simplifying axiom a8

Johan G. F. Belinfante
2004 October 29

```
In[1]:=  SetDirectory["i:"]; << goedel62.28b; << tools.m

        :Package Title: goedel62.28b        2004 October 28 at 11:50 p.m.

        It is now:  2004 Oct 29 at 22:4

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 October 28

        weightlimit = 40
```

summary

This notebook is concerned with obtaining a simple variable-free statement of a certain version of the axiom of choice attributed by Jean Rubin to Bertrand Russell.   Russell's formulation of the axiom of choice involves selecting one element from each of a collection of pairwise disjoint sets without introducing the concept of function. There are class forms and set forms of this axiom. Only the set form, called axiom **a8** in Rubin's book, will be stated in variable-free form.

```
In[2]:=  "Jean E. Rubin, Set Theory for the Mathematician,
           Holden-Day, San Francisco, 1967. (see page 80)"

Out[2]=  Jean E. Rubin, Set Theory for the
           Mathematician, Holden-Day, San Francisco, 1967. (see page 80)
```

To get a fairly simple result one needs to be quite clever about using **class** versus **assert**, and turning off the **cond** flag at an appropriate point. In the course of this work it was discovered that two simplifications can be made in the axiom **a8** as stated in Rubin's book. First, the class **x** is need not be required to be nonempty. Second, the requirement that the selecting class be contained in the union of the disjoint collection can be omitted. An oversight in the definition of pairwise disjoint in Rubin's book is also corrected. In this notebook, the focus is only on the variable-free statement of axiom **a8**, not on its consequences nor its relation to other forms of the axiom of

choice. Consequently, none of the rewrite rules derived here depend on whether the axiom **a8**, or any other form of the axiom of choice, is actually true or not.

pairwise disjoint collections

A collection of sets **x** is **pairwise disjoint** if each pair of elements in **x** are either equal or disjoint. On page 54, definition 3.2.7 (b), Rubin has accidentally omitted the possibility they might be equal. Correcting this oversight, the class of all collections of sets satisfying the pairwise disjointness condition is:

```
In[3]:= class[x, forall[u, v,
          implies[and[member[u, x], member[v, x]], or[equal[u, v], disjoint[u, v]]]]]

Out[3]= cliques[union[DISJOINT, Id]]
```

informal statement of axiom a8

The version **a8** of the axiom of choice says that for any nonempty pairwise disjoint set **x** of nonempty sets, there is a set **c** obtained by selecting one element from each member of **x**. The class **c** (which presumably stands for choice) in Rubin's book is here replaced by **z**. This change in notation is necessitated by the fact that the **TOOLS.M** file defines **c** to be a generic set rather than a pure variable.

removing the quantifiers in axiom a8

Rubin's formulation of axiom **a8** will now be studied. The innermost quantifier in Rubin's formulation of axiom **a8** is used to formulate the statement that **z** picks exactly one member from each member **u** of **x**. This quantifier can be eliminated as follows:

```
In[4]:= assert[exists[v, and[member[v, u], equal[intersection[z, u], singleton[v]]]]]

Out[4]= member[intersection[u, z], range[SINGLETON]]
```

A single application of **class** serves to eliminate the quantifiers on **z** and **u** at the same time:

```
In[5]:= class[x, exists[z, and[subclass[z, U[x]], forall[u,
            implies[member[u, x], member[intersection[u, z], range[SINGLETON]]]]]]]]

Out[5]= fix[composite[inverse[BIGCUP], S, UB[image[inverse[CAP], range[SINGLETON]]]]]
```

The idea behind the condition **subclass[z, U[x]]** is that the set **z** does not hold any thing else besides the selected members of the members of **x**. It seems intuitively clear that this condition is not really needed. If one simply omits this condition, one instead obtains a simpler result:

```
In[6]:= class[x, exists[z, forall[u,
           implies[member[u, x], member[intersection[u, z], range[SINGLETON]]]]]]

Out[6]= domain[UB[image[inverse[CAP], range[SINGLETON]]]]
```

It will be shown in the next section that these two expressions are in fact equal, so one is free to use the simpler of these to formulate the axiom of choice. The class of non-empty collections of sets that do not hold the empty set is:

```
In[7]:= class[x, and[not[equal[0, x]], not[member[0, x]]]]

Out[7]= intersection[complement[singleton[0]], P[complement[singleton[0]]]]
```

It follows that Rubin's axiom **a8** can be written as:

```
In[8]:= subclass[intersection[complement[singleton[0]],
          P[complement[singleton[0]]], cliques[union[DISJOINT, Id]]],
         domain[UB[image[inverse[CAP], range[SINGLETON]]]]]

Out[8]= subclass[
         intersection[cliques[union[DISJOINT, Id]], P[complement[singleton[0]]]],
         domain[UB[image[inverse[CAP], range[SINGLETON]]]]]
```

