HULL[range[CLIQUES]]

Johan G. F. Belinfante
2006 October 14

```
In[1]:= SetDirectory["l:"]; << goedel86.12c; << tools.m

        :Package Title: goedel86.12c          2006 October 12 at 10:55 p.m.

        It is now:  2006 Oct 14 at 3:23

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 October 12

        weightlimit = 40
```

summary

A formula for **HULL[range[CLIQUES]]** is derived.

derivation

Theorem. Every member of **x** is a clique of the reflexive symmetric relation **composite[inverse[E], id[x], E]**.

```
In[2]:= Map[empty[class[pair[s, t], not[#]]] &, SubstTest[implies,
           and[member[pair[s, u], composite[Id, z]], member[pair[u, t], composite[Id, y]]],
           member[pair[s, t], composite[y, z]], y → inverse[z]] /. z → composite[id[x], E]]

Out[2]= or[not[member[u, x]], subclass[cart[u, u], composite[inverse[E], id[x], E]]] == True

In[3]:= (% /. {u → u_, x → x_}) /. Equal → SetDelayed
```

The variable **u** is easy to eliminate:

```
In[4]:= Map[equal[V, #] &,
         complement[dif[x, cliques[composite[inverse[E], id[x], E]]]] // Normality]

Out[4]= subclass[x, cliques[composite[inverse[E], id[x], E]]] == True

In[5]:= subclass[x_, cliques[composite[inverse[E], id[x_], E]]] := True
```

Observation:

```
In[6]:= equal[dif[x, cliques[composite[inverse[E], id[x], E]]], 0]

Out[6]= True
```

This justifies a rewrite rule:

```
In[7]:= intersection[x_, complement[cliques[composite[inverse[E], id[x_], E]]]] := 0
```

One can now use **reify** to eliminate the variable **x**.

```
In[8]:= Map[empty, SubstTest[reify, x, intersection[x,
            complement[cliques[composite[inverse[E], f[x], E]]]], f → id]] // Reverse

Out[8]= subclass[composite[CLIQUES, BIGCUP, IMAGE[CART], IMAGE[DUP]], S] == True

In[9]:= % /. Equal → SetDelayed
```

This is one of the properties that characterize **HULL** functions. Another property is idempotence. This property is recognized by existing rewrite rules. From it one can deduce this corollary, which will be needed shortly.

```
In[10]:= SubstTest[implies, and[FUNCTION[x], idempotent[x]], equal[fix[x], range[x]],
          x -> composite[CLIQUES, BIGCUP, IMAGE[CART], IMAGE[DUP]]]

Out[10]= equal[fix[composite[CLIQUES, BIGCUP, IMAGE[CART], IMAGE[DUP]]], range[CLIQUES]] == True

In[11]:= fix[composite[CLIQUES, BIGCUP, IMAGE[CART], IMAGE[DUP]]] := range[CLIQUES]
```

Lemma. Since **CLIQUES** and **BIGCUP** both subcommute with **S**, so does their composite.

```
In[12]:= SubstTest[implies, and[subcommute[x, S], subcommute[y, S]],
          subcommute[composite[x, y], S],
          {x → CLIQUES, y → BIGCUP}]

Out[12]= subclass[composite[CLIQUES, inverse[POWER], S], composite[S, CLIQUES, BIGCUP]] == True

In[13]:= % /. Equal → SetDelayed
```

Corollary.

```
In[14]:= SubstTest[implies, and[subcommute[x, S], subcommute[y, S]],
          subcommute[composite[x, y], S],
          {x → composite[CLIQUES, BIGCUP], y → composite[IMAGE[CART], IMAGE[DUP]]}]

Out[14]= subclass[composite[CLIQUES, BIGCUP, id[P[image[CART, Id]]], S, IMAGE[CART],
            IMAGE[DUP]], composite[S, CLIQUES, BIGCUP, IMAGE[CART], IMAGE[DUP]]] == True

In[15]:= % /. Equal → SetDelayed
```

Theorem. (This is an application of a characterization of **HULL** functions as idempotent functions which are contained in **S** and which subcommute with **S**.)

```
In[16]:= SubstTest[implies, and[idempotent[x], FUNCTION[x], subclass[x, S], subcommute[x, S]],
          equal[x, HULL[fix[x]]], x -> composite[CLIQUES, BIGCUP, IMAGE[CART], IMAGE[DUP]]]

Out[16]= equal[composite[CLIQUES, BIGCUP, IMAGE[CART], IMAGE[DUP]], HULL[range[CLIQUES]]] == True

In[17]:= composite[CLIQUES, BIGCUP, IMAGE[CART], IMAGE[DUP]] := HULL[range[CLIQUES]]
```

Aclosure property

Lemma.

```
In[18]:= SubstTest[subclass, fix[composite[x, y]], range[x],
           {x → CLIQUES, y -> composite[BIGCUP, IMAGE[CART], IMAGE[DUP]]}]

Out[18]= subclass[fix[HULL[range[CLIQUES]]], range[CLIQUES]] == True

In[19]:= % /. Equal → SetDelayed
```

Theorem.

```
In[20]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> fix[HULL[range[CLIQUES]]], v -> range[CLIQUES]}]

Out[20]= True == equal[fix[HULL[range[CLIQUES]]], range[CLIQUES]]

In[21]:= fix[HULL[range[CLIQUES]]] := range[CLIQUES]
```

Corollary. The intersection of any set of cliques is a clique.

```
In[22]:= SubstTest[Aclosure, fix[HULL[x]], x → range[CLIQUES]]

Out[22]= Aclosure[range[CLIQUES]] == range[CLIQUES]

In[23]:= Aclosure[range[CLIQUES]] := range[CLIQUES]
```

other properties of range[CLIQUES]

Theorem.

```
In[24]:= IminComp[CLIQUES, composite[BIGCUP, IMAGE[CART], IMAGE[DUP]], V]

Out[24]= image[inverse[S], range[CLIQUES]] == V

In[25]:= image[inverse[S], range[CLIQUES]] := V
```

Theorem.

```
In[26]:= IminComp[S, E, range[CLIQUES]]

Out[26]= U[range[CLIQUES]] == V

In[27]:= U[range[CLIQUES]] := V
```

Corollary.

```
In[28]:= SubstTest[member, U[x], V, x -> range[CLIQUES]] // Reverse
Out[28]= member[range[CLIQUES], V] == False

In[29]:= % /. Equal → SetDelayed

In[30]:= member[range[CLIQUES], x] // AssertTest
Out[30]= member[range[CLIQUES], x] == False

In[31]:= member[range[CLIQUES], x_] := False
```

More generally: