PRELUDE

Johan G. F. Belinfante
2012 May 16

```
In[1]:= SetDirectory["l:"]; << goedel.12may15a

        :Package Title: goedel.12may15a                    2012 May 15 at 1:05 p.m.

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

        It is now:  2012 May 16 at 9:11

        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 May 16 at 9:28
```

summary

In the monograph ***On numbers and games*** John H. Conway in 1976 showed that the class of ordinals can be embedded in a (proper class) field. Philip Ehrlich simplified Conway's constructions by introducing a relation here called **PRELUDE**.

```
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 intention is to use this relation to apply the general theory of well-founded recursion to define important functions in the theory of Conway games. In this notebook this relation is formally defined and it is shown that it is well-founded and that its inverse is thin.

review

A brief review of the development of the theory of Conway games within the **GOEDEL** program will suffice. The class **GAMES** of Conway games can be characterized as the smallest class satisfying $\mathbf{P[x] \times P[x] \subset x}$. The existence of this class was established in a series of three notebooks using a transfinite well-founded recursive construction involving the function **gamerec**, defined as follows.

```
In[3]:= composite[rec[composite[CART, DUP, POWER, BIGCUP, IMAGE[SECOND], SECOND],
          composite[inverse[E], id[OMEGA]]], id[OMEGA]]

Out[3]= gamerec
```

The class **GAMES** is the sum class of the range of the function **gamerec**.

```
In[4]:= U[range[gamerec]]

Out[4]= GAMES
```

The class **GAMES** was shown to satisfy the following three rewrite rules:

```
In[5]:= fix[GAMES]

Out[5]= P[GAMES]

In[6]:= cart[P[GAMES], P[GAMES]]

Out[6]= GAMES

In[7]:= implies[subclass[cart[P[x], P[x]], x], subclass[GAMES, x]]

Out[7]= True
```

the birthday function

A game **x** is said to be **alive** at time **t** if **t** is an ordinal, and **x ∈ APPLY[gamerec, t]**. The **birthday** of a game **x** is the least ordinal **t** for which **x** is alive at time **t**. The birthday function **BDAY** is defined by the rewrite rule:

```
In[8]:= VERTSECT[complement[composite[complement[inverse[E]], inverse[gamerec], E]]]

Out[8]= BDAY
```

A game **x** is alive at any (ordinal) time **ord[t]** except before its birth.

```
In[9]:= member[ord[t], APPLY[BDAY, x]]

Out[9]= not[member[x, APPLY[gamerec, ord[t]]]]
```

Theorem. The relation **inverse[BDAY] ∘ E ∘ BDAY** is well-founded.

```
In[10]:= SubstTest[WELLFOUNDED, composite[inverse[funpart[u]], wf[v], funpart[u]],
           {u → BDAY, v → composite[id[OMEGA], E]}] // Reverse

Out[10]= WELLFOUNDED[composite[inverse[BDAY], E, BDAY]] == True

In[11]:= WELLFOUNDED[composite[inverse[BDAY], E, BDAY]] := True
```

definition of PRELUDE

Any (Conway) game is an ordered pair of sets of games, called the **options** for the players "left" and "right". It will be said that a game **x** is a **prelude** to a game **y** if **x** is one of the left or right options of the game **y**. Formally the relation **PRELUDE** is defined as follows.

```
In[12]:= union[composite[id[GAMES], inverse[FIRST], E],
            composite[id[GAMES], inverse[SECOND], E]] := PRELUDE
```

Theorem. Domain of the **PRELUDE** relation.

```
In[13]:= SubstTest[domain, union[u, v], {u -> composite[id[GAMES], inverse[FIRST], E],
             v -> composite[id[GAMES], inverse[SECOND], E]}] // Reverse

Out[13]= domain[PRELUDE] == GAMES
```

```
In[14]:= domain[PRELUDE] := GAMES
```

Theorem. An upper bound for the range of the **PRELUDE** relation.

