Aclosure[binclosed[x]]

Johan G. F. Belinfante
2003 May 21

```
In[1]:= << goedel52.r79; << tools.m

        :Package Title: goedel52.r79        2003 May 21 at 1:00 p.m.

        It is now:  2003 May 22 at 12:47

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 May 21

        weightlimit = 40
```

■ summary

This notebook is concerned with the class **binclosed[x]** of sets that are closed under a binary operation **x**. It is shown that this class is closed under arbitrary intersections. A corollary is that the intersection of any (nonempty) set of topologies is a topology.

■ definition of binclosed[x]

The class of all sets closed under a binary operation **x** is formally defined by the membership rule

```
In[2]:= member[y_, binclosed[x_]] := and[member[y, V], subclass[image[x, cart[y, y]], y]]
```

It is to be emphasized that no assumption is made concerning the nature of the class **x**. This definition resembles that of the class of sets invariant under an operation **x**,

```
In[3]:= member[y, invar[x]]

Out[3]= and[member[y, V], subclass[image[x, y], y]]
```

The class **invar[x]** can if fact be viewed as just a particular case:

```
In[4]:= binclosed[composite[x, inverse[DUP]]] // Normality

Out[4]= binclosed[composite[x, inverse[DUP]]] == invar[x]

In[5]:= binclosed[composite[x_, inverse[DUP]]] := invar[x]
```

In view of this, one may reasonably expect that at least some of the properties of the class **invar[x]** can be extended to the class **binclosed[x]**.

■ special cases

In earlier work, these three special cases were studied and given special names:

```
In[6]:= binclosed[CAP] // Normality
```

```
Out[6]= binclosed[CAP] == CAPclosed
```

```
In[7]:= binclosed[CAP] := CAPclosed

        binclosed[CUP] // Normality

        binclosed[CUP] == CUPclosed
```

```
In[8]:= binclosed[CUP] := CUPclosed
```

The third case requires a different technique:

```
In[9]:= symdif[DIFclosed, binclosed[DIF]] // Normality
```

```
Out[9]= union[intersection[DIFclosed, complement[binclosed[DIF]]],
          intersection[binclosed[DIF], complement[DIFclosed]]] == 0
```

```
In[10]:= union[intersection[DIFclosed, complement[binclosed[DIF]]],
           intersection[binclosed[DIF], complement[DIFclosed]]] := 0
```

```
In[11]:= SubstTest[equal, 0, symdif[x, y], {x -> DIFclosed, y -> binclosed[DIF]}]
```

```
Out[11]= True == equal[DIFclosed, binclosed[DIF]]
```

```
In[12]:= binclosed[DIF] := DIFclosed
```

The following example involves the rotation−invariant function **RIF** which converts cartesian products to composites.

```
In[13]:= binclosed[composite[SWAP, RIF]] // Normality
```

```
Out[13]= binclosed[composite[SWAP, RIF]] == image[inverse[IMAGE[id[cart[V, V]]]], TRV]
```

```
In[14]:= binclosed[composite[SWAP, RIF]] := image[inverse[IMAGE[id[cart[V, V]]]], TRV]
```

This is the class of sets **x** for which **composite[Id,x]** is a transitivive relation.

■ normalization concerns

A general formula for the class **binclosed[x]** can be derived by the symmetric diffence technique similar to that illustrated in the preceding section:

```
In[15]:= Map[equal[0, #] &,
           symdif[binclosed[x], fix[composite[S, IMAGE[x], CART, DUP]]] // Renormality]
```

```
Out[15]= equal[binclosed[x], fix[composite[S, IMAGE[x], CART, DUP]]] == True
```

```
In[16]:= fix[composite[S, IMAGE[x_], CART, DUP]] := binclosed[x]
```

To improve the performance of **Normality** tests, it helps to add the following simplification rule:

```
In[17]:= binclosed[x] // Normality // Reverse

Out[17]= intersection[binclosed[x], cliques[domain[VERTSECT[x]]]] == binclosed[x]

In[18]:= intersection[binclosed[x_], cliques[domain[VERTSECT[x_]]]] := binclosed[x]
```

