new Uclosure rules

Johan G. F. Belinfante
2003 September 13

```
In[1]:= << goedel52.s92; << tools.m

        :Package Title: goedel52.s92      2003 September 11 at 12:40 noon

        It is now:  2003 Sep 15 at 8:56

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 August 9

        weightlimit = 40
```

summary

In this notebook several new rules for **Uclosure** are derived.

three easy rules

Any class that contains **range[SINGLETON]** has **Uclosure** equal to **V**. One could use this to prove the three rules in this section. For the three cases considered here, the same conclusion can also be obtained more quickly as follows:

```
In[2]:= Map[Reverse[SubstTest[range, CORE[x], x -> #]] &,
          {binclosed[CAP], binclosed[CUP], fix[ACLOSURE]}] // TableForm

Out[2]//TableForm=
        Uclosure[binclosed[CAP]] == V
        Uclosure[binclosed[CUP]] == V
        Uclosure[fix[ACLOSURE]] == V

In[3]:= Uclosure[binclosed[CAP]] := V

In[4]:= Uclosure[binclosed[CUP]] := V

In[5]:= Uclosure[fix[ACLOSURE]] := V
```

use of subvar

The following fact is needed later in this notebook.

```
In[6]:= SubstTest[Uclosure, subvar[y], y -> cart[complement[x], V]]

Out[6]= Uclosure[complement[P[x]]] == union[complement[P[x]], singleton[0]]
```

```
In[7]:= Uclosure[complement[P[x_]]] := union[complement[P[x]], singleton[0]]
```

The following is not used again in this notebook, but seems interesting anyhow.

```
In[8]:= SubstTest[Uclosure, subvar[y], y -> Di]

Out[8]= Uclosure[complement[range[SINGLETON]]] == complement[range[SINGLETON]]

In[9]:= Uclosure[complement[range[SINGLETON]]] := complement[range[SINGLETON]]
```

a lemma involving ADJOIN

```
In[10]:= SubstTest[implies, member[y, V], member[image[funpart[z], y], V], z -> ADJOIN[x]]

Out[10]= or[member[image[ADJOIN[x], y], V], not[member[y, V]]] == True

In[11]:= or[member[image[ADJOIN[x_], y_], V], not[member[y_, V]]] := True

In[12]:= SubstTest[implies, member[w, V], member[image[funpart[z], w], V],
          {w -> image[ADJOIN[x], y], z -> IMAGE[id[U[y]]]}]

Out[12]= or[member[y, V], not[member[x, V]], not[member[image[ADJOIN[x], y], V]]] == True

In[13]:= or[member[y_, V], not[member[x_, V]], not[member[image[ADJOIN[x_], y_], V]]] := True

In[14]:= Map[or[member[x, V], #] &,
           SubstTest[implies, equal[0, w], member[w, V], w -> image[ADJOIN[x], y]]]

Out[14]= or[member[x, V], member[image[ADJOIN[x], y], V]] == True

In[15]:= or[member[x_, V], member[image[ADJOIN[x_], y_], V]] := True

In[16]:= or[and[member[x, V], not[member[y, V]]], member[image[ADJOIN[x], y], V]] // NotNotTest

Out[16]= or[and[member[x, V], not[member[y, V]]], member[image[ADJOIN[x], y], V]] == True

In[17]:= or[and[member[x_, V], not[member[y_, V]]], member[image[ADJOIN[x_], y_], V]] := True

In[18]:= equiv[member[image[ADJOIN[x], y], V], or[not[member[x, V]], member[y, V]]]

Out[18]= True

In[19]:= member[image[ADJOIN[x_], y_], V] := or[member[y, V], not[member[x, V]]]
```

Uclosure of image[ADJOIN[singleton[0]], range[SINGLETON]]

The class **image[ADJOIN[singleton[0]], range[SINGLETON]]** is the class of all sets of the form **pairset[0,x]**. These are known in topology as the indiscrete or trivial topologies.

```
In[20]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
              {u -> image[SINGLETON, y], v -> range[SINGLETON]}]

Out[20]= subclass[image[w, image[SINGLETON, y]], image[w, range[SINGLETON]]] == True
```

```
In[21]:= subclass[image[w_, image[SINGLETON, y_]], image[w_, range[SINGLETON]]] := True
```

