cliques families

Johan G. F. Belinfante
2010 August 14

In[1]:= SetDirectory["l:"]; << goedel.10aug12a; << tools.m

:Package Title: goedel.10aug12a 2010 August 12 at 1:00 p.m.
It is now: 2010 Aug 14 at 7:7
Loading Simplification Rules

TOOLS.M Revised 2010 August 11

weightlimit = 40

summary

Rewrite rules for cliques families are derived, generalizing similar formulas discovered in 2003 for the special case of function families.

function families

A curious formula for function families was derived 2003 June 5. A function family is a relation whose vertical sections are all functions. The GOEDEL program does not retain a separate predicate for this because it can transform this to a statement involving existing concepts:

In[2]:= assert[forall[y, FUNCTION[image[x, set[y]]]]]

Out[2]= and[FUNCTION[rotate[inverse[x]]], subclass[range[x], cart[V, V]]]

Function families were also studied using William McCune’s automated reasoning program Otter. A different definition of function family was used in the Otter work, which the GOEDEL program rewrites using this rule:

In[3]:= range[complement[composite[E, complement[x]]]]

Out[3]= domain[LB[x]]

It was shown in 2003 that the two definitions coincide, resulting in the following rewrite rule:

In[4]:= subclass[domain[LB[x]], FUNS]

Out[4]= and[FUNCTION[rotate[inverse[x]]], subclass[range[x], cart[V, V]]]

A similar rule holds with LB replaced by UB.
Two new rules of a similar nature can be derived.

Theorem.

In[6]:= SubstTest[subclass, image(INVERSE, u], image(INVERSE, v],
{u \rightarrow \text{domain}[UB[x]], v \rightarrow \text{image}[INVERSE, FUNS]}]//Reverse


In[7]:= subclass[image[INVERSE, domain[UB[x_]]], FUNS] := FUNCTION[rotate[function[composite[x, SWAP]]]]

Corollary.

In[8]:= SubstTest[subclass, image(INVERSE, domain[UB[t]]], FUNS, t \rightarrow \text{inverse}[x]]//Reverse

Out[8]= subclass[image[INVERSE, domain[LB[x]]], FUNS] = FUNCTION[rotate[function[composite[\text{inverse}[x], SWAP]]]]

In[9]:= subclass[image[INVERSE, domain[LB[x_]]], FUNS] := FUNCTION[rotate[function[composite[\text{inverse}[x], SWAP]]]]

\section*{cliques families}

In this notebook the more general concept of a \textit{cliques family} is studied. The statement that every vertical section of \( x \) is a clique for \( y \) is transformed by the \textsc{goedel} program as follows:

In[10]:= assert[\forall t, \text{subclass}[	ext{P[}\text{image}[x, \text{set}[t]], \text{cliques}[y]]]

Out[10]= subclass[\text{composite}[x, \text{inverse}[x]], y]

The following theorem yields a rewrite rule that transforms an equivalent statement, generalizing the rewrite rule discovered in 2003.

Theorem.

In[11]:= SubstTest[assert, \forall t, \text{subclass}[	ext{P[}\text{image}[x, \text{set}[t]], z], z \rightarrow \text{cliques}[y]]

Out[11]= subclass[\text{domain}[LB[x]], \text{cliques}[y]] = subclass[\text{composite}[x, \text{inverse}[x]], y]

In[12]:= subclass[\text{domain}[LB[x_]], \text{cliques}[y_]] := subclass[\text{composite}[x, \text{inverse}[x]], y]

Corollary.

In[13]:= SubstTest[subclass, domain[LB[t]], \text{cliques}[y], t \rightarrow \text{inverse}[x]]//Reverse

Out[13]= subclass[\text{domain}[UB[x]], \text{cliques}[y]] = subclass[\text{composite}[\text{inverse}[x], x], y]
special cases

The following special cases are of interest.

Theorem.

In[17]:= SubstTest[subclass, domain[LB[x]], cliques[y], y\rightarrow Id] // Reverse
Out[17]= subclass[domain[LB[x]], union[range[SINGLETON], set[0]]] := FUNCTION[composite[Id, x]]

In[18]:= subclass[domain[LB[x]], union[range[SINGLETON], set[0]]] := FUNCTION[composite[Id, x]]

Another case of interest is this:

Theorem.

In[19]:= SubstTest[subclass, domain[UB[x]], cliques[y], y\rightarrow Id] // Reverse
Out[19]= subclass[domain[UB[x]], union[range[SINGLETON], set[0]]] := FUNCTION[inverse[x]]

In[20]:= subclass[domain[UB[x]], union[range[SINGLETON], set[0]]] := FUNCTION[inverse[x]]

Corollary.

In[28]:= equal[domain[UB[x]], set[0]] // AssertTest
Out[28]= equal[domain[UB[x]], set[0]] := equal[0, domain[x]]

In[29]:= equal[domain[UB[x]], set[0]] := equal[0, domain[x]]

Corollary.

In[31]:= SubstTest[equal, domain[UB[t]], set[0], t \rightarrow inverse[x]] // Reverse
Out[31]= equal[domain[LB[x]], set[0]] := equal[0, domain[x]]

In[32]:= equal[domain[LB[x]], set[0]] := equal[0, domain[x]]