```
In[15]:= Map[subclass[#, GAMES] &,
            SubstTest[range, union[u, v], {u -> composite[id[GAMES], inverse[FIRST], E],
              v -> composite[id[GAMES], inverse[SECOND], E]}]] // Reverse

Out[15]= subclass[range[PRELUDE], GAMES] == True
```

```
In[16]:= subclass[range[PRELUDE], GAMES] := True
```

simplification rules for PRELUDE

Theorem. Simpllification rule.

```
In[17]:= SubstTest[composite, Id, union[u, v], {u -> composite[id[GAMES], inverse[FIRST], E],
             v -> composite[id[GAMES], inverse[SECOND], E]}] // Reverse

Out[17]= composite[Id, PRELUDE] == PRELUDE
```

```
In[18]:= composite[Id, PRELUDE] := PRELUDE
```

Theorem. Normalization rule for the inverse of the **PRELUDE** relation.

```
In[19]:= SubstTest[inverse, union[u, v], {u -> composite[id[GAMES], inverse[FIRST], E],
            v -> composite[id[GAMES], inverse[SECOND], E]}]

Out[19]= union[composite[inverse[E], FIRST, id[GAMES]],
            composite[inverse[E], SECOND, id[GAMES]]] == inverse[PRELUDE]
```

```
In[20]:= union[composite[inverse[E], FIRST, id[GAMES]],
            composite[inverse[E], SECOND, id[GAMES]]] := inverse[PRELUDE]
```

Theorem. Simplification rule.

```
In[21]:= SubstTest[composite, id[GAMES],
            union[u, v], {u -> composite[id[GAMES], inverse[FIRST], E],
             v -> composite[id[GAMES], inverse[SECOND], E]}] // Reverse
Out[21]= composite[id[GAMES], PRELUDE] == PRELUDE

In[22]:= composite[id[GAMES], PRELUDE] := PRELUDE
```

Theorem. Simplification rule.

```
In[23]:= ImageComp[id[GAMES], PRELUDE, x] // Reverse
Out[23]= intersection[GAMES, image[PRELUDE, x]] == image[PRELUDE, x]

In[24]:= intersection[GAMES, image[PRELUDE, x_]] := image[PRELUDE, x]
```

Theorem. Simplification rule.

```
In[25]:= IminComp[id[GAMES], PRELUDE, x] // Reverse
Out[25]= image[inverse[PRELUDE], intersection[GAMES, x]] == image[inverse[PRELUDE], x]

In[26]:= image[inverse[PRELUDE], intersection[GAMES, x_]] := image[inverse[PRELUDE], x]
```

Theorem. Simplification rule.

```
In[27]:= ImageComp[PRELUDE, id[GAMES], x] // Reverse
Out[27]= image[PRELUDE, intersection[GAMES, x]] == image[PRELUDE, x]

In[28]:= image[PRELUDE, intersection[GAMES, x_]] := image[PRELUDE, x]
```

Theorem. Simplification rule.

```
In[29]:= IminComp[PRELUDE, id[GAMES], x] // Reverse
Out[29]= intersection[GAMES, image[inverse[PRELUDE], x]] == image[inverse[PRELUDE], x]

In[30]:= intersection[GAMES, image[inverse[PRELUDE], x_]] := image[inverse[PRELUDE], x]
```

Lemma. Simplification rule.

```
In[31]:= equiv[and[member[x, GAMES], member[first[x], V]], member[x, GAMES]]
Out[31]= True

In[32]:= and[member[x_, GAMES], member[first[x_], V]] := member[x, GAMES]
```

Theorem. Membership rule.

```
In[33]:= SubstTest[member, pair[x, y], union[u, v], {u -> composite[id[GAMES], inverse[FIRST], E],
           v -> composite[id[GAMES], inverse[SECOND], E]}] // Reverse

Out[33]= member[pair[x, y], PRELUDE] == or[and[member[x, first[y]], member[y, GAMES]],
           and[member[x, second[y]], member[y, GAMES]]]

In[34]:= member[pair[x_, y_], PRELUDE] := or[and[member[x, first[y]], member[y, GAMES]],
           and[member[x, second[y]], member[y, GAMES]]]
```

birthdays of preludes

It is shown in this section that the preludes (left and right options) of a game were born before the game itself.

Theorem. If a game **x** is alive at time **t**, then **first[x] ⊂ U[image[gamerec,t]]**.

