equivalence relations on a given set

Johan G. F. Belinfante
2008 November 6

```
In[1]:= SetDirectory["l:"]; << goedel.08nov06a; << tools.m

        :Package Title: goedel.08nov06a          2008 November 6 at 12:00 noon

        It is now:  2008 Nov 6 at 15:32

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 October 21

        weightlimit = 40
```

summary

The class of equivalence relations on a given set **x** is a set which is closed under arbitrary intersections. A similar statement holds for partial orders.

fix[HULL[image[inverse[IMAGE[inverse[DUP]]], set[x]]]]

The class of sets with a given fixed point set is closed under arbitrary intersections. The derivation in this section is based on an analogy with a similar result derived in September 2003 for the class **binclosed[x]**. The key result needed for this is the following:

```
In[2]:= implies[subclass[image[inverse[S], x], image[inverse[HULL[x]], x]],
          equal[fix[HULL[x]], x]]

Out[2]= True
```

The starting point is the following observation:

```
In[3]:= SubstTest[implies, and[member[t, u], subclass[u, v]], member[t, v],
           {v -> image[inverse[IMAGE[inverse[DUP]]], set[x]]}] // Reverse

Out[3]= or[equal[x, fix[t]], not[member[t, u]],
           not[subclass[image[IMAGE[inverse[DUP]], u], set[x]]]] == True

In[4]:= (% /. {t → t_, u → u_, x → x_}) /. Equal → SetDelayed
```

Eliminating the variable **t** yields:

```
In[5]:= Map[or[subclass[y, fix[A[x]]], equal[V, #]] &,
         SubstTest[class, t, or[equal[y, fix[t]], not[member[t, x]], not[subclass[z, set[y]]]],
          z -> image[IMAGE[inverse[DUP]], x]]]
```

```
Out[5]= or[not[subclass[image[IMAGE[inverse[DUP]], x], set[y]]], subclass[y, fix[A[x]]]] == True
```

```
In[6]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

A similar inclusion in the opposite direction is obtained, but with **A[x]** replaced by **U[x]**.

```
In[7]:= Map[equal[V, #] &,
          SubstTest[class, t, or[equal[y, fix[t]], not[member[t, x]], not[subclass[z, set[y]]]],
           z -> image[IMAGE[inverse[DUP]], x]]] // MapNotNot
```

```
Out[7]= or[not[subclass[image[IMAGE[inverse[DUP]], x], set[y]]], subclass[fix[U[x]], y]] == True
```

```
In[8]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The following lemma is needed to relate the **U[x]** result to a corresponding **A[x]** result.

```
In[9]:= SubstTest[implies, subclass[u, v],
          subclass[fix[u], fix[v]], {u → A[x], v → U[x]}] // Reverse
```

```
Out[9]= or[equal[0, x], subclass[fix[A[x]], fix[U[x]]]] == True
```

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

Lemma.

```
In[11]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4],
            not[implies[and[p1, p2], p4]], {p1 → not[empty[x]], p2 -> subclass[fix[U[x]], y],
             p3 -> subclass[fix[A[x]], fix[U[x]]], p4 -> subclass[fix[A[x]], y]}]] // Reverse
```

```
Out[11]= or[equal[0, x], not[subclass[fix[U[x]], y]], subclass[fix[A[x]], y]] == True
```

```
In[12]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. The class **U[x]** is replaced with **A[x]**, and two inclusions are combined into an equation. (Comment: The following step has been deliberately omitted to expedite the derivation: **implies[and[p3, p5], p6]**.)

```
In[13]:= Map[not, SubstTest[and, implies[p2, p3], implies[p2, p4],
            implies[and[p1, p4], p5], not[implies[and[p1, p2], p6]],
            {p1 → not[empty[x]], p2 -> subclass[image[IMAGE[inverse[DUP]], x], set[y]],
             p3 -> subclass[y, fix[A[x]]], p4 -> subclass[fix[U[x]], y],
             p5 -> subclass[fix[A[x]], y], p6 → equal[fix[A[x]], y]}]] // Reverse
```

```
Out[13]= or[equal[0, x], equal[y, fix[A[x]]],
           not[subclass[image[IMAGE[inverse[DUP]], x], set[y]]]] == True
```

```
In[14]:= or[equal[0, x_], equal[y_, fix[A[x_]]],
           not[subclass[image[IMAGE[inverse[DUP]], x_], set[y_]]]] := True
```

Corollary. This is done by analogy with the 2003 derivation, replacing **binclosed[x]** with **image[inverse[IMAGE[inverse[-DUP]]],set[x]]**.

