the power tower

Johan G. F. Belinfante and Ming Li
2006 April 14

```
In[1]:= SetDirectory["l:"]; << goedel80.13b; << tools.m

        :Package Title: goedel80.13b          2006 April 13 at 3:05 p.m.

        It is now:  2006 Apr 14 at 16:54

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 March 7

        weightlimit = 40
```

summary

The Smullyan-Fitting double induction theorem is applied to the set of finite levels of the Zermelo-von Neumann cumulative hierarchy.

introduction: the power tower

The finite levels of the Zermelo-von Neumann cumulative hierarchy are the sets **0, P[0], P[P[0]], ...** obtained by iterating the power set functor, starting with the empty set. These are the values of the function **IMAGE[inverse[RANK]]** at the natural numbers:

```
In[2]:= APPLY[IMAGE[inverse[RANK]], 0]

Out[2]= 0

In[3]:= APPLY[IMAGE[inverse[RANK]], set[0]] == P[0]

Out[3]= True

In[4]:= APPLY[IMAGE[inverse[RANK]], succ[set[0]]] == P[P[0]]

Out[4]= True

In[5]:= APPLY[IMAGE[inverse[RANK]], succ[succ[set[0]]] ] == P[P[P[0]]]

Out[5]= True
```

Note that these sets form an infinite chain; each one is a subset of the next one, obtained by iteratively applying the function **POWER**. The set of all these finite levels of the cumulative hierarchy can be described as the orbit of **POWER** starting at **0**, that is, the minimal set which holds the empty set and is invariant under **POWER**.

```
In[6]:= range[iterate[POWER, set[0]]]

Out[6]= image[IMAGE[inverse[RANK]], omega]
```

In this notebook, some rewrite rules related to this collection of sets are derived, and an application is made of the corollary about progressive functions of the double induction theorem presented by Smullyan and Fitting. See the notebooks **dbl-ind1.nb** and **dbl-ind2.nb** for details.

```
In[7]:= "Raymond M. Smullyan and Melvin Fitting, Set Theory and the Continuum Problem,
           Oxford Science Publications, Clarendon Press, Oxford, 1996.  (see page 34)";
```

The function to which this corollary is applied is not **POWER** itself, but its restriction to the class of full sets.

```
In[8]:= intersection[S, POWER]

Out[8]= composite[POWER, id[FULL]]
```

---

lemmas

Lemma.

```
In[9]:= SubstTest[implies, and[invariant[u, w], subclass[v, w]],
          subclass[range[iterate[u, v]], w], {u → POWER, v → set[0], w → FULL}]

Out[9]= subclass[image[IMAGE[inverse[RANK]], omega], FULL] == True

In[10]:= subclass[image[IMAGE[inverse[RANK]], omega], FULL] := True
```

Corollary.

```
In[11]:= equal[intersection[omega, image[inverse[IMAGE[inverse[RANK]]], FULL]], omega]

Out[11]= True

In[12]:= intersection[omega, image[inverse[IMAGE[inverse[RANK]]], FULL]] := omega
```

Lemma.

```
In[13]:= Assoc[IMAGE[inverse[RANK]], composite[id[OMEGA], SUCC], id[omega]]

Out[13]= composite[IMAGE[inverse[RANK]], id[omega], SUCC] ==
          composite[POWER, IMAGE[inverse[RANK]], id[omega]]

In[14]:= composite[IMAGE[inverse[RANK]], id[omega], SUCC] :=
          composite[POWER, IMAGE[inverse[RANK]], id[omega]]
```

An application of the uniqueness of iteration.

```
In[15]:= SubstTest[implies,
          and[equal[composite[u, w], composite[w, SUCC]], equal[image[w, set[0]], v]],
          equal[composite[w, id[omega]], iterate[u, v]],
          {u → composite[POWER, id[FULL]], v → set[0], w → iterate[POWER, set[0]]}]

Out[15]= equal[composite[IMAGE[inverse[RANK]], id[omega]],
           iterate[composite[POWER, id[FULL]], set[0]]] == True

In[16]:= iterate[composite[POWER, id[FULL]], set[0]] :=
          composite[IMAGE[inverse[RANK]], id[omega]]
```

The range of iterate is a minimal invariant class.

```
In[17]:= SubstTest[range, iterate[funpart[x], set[0]], x → composite[POWER, id[FULL]]] // Reverse

Out[17]= hull[invar[composite[POWER, id[FULL]]], set[0]] == image[IMAGE[inverse[RANK]], omega]

In[18]:= hull[invar[composite[POWER, id[FULL]]], set[0]] := image[IMAGE[inverse[RANK]], omega]
```

total ordering of the levels of the power tower

Application of a dichotomy corollary of double-induction to the progressing function **intersection[S, POWER]**.

```
In[19]:= SubstTest[subclass, cart[hull[invar[funpart[intersection[S, x]]], set[0]],
           hull[invar[funpart[intersection[S, x]]], set[0]]],
          union[inverse[S], composite[S, funpart[intersection[S, x]]]], x → POWER]

Out[19]= subclass[cart[image[IMAGE[inverse[RANK]], omega], image[IMAGE[inverse[RANK]], omega]],
           union[composite[S, POWER, id[FULL]], inverse[S]]] == True

In[20]:= subclass[cart[image[IMAGE[inverse[RANK]], omega], image[IMAGE[inverse[RANK]], omega]],
           union[composite[S, POWER, id[FULL]], inverse[S]]] := True
```

Lemma.

```
In[21]:= SubstTest[subclass, intersection[x, y], y, {x → POWER, y → S}]

Out[21]= subclass[composite[POWER, id[FULL]], S] == True

In[22]:= subclass[composite[POWER, id[FULL]], S] := True
```

Corollary.

```
In[23]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> cart[image[IMAGE[inverse[RANK]], omega], image[IMAGE[inverse[RANK]], omega]],
           v -> union[composite[S, POWER, id[FULL]], inverse[S]], w -> union[S, inverse[S]]}]

Out[23]= subclass[cart[image[IMAGE[inverse[RANK]], omega], image[IMAGE[inverse[RANK]], omega]],
           union[S, inverse[S]]] == True
```

```
In[24]:= subclass[cart[image[IMAGE[inverse[RANK]], omega], image[IMAGE[inverse[RANK]], omega]],
           union[S, inverse[S]]] := True
```

Restatement: The levels of the power tower are totally ordered by inclusion.

```
In[25]:= TOTALORDER[
          restrict[S, image[IMAGE[inverse[RANK]], omega], image[IMAGE[inverse[RANK]], omega]]]

Out[25]= True
```