generalizing the second form of induction

Johan G. F. Belinfante and William Drobny
2008 August 9

```
In[1]:= SetDirectory["l:"]; << goedel.08aug08a;<< tools.m

        :Package Title: goedel.08aug08a                 2008 August 8 at 4:45 p.m.

        It is now:  2008 Aug 9 at 17:7

        Loading Simplification Rules

        TOOLS.M                                       Revised 2008 July 5

        weightlimit = 40
```

summary

In arithmetic two forms of induction are encountered. The first form of induction begins with **0** as base case, and then proceeds from each natural number to its successor.

```
In[2]:= implies[and[member[0, x], invariant[SUCC, x]], subclass[omega, x]]

Out[2]= True
```

The second form of induction goes to each natural number from the set of all its predecessors. Both forms of induction are logically equivalent, but the statement of the second form of induction is in one respect simpler than the statement for the first form of induction in that only two literals are needed instead of three. In the second form of induction one no longer needs an explicit literal for the base case:

```
In[3]:= implies[subclass[intersection[omega, P[x]], x], subclass[omega, x]]

Out[3]= True
```

The second form of induction has many generalizations. For example, the set **omega** can be replaced by any ordinal number, and it looks much the same.

```
In[4]:= implies[subclass[intersection[ord[x], P[y]], y], subclass[ord[x], y]]

Out[4]= True
```

The second form of induction also works for the class **OMEGA** of all ordinal numbers. (Comment: The class **OMEGA** is not a set, so the following statement of this generalization requires a theory that admits proper classes.)

```
In[5]:= implies[subclass[intersection[OMEGA, P[x]], x], subclass[OMEGA, x]]

Out[5]= True
```

A more grandiose form of induction is epsilon induction, in which the proper class **OMEGA** is relaced by a still larger proper class **REGULAR**. This form of induction looks just the same otherwise:

```
In[6]:=  implies[subclass[intersection[REGULAR, P[x]], x], subclass[REGULAR, x]]

Out[6]=  True
```

If the axiom of regularity holds, then the class **REGULAR** is equal to the univeral class **V** of all sets, and the statement can be simplified as follows:

```
In[7]:=  implies[and[AxReg, subclass[P[x], x]], equal[V, x]]

Out[7]=  True
```

Comment. Cantor's theorem implies that the power class **P[x]** cannot be a subclass of **x** when **x** is a set. In the statement of epsilon induction, the variable **x** refers to a proper class, not a set. One can obtain a version of this theorem that avoids proper classes by replacing the class **x** with its complement, which could be a set. If only the case that this complement is a set is considered, one can then quantify over this set variable, eliminating variables altogether. The variable-free statement one obtains is precisely the axiom of regularity:

```
In[8]:=  assert[forall[x, implies[subclass[P[complement[x]], complement[x]], empty[x]]]]

Out[8]=  AxReg
```

In this notebook the connection between the second form of induction and yet another form of induction, called well-founded induction, is explored. From a certain point of view, well-founded induction could be considered trivial as it just amounts to a restatement of the definition of well-foundedness, but its connection with the second form of induction is less obvious. Two forms of this connection are considered in this notebook, one with a sethood literal, and another form in which the sethood literal is replaced with a thin-ness literal. A relation is called **thin** if all its vertical sections are sets. Clearly, functions are thin because their vertical sections are either empty or singletons. The so-called axiom of sum sets implies that the inverse of the membership relation **E** is thin, and the axiom of power sets implies that the inverse of the subset relation **S** is thin, and the same goes for the inverse of the proper subset relation **PS**.

