full and well-founded induction

Johan G. F. Belinfante
2010 March 10

```
In[1]:= SetDirectory["l:"]; << goedel.10mar08a;<< tools.m

        :Package Title: goedel.10mar08a               2010 March 8 at 4:50 p.m.

        It is now:  2010 Mar 10 at 10:49

        Loading Simplification Rules

        TOOLS.M                                    Revised 2010 February 26

        weightlimit = 40
```

summary

Rewrite rules are derived for well-founded induction that closely resemble this standard version of the second form of induction:

```
In[2]:= implies[subclass[intersection[omega, P[x]], x], subclass[omega, x]]

Out[2]= True
```

In particular, it is shown that **omega** can be replaced with any full, regular set. (If the axiom of regularity is assumed, then any set is regular.)

derivation

If a relation **x** is well-founded and if its inverse is thin, then the empty set is the only class that is subvariant under **x**. To obtain a conclusion similar to that in the second form of induction, one need only apply this to the class **domain[x] ~ y**.

Lemma. A basic form of well-founded induction resembling the second form of induction.

```
In[3]:= SubstTest[implies, and[WELLFOUNDED[x], thin[inverse[x]], subclass[t, image[x, t]]],
          equal[0, t], t → dif[domain[x], y]] // Reverse

Out[3]= or[not[equal[V, domain[VERTSECT[inverse[x]]]]],
          not[subclass[domain[x], union[y, image[x, complement[y]]]]],
          not[WELLFOUNDED[x]], subclass[domain[x], y]] == True

In[4]:= or[not[equal[V, domain[VERTSECT[inverse[x_]]]]],
          not[subclass[domain[x_], union[image[x_, complement[y_]], y_]]],
          not[WELLFOUNDED[x_]], subclass[domain[x_], y_]] := True
```

Theorem. A specialization of well-founded induction to restrictions of the membership relation.

```
In[5]:= SubstTest[implies, and[WELLFOUNDED[t], thin[inverse[t]],
          subclass[domain[t], union[image[t, complement[y]], y]]],
        subclass[domain[t], y], t → composite[id[x], E]] // Reverse

Out[5]= or[not[subclass[intersection[P[y], U[x]], y]], not[subclass[U[x], union[x, y]]],
        not[WELLFOUNDED[composite[id[x], E]]], subclass[U[x], y]] == True

In[6]:= or[not[subclass[intersection[P[y_], U[x_]], y_]], not[subclass[U[x_], union[x_, y_]]],
        not[WELLFOUNDED[composite[id[x_], E]]], subclass[U[x_], y_]] := True
```

The hypothesis $\mathbf{U[x] \subset x \bigcup y}$ is certainly satsified when $\mathbf{x}$ is full, that is, if $\mathbf{U[x] \subset x}$. Here the wrapper **tc[x]** is used for a generic full class.

Corollary. Full induction, using a **tc** wrapper.

```
In[7]:= SubstTest[implies,
        and[subclass[intersection[P[y], U[t]], y], subclass[U[t], union[t, y]],
         WELLFOUNDED[composite[id[t], E]]], subclass[U[t], y], t → tc[x]] // Reverse

Out[7]= or[not[subclass[x, REGULAR]],
        not[subclass[intersection[P[y], U[tc[x]]], y]], subclass[U[tc[x]], y]] == True

In[8]:= or[not[subclass[x_, REGULAR]],
        not[subclass[intersection[P[y_], U[tc[x_]]], y_]], subclass[U[tc[x_]], y_]] := True
```

To eliminate the sum class constructor $\mathbf{U}$, the preceding result is specialized to a successor set.

Corollary. A form of full induction without the sum class constructor.

```
In[9]:= SubstTest[or, not[subclass[t, REGULAR]], not[subclass[intersection[P[y], U[tc[t]]], y]],
        subclass[U[tc[t]], y], t → succ[setpart[x]]] // Reverse

Out[9]= or[not[member[setpart[x], REGULAR]],
        not[subclass[intersection[P[y], tc[setpart[x]]], y]],
        subclass[tc[setpart[x]], y]] == True

In[10]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The double wrapper **tc[setpart[x]]** will now be eliminated, one step at a time. First, the **setpart** wrapper is eliminated.

Lemma. (Removing the **setpart** wrapper.)

```
In[11]:= SubstTest[implies, and[equal[x, setpart[t]], member[x, REGULAR],
           subclass[intersection[tc[x], P[y]], y]], subclass[tc[x], y], t → x] // Reverse

Out[11]= or[not[member[x, REGULAR]],
          not[subclass[intersection[P[y], tc[x]], y]], subclass[tc[x], y]] == True

In[12]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

The **tc** wrapper is removed in the following theorem.

