HULL[EQV]

Johan G. F. Belinfante
2003 November 6

summary

A formula for HULL[EQV] is derived, which amounts to the statement that the smallest equivalence relation that contains a given relation $x$ is the transitive closure of the union of $x$ and $\text{inverse}[x]$.

introduction

The starting point is an equality substitution rule for the constructor $\text{trv}$.

```
In[2]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p2, p4], implies[p3, p5], implies[and[p4, p5], p6], not[implies[p1, p6]], {p1 -> equal[x, y], p2 -> subclass[x, y], p3 -> subclass[y, x], p4 -> subclass[trv[x], trv[y]], p5 -> subclass[trv[y], trv[x]], p6 -> equal[trv[x], trv[y]]}]]

Out[2]= or[equal[trv[x], trv[y]], not[equal[x, y]]] = True

In[3]:= or[equal[trv[x_], trv[y_]], not[equal[x_, y_]]] := True
```

The substitution rule is used to show that transitive closures of symmetric relations are symmetric:

```
In[4]:= SubstTest[implies, equal[x, y], equal[trv[x], trv[y]], y -> inverse[x]]

Out[4]= or[equal[inverse[trv[x]], trv[x]], not[equal[x, inverse[x]]]] = True

In[5]:= or[equal[inverse[trv[x_]], trv[x_]], not[equal[x_, inverse[x_]]]] := True
```

This is applies to the case of a union of a class and its inverse.

```
In[6]:= SubstTest[implies, equal[y, inverse[y]], equal[inverse[trv[y]], trv[y]], y -> union[composite[Id, x], inverse[x]]]

Out[6]= equal[inverse[trv[union[composite[Id, x], inverse[x]]]], trv[union[composite[Id, x], inverse[x]]]] = True
```
The \texttt{trv} constructor only looks at the relational part of its argument. This enables one to obtain a cleaner result:

\begin{verbatim}
In[8]:= equal[inverse[trv[union[x, inverse[x]]]], trv[union[x, inverse[x]]]] // AssertTest
Out[8]= equal[inverse[trv[union[x, inverse[x]]]], trv[union[x, inverse[x]]]] := True
\end{verbatim}

Restatement:

\begin{verbatim}
In[10]:= EQUIVALENCE[trv[union[x, inverse[x]]]]
Out[10]= True
\end{verbatim}

\section*{Preliminary Ideas}

What is sought is a variable-free version of the above results, using the functions \texttt{HULL[SYM]} and \texttt{HULL[TRV]}. To begin with, note that the function that takes \texttt{x} to \texttt{union[x, inverse[x]]} is the following:

\begin{verbatim}
In[11]:= lambda[x, union[x, inverse[x]]]
\end{verbatim}

The results are prettier when one replaces \texttt{IMAGE[SWAP]} with \texttt{INVERSE}. The fixed point set then is the set of symmetric relations:

\begin{verbatim}
In[12]:= composite[CUP, id[INVERSE], inverse[FIRST]] // fix
Out[12]= SYM
\end{verbatim}

A normalization result is derived:

\begin{verbatim}
In[13]:= composite[CUP, id[INVERSE], inverse[FIRST]] // VSNormality // Reverse
          composite[inverse[S], CUP, id[IMAGE[SWAP]], inverse[FIRST]],
          id[P[cart[V, V]]]] ==
          composite[CUP, id[INVERSE], inverse[FIRST]]
\end{verbatim}

With this place, one readily derives the idempotence of this function:

\begin{verbatim}
In[15]:= composite[CUP, id[INVERSE], inverse[FIRST],
          CUP, id[INVERSE], inverse[FIRST]] // VSNormality
Out[15]= composite[CUP, id[INVERSE], inverse[FIRST], CUP, id[INVERSE], inverse[FIRST]] ==
          composite[CUP, id[INVERSE], inverse[FIRST]]
\end{verbatim}

Restatement:

\begin{verbatim}
In[17]:= idempotent[x_] := equal[composite[x, x], x]
\end{verbatim}
In[18]:= composite[CUP, id[INVERSE], inverse[FIRST]] // idempotent
Out[18]= True

To show that this function is HULL[SYM], one needs to know that it subcommutes with S. This is readily established:

In[19]:= Map[equal[0, #] &,
dif[composite[CUP, id[INVERSE], inverse[FIRST], inverse[IMAGE[SWAP]]], S] //
RelnRenormality]

In[20]:= subclass[composite[CUP, id[INVERSE], inverse[FIRST], inverse[IMAGE[SWAP]]], S] := True

In[21]:= subclass[composite[CUP, id[INVERSE], inverse[FIRST], S],
    composite[S, CUP, id[INVERSE], inverse[FIRST]]] // AssertTest
Out[21]= subclass[composite[CUP, id[INVERSE], inverse[FIRST], S],
    composite[S, CUP, id[INVERSE], inverse[FIRST]]] = True

In[22]:= % /. Equal -> SetDelayed

Restatement:

In[23]:= subcommute[composite[CUP, id[INVERSE], inverse[FIRST]], S]
Out[23]= True

The theorem that characterizes HULL is applied to finish the job:

In[24]:= SubstTest[implies, and[FUNCTION[x], idempotent[x], subcommute[x, S], subclass[x, S]],
    equal[x, HULL[fix[x]]], x -> composite[CUP, id[INVERSE], inverse[FIRST]]]
Out[24]= equal[composite[CUP, id[INVERSE], inverse[FIRST]], HULL[SYM]] = True

In[25]:= composite[CUP, id[INVERSE], inverse[FIRST]] := HULL[SYM]

Although it is not needed here, it may be of interest to note that there is a similar formula with CAP in place of CUP.

In[26]:= Assoc[composite[CAP, id[IMAGE[SWAP]]], inverse[FIRST], id[P[cart[V, V]]]]
Out[26]= composite[CAP, id[INVERSE], inverse[FIRST]] = composite[CORE[SYM], id[P[cart[V, V]]]]

In[27]:= composite[CAP, id[INVERSE], inverse[FIRST]] := composite[CORE[SYM], id[P[cart[V, V]]]]

---

**a tricky business**

The connection between HULL[TRV] and trv[x] is via this formula:

In[28]:= U[image[HULL[TRV], P[x]]]
Out[28]= trv[x]

Note that the function that takes x to trv[x] is not HULL[TRV] itself, but something related to it in a simple fashion:
The formula for HULL[EQV] now follows immediately:
In[41]:= SubstTest[implies, invariant[HULL[x], fix[HULL[y]]],
equal[composite[HULL[x], HULL[y]], HULL[intersection[fix[HULL[x]], fix[HULL[y]]]]],
{x -> TRV, y -> SYM}]

Out[41]= equal[composite[HULL[TRV], HULL[SYM]], HULL[EQV]] = True

In[42]:= composite[HULL[TRV], HULL[SYM]] := HULL[EQV]

Corollary:

In[43]:= ImageComp[HULL[TRV], HULL[SYM], V] // Reverse

Out[43]= image[HULL[TRV], SYM] = EQV

In[44]:= image[HULL[TRV], SYM] := EQV