transvariance theorem about iterate

Johan G. F. Belinfante
2005 April 17

```
In[1]:= SetDirectory["i:"]; << goedel68.16a; << tools.m

        :Package Title: goedel68.16a          2005 April 16 at 6:00 p.m.

        It is now:  2005 Apr 17 at 22:24

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 April 16

        weightlimit = 40
```

summary

It is shown that if **x** and **y** commute, and if **z** is **(x, y)**-transvariant, then **iterate[x,z]** is a subclass of **iterate[y,z]**. Several corollaries are derived from this.

commutativity lemma

The lemma to be proved below follows from the observation that if **x** and **y** commute, then any power of **x** commutes with **y**.

```
In[2]:= SubstTest[implies, equal[u, v], equal[image[u, z], image[v, z]],
          {u → composite[image[power[x], w], y], v → composite[y, image[power[x], w]]}]

Out[2]= or[equal[image[y, image[iterate[x, z], w]], image[iterate[x, image[y, z]], w]],
          not[equal[composite[y, image[power[x], w]],
            composite[image[power[x], w], y]]]] == True

In[3]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed

In[4]:= Map[not,
         SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
          {p1 → commute[x, y], p2 → commute[image[power[x], w], y], p3 -> equal[
             image[y, image[iterate[x, z], w]], image[iterate[x, image[y, z]], w]]}]]

Out[4]= or[equal[image[y, image[iterate[x, z], w]], image[iterate[x, image[y, z]], w]],
          not[equal[composite[x, y], composite[y, x]]]] == True
```

```
In[5]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

---

transvariance

Lemma 2a.

```
In[6]:= SubstTest[implies, subclass[u, v],
         subclass[iterate[y, u], iterate[y, v]], {u → image[x, z], v → image[y, z]}]

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

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

Lemma 2b.

```
In[8]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
         {u -> iterate[y, image[x, z]], v -> composite[y, iterate[y, z]]}]

Out[8]= or[not[subclass[iterate[y, image[x, z]], composite[y, iterate[y, z]]]],
          subclass[image[iterate[y, image[x, z]], w],
           image[y, image[iterate[y, z], w]]]] == True

In[9]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

Theorem 2.

```
In[10]:= Map[not,
          SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> subclass[image[x, z], image[y, z]],
            p2 -> subclass[iterate[y, image[x, z]], composite[y, iterate[y, z]]],
            p3 -> subclass[image[iterate[y, image[x, z]], w],
              image[y, image[iterate[y, z], w]]]}]]

Out[10]= or[not[subclass[image[x, z], image[y, z]]],
           subclass[image[iterate[y, image[x, z]], w],
            image[y, image[iterate[y, z], w]]]] == True

In[11]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

the main derivation

```
In[12]:= Map[not, SubstTest[and, implies[p3, p4],
           implies[p1, p5], implies[and[p4, p5], p6], implies[p2, p7],
           implies[and[p6, p7], p8], not[implies[and[p1, p2, p3], p8]],
           {p1 → commute[x, y], p2 → subclass[image[x, z], image[y, z]],
            p3 → subclass[image[iterate[x, z], w], image[iterate[y, z], w]],
            p4 -> subclass[image[x, image[iterate[x, z], w]],
              image[x, image[iterate[y, z], w]]],
            p5 → equal[image[x, image[iterate[y, z], w]],
              image[iterate[y, image[x, z]], w]],
            p6 → subclass[image[x, image[iterate[x, z], w]],
              image[iterate[y, image[x, z]], w]],
            p7 -> subclass[image[iterate[y, image[x, z]], w],
              image[y, image[iterate[y, z], w]]],
            p8 -> subclass[image[x, image[iterate[x, z], w]],
              image[y, image[iterate[y, z], w]]]}]]

Out[12]= or[not[equal[composite[x, y], composite[y, x]]],
           not[subclass[image[x, z], image[y, z]]],
           not[subclass[image[iterate[x, z], w], image[iterate[y, z], w]]],
           subclass[image[x, image[iterate[x, z], w]],
            image[y, image[iterate[y, z], w]]]] == True

In[13]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

eliminating w

The next step is to eliminate the variable **w**. For this one can specialize to singletons. The following observation is the key.

```
In[14]:= class[w, subclass[image[x, set[w]], image[y, set[w]]]]

Out[14]= complement[fix[composite[complement[inverse[y]], x]]]
```

It is useful to introduce this abbreviation

```
In[15]:= temp[x_, y_, z_] := complement[
           fix[composite[complement[inverse[iterate[y, z]]], iterate[x, z]]]]
```

The following membership rule is introduced on a temporary basis.

```
In[16]:= member[w, fix[composite[complement[inverse[x]], y]]] // AssertTest

Out[16]= member[w, fix[composite[complement[inverse[x]], y]]] ==
          not[subclass[image[y, set[w]], image[x, set[w]]]]

In[17]:= member[w_, fix[composite[complement[inverse[x_]], y_]]] :=
          not[subclass[image[y, set[w]], image[x, set[w]]]]
```

Restatement of the theorem proved.

