rank[U[x]]

Johan G. F. Belinfante
2006 May 21

```
In[1]:= SetDirectory["l:"]; << goedel81.18a; << tools.m

        :Package Title: goedel81.18a         2006 May 18 at 9:45 p.m.

        It is now:  2006 May 21 at 18:8

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 May 8

        weightlimit = 40
```

summary

In this notebook it is shown that for any class, the rank of the sum class is equal to the sum class of the rank. This yields a new rewrite rule for **rank[U[x]]** that renders obsolete two existing rules that will be removed:

```
In[2]:= subclass[rank[x], succ[rank[U[x]]]]

Out[2]= True

In[3]:= subclass[rank[x_], succ[rank[U[x_]]]] =.

In[4]:= member[rank[x], rank[U[x]]]

Out[4]= False

In[5]:= member[rank[x_], rank[U[x_]]] =.
```

At the end of this notebook, replacements for these removed rules will be derived.

lemma

The following lemma resembles a similar one with the regularity hypothesis for the other variable.

```
In[6]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p1, p3], p4],
          not[implies[and[p1, p2], p4]], {p1 → member[x, y], p2 → member[y, REGULAR],
           p3 → member[x, REGULAR], p4 -> member[rank[x], rank[y]]}]]

Out[6]= or[member[rank[x], rank[y]], not[member[x, y]], not[member[y, REGULAR]]] == True

In[7]:= or[member[rank[x_], rank[y_]], not[member[x_, y_]], not[member[y_, REGULAR]]] := True
```

The regularity hypothesis can be replaced with a **reg** wrapper:

```
In[8]:= SubstTest[or, member[rank[x], rank[z]],
         not[member[x, z]], not[member[z, REGULAR]], z -> reg[y]]

Out[8]= or[member[rank[x], rank[reg[y]]], not[member[x, reg[y]]]] == True

In[9]:= or[member[rank[x_], rank[reg[y_]]], not[member[x_, reg[y_]]]] := True
```

rank of a sum class

In this section, the variable **x** is wrapped with **reg**. In the following lemma, two temporary variables **u** and **v** serve to expand the definition of sum class:

```
In[10]:= Map[not, SubstTest[and, implies[and[p1, p5], p3],
           implies[p2, p5], implies[p2, p4], implies[and[p3, p4], p6],
           not[implies[and[p1, p2], p6]], {p1 -> member[u, v], p2 -> member[v, reg[x]],
            p3 -> member[rank[u], rank[v]], p4 -> member[rank[v], rank[reg[x]]],
            p5 → member[v, REGULAR], p6 → member[rank[u], U[rank[reg[x]]]]}]]

Out[10]= or[member[rank[u], U[rank[reg[x]]]], not[member[u, v]], not[member[v, reg[x]]]] == True

In[11]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

The temporary variables can be simultaneously eliminated:

```
In[12]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[u, v], implies[and[member[u, v], member[v, y]], member[u, z]],
            {y → reg[x], z -> image[inverse[RANK], U[rank[reg[x]]]]}]] // Reverse

Out[12]= subclass[U[reg[x]], image[inverse[RANK], U[rank[reg[x]]]]] == True

In[13]:= (% /. x → x_) /. Equal → SetDelayed
```

This statement about an inverse image can be replaced with one about a direct image:

```
In[14]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> U[reg[x]], v -> image[inverse[RANK], U[rank[reg[x]]]], w → RANK}]

Out[14]= subclass[image[RANK, U[reg[x]]], U[rank[reg[x]]]] == True

In[15]:= (% /. x → x_) /. Equal → SetDelayed
```

Temporary lemma.

