ac1 => a8

Johan G. F. Belinfante
2004 November 13

```
In[1]:= SetDirectory["i:"]; << goedel63.12b; << tools.m
```

```
:Package Title: goedel63.12b        2004 November 12 at 9:30 a.m.

It is now:  2004 Nov 13 at 15:5

Loading Simplification Rules

TOOLS.M                       Revised 2004 November 2

weightlimit = 40
```

---

summary

It is shown in this notebook that **ac1 => a8**. These are two set versions of the axiom of choice. The axiom **ac1** says that any set of nonempty sets has a choice function, while axiom **a8** says that for any pairwise disjoint set of nonempty sets, there is a selecting set whose intersection with each member of pairwise disjoint collection is a singleton. The derivation presented below is based on Jean Rubin's remark that one can take the selecting set to be the range of a choice function.

```
In[2]:= "Jean E. Rubin, Set Theory for the Mathematician,
           Holden-Day, San Francisco, 1967. (see bottom of page 78)";
```

In addition, a general statement is derived that holds independently of whether the axiom of choice is valid.

choice processes

Any subclass **z** of **inverse[E]** can be regarded as a choice process in which more than one choice is permitted. The condition **member[v, image[inverse[z], u]]** can be interpreted as saying that **v** is one of the sets from which and element of **u** has been selected by this process. In this section it is shown that if this is the case, then **u** and **v** are not disjoint. The following lemma just serves to introduce variables into the statement **subclass[z, inverse[E]]**.

```
In[3]:= Map[or[#, member[v, u]] &, SubstTest[implies, and[member[s, z], subclass[z, w]],
          member[s, w], {s → pair[u, v], w → inverse[E]}]]

Out[3]= or[member[v, u], not[member[pair[u, v], z]],
          not[subclass[z, inverse[E]]]] == True

In[4]:= (% /. {u → u_, v → v_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[5]:= Map[not, SubstTest[and, implies[and[p1, p2, p3, p4], p5],
          implies[and[p3, p5], p6], not[implies[and[p1, p2, p3, p4], p6]],
          {p1 → member[v, V], p2 → subclass[z, inverse[E]], p3 -> member[t, u],
           p4 → member[pair[v, t], z], p5 → member[t, v], p6 → not[disjoint[u, v]]}]]

Out[5]= or[not[equal[0, intersection[u, v]]], not[member[t, u]], not[member[v, V]],
          not[member[pair[v, t], z]], not[subclass[z, inverse[E]]]] == True

In[6]:= (% /. {t → t_, u → u_, v → v_, z → z_}) /. Equal → SetDelayed
```

Theorem: If chosing from **v** produces an element of **u**, then the sets **u** and **v** are not disjoint.

```
In[7]:= Map[equal[V, #] &, SubstTest[class, t,
           or[not[equal[0, intersection[u, v]]], not[member[t, u]], not[member[v, V]],
            not[member[pair[v, t], z]], not[subclass[z, w]]], w → inverse[E]]] // Reverse

Out[7]= or[not[equal[0, intersection[u, v]]],
          not[member[v, image[inverse[z], u]]], not[subclass[z, inverse[E]]]] == True

In[8]:= (% /. {u → u_, v → v_, z → z_}) /. Equal → SetDelayed
```

pairwise disjointness

This first step introduces variables into the statement that **x** is a pairwise disjoint collection:

```
In[9]:= SubstTest[implies, and[member[r, s], subclass[s, t]],
         member[r, t], {r → pair[u, v], s → cart[x, x], t → union[DISJOINT, Id]}]

Out[9]= or[equal[0, intersection[u, v]], equal[u, v], not[member[u, x]],
          not[member[v, x]], not[subclass[cart[x, x], union[DISJOINT, Id]]]] == True

In[10]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

Lemma. When a choice process is applied to a pairwise disjoint collection of sets, the only set from which an element of a member **u** of the pairwise disjoint collection can be selected is **u** itself.

