range[CLIQUES]

Johan G. F. Belinfante
2006 October 11

```
In[1]:= SetDirectory["l:"]; << goedel86.10b; << tools.m

        :Package Title: goedel86.10b          2006 October 10 at 10:35 p.m.

        It is now:  2006 Oct 11 at 22:41

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 October 10

        weightlimit = 40
```

summary

A formula for the **Uclosure** of the class **range[CLIQUES]** is derived by using upper and lower bounds. Counterexamples are provided to whow that these upper and lower bounds a proper.

a question

A class **w** is a clique of a relation **x** if every pair of elements of **w** are related by **x**, that is, if the cartesian square of **w** is contained in **x**. The class of all cliques of **x** is

```
In[2]:= class[w, subclass[cart[w, w], x]]

Out[2]= cliques[x]
```

The following will be needed later:

```
In[3]:= Map[implies[member[x, y], #] &,
          SubstTest[member, x, image[inverse[CLIQUES], z], z → range[CLIQUES]]] // Reverse

Out[3]= or[member[cliques[x], range[CLIQUES]], not[member[x, y]]] == True

In[4]:= or[member[cliques[x_], range[CLIQUES]], not[member[x_, y_]]] := True
```

The class of all sets of the form **cliques[x]** is:

```
In[5]:= class[y, exists[x, equal[y, cliques[x]]]]

Out[5]= range[CLIQUES]
```

How can one characterize the class **range[CLIQUES]** and what are its properties? When is a class **y** equal to **cliques[x]** for some relation?

a lower bound and an upper bound

One can get lower bounds on **range[CLIQUES]** by looking at examples. For instance,

```
In[6]:= Map[subclass[#, range[CLIQUES]] &, ImageComp[CLIQUES, CART, V]]

Out[6]= subclass[range[POWER], range[CLIQUES]] == True

In[7]:= subclass[range[POWER], range[CLIQUES]] := True
```

An upper bound follows from the observation that any subset of a clique is a clique. That is, the class cliques[x] is hereditary. The class of all hereditary sets therefore provides an upper bound:

```
In[8]:= Map[equal[0, composite[Id, complement[#]]] &,
          SubstTest[class, pair[x, y], implies[member[pair[x, y], w], member[y, z]],
           {w → CLIQUES, z → fix[IMAGE[inverse[S]]]}]] // Reverse

Out[8]= subclass[range[CLIQUES], fix[IMAGE[inverse[S]]]] == True

In[9]:= subclass[range[CLIQUES], fix[IMAGE[inverse[S]]]] := True
```

Counterexamples will be provided below to show that neither of these inclusions can be sharpened to an equation. Both the upper and lower bounds have the same **Uclosure**, and therefore can be used to compute the **Uclosure** of **range[CLIQUES]**.

Uclosure[range[CLIQUES]]

The **Uclosure** of a class **x** is the class of all unions of subsets of **x**.

```
In[10]:= class[u, exists[s, and[subclass[s, x], equal[u, U[s]]]]]

Out[10]= Uclosure[x]
```

The **Uclosure** functor is monotone. From the lower bound for **range[CLIQUES]** one derives a lower bound on its **Uclosure**.

```
In[11]:= SubstTest[implies, subclass[u, v],
           subclass[Uclosure[u], Uclosure[v]], {u → range[POWER], v → range[CLIQUES]}]

Out[11]= subclass[fix[IMAGE[inverse[S]]], Uclosure[range[CLIQUES]]] == True

In[12]:= % /. Equal → SetDelayed
```

Similarly, from the upper bound, one derives an upper bound on the **Uclosure**.

```
In[13]:= SubstTest[implies, subclass[u, v], subclass[Uclosure[u], Uclosure[v]],
          {u → range[CLIQUES], v → fix[IMAGE[inverse[S]]]}]

Out[13]= subclass[Uclosure[range[CLIQUES]], fix[IMAGE[inverse[S]]]] == True

In[14]:= % /. Equal → SetDelayed
```

Combining these inclusions yields an equation:

```
In[15]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> Uclosure[range[CLIQUES]], v -> fix[IMAGE[inverse[S]]]}]

Out[15]= True == equal[fix[IMAGE[inverse[S]]], Uclosure[range[CLIQUES]]]

In[16]:= Uclosure[range[CLIQUES]] := fix[IMAGE[inverse[S]]]
```

Corollary.

```
In[17]:= SubstTest[CORE, Uclosure[x], x → range[CLIQUES]] // Reverse

Out[17]= CORE[range[CLIQUES]] == CORE[fix[IMAGE[inverse[S]]]]

In[18]:= CORE[range[CLIQUES]] := CORE[fix[IMAGE[inverse[S]]]]
```

the lower bound is proper

A counterexample is presented in this section to show that the lower bound on **range[CLIQUES]** is proper. For the union of two cartesian squares, the cliques class is the union of two power classes.

```
In[19]:= cliques[union[cartsq[x], cartsq[y]]]

Out[19]= union[P[x], P[y]]
```