```
In[16]:= SubstTest[tc, image[RANK, reg[y]], y → U[reg[x]]]

Out[16]= tc[image[RANK, U[reg[x]]]] == rank[U[reg[x]]]

In[17]:= tc[image[RANK, U[reg[x_]]]] := rank[U[reg[x]]]
```

Applying transitive closure eliminates the image with **RANK** altogether:

```
In[18]:= SubstTest[implies, subclass[u, v], subclass[tc[u], tc[v]],
          {u -> image[RANK, U[reg[x]]], v -> U[rank[reg[x]]]}]

Out[18]= subclass[rank[U[reg[x]]], U[rank[reg[x]]]] == True

In[19]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[20]:= Map[not, SubstTest[subclass, ord[u],
            U[ord[v]], {u -> rank[U[reg[x]]], v -> rank[reg[x]]}]] // Reverse

Out[20]= member[U[rank[reg[x]]], rank[U[reg[x]]]] == False

In[21]:= (% /. x → x_) /. Equal → SetDelayed
```

Removing the **reg** wrapper yields:

```
In[22]:= SubstTest[implies, equal[x, reg[y]], subclass[rank[U[x]], U[rank[x]]], y → x]

Out[22]= or[not[member[x, REGULAR]], subclass[rank[U[x]], U[rank[x]]] == True

In[23]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[24]:= Map[not, SubstTest[implies, subclass[x, y], subclass[rank[x], rank[y]], y → P[U[x]]]]

Out[24]= member[succ[rank[U[x]]], rank[x]] == False

In[25]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[26]:= SubstTest[subclass, ord[u], ord[v], {u → rank[reg[x]], v → succ[rank[U[reg[x]]]]}]

Out[26]= subclass[rank[reg[x]], succ[rank[U[reg[x]]]]] == True

In[27]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[28]:= equal[union[rank[x], U[rank[x]]], rank[x]]

Out[28]= True

In[29]:= union[rank[x_], U[rank[x_]]] := rank[x]
```

Theorem.

```
In[30]:= SubstTest[implies, subclass[u, v], subclass[U[u], U[v]],
           {u -> rank[reg[x]], v -> succ[rank[U[reg[x]]]]}]

Out[30]= subclass[U[rank[reg[x]]], rank[U[reg[x]]]] == True

In[31]:= (% /. x → x_) /. Equal → SetDelayed
```

Removing the **reg** wrapper yields:

```
In[32]:= SubstTest[implies, equal[x, reg[y]], subclass[U[rank[x]], rank[U[x]]], y → reg[x]]

Out[32]= or[not[member[x, REGULAR]], subclass[U[rank[x]], rank[U[x]]]] == True

In[33]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining the two inclusions yields an equation:

```
In[34]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
           {p → member[x, REGULAR], u → U[rank[x]], v → rank[U[x]]}] // Reverse

Out[34]= or[equal[rank[U[x]], U[rank[x]]], not[member[x, REGULAR]]] == True

In[35]:= (% /. x → x_) /. Equal → SetDelayed
```

---

a rewrite rule for V=U[rank[x]]

In the next section a rewrite rule for the statement **V = U[rank[x]]** is needed. Our starting point is the observation that the sum class of an ordinal cannot be equal to the universal class.

```
In[36]:= Map[not, SubstTest[implies, member[y, V], not[equal[y, V]], y → U[ord[x]]]]

Out[36]= equal[V, U[ord[x]]] == False

In[37]:= equal[V, U[ord[x_]]] := False
```

The rank of a regular set is an ordinal.

```
In[38]:= SubstTest[equal, V, U[ord[y]], y → rank[reg[x]]]

Out[38]= equal[V, U[rank[reg[x]]]] == False

In[39]:= (% /. x → x_) /. Equal → SetDelayed
```

Removing the **reg** wrapper yields:

```
In[40]:= SubstTest[implies, equal[x, reg[y]], not[equal[V, U[rank[x]]]], y → x]

Out[40]= or[not[equal[V, U[rank[x]]]], not[member[x, REGULAR]]] == True

In[41]:= (% /. x → x_) /. Equal → SetDelayed
```

The converse also holds:

```
In[42]:= SubstTest[implies, equal[u, V], equal[U[u], V], u → rank[x]]
Out[42]= or[equal[V, U[rank[x]]], member[x, REGULAR]] == True

In[43]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining the two cases yields this rewrite rule:

```
In[44]:= equiv[equal[V, U[rank[x]]], not[member[x, REGULAR]]]
Out[44]= True

In[45]:= equal[V, U[rank[x_]]] := not[member[x, REGULAR]]
```

the case x is not regular

When **x** is not regular, neither is **U[x]**, and both have rank equal to **V**.

