domains of thin relations

Johan G. F. Belinfante
2004 August 4

```
In[1]:= SetDirectory["i:"]; << goedel60.03a; << tools.m;

        :Package Title: goedel60.03a            2004 August 3 at 5:15 p.m.

        It is now:  2004 Aug 4 at 16:29

        Loading Simplification Rules

        TOOLS.M                     Revised 2004 June 22

        weightlimit = 40
```

summary

For a thin relation, the class of subsets of its domain is equal to the class of domains of its subsets. This result is reformulated using thinpart wrappers, and some related rewrite rules are derived. It remains unclear whether this holds for all relations.

a new version of an old theorem

The following theorem is already known, and will be used to derive an equivalent formulation using **thinpart** wrappers.

```
In[2]:= implies[thin[x], equal[image[IMAGE[FIRST], P[x]], P[domain[x]]]]

Out[2]= True
```

One can use this to derive a version involving **thinpart** wrappers:

```
In[3]:= SubstTest[implies, thin[y],
         equal[image[IMAGE[FIRST], P[y]], P[domain[y]]], y → thinpart[x]]

Out[3]= equal[image[IMAGE[FIRST], P[thinpart[x]]], P[domain[thinpart[x]]]] == True

In[4]:= image[IMAGE[FIRST], P[thinpart[x_]]] := P[domain[thinpart[x]]]
```

This rule would not be needed if one adopts the following conditional rewrite rule. Since conditional rewrite rules cause the **GOEDEL** program to slow down, the **cond** flag is included here so that the conditional rule can be turned off. When the **cond** flag is turned off, the **thinpart** version is no longer superfluous, so it seems best to retain all of these versions.

```
In[5]:=  image[IMAGE[FIRST], P[x_]] := P[domain[x]] /; cond && thin[x]
```

a formula needed for a corollary

Originally, a long sequence of lemmas was used to derive the formula for **domain[UB[composite[E, inverse[E]]]** obtained in this section. For brevity, a more direct albeit less motivated method is used here.

```
In[6]:=  simplify = False;

In[7]:=  fix[composite[DISJOINT, ADJOIN[singleton[0]],
            BIGCUP, id[P[complement[singleton[0]]]], E]] // Renormality

Out[7]=  fix[composite[DISJOINT, ADJOIN[singleton[0]],
            BIGCUP, id[P[complement[singleton[0]]]], E]] == 0

In[8]:=  % /. Equal → SetDelayed

In[9]:=  SubstTest[composite, x, id[domain[x]],
           x -> dif[composite[ADJOIN[singleton[0]], BIGCUP,
              id[P[complement[singleton[0]]]], E], composite[E, inverse[E]]]] // Reverse

Out[9]=  intersection[DISJOINT, composite[ADJOIN[singleton[0]],
            BIGCUP, id[P[complement[singleton[0]]]], E]] == 0

In[10]:=  % /. Equal → SetDelayed

In[11]:=  SubstTest[equal, 0, dif[u, v], {u -> composite[
               ADJOIN[singleton[0]], BIGCUP, id[P[complement[singleton[0]]]], E],
            v -> composite[E, inverse[E]]}] // Reverse

Out[11]=  subclass[composite[ADJOIN[singleton[0]], BIGCUP,
             id[P[complement[singleton[0]]]], E], composite[E, inverse[E]]] == True

In[12]:=  % /. Equal → SetDelayed

In[13]:=  SubstTest[implies, subclass[u, v], subclass[domain[u], domain[v]],
           {u -> composite[ADJOIN[singleton[0]], BIGCUP,
              id[P[complement[singleton[0]]]]], v -> UB[composite[E, inverse[E]]]}]

Out[13]=  subclass[P[complement[singleton[0]]],
            domain[UB[composite[E, inverse[E]]]]] == True

In[14]:=  % /. Equal → SetDelayed
```

