# FINITE Induction

Johan G. F. Belinfante
2003 December 1

```
In[1]:= << goedel52.t17; << tools.m

        :Package Title: goedel52.t17      2003 November 13 at 10:55 p.m.

        It is now:  2003 Dec 1 at 9:24

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 November 15

        weightlimit = 40
```

## summary

This notebook is devoted to a derivation of the theorem on **FINITE** induction, loosely following **Otter**'s proof, using Theorem **CUP−SUB** to derive Lemma **SBV−PS3**. Among the improvements made here is an earlier introduction of the cover relation **K = dif[PS, composite[PS,PS]]**, where **PS = dif[S, Id]** is the proper subset relation.

## a formula connecting CUP and range[SINGLETON] with K

The **GOEDEL** program currently contains a connection between **K** and **range[SINGLETON]**, but it involves **DIF**, not **CUP**.

```
In[2]:= intersection[S, inverse[image[inverse[DIF], range[SINGLETON]]]]

Out[2]= K
```

Lemma 1.

```
In[3]:= Assoc[inverse[S], SINGLETON, inverse[E]]

Out[3]= composite[inverse[E], IMAGE[inverse[S]], IMAGE[SINGLETON]] ==
         union[cart[complement[singleton[0]], singleton[0]],
          composite[inverse[E], IMAGE[SINGLETON]]]

In[4]:= composite[inverse[E], IMAGE[inverse[S]], IMAGE[SINGLETON]] :=
         union[cart[complement[singleton[0]], singleton[0]],
          composite[inverse[E], IMAGE[SINGLETON]]]
```

Lemma 2.

```
In[5]:= fix[composite[inverse[DIF], inverse[E], IMAGE[SINGLETON], FIRST]] // Renormality

Out[5]= fix[composite[inverse[DIF], inverse[E], IMAGE[SINGLETON], FIRST]] ==
         image[inverse[DIF], range[SINGLETON]]
```

```
In[6]:= fix[composite[inverse[DIF], inverse[E], IMAGE[SINGLETON], FIRST]] :=
         image[inverse[DIF], range[SINGLETON]]
```

The connection with **CUP** follows by applying a rotation argument:

```
In[7]:= SubstTest[composite, rotate[rotate[x]],
         id[cart[V, range[SINGLETON]]], inverse[FIRST], x -> rotate[CUP]]

Out[7]= composite[CUP, id[cart[V, range[SINGLETON]]], inverse[FIRST]] ==
         union[K, id[complement[singleton[0]]]]

In[8]:= composite[CUP, id[cart[V, range[SINGLETON]]], inverse[FIRST]] :=
         union[K, id[complement[singleton[0]]]]
```

The following corollary is also used in the derivation of **FINITE** induction.

```
In[9]:= ImageComp[composite[CUP, id[cart[V, range[SINGLETON]]]], inverse[FIRST], x] // Reverse

Out[9]= image[CUP, cart[x, range[SINGLETON]]] ==
         union[image[K, x], intersection[x, complement[singleton[0]]]]

In[10]:= image[CUP, cart[x_, range[SINGLETON]]] :=
          union[image[K, x], intersection[x, complement[singleton[0]]]]
```

In particular, this implies:

```
In[11]:= subclass[image[CUP, cart[x, range[SINGLETON]]], x]

Out[11]= subclass[image[K, x], x]
```

## two lemmas from Otter's proof of Theorem CUP−ADJ

Unlike **Otter**'s proof of Theorem **CUP−SUB**, which used Theorem **CUP−ADJ**, the present derivation is more direct, but it does require two lemmas used in **Otter**'s proof of Theorem **CUP−ADJ**.

Lemma 1.

```
In[12]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
             {u -> pair[dif[y, singleton[x]], singleton[x]],
           v -> cart[z, range[SINGLETON]], w -> image[inverse[CUP], z]}]

Out[12]= or[member[union[y, singleton[x]], z], not[member[x, V]],
           not[member[intersection[y, complement[singleton[x]]], z]],
           not[subclass[image[K, z], z]] == True

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

Lemma 2.

