subclass[P[x], FINITE]

Johan G. F. Belinfante
2006 March 19

```
In[1]:= SetDirectory["l:"]; << goedel79.18a; << tools.m

        :Package Title: goedel79.18a           2006 March 18 at 9:50 p.m.

        It is now:  2006 Mar 20 at 3:43

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 March 7

        weightlimit = 40
```

summary

Must every proper class contain an infinite subset? In this notebook it is shown that a subclass of **REGULAR** contains only finite subsets if and only if it is a finite set.

acknowledgment

The class of all subsets of a class **x** is its power class **P[x]**. A class **x** has only finite subsets if **subclass[P[x], FINITE]**. Any set belongs to its own power class, so for sets this condition is equivalent to **member[x, FINITE]**. The question whether these statements are also equivalent for proper classes was posted by the author on the Foundations of Mathematics (FOM) e-mail list on 2006 March 2. In response to this inquiry, Harvey Friedman sketched a proof that the statements are equivalent if one assumes the axiom of regularity, and that these statements need not be equivalent more generally. In the **GOEDEL** program, the axiom of regularity is not automatically assumed to hold. The class **V** of all sets does however contain a subclass **REGULAR** in which the axiom of regularity holds. The class **REGULAR** is a proper class which is its own power class and its own sum class. In this notebook, the proof sketched by Professor Friedman is adapted to show that for any subclass **x** of **REGULAR**, the statements **subclass[P[x], FINITE]** and **member[x, FINITE]** are equivelent.

derivation

The general strategy is to use the **RANK** function to transfer the question from subclasses of **REGULAR** to subclasses of the class **OMEGA** of ordinal numbers. This function has domain **REGULAR** and range **OMEGA**. Any function preserves sethood and finiteness. The function **REGULAR** is not one-to-one, but it does possess the important property that the inverse image of a set is a set: in the language of the **GOEDEL** program, this is expressed succinctly as follows.

```
In[2]:= thin[inverse[RANK]]

Out[2]= True
```

Lemma 1. The **setpart** wrapper can be used produce a generic set. The **RANK** function has the property that the inverse image of **setpart[y]** is a set, and the intersection of this with the given class **x** is therefore a subset of **x**.

```
In[3]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
          {u -> intersection[x, image[inverse[RANK], setpart[y]]], v -> P[x], w → FINITE}]

Out[3]= or[member[intersection[x, image[inverse[RANK], setpart[y]]], FINITE],
           not[subclass[P[x], FINITE]]] == True

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

Lemma 2. The image of a finite set under any function is a finite set, so this is true in particular for the **RANK** function.

```
In[5]:= SubstTest[implies, and[FUNCTION[z], member[x, FINITE]],
          member[image[z, x], FINITE], z → RANK]

Out[5]= or[member[image[RANK, x], FINITE], not[member[x, FINITE]]] == True

In[6]:= or[member[image[RANK, x_], FINITE], not[member[x_, FINITE]]] := True
```

Lemma 3. Applying Lemma 2 to the finite set that appears in Lemma 1, one finds:

```
In[7]:= SubstTest[implies, member[z, FINITE], member[image[RANK, z], FINITE],
          z -> intersection[x, image[inverse[RANK], setpart[y]]]]

Out[7]= or[member[intersection[image[RANK, x], setpart[y]], FINITE],
           not[member[intersection[x, image[inverse[RANK], setpart[y]]], FINITE]]] == True

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

Lemma 4. Combining Lemma 1 and Lemma 3 yields:

```
In[9]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], not[implies[p1, p3]], {p1 → subclass[P[x], FINITE],
            p2 -> member[intersection[x, image[inverse[RANK], setpart[y]]], FINITE],
            p3 -> member[intersection[image[RANK, x], setpart[y]], FINITE]}]]

Out[9]= or[member[intersection[image[RANK, x], setpart[y]], FINITE],
           not[subclass[P[x], FINITE]]] == True

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

Theorem 1. The variable **y** and the **setpart** wrapper can be eliminated all at once. This result says that if every subset of **x** is finite, then every subset of **image[RANK, x]** is finite.

