game wrapper

Johan G. F. Belinfante
2012 May 15

```
In[1]:= SetDirectory["l:"]; << goedel.12may08a
```

```
:Package Title: goedel.12may08a                    2012 May 8 at 12:10 noon

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

It is now:  2012 May 15 at 8:5

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 15 at 8:22
```

summary

A **game** wrapper is introduced for Conway games.

game wrapper

The wrapper **game[x]** is defined as follows.

```
In[2]:= PAIR[intersection[GAMES, setpart[first[x_]]],
          intersection[GAMES, setpart[second[x_]]]] := game[x]
```

Theorem. Wrapper introduction rule for **game[x]**.

```
In[3]:= SubstTest[member, PAIR[u, v], cart[t, t], {u → intersection[GAMES, setpart[first[x]]],
           v -> intersection[GAMES, setpart[second[x]]], t → P[GAMES]}] // Reverse
```

```
Out[3]= member[game[x], GAMES] == True
```

```
In[4]:= member[game[x_], GAMES] := True
```

The **GOEDEL** program has two separate ordered pair constructors. The following theorem provides some flexibility on this score when dealing with the **game** wrapper.

Theorem. A variant of the definition of the **game** wrapper.

```
In[5]:= SubstTest[equal, pair[u, v], PAIR[u, v], {u → intersection[GAMES, setpart[first[x]]],
          v -> intersection[GAMES, setpart[second[x]]]}] // Reverse

Out[5]= equal[game[x], pair[intersection[GAMES, setpart[first[x]]],
          intersection[GAMES, setpart[second[x]]]]] == True

In[6]:= pair[intersection[GAMES, setpart[first[x_]]],
         intersection[GAMES, setpart[second[x_]]]] := game[x]
```

---

properties of game[x]

Theorem. Sethood.

```
In[7]:= SubstTest[implies, member[t, w], member[t, V], {t → game[x], w → GAMES}] // Reverse

Out[7]= member[game[x], V] == True

In[8]:= member[game[x_], V] := True
```

Theorem.

```
In[9]:= Map[or[subclass[first[game[x]], GAMES], #] &,
         SubstTest[member, game[x], cartsq[t], t → P[GAMES]]]

Out[9]= subclass[first[game[x]], GAMES] == True

In[10]:= subclass[first[game[x_]], GAMES] := True
```

Theorem.

```
In[11]:= Map[or[subclass[second[game[x]], GAMES], #] &,
          SubstTest[member, game[x], cartsq[t], t → P[GAMES]]]

Out[11]= subclass[second[game[x]], GAMES] == True

In[12]:= subclass[second[game[x_]], GAMES] := True
```

Theorem.

```
In[13]:= SubstTest[member, game[x], cartsq[t], t → P[GAMES]]

Out[13]= member[first[game[x]], V] == True

In[14]:= member[first[game[x_]], V] := True
```

Theorem.

```
In[15]:= Map[equal[first[x], #] &,
           SubstTest[first, PAIR[u, v], {u → intersection[GAMES, setpart[first[x]]],
             v -> intersection[GAMES, setpart[second[x]]]}]] // Reverse

Out[15]= equal[first[x], first[game[x]]] == and[member[first[x], V], subclass[first[x], GAMES]]
```

```
In[16]:= equal[first[x_], first[game[x_]]] :=
          and[member[first[x], V], subclass[first[x], GAMES]]
```

Theorem.

```
In[17]:= Map[equal[second[x], #] &,
           SubstTest[second, PAIR[u, v], {u → intersection[GAMES, setpart[first[x]]],
             v -> intersection[GAMES, setpart[second[x]]]}]] // Reverse

Out[17]= equal[second[x], second[game[x]]] ==
          and[member[first[x], V], subclass[second[x], GAMES]]

In[18]:= equal[second[x_], second[game[x_]]] :=
          and[member[first[x], V], subclass[second[x], GAMES]]
```

---

wrapper elimination rule

Theorem.

```
In[19]:= SubstTest[member, x, cartsq[t], t → P[GAMES]]

Out[19]= and[member[first[x], V], subclass[first[x], GAMES], subclass[second[x], GAMES]] ==
          member[x, GAMES]

In[20]:= and[member[first[x_], V], subclass[first[x_], GAMES],
           subclass[second[x_], GAMES]] := member[x, GAMES]
```