```
In[14]:= Map[or[not[member[x, y]], #] &, SubstTest[implies,
           and[equal[y, w], member[w, z]], member[y, z], w -> union[y, singleton[x]]]]

Out[14]= or[member[y, z], not[member[x, y]], not[member[union[y, singleton[x]], z]]] == True

In[15]:= (% /. {x -> x_, y -> y_, z -> z_}) /. Equal -> SetDelayed
```

## a lemma similar to CUP−SUB

A variant of Theorem **CUP−SUB** now follows. There are two Skolem functions **m** and **n** in **Otter**'s proof of Lemma **SBV−PS3** that correspond to two additonal free variables **u** and **v**, respectively, in the present derivation.

```
In[16]:= Map[not, SubstTest[and, implies[p2, p3], implies[and[p1, p3, p4], p5],
           implies[and[p2, p5], p6], not[implies[and[p1, p2, p4], p6]], {p1 -> invariant[K, x],
            p2 -> member[u, v], p3 -> member[u, V], p4 -> member[dif[v, singleton[u]], x],
            p5 -> member[union[v, singleton[u]], x], p6 -> member[v, x]}]]

Out[16]= or[member[v, x], not[member[u, v]],
           not[member[intersection[v, complement[singleton[u]]], x]],
           not[subclass[image[K, x], x]]] == True

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

The analog of Lemma **CUP−SUB** derived here can be restated as follows:

```
In[18]:= implies[
          and[member[u, v], member[dif[v, singleton[u]], x], invariant[K, x]], member[v, x]]

Out[18]= True
```

## simulating Otter's proof of SBV−PS3

The only clause that **Otter** used involving **PS** in the proof of lemma **SBV−PS3** is recognized as true by the **GOEDEL** program, and from it one can deduce a useful consequence:

```
In[19]:= SubstTest[not,
          and[member[t, z], member[v, complement[image[w, z]]], member[pair[t, v], w]],
             {w -> PS, t -> dif[v, singleton[u]], z -> dif[P[y], x]}]

Out[19]= or[member[v, image[PS, intersection[complement[x], P[y]]]],
           member[intersection[v, complement[singleton[u]]], x], not[member[u, v]],
           not[member[v, V]], not[subclass[v, union[y, singleton[u]]]]] == True

In[20]:= (% /. {u -> u_, v -> v_, x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Restatement:

```
In[21]:= implies[and[member[u, v], member[v, V], member[dif[v, singleton[u]], dif[P[y], x]]],
          member[v, image[PS, dif[P[y], x]]]]

Out[21]= True
```

Most of the remaining clauses in **Otter**'s proof of lemma **SBV−PS3** amount to the following argument:

```
In[22]:= Map[not, SubstTest[and, implies[hyp, p1], implies[hyp, p2], implies[hyp, p3],
         implies[hyp, p4], implies[and[hyp, p1, p2, p3, p4], p5],
         not[implies[hyp, p5]],
           {hyp -> and[invariant[K, x], member[u, v], member[v, dif[P[y], x]]],
         p1 -> member[v, V], p2 -> member[dif[v, singleton[u]], V],
         p3 -> subclass[dif[v, singleton[u]], y],
            p4 -> not[member[dif[v, singleton[u]], x]],
         p5 -> member[v, image[PS, dif[P[y], x]]]}]]

Out[22]= or[member[v, x], member[v, image[PS, intersection[complement[x], P[y]]]],
           not[member[u, v]], not[member[v, V]],
           not[subclass[v, y]], not[subclass[image[K, x], x]]] == True

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

The best strategy now seems to be to get rid of both variables **u, v** corresponding to **Otter**'s Skolem functions at the same time:

```
In[24]:= Map[equal[0, composite[Id, complement[#]]] &, SubstTest[class, pair[u, v],
            or[member[v, w], not[member[u, v]], not[member[v, V]], not[subclass[z, x]]],
             {w -> union[x, complement[P[y]], image[PS, intersection[complement[x], P[y]]]],
             z -> image[K, x]}]] // Reverse

Out[24]= or[not[subclass[image[K, x], x]], subclass[P[y],
            union[x, image[PS, intersection[complement[x], P[y]]], singleton[0]]]] == True

In[25]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

This can be restated as follows:

```
In[26]:= implies[invariant[K, x],
          subclass[dif[P[y], union[x, singleton[0]]], image[PS, dif[P[y], x]]]]

Out[26]= True
```

What one really wants is a subvariance statement. This is easily derived with a few additional steps.

