epsilon induction

Johan G. F. Belinfante
2004 December 29

```
In[1]:= SetDirectory["i:"]; << goedel64.28a; << tools.m

        :Package Title: goedel64.28a              2004 December 28 at 8:50 a.m.

        It is now:  2004 Dec 30 at 10:11

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 December 21

        weightlimit = 40
```

epsilon induction

In this notebook, the proof of theorem **TC-REG-2** obtained by **Otter** on 2001 March 24 is rederived. This theorem is similar to what Thomas Jech calls proof by epsilon induction. See Theorem 25(A) on page 73 of the following reference:

```
In[2]:= "Thomas Jech, Set Theory, Academic Press,San Diego (1978).";
```

The notation and terminology differs slightly. What is here called the full class **x** he calls a transitive class, which he denotes by **T**. Jech talks about a property **Φ** instead of the class **y** of sets that satisfy some property. Jech assumes that the axiom of regularity holds, in which case **REGULAR = V**, so one can omit the intersection with the class **REGULAR** that appears in theorem **TC-REG-2**. Jech adds an unnecessary hypothesis that the empty set belongs to the class **y**. When the axiom of regularity holds, any non-empty full class **x** holds the empty set. If **x** is not empty, the condition **subclass[intersection[x,P[y]],y]** implies that the empty set belongs to **y**. When **x** is empty, the theorem holds trivially.

main argument

Lemma.

```
In[3]:= SubstTest[implies, and[member[z, w], subclass[w, y]],
         member[z, y], w → intersection[x, P[y]]]

Out[3]= or[member[z, y], not[member[z, x]], not[subclass[z, y]],
          not[subclass[intersection[x, P[y]], y]]] == True

In[4]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Main part of **Otter**'s argument.

```
In[5]:= Map[not, SubstTest[and, implies[and[p1, p4], p5], implies[and[p3, p5], p6],
          implies[and[p2, p4, p6], p7], not[implies[and[p1, p2, p3, p4], p7]],
          {p1 → full[x], p2 → subclass[intersection[x, P[y]], y],
           p3 → subclass[intersection[x, z], y], p4 → member[z, x],
           p5 → subclass[z, x], p6 → subclass[z, y], p7 → member[z, y]}]]

Out[5]= or[member[z, y], not[member[z, x]], not[subclass[intersection[x, z], y]],
          not[subclass[intersection[x, P[y]], y]], not[subclass[U[x], x]]] == True

In[6]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Elimination of the variable **z**.

```
In[7]:= Map[equal[V, #] &, SubstTest[class, z, or[member[z, y], not[member[z, x]],
            not[subclass[intersection[x, z], y]], not[subclass[w, y]],
            not[subclass[U[x], x]]], w → intersection[x, P[y]]]] // Reverse

Out[7]= or[not[subclass[intersection[x, P[y]], y]], not[subclass[U[x], x]],
          subclass[intersection[x, P[union[y, complement[x]]]], y]] == True

In[8]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

---

the rest of the story

The following special case is available, and will be used to prove the general case.

```
In[9]:= implies[subclass[P[x], x], subclass[REGULAR, x]]

Out[9]= True
```

A straightfoward substitution yields a result with the desired conclusion, but a different hypothesis.

```
In[10]:= SubstTest[implies, subclass[P[z], z],
          subclass[REGULAR, z], z → union[y, complement[x]]]

Out[10]= or[not[subclass[intersection[x, P[union[y, complement[x]]]], y]],
           subclass[intersection[REGULAR, x], y]] == True

In[11]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

This result is combined with that of the preceding section.

```
In[12]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[p3, p4], not[implies[and[p1, p2], p4]],
           {p1 → full[x], p2 -> subclass[intersection[x, P[y]], y],
            p3 -> subclass[intersection[x, P[union[y, complement[x]]]], y],
            p4 -> subclass[intersection[REGULAR, x], y]}]]

Out[12]= or[not[subclass[intersection[x, P[y]], y]],
           not[subclass[U[x], x]], subclass[intersection[REGULAR, x], y]] == True

In[13]:= or[not[subclass[intersection[x_, P[y_]], y_]],
           not[subclass[U[x_], x_]], subclass[intersection[REGULAR, x_], y_]] := True
```