SUCC and S

Johan G. F. Belinfante and Ming Li
2006 April 24

```
In[1]:= SetDirectory["l:"]; << goedel80.23a; << tools.m

        :Package Title: goedel80.23a          2006 April 23 at 6:45 p.m.

        It is now:  2006 Apr 24 at 18:31

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 March 7

        weightlimit = 40
```

summary

This notebook provides an elementary example that may help to illustrate a theorem about progessing functions.

```
In[2]:= "Raymond M. Smullyan and Melvin Fitting, Set Theory and the Continuum Problem,
           Oxford Science Publications, Clarendon Press, Oxford, 1996.  (see page 35)";
```

A function **x** is **progressing** if it is a subclass of the subset relation **S**. Some examples are:

```
In[3]:= Select[Funs[NamedClasses], subclass[#, S] &]

Out[3]= {0, ACLOSURE, FACTORIAL, Id, PRIMESEQ, SUCC, TC, UCLOSURE}

In[4]:= Select[UnaryFunctors, subclass[#[x], S] &]

Out[4]= {ADJOIN, HULL, id, plus}
```

Many of these functions are also **monotone** with respect to inclusion:

```
In[5]:= Map[assert[monotone[#, S, S]] &,
         Union[{0, ACLOSURE, FACTORIAL, Id, PRIMESEQ, SUCC, TC, UCLOSURE},
          Map[#[x] &, {ADJOIN, HULL, id, plus}]]]

Out[5]= {True, True, True, True, subclass[composite[PRIMESEQ, S, inverse[PRIMESEQ]], S],
         subclass[composite[SUCC, S, inverse[SUCC]], S], True, True, True, True, True, True}
```

In this notebook it is shown that the successor function is not monotone in general, although its restriction to **omega** is monotone.

the restriction to omega

The restriction of **SUCC** to the set **omega** is:

```
In[6]:= restrict[SUCC, omega, omega]

Out[6]= composite[id[omega], SUCC]
```

Lemma.

```
In[7]:= subclass[composite[id[omega], E], composite[S, id[omega], SUCC]] // AssertTest

Out[7]= subclass[composite[id[omega], E], composite[S, id[omega], SUCC]] == True

In[8]:= subclass[composite[id[omega], E], composite[S, id[omega], SUCC]] := True
```

Theorem. The restriction of **SUCC** to **omega** is monotone.

```
In[9]:= SubstTest[subtwine, funpart[x],
          restrict[S, omega, omega], S, x → composite[id[omega], SUCC]] // Reverse

Out[9]= subclass[composite[id[omega], SUCC, S, inverse[SUCC], id[omega]], S] == True

In[10]:= subclass[composite[id[omega], SUCC, S, inverse[SUCC], id[omega]], S] := True
```

a counterexample

In this section it is shown that the unrestricted successor function is not monotone. The basic idea is this: Although **0** is a subclass of **set[set[0]]**, the successor of **0** is not a subclass of the successor of **set[set[0]]**.

```
In[11]:= implies[subclass[x, y], subclass[succ[x], succ[y]]] /. {x → 0, y → set[set[0]]}

Out[11]= False
```

Lemma.

```
In[12]:= Map[not, SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
            {u → set[PAIR[0, set[set[0]]]], v → S, w -> composite[inverse[SUCC], S, SUCC]}]]

Out[12]= subclass[S, composite[inverse[SUCC], S, SUCC]] == False

In[13]:= % /. Equal → SetDelayed
```

Theorem. The unrestricted successor function is not monotone.

```
In[14]:= (subclass[S, composite[inverse[funpart[x]], S, funpart[x]]] // AssertTest // Reverse) /.
           x → SUCC

Out[14]= subclass[composite[SUCC, S, inverse[SUCC]], S] == False
```

```
In[15]:= subclass[composite[SUCC, S, inverse[SUCC]], S] := False
```