membership rule for iterate[g,s]

Johan G. F. Belinfante
Revised 2002 April 17

```
<< goedel52.n23; << tests.m

:Package Title: GOEDEL52.N23     2002 April 15 at 12:40 midnite

It is now:  2002 Apr 17 at 8:50

Loading Simplification Rules

TESTS.M                   Revised 2002 April 16

weightlimit = 40


Context switch to ‘Goedel‘Private is needed for ReplaceTest

Just ignore the error message about Unterminated use of BeginPackage

Get::bebal : Unterminated uses of BeginPackage or Begin in << tests.m.
```

■ Introduction

To facilitate definition by iteration, we introduce a new relation **iterate[g,s]** whose domain is a subset of the set **omega** of natural numbers. The vertical section of this relation at a natural number **n** is a class, which can be thought of as the **n**−th generation, which for non−zero **n** is obtained from the preceding generation by some relation **g**. For **n = 0**, the vertical section is the specified class **s**, the starting generation. The basic tool used is **subvar** which was discussed in detail at the FTP2000 meeting in St. Andrews. Briefly this is:

```
class[y, subclass[y, image[x, y]]]

subvar[x]
```

■ Adding the new rule

Before adding a new wrapped membership rule on the fly, one first needs to temporarily remove the default wrapped membership rule.

```
class[u_, member[u_, x_]] =.
```

We can now add the new wrapped membership rule, which is our definition of **iterate[g,s]**:

```
class[w_, member[x_, iterate[y_, z_]]] :=
    Module[{p = Unique[]}, class[w, exists[p,
                and[member[x, p], subclass[p, union[cart[singleton[0], z],
                           composite[y, p, inverse[SUCC], id[omega]]]]]]]]
```

Restore the default rule.

```
class[u_, member[u_, x_]] := x /; FreeQ[x, u] && AtomQ[u]
```

■ Lemmas needed

Verifying that this proposed definition of **iterate[g,s]** can be rewritten in terms of **subvar** is hard without some lemmas which we provide here. The first lemma is:

```
SubstTest[fix, composite[inverse[IMAGE[SWAP]], x, w, IMAGE[SWAP]],
    w -> IMAGE[id[cart[z, y]]]]
```

```
fix[composite[inverse[IMAGE[SWAP]], x, IMAGE[SWAP], IMAGE[id[cart[y, z]]]]] ==
 image[inverse[IMAGE[SWAP]], fix[composite[x, IMAGE[id[cart[z, y]]]]]]
```

```
fix[composite[inverse[IMAGE[SWAP]], x_, IMAGE[SWAP], IMAGE[id[cart[y_, z_]]]]] :=
 image[inverse[IMAGE[SWAP]], fix[composite[x, IMAGE[id[cart[z, y]]]]]]
```

Lemma 2:

```
SubstTest[subvar, composite[SWAP, z, SWAP],
  z -> union[cross[x, y], id[cart[u, v]]]] // Reverse
```

```
intersection[image[inverse[IMAGE[SWAP]], subvar[union[cross[x, y], id[cart[u, v]]]]],
  P[cart[V, V]]] == subvar[union[cross[y, x], id[cart[v, u]]]]
```

```
intersection[
  image[inverse[IMAGE[SWAP]], subvar[union[cross[x_, y_], id[cart[u_, v_]]]]],
  P[cart[V, V]]] := subvar[union[cross[y, x], id[cart[v, u]]]]
```

Lemma 3:

```
subvar[union[x, id[intersection[y, z]]]] // Normality // Reverse
```

```
intersection[subvar[union[x, id[y]]], subvar[union[x, id[z]]]] ==
 subvar[union[x, id[intersection[y, z]]]]
```

```
intersection[subvar[union[x_, id[y_]]], subvar[union[x_, id[z_]]]] :=
 subvar[union[x, id[intersection[y, z]]]]
```

Lemma 4:

```
intersection[image[inverse[IMAGE[SWAP]],
   subvar[union[cross[x, y], id[intersection[complement[cart[V, complement[u]]],
       complement[cart[complement[v], V]]]]]], P[cart[V, V]]] // Normality
```

```
intersection[image[inverse[IMAGE[SWAP]],
   subvar[union[cross[x, y], id[intersection[complement[cart[V, complement[u]]],
       complement[cart[complement[v], V]]]]]],
  P[cart[V, V]]] == subvar[union[cross[y, x], id[cart[u, v]]]]
```

```
intersection[image[inverse[IMAGE[SWAP]],
   subvar[union[cross[x_, y_], id[intersection[complement[cart[V, complement[u_]]],
       complement[cart[complement[v_], V]]]]]],
  P[cart[V, V]]] := subvar[union[cross[y, x], id[cart[u, v]]]]
```

