subvar for idempotent functions

Johan G. F. Belinfante
2003 November 4

```
In[1]:= << goedel52.t11; << tools.m

        :Package Title: goedel52.t11      2003 November 2 at 8:30 p.m.

        It is now:  2003 Nov 7 at 9:18

        Loading Simplification Rules

        TOOLS.M                       Revised 2003 October 28

        weightlimit = 40
```

summary

Rewrite rules for **subvar[x]** are derived, especially for the case that **x** is an idempotent function.

```
In[2]:= idempotent[x_] := equal[composite[x, x], x]
```

a general result

```
In[3]:= implies[and[FUNCTION[x], idempotent[x]], equal[subvar[x], P[fix[x]]]]

Out[3]= or[equal[P[fix[x]], subvar[x]], not[equal[x, composite[x, x]]], not[FUNCTION[x]]]

In[4]:= SubstTest[implies, and[equal[u, v], subclass[v, w]],
         subclass[u, w], {u -> P[range[x]], v -> P[fix[x]], w -> subvar[x]}]

Out[4]= or[not[equal[fix[x], range[x]]], subclass[P[range[x]], subvar[x]]] == True

In[5]:= or[not[equal[fix[x_], range[x_]]], subclass[P[range[x_]], subvar[x_]]] := True

In[6]:= SubstTest[and, implies[p, subclass[u, v]], subclass[v, u],
            {p -> equal[fix[x], range[x]], u -> P[range[x]], v -> subvar[x]}] // Reverse

Out[6]= or[equal[P[range[x]], subvar[x]], not[equal[fix[x], range[x]]]] == True

In[7]:= or[equal[P[range[x_]], subvar[x_]], not[equal[fix[x_], range[x_]]]] := True

In[8]:= SubstTest[implies, and[equal[u, v], equal[v, x]], equal[u, x], {u -> P[y], v -> P[z]}]

Out[8]= or[equal[x, P[y]], not[equal[x, P[z]]], not[equal[y, z]]] == True

In[9]:= or[equal[x_, P[y_]], not[equal[x_, P[z_]]], not[equal[y_, z_]]] := True
```

```
In[10]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
           implies[p3, p4], implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 -> FUNCTION[x], p2 -> idempotent[x], p3 -> equal[fix[x], range[x]],
            p4 -> equal[subvar[x], P[range[x]]], p5 -> equal[subvar[x], P[fix[x]]]}]]

Out[10]= or[equal[P[fix[x]], subvar[x]],
           not[equal[x, composite[x, x]]], not[FUNCTION[x]]] == True

In[11]:= or[equal[P[fix[x_]], subvar[x_]],
           not[equal[x_, composite[x_, x_]]], not[FUNCTION[x_]]] := True
```

---

examples

```
In[12]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> ACLOSURE]

Out[12]= equal[P[fix[ACLOSURE]], subvar[ACLOSURE]] == True

In[13]:= subvar[ACLOSURE] := P[fix[ACLOSURE]]

In[14]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> ADJOIN[x]]

Out[14]= equal[P[image[S, singleton[x]]], subvar[ADJOIN[x]]] == True

In[15]:= subvar[ADJOIN[x_]] := P[image[S, singleton[x]]]

In[16]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> CARD]

Out[16]= equal[P[fix[CARD]], subvar[CARD]] == True

In[17]:= subvar[CARD] := P[fix[CARD]]

In[18]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> CORE[x]]

Out[18]= equal[P[Uclosure[x]], subvar[CORE[x]]] == True

In[19]:= subvar[CORE[x_]] := P[Uclosure[x]]

In[20]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> FUNPART]

Out[20]= equal[P[FUNS], subvar[FUNPART]] == True

In[21]:= subvar[FUNPART] := P[FUNS]

In[22]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> HC]

Out[22]= equal[P[FULL], subvar[HC]] == True

In[23]:= subvar[HC] := P[FULL]

In[24]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> HULL[x]]

Out[24]= equal[P[fix[HULL[x]]], subvar[HULL[x]]] == True
```

```
In[25]:= subvar[HULL[x_]] := P[fix[HULL[x]]]

In[26]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> IMAGE[id[x]]]

Out[26]= equal[P[P[x]], subvar[IMAGE[id[x]]]] == True

In[27]:= subvar[IMAGE[id[x_]]] := P[P[x]]

In[28]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> RANK]

Out[28]= equal[P[OMEGA], subvar[RANK]] == True

In[29]:= subvar[RANK] := P[OMEGA]

In[30]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> TC]

Out[30]= equal[P[FULL], subvar[TC]] == True

In[31]:= subvar[TC] := P[FULL]

In[32]:= SubstTest[implies, and[FUNCTION[z], idempotent[z]],
          equal[subvar[z], P[fix[z]]], z -> UCLOSURE]

Out[32]= equal[P[fix[UCLOSURE]], subvar[UCLOSURE]] == True

In[33]:= subvar[UCLOSURE] := P[fix[UCLOSURE]]
```

some other cases

Some other cases that require little reasoning can be added:

```
In[34]:= equal[V, subvar[COMMUTE]]

Out[34]= True

In[35]:= subvar[COMMUTE] := V

In[36]:= equal[V, subvar[SUBCOMMUTE]]

Out[36]= True

In[37]:= subvar[SUBCOMMUTE] := V

In[38]:= equal[subvar[COARSER], V]

Out[38]= True

In[39]:= subvar[COARSER] := V
```