gamerec is one-to-one

Johan G. F. Belinfante
2012 May 25

```
In[1]:= SetDirectory["l:"]; << goedel.12may22a
```

```
:Package Title: goedel.12may22a                    2012 May 22 at 2:40 p.m.

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

It is now:  2012 May 27 at 5:2

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 27 at 5:19
```

summary

If **ord[u] ∈ ord[v]**, then **APPLY[gamerec, ord[u]]** is a proper subset of **APPLY[gamerec, ord[v]]**. It follows that the function **gamerec** is strictly monotone, and therefore one-to-one.

derivation

Lemma. The set **APPLY[gamerec, ord[t]]** can not satisfy the recursion equation that **GAMES** satisfies because it is a set.

```
In[2]:= Map[not, SubstTest[implies, subclass[cartsq[P[w]], w],
           not[member[w, V]], w -> APPLY[gamerec, ord[t]]]] // Reverse
```

```
Out[2]= subclass[cart[P[APPLY[gamerec, ord[t]]], P[APPLY[gamerec, ord[t]]],
          APPLY[gamerec, ord[t]]] == False
```

```
In[3]:= (% /. t → t_) /. Equal → SetDelayed
```

Observation. The following inclusion holds.

```
In[4]:= subclass[U[image[gamerec, ord[t]]], APPLY[gamerec, ord[t]]]
```

```
Out[4]= True
```

Theorem. The inclusion **U[image[gamerec, ord[t]]] ⊂ APPLY[gamerec, ord[t]]** is proper.

```
In[5]:= Map[not, SubstTest[implies, equal[w, U[image[gamerec, ord[t]]]], subclass[
            cartsq[P[w]], APPLY[gamerec, ord[t]]], w → APPLY[gamerec, ord[t]]]] // Reverse

Out[5]= equal[APPLY[gamerec, ord[t]], U[image[gamerec, ord[t]]]] == False

In[6]:= equal[APPLY[gamerec, ord[t_]], U[image[gamerec, ord[t_]]]] := False
```

Corollary.

```
In[7]:= SubstTest[equal, APPLY[gamerec, ord[w]],
          U[image[gamerec, ord[w]]], w → succ[ord[t]]] // Reverse

Out[7]= equal[APPLY[gamerec, ord[t]], APPLY[gamerec, succ[ord[t]]]] == False

In[8]:= equal[APPLY[gamerec, ord[t_]], APPLY[gamerec, succ[ord[t_]]]] := False
```

Theorem.

```
In[9]:= SubstTest[implies, not[member[ord[w], ord[t]]],
          subclass[APPLY[gamerec, ord[t]], APPLY[gamerec, ord[w]]], w → succ[ord[t]]] // Reverse

Out[9]= subclass[APPLY[gamerec, ord[t]], APPLY[gamerec, succ[ord[t]]]] == True

In[10]:= subclass[APPLY[gamerec, ord[t_]], APPLY[gamerec, succ[ord[t_]]]] := True
```

Lemma.

```
In[11]:= SubstTest[implies, subclass[ord[t], ord[v]],
           subclass[APPLY[gamerec, ord[t]], APPLY[gamerec, ord[v]]], t → succ[ord[u]]] // Reverse

Out[11]= or[not[member[ord[u], ord[v]]],
           subclass[APPLY[gamerec, succ[ord[u]]], APPLY[gamerec, ord[v]]]] == True

In[12]:= or[not[member[ord[u_], ord[v_]]],
           subclass[APPLY[gamerec, succ[ord[u_]]], APPLY[gamerec, ord[v_]]]] := True
```

Theorem.

```
In[13]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> APPLY[gamerec, succ[ord[t]]], v -> APPLY[gamerec, ord[t]]}] // Reverse

Out[13]= subclass[APPLY[gamerec, succ[ord[t]]], APPLY[gamerec, ord[t]]] == False

In[14]:= subclass[APPLY[gamerec, succ[ord[t_]]], APPLY[gamerec, ord[t_]]] := False
```

Theorem.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p2, p3], not[implies[p1, p3]], {p1 → member[ord[u], ord[v]],
             p2 -> subclass[APPLY[gamerec, succ[ord[u]]], APPLY[gamerec, ord[v]]],
             p3 → not[equal[APPLY[gamerec, ord[u]], APPLY[gamerec, ord[v]]]]}]] // Reverse

Out[15]= or[not[equal[APPLY[gamerec, ord[u]], APPLY[gamerec, ord[v]]]],
           not[member[ord[u], ord[v]]]] == True
```

```
In[16]:= (% /. {u → u_, v → v_}) /. Equal → SetDelayed
```

Corollary.

```
In[17]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], not[implies[p1, p4]],
           {p1 -> equal[APPLY[gamerec, ord[u]], APPLY[gamerec, ord[v]]],
            p2 -> not[member[ord[u], ord[v]]], p3 -> not[member[ord[u], ord[v]]],
            p4 → equal[ord[u], ord[v]]}]] // Reverse

Out[17]= or[equal[ord[u], ord[v]],
          not[equal[APPLY[gamerec, ord[u]], APPLY[gamerec, ord[v]]]]] == True
```

```
In[18]:= (% /. {u → u_, v → v_}) /. Equal → SetDelayed
```

Theorem. (A better rewrite rule.)

```
In[19]:= equiv[equal[APPLY[gamerec, ord[u]], APPLY[gamerec, ord[v]]], equal[ord[u], ord[v]]]

Out[19]= True
```

```
In[20]:= equal[APPLY[gamerec, ord[u_]], APPLY[gamerec, ord[v_]]] := equal[ord[u], ord[v]]
```

A variable-free restatement of this will now be derived.

Lemma. (Remove the **ord** wrappers.)

```
In[21]:= SubstTest[implies, and[equal[u, ord[x]], equal[v, ord[y]]], or[equal[u, v],
           not[equal[APPLY[gamerec, u], APPLY[gamerec, v]]]], {x → u, y → v}] // Reverse

Out[21]= or[equal[u, v], not[equal[APPLY[gamerec, u], APPLY[gamerec, v]]],
          not[member[u, OMEGA]], not[member[v, OMEGA]]] == True
```

```
In[22]:= (% /. {u → u_, v → v_}) /. Equal → SetDelayed
```

Theorem. (A variable-free restatement.)

```
In[23]:= Map[equal[V, domain[#]] &, SubstTest[reify, t, case[or[equal[first[t], second[t]],
           not[equal[APPLY[funpart[z], first[t]], APPLY[funpart[z], second[t]]]],
           not[member[first[t], w]], not[member[second[t], w]]]], {w → OMEGA, z → gamerec}]]

Out[23]= subclass[cart[OMEGA, OMEGA],
          union[Id, composite[complement[inverse[gamerec]], gamerec]]] == True
```

```
In[24]:= % /. Equal → SetDelayed
```

A better formulation can be derived.

Lemma.

```
In[25]:= SubstTest[implies, subclass[u, v],
           subclass[image[t, u], image[t, v]], {t → id[Di], u -> cart[OMEGA, OMEGA],
            v -> union[Id, composite[complement[inverse[gamerec]], gamerec]]}] // Reverse

Out[25]= subclass[composite[id[OMEGA], Di, id[OMEGA]],
           composite[complement[inverse[gamerec]], gamerec]] == True

In[26]:= % /. Equal→ SetDelayed
```

Theorem. The function **gamerec** is one-to-one.

```
In[27]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
           {u -> composite[id[OMEGA], Di, id[OMEGA]], v ->
             composite[complement[inverse[gamerec]], gamerec], w → inverse[gamerec]}] // Reverse

Out[27]= FUNCTION[inverse[gamerec]] == True

In[28]:= FUNCTION[inverse[gamerec]] := True
```

strict monotonicity

Lemma.

```
In[29]:= Map[not, SubstTest[and, implies[and[p1, p2], not[p3]],
            implies[p2, p4], implies[p4, p3], not[implies[p1, not[p2]]],
            {p1 → subclass[APPLY[gamerec, ord[u]], APPLY[gamerec, ord[v]]],
             p2 → member[ord[v], ord[u]],
             p3 → subclass[APPLY[gamerec, ord[v]], APPLY[gamerec, ord[u]]],
             p4 → not[member[ord[u], ord[v]]]}]] // Reverse

Out[29]= or[not[member[ord[v], ord[u]]],
           not[subclass[APPLY[gamerec, ord[u]], APPLY[gamerec, ord[v]]]]] == True

In[30]:= (% /. {u → u_, v → v_}) /. Equal→ SetDelayed
```

Theorem. A better formulation.

```
In[31]:= equiv[subclass[APPLY[gamerec, ord[u]], APPLY[gamerec, ord[v]]],
           not[member[ord[v], ord[u]]]]

Out[31]= True

In[32]:= subclass[APPLY[gamerec, ord[u_]], APPLY[gamerec, ord[v_]]] :=
           not[member[ord[v], ord[u]]]
```

Lemma. (Remove **ord** wrappers.)

```
In[33]:= SubstTest[implies, and[equal[u, ord[x]], equal[v, ord[y]]],
           implies[subclass[APPLY[gamerec, u], APPLY[gamerec, v]], subclass[u, v]],
           {x → u, y → v}] // Reverse

Out[33]= or[not[member[u, OMEGA]], not[member[v, OMEGA]],
           not[subclass[APPLY[gamerec, u], APPLY[gamerec, v]]], subclass[u, v]] == True

In[34]:= (% /. {u → u_, v → v_}) /. Equal → SetDelayed
```

Lemma. (A variable-free restatement.)

```
In[35]:= Map[equal[V, domain[#]] &, SubstTest[reify, t, case[or[subclass[first[t], second[t]],
             not[subclass[APPLY[funpart[z], first[t]], APPLY[funpart[z], second[t]]]],
             not[member[first[t], w]], not[member[second[t], w]]], {w → OMEGA, z → gamerec}]]

Out[35]= subclass[cart[OMEGA, OMEGA],
           union[S, composite[inverse[gamerec], complement[S], gamerec]]] == True

In[36]:= % /. Equal → SetDelayed
```

A better formulation can be derived.

Lemma.

```
In[37]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → id[complement[S]], u -> cart[OMEGA, OMEGA],
            v -> union[S, composite[inverse[gamerec], complement[S], gamerec]]}] // Reverse

Out[37]= subclass[composite[inverse[E], id[OMEGA]],
           composite[inverse[gamerec], complement[S], gamerec]] == True

In[38]:= % /. Equal → SetDelayed
```

Lemma.

```
In[39]:= Map[composite[gamerec, #] &,
           Assoc[composite[id[OMEGA], complement[S]], id[OMEGA], inverse[gamerec]]] // Reverse

Out[39]= composite[inverse[E], IMAGE[gamerec], inverse[gamerec]] ==
          composite[gamerec, complement[S], inverse[gamerec]]

In[40]:= % /. Equal → SetDelayed
```

Theorem. Strict monotonicity property.

```
In[41]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → coflip[cross[gamerec, gamerec]], u -> composite[inverse[E], id[OMEGA]],
            v -> composite[inverse[gamerec], complement[S], gamerec]}] // Reverse

Out[41]= subclass[composite[inverse[gamerec], S, gamerec], S] == True

In[42]:= subclass[composite[inverse[gamerec], S, gamerec], S] := True
```