■ The biggie:

With these lemmas in place we can verify the basic formula connecting **iterate[g,s]** and **subvar**.

```
iterate[g, s] // Normality // InvertFix
```

```
iterate[g, s] ==
 U[subvar[union[cross[composite[id[omega], SUCC], g], id[cart[singleton[0], s]]]]]
```

```
U[subvar[union[cross[composite[id[omega], SUCC], g_], id[cart[singleton[0], s_]]]]] :=
 iterate[g, s]
```

■ The starting generation

We show here that the starting generation is the specified class **s**.

```
SubstTest[subclass, fix[w], U[subvar[w]],
    w -> union[cross[composite[id[omega], SUCC], g], id[cart[singleton[0], s]]]]
```

```
subclass[cart[singleton[0], s], iterate[g, s]] == True
```

```
subclass[cart[singleton[0], s_], iterate[g_, s_]] := True
```

```
SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
    {u -> cart[singleton[0], s], v -> iterate[g, s], w -> singleton[0]}]
```

```
subclass[s, image[iterate[g, s], singleton[0]]] == True
```

```
subclass[s_, image[iterate[g_, s_], singleton[0]]] := True
```

```
SubstTest[subclass, U[subvar[w]], range[w],
    w -> union[cross[composite[id[omega], SUCC], g], id[cart[singleton[0], s]]]]
```

```
subclass[iterate[g, s],
  union[cart[intersection[omega, complement[singleton[0]]], range[g]],
   cart[singleton[0], s]]] == True
```

```
subclass[iterate[g_, s_],
  union[cart[intersection[omega, complement[singleton[0]]], range[g_]],
   cart[singleton[0], s_]]] := True
```

```
SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
    {u -> iterate[g, s],
  v -> union[cart[intersection[omega, complement[singleton[0]]], range[g]],
    cart[singleton[0], s]], w -> singleton[0]}]
```

```
subclass[image[iterate[g, s], singleton[0]], s] == True
```

```
subclass[image[iterate[g_, s_], singleton[0]], s_] := True
```

```
SubstTest[and, subclass[u, s], subclass[s, u],
u -> image[iterate[g, s], singleton[0]]] // Reverse
```

```
equal[s, image[iterate[g, s], singleton[0]]] == True
```

Thus:

```
image[iterate[g_, s_], singleton[0]] := s
```

■ iterate[g,s] is a relation

```
Map[subclass[#, cart[V, V]] &, SubstTest[image, union[u, v], U[subvar[x]],
x -> union[u, v]] /.
  {u -> id[cart[singleton[0], s]], v -> cross[composite[id[omega], SUCC], g]}]
```

```
subclass[iterate[g, s], cart[V, V]] == True
```

```
subclass[iterate[g_, s_], cart[V, V]] := True
```

```
equal[composite[Id, iterate[g, s]], iterate[g, s]]
```

```
True
```

```
composite[Id, iterate[g_, s_]] := iterate[g, s]
```

■ Domain

We show that the domain of **iterate[g,s]** is a subset of the natural numbers.

```
Map[subclass[domain[#], omega] &, SubstTest[image, union[u, v], U[subvar[x]],
x -> union[u, v]] /.
  {u -> id[cart[singleton[0], s]], v -> cross[composite[id[omega], SUCC], g]}]
```

```
subclass[domain[iterate[g, s]], omega] == True
```

```
subclass[domain[iterate[g_, s_]], omega] := True
```

```
equal[intersection[domain[iterate[g, s]], omega], domain[iterate[g, s]]]
```

```
True
```

```
intersection[omega, domain[iterate[g_, s_]]] := domain[iterate[g, s]]
```

```
Assoc[iterate[g, s], id[domain[iterate[g, s]]], id[omega]] // Reverse
```

```
composite[iterate[g, s], id[omega]] == iterate[g, s]
```

```
composite[iterate[g_, s_], id[omega]] := iterate[g, s]
```

