subvar[x] need not be closed under intersections

Johan G. F. Belinfante
2007 July 5

```
In[1]:= SetDirectory["l:"]; << goedel95.04a; << tools.m

        :Package Title: goedel95.04a      2007 July 4 at 9:55 p.m.

        It is now:  2007 Jul 5 at 22:23

        Loading Simplification Rules

        TOOLS.M                                      Revised 2007 June 25

        weightlimit = 40
```

summary

The class **subvar[x]** is closed under arbitrary unions, but need not be closed under binary intersections. An explicit counter-example is constructed using low-rank sets. One can systematically generate as many low rank sets as desired using **ens**. Only a few of these are actually needed.

```
In[2]:= Map[Print["ens[", #, "] = ", ens[#]] &, Range[0, 6]];

        ens[0] = 0

        ens[1] = set[0]

        ens[2] = set[set[0]]

        ens[3] = succ[set[0]]

        ens[4] = set[set[set[0]]]

        ens[5] = set[0, set[set[0]]]

        ens[6] = succ[set[set[0]]]
```

general results

Theorem. If **x** and **y** are members of a class **z** that is closed under binary intersections, then **intersection[x,y]** is also a member of **z**.

```
In[3]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
          {u → PAIR[x, y], v -> cartsq[z], w -> image[inverse[CAP], z]}] // Reverse

Out[3]= or[member[intersection[x, y], z], not[member[x, z]],
          not[member[y, z]], not[subclass[image[CAP, cart[z, z]], z]]] == True
```

```
In[4]:= or[member[intersection[x_, y_], z_], not[member[x_, z_]],
          not[member[y_, z_]], not[subclass[image[CAP, cart[z_, z_]], z_]]] := True
```

Theorem. The constructor **subvar[x]** and the function **SUBVAR** are related as follows.

```
In[5]:= SubstTest[member, setpart[x], image[inverse[SUBVAR], v], v → range[SUBVAR]]

Out[5]= member[subvar[setpart[x]], range[SUBVAR]] == True

In[6]:= member[subvar[setpart[x_]], range[SUBVAR]] := True
```

Theorem. If a class is closed under arbitrary intersections, then it is closed under binary intersections.

```
In[7]:= SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
          {u → PAIR[x, y], v -> cartsq[Aclosure[z]], w -> image[inverse[CAP], Aclosure[z]]}] //
         Reverse

Out[7]= or[member[intersection[x, y], Aclosure[z]],
          not[member[x, Aclosure[z]]], not[member[y, Aclosure[z]]]] == True

In[8]:= or[member[intersection[x_, y_], Aclosure[z_]],
          not[member[x_, Aclosure[z_]]], not[member[y_, Aclosure[z_]]]] := True
```

---

the counterexample

Theorem. (A specific counterexample.)

```
In[9]:= Map[not, SubstTest[implies,
           and[member[x, z], member[y, z], subclass[image[CAP, cart[z, z]], z]],
           member[intersection[x, y], z], {x → ens[3], y → ens[5],
            z -> subvar[union[cart[ens[1], ens[6]], cart[ens[6], ens[1]]]]}]] // Reverse

Out[9]= subclass[image[CAP, cart[
            subvar[union[cart[set[0], succ[set[set[0]]]], cart[succ[set[set[0]]], set[0]]]],
            subvar[union[cart[set[0], succ[set[set[0]]]], cart[succ[set[set[0]]], set[0]]]]],
          subvar[union[cart[set[0], succ[set[set[0]]]], cart[succ[set[set[0]]], set[0]]]]] ==
         False

In[10]:= % /. Equal → SetDelayed
```

Lemma.

```
In[11]:= SubstTest[member, subvar[setpart[x]], range[SUBVAR],
           x -> union[cart[ens[1], ens[6]], cart[ens[6], ens[1]]]] // Reverse

Out[11]= member[
           subvar[union[cart[set[0], succ[set[set[0]]]], cart[succ[set[set[0]]], set[0]]]],
           range[SUBVAR]] == True

In[12]:= % /. Equal → SetDelayed
```

Main theorem.

```
In[13]:= Map[not, SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w], {u ->
             subvar[union[cart[set[0], succ[set[set[0]]]], cart[succ[set[set[0]]], set[0]]]],
           v → range[SUBVAR], w → binclosed[CAP]}] // Reverse]
```

```
Out[13]= subclass[range[SUBVAR], binclosed[CAP]] == False
```

```
In[14]:= subclass[range[SUBVAR], binclosed[CAP]] := False
```

Corollary.

```
In[15]:= Map[not, SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
            {u → range[SUBVAR], v → fix[ACLOSURE], w → binclosed[CAP]}] // Reverse]
```

```
Out[15]= subclass[range[SUBVAR], fix[ACLOSURE]] == False
```

```
In[16]:= subclass[range[SUBVAR], fix[ACLOSURE]] := False
```

Corollary. The power set of any set is a topology, but in general, **subvar[x]** need not be a topology.

```
In[17]:= Map[not, SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
            {u → range[SUBVAR], v → TOPS, w → binclosed[CAP]}] // Reverse]
```

```
Out[17]= subclass[range[SUBVAR], TOPS] == False
```

```
In[18]:= subclass[range[SUBVAR], TOPS] := False
```