```
In[11]:= Map[equal[V, #] &,
           SubstTest[class, y, implies[subclass[u, v], member[intersection[w, setpart[y]], v]],
            {u → P[x], v → FINITE, w → image[RANK, x]}]] // Reverse

Out[11]= or[not[subclass[P[x], FINITE]], subclass[P[image[RANK, x]], FINITE]] == True

In[12]:= (% /. x → x_) /. Equal → SetDelayed
```

---

listing ordinals in order

The class **image[RANK,x]** is a subclass of the class **OMEGA** of all ordinal numbers. For convenience, the following equation is made into a rewrite rule:

```
In[13]:= equal[intersection[image[RANK, x], OMEGA], image[RANK, x]]

Out[13]= True

In[14]:= intersection[OMEGA, image[RANK, x_]] := image[RANK, x]
```

The function **ordlist[x]** iteratively lists ordinals in a class **x** in increasing order. Its domain is a natural number if **x** only contains a finite number of ordinals, and is equal to the set **omega** of all natural numbers otherwise. In the latter case, its range is a countably infinite subset of x

```
In[15]:= SubstTest[member, card[y], omega, y → range[ordlist[x]]] // Reverse

Out[15]= member[range[ordlist[x]], FINITE] == member[intersection[OMEGA, x], FINITE]

In[16]:= member[range[ordlist[x_]], FINITE] := member[intersection[OMEGA, x], FINITE]
```

If a class **x** only has finite subsets, then it can only have a finite number of ordinals among its members.

```
In[17]:= SubstTest[implies, and[member[u, v], subclass[v, w]],
           member[u, w], {u → range[ordlist[x]], v → P[x], w → FINITE}]

Out[17]= or[member[intersection[OMEGA, x], FINITE], not[subclass[P[x], FINITE]]] == True

In[18]:= or[member[intersection[OMEGA, x_], FINITE], not[subclass[P[x_], FINITE]]] := True
```

This fact is now applied to the class **image[RANK, x]**. If this class of ordinals has only finite subsets, then it must itself be finite.

```
In[19]:= SubstTest[implies, subclass[P[y], FINITE],
           member[intersection[OMEGA, y], FINITE], y → image[RANK, x]]

Out[19]= or[member[image[RANK, x], FINITE], not[subclass[P[image[RANK, x]], FINITE]]] == True

In[20]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. If **image[RANK, x]** is a set, then **intersection[REGULAR, x]** is a set.

```
In[21]:= SubstTest[implies, member[z, y], member[z, V], z → image[RANK, x]]

Out[21]= or[member[intersection[REGULAR, x], V], not[member[image[RANK, x], y]]] == True

In[22]:= or[member[intersection[REGULAR, x_], V], not[member[image[RANK, x_], y_]]] := True
```

Theorem. If a class **x** has only finite subsets, then **intersection[REGULAR, x]** is a set.

```
In[23]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], implies[p3, p4], not[implies[p1, p4]],
           {p1 → subclass[P[x], FINITE], p2 -> subclass[P[image[RANK, x]], FINITE],
            p3 → member[image[RANK, x], FINITE], p4 → member[intersection[REGULAR, x], V]}]]

Out[23]= or[member[intersection[REGULAR, x], V], not[subclass[P[x], FINITE]]] == True

In[24]:= or[member[intersection[REGULAR, x_], V], not[subclass[P[x_], FINITE]]] := True
```

Corollary.

```
In[25]:= SubstTest[implies,
          and[equal[y, intersection[REGULAR, x]], subclass[P[x], FINITE]], member[y, V], y → x]

Out[25]= or[member[x, V], not[subclass[x, REGULAR]], not[subclass[P[x], FINITE]]] == True

In[26]:= or[member[x_, V], not[subclass[x_, REGULAR]], not[subclass[P[x_], FINITE]]] := True
```

---

a conditional rewrite rule

In this section, a conditional rewrite rule is derived that says that a subclass of **REGULAR** has no infinite subsets if and only if it is finite.

```
In[27]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p2, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 → subclass[x, REGULAR],
            p2 → subclass[P[x], FINITE], p3 → member[x, V], p4 → member[x, FINITE]}]]

Out[27]= or[member[x, FINITE], not[subclass[x, REGULAR]], not[subclass[P[x], FINITE]]] == True

In[28]:= or[member[x_, FINITE], not[subclass[x_, REGULAR]], not[subclass[P[x_], FINITE]]] := True
```

A logical equivalence holds, and can be made into a conditional rewrite rule.

```
In[29]:= implies[subclass[x, REGULAR], equiv[subclass[P[x], FINITE], member[x, FINITE]]]

Out[29]= True

In[30]:= subclass[P[x_], FINITE] := member[x, FINITE] /; subclass[x, REGULAR]
```