So all that on needs to show is that in general, the union of two power sets need not be a power set. A simple example is the union of the power set of **set[0]** and the power set of **set[set[0]]**. If this union were a power set, that power set would have to hold the union of these two sets, which it does not.

```
In[20]:= SubstTest[implies, member[x, V], member[cliques[x], range[CLIQUES]],
          x -> union[cartsq[set[0]], cartsq[set[set[0]]]]]

Out[20]= member[union[set[0], succ[set[set[0]]], range[CLIQUES]] == True

In[21]:= member[union[set[0], succ[set[set[0]]], range[CLIQUES]] := True

In[22]:= Map[not, SubstTest[implies, and[member[u, v], equal[v, w]], member[u, w],
            {u → union[set[0], succ[set[set[0]]], v -> range[CLIQUES], w → range[POWER]}]]

Out[22]= equal[range[CLIQUES], range[POWER]] == False

In[23]:= equal[range[CLIQUES], range[POWER]] := False
```

the upper bound is proper

In this section it is shown that there is no relation **x** for which **cliques[x]** is **image[inverse[S], omega]**, the set of all finite subsets of the set of natural numbers. If there were such a relation, it would satisfy:

```
In[24]:= SubstTest[implies, equal[u, v], equal[image[w, u], image[w, v]], {u → cliques[x],
           v → image[inverse[S], omega], w → composite[cross[inverse[E], inverse[E]], DUP]}]

Out[24]= or[equal[cart[omega, omega],
            composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]]],
           not[equal[cliques[x], image[inverse[S], omega]]]] == True

In[25]:= (% /. x → x_) /. Equal → SetDelayed
```

But that would imply that **cliques[x]** is equal to **P[omega]**, the set of all subsets of **omega**.

```
In[26]:= SubstTest[implies, equal[u, v], equal[cliques[u], cliques[v]], {u → cart[omega, omega],
           v -> composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]]}]

Out[26]= or[equal[cliques[x], P[omega]], not[equal[cart[omega, omega],
             composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]]]]] == True

In[27]:= (% /. x → x_) /. Equal → SetDelayed
```

Since the sets **image[inverse[S], omega]** and **P[omega]** are not equal, this is a contradiction.

```
In[28]:= SubstTest[and, implies[p1, p2], implies[p2, p3], p1,
           {p1 -> equal[cliques[x], image[inverse[S], omega]], p2 → equal[cart[omega, omega],
              composite[id[fix[x]], intersection[x, inverse[x]], id[fix[x]]]],
            p3 -> equal[cliques[x], P[omega]]}] // MapNotNot

Out[28]= equal[cliques[x], image[inverse[S], omega]] == False

In[29]:= equal[cliques[x_], image[inverse[S], omega]] := False

In[30]:= member[image[inverse[S], omega], range[CLIQUES]] // AssertTest

Out[30]= member[image[inverse[S], omega], range[CLIQUES]] == False

In[31]:= member[image[inverse[S], omega], range[CLIQUES]] := False

In[32]:= Map[not, SubstTest[implies, and[member[u, v], equal[v, w]], member[u, w],
           {u -> image[inverse[S], omega], v → fix[IMAGE[inverse[S]]], w -> range[CLIQUES]}]]

Out[32]= equal[fix[IMAGE[inverse[S]]], range[CLIQUES]] == False

In[33]:= equal[fix[IMAGE[inverse[S]]], range[CLIQUES]] := False
```

chain condition

In this section it is shown that the union of a chain of cliques is a clique. A class **x** is a chain of sets if the following holds:

```
In[34]:= assert[forall[u, v,
           implies[and[member[u, x], member[v, x]], or[subclass[u, v], subclass[v, u]]]]]

Out[34]= subclass[cart[x, x], union[S, inverse[S]]]
```

Lemma.

```
In[35]:= SubstTest[implies, and[member[x, y], subclass[y, w]], member[x, w], w → cliques[z]]

Out[35]= or[not[member[x, y]], not[subclass[y, cliques[z]]], subclass[cart[x, x], z]] == True

In[36]:= or[not[member[x_, y_]],
           not[subclass[y_, cliques[z_]]], subclass[cart[x_, x_], z_]] := True
```

Lemma.

```
In[37]:= SubstTest[implies, and[member[r, s], subclass[s, t]],
          member[r, t], {r → pair[u, v], s → cartsq[w]}]

Out[37]= or[member[pair[u, v], t], not[member[u, w]],
           not[member[v, w]], not[subclass[cart[w, w], t]]] == True

In[38]:= (% /. {u → u_, v → v_, w → w_, t → t_}) /. Equal → SetDelayed
```

