[image[CROSS, id[image[INVERSE, FUNS]]], V]],
              inverse[SECOND]], u → QO, v → RFX}] // Reverse

Out[19]= subclass[image[IMG, cart[image[CROSS, id[image[INVERSE, FUNS]]], QO]], RFX] == True

In[20]:= % /. Equal → SetDelayed
```

Lemma.

```
In[21]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t -> composite[IMG, id[cart[image[CROSS, id[image[INVERSE, FUNS]]], V]],
              inverse[SECOND]], u → QO, v → TRV}] // Reverse

Out[21]= subclass[image[IMG, cart[image[CROSS, id[image[INVERSE, FUNS]]], QO]], TRV] == True

In[22]:= % /. Equal → SetDelayed
```

Lemma.

```
In[24]:= SubstTest[subclass, t, intersection[u, v],
           {t -> image[IMG, cart[image[CROSS, id[image[INVERSE, FUNS]]], QO]],
            u → RFX, v → TRV}] // Reverse

Out[24]= subclass[image[IMG, cart[image[CROSS, id[image[INVERSE, FUNS]]], QO]], QO] == True

In[25]:= % /. Equal → SetDelayed
```

Lemma.

```
In[31]:= Map[equal[V, #] &,
          dif[cart[image[CART, Id], QO], image[inverse[CAP], QO]] // complement // Normality]

Out[31]= subclass[image[CAP, cart[image[CART, Id], QO]], QO] == True

In[32]:= % /. Equal → SetDelayed
```

Theorem. Restrictions of quasiorders are quasiorders.

```
In[33]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[CAP, cart[image[CART, Id], QO]], v -> QO}]

Out[33]= equal[QO, image[CAP, cart[image[CART, Id], QO]]] == True

In[35]:= image[CAP, cart[image[CART, Id], QO]] := QO
```

Lemma.

```
In[36]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t -> composite[IMG, id[cart[V, QO]], inverse[FIRST], CROSS, DUP, INVERSE],
            u → P[Id], v → FUNS}] // Reverse

Out[36]= subclass[QO, image[IMG, cart[image[CROSS, id[image[INVERSE, FUNS]]], QO]]] == True

In[37]:= % /. Equal → SetDelayed
```

Theorem.

```
In[38]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[IMG, cart[image[CROSS, id[image[INVERSE, FUNS]]], QO]], v -> QO}]

Out[38]= equal[QO, image[IMG, cart[image[CROSS, id[image[INVERSE, FUNS]]], QO]]] == True

In[40]:= image[IMG, cart[image[CROSS, id[image[INVERSE, FUNS]]], QO]] := QO

In[46]:= abstract[x, image[IMG, cart[image[CROSS, id[x]], QO]]]

Out[46]= composite[IMG, id[cart[V, QO]], inverse[FIRST], CROSS, DUP]
```

Lemma.

```
In[48]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t -> composite[IMG, id[cart[V, QO]], inverse[FIRST], CROSS, DUP],
            u → BIJ, v → image[INVERSE, FUNS]}] // Reverse

Out[48]= subclass[image[IMG, cart[image[CROSS, id[BIJ]], QO]], QO] == True

In[49]:= % /. Equal → SetDelayed
```

Theorem.

```
In[50]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[IMG, cart[image[CROSS, id[BIJ]], QO]], v -> QO}]

Out[50]= equal[QO, image[IMG, cart[image[CROSS, id[BIJ]], QO]]] == True

In[52]:= image[IMG, cart[image[CROSS, id[BIJ]], QO]] := QO
```

Lemma.

```
In[53]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
           {u → SIMILAR, v → composite[IMG, id[cart[image[CROSS, id[BIJ]], V]], inverse[SECOND]],
            w → QO}] // Reverse

Out[53]= subclass[image[SIMILAR, QO], QO] == True

In[54]:= % /. Equal → SetDelayed
```

Theorem. Any relation similar to a quasiorder is a quasiorder.

```
In[55]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> image[SIMILAR, QO], v -> QO}]

Out[55]= equal[QO, image[SIMILAR, QO]] == True

In[57]:= image[SIMILAR, QO] := QO
```