recursive game construction, part 1.

Johan G. F. Belinfante
2012 April 25

```
In[1]:= SetDirectory["l:"]; << goedel.12apr24a

        :Package Title: goedel.12apr24a                  2012 April 24 at 5:35 p.m.

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

        It is now:  2012 Apr 25 at 12:22

        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 25 at 12:39
```

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 following paper gives a simplified account of Conway's constructions.

```
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)";
```

Conway's 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**. It was recently shown in the notebook **allgames.nb** that such a class satisfies the inclusion **P[x] × P[x] ⊂ x**, and must therefore be a proper class.

This is the first of a series of notebooks in which the theory of well-founded recursion due to A. Tarski and R. Montague will be used to construct the least class **GAMES** satisfying the equation **GAMES = P[GAMES] × P[GAMES]**. The theory of well-founded recursion was incorporated into the **GOEDEL** program in a series of notebooks in 2004, based on an exposition on pages 65-66 of the following book.

```
In[3]:= "Azriel Levy, Basic Set Theory, reprinted by Dover Publications, 2002.";
```

In the **GOEDEL** program, well founded recursion is available in the the form of a collection of rewrite rules for a constructor **rec[x, y]**. In this notebook a function **gamerec** is introduced, and used to define the class **GAMES**.

basic definitions

The following abbreviation will be used for the restriction to the class of ordinals of the total solution of a certain recursion equation.

```
In[4]:= composite[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
          composite[inverse[E], id[OMEGA]]], id[OMEGA]] := gamerec
```

Basic properties of **gamerec** will now be derived, using the general theory of well founded recursion, with the following standard replacement.

```
In[5]:= standard := {x → composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
          y → composite[inverse[E], id[OMEGA]]}
```

Lemma. The total solution is a function.

```
In[6]:= SubstTest[or, FUNCTION[rec[funpart[x], thinpart[y]]],
          not[WELLFOUNDED[inverse[y]]], standard] // Reverse
Out[6]= FUNCTION[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
          composite[inverse[E], id[OMEGA]]]] == True

In[7]:= FUNCTION[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
          composite[inverse[E], id[OMEGA]]]] := True
```

Theorem. The class **gamerec** is a function.

```
In[8]:= SubstTest[FUNCTION, composite[funpart[t], id[OMEGA]],
          t -> rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
            composite[inverse[E], id[OMEGA]]]] // Reverse
Out[8]= FUNCTION[gamerec] == True

In[9]:= FUNCTION[gamerec] := True
```

Corollary. Rewrite rule for vertical sections of **gamerec**.

```
In[10]:= SubstTest[image, funpart[t], set[z], t → gamerec] // Reverse
Out[10]= image[gamerec, set[z]] == set[APPLY[gamerec, z]]

In[11]:= image[gamerec, set[z_]] := set[APPLY[gamerec, z]]
```

Lemma. The domain of the total solution of the recursion is the class of all sets.

```
In[12]:= SubstTest[implies, and[FUNCTION[x], equal[domain[x], cart[V, V]], thin[y],
           WELLFOUNDED[inverse[y]]], equal[V, domain[rec[x, y]]], standard] // Reverse
Out[12]= equal[V, domain[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]]]] == True
```

```
In[13]:= domain[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]]] := V
```

Theorem. The domain of **gamerec** is the class Ω of all ordinals.

```
In[14]:= IminComp[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]], id[OMEGA], V]

Out[14]= domain[gamerec] == OMEGA

In[15]:= domain[gamerec] := OMEGA
```

an upper bound for range[gamerec]

Lemma. A simplification rule.

```
In[19]:= ImageComp[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]], id[OMEGA], V] // Reverse

Out[19]= image[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]], OMEGA] == range[gamerec]

In[20]:= image[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]], OMEGA] := range[gamerec]
```

Theorem. An upper bound for **range[gamerec]**.

```
In[21]:= SubstTest[subclass, image[rec[x, y], OMEGA], range[x], standard] // Reverse

Out[21]= subclass[range[gamerec], image[CART, id[range[POWER]]]] == True

In[22]:= subclass[range[gamerec], image[CART, id[range[POWER]]]] := True
```

Corollary. The class **range[gamerec]** is a class of cartesian squares.

```
In[23]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → range[gamerec], v → image[CART, id[range[POWER]]], w → image[CART, Id]}] // Reverse

Out[23]= subclass[range[gamerec], image[CART, Id]] == True

In[24]:= subclass[range[gamerec], image[CART, Id]] := True
```

the class of games

The class of all games is here defined by the following equation.

```
In[25]:= U[range[gamerec]] := GAMES
```

Theorem. The class of games is a relation.

```
In[27]:= SubstTest[implies, subclass[u, v], subclass[U[u], U[v]],
           {u -> range[gamerec], v -> image[CART, Id]}] // Reverse

Out[27]= subclass[GAMES, cart[V, V]] == True

In[28]:= subclass[GAMES, cart[V, V]] := True
```

the recursion equation

Lemma. The recursion equation.

```
In[29]:= SubstTest[or, equal[composite[x,
             id[composite[IMAGE[composite[id[rec[x, y]], inverse[FIRST]]], VERTSECT[y]]],
             inverse[FIRST]], rec[x, y]], not[equal[V, domain[VERTSECT[y]]]],
           not[equal[cart[V, V], domain[x]]], not[FUNCTION[x]],
           not[WELLFOUNDED[inverse[y]]], standard] // Reverse

Out[29]= equal[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
            composite[inverse[E], id[OMEGA]]],
           union[cart[complement[OMEGA], set[cart[set[0], set[0]]]], composite[CART, DUP,
             POWER, BIGCUP, IMAGE[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
               composite[inverse[E], id[OMEGA]]]], id[OMEGA]]]] == True

In[30]:= union[cart[complement[OMEGA], set[cart[set[0], set[0]]]], composite[CART, DUP, POWER,
            BIGCUP, IMAGE[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
              composite[inverse[E], id[OMEGA]]]], id[OMEGA]]] :=
          rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]]
```

