wf ⇒ no infinite descent

Johan G. F. Belinfante
2008 August 19

```
In[1]:= SetDirectory["l:"]; << goedel.08aug18a;<< tools.m

        :Package Title: goedel.08aug18a                2008 August 18 at 11:45 p.m.

        It is now:  2008 Aug 19 at 21:58

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 July 5

        weightlimit = 40
```

summary

If a relation $<$ is well-founded, there can be no infinitely descending chain: $\cdots < y_3 < y_2 < y_1 < y_0$.

the successor relation on natural numbers

The successor relation on natural numbers is well-founded:

```
In[2]:= WELLFOUNDED[composite[id[omega], SUCC]]

Out[2]= True
```

Lemma.

```
In[3]:= Map[not, SubstTest[implies, and[member[w, x], subclass[x, y]], member[w, y],
           {w → omega, x -> image[RC[omega], omega], y -> set[0]}]] // Reverse

Out[3]= subclass[image[RC[omega], omega], set[0]] == False

In[4]:= % /. Equal→ SetDelayed
```

Theorem. The inverse of the successor relation on natural numbers is not well-founded.

```
In[5]:= WELLFOUNDED[composite[inverse[SUCC], id[omega]]] // AssertTest

Out[5]= WELLFOUNDED[composite[inverse[SUCC], id[omega]]] == False

In[6]:= WELLFOUNDED[composite[inverse[SUCC], id[omega]]] := False
```

This makes sense intuitively: for the natural numbers, one can go up forever, but one cannot go down forever.

no infinite descent

Lemma.

```
In[7]:= Map[implies[#, member[pair[APPLY[funpart[t], m], APPLY[funpart[t], n]], x]] &,
         member[pair[m, n], composite[inverse[funpart[t]], x, funpart[t]]] // AssertTest]

Out[7]= or[member[pair[APPLY[funpart[t], m], APPLY[funpart[t], n]], x],
          not[member[pair[m, n], composite[inverse[funpart[t]], x, funpart[t]]]]] == True

In[8]:= (% /. {m → m_, n → n_, t → t_, x → x_}) /. Equal → SetDelayed
```

An infinite list can be regarded as a function **funpart[t]** whose domain is the set **omega** of natural numbers. The condition that this list be infinitely descending can be formulated as follows:

```
In[9]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
          {u → pair[succ[nat[n]], nat[n]], v -> composite[inverse[SUCC], id[omega]],
           w -> composite[inverse[funpart[t]], x, funpart[t]]}] // Reverse

Out[9]= or[member[pair[succ[nat[n]], nat[n]], composite[inverse[funpart[t]], x, funpart[t]]],
          not[subclass[composite[inverse[SUCC], id[omega]],
            composite[inverse[funpart[t]], x, funpart[t]]]]] == True

In[10]:= (% /. {n → n_, t → t_, x → x_}) /. Equal → SetDelayed
```

Theorem.

```
In[11]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
            not[implies[p1, p3]], {p1 -> subclass[composite[inverse[SUCC], id[omega]],
               composite[inverse[funpart[t]], x, funpart[t]]], p2 -> member[
               pair[succ[nat[n]], nat[n]], composite[inverse[funpart[t]], x, funpart[t]]],
             p3 -> member[pair[APPLY[funpart[t], succ[nat[n]]],
                APPLY[funpart[t], nat[n]]], x]}]] // Reverse

Out[11]= or[member[pair[APPLY[funpart[t], succ[nat[n]]], APPLY[funpart[t], nat[n]]], x],
           not[subclass[composite[inverse[SUCC], id[omega]],
             composite[inverse[funpart[t]], x, funpart[t]]]]] == True

In[12]:= or[member[pair[APPLY[funpart[t_], succ[nat[n_]]], APPLY[funpart[t_], nat[n_]]], x_],
           not[subclass[composite[inverse[SUCC], id[omega]],
             composite[inverse[funpart[t_]], x_, funpart[t_]]]]] := True
```

Theorem. (Wellfounded ⇒ no infinite descent.)

```
In[13]:= Map[not, SubstTest[implies, and[subclass[u, v], WELLFOUNDED[v]], WELLFOUNDED[u],
            {u -> composite[inverse[SUCC], id[omega]],
             v → composite[inverse[funpart[t]], wf[x], funpart[t]]}]] // Reverse

Out[13]= subclass[composite[inverse[SUCC], id[omega]],
           composite[inverse[funpart[t]], wf[x], funpart[t]]] == False
```

```
In[14]:= subclass[composite[inverse[SUCC], id[omega]],
           composite[inverse[funpart[t_]], wf[x_], funpart[t_]]] := False
```

Corollary. (Restatement without wrappers.)

```
In[15]:= SubstTest[implies, and[equal[x, funpart[t]], equal[y, wf[u]]],
           not[subclass[composite[inverse[SUCC], id[omega]], composite[inverse[x], y, x]]],
           {t → x, u → y}] // Reverse

Out[15]= or[not[FUNCTION[x]],
           not[subclass[composite[inverse[SUCC], id[omega]], composite[inverse[x], y, x]]],
           not[WELLFOUNDED[y]]] == True

In[16]:= or[not[FUNCTION[x_]],
           not[subclass[composite[inverse[SUCC], id[omega]], composite[inverse[x_], y_, x_]]],
           not[WELLFOUNDED[y_]]] := True
```

Contrapositive form:

```
In[17]:= and[FUNCTION[x], subclass[composite[inverse[SUCC], id[omega]],
             composite[inverse[x], y, x]], WELLFOUNDED[y]] // NotNotTest

Out[17]= and[FUNCTION[x], subclass[composite[inverse[SUCC], id[omega]],
             composite[inverse[x], y, x]], WELLFOUNDED[y]] == False

In[18]:= and[FUNCTION[x_], subclass[composite[inverse[SUCC], id[omega]],
             composite[inverse[x_], y_, x_]], WELLFOUNDED[y_]] := False
```