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.

\[ \text{fix[HULL[image[inverse[IMAGE[inverse[DUP]]], set[x]]]]} \]

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 \( \text{binclosed}[x] \). The key result needed for this is the following:

\[
\text{In[2]:= implies[subclass[image[inverse[S], x], image[inverse[HULL[x]], x]], equal[fix[HULL[x]], x]]}
\]

\[
\text{Out[2]= True}
\]

The starting point is the following observation:

\[
\text{In[3]:= SubstTest[implies, and[member[t, u], subclass[u, v]], member[t, v],}
\]

\[
\{v -> image[inverse[IMAGE[inverse[DUP]]], set[x]]\}] // Reverse
\]

\[
\text{Out[3]= or[equal[x, fix[t]], not[member[t, u]],}
\]

\[
\text{not[subclass[image[IMAGE[inverse[DUP]]], u], set[x]]}] = True
\]

\[
\text{In[4]:= (\% /. \{t -> \_, u -> \_, x -> \_\}) /. Equal \rightarrow SetDelayed}
\]

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]], notsubclass[z, set[y]]]], 
   z -> image[IMAGE[inverse[DUP]], x]]

Out[5]= or[notsubclass[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]], notsubclass[z, set[y]]]], 
   z -> image[IMAGE[inverse[DUP]], x]] // MapNotNot

Out[7]= or[notsubclass[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], notsubclass[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]]], 
   notsubclass[image[IMAGE[inverse[DUP]], x], set[y]]] := True

In[14]:= or[equal[0, x_], equal[y_, fix[A[x_]]], 
   notsubclass[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]]].`

\[ \text{Corollary. (A slightly weaker result, which would have sufficed for the application in the next section.)} \]
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.
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.

**A sethood theorem for partial orders**

The class of partial order relations on \( x \) is a set.

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]]