Hilbert's paradox

Johan G. F. Belinfante and Benjamin Lamothe
2005 January 15

```
In[1]:= SetDirectory["i:"]; << goedel65.13a; << tools.m

        :Package Title: goedel65.13a           2005 January 13 at 3:20 p.m.

        It is now:  2005 Jan 15 at 3:46

        Loading Simplification Rules

        TOOLS.M                      Revised 2005 January 7

        weightlimit = 40
```

Hilbert's paradox

There is no set that is invariant under **POWER** and closed under arbitrary unions. This paradox was stated by Hilbert in 1905 course notes at Göttingen.

"Volker Peckhaus and Reinhard Kahle, *Hilbert's Paradox*, ***Historia Mathematica*, vol. 29 (2002), pp. 157–175";**

---

removing quantifiers

A class **x** is invariant under **POWER** if the power set of any member is a member.

```
In[2]:= assert[forall[y, implies[member[y, x], member[P[y], x]]]]

Out[2]= subclass[image[POWER, x], x]
```

A class **x** is invariant under arbitrary unions if the sum class of any subset is a member. Such classes are called **Uclosed**.

```
In[3]:= assert[forall[y, implies[subclass[y, x], member[U[y], x]]]]

Out[3]= subclass[Uclosure[x], x]
```

a comment

The Hilbert paradox is that there is no set satisfying both of these conditions. The empty set, for example, is invariant under **POWER**, but it is not Uclosed. Any Uclosed class must hold the sum class of the empty subset.

```
In[5]:= implies[subclass[Uclosure[x], x], member[0, x]]

Out[5]= True
```

a lemma related to Russell's paradox

There is no set **x** whose power set belongs to itself. The **GOEDEL** program already knows some version of this:

```
In[18]:= class[x, member[P[x], P[x]]]

Out[18]= 0
```

The class in question is the fixed point set of the following relation.

```
In[19]:= class[pair[x, y], member[P[x], P[y]]]

Out[19]= composite[S, POWER]
```

The available rewrite rule is this variable-free statement:

```
In[27]:= fix[composite[S, POWER]]

Out[27]= 0
```

One can introduce variables into this statement:

```
In[23]:= Map[and[member[x, y], #] &,
          (member[x, fix[y]] // AssertTest // Reverse) /. y -> composite[S, POWER]]

Out[23]= and[member[x, y], subclass[P[x], x]] == False

In[24]:= and[member[x_, y_], subclass[P[x_], x_]] := False
```

Corollary.

```
In[25]:= member[P[x], RUSSELL] // AssertTest

Out[25]= member[P[x], RUSSELL] == member[x, V]

In[26]:= member[P[x_], RUSSELL] := member[x, V]
```

Contrapositive form:

```
In[31]:= implies[subclass[P[x], x], not[member[x, y]]] // NotNotTest

Out[31]= or[not[member[x, y]], not[subclass[P[x], x]]] == True

In[32]:= or[not[member[x_, y_]], not[subclass[P[x_], x_]]] := True
```

Corollary.

```
In[44]:= Map[implies[member[x, y], #] &,
          SubstTest[implies, subclass[P[z], z], not[member[z, V]], z → P[U[x]]]]

Out[44]= or[not[member[x, y]], not[subclass[P[U[x]], U[x]]]] == True

In[45]:= or[not[member[x_, y_]], not[subclass[P[U[x_]], U[x_]]]] := True
```

---

derivation of the paradox

Lemma.

```
In[13]:= SubstTest[implies, and[member[x, z], subclass[z, w]],
          member[x, w], w → image[inverse[POWER], y]]

Out[13]= or[member[P[x], y], not[member[x, z]],
           not[subclass[image[POWER, z], y]]] == True

In[14]:= or[member[P[x_], y_], not[member[x_, z_]],
           not[subclass[image[POWER, z_], y_]]] := True
```

Hilbert's paradox:

```
In[51]:= Map[not, SubstTest[and, implies[and[p1, p2], p4], implies[and[p3, p4], p5],
           implies[p5, p6], implies[p6, not[p1]], not[implies[and[p2, p3], not[p1]]],
           {p1 → member[x, y], p2 → subclass[Uclosure[x], x],
            p3 → invariant[POWER, x], p4 → member[U[x], x],
            p5 → member[P[U[x]], x], p6 → subclass[P[U[x]], U[x]]}]]

Out[51]= or[not[member[x, y]], not[subclass[image[POWER, x], x]],
           not[subclass[Uclosure[x], x]]] == True
```

```
In[52]:= or[not[member[x_, y_]], not[subclass[image[POWER, x_], x_]],
           not[subclass[Uclosure[x_], x_]]] := True
```

Variable-free formulation:

```
In[61]:= Map[complement,
           SubstTest[class, x, or[not[member[x, V]], not[subclass[image[y, x], x]],
             not[subclass[Uclosure[x], x]]], y → POWER]] // Reverse
```

```
Out[61]= intersection[fix[UCLOSURE], invar[POWER]] == 0
```

```
In[62]:= intersection[fix[UCLOSURE], invar[POWER]] := 0
```