It should be noted that the condition that the pairwise disjoint collection **x** be non-empty has automatically dropped out of this, thereby further simplifying the statement of axiom **a8**. The reason for this is that the empty set belongs to the class **domain[-UB[image[inverse[CAP], range[SINGLETON]]]]**.

```
In[9]:= member[0, domain[UB[image[inverse[CAP], range[SINGLETON]]]]]

Out[9]= True
```

eliminating the cover condition subclass[z, U[x]]

It will be shown in this section that it is safe to omit the cover condition **subclass[z, U[x]]** in Rubin's statement of axiom **a8**. Justifying this technical simplification required a substantial amount of effort due to the presence in the **GOEDEL** program of various rewrite rules about complements that affect **fix**, inverse images of **CAP** and the upper bound constructor **UB**. The following is actually the key step, but it is not needed until the very end.

```
In[10]:= ImageComp[IMAGE[id[z]], IMAGE[id[U[x]]], x]

Out[10]= image[IMAGE[id[intersection[z, U[x]]]], x] == image[IMAGE[id[z]], x]

In[11]:= image[IMAGE[id[intersection[z_, U[x_]]]], x_] := image[IMAGE[id[z]], x]
```

The idea is that if **z** satisfies the condition about selecting a single element from each member of **x**, then **intersection[z, U[x]]** also satisfies this condition as well as the additional condition of being a subset of **U[x]**. A membership rule for the ternary relation **composite[CAP, cross[BIGCUP,Id]]** will be needed. Turning off the simplify flag reduces the execution time from about 30 seconds to 4 seconds. (Turning off the **cond** flag does not help, and in fact blocks this derivation.)

```
In[12]:= simplify = False;

In[13]:= member[pair[pair[x, z], w], composite[CAP, cross[BIGCUP, Id]]] // AssertTest

Out[13]= member[pair[pair[x, z], w], composite[CAP, cross[BIGCUP, Id]]] ==
          and[equal[w, intersection[z, U[x]]], member[x, V], member[z, V]]

In[14]:= member[pair[pair[x_, z_], w_], composite[CAP, cross[BIGCUP, Id]]] :=
          and[equal[w, intersection[z, U[x]]], member[x, V], member[z, V]]
```

The following normalization condition for this ternary relation is also needed:

```
In[15]:= composite[CAP, cross[BIGCUP, Id]] // VSTriNormality // Reverse

Out[15]= composite[intersection[complement[
              composite[SECOND, intersection[composite[inverse[FIRST], BIGCUP, FIRST],
                composite[inverse[DIF], E, inverse[E], SECOND]]]],
            composite[inverse[S], CAP, cross[BIGCUP, Id]]], id[cart[V, V]]] ==
          composite[CAP, cross[BIGCUP, Id]]

In[16]:= % /. Equal → SetDelayed
```

The following two technical lemmas deal with rules about complements.

```
In[17]:= (composite[image[inverse[CAP], complement[y]], inverse[E]] // RelnNormality //
            Reverse) /. y → complement[x]

Out[17]= fix[composite[inverse[SECOND],
            S, id[x], fix[composite[complement[inverse[rotate[
                 composite[inverse[SECOND], E, inverse[E], DIF, SWAP]]]], intersection[
               composite[S, SECOND], composite[inverse[E], FIRST, FIRST]]]]]] ==
          composite[image[inverse[CAP], x], inverse[E]]

In[18]:= fix[composite[inverse[SECOND], S,
            id[x_], fix[composite[complement[inverse[rotate[
                 composite[inverse[SECOND], E, inverse[E], DIF, SWAP]]]], intersection[
               composite[S, SECOND], composite[inverse[E], FIRST, FIRST]]]]]] :=
          composite[image[inverse[CAP], x], inverse[E]]

In[19]:= complement[lb[composite[SECOND,
             intersection[composite[inverse[FIRST], BIGCUP], composite[inverse[CAP],
               complement[image[inverse[CAP], y]], inverse[E]]]], V]] // Normality

Out[19]= complement[lb[composite[SECOND,
             intersection[composite[inverse[FIRST], BIGCUP], composite[inverse[CAP],
               complement[image[inverse[CAP], y]], inverse[E]]]], V]] ==
          fix[composite[inverse[BIGCUP], S, UB[image[inverse[CAP], y]]]]

In[20]:= complement[lb[composite[SECOND,
             intersection[composite[inverse[FIRST], BIGCUP], composite[inverse[CAP],
               complement[image[inverse[CAP], y_]], inverse[E]]]], V]] :=
          fix[composite[inverse[BIGCUP], S, UB[image[inverse[CAP], y]]]]
```

The main result now follows:

```
In[21]:= Map[domain[complement[#]] &, composite[SECOND,
            intersection[composite[inverse[FIRST], BIGCUP], composite[inverse[CAP],
              complement[image[inverse[CAP], y]], inverse[E]]]] // VSNormality]

Out[21]= fix[composite[inverse[BIGCUP], S, UB[image[inverse[CAP], y]]]] ==
          domain[UB[image[inverse[CAP], y]]]

In[22]:= fix[composite[inverse[BIGCUP], S, UB[image[inverse[CAP], y_]]]] :=
          domain[UB[image[inverse[CAP], y]]]
```