Lemma.

```
In[31]:= SubstTest[composite, union[u, v], id[OMEGA],
           {u -> cart[complement[OMEGA], set[cart[set[0], set[0]]]],
            v -> composite[CART, DUP, POWER, BIGCUP, IMAGE[rec[composite[CART, DUP, POWER, BIGCUP,
                 IMAGE[SECOND], SECOND], composite[inverse[E], id[OMEGA]]]], id[OMEGA]]}]

Out[31]= composite[CART, DUP, POWER, BIGCUP,
           IMAGE[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
             composite[inverse[E], id[OMEGA]]]], id[OMEGA]] == gamerec

In[32]:= % /. Equal -> SetDelayed
```

Theorem. A simplification rule.

```
In[33]:= Map[U[fix[U[#]]] &, ImageComp[composite[CART, DUP, POWER, BIGCUP,
            IMAGE[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
              composite[inverse[E], id[OMEGA]]]]], id[OMEGA], V]]

Out[33]= U[fix[GAMES]] == GAMES

In[34]:= U[fix[GAMES]] := GAMES
```

Lemma. The total recursion solution is determined by its restriction to the class of ordinals.

```
In[35]:= SubstTest[implies,
           and[FUNCTION[x], equal[domain[x], cart[V, V]], thin[y], WELLFOUNDED[inverse[y]]],
           equal[rec[x, y], composite[x, HISTORY[rec[x, y], y]]], standard] // Reverse

Out[35]= equal[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
            composite[inverse[E], id[OMEGA]]],
           union[gamerec, cart[complement[OMEGA], set[cart[set[0], set[0]]]]]] == True

In[36]:= union[gamerec, cart[complement[OMEGA], set[cart[set[0], set[0]]]]] :=
          rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]]
```

Theorem. Recursion equation for **gamerec**.

```
In[37]:= Map[composite[CART, DUP, POWER, BIGCUP, #, id[OMEGA]] &, SubstTest[IMAGE,
           union[t, cart[complement[OMEGA], set[cart[set[0], set[0]]]]], t → gamerec]]

Out[37]= composite[CART, DUP, POWER, BIGCUP, IMAGE[gamerec], id[OMEGA]] == gamerec

In[38]:= composite[CART, DUP, POWER, BIGCUP, IMAGE[gamerec], id[OMEGA]] := gamerec
```

an APPLY rule for gamerec

Lemma. Simplification rule.

```
In[39]:= ImageComp[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]], id[OMEGA], ord[t]] // Reverse

Out[39]= image[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]], ord[t]] == image[gamerec, ord[t]]

In[40]:= image[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]], ord[t_]] := image[gamerec, ord[t]]
```

Lemma. Simplification rule.

```
In[41]:= ApComp[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]], id[OMEGA], ord[t]]

Out[41]= APPLY[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]], ord[t]] == APPLY[gamerec, ord[t]]

In[42]:= APPLY[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
           composite[inverse[E], id[OMEGA]]], ord[t_]] := APPLY[gamerec, ord[t]]
```

Theorem. An **APPLY** rule for **gamerec**.

```
In[43]:= (SubstTest[or, equal[APPLY[funpart[x], PAIR[z, composite[rec[funpart[x], thinpart[y]],
                id[image[thinpart[y], singleton[z]]]]], APPLY[rec[funpart[x], thinpart[y]],
              z]], not[member[z, domain[rec[funpart[x], thinpart[y]]]]],
            not[WELLFOUNDED[inverse[y]]], standard] // Reverse) /. z → ord[t]

Out[43]= equal[APPLY[gamerec, ord[t]],
           cart[P[U[image[gamerec, ord[t]]]], P[U[image[gamerec, ord[t]]]]] == True

In[44]:= cart[P[U[image[gamerec, ord[t_]]]], P[U[image[gamerec, ord[t_]]]] :=
          APPLY[gamerec, ord[t]]
```

Theorem.

```
In[46]:= SubstTest[member, APPLY[funpart[t], w], range[funpart[t]], t → gamerec] // Reverse

Out[46]= member[APPLY[gamerec, w], range[gamerec]] == member[w, OMEGA]

In[47]:= member[APPLY[gamerec, w_], range[gamerec]] := member[w, OMEGA]
```

Corollary. A lower bound for the class of all games.

```
In[48]:= SubstTest[implies, member[u, v], subclass[u, U[v]],
           {u → APPLY[gamerec, ord[z]], v -> range[gamerec]}] // Reverse

Out[48]= subclass[APPLY[gamerec, ord[z]], GAMES] == True

In[49]:= subclass[APPLY[gamerec, ord[z_]], GAMES] := True
```