Axiom Group A in Gödel’s monograph

Johan G. F. Belinfante
2008 November 12

```
In[1]:= SetDirectory["l:"]; << goedel.08nov11a;<< tools.m

        :Package Title: goedel.08nov11a          2008 November 11 at 11:55 a.m.

        It is now:  2008 Nov 12 at 6:58

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 October 21

        weightlimit = 40
```

summary

Axiom Group A in Kurt Gödel’s monograph is reviewed here to clarify how his notations and axioms relate to those used in the **GOEDEL** program.

```
In[2]:=  "Kurt Gödel, The Consistency of the Axiom of Choice
          and of the Generalized Continuum Hypothesis with the Axioms
          of Set Theory, Princeton University Press, 1940.     QA9 .G54";
```

connectives and quantifiers

The following notations are explained in the third paragraph on page 2 of Gödel’s monograph: **(X) p** means that the proposition **p** is true for every class **X**, Gödel writes **(∃X) p** for the statement that there exists a class **X** such that **p** is true. The notation **∼ p** means **not[p]**. Gödel writes **p.q** for **and[p, q]**. The notation **p ∨ q** means **or[p, q]**. The notation **p ⊃ q** means **implies[p, q]**. The notation **p ≡ q** means **equiv[p, q]**. The notation **X = Y** means **equal[X, Y]**. Gödel’s notation **(E! X) p** means that there is a unique class **X** such that **p** is true.

The notation **x ϵ y** means **member[x, y]**. (This notation is apparently due to Peano. Epsilon is the first letter of the Greek word for "is".) Membership is a primitive (undefined) concept.

quantifiers and models

In the **GOEDEL** program quantifiers are restricted to sets. The notation **(x) p** translates to

```
In[5]:= assert[forall[x, p[x]]]

Out[5]= equal[V, class[x, p[x]]]
```

The notation **(∃ x) p** translates to

```
In[6]:= assert[exists[x, p[x]]]

Out[6]= not[equal[0, class[x, p[x]]]]
```

The **GOEDEL** program also has a (seldom-used) quantifier for unique existence:

```
In[7]:= ?? existsunique

        existsunique[x,p] is the statement that there exists a unique x such that p
```

Thus the notion **(E! x) p** can be translated as follows:

```
In[8]:= assert[existsunique[x, p[x]]]

Out[8]= member[class[x, p[x]], range[SINGLETON]]
```

The meaning of the quantifier **existsunique** can be elucidated if one introduces the following temporary abbreviation:

```
In[9]:= class[x_, p[x_]] := MODELS[p]

In[10]:= assert[and[exists[x, p[x]], forall[x, y, implies[and[p[x], p[y]], equal[x, y]]]]] //
           not // not

Out[10]= member[MODELS[p], range[SINGLETON]]
```

That is, there is only one model for the proposition **p[x]**. Similarly, the existential and universal quantifiers correspond to statements that the class of models is not empty, and that the class of models is the class of all sets, respectively:

```
In[11]:= assert[exists[x, p[x]]]

Out[11]= not[equal[0, MODELS[p]]]

In[12]:= assert[forall[x, p[x]]]

Out[12]= equal[V, MODELS[p]]
```

page 3

The concept of **class** is primitive (undefined). The statement that **X** is a class is written **𝔆𝔩𝔰(X)**. In the **GOEDEL** program, it is assumed, following Art Quaife, that everything is a class, and so no notation is introduced for this predicate. The notation **𝔐(x)** means that **x** is a set (German: Menge). Again, following Quaife, this is rendered as **member[x, V]**. Gödel uses capital letters **X, Y, ...** for classes and lower case letters **x, y, ...** for sets. This convention is abandoned in the **GOEDEL** program. If something is a set one usually needs to state this explicitly. An exception is that in the notation **class[x, ...]** or **class[pair[x, y], ...]**, it is tacitly assumed that **x, y, ...** are sets. Also in the **GOEDEL** program , quantifiers are tacitly assumed to refer to set variables.

axioms of group A

In the statement of the axioms all free variables are assumed to be universally quantified. This convention is followed in the **GOEDEL** program, and if effect provides a universal quantifier for class variables. The only way to express existence for proper classes is by Skolemization: one needs to introduce a name for the class. (To assert the existence of any proper class for which there is no specific construction, one must introduce a new name for it, thereby extending the language of set theory. Of course, any such extension raises the issue of consistency.)