```
In[35]:= Map[implies[#, subclass[first[x], U[image[gamerec, ord[t]]]]] &,
           SubstTest[member, x, cartsq[w], w → P[U[image[gamerec, ord[t]]]]] // Reverse

Out[35]= or[not[member[x, APPLY[gamerec, ord[t]]]],
           subclass[first[x], U[image[gamerec, ord[t]]]]] == True

In[36]:= or[not[member[x_, APPLY[gamerec, ord[t_]]]],
           subclass[first[x_], U[image[gamerec, ord[t_]]]]] := True
```

Dual. If a game **x** is alive at time **t**, then **second[x] ⊂ U[image[gamerec,t]]**.

```
In[37]:= Map[implies[#, subclass[second[x], U[image[gamerec, ord[t]]]]] &,
           SubstTest[member, x, cartsq[w], w → P[U[image[gamerec, ord[t]]]]] // Reverse

Out[37]= or[not[member[x, APPLY[gamerec, ord[t]]]],
           subclass[second[x], U[image[gamerec, ord[t]]]]] == True

In[38]:= or[not[member[x_, APPLY[gamerec, ord[t_]]]],
           subclass[second[x_], U[image[gamerec, ord[t_]]]]] := True
```

Corollary. (The **ord** wrapper is removed, and the above result is then specializied to the birthday of the game. This has a redundant literal.)

```
In[39]:= (SubstTest[implies, equal[t, ord[w]], or[not[member[x, APPLY[gamerec, t]]],
             subclass[first[x], U[image[gamerec, t]]]], w → t] /. t → APPLY[BDAY, x]) // Reverse

Out[39]= or[not[member[x, GAMES]], not[member[x, APPLY[gamerec, APPLY[BDAY, x]]]],
           subclass[first[x], U[image[gamerec, APPLY[BDAY, x]]]]] == True

In[40]:= (% /. x → x_) /. Equal → SetDelayed
```

The following result is better.

Theorem.

```
In[41]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 -> member[x, GAMES], p2 -> member[x, APPLY[gamerec, APPLY[BDAY, x]]],
            p3 -> subclass[first[x], U[image[gamerec, APPLY[BDAY, x]]]]}]] // Reverse

Out[41]= or[not[member[x, GAMES]], subclass[first[x], U[image[gamerec, APPLY[BDAY, x]]]]] == True

In[42]:= or[not[member[x_, GAMES]],
           subclass[first[x_], U[image[gamerec, APPLY[BDAY, x_]]]]] := True
```

Lemma for the dual.

```
In[43]:= (SubstTest[implies, equal[t, ord[w]], or[not[member[x, APPLY[gamerec, t]]],
             subclass[second[x], U[image[gamerec, t]]]], w → t] /. t → APPLY[BDAY, x]) // Reverse

Out[43]= or[not[member[x, GAMES]], not[member[x, APPLY[gamerec, APPLY[BDAY, x]]]],
           subclass[second[x], U[image[gamerec, APPLY[BDAY, x]]]]] == True

In[44]:= (% /. x → x_) /. Equal → SetDelayed
```

Dual Theorem.

```
In[45]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 -> member[x, GAMES], p2 -> member[x, APPLY[gamerec, APPLY[BDAY, x]]],
            p3 -> subclass[second[x], U[image[gamerec, APPLY[BDAY, x]]]]}]] // Reverse

Out[45]= or[not[member[x, GAMES]], subclass[second[x], U[image[gamerec, APPLY[BDAY, x]]]]] == True

In[46]:= or[not[member[x_, GAMES]],
           subclass[second[x_], U[image[gamerec, APPLY[BDAY, x_]]]]] := True
```

Lemma.