■ Monotonicity

One of the monotonicity properties of **iterate[g,s]** is established here.

```
SubstTest[implies, subclass[u, v], subclass[subvar[u], subvar[v]],
    {u -> union[x, id[y]], v -> union[x, id[z]]}]
```

```
or[not[subclass[y, union[z, fix[x]]]],
  subclass[subvar[union[x, id[y]]], subvar[union[x, id[z]]]]] == True
```

```
or[not[subclass[y_, union[z_, fix[x_]]]],
  subclass[subvar[union[x_, id[y_]]], subvar[union[x_, id[z_]]]]] := True

Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
        implies[p3, p4], not[implies[p1, p4]],
        {p1 -> subclass[y, z], p2 -> subclass[y, union[z, fix[x]]],
            p3 -> subclass[subvar[union[x, id[y]]], subvar[union[x, id[z]]]],
            p4 -> subclass[U[subvar[union[x, id[y]]]], U[subvar[union[x, id[z]]]]]}]]

or[not[subclass[y, z]],
  subclass[U[subvar[union[x, id[y]]]], U[subvar[union[x, id[z]]]]] == True

or[not[subclass[y_, z_]],
  subclass[U[subvar[union[x_, id[y_]]]], U[subvar[union[x_, id[z_]]]]] := True

SubstTest[implies, subclass[y, z],
    subclass[U[subvar[union[x, id[y]]]], U[subvar[union[x, id[z]]]]],
    {x -> cross[composite[id[omega], SUCC], g],
        y -> cart[singleton[0], s], z -> cart[singleton[0], t]}]

or[not[subclass[s, t]], subclass[iterate[g, s], iterate[g, t]]] == True

or[not[subclass[s_, t_]], subclass[iterate[g_, s_], iterate[g_, t_]]] := True
```

■ Recursion Formula

```
Map[composite[#, SUCC] &,
 iterate[g, s] ==
  union[cart[singleton[0], s], composite[g, iterate[g, s], inverse[SUCC], id[omega]]]]

composite[iterate[g, s], SUCC] == composite[g, iterate[g, s]]
```

■ The case s = 0.

It is shown that if the starting generation is empty, then so are all generations.

```
SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
 {u → domain[iterate[g, s]], v → omega, w → SUCC}]

subclass[image[SUCC, domain[iterate[g, s]]], omega] == True

subclass[image[SUCC, domain[iterate[g_, s_]]], omega] := True

SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
 {u → image[inverse[iterate[g, s]], t], v -> domain[iterate[g, s]], w → SUCC}]

subclass[image[SUCC, image[inverse[iterate[g, s]], t]],
  image[SUCC, domain[iterate[g, s]]]] == True

subclass[image[SUCC, image[inverse[iterate[g_, s_]], t_]],
  image[SUCC, domain[iterate[g_, s_]]]] := True

SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
 {u -> image[SUCC, image[inverse[iterate[g, s]], t]],
  v -> image[SUCC, domain[iterate[g, s]]], w → omega}]

subclass[image[SUCC, image[inverse[iterate[g, s]], t]], omega] == True
```

```
subclass[image[SUCC, image[inverse[iterate[g_, s_]], t_]], omega] := True
```

```
equal[intersection[omega, image[SUCC, image[inverse[iterate[g, s]], t]]],
 image[SUCC, image[inverse[iterate[g, s]], t]]]
```

```
True
```

```
intersection[omega, image[SUCC, image[inverse[iterate[g_, s_]], t_]]] :=
 image[SUCC, image[inverse[iterate[g, s]], t]]
```

```
SubstTest[implies, and[subclass[x, y], member[y, V]], member[x, V],
 {x → domain[iterate[g, s]], y → omega}]
```

```
member[domain[iterate[g, s]], V] == True
```

```
member[domain[iterate[g_, s_]], V] := True
```

```
Map[subclass[#, image[SUCC, domain[iterate[g, 0]]]] &,
  Map[domain, SubstTest[image, w, U[subvar[w]],
     w → union[id[cart[singleton[0], s]], cross[composite[id[omega], SUCC], g]]]] /.
   s → 0] // Reverse
```

```
subclass[domain[iterate[g, 0]], image[SUCC, domain[iterate[g, 0]]]] == True
```

```
subclass[domain[iterate[g_, 0]], image[SUCC, domain[iterate[g_, 0]]]] := True
```

```
AssInt[subvar[SUCC], REGULAR, P[omega]]
```

```
intersection[P[omega], subvar[SUCC]] == singleton[0]
```

```
intersection[P[omega], subvar[SUCC]] := singleton[0]
```

```
SubstTest[member, w, intersection[u, v],
 {u → subvar[SUCC], v → P[omega], w → domain[iterate[g, 0]]}]
```

```
equal[0, domain[iterate[g, 0]]] == True
```

```
domain[iterate[g_, 0]] := 0
```

```
SubstTest[composite, w, id[domain[w]], w → iterate[g, 0]] // Reverse
```

```
iterate[g, 0] == 0
```

```
iterate[g_, 0] := 0
```

■ Agenda: Other Examples.

We intend to work out other examples, both interesting ones and trivial ones, but we do not do so here. For example, one would expect that if the starting generation is the singleton of **0** and **g = SUCC**, we ought to get **iterate[-SUCC,singleton[0]] = id[omega]**, and so on.