```
In[15]:= SubstTest[implies, subclass[w, image[inverse[IMAGE[inverse[DUP]]], set[x]]],
           or[equal[0, w], member[A[w], image[inverse[IMAGE[inverse[DUP]]], set[x]]]],
          w -> intersection[image[inverse[IMAGE[inverse[DUP]]], set[x]], image[S, set[y]]]] //
          Reverse

Out[15]= or[equal[x, fix[hull[image[inverse[IMAGE[inverse[DUP]]], set[x]], y]]],
           not[member[x, V]], not[member[y, V]], not[subclass[fix[y], x]]] == True

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

The following temporary rewrite rule is needed to proceed.

```
In[17]:= SubstTest[member, pair[y, x], inverse[t],
          t -> composite[inverse[S], inverse[IMAGE[inverse[DUP]]]]]

Out[17]= member[pair[x, y], composite[inverse[S], inverse[IMAGE[inverse[DUP]]]]] ==
          and[member[x, V], member[y, V], subclass[fix[y], x]]

In[18]:= member[pair[x_, y_], composite[inverse[S], inverse[IMAGE[inverse[DUP]]]]] :=
          and[member[x, V], member[y, V], subclass[fix[y], x]]
```

The variable **y** is eliminated the same way that was done in 2003.

```
In[19]:= Map[equal[V, #] &,
          SubstTest[class, y, implies[member[y, domain[HULL[z]]], member[hull[z, y], z]],
           z -> image[inverse[IMAGE[inverse[DUP]]], set[x]]]]

Out[19]= subclass[image[inverse[S], image[inverse[IMAGE[inverse[DUP]]], set[x]]],
           image[inverse[HULL[image[inverse[IMAGE[inverse[DUP]]], set[x]]]],
            image[inverse[IMAGE[inverse[DUP]]], set[x]]]] == True

In[20]:= (% /. x → x_) /. Equal → SetDelayed
```

A general lemma is now applied to obtain the main result: The class of sets with a given fixed-point set is closed under arbitrary intersections.

```
In[21]:= SubstTest[implies, subclass[image[inverse[S], y], image[inverse[HULL[y]], y]],
           equal[fix[HULL[y]], y], y -> image[inverse[IMAGE[inverse[DUP]]], set[x]]] // Reverse

Out[21]= equal[fix[HULL[image[inverse[IMAGE[inverse[DUP]]], set[x]]]],
           image[inverse[IMAGE[inverse[DUP]]], set[x]]] == True

In[22]:= fix[HULL[image[inverse[IMAGE[inverse[DUP]]], set[x_]]]] :=
          image[inverse[IMAGE[inverse[DUP]]], set[x]]
```

Corollary. (A slightly weaker result, which would have sufficed for the application in the next section.)

```
In[23]:= SubstTest[Aclosure, fix[HULL[t]],
           t -> image[inverse[IMAGE[inverse[DUP]]], set[x]]] // Reverse

Out[23]= Aclosure[image[inverse[IMAGE[inverse[DUP]]], set[x]]] ==
          image[inverse[IMAGE[inverse[DUP]]], set[x]]

In[24]:= Aclosure[image[inverse[IMAGE[inverse[DUP]]], set[x_]]] :=
          image[inverse[IMAGE[inverse[DUP]]], set[x]]
```

a sethood result

In this section it is shown that the class of equivalence relations with a given fixed point set is a set. The same derivation can be adapted to other types of relations.

Lemma.

```
In[25]:= SubstTest[implies, empty[t],
           member[intersection[image[inverse[IMAGE[inverse[DUP]]], t], P[cart[x, x]]], V],
           t → set[x]] // Reverse

Out[25]= or[member[x, V], member[
             intersection[image[inverse[IMAGE[inverse[DUP]]], set[x]], P[cart[x, x]]], V]] == True

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

A redundant sethood literal can be eliminated.