Lemma.

```
In[22]:= SubstTest[implies, member[u, P[v]], member[U[u], Uclosure[v]],
           {u -> image[ADJOIN[singleton[0]], image[SINGLETON, y]],
          v -> image[ADJOIN[singleton[0]], range[SINGLETON]]}]

Out[22]= or[member[union[y, intersection[image[V, y], singleton[0]]],
           Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]]], not[member[y, V]]] == True

In[23]:= (% /. y -> y_) /. Equal -> SetDelayed

In[24]:= image[inverse[ADJOIN[singleton[0]]], z] // Renormality // Reverse

Out[24]= fix[composite[S, IMAGE[id[complement[singleton[0]]]],
           id[intersection[z, complement[P[complement[singleton[0]]]]]], S]] ==
         image[inverse[ADJOIN[singleton[0]]], z]

In[25]:= fix[composite[S, IMAGE[id[complement[singleton[0]]]],
           id[intersection[z, complement[P[complement[singleton[0]]]]]], S]] :=
         image[inverse[ADJOIN[singleton[0]]], z]

In[26]:= SubstTest[class, y,
          or[member[union[y, intersection[image[V, y], singleton[0]]], z], not[member[y, V]]],
          z -> Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]]] // Reverse

Out[26]= union[image[inverse[ADJOIN[singleton[0]]],
           Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]]], singleton[0]] == V

In[27]:= union[image[inverse[ADJOIN[singleton[0]]],
           Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]]], singleton[0]] := V

In[28]:= member[singleton[0],
          image[ADJOIN[singleton[0]], complement[singleton[0]]]] // AssertTest

Out[28]= member[singleton[0], image[ADJOIN[singleton[0]], complement[singleton[0]]]] == True

In[29]:= member[singleton[0], image[ADJOIN[singleton[0]], complement[singleton[0]]]] := True

In[30]:= equal[union[image[ADJOIN[singleton[0]], complement[singleton[0]]],
          singleton[singleton[0]]],
           image[ADJOIN[singleton[0]], complement[singleton[0]]]]

Out[30]= True

In[31]:= union[image[ADJOIN[singleton[0]], complement[singleton[0]]],
          singleton[singleton[0]]] := image[ADJOIN[singleton[0]], complement[singleton[0]]]

In[32]:= SubstTest[image, w, union[u, v], {u -> complement[singleton[0]],
           v -> singleton[0], w -> ADJOIN[singleton[0]]}] // Reverse

Out[32]= image[ADJOIN[singleton[0]], complement[singleton[0]]] ==
         complement[P[complement[singleton[0]]]]

In[33]:= image[ADJOIN[singleton[0]], complement[singleton[0]]] :=
         complement[P[complement[singleton[0]]]]
```

```
In[34]:= Map[not, SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u -> complement[singleton[0]], v -> image[inverse[ADJOIN[singleton[0]]],
            Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]],
              w -> ADJOIN[singleton[0]]}]]

Out[34]= member[0, U[complement[Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]]]]] ==
          False

In[35]:= member[0, U[complement[Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]]]]] :=
          False

In[36]:= SubstTest[implies, subclass[u, v], subclass[Uclosure[u], Uclosure[v]],
          {u -> image[ADJOIN[singleton[0]], range[SINGLETON]],
           v -> complement[P[complement[singleton[0]]]]}]

Out[36]= subclass[intersection[P[complement[singleton[0]]],
           Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]]], singleton[0]] == True

In[37]:= subclass[intersection[P[complement[singleton[0]]],
           Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]]], singleton[0]] := True

In[38]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]],
           v -> union[singleton[0], complement[P[complement[singleton[0]]]]]}] // Reverse

Out[38]= equal[Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]],
           union[complement[P[complement[singleton[0]]]], singleton[0]]] == True
```

The **Uclosure** of the class of trivial topologies is the class of all sets that are either empty or hold the empty set as a member.

```
       equal[Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]],
         union[complement[P[complement[singleton[0]]]], singleton[0]]] == True

In[39]:= Uclosure[image[ADJOIN[singleton[0]], range[SINGLETON]]] :=
          union[complement[P[complement[singleton[0]]]], singleton[0]]
```