Axiom 1. **𝔆𝔩𝔰(x)**. That is, every set is a class. This axiom is tacitly assumed in the **GOEDEL** program, but can not be expressed explicitly.

Axiom 2. **X ϵ Y .⊃. 𝔐(x)**. Every member of a class is a set.

```
In[13]:= implies[member[x, y], member[x, V]]

Out[13]= True
```

As noted by Quaife, the definition of **subclass** can be used to eliminate the variable **x**, and this axiom can be restated more simply as:

```
In[19]:= subclass[y, V]

Out[19]= True
```

Axiom 3. **(u) (u ϵ X. ≡ . u ϵ Y) .⊃. X = Y**. Principle of extensionality. Two classes are the same if their elements are the same.

```
In[15]:= assert[implies[forall[u, equiv[member[u, x], member[u, y]]], equal[x, y]]]

Out[15]= True
```

Again, one could eliminate the set-variable **u** and rewrite this as follows:

```
In[16]:= implies[and[subclass[x, y], subclass[y, x]], equal[x, y]]

Out[16]= True
```

Axiom 4. Axiom of Non-ordered Pairsets. **(x) (y) (∃z) [u ϵ z . ≡ : u = x .∨. u=y]**. There is a set whose only members are the sets **x** and **y**.

```
In[17]:= assert[forall[x, y, exists[z, equiv[member[u, z], or[equal[u, x], equal[u, y]]]]]]

Out[17]= True
```

A class which is not a set is called a **proper class**. Gödel defines the predicate **𝔓𝔯(X) ≡ ~ 𝔐(X)**. In the **GOEDEL** program this translates to

```
In[20]:= not[member[x, V]];
```

It follows from the principle of extensionality that there is only set whose only elements are **x** and **y**, an this set is defined to be **{x y}**. When **x = y**, this is further abbreviated to **{x}**. Thus Gödel makes these definitions:

u ϵ {x y} ≡ (u=x ∨ u=y).
{x} = {x, x}.

The singleton **{x}** is the set whose sole member is the set **x**. Note that Gödel has here made various sethood hypotheses implicit by his convention that lower case letters refer to sets.

In the **GOEDEL** program one must be more explicit about the sethood hypotheses. The following statement is a fairly literal translation of Gödel's definition of the non-ordered pairset:

```
In[29]:= implies[and[member[x, V], member[y, V]],
           equiv[member[u, set[x, y]], or[equal[u, x], equal[u, y]]]] // not // not

Out[29]= True
```

In the **GOEDEL** program, the sethood hypotheses on the variables **x** and **y** can be made implicit by introducing quantifiers:

```
In[35]:= assert[forall[x, y, equiv[member[u, set[x, y]], or[equal[u, x], equal[u, y]]]]]

Out[35]= True
```

Comment. One cannot here simply omit the hypotheses that **x** and **y** are sets. However, one could alter the statement as follows, transferring the sethood hypothesis from **x** and **y** to **u**.

```
In[34]:= equiv[member[u, set[x, y]], and[member[u, V], or[equal[u, x], equal[u, y]]]]

Out[34]= True
```

Again, one could make the sethood hypothesis on **u** implicit by introducing a quantifier:

```
In[33]:= assert[forall[u, equiv[member[u, set[x, y]], or[equal[u, x], equal[u, y]]]]]

Out[33]= True
```

The definition of singleton does not require any sethood hypothesis.

```
In[30]:= equal[set[x], set[x, x]]

Out[30]= True
```

The notation set in the **GOEDEL** program extends to any finite number of arguments: **set[x, y, ...]** is the set whose only members are **x, y, ...** . For compatibility with Quaife's (and the author's) work using McCune's automated reasoning program **Otter**, the following synonyms are also permitted:

```
In[36]:= singleton[x]

Out[36]= set[x]

In[37]:= pairset[x, y]

Out[37]= set[x, y]
```