```
In[11]:= Map[not, SubstTest[and, implies[and[p1, p4], p5],
           implies[p4, p6], implies[and[p2, p3, p5, p6], p7],
           not[implies[and[p1, p2, p3, p4], p7]], {p1 → subclass[z, inverse[E]],
            p2 → subclass[cart[domain[z], domain[z]], union[DISJOINT, Id]],
            p3 → member[u, domain[z]],
            p4 → member[v, image[inverse[z], u]], p5 → not[disjoint[u, v]],
            p6 → member[v, domain[z]], p7 → equal[u, v]}]]

Out[11]= or[equal[u, v], not[member[u, domain[z]]],
           not[member[v, image[inverse[z], u]]], not[subclass[z, inverse[E]]],
           not[subclass[cart[domain[z], domain[z]], union[DISJOINT, Id]]]] == True

In[12]:= (% /. {t → t_, u → u_, v → v_, z → z_}) /. Equal → SetDelayed
```

The variable **v** in the preceding statement can be eliminated as follows:

```
In[13]:= Map[equal[V, #] &, SubstTest[class, v, or[equal[u, v],
             not[member[u, domain[z]]], not[member[v, image[inverse[z], u]]],
             not[subclass[z, inverse[E]]], not[subclass[x, y]]],
            {x → P[domain[z]], y → cliques[union[DISJOINT, Id]]}]] // Reverse

Out[13]= or[not[member[u, domain[z]]], not[subclass[z, inverse[E]]],
           not[subclass[cart[domain[z], domain[z]], union[DISJOINT, Id]]],
           subclass[image[inverse[z], u], singleton[u]]] == True

In[14]:= (% /. {u → u_, z → z_}) /. Equal → SetDelayed
```

range of the choice function

The following lemma yields an inclusion in the opposite direction, needed to replace **subclass[image[inverse[z],u], singleton[u]]** with an equation.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[and[p1, p2], p3], implies[and[p1, p2, p3], p4],
           not[implies[p1, p4]],
           {p1 → and[FUNCTION[z], subclass[z, inverse[E]], member[u, domain[z]]],
            p2 → member[pair[u, APPLY[z, u]], z], p3 → member[APPLY[z, u], u],
            p4 → member[u, image[inverse[z], u]]}]]

Out[15]= or[member[u, image[inverse[z], u]], not[FUNCTION[z]],
           not[member[u, domain[z]]], not[subclass[z, inverse[E]]]] == True

In[16]:= (% /. {u → u_, z → z_}) /. Equal → SetDelayed
```

Theorem: When a choice function **z** is applied to a pairwise disjoint collection of sets, the range of **z** holds exactly one element from each member of the collection:

```
In[17]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[p1, p4], implies[and[p3, p4], p5], implies[and[p2, p5], p6],
           implies[and[p1, p6], p7], not[implies[p1, p7]],
           {p1 → and[FUNCTION[z], subclass[z, inverse[E]], subclass[
               cart[domain[z], domain[z]], union[DISJOINT, Id]], member[u, domain[z]]],
            p2 → equal[intersection[u, range[z]], image[z, image[inverse[z], u]]],
            p3 -> subclass[image[inverse[z], u], singleton[u]],
            p4 → member[u, image[inverse[z], u]],
            p5 → equal[image[inverse[z], u], singleton[u]],
            p6 -> equal[intersection[u, range[z]], image[z, singleton[u]]],
            p7 → member[intersection[u, range[z]], range[SINGLETON]]}]]

Out[17]= or[member[intersection[u, range[z]], range[SINGLETON]], not[FUNCTION[z]],
           not[member[u, domain[z]]], not[subclass[z, inverse[E]]],
           not[subclass[cart[domain[z], domain[z]], union[DISJOINT, Id]]]] == True

In[18]:= (% /. {u → u_, z → z_}) /. Equal → SetDelayed
```

removing the variables

The next step is to remove the variable **u**.

```
In[19]:= Map[equal[V, #] &, SubstTest[class, u,
           or[member[intersection[u, r], s], not[subclass[p, y]], not[member[u, t]],
            not[subclass[z, v]], not[subclass[w, x]]], {r → range[z],
            s → range[SINGLETON], t → domain[z], v → inverse[E], y → FUNS, p → P[z],
            w → P[domain[z]], x → cliques[union[DISJOINT, Id]]}]] // Reverse

Out[19]= or[not[FUNCTION[z]], not[subclass[z, inverse[E]]],
          not[subclass[cart[domain[z], domain[z]], union[DISJOINT, Id]]],
          subclass[image[IMAGE[id[range[z]]], domain[z]], range[SINGLETON]]] == True

In[20]:= or[not[FUNCTION[z_]], not[subclass[z_, inverse[E]]],
          not[subclass[cart[domain[z_], domain[z_]], union[DISJOINT, Id]]],
          subclass[image[IMAGE[id[range[z_]]], domain[z_]], range[SINGLETON]]] := True
```

