all games

Johan G. F. Belinfante
2012 April 24

```
In[1]:= SetDirectory["l:"]; << goedel.12apr23a
```

```
:Package Title: goedel.12apr23a                    2012 April 23 at 1:15 p.m.

Loading takes about seventeen minutes, half that time due to builtin pauses.

It is now:  2012 Apr 24 at 17:3

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2012 Apr 24 at 17:20
```

summary

John H. Conway in 1976 published a book ***On numbers and games*** in which it was shown that the class of ordinals can be embedded in a field. The present notebook was inspired by the following paper in which a simplified account of Conway's constructions is given.

```
In[2]:= "Philip Ehrlich, The Absolute Arithmetic
         Continuum and the Unification of all Numbers Great and Small
         The Bulletin of Symbolic Logic, volume 18, pages 1-45 (2012)";
```

The class of all games can be characterized as the smallest class **GAMES** with the property that for any subsets **x** and **y** of **GAMES**, the ordered pair **(x, y)** is a again member of **GAMES**. The existence of the class **GAMES** is proved by a transfinite recursion to be presented in a series of notebooks.

```
In[3]:= assert[forall[x, y,
          implies[and[subclass[x, GAMES], subclass[y, GAMES]], member[pair[x, y], GAMES]]]]
```

```
Out[3]= subclass[cart[P[GAMES], P[GAMES]], GAMES]
```

All that will be shown below is that any class **x** satisfying the inclusion **P[x] × P[x] ⊂ x** must be a proper class.

observation

The inclusion **P[x] × P[x] ⊂ x** is satisfied by the universal class **V** of all sets, as well as by the class **V × V**. If Kuratowski's construction of ordered pairs is assumed, then the class **REGULAR** and its cartesian square also satisfy this inclusion.

```
In[5]:= Map[subclass[cartsq[P[#]], #] &, {V, cart[V, V], REGULAR, cart[REGULAR, REGULAR]}]
Out[5]= {True, True, True, True}
```

What is not so obvious, and will later need to be established by transfinite recursion, is that there is a smallest such class.

the simplest game

Lemma. If **P[x] × P[x] ⊂ x**, then **x** is not empty.

```
In[6]:= SubstTest[implies, and[member[u, v], subclass[v, w]],
          member[u, w], {u → pair[0, 0], v -> cart[P[x], P[x]], w → x}] // Reverse
Out[6]= or[member[pair[0, 0], x], not[subclass[cart[P[x], P[x]], x]]] == True

In[7]:= or[member[pair[0, 0], x_], not[subclass[cart[P[x_], P[x_]], x_]]] := True
```

It follows that if there is a class of all games, then it holds the simplest game, **pair[0, 0]**.

Lemma.

```
In[8]:= or[member[0, fix[x]], not[member[pair[0, 0], x]]] // AssertTest
Out[8]= or[member[0, fix[x]], not[member[pair[0, 0], x]]] == True

In[9]:= or[member[0, fix[x_]], not[member[pair[0, 0], x_]]] := True
```

Theorem.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 -> subclass[cart[P[x], P[x]], x],
            p2 -> member[pair[0, 0], x], p3 → member[0, fix[x]]}]] // Reverse
Out[10]= or[member[0, fix[x]], not[subclass[cart[P[x], P[x]], x]]] == True

In[11]:= or[member[0, fix[x_]], not[subclass[cart[P[x_], P[x_]], x_]]] := True
```

some observations

Observation. If **P[x] × P[x] ⊂ x**, then the same holds for **Id ∘ x**.

```
In[13]:= implies[subclass[cart[P[x], P[x]], x], subclass[cart[P[y], P[y]], y]] /.
          y → composite[Id, x]

Out[13]= True
```

Observation. If **P[x] × P[x] ⊂ x**, then the same holds for **P[x] × P[x].**

```
In[14]:= implies[subclass[cart[P[x], P[x]], x], subclass[cart[P[y], P[y]], y]] /.
          y → cart[P[x], P[x]]

Out[14]= True
```

Observation. If **P[x] × P[x] ⊂ x**, then **P[x] ⊂ fix[x]**.

```
In[12]:= implies[subclass[cart[P[x], P[x]], x], subclass[P[x], fix[x]]]

Out[12]= True
```

the main theorem

Theorem. If **P[x] × P[x] ⊂ x**, then **P[id[fix[x]]] ⊂ fix[x]**.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
            not[implies[p1, p3]], {p1 -> subclass[cart[P[x], P[x]], x],
             p2 → subclass[P[x], fix[x]], p3 -> subclass[P[id[fix[x]]], fix[x]]}]] // Reverse

Out[15]= or[not[subclass[cart[P[x], P[x]], x]], subclass[P[id[fix[x]]], fix[x]]] == True

In[16]:= or[not[subclass[cart[P[x_], P[x_]], x_]], subclass[P[id[fix[x_]]], fix[x_]]] := True
```

Lemma. If **x** is a set, then **x** is equipollent to **id[x]**.

```
In[17]:= Map[implies[member[x, y], #] &, SubstTest[implies, member[t, BIJ],
            member[pair[domain[t], range[t]], Q], t → composite[DUP, id[x]]]] // Reverse

Out[17]= or[member[pair[x, id[x]], Q], not[member[x, y]]] == True

In[18]:= or[member[pair[x_, id[x_]], Q], not[member[x_, y_]]] := True
```

Theorem. If **x** is a subset of **y**, then **x** is equipollent to a subset of **y**.

```
In[19]:= Map[implies[member[x, z], #] &, SubstTest[implies, and[member[x, u], subclass[u, v]],
            member[x, v], {u → P[y], v → image[Q, P[y]]}]] // Reverse

Out[19]= or[member[x, image[Q, P[y]]], not[member[x, z]], not[subclass[x, y]]] == True
```

```
In[20]:= or[member[x_, image[Q, P[y_]]], not[member[x_, z_]], not[subclass[x_, y_]]] := True
```

Lemma.

```
In[21]:= SubstTest[implies, and[member[pair[v, u], Q], member[u, image[Q, P[w]]]],
           member[v, image[Q, P[w]]], {u → P[id[x]], v → P[x], w → x}] // Reverse
```

```
Out[21]= or[not[member[P[id[x]], image[Q, P[x]]]], not[member[pair[P[x], P[id[x]]], Q]]] == True
```

```
In[22]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. If $\mathbf{P[x] \times P[x] \subset x}$, then $\mathbf{x}$ is not a set.

```
In[23]:= Map[not, SubstTest[and, p1, p2,
            (*implies[p1,p4],implies[p2,p3],*) implies[p4, p5], implies[and[p3, p4], p6],
            (*implies[p5,p7],*) implies[p7, not[p6]], {p1 → member[x, y],
             p2 → subclass[cart[P[x], P[x]], x], p3 -> subclass[P[id[fix[x]]], fix[x]],
             p4 → member[fix[x], V], p5 -> member[pair[fix[x], id[fix[x]]], Q],
             p6 -> member[P[id[fix[x]]], image[Q, P[fix[x]]]],
             p7 -> member[pair[P[fix[x]], P[id[fix[x]]]], Q]}]] // Reverse
```

```
Out[23]= or[not[member[x, y]], not[subclass[cart[P[x], P[x]], x]]] == True
```

```
In[24]:= or[not[member[x_, y_]], not[subclass[cart[P[x_], P[x_]], x_]]] := True
```