```
In[27]:= SubstTest[and, implies[p, q], or[p, q], {p -> member[x, V], q -> member[
             intersection[image[inverse[IMAGE[inverse[DUP]]], set[x]], P[cart[x, x]]], V]}]

Out[27]= member[intersection[image[inverse[IMAGE[inverse[DUP]]], set[x]], P[cart[x, x]]], V] ==
          True

In[28]:= member[intersection[
            image[inverse[IMAGE[inverse[DUP]]], set[x_]], P[cart[x_, x_]]], V] := True
```

Lemma.

```
In[29]:= Map[equal[V, #] &, SubstTest[class, t,
           implies[member[t, u], implies[member[t, v], subclass[t, cart[x, x]]]],
           {u → EQV, v -> image[inverse[IMAGE[inverse[DUP]]], set[x]]}]]

Out[29]= subclass[
           U[intersection[EQV, image[inverse[IMAGE[inverse[DUP]]], set[x]]]], cart[x, x]] == True

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

Theorem. The class of equivalence relations on **x** is a set.

```
In[31]:= SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
           {u -> intersection[image[inverse[IMAGE[inverse[DUP]]], set[x]], EQV],
            v -> intersection[image[inverse[IMAGE[inverse[DUP]]], set[x]],
              P[cart[x, x]]]}] // Reverse

Out[31]= member[intersection[EQV, image[inverse[IMAGE[inverse[DUP]]], set[x]]], V] == True

In[32]:= member[intersection[EQV, image[inverse[IMAGE[inverse[DUP]]], set[x_]]], V] := True
```

comments

The class of equivalence relations on **x** is closed under arbitrary intersections. One can state this in two equivalent ways. No new rewrite rules are needed.

```
In[33]:= Aclosure[intersection[EQV, image[inverse[IMAGE[inverse[DUP]]], set[x]]]]

Out[33]= intersection[EQV, image[inverse[IMAGE[inverse[DUP]]], set[x]]]

In[34]:= fix[HULL[intersection[EQV, image[inverse[IMAGE[inverse[DUP]]], set[x]]]]]

Out[34]= intersection[EQV, image[inverse[IMAGE[inverse[DUP]]], set[x]]]
```

a sethood theorem for partial orders

```
In[35]:= Map[equal[V, #] &, SubstTest[class, t,
           implies[member[t, u], implies[member[t, v], subclass[t, cart[x, x]]]],
           {u → PO, v -> image[inverse[IMAGE[inverse[DUP]]], set[x]]}]]

Out[35]= subclass[
           U[intersection[PO, image[inverse[IMAGE[inverse[DUP]]], set[x]]]], cart[x, x]] == True

In[36]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. The class of partial order relations on **x** is a set.

```
In[37]:= SubstTest[implies, and[subclass[u, v], member[v, V]], member[u, V],
           {u -> intersection[image[inverse[IMAGE[inverse[DUP]]], set[x]], PO],
            v -> intersection[image[inverse[IMAGE[inverse[DUP]]], set[x]],
              P[cart[x, x]]]}] // Reverse

Out[37]= member[intersection[PO, image[inverse[IMAGE[inverse[DUP]]], set[x]]], V] == True

In[38]:= member[intersection[PO, image[inverse[IMAGE[inverse[DUP]]], set[x_]]], V] := True
```

Corollary. The class of partial orders on a given set **x** is closed under arbitrary intersections. No new rewrite rules are needed for this.

```
In[39]:= Aclosure[intersection[PO, image[inverse[IMAGE[inverse[DUP]]], set[x]]]]

Out[39]= intersection[PO, image[inverse[IMAGE[inverse[DUP]]], set[x]]]

In[40]:= fix[HULL[intersection[PO, image[inverse[IMAGE[inverse[DUP]]], set[x]]]]]

Out[40]= intersection[PO, image[inverse[IMAGE[inverse[DUP]]], set[x]]]
```