If one specializes this to sets, one can restate what has been proved as follows: The range of a choice function for a pairwise-disjoint collection of non-empty sets is a selecting set for that collection.

```
In[21]:= implies[and[member[z, X[composite[inverse[E], id[domain[z]]]]],
           subclass[cart[domain[z], domain[z]], union[DISJOINT, Id]]],
          subclass[image[IMAGE[id[range[z]]], domain[z]], range[SINGLETON]]]

Out[21]= True
```

The variable **z** is removed via an application of **Normality**.

```
In[22]:= Map[equal[V, #] &, complement[dif[intersection[FUNS, P[inverse[E]],
             image[inverse[IMAGE[FIRST]], cliques[union[DISJOINT, Id]]]], image[
             inverse[DORA], UB[image[inverse[CAP], range[SINGLETON]]]]]] // Normality]

Out[22]= subclass[image[CAP, composite[image[DORA, intersection[FUNS,
              image[inverse[IMAGE[FIRST]], cliques[union[DISJOINT, Id]]],
              P[inverse[E]]]], E]], range[SINGLETON]] == True

In[23]:= % /. Equal → SetDelayed
```

---

eliminating the specifics of the construction

The value of the function **DORA** is the ordered pair of the domain and range of its argument. By eliminating this function, one obtains a more existential statement which hides the specifics of the construction. The following lemma is the key step:

```
In[24]:= Map[image[#, UB[x]] &, Assoc[FIRST, DORA, inverse[DORA]]] // Reverse

Out[24]= image[IMAGE[FIRST], image[inverse[DORA], UB[x]]] ==
          image[inverse[UB[x]], complement[singleton[0]]]

In[25]:= image[IMAGE[FIRST], image[inverse[DORA], UB[x_]]] :=
          image[inverse[UB[x]], complement[singleton[0]]]
```

Another lemma helps to clean up the result:

```
In[26]:= SubstTest[image, x, union[y, z],
           {x → inverse[UB[image[inverse[CAP], range[SINGLETON]]]],
            y → singleton[0], z → complement[singleton[0]]}] // Reverse

Out[26]= image[inverse[UB[image[inverse[CAP], range[SINGLETON]]]],
           complement[singleton[0]]] ==
          domain[UB[image[inverse[CAP], range[SINGLETON]]]]

In[27]:= image[inverse[UB[image[inverse[CAP], range[SINGLETON]]]],
           complement[singleton[0]]] :=
          domain[UB[image[inverse[CAP], range[SINGLETON]]]]
```

Eliminating **DORA** yields the following statement:

```
In[28]:= SubstTest[implies, subclass[x, z], subclass[image[u, x], image[u, z]],
           {u → IMAGE[FIRST], x → intersection[FUNS, P[inverse[E]]], z ->
             union[image[inverse[DORA], UB[image[inverse[CAP], range[SINGLETON]]]],
              image[inverse[IMAGE[FIRST]], t]]}] /.
          t → complement[cliques[union[DISJOINT, Id]]]

Out[28]= subclass[intersection[cliques[union[DISJOINT, Id]],
            image[IMAGE[FIRST], intersection[FUNS, P[inverse[E]]]]],
           domain[UB[image[inverse[CAP], range[SINGLETON]]]]] == True

In[29]:= subclass[intersection[cliques[union[DISJOINT, Id]],
            image[IMAGE[FIRST], intersection[FUNS, P[inverse[E]]]]],
           domain[UB[image[inverse[CAP], range[SINGLETON]]]]] := True
```

So far, all the results obtained are completely general, and do not depend on whether or not the axiom of choice is valid.

corollary: axch => a8

In[30]:=
```
SubstTest[implies, and[equal[u, v], subclass[intersection[w, v], x]],
 subclass[intersection[u, w], x],
 {u -> P[complement[singleton[0]]],
  v -> image[IMAGE[FIRST], intersection[FUNS, P[inverse[E]]]],
  w -> cliques[union[DISJOINT, Id]],
  x -> domain[UB[image[inverse[CAP], range[SINGLETON]]]]}]
```

Out[30]=
```
or[not[axch], subclass[
   intersection[cliques[union[DISJOINT, Id]], P[complement[singleton[0]]]],
   domain[UB[image[inverse[CAP], range[SINGLETON]]]]]] == True
```

In[31]:=
```
or[not[axch], subclass[
   intersection[cliques[union[DISJOINT, Id]], P[complement[singleton[0]]]],
   domain[UB[image[inverse[CAP], range[SINGLETON]]]]]] := True
```