iterating an acyclic function

Johan G. F. Belinfante
2006 February 14

```
In[1]:= SetDirectory["l:"]; << goedel78.14a; << tools.m

        :Package Title: goedel78.14a           2006 February 14 at 9:00 a.m.

        It is now:  2006 Feb 14 at 15:1

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 February 3

        weightlimit = 40
```

summary

If **x** is an acyclic function, then **iterate[x, set[y]]** is one-to-one.

derivation

Lemma. A shifting rule for **iterate**. Later vertical sections of **iterate[x,y]** are bounded in terms of earlier ones.

```
In[2]:= SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
         {u → set[pair[s, t]], v → composite[id[omega], E], w → iterate[x, y]}]

Out[2]= or[not[member[s, t]], not[member[t, omega]], subclass[
          image[iterate[x, y], set[t]], image[trv[x], image[iterate[x, y], set[s]]]]] == True

In[3]:= (% /. {s → s_, t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

If **x** is acyclic, one has:

```
In[4]:= SubstTest[implies, subclass[u, v],
         subclass[composite[u, w], composite[v, w]], {u → trv[x], v → Di, w → iterate[x, y]}]

Out[4]= or[not[equal[0, fix[trv[x]]]],
          subclass[composite[trv[x], iterate[x, y]], composite[Di, iterate[x, y]]]] == True

In[5]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
          {u -> composite[trv[x], iterate[x, y]], v -> composite[Di, iterate[x, y]], w → z}]

Out[6]= or[not[subclass[composite[trv[x], iterate[x, y]], composite[Di, iterate[x, y]]]],
           subclass[image[trv[x], image[iterate[x, y], z]],
            image[Di, image[iterate[x, y], z]]]] == True

In[7]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Combining the above lemmas yields the following condition on later sections when one iterates an acyclic relation.

```
In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p4, p5],
           implies[and[p3, p5], p6], not[implies[and[p1, p4], p6]], {p1 -> equal[0, fix[trv[x]]],
            p2 -> subclass[composite[trv[x], iterate[x, y]], composite[Di, iterate[x, y]]],
            p3 -> subclass[image[trv[x], image[iterate[x, y], set[u]]],
              image[Di, image[iterate[x, y], set[u]]]],
            p4 → and[member[u, v], member[v, omega]], p5 -> subclass[
              image[iterate[x, y], set[v]], image[trv[x], image[iterate[x, y], set[u]]]],
            p6 → subclass[image[iterate[x, y], set[v]],
              image[Di, image[iterate[x, y], set[u]]]]}]]

Out[8]= or[not[equal[0, fix[trv[x]]]], not[member[u, v]], not[member[v, omega]],
           subclass[image[iterate[x, y], set[v]], image[Di, image[iterate[x, y], set[u]]]]] == True

In[9]:= or[not[equal[0, fix[trv[x_]]]], not[member[u_, v_]], not[member[v_, omega]], subclass[
           image[iterate[x_, y_], set[v_]], image[Di, image[iterate[x_, y_], set[u_]]]]] := True
```

---

acyclic functions

Corollary:

```
In[10]:= Map[or[#, not[member[u, domain[iterate[funpart[w], set[z]]]]], not[equal[
              APPLY[iterate[funpart[w], set[z]], u], APPLY[iterate[funpart[w], set[z]], v]]] &,
           SubstTest[or, not[equal[0, fix[trv[x]]]], not[member[u, v]], not[member[v, omega]],
            subclass[image[iterate[x, y], set[v]], image[Di, image[iterate[x, y], set[u]]]],
            {x → funpart[w], y → set[z]}]]

Out[10]= or[not[equal[0, fix[trv[funpart[w]]]]],
            not[equal[APPLY[iterate[funpart[w], set[z]], u],
              APPLY[iterate[funpart[w], set[z]], v]]], not[member[u, v]],
            not[member[u, domain[iterate[funpart[w], set[z]]]]], not[member[v, omega]]] == True

In[11]:= (% /. {u → u_, v → v_, w → w_, z → z_}) /. Equal → SetDelayed
```

