cliques of a thin relation

Johan G. F. Belinfante
2011 October 4

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 \texttt{reify} and \texttt{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 \texttt{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 \texttt{vertical section} of $x$ at a set $u$ is the image under $x$ of the singleton of $u$.

$$\texttt{In[2]:=}\quad \texttt{class[v, and[member[u, V], member[pair[u, v], x]]]}$$

$$\texttt{Out[2]=}\quad \texttt{image[x, set[u]]}$$

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

$$\texttt{In[3]:=}\quad \texttt{member[pair[u, v], VERTSECT[x]]}$$

$$\texttt{Out[3]=}\quad \texttt{and[equal[v, image[x, set[u]]], member[u, V], member[v, V]]}$$

A relation $x$ is \texttt{thin} if every vertical section is a set. The condition that $x$ is thin can be stated as follows:
It has been observed that eliminating set variables can often be significantly speeded up by making use of \texttt{reify}. For any class expression \( f[x] \) that may involve a set variable \( x \), the relation \( \texttt{reify}(x, f[x]) \) is the class \( \{ \text{pair}(x, y) \colon \text{pair}(x, y) \in f[x] \} \). The special rewrite rules for \texttt{reify} in the \texttt{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 \texttt{case} to convert statements to classes. For any statement \( p \), the class \( \texttt{case}(p) \) 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:

\begin{verbatim}
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]]]}
\end{verbatim}

### derivation

A class \( y \) is a \textbf{clique} of \( x \) if \( y \times y \subseteq x \). 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 \texttt{reify} and \texttt{case}. Rewrite rules in the \texttt{GOEDEL} program can sometimes make up for missing proof steps. In this derivation, to speed up execution, one proof step was omitted, as indicated with \( (*) \ldots (*) \).

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

\begin{verbatim}
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 \rightarrow member[w, y], p2 \rightarrow subclass[cart[y, y], x],
    p3 \rightarrow equal[V, domain[VERTSECT[x]]], p4 \rightarrow subclass[y, image[x, set[w]]],
    p5 \rightarrow member[image[x, set[w]], V], p6 \rightarrow 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
\end{verbatim}
final remarks

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

\begin{verbatim}
In[9]:= class[pair[x_, y_], member[y_, z_]] := reify[x, z]
\end{verbatim}

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

\begin{verbatim}
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]]]]]]
\end{verbatim}

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

\begin{verbatim}
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]]]]}
\end{verbatim}