One more step yields the main result of this section:

```
In[15]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> P[complement[singleton[0]]], v -> domain[UB[composite[E, inverse[E]]]]}]

Out[15]= True ==
          equal[domain[UB[composite[E, inverse[E]]]], P[complement[singleton[0]]]]

In[16]:= domain[UB[composite[E, inverse[E]]]] := P[complement[singleton[0]]]
```

a corollary

The formula derived in the preceding section is used here to derive a formula for **domain[-UB[composite[E, x]]]** in the case that **x** is thin. The result of the preceding section is actually a special case of this more general result because the inverse membership relation **inverse[E]** is thin.

```
In[17]:= domain[UB[composite[E, thinpart[x]]]] // Normality

Out[17]= domain[UB[composite[E, thinpart[x]]]] == P[domain[thinpart[x]]]

In[18]:= domain[UB[composite[E, thinpart[x_]]]] := P[domain[thinpart[x]]]

In[19]:= SubstTest[implies,
          and[equal[u, v], equal[domain[UB[composite[E, v]]], P[domain[v]]]],
          equal[domain[UB[composite[E, u]]], P[domain[u]]],
          {u → composite[Id, x], v → thinpart[x]}]

Out[19]= or[equal[domain[UB[composite[E, x]]], P[domain[x]]],
           not[equal[V, domain[VERTSECT[x]]]]] == True

In[20]:= or[equal[domain[UB[composite[E, x_]]], P[domain[x_]]],
           not[equal[V, domain[VERTSECT[x_]]]]] := True
```

Restatement:

```
In[21]:= implies[thin[x], equal[domain[UB[composite[E, x]]], P[domain[x]]]]

Out[21]= True
```

Conditional rule:

```
In[22]:= domain[UB[composite[E, x_]]] := P[domain[x]] /; cond && thin[x]
```

comment

The following weaker result holds without the assumption of thinness. It was this result that led to the discovery of the theorem in the preceding section.

```
In[23]:= Map[equal[V, class[z, #]] &, Map[equal[V, class[y, #]] &,
           SubstTest[implies, and[member[pair[u, v], composite[Id, w]], member[u, t]],
            member[v, image[w, t]],
            {u → composite[id[y], x, id[z]], v → z, w → IMAGE[FIRST], t → P[x]}]]]

Out[23]= subclass[domain[UB[composite[E, x]]], image[IMAGE[FIRST], P[x]]] == True

In[24]:= subclass[domain[UB[composite[E, x_]]], image[IMAGE[FIRST], P[x_]]] := True
```

an open question

The **thinness** hypothesis suffices for the theorem about the equality of the class of subsets of the domain of a relation and the class of domains of its subsets, but is this hypothesis really needed? It should perhaps be pointed out that if one only deals with sets, this question does not arise, because all sets are thin. The thinness requirement is only of concern when one wants to be able to deal also with proper classes. Unconditional rewrite rules containing free variables can only be added to the **GOEDEL** program if the result in question holds for arbitrary classes. If one merely wishes to add an unconditional rewrite rule for a result that holds only for sets, the standard procedure is to make use of Gödel's algorithm (that is, the **class** rules) to eliminate the set-variable altogether. For the theorem under consideration, the specialization to sets does hold, and the standard process of eliminating set variables in this case yields the elegant result that the relation **inverse[S]** commutes with the function **IMAGE[FIRST]**, a fact already available in the **GOEDEL** program.

```
In[25]:= commute[inverse[S], IMAGE[FIRST]]

Out[25]= True
```

Note that the process of eliminating set variables using Gödel's algorithm often yields concise statements about proper classes. Both **inverse[S]** and **IMAGE[FIRST]**, for instance, are proper classes. In order to be able to take advantage of such results, one needs to have rewrite rules that apply to proper classes as well as rules about sets. If the conditional rewrite rules derived in this notebook could be replaced with unconditional rewrite rules, it would speed up the application of these rules. In the present case it does in fact seeem plausible that one could dispense with the **thinness** requirement. For example, the result in question holds for all cartesian products, whether or not they are thin:

```
In[26]:=  (equal[image[IMAGE[FIRST], P[x]], P[domain[x]]] /. x → cart[u, v]) // assert

Out[26]=  True
```

It is conceivable, for example, that some suitable global form of the axiom of choice might imply a more general theorem. In the **GOEDEL** program, as in our **Otter** work, neither the axiom of choice nor the axiom of regularity is routinely assumed, but in both cases, there are flags that can be set if one wishes to investigate consequences of these axioms.