The following rewrite rule is closely related:

```
In[19]:= SubstTest[equal, intersection[u, v], u,
           {u -> binclosed[x], v -> cliques[domain[VERTSECT[x]]]}] // Reverse

Out[19]= subclass[binclosed[x], cliques[domain[VERTSECT[x]]]] == True

In[20]:= subclass[binclosed[x_], cliques[domain[VERTSECT[x_]]]] := True
```

With these rewrite rules in place, the following expression simplifies as one would expect:

```
In[21]:= class[y, subclass[image[x, cart[y, y]], y]]

Out[21]= binclosed[x]
```

■ the Aclosure theorem

In this section it will be shown that the class **binclosed[x]** is closed under arbitrary intersections. Some lemmas are needed. Here is one:

```
In[22]:= SubstTest[implies, and[member[y, x], subclass[x, z]],
           member[y, z], z -> binclosed[w]] // MapNotNot

Out[22]= or[not[member[y, x]], not[subclass[x, binclosed[w]]],
           subclass[image[w, cart[y, y]], y]] == True

In[23]:= or[not[member[y_, x_]], not[subclass[x_, binclosed[w_]]],
           subclass[image[w_, cart[y_, y_]], y_]] := True
```

Cartesian squares satisfy a monotonicity property, which implies:

```
In[24]:= SubstTest[implies, subclass[u, v],
          subclass[image[x, u], image[x, v]], {u -> cart[y, y], v -> cart[z, z]}]

Out[24]= or[not[subclass[y, z]], subclass[image[x, cart[y, y]], image[x, cart[z, z]]]] == True

In[25]:= or[not[subclass[y_, z_]],
           subclass[image[x_, cart[y_, y_]], image[x_, cart[z_, z_]]]] := True
```

These lemmas enter into the following deduction:

```
In[26]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4],
           implies[and[p2, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 -> member[y, z], p2 -> subclass[image[x, cart[y, y]], y], p3 -> subclass[A[z], y],
                 p4 -> subclass[image[x, cart[A[z], A[z]]], image[x, cart[y, y]]],
            p5 -> subclass[image[x, cart[A[z], A[z]]], y]}]]

Out[26]= or[not[member[y, z]], not[subclass[image[x, cart[y, y]], y]],
           subclass[image[x, cart[A[z], A[z]]], y]] == True
```

```
In[27]:= or[not[member[y_, z_]], not[subclass[image[x_, cart[y_, y_]], y_]],
           subclass[image[x_, cart[A[z_], A[z_]]], y_]] := True

In[28]:= Map[equal[V, #] &,
           SubstTest[class, y, implies[and[member[y, z], subclass[image[u, cart[y, y]], y]],
                 subclass[v, y]], {u -> x, v -> image[x, cart[A[z], A[z]]]}] // Reverse

Out[28]= subclass[image[x, cart[A[z], A[z]]], A[intersection[z, binclosed[x]]]] == True

In[29]:= subclass[image[x_, cart[A[z_], A[z_]]], A[intersection[z_, binclosed[x_]]]] := True

In[30]:= SubstTest[implies, equal[y, z], equal[A[y], A[z]], y -> intersection[binclosed[x], z]]

Out[30]= or[equal[A[z], A[intersection[z, binclosed[x]]]],
           not[subclass[z, binclosed[x]]]] == True

In[31]:= or[equal[A[z_], A[intersection[z_, binclosed[x_]]]],
           not[subclass[z_, binclosed[x_]]]] := True

In[32]:= SubstTest[implies, and[subclass[u, v], equal[v, w]], subclass[u, w],
             {u -> image[x, cart[A[z], A[z]]], v -> A[intersection[binclosed[x], z]], w -> A[z]}]

Out[32]= or[not[equal[A[z], A[intersection[z, binclosed[x]]]]],
           subclass[image[x, cart[A[z], A[z]]], A[z]]] == True

In[33]:= or[not[equal[A[z_], A[intersection[z_, binclosed[x_]]]]],
           subclass[image[x_, cart[A[z_], A[z_]]], A[z_]]] := True

In[34]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 -> subclass[z, binclosed[x]],
            p2 -> equal[A[z], A[intersection[binclosed[x], z]]],
             p3 -> subclass[image[x, cart[A[z], A[z]]], A[z]]}]]

Out[34]= or[not[subclass[z, binclosed[x]]], subclass[image[x, cart[A[z], A[z]]], A[z]]] == True

In[35]:= or[not[subclass[z_, binclosed[x_]]],
           subclass[image[x_, cart[A[z_], A[z_]]], A[z_]]] := True
```

