subvar from SUBVAR

Johan G. F. Belinfante
2004 January 22

In[1]:= << goede153.19c; << tools.m

:Package Title: goede153.19c 2004 January 19 at 4:30 p.m.
It is now: 2004 Jan 23 at 10:24
Loading Simplification Rules
TOOLS.M Revised 2004 January 3

weightlimit = 40

summary

The functor subvar can be applied to any class, not just to sets. The function SUBVAR only takes x to subvar[x] when x is a set. One would therefore be inclined to conclude that the constructor subvar must contain more information than the function SUBVAR. Nonetheless it will be shown below that one can use the function SUBVAR to construct subvar[x] for any class. The idea is that subvar[x] is the union of the class of sets obtained by applying subvar to all subsets of x.

a lower bound

On account of the monotonicity property of subvar, one can approximate its action on proper classes in terms of large subsets. A lower bound is obtained as follows:

In[2]:= Map[equal[V, #] &, SubstTest[class, w, implies[subclass[w, x], subclass[subvar[w], z]], z -> subvar[x]]] // Reverse

Out[2]= subclass[U[image[SUBVAR, P[x]]], subvar[x]] = True

In[3]:= (% /. x -> x_) /. Equal -> SetDelayed

an upper bound

The key fact that allows one to obtain an upper bound is this:

In[4]:= implies[member[y, subvar[x]], member[y, subvar[restrict[x, y, y]]]]


Note that if y is a set, then restrict[x,y,y] is a subset of the class x. Some lemmas are needed to exploit this idea.
The above results are assembled in the following theorem to obtain an upper bound for \texttt{subvar[x]}.

\[
\text{Map[equal[V, class[y, not[#]]] &, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p3, p4], implies[and[p2, p4], p5], not[implies[p1, p5]], p3 -> member[y, subvar[x]], p2 -> member[y, subvar[restrict[x, y, y]]], p3 -> member[restrict[x, y, y]], image[SUBVAR, P[x]], p4 -> member[subvar[restrict[x, y, y]], image[SUBVAR, P[x]]], p5 -> member[y, U[image[SUBVAR, P[x]]]]]]
\]

\[
\text{subclass[subvar[x], U[image[SUBVAR, P[x]]]] = True}
\]

\[
\text{(/. x -> x_)} / \text{Equal -> SetDelayed}
\]

Putting these facts together, one obtains:

\[
\text{SubstTest[and, subclass[u, v], subclass[v, u], U[image[SUBVAR, P[x]]]]}
\]

\[
\text{True = equal[subvar[x], U[image[SUBVAR, P[x]]]]}
\]

\[
\text{U[image[SUBVAR, P[x_]]] = subvar[x]}
\]

\section*{some corollaries}

Corollary.
In[17] := SubstTest[U, image[SUBVAR, P[x]], x -> V]  
Out[17] = U[range[SUBVAR]] = V  
In[18] := U[range[SUBVAR]] := V  

Corollary.

In[19] := composite[inverse[E], SUBVAR, inverse[S]] // RelnNormality  
Out[19] = composite[inverse[E], SUBVAR, inverse[S]] = composite[inverse[E], SUBVAR]  
In[20] := composite[inverse[E], SUBVAR, inverse[S]] := composite[inverse[E], SUBVAR]  

Corollary.

In[21] := SubstTest[VERTSECT,  
 composite[inverse[E], funpart[x], inverse[S]], x -> SUBVAR] // Reverse  
Out[21] = composite[BIGCUP, IMAGE[SUBVAR], POWER] := SUBVAR  
In[22] := composite[BIGCUP, IMAGE[SUBVAR], POWER] := SUBVAR  

---

an inclusion and a counterexample

In[23] := SubstTest[implies, subclass[x, y], subclass[image[z, x], image[z, y]],  
   {y -> P[U[x]], z -> composite[inverse[E], SUBVAR]}]  
Out[23] = subclass[U[image[SUBVAR, x]], subvar[U[x]]] = True  
In[24] := subclass[U[image[SUBVAR, x]], subvar[U[x_]]] := True  

The reverse inclusion does not hold in general, as shown by the following counterexample:

In[25] := Map[not, SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],  
   {u -> succ[singleton[0]], v -> P[succ[singleton[0]]],  
    w -> union[pairset[0, singleton[singleton[0]]], succ[singleton[0]]]}]]  
Out[25] = subclass[P[succ[singleton[0]]]],  
   union[pairset[0, singleton[singleton[0]]], succ[singleton[0]]]] = False  
In[26] := % /. Equal -> SetDelayed  
In[27] := (subclass[subvar[U[z]], U[image[SUBVAR, z]]] /. z -> pairset[id[x], id[y]]) /.  
{x -> singleton[0], y -> singleton[singleton[0]]}  
Out[27] = False