```
In[47]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → range[BDAY], v → OMEGA, w -> union[image[S, set[ord[t]]], ord[t]]}] // Reverse

Out[47]= subclass[range[BDAY], union[image[S, set[ord[t]]], ord[t]]] == True

In[48]:= (% /. t → t_) /. Equal → SetDelayed
```

Theorem.

```
In[49]:= Map[subclass[image[#, ord[t]], ord[t]] &,
          Assoc[BDAY, inverse[BDAY], composite[complement[E], id[OMEGA]]]]

Out[49]= subclass[image[BDAY, U[image[gamerec, ord[t]]]], ord[t]] == True

In[50]:= subclass[image[BDAY, U[image[gamerec, ord[t_]]]], ord[t_]] := True
```

Lemma. (The **ord** wrapper is removed, and the result is then specializied to the birthday of the game.)

```
In[51]:= (SubstTest[implies, equal[t, ord[w]], subclass[image[BDAY, U[image[gamerec, t]]], t],
            w → t] // Reverse) /. t → APPLY[BDAY, x]

Out[51]= or[not[member[x, GAMES]],
           subclass[image[BDAY, U[image[gamerec, APPLY[BDAY, x]]]], APPLY[BDAY, x]]] == True

In[52]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma.

```
In[53]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p1, p4],
            implies[and[p3, p4], p5], not[implies[p1, p5]], {p1 -> member[x, GAMES],
             p2 -> subclass[first[x], U[image[gamerec, APPLY[BDAY, x]]]], p3 →
              subclass[image[BDAY, first[x]], image[BDAY, U[image[gamerec, APPLY[BDAY, x]]]]],
             p4 -> subclass[image[BDAY, U[image[gamerec, APPLY[BDAY, x]]]], APPLY[BDAY, x]],
             p5 → subclass[image[BDAY, first[x]], APPLY[BDAY, x]]}]] // Reverse

Out[53]= or[not[member[x, GAMES]], subclass[image[BDAY, first[x]], APPLY[BDAY, x]]] == True

In[54]:= (% /. x → x_) /. Equal→ SetDelayed
```

Dual lemma.

```
In[55]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p1, p4],
            implies[and[p3, p4], p5], not[implies[p1, p5]], {p1 -> member[x, GAMES],
             p2 -> subclass[second[x], U[image[gamerec, APPLY[BDAY, x]]]], p3 →
              subclass[image[BDAY, second[x]], image[BDAY, U[image[gamerec, APPLY[BDAY, x]]]]],
             p4 -> subclass[image[BDAY, U[image[gamerec, APPLY[BDAY, x]]]], APPLY[BDAY, x]],
             p5 → subclass[image[BDAY, second[x]], APPLY[BDAY, x]]}]] // Reverse

Out[55]= or[not[member[x, GAMES]], subclass[image[BDAY, second[x]], APPLY[BDAY, x]]] == True

In[56]:= (% /. x → x_) /. Equal→ SetDelayed
```

The preceding two lemmas both contain a redundant literal. The following result allows one to eliminate this literal.

Lemma.

```
In[57]:= SubstTest[implies, equal[v, V], subclass[u, v], v -> APPLY[BDAY, x]] // Reverse

Out[57]= or[member[x, GAMES], subclass[u, APPLY[BDAY, x]]] == True

In[58]:= (% /. {x → x_, u → u_}) /. Equal→ SetDelayed
```

Theorem. The left options of a game were born before the game itself.

```
In[59]:= SubstTest[and, implies[p, q], or[p, q],
           {p -> member[x, GAMES], q -> subclass[image[BDAY, first[x]], APPLY[BDAY, x]]}]

Out[59]= subclass[image[BDAY, first[x]], APPLY[BDAY, x]] == True

In[60]:= subclass[image[BDAY, first[x_]], APPLY[BDAY, x_]] := True
```

Dual Theorem. The right options of a game were born before the game itself.

```
In[61]:= SubstTest[and, implies[p, q], or[p, q],
          {p -> member[x, GAMES], q -> subclass[image[BDAY, second[x]], APPLY[BDAY, x]]}]

Out[61]= subclass[image[BDAY, second[x]], APPLY[BDAY, x]] == True

In[62]:= subclass[image[BDAY, second[x_]], APPLY[BDAY, x_]] := True
```

The next step is to eliminate the variable **x**.