To eliminate the variables **u** and **v**, one needs to reformulate the above without **APPLY**. The following provides a step in this direction.

```
In[12]:= member[pair[u, v], composite[inverse[funpart[z]], funpart[z]]] // AssertTest

Out[12]= member[pair[u, v], composite[inverse[funpart[z]], funpart[z]]] ==
          and[equal[APPLY[funpart[z], u], APPLY[funpart[z], v]],
           member[u, domain[funpart[z]]], member[v, V]]

In[13]:= member[pair[u_, v_], composite[inverse[funpart[z_]], funpart[z_]]] :=
          and[equal[APPLY[funpart[z], u], APPLY[funpart[z], v]],
           member[u, domain[funpart[z]]], member[v, V]]
```

Applying this to the case of **iterate** yields:

```
In[14]:= Map[implies[#, equal[APPLY[iterate[funpart[x], set[y]], u],
             APPLY[iterate[funpart[x], set[y]], v]]] &, SubstTest[member, pair[u, v],
           composite[inverse[funpart[z]], funpart[z]], z → iterate[funpart[x], set[y]]]]

Out[14]= or[equal[APPLY[iterate[funpart[x], set[y]], u], APPLY[iterate[funpart[x], set[y]], v]],
           not[member[pair[u, v], composite[
              inverse[iterate[funpart[x], set[y]]], iterate[funpart[x], set[y]]]]]] == True

In[15]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[16]:= SubstTest[implies, member[pair[u, v], composite[z, y]],
          member[v, range[z]], z → inverse[x]]

Out[16]= or[member[v, domain[x]], not[member[pair[u, v], composite[inverse[x], y]]]] == True

In[17]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

The following result accomplishes the desired elimination of **APPLY**.

```
In[18]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[p1, p4], implies[p4, p5], implies[and[p2, p3, p5, p6], p7],
           not[implies[and[p1, p6], p7]], {p1 -> member[pair[u, v],
              composite[inverse[iterate[funpart[x], set[y]]], iterate[funpart[x], set[y]]]],
            p2 -> equal[APPLY[iterate[funpart[x], set[y]], u],
              APPLY[iterate[funpart[x], set[y]], v]],
            p3 -> member[u, domain[iterate[funpart[x], set[y]]]],
            p4 → member[v, domain[iterate[funpart[x], set[y]]]],
            p5 → member[v, omega],
            p6 -> and[equal[0, fix[trv[funpart[x]]]], member[y, domain[funpart[x]]]],
            p7 → not[member[u, v]]}]]

Out[18]= or[not[equal[0, fix[trv[funpart[x]]]]], not[member[u, v]],
           not[member[y, domain[funpart[x]]]], not[member[pair[u, v], composite[
              inverse[iterate[funpart[x], set[y]]], iterate[funpart[x], set[y]]]]]] == True

In[19]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

The next lemma is a step toward symmetrizing the above with respect to the variables **u** and **v**.

```
In[20]:= Map[implies[member[pair[v, u], composite[inverse[y], x]], #] &,
          SubstTest[member, pair[u, v], inverse[z], z → composite[inverse[y], x]]]

Out[20]= or[member[pair[u, v], composite[inverse[x], y]],
           not[member[pair[v, u], composite[inverse[y], x]]]] == True

In[21]:= or[member[pair[u_, v_], composite[inverse[x_], y_]],
           not[member[pair[v_, u_], composite[inverse[y_], x_]]]] := True
```

One can now conclude that not only can **u** not be less than **v**, but also **v** cannot be less than **u**.

```
In[22]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[and[p1, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 -> and[equal[0, fix[trv[funpart[x]]]], member[y, domain[funpart[x]]]],
            p2 -> member[pair[u, v],
              composite[inverse[iterate[funpart[x], set[y]]], iterate[funpart[x], set[y]]]],
            p3 -> member[pair[v, u], composite[inverse[iterate[funpart[x], set[y]]],
               iterate[funpart[x], set[y]]]],
            p4 -> not[member[v, u]]}]]

Out[22]= or[not[equal[0, fix[trv[funpart[x]]]]], not[member[v, u]],
           not[member[y, domain[funpart[x]]]], not[member[pair[u, v], composite[
              inverse[iterate[funpart[x], set[y]]], iterate[funpart[x], set[y]]]]]] == True

In[23]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Using the law of trichotomy for natural numbers, one concludes that **u** and **v** must be equal.