two forms of well-founded induction

Theorem. (A form of well-founded induction with a membership literal.)

```
In[9]:=  Map[implies[member[w, x], #] &,
           SubstTest[implies, and[member[t, u], equal[u, v]], member[t, v],
            {t → w, u → intersection[P[y], subvar[z]], v → set[0]}]] // Reverse

Out[9]=  or[equal[0, w], not[member[w, x]], not[subclass[w, y]],
           not[subclass[w, image[z, w]]], not[WELLFOUNDED[composite[id[y], z]]]] == True

In[10]:=  or[equal[0, w_], not[member[w_, x_]], not[subclass[w_, y_]],
            not[subclass[w_, image[z_, w_]]], not[WELLFOUNDED[composite[id[y_], z_]]]] := True
```

To see that the second form of induction for natural numbers is a special case of this, one needs to make the following replacements in the above statement of well-founded induction:

```
In[11]:= {equal[0, w], member[w, x], subclass[w, y], subclass[w, image[z, w]],
           WELLFOUNDED[composite[id[y], z]]} /. {w → dif[omega, t], x → V, y → omega, z → E}

Out[11]= {subclass[omega, t], True, True, subclass[intersection[omega, P[t]], t], True}
```

Lemma. (A simplification rule.)

```
In[12]:= AssInt[ord[x], P[ord[x]], P[y]]

Out[12]= intersection[ord[x], P[intersection[y, ord[x]]]] == intersection[ord[x], P[y]]

In[13]:= intersection[ord[x_], P[intersection[y_, ord[x_]]]] := intersection[ord[x], P[y]]
```

Lemma. (Another simplification rule.)

```
In[14]:= AssInt[ord[x], P[ord[x]], P[union[y, complement[ord[x]]]]] // Reverse

Out[14]= intersection[ord[x], P[union[y, complement[ord[x]]]]] == intersection[ord[x], P[y]]

In[15]:= intersection[ord[x_], P[union[y_, complement[ord[x_]]]]] := intersection[ord[x], P[y]]
```

With these two lemmas in place, it is now easy to see that well-founded induction implies the more general ordinal number form of second induction:

```
In[16]:= {equal[0, w], member[w, x], subclass[w, y], subclass[w, image[z, w]],
           WELLFOUNDED[composite[id[y], z]]} /. {w → dif[ord[s], t], x → V, y → ord[s], z → E}

Out[16]= {subclass[ord[s], t], True, True, subclass[intersection[ord[s], P[t]], t], True}
```

Theorem. The sethood literal in the statement of well-founded induction can be replaced with a suitable thin-ness condition.

```
In[17]:= Map[not, SubstTest[and, implies[and[p1, p3], p4],
            implies[and[p0, p2, p4], p5], not[implies[and[p0, p1, p2, p3], p5]],
            {p0 → subclass[x, domain[VERTSECT[inverse[z]]]], p1 -> subclass[x, y],
             p2 -> subclass[x, image[z, x]], p3 -> WELLFOUNDED[composite[id[y], z]],
             p4 -> WELLFOUNDED[composite[id[x], z]], p5 -> equal[0, x]}]] // Reverse

Out[17]= or[equal[0, x], not[subclass[x, y]], not[subclass[x, domain[VERTSECT[inverse[z]]]]],
           not[subclass[x, image[z, x]]], not[WELLFOUNDED[composite[id[y], z]]]] == True

In[18]:= or[equal[0, x_], not[subclass[x_, y_]],
           not[subclass[x_, domain[VERTSECT[inverse[z_]]]]],
           not[subclass[x_, image[z_, x_]]], not[WELLFOUNDED[composite[id[y_], z_]]]] := True
```

The principle of epsilon induction is a special case of this:

```
In[19]:= {equal[0, x], subclass[x, y], subclass[x, domain[VERTSECT[inverse[z]]]],
           subclass[x, image[z, x]], WELLFOUNDED[composite[id[y], z]]} /.
          {x → dif[REGULAR, t], y → REGULAR, z → E}

Out[19]= {subclass[REGULAR, t], True, True, subclass[P[intersection[REGULAR, t]], t], True}
```

Comment. The following rewrite rule kicked in here:

```
In[20]:= intersection[REGULAR, P[t]]

Out[20]= P[intersection[REGULAR, t]]
```