# rank[Uclosure[x]]

Johan G. F. Belinfante
2006 May 21

```
In[1]:= SetDirectory["l:"]; << goedel81.21a; << tools.m

        :Package Title: goedel81.21a         2006 May 21 at 2:20 p.m.

        It is now:  2006 May 21 at 19:35

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 May 8

        weightlimit = 40
```

## summary

A formula for **rank[Uclosure[x]]** is derived, based on the idea that one can sandwich **Uclosure[x]** between **set[U[x]]** and **P[U[x]]**.

```
In[2]:= subclass[set[U[x]], Uclosure[x]]

Out[2]= True

In[3]:= subclass[Uclosure[x], P[U[x]]]

Out[3]= True
```

When **x** is set, these have the same the same rank.

```
In[4]:= equal[rank[set[U[x]]], rank[P[U[x]]]]

Out[4]= member[x, V]
```

## derivation

For any class, **Uclosure[x] i**s contained in **P[U[x]]**.

```
In[5]:= Map[not, SubstTest[implies, subclass[u, v],
          subclass[rank[u], rank[v]], {u → Uclosure[x], v → P[U[x]]}]]

Out[5]= member[succ[U[rank[x]]], rank[Uclosure[x]]] == False

In[6]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[7]:= SubstTest[ord, U[ord[y]], y → rank[reg[x]]]

Out[7]= ord[U[rank[reg[x]]]] == U[rank[reg[x]]]

In[8]:= ord[U[rank[reg[x_]]]] := U[rank[reg[x]]]
```

Lemma.

```
In[9]:= SubstTest[and, not[equal[ord[u], ord[v]]], not[member[ord[u], ord[v]]],
         {u → rank[Uclosure[reg[x]]], v → U[rank[reg[x]]]}]

Out[9]= and[not[equal[rank[Uclosure[reg[x]]], U[rank[reg[x]]]]],
          not[member[rank[Uclosure[reg[x]]], U[rank[reg[x]]]]] ==
         member[U[rank[reg[x]]], rank[Uclosure[reg[x]]]]

In[10]:= and[not[equal[rank[Uclosure[reg[x_]]], U[rank[reg[x_]]]]],
           not[member[rank[Uclosure[reg[x_]]], U[rank[reg[x_]]]]]] :=
          member[U[rank[reg[x]]], rank[Uclosure[reg[x]]]]
```

Since **set[U[x]]** is a subclass of **Uclosure[x]**, one finds

```
In[11]:= SubstTest[implies, subclass[u, v],
          subclass[rank[u], rank[v]], {u → set[U[reg[x]]], v → Uclosure[reg[x]]}]

Out[11]= member[U[rank[reg[x]]], rank[Uclosure[reg[x]]]] == True

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

The following result about ordinal numbers is needed:

```
In[13]:= equiv[and[member[ord[x], ord[y]], not[member[succ[ord[x]], ord[y]]]],
           equal[ord[y], succ[ord[x]]]] // assert

Out[13]= True

In[14]:= and[member[ord[x_], ord[y_]], not[member[succ[ord[x_]], ord[y_]]]] :=
          equal[ord[y], succ[ord[x]]]
```

Application:

```
In[15]:= SubstTest[and, member[ord[u], ord[v]], not[member[succ[ord[u]], ord[v]]],
           {u → U[rank[reg[x]]], v → rank[Uclosure[reg[x]]]}] // Reverse

Out[15]= equal[rank[Uclosure[reg[x]]], succ[U[rank[reg[x]]]]] == True

In[16]:= (% /. x → x_) /. Equal → SetDelayed
```

Now the **reg** wrapper is removed:

```
In[17]:= SubstTest[implies, equal[x, reg[y]], equal[rank[Uclosure[x]], succ[U[rank[x]]]], y → x]

Out[17]= or[equal[rank[Uclosure[x]], succ[U[rank[x]]]], not[member[x, REGULAR]]] == True
```

```
In[18]:= (% /. x → x_) /. Equal → SetDelayed
```

The same conclusion holds when **x** is not a regular set.

```
In[19]:= SubstTest[implies, and[equal[u, v], equal[v, w]],
          equal[u, w], {u -> rank[Uclosure[x]], v → V, w -> succ[U[rank[x]]]}]

Out[19]= or[equal[rank[Uclosure[x]], succ[U[rank[x]]]], member[x, REGULAR]] == True
```

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

Combining the case that **x** is regular with the case that **x** is not regular, one finds an unconditional equation that can be made into a rewrite rule.

```
In[21]:= SubstTest[and, or[p1, p2], implies[p1, p2],
          {p1 → member[x, REGULAR], p2 -> equal[rank[Uclosure[x]], succ[U[rank[x]]]]}]

Out[21]= True == equal[rank[Uclosure[x]], succ[U[rank[x]]]]
```

```
In[22]:= rank[Uclosure[x_]] := succ[U[rank[x]]]
```

A variable-free version is obtained using **reify**.

```
In[23]:= Map[VERTSECT, SubstTest[reify, x, rank[f[x]], f → Uclosure]] // Reverse

Out[23]= composite[RANK, UCLOSURE] == composite[SUCC, BIGCUP, RANK]
```

```
In[24]:= composite[RANK, UCLOSURE] := composite[SUCC, BIGCUP, RANK]
```