A negative variant of this is needed:

```
In[36]:= and[not[subclass[image[x, cart[A[z], A[z]]], A[z]]],
           subclass[z, binclosed[x]]] // NotNotTest

Out[36]= and[not[subclass[image[x, cart[A[z], A[z]]], A[z]]], subclass[z, binclosed[x]]] == False

In[37]:= and[not[subclass[image[x_, cart[A[z_], A[z_]]], A[z_]]],
           subclass[z_, binclosed[x_]]] := False

In[38]:= intersection[P[binclosed[x]], complement[singleton[0]],
           complement[image[inverse[BIGCAP], binclosed[x]]]] // Normality

Out[38]= intersection[complement[image[inverse[BIGCAP], binclosed[x]]],
           complement[singleton[0]], P[binclosed[x]]] == 0

In[39]:= intersection[complement[image[inverse[BIGCAP], binclosed[w_]]],
           complement[singleton[0]], P[binclosed[w_]]] := 0

In[40]:= SubstTest[equal, 0, dif[u, v], {u -> dif[P[binclosed[x]], singleton[0]],
             v -> image[inverse[BIGCAP], binclosed[x]]}] // Reverse

Out[40]= subclass[P[binclosed[x]],
           union[image[inverse[BIGCAP], binclosed[x]], singleton[0]]] == True
```

```
In[41]:= subclass[P[binclosed[x_]],
           union[image[inverse[BIGCAP], binclosed[x_]], singleton[0]]] := True

In[42]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
            {u -> P[binclosed[x]],
           v -> union[image[inverse[BIGCAP], binclosed[x]], singleton[0]], w -> BIGCAP}]

Out[42]= subclass[Aclosure[binclosed[x]], binclosed[x]] == True

In[43]:= subclass[Aclosure[binclosed[x_]], binclosed[x_]] := True
```

The inclusion in the other direction is obvious, and hence an equation is otained:

```
In[44]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> Aclosure[binclosed[x]], v -> binclosed[x]}] // Reverse

Out[44]= equal[Aclosure[binclosed[x]], binclosed[x]] == True
```

This says that **binclosed[x]** is closed under arbitrary intersections.

```
In[45]:= Aclosure[binclosed[x_]] := binclosed[x]
```

This result is an extension of a corresponding result about **invar[x]**.

■ a corollary for topology

An immediate corollary is that the intersection of any class of topologies is a topology.

```
In[46]:= SubstTest[Aclosure, binclosed[x], x -> CAP]

Out[46]= Aclosure[CAPclosed] == CAPclosed

In[47]:= Aclosure[CAPclosed] := CAPclosed
```

The intersection of two Aclosed classes is Aclosed:

```
In[48]:= SubstTest[implies, and[equal[x, Aclosure[x]], equal[y, Aclosure[y]]],
          equal[intersection[x, y], Aclosure[intersection[x, y]]],
             {x -> fix[UCLOSURE], y -> CAPclosed}]

Out[48]= equal[TOPS, Aclosure[TOPS]] == True
```

This succinct formula says that the class **TOPS** of all topologies is closed under arbitrary intersections.

```
In[49]:= Aclosure[TOPS] := TOPS
```

One can also reformulate this using variables as follows:

```
In[53]:= SubstTest[implies, and[member[x, P[y]], not[equal[x, 0]]],
          member[A[x], Aclosure[y]], y → TOPS]

Out[53]= or[and[equal[A[x], image[CAP, cart[A[x], A[x]]]], equal[A[x], Uclosure[A[x]]],
           equal[0, x], not[member[x, V]], not[subclass[x, TOPS]]] == True

In[54]:= or[and[equal[A[x_], image[CAP, cart[A[x_], A[x_]]]], equal[A[x_], Uclosure[A[x_]]],
           equal[0, x_], not[member[x_, V]], not[subclass[x_, TOPS]]] := True
```

This says that if **x** is a nonempty set of topologies, then its intersection **A[x]** is also a topology:

```
In[55]:= implies[and[member[x, V], not[equal[x, 0]], subclass[x, TOPS]], member[A[x], TOPS]]
Out[55]= True
```