```
In[46]:= SubstTest[implies, and[equal[u, v], equal[v, w]], equal[u, w],
          {u → U[rank[x]], v → V, w → rank[U[x]]}]
Out[46]= or[equal[rank[U[x]], U[rank[x]]], member[x, REGULAR]] == True

In[47]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining this result with the earlier one for the case of a regular set yields a simple equation that can be made into a rewrite rule:

```
In[48]:= SubstTest[and, implies[p1, p2], or[p1, p2],
          {p1 → member[x, REGULAR], p2 → equal[U[rank[x]], rank[U[x]]]}]
Out[48]= True == equal[rank[U[x]], U[rank[x]]]

In[49]:= rank[U[x_]] := U[rank[x]]
```

Corollary. It is convenient to use **reify** to remove the variable x:

```
In[50]:= Map[VERTSECT, SubstTest[reify, x, rank[f[x]], f → U]] // Reverse
Out[50]= composite[RANK, BIGCUP] == composite[BIGCUP, RANK]

In[51]:= composite[RANK, BIGCUP] := composite[BIGCUP, RANK]
```

a replacement rule

Lemma.

```
In[52]:= SubstTest[subclass, ord[y], succ[U[ord[y]]], y → rank[reg[x]]]

Out[52]= subclass[rank[reg[x]], succ[U[rank[reg[x]]]]] == True

In[53]:= (% /. x → x_) /. Equal → SetDelayed
```

Removing the **reg** wrapper yields:

```
In[54]:= SubstTest[implies, equal[x, reg[y]], subclass[rank[x], succ[U[rank[x]]]], y → x]

Out[54]= or[not[member[x, REGULAR]], subclass[rank[x], succ[U[rank[x]]]]] == True

In[55]:= (% /. x → x_) /. Equal → SetDelayed
```

When **x** is not regular, the same conclusion holds:

```
In[56]:= SubstTest[implies, equal[V, y], subclass[y, succ[U[y]]], y → rank[x]]

Out[56]= or[member[x, REGULAR], subclass[rank[x], succ[U[rank[x]]]]] == True

In[57]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining the two cases yields this replacement for one of the removed rewrite rules.

```
In[58]:= SubstTest[and, implies[p1, p2], or[p1, p2],
           {p1 → member[x, REGULAR], p2 -> subclass[rank[x], succ[U[rank[x]]]]}] // Reverse

Out[58]= subclass[rank[x], succ[U[rank[x]]]] == True

In[59]:= subclass[rank[x_], succ[U[rank[x_]]]] := True
```

another replacement rule

Lemma.

```
In[60]:= SubstTest[member, ord[y], U[ord[y]], y → rank[reg[x]]]

Out[60]= member[rank[reg[x]], U[rank[reg[x]]]] == False

In[61]:= (% /. x → x_) /. Equal → SetDelayed
```

Removing the **reg** wrapper yields:

```
In[62]:= SubstTest[implies, equal[x, reg[y]], not[member[rank[x], U[rank[x]]]], y → x]

Out[62]= or[not[member[x, REGULAR]], not[member[rank[x], U[rank[x]]]]] == True

In[63]:= (% /. x → x_) /. Equal → SetDelayed
```

When **x** is not regular, the same conclusion holds. No new rewrite rule is needed for this. Combining the two cases yields:

```
In[64]:= Map[not, SubstTest[and, implies[p1, p2], or[p1, p2],
            {p1 → member[x, REGULAR], p2 -> not[member[rank[x], U[rank[x]]]]}]] // Reverse

Out[64]= member[rank[x], U[rank[x]]] == False

In[65]:= member[rank[x_], U[rank[x_]]] := False
```