Theorem. A generalized second form of induction for any full, regular set **x**.

```
In[13]:= SubstTest[implies, and[equal[x, tc[t]], member[x, REGULAR],
           subclass[intersection[x, P[y]], y]], subclass[x, y], t → x] // Reverse

Out[13]= or[not[member[x, REGULAR]], not[subclass[intersection[x, P[y]], y]],
           not[subclass[U[x], x]], subclass[x, y]] == True

In[14]:= or[not[member[x_, REGULAR]], not[subclass[intersection[x_, P[y_]], y_]],
           not[subclass[U[x_], x_]], subclass[x_, y_]] := True
```

The term **full induction** will be used in the rest of this notebook to refer to this generalization of the second form of induction.

examples

Two simple examples of full induction are given in this section. The first special case involves the power class of the set of natural numbers.

Theorem. Full induction for the case of **P[omega]**.

```
In[15]:= SubstTest[implies, and[member[t, REGULAR], subclass[intersection[t, P[x]], x],
           subclass[U[t], t]], subclass[t, x], t → P[omega]] // Reverse

Out[15]= or[not[subclass[P[intersection[omega, x]], x]], subclass[P[omega], x]] == True

In[16]:= or[not[subclass[P[intersection[omega, x_]], x_]], subclass[P[omega], x_]] := True
```

A slight variant of this is obtained by restricting attention to finite subsets of the set **omega** of natural numbers.

```
In[17]:= intersection[FINITE, P[omega]]

Out[17]= image[inverse[S], omega]
```

Theorem. A form of full induction for the case of **FINITE ⋂ P[omega]**.

```
In[18]:= SubstTest[implies, and[member[t, REGULAR], subclass[intersection[t, P[x]], x],
           subclass[U[t], t]], subclass[t, x], t → image[inverse[S], omega]] // Reverse

Out[18]= or[not[subclass[intersection[image[inverse[S], omega], P[x]], x]],
           subclass[image[inverse[S], omega], x]] == True

In[19]:= or[not[subclass[intersection[image[inverse[S], omega], P[x_]], x_]],
           subclass[image[inverse[S], omega], x_]] := True
```

eliminating variables

When the variable **y** in the statement of full induction is eliminated, one obtains a statement involving the compound constructor **allclosed[id[x]]**. The membership rule for this expression is as follows:

```
In[32]:= member[y, allclosed[id[x]]]

Out[32]= and[member[y, V], subclass[intersection[x, P[y]], y]]
```

Lemma. Eliminating the variable **y**.

```
In[23]:= Map[equal[V, #] &, SubstTest[class, t,
           implies[and[member[x, s], subclass[intersection[x, P[t]], t]], subclass[x, t]],
           s → intersection[FULL, REGULAR]]]

Out[23]= or[not[member[x, REGULAR]],
           not[subclass[U[x], x]], subclass[x, A[allclosed[id[x]]]]] == True

In[24]:= (% /. x → x_) /. Equal → SetDelayed
```

The inclusion in the conclusion of the above lemma can be sharpened to an equation.

Theorem. A form of full induction with only one variable.

```
In[30]:= SubstTest[and, implies[p, subclass[x, t]], implies[p, subclass[t, x]],
          {p → and[member[x, REGULAR], full[x]], t → A[allclosed[id[x]]]}]

Out[30]= or[equal[x, A[allclosed[id[x]]]],
           not[member[x, REGULAR]], not[subclass[U[x], x]]] == True

In[31]:= or[equal[x_, A[allclosed[id[x_]]]],
           not[member[x_, REGULAR]], not[subclass[U[x_], x_]]] := True
```

It is also possible to eliminate both variables, obtaining a variable-free statement of full induction. The resulting statement involves the relation **UB[union[E, complement[S]]]**. The membership rule for this relation is:

```
In[33]:= member[pair[x, y], UB[union[E, complement[S]]]]

Out[33]= and[member[x, V], member[y, V], subclass[intersection[x, P[y]], setpart[y]]]
```

Theorem. A variable-free statement of full induction.

```
In[35]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[x, y],
           implies[and[member[x, s], subclass[intersection[x, P[y]], y]], subclass[x, y]],
           s → intersection[FULL, REGULAR]]]

Out[35]= subclass[composite[UB[union[E, complement[S]]], id[intersection[FULL, REGULAR]]], S] ==
          True

In[36]:= subclass[
           composite[UB[union[E, complement[S]]], id[intersection[FULL, REGULAR]]], S] := True
```

Comment. The following specialization of this is already available in the **GOEDEL** program.

```
In[37]:= subclass[composite[UB[union[E, complement[S]]], id[OMEGA]], S]

Out[37]= True
```