Theorem.

```
In[63]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, x, case[subclass[image[t, first[x]], APPLY[t, x]]], t → BDAY]]

Out[63]= subclass[GAMES, fix[composite[inverse[BDAY], S, IMAGE[BDAY], FIRST]]] == True

In[64]:= subclass[GAMES, fix[composite[inverse[BDAY], S, IMAGE[BDAY], FIRST]]] := True
```

Dual theorem.

```
In[65]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, x, case[subclass[image[t, second[x]], APPLY[t, x]]], t → BDAY]]

Out[65]= subclass[GAMES, fix[composite[inverse[BDAY], S, IMAGE[BDAY], SECOND]]] == True

In[66]:= subclass[GAMES, fix[composite[inverse[BDAY], S, IMAGE[BDAY], SECOND]]] := True
```

Corollary.

```
In[67]:= subclass[composite[id[GAMES], inverse[FIRST], E],
            composite[inverse[BDAY], E, BDAY]] // AssertTest // InvertFix

Out[67]= subclass[composite[id[GAMES], inverse[FIRST], E],
           composite[inverse[BDAY], E, BDAY]] == True

In[68]:= subclass[composite[id[GAMES], inverse[FIRST], E],
           composite[inverse[BDAY], E, BDAY]] := True
```

Dual corollary.

```
In[69]:= subclass[composite[id[GAMES], inverse[SECOND], E],
            composite[inverse[BDAY], E, BDAY]] // AssertTest // InvertFix

Out[69]= subclass[composite[id[GAMES], inverse[SECOND], E],
           composite[inverse[BDAY], E, BDAY]] == True

In[70]:= subclass[composite[id[GAMES], inverse[SECOND], E],
           composite[inverse[BDAY], E, BDAY]] := True
```

The preceding two results can be combined as follows.

Theorem. The relation **PRELUDE** is a subclass of the well-founded relation **inverse[BDAY] ∘ E ∘ BDAY**.

```
In[71]:= SubstTest[subclass, union[u, v], composite[inverse[BDAY], E, BDAY],
           {u -> composite[id[GAMES], inverse[FIRST], E],
            v -> composite[id[GAMES], inverse[SECOND], E]}] // Reverse

Out[71]= subclass[PRELUDE, composite[inverse[BDAY], E, BDAY]] == True

In[72]:= subclass[PRELUDE, composite[inverse[BDAY], E, BDAY]] := True
```

Theorem. The relation **PRELUDE** is well-founded.

```
In[73]:= SubstTest[implies, and[subclass[u, v], WELLFOUNDED[v]], WELLFOUNDED[u],
           {u → PRELUDE, v -> composite[inverse[BDAY], E, BDAY]}] // Reverse

Out[73]= WELLFOUNDED[PRELUDE] == True

In[74]:= WELLFOUNDED[PRELUDE] := True
```

VERTSECT rule

Theorem.

```
In[75]:= SubstTest[VERTSECT, union[u, v], {u → inverse[composite[id[GAMES], inverse[FIRST], E]],
            v → inverse[composite[id[GAMES], inverse[SECOND], E]]}] // Reverse

Out[75]= VERTSECT[inverse[PRELUDE]] ==
          union[cart[complement[GAMES], set[0]], composite[CUP, id[GAMES]]]

In[76]:= VERTSECT[inverse[PRELUDE]] :=
          union[cart[complement[GAMES], set[0]], composite[CUP, id[GAMES]]]
```

Corollary. The inverse of **PRELUDE** is thin. No new rewrite rule is needed for this.

```
In[77]:= thin[inverse[PRELUDE]]

Out[77]= True
```