the case of fix[UCLOSURE]

```
In[40]:= Map[not, SubstTest[implies, subclass[u, v], subclass[Uclosure[u], Uclosure[v]],
           {u -> image[ADJOIN[singleton[0]], range[SINGLETON]], v -> fix[UCLOSURE]}]]

Out[40]= member[0, U[complement[Uclosure[fix[UCLOSURE]]]]] == False

In[41]:= member[0, U[complement[Uclosure[fix[UCLOSURE]]]]] := False

In[42]:= SubstTest[implies, subclass[u, v], subclass[Uclosure[u], Uclosure[v]],
          {u -> fix[UCLOSURE], v -> complement[P[complement[singleton[0]]]]}]

Out[42]= subclass[intersection[P[complement[singleton[0]]], Uclosure[fix[UCLOSURE]]],
           singleton[0]] == True

In[43]:= subclass[intersection[P[complement[singleton[0]]], Uclosure[fix[UCLOSURE]]],
           singleton[0]] := True

In[44]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> Uclosure[fix[UCLOSURE]],
           v -> union[singleton[0], complement[P[complement[singleton[0]]]]]}] // Reverse

Out[44]= equal[Uclosure[fix[UCLOSURE]],
           union[complement[P[complement[singleton[0]]]], singleton[0]]] == True
```

```
In[45]:= Uclosure[fix[UCLOSURE]] :=
          union[complement[P[complement[singleton[0]]]], singleton[0]]
```

the case of binclosed[DIF]

The class **binclosed[DIF]** is the class of all collections of sets that are closed under the operation **dif[x,y] = intersection[x, complement[y]]**.

```
In[46]:= SubstTest[implies, subclass[u, v], subclass[Uclosure[u], Uclosure[v]],
             {u -> binclosed[DIF], v -> union[singleton[0], image[E, singleton[0]]]}]

Out[46]= subclass[intersection[P[complement[singleton[0]]], Uclosure[binclosed[DIF]]],
           singleton[0]] == True

In[47]:= subclass[intersection[P[complement[singleton[0]]], Uclosure[binclosed[DIF]]],
           singleton[0]] := True

In[48]:= Map[equal[V, #] &, SubstTest[class, x, member[singleton[x], z],
            z -> image[inverse[ADJOIN[singleton[0]]], binclosed[DIF]]]] // Reverse

Out[48]= subclass[image[ADJOIN[singleton[0]], range[SINGLETON]], binclosed[DIF]] == True

In[49]:= subclass[image[ADJOIN[singleton[0]], range[SINGLETON]], binclosed[DIF]] := True

In[50]:= Map[not, SubstTest[implies, subclass[u, v], subclass[Uclosure[u], Uclosure[v]],
             {u -> image[ADJOIN[singleton[0]], range[SINGLETON]], v -> binclosed[DIF]}]]

Out[50]= member[0, U[complement[Uclosure[binclosed[DIF]]]]] == False

In[51]:= member[0, U[complement[Uclosure[binclosed[DIF]]]]] := False

In[52]:= SubstTest[and, subclass[u, v], subclass[v, u],
             {u -> Uclosure[binclosed[DIF]],
            v -> union[singleton[0], complement[P[complement[singleton[0]]]]]}] // Reverse

Out[52]= equal[Uclosure[binclosed[DIF]],
           union[complement[P[complement[singleton[0]]]], singleton[0]]] == True

In[53]:= Uclosure[binclosed[DIF]] :=
          union[complement[P[complement[singleton[0]]]], singleton[0]]
```

the case of TOPS

The **Uclosure** of the class of all topologies is the same as that of the class of trivial topologies.

```
In[54]:= Map[not, SubstTest[implies, subclass[u, v], subclass[Uclosure[u], Uclosure[v]],
           {u -> image[ADJOIN[singleton[0]], range[SINGLETON]], v -> TOPS}]]

Out[54]= member[0, U[complement[Uclosure[TOPS]]]] == False

In[55]:= member[0, U[complement[Uclosure[TOPS]]]] := False

In[56]:= SubstTest[implies, subclass[u, v],
          subclass[Uclosure[u], Uclosure[v]], {u -> TOPS, v -> image[E, singleton[0]]}]

Out[56]= subclass[intersection[P[complement[singleton[0]]], Uclosure[TOPS]], singleton[0]] ==
          True
```

```
In[57]:= subclass[
           intersection[P[complement[singleton[0]]], Uclosure[TOPS]], singleton[0]] := True

In[58]:= SubstTest[and, subclass[u, v], subclass[v, u],
             {u -> Uclosure[TOPS],
              v -> union[singleton[0], complement[P[complement[singleton[0]]]]]}] // Reverse

Out[58]= equal[Uclosure[TOPS],
           union[complement[P[complement[singleton[0]]]], singleton[0]]] == True

In[59]:= Uclosure[TOPS] := union[complement[P[complement[singleton[0]]]], singleton[0]]
```