```
In[18]:= implies[and[commute[x, y], transvariant[x, y, z], member[w, temp[x, y, z]]],
          member[succ[w], temp[x, y, z]]]

Out[18]= True
```

This restatement is the key to the elimination of **w**. The following takes a while:

```
In[19]:= Map[equal[V, #] &, SubstTest[class, w, implies[
             and[equal[u, v], subclass[r, s], member[w, t]], member[succ[w], t]],
            {t → temp[x, y, z], u → composite[x, y], v → composite[y, x],
             r → image[x, z], s → image[y, z]}]] // Reverse

Out[19]= or[not[equal[composite[x, y], composite[y, x]]],
           not[subclass[image[x, z], image[y, z]]], subclass[image[inverse[SUCC],
             fix[composite[complement[inverse[iterate[y, z]]], iterate[x, z]]]],
            fix[composite[complement[inverse[iterate[y, z]]], iterate[x, z]]]]] == True

In[20]:= (% /. {x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

---

application of induction

The following negative form of induction will be used.

```
In[21]:= SubstTest[implies, INDUCTIVE[y], subclass[omega, y], y → complement[x]]

Out[21]= or[equal[0, intersection[omega, x]], member[0, x],
           not[subclass[image[inverse[SUCC], x], x]]] == True

In[22]:= or[equal[0, intersection[omega, x_]], member[0, x_],
           not[subclass[image[inverse[SUCC], x_], x_]]] := True
```

In the case at hand, the base case is trivial:

```
In[23]:= SubstTest[or, equal[0, intersection[omega, w]], member[0, w],
          not[subclass[image[inverse[SUCC], w], w]], w → complement[temp[x, y, z]]]

Out[23]= or[not[subclass[image[inverse[SUCC],
              fix[composite[complement[inverse[iterate[y, z]]], iterate[x, z]]]],
             fix[composite[complement[inverse[iterate[y, z]]], iterate[x, z]]]],
           subclass[iterate[x, z], iterate[y, z]]] == True

In[24]:= (% /. {x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

The main theorem follows:

```
In[25]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → and[commute[x, y], transvariant[x, y, z]],
            p2 → subclass[image[inverse[SUCC],
               fix[composite[complement[inverse[iterate[y, z]]], iterate[x, z]]]],
              fix[composite[complement[inverse[iterate[y, z]]], iterate[x, z]]]],
            p3 → subclass[omega, temp[x, y, z]]}]]

Out[25]= or[not[equal[composite[x, y], composite[y, x]]],
           not[subclass[image[x, z], image[y, z]]],
           subclass[iterate[x, z], iterate[y, z]]] == True

In[26]:= or[not[equal[composite[x_, y_], composite[y_, x_]]],
           not[subclass[image[x_, z_], image[y_, z_]]],
           subclass[iterate[x_, z_], iterate[y_, z_]]] := True
```

Restatement.

```
In[27]:= implies[and[commute[x, y], transvariant[x, y, z]],
          subclass[iterate[x, z], iterate[y, z]]]

Out[27]= True
```

---

an iterate formula

Lemma.

```
In[28]:= Assoc[cross[Id, z], cross[inverse[x], Id], power[x]]

Out[28]= composite[cross[Id, composite[z, x]], power[x]] ==
          composite[cross[inverse[x], z], power[x]]

In[29]:= composite[cross[Id, composite[z, x]], power[x]] :=
          composite[cross[inverse[x], z], power[x]]
```

The uniqueness of iteration yields the following formula which resembles an existing one.

```
In[30]:= SubstTest[implies,
          and[equal[composite[w, SUCC], composite[u, w]], equal[image[w, set[0]], v]],
          equal[composite[w, id[omega]], iterate[u, v]],
          {u -> cross[x, Id], v -> composite[Id, y],
           w → composite[cross[Id, y], power[inverse[x]]]}]

Out[30]= equal[composite[SWAP, cross[y, Id], power[x]],
           iterate[cross[x, Id], composite[Id, y]]] == True

In[31]:= iterate[cross[x_, Id], composite[Id, y_]] :=
          composite[SWAP, cross[y, Id], power[x]]
```

---

subtwine corollaries

The condition **equal[composite[x,y],composite[y,z]]** is often described as an **intertwining** condition. Replacing equal with subclass yields what one might call a **subtwining** condition. In this section some corollaries of the transvariance theorem are deduced which have a subtwining condition as hypothesis.

```
In[32]:= SubstTest[implies, and[commute[u, v], transvariant[u, v, w]],
          subclass[iterate[u, w], iterate[v, w]],
          {u → cross[Id, x], v → cross[inverse[z], Id], w → composite[Id, y]}]

Out[32]= or[not[subclass[composite[x, y], composite[y, z]]],
           subclass[composite[cross[inverse[y], Id], power[x]],
            composite[cross[Id, y], power[z]]]] == True

In[33]:= or[not[subclass[composite[x_, y_], composite[y_, z_]]],
           subclass[composite[cross[inverse[y_], Id], power[x_]],
            composite[cross[Id, y_], power[z_]]]] := True
```

Corollary.

