cliques of a thin relation

Johan G. F. Belinfante
2011 October 4

```
In[1]:= SetDirectory["l:"]; << goedel.11oct01a

:Package Title: goedel.11oct01a                2011 October 1 at 4:20 p.m.

Loading takes about thirteen minutes, half that time due to builtin pauses.

It is now:  2011 Oct 4 at 11:44

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2011 Oct 4 at 11:56
```

summary

Every clique of a thin relation is a set. This fact is derived by introducing a temporary variable and then eliminating it using a combination of **reify** and **case**.

introduction

For convenience to the reader, some basic definitions and facts are reviewed in this section, but nothing new is derived here. The advantage of using the recently introduced **case** constructor is illustrated by a simple example here. In the final section, an attempt is made to demystify this by considering more general types of statements.

Recall that the **vertical section** of **x** at a set **u** is the image under **x** of the singleton of **u**.

```
In[2]:= class[v, and[member[u, V], member[pair[u, v], x]]]

Out[2]= image[x, set[u]]
```

The function **VERTSECT[x]** takes a set **u** to the vertical section **v = image[x, {u}]** whenever the latter is a set.

```
In[3]:= member[pair[u, v], VERTSECT[x]]

Out[3]= and[equal[v, image[x, set[u]]], member[u, V], member[v, V]]
```

A relation **x** is **thin** if every vertical section is a set. The condition that **x** is thin can be stated as follows:

```
In[4]:= assert[forall[y, member[image[x, set[y]], V]]] // Timing

Out[4]= {0.796 Second, equal[V, domain[VERTSECT[x]]]}
```

It has been observed that eliminating set variables can often be significantly speeded up by making use of **reify**. For any class expression **f[x]** that may involve a set variable **x**, the relation **reify[x, f[x]]** is the class **{ pair[x, y]: pair[x, y] ∈ f[x]}**. The special rewrite rules for **reify** in the **GOEDEL** program generally execute much faster than the more general **class** rules, probably because they do not affect terms that do not involve the variable **x**.

Although reification can only be used to eliminate set variables in class expressions, one can eliminate set variables in statements by using **case** to convert statements to classes. For any statement **p**, the class **case[p]** is equal to **V** if **p** is true, and is empty if **p** is false. The particular instance of quantification considered above is equivalent to the following, which executes about six times faster:

```
In[5]:= equal[V, domain[reify[w, case[member[image[x, set[w]], V]]]]] // Timing

Out[5]= {0.125 Second, equal[V, domain[VERTSECT[x]]]}
```

---

derivation

A class **y** is a **clique** of **x** if **y × y ⊂ x**. In this section it is shown that every clique of a thin relation is a set. The entire derivation is done all at once. A temporary variable **w** is introduced, and then eliminated using a combination of **reify** and **case**. Rewrite rules in the **GOEDEL** program can somethimes make up for missing proof steps. In this derivation, to speed up execution, one proof step was omitted, as indicated with **(* ⋯ *)**.

Theorem. Every clique of a thin relation is a set.

```
In[7]:= Map[equal[V, domain[reify[w, case[#]]]] &,
          Map[not, SubstTest[and, implies[and[p1, p2], p4], implies[p3, p5],
             (* implies[and[p4,p5],p6], *) not[implies[and[p1, p2, p3], p6]],
             {p1 → member[w, y], p2 -> subclass[cart[y, y], x],
              p3 -> equal[V, domain[VERTSECT[x]]], p4 → subclass[y, image[x, set[w]]],
              p5 → member[image[x, set[w]], V], p6 → member[y, V]}]]] // Reverse

Out[7]= or[member[y, V], not[equal[V, domain[VERTSECT[x]]]], not[subclass[cart[y, y], x]]] == True

In[8]:= or[member[y_, V], not[equal[V, domain[VERTSECT[x_]]]],
          not[subclass[cart[y_, y_], x_]]] := True
```

final remarks

In this final section, an attempt is made to show the equivalence of quantification with the use of **reify** and **case**, at least for certain special types of statements. The following temporary rewrite rule is introduced here to automatically convert certain types of **class** expressions to **reify** expressions. It should be noted thay making this rule permanent would not be a good idea because the **reify** rules currently in the **GOEDEL** program are not complete, whereas Gödel proved that his **class** rules are complete.

```
In[9]:= class[pair[x_, y_], member[y_, z_]] := reify[x, z]
```

The following computations show the equivalence of quantification with a certain expression involving **reify** and **case** for membership statements.

```
In[10]:= assert[forall[x, member[f[x], g[x]]]] // Timing

Out[10]= {0.438 Second,
          equal[V, fix[composite[inverse[reify[x, g[x]]], VERTSECT[reify[x, f[x]]]]]]}

In[11]:= equal[V, domain[reify[x, case[member[f[x], g[x]]]]]] // Timing

Out[11]= {0.109 Second,
          equal[V, fix[composite[inverse[reify[x, g[x]]], VERTSECT[reify[x, f[x]]]]]]}
```

A similar computation can be done for equality statements. Note that the use of **reify** is much less effective for equality statements.

```
In[12]:= assert[forall[x, equal[f[x], g[x]]]] // Timing

Out[12]= {0.313 Second, equal[composite[Id, reify[x, f[x]]], composite[Id, reify[x, g[x]]]]}

In[13]:= equal[V, domain[reify[x, case[equal[f[x], g[x]]]]]] // Timing

Out[13]= {0.859 Second, equal[composite[Id, reify[x, f[x]]], composite[Id, reify[x, g[x]]]]}
```