```
In[24]:= Map[not, SubstTest[and, implies[p1, p7], implies[p1, p8],
           implies[p1, p3], implies[p1, p4], implies[p3, p5], implies[p4, p6],
           implies[and[p5, p6, p7, p8], p9], not[implies[p1, p9]], {p1 -> and[
              equal[0, fix[trv[funpart[x]]]], member[y, domain[funpart[x]]], member[pair[u, v],
               composite[inverse[iterate[funpart[x], set[y]]], iterate[funpart[x], set[y]]]]],
            p3 -> member[u, domain[iterate[funpart[x], set[y]]]],
            p4 → member[v, domain[iterate[funpart[x], set[y]]]],
            p5 → member[u, omega], p6 → member[v, omega], p7 → not[member[u, v]],
            p8 -> not[member[v, u]], p9 → equal[u, v]}]]

Out[24]= or[equal[u, v], not[equal[0, fix[trv[funpart[x]]]]],
           not[member[y, domain[funpart[x]]]], not[member[pair[u, v], composite[
              inverse[iterate[funpart[x], set[y]]], iterate[funpart[x], set[y]]]]]] == True

In[25]:= (% /. {u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

eliminating the variables u and v

Lemma.

```
In[26]:= SubstTest[fix, composite[u, id[omega]], u → composite[z, iterate[x, y]]] // Reverse

Out[26]= intersection[omega, fix[composite[z, iterate[x, y]]]] == fix[composite[z, iterate[x, y]]]
```

```
In[27]:= intersection[omega, fix[composite[z_, iterate[x_, y_]]]] :=
          fix[composite[z, iterate[x, y]]]

In[28]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[u, v], implies[and[member[y, q], equal[0, r],
              member[u, omega], member[v, omega], member[pair[u, v], w]], equal[u, v]],
            {q -> domain[funpart[x]], r → fix[trv[funpart[x]]],
             w -> composite[inverse[iterate[funpart[x], set[y]]],
               iterate[funpart[x], set[y]]]}]] // Reverse

Out[28]= or[FUNCTION[inverse[iterate[funpart[x], set[y]]]],
           not[equal[0, fix[trv[funpart[x]]]], not[member[y, domain[funpart[x]]]]] == True

In[29]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Remove the **funpart** wrapper.

```
In[30]:= SubstTest[implies,
          and[equal[x, funpart[w]], equal[0, fix[trv[x]]], member[y, domain[x]]],
          FUNCTION[inverse[iterate[x, set[y]]]], w → x]

Out[30]= or[FUNCTION[inverse[iterate[x, set[y]]]],
           not[equal[0, fix[trv[x]]]], not[FUNCTION[x]], not[member[y, domain[x]]]] == True

In[31]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The following lemma permits one to remove the membership hypothesis about **y**.

```
In[32]:= SubstTest[implies, equal[z, set[PAIR[0, y]]],
          FUNCTION[inverse[z]], z → iterate[x, set[y]]]

Out[32]= or[FUNCTION[inverse[iterate[x, set[y]]]], member[y, domain[x]]] == True

In[33]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The main result now follows:

```
In[34]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], or[p2, p3],
           not[implies[p1, p3]], {p1 → and[FUNCTION[x], equal[0, fix[trv[x]]]],
            p2 → member[y, domain[x]], p3 → FUNCTION[inverse[iterate[x, set[y]]]]}]]

Out[34]= or[FUNCTION[inverse[iterate[x, set[y]]]],
           not[equal[0, fix[trv[x]]]], not[FUNCTION[x]]] == True

In[35]:= or[FUNCTION[inverse[iterate[x_, set[y_]]]],
           not[equal[0, fix[trv[x_]]]], not[FUNCTION[x_]]] := True
```