```
In[27]:= SubstTest[implies, and[equal[u, v], subclass[v, w]], subclass[u, w],
             {u -> dif[P[y], x],
           v -> dif[P[y], union[x, singleton[0]]], w -> image[PS, dif[P[y], x]]}]

Out[27]= or[not[member[0, x]], not[subclass[P[y],
             union[x, image[PS, intersection[complement[x], P[y]]], singleton[0]]],
           subclass[P[y], union[x, image[PS, intersection[complement[x], P[y]]]]]] == True

In[28]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[29]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 -> invariant[K, x], p2 -> member[0, x],
         p3 -> subclass[dif[P[y], union[x, singleton[0]]], image[PS, dif[P[y], x]]],
         p4 -> subvariant[PS, dif[P[y], x]]}]]

Out[29]= or[not[member[0, x]], not[subclass[image[K, x], x]],
           subclass[P[y], union[x, image[PS, intersection[complement[x], P[y]]]]]] == True

In[30]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

The next lemma corresponds to clause 4017 in **Otter**'s proof.

```
In[31]:= SubstTest[implies, and[subclass[u, v], member[v, V]],
          member[u, V], {u -> intersection[P[y], x], v -> P[y]}]

Out[31]= or[member[intersection[x, P[y]], V], not[member[y, V]]] == True

In[32]:= or[member[intersection[x_, P[y_]], V], not[member[y_, V]]] := True
```

The following is analogous to **Otter**'s lemma **SBV−PS3**, but restated in terms of **K**−invariance.

```
In[33]:= implies[and[member[0, x], member[y, V], invariant[K, x]],
             member[dif[P[y], x], subvar[PS]]] // not // not

Out[33]= True
```

## FINITE induction

All that remains to derive the theorem on **FINITE** induction is to eliminate the set variable **y**. The elimination of this variable requires the following class:

```
In[34]:= class[y, member[dif[P[y], x], z]] /. z -> subvar[PS]

Out[34]= image[inverse[POWER], image[inverse[IMAGE[id[complement[x]]]], subvar[PS]]]

In[35]:= or[and[member[intersection[complement[x], P[y]], V],
              subclass[P[y], union[x, image[PS, intersection[complement[x], P[y]]]]]],
            not[member[0, x]], not[member[y, V]], not[invariant[K, x]]] // NotNotTest

Out[35]= or[and[member[intersection[complement[x], P[y]], V],
              subclass[P[y], union[x, image[PS, intersection[complement[x], P[y]]]]]],
            not[member[0, x]], not[member[y, V]], not[subclass[image[K, x], x]]] == True

In[36]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

The variable **y** is now eliminated:

```
In[37]:= Map[equal[V, #] &,
            SubstTest[class, y, implies[and[member[0, x], member[y, V], subclass[w, x]],
              member[y, z]],
              {w -> image[K, x], z -> image[inverse[POWER],
                image[inverse[IMAGE[id[complement[x]]]], subvar[PS]]]}]] // Reverse

Out[37]= or[not[member[0, x]], not[subclass[image[K, x], x]],
            subclass[image[IMAGE[id[complement[x]]], range[POWER]], subvar[PS]]] == True

In[38]:= (% /. x -> x_) /. Equal -> SetDelayed
```

To obtain a connection with **FINITE** one needs to apply **U** to both sides of the positive **subclass** literal in the above statement.

```
In[39]:= SubstTest[implies, subclass[u, v], subclass[U[u], U[v]],
           {u -> image[IMAGE[id[complement[x]]], range[POWER]], v -> subvar[PS]}]

Out[39]= or[not[subclass[image[IMAGE[id[complement[x]]], range[POWER]], subvar[PS]]],
            subclass[FINITE, x]] == True

In[40]:= or[not[subclass[image[IMAGE[id[complement[x_]]], range[POWER]], subvar[PS]]],
            subclass[FINITE, x_]] := True
```

This is the theorem on **FINITE** induction:

```
In[41]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
              {p1 -> and[member[0, x], invariant[K, x]],
            p2 -> subclass[image[IMAGE[id[complement[x]]], range[POWER]], subvar[PS]],
            p3 -> subclass[FINITE, x]}]]

Out[41]= or[not[member[0, x]], not[subclass[image[K, x], x]], subclass[FINITE, x]] == True

In[42]:= or[not[member[0, x_]], not[subclass[image[K, x_], x_]], subclass[FINITE, x_]] := True
```