```
In[37]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
          {u -> composite[cross[inverse[y], Id], power[x]],
           v -> composite[cross[Id, y], power[z]]}]

Out[37]= or[not[subclass[composite[cross[inverse[y], Id], power[x]],
             composite[cross[Id, y], power[z]]]], subclass[
            composite[image[power[x], w], y], composite[y, image[power[z], w]]]] == True

In[38]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

This is the basic subtwining theorem

```
In[39]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → subclass[composite[x, y], composite[y, z]],
            p2 -> subclass[composite[cross[inverse[y], Id], power[x]],
              composite[cross[Id, y], power[z]]],
            p3 -> subclass[composite[image[power[x], w], y],
              composite[y, image[power[z], w]]]}]]

Out[39]= or[not[subclass[composite[x, y], composite[y, z]]], subclass[
           composite[image[power[x], w], y], composite[y, image[power[z], w]]]] == True

In[40]:= or[not[subclass[composite[x_, y_], composite[y_, z_]]],
           subclass[composite[image[power[x_], w_], y_],
            composite[y_, image[power[z_], w_]]]] := True
```

Corollary.

```
In[45]:= SubstTest[or, not[subclass[composite[x, y], composite[y, z]]],
          subclass[composite[image[power[x], w], y], composite[y, image[power[z], w]]],
          w → complement[set[0]]]

Out[45]= or[not[subclass[composite[x, y], composite[y, z]]],
           subclass[composite[trv[x], y], composite[y, trv[z]]]] == True

In[46]:= or[not[subclass[composite[x_, y_], composite[y_, z_]]],
           subclass[composite[trv[x_], y_], composite[y_, trv[z_]]]] := True
```

---

another such result

Theorem.

```
In[42]:= SubstTest[implies, and[commute[u, v], transvariant[u, v, w]],
          subclass[iterate[u, w], iterate[v, w]],
          {u → cross[inverse[z], Id], v → cross[Id, x], w → composite[Id, y]}]

Out[42]= or[not[subclass[composite[y, z], composite[x, y]]],
           subclass[composite[cross[Id, y], power[z]],
            composite[cross[inverse[y], Id], power[x]]]] == True

In[43]:= or[not[subclass[composite[y_, z_], composite[x_, y_]]],
           subclass[composite[cross[Id, y_], power[z_]],
            composite[cross[inverse[y_], Id], power[x_]]]] := True
```

Corollary.

```
In[48]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
          {u -> composite[cross[Id, y], power[z]],
           v -> composite[cross[inverse[y], Id], power[x]]}]

Out[48]= or[not[subclass[composite[cross[Id, y], power[z]],
             composite[cross[inverse[y], Id], power[x]]], subclass[
            composite[y, image[power[z], w]], composite[image[power[x], w], y]]] == True

In[49]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal -> SetDelayed

In[50]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → subclass[composite[y, z], composite[x, y]],
            p2 -> subclass[composite[cross[Id, y], power[z]],
              composite[cross[inverse[y], Id], power[x]]],
            p3 -> subclass[composite[y, image[power[z], w]],
              composite[image[power[x], w], y]]}]]

Out[50]= or[not[subclass[composite[y, z], composite[x, y]]], subclass[
            composite[y, image[power[z], w]], composite[image[power[x], w], y]]] == True

In[51]:= or[not[subclass[composite[y_, z_], composite[x_, y_]]],
           subclass[composite[y_, image[power[z_], w_]],
            composite[image[power[x_], w_], y_]]] := True

In[52]:= SubstTest[or, not[subclass[composite[y, z], composite[x, y]]],
          subclass[composite[y, image[power[z], w]], composite[image[power[x], w], y]],
          w → complement[set[0]]]

Out[52]= or[not[subclass[composite[y, z], composite[x, y]]],
           subclass[composite[y, trv[z]], composite[trv[x], y]]] == True

In[53]:= or[not[subclass[composite[y_, z_], composite[x_, y_]]],
           subclass[composite[y_, trv[z_]], composite[trv[x_], y_]]] := True
```

---

the intertwining case

The two subtwining theorems can be combined to yield a theorem about intertwining relations.

In[58]:=
```
Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
  implies[p2, p4], implies[p3, p5], implies[and[p4, p5], p6],
  not[implies[p1, p6]], {p1 → equal[composite[x, y], composite[y, z]],
   p2 → subclass[composite[x, y], composite[y, z]],
   p3 → subclass[composite[y, z], composite[x, y]],
   p4 -> subclass[composite[trv[x], y], composite[y, trv[z]]],
   p5 -> subclass[composite[y, trv[z]], composite[trv[x], y]],
   p6 → equal[composite[trv[x], y], composite[y, trv[z]]]}]]
```

Out[58]=
```
or[equal[composite[y, trv[z]], composite[trv[x], y]],
  not[equal[composite[x, y], composite[y, z]]]] == True
```

In[60]:=
```
or[equal[composite[y_, trv[z_]], composite[trv[x_], y_]],
  not[equal[composite[x_, y_], composite[y_, z_]]]] := True
```