monotone images of cliques

Johan G. F. Belinfante
2006 May 1

```
In[1]:= SetDirectory["l:"]; << goedel80.30a; << tools.m

        :Package Title: goedel80.30a          2006 April 30 at 11:40 a.m.

        It is now:  2006 May 1 at 21:57

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 March 7

        weightlimit = 40
```

summary

The monotone image of a clique is a clique.

derivation

Lemma

```
In[2]:= SubstTest[implies, subclass[v, y],
         subclass[image[u, v], image[u, y]], {u → cross[x, x], v -> cart[w, w]}]

Out[2]= or[not[subclass[cart[w, w], y]],
          subclass[cart[image[x, w], image[x, w]], composite[x, y, inverse[x]]]] == True

In[3]:= (% /. {w → w_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[4]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
          implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
          {p1 → subclass[cart[w, w], y], p2 → monotone[x, y, z],
           p3 -> subclass[cart[image[x, w], image[x, w]], composite[x, y, inverse[x]]],
           p4 -> subclass[cart[image[x, w], image[x, w]], z]}]]

Out[4]= or[not[subclass[cart[w, w], y]], not[subclass[composite[x, y, inverse[x]], z]],
          subclass[cart[image[x, w], image[x, w]], z]] == True

In[5]:= or[not[subclass[cart[w_, w_], y_]], not[subclass[composite[x_, y_, inverse[x_]], z_]],
          subclass[cart[image[x_, w_], image[x_, w_]], z_]] := True
```

Removing the variable **w** is a bit tricky. The following observation suggested a course of action.

```
In[6]:= Map[equal[V, #] &, SubstTest[class, w, or[not[subclass[cart[w, w], y]],
          not[subclass[t, z]], subclass[cart[image[u, w], image[u, w]], z]],
         t → composite[u, y, inverse[u]]]] // Reverse

Out[6]= or[not[subclass[composite[u, y, inverse[u]], z]], subclass[cliques[y],
         cliques[complement[composite[inverse[u], complement[z], u]]]]] == True
```

The above observation suggested the following. This only works when **x** is thin. The main application is to the case of functions, but substituting **funpart** for **thinpart** did not yield a useful result. The problem is that for functions, another rewrite rule kicks in, preempting the desired simpiflication being made. The course taken here is to specialize to the case of functions at a later stage.

```
In[7]:= cliques[complement[composite[inverse[thinpart[x]], complement[y], thinpart[x]]]] //
        Normality

Out[7]= cliques[complement[composite[inverse[thinpart[x]], complement[y], thinpart[x]]]] ==
        image[inverse[IMAGE[thinpart[x]]], cliques[y]]

In[8]:= cliques[complement[composite[inverse[thinpart[x_]], complement[y_], thinpart[x_]]]] :=
        image[inverse[IMAGE[thinpart[x]]], cliques[y]]
```

The earlier observation now yields a useful result:

```
In[9]:= (Map[equal[V, #] &, SubstTest[class, w, or[not[subclass[cart[w, w], y]],
           not[subclass[t, z]], subclass[cart[image[u, w], image[u, w]], z]],
          t → composite[u, y, inverse[u]]]] // Reverse) /. u → thinpart[x]

Out[9]= or[not[subclass[composite[thinpart[x], y, inverse[thinpart[x]]], z]],
         subclass[image[IMAGE[thinpart[x]], cliques[y]], cliques[z]]] == True

In[10]:= or[not[subclass[composite[thinpart[x_], y_, inverse[thinpart[x_]]], z_]],
          subclass[image[IMAGE[thinpart[x_]], cliques[y_]], cliques[z_]]] := True
```

The specialization to functions can now be done:

```
In[11]:= SubstTest[implies, monotone[thinpart[w], y, z],
         conduct[IMAGE[thinpart[w]], cliques[y], cliques[z]], w → funpart[x]]

Out[11]= or[not[subclass[composite[funpart[x], y, inverse[funpart[x]]], z]],
          subclass[image[IMAGE[funpart[x]], cliques[y]], cliques[z]]] == True

In[12]:= or[not[subclass[composite[funpart[x_], y_, inverse[funpart[x_]]], z_]],
          subclass[image[IMAGE[funpart[x_]], cliques[y_]], cliques[z_]]] := True
```

The **funpart** wrapper can be removed:

```
In[13]:= SubstTest[implies, and[equal[x, funpart[w]], monotone[x, y, z]],
         conduct[IMAGE[x], cliques[y], cliques[z]], w → x]

Out[13]= or[not[FUNCTION[x]], not[subclass[composite[x, y, inverse[x]], z]],
          subclass[image[IMAGE[x], cliques[y]], cliques[z]]] == True
```

```
In[14]:= or[not[FUNCTION[x_]], not[subclass[composite[x_, y_, inverse[x_]], z_]],
           subclass[image[IMAGE[x_], cliques[y_]], cliques[z_]]] := True
```

monotone functions preserve chains

Lemma.

```
In[15]:= SubstTest[implies, monotone[x, u, v],
          monotone[x, u, union[v, w]], {u → inverse[y], v → inverse[z]}]
```

```
Out[15]= or[not[subclass[composite[x, y, inverse[x]], z]],
           subclass[composite[x, inverse[y], inverse[x]], union[w, inverse[z]]]] == True
```

```
In[16]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[17]:= SubstTest[implies, and[FUNCTION[x], monotone[x, u, v]],
          conduct[IMAGE[x], cliques[u], cliques[v]],
          {u → union[y, inverse[y]], v → union[z, inverse[z]]}]
```

```
Out[17]= or[not[FUNCTION[x]], not[subclass[composite[x, y, inverse[x]], union[z, inverse[z]]]],
           not[subclass[composite[x, inverse[y], inverse[x]], union[z, inverse[z]]]],
           subclass[image[IMAGE[x], cliques[union[y, inverse[y]]]],
            cliques[union[z, inverse[z]]]]] == True
```

```
In[18]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

```
In[19]:= Map[not, SubstTest[and, implies[p2, p3], implies[p2, p4],
           implies[and[p1, p3, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 → FUNCTION[x], p2 → monotone[x, y, z], p3 → monotone[x, y, union[z, inverse[z]]],
            p4 → monotone[x, inverse[y], union[z, inverse[z]]], p5 →
             conduct[IMAGE[x], cliques[union[y, inverse[y]]], cliques[union[z, inverse[z]]]]}]]
```

```
Out[19]= or[not[FUNCTION[x]], not[subclass[composite[x, y, inverse[x]], z]],
           subclass[image[IMAGE[x], cliques[union[y, inverse[y]]]],
            cliques[union[z, inverse[z]]]]] == True
```

```
In[21]:= or[not[FUNCTION[x_]], not[subclass[composite[x_, y_, inverse[x_]], z_]],
           subclass[image[IMAGE[x_], cliques[union[inverse[y_], y_]]],
            cliques[union[inverse[z_], z_]]]] := True
```