Lemma. Simplification rule.

```
In[21]:= equiv[and[member[x, GAMES], member[first[x], V]], member[x, GAMES]]

Out[21]= True

In[22]:= and[member[x_, GAMES], member[first[x_], V]] := member[x, GAMES]
```

Theorem.

```
In[23]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
            {p1 -> member[x, GAMES], p2 -> equal[x, PAIR[first[x], second[x]]],
             p3 -> equal[PAIR[first[x], second[x]], pair[first[x], second[x]]],
             p4 → equal[x, pair[first[x], second[x]]]}]] // Reverse

Out[23]= or[equal[x, pair[first[x], second[x]]], not[member[x, GAMES]]] == True

In[24]:= or[equal[x_, pair[first[x_], second[x_]]], not[member[x_, GAMES]]] := True
```

Lemma.

```
In[25]:= SubstTest[equal, PAIR[t, u], PAIR[v, w],
          {t → first[x], u → second[x], v -> intersection[GAMES, setpart[first[x]]],
           w -> intersection[GAMES, setpart[second[x]]]}]

Out[25]= equal[cart[set[first[x]], set[second[x]]],
           cart[set[intersection[GAMES, setpart[first[x]]]],
            set[intersection[GAMES, setpart[second[x]]]]]] ==
          equal[game[x], PAIR[first[x], second[x]]]

In[26]:= equal[cart[set[first[x_]], set[second[x_]]],
           cart[set[intersection[GAMES, setpart[first[x_]]]],
            set[intersection[GAMES, setpart[second[x_]]]]]] :=
          equal[game[x], PAIR[first[x], second[x]]]
```

Theorem.

```
In[27]:= SubstTest[implies, and[equal[t, v], equal[u, w]], equal[PAIR[t, u], PAIR[v, w]],
           {t → first[x], u → second[x], v -> intersection[GAMES, setpart[first[x]]],
            w -> intersection[GAMES, setpart[second[x]]]}] // Reverse

Out[27]= or[equal[game[x], PAIR[first[x], second[x]]], not[member[x, GAMES]]] == True

In[28]:= or[equal[game[x_], PAIR[first[x_], second[x_]]], not[member[x_, GAMES]]] := True
```

Lemma.

```
In[29]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
            {p1 → member[x, GAMES], p2 -> equal[game[x], PAIR[first[x], second[x]]],
             p3 -> equal[x, PAIR[first[x], second[x]]], p4 → equal[x, game[x]]}]] // Reverse

Out[29]= or[equal[x, game[x]], not[member[x, GAMES]]] == True

In[30]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Wrapper elimination rule for **game[x]**.

```
In[31]:= equiv[equal[x, game[x]], member[x, GAMES]]

Out[31]= True

In[32]:= equal[x_, game[x_]] := member[x, GAMES]
```

Corollary.

```
In[42]:= equal[game[game[x]], game[x]]

Out[42]= True

In[44]:= game[game[x_]] := game[x]
```

Theorem. Another propery of the game wrapper

```
In[33]:= Map[not, SubstTest[implies, member[t, w], not[equal[t, V]], {t → game[x], w → GAMES}]] //
          Reverse

Out[33]= equal[V, game[x]] == False

In[34]:= equal[V, game[x_]] := False
```

---

reify rule

Theorem.

```
In[35]:= SubstTest[reify, x, PAIR[intersection[t, setpart[first[f[x]]]],
            intersection[t, setpart[second[f[x]]]], t → GAMES] // Reverse

Out[35]= reify[x, game[f[x]]] ==
          union[cart[union[complement[domain[VERTSECT[reify[x, f[x]]]]], complement[
              image[inverse[VERTSECT[reify[x, f[x]]]], cart[V, V]]]], PAIR[0, 0]], composite[
            inverse[E], cross[IMAGE[id[GAMES]], IMAGE[id[GAMES]]], VERTSECT[reify[x, f[x]]]]]

In[36]:= reify[x_, game[y_]] :=
          union[cart[union[complement[domain[VERTSECT[reify[x, y]]]], complement[
              image[inverse[VERTSECT[reify[x, y]]], cart[V, V]]]], PAIR[0, 0]], composite[
            inverse[E], cross[IMAGE[id[GAMES]], IMAGE[id[GAMES]]], VERTSECT[reify[x, y]]]]
```