Lemma. When **u** is contained in **v**, one has:

```
In[39]:= Map[not, SubstTest[and, implies[and[p0, p2], p6], implies[and[p0, p4], p7],
           implies[and[p1, p5], p8], not[implies[and[p0, p1, p2, p3, p4, p5], p9]],
           {p0 → subclass[x, cliques[y]], p1 → member[s, u], p2 → member[u, x], p3 → member[t, v],
            p4 -> member[v, x], p5 -> subclass[u, v], p6 → subclass[cartsq[u], y],
            p7 → subclass[cartsq[v], y], p8 → member[s, v], p9 → member[pair[s, t], y]}]]

Out[39]= or[member[pair[s, t], y], not[member[s, u]], not[member[t, v]], not[member[u, x]],
           not[member[v, x]], not[subclass[u, v]], not[subclass[x, cliques[y]]]] == True

In[40]:= (% /. {s → s_, t → t_, u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

A similar result holds when **v** is contained in **u**.

```
In[41]:= Map[not, SubstTest[and, implies[and[p0, p2], p6], implies[and[p0, p4], p7],
           implies[and[p3, p5], p8], not[implies[and[p0, p1, p2, p3, p4, p5], p9]],
           {p0 → subclass[x, cliques[y]], p1 → member[s, u], p2 → member[u, x], p3 → member[t, v],
            p4 -> member[v, x], p5 -> subclass[v, u], p6 → subclass[cartsq[u], y],
            p7 → subclass[cartsq[v], y], p8 → member[t, u], p9 → member[pair[s, t], y]}]]

Out[41]= or[member[pair[s, t], y], not[member[s, u]], not[member[t, v]], not[member[u, x]],
           not[member[v, x]], not[subclass[v, u]], not[subclass[x, cliques[y]]]] == True
```

```
In[42]:= (% /. {s → s_, t → t_, u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

To combine these two cases, one needs:

```
In[43]:= or[and[not[subclass[u, v]], not[subclass[v, u]]],
            member[pair[s, t], y], not[member[s, u]], not[member[t, v]], not[member[u, x]],
            not[member[v, x]], not[subclass[x, cliques[y]]]] // NotNotTest

Out[43]= or[and[not[subclass[u, v]], not[subclass[v, u]]],
            member[pair[s, t], y], not[member[s, u]], not[member[t, v]],
            not[member[u, x]], not[member[v, x]], not[subclass[x, cliques[y]]]] == True

In[44]:= (% /. {s → s_, t → t_, u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[45]:= SubstTest[implies, and[member[r, s], subclass[s, t]],
           member[r, t], {r → pair[u, v], s → cart[x, x], t → union[S, inverse[S]]}]

Out[45]= or[not[member[u, x]], not[member[v, x]],
            not[subclass[cart[x, x], union[S, inverse[S]]]],
            subclass[u, v], subclass[v, u]] == True

In[46]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

Lemma.

```
In[47]:= Map[not, SubstTest[and, implies[and[p1, p3], p4], not[implies[and[p1, p2, p3], p5]],
             {p1 → and[member[s, u], member[t, v], member[u, x], member[v, x]],
              p2 -> subclass[x, cliques[y]], p3 -> subclass[cart[x, x], union[S, inverse[S]]],
              p4 → or[subclass[u, v], subclass[v, u]], p5 → member[pair[s, t], y]}]]

Out[47]= or[member[pair[s, t], y], not[member[s, u]], not[member[t, v]],
            not[member[u, x]], not[member[v, x]], not[subclass[x, cliques[y]]],
            not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

In[48]:= (% /. {s → s_, t → t_, u → u_, v → v_, x → x_, y → y_}) /. Equal → SetDelayed
```

Eliminating the variables **u** and **v** yields:

```
In[49]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[u, v],
              or[member[pair[s, t], y], not[member[s, u]], not[member[t, v]], not[member[u, x]],
               not[member[v, x]], not[subclass[x, r]], not[subclass[cart[x, x], z]]],
              {r → cliques[y], z -> union[S, inverse[S]]}]] // Reverse

Out[49]= or[member[pair[s, t], y], not[member[s, U[x]]], not[member[t, U[x]]],
            not[subclass[x, cliques[y]]], not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

In[50]:= (% /. {s → s_, t → t_, x → x_, y → y_}) /. Equal → SetDelayed
```

Eliminating the variables **s** and **t** yields the main theorem: the union of a chain of cliques is a clique.

```
In[51]:= Map[empty[composite[Id, complement[#]]] &,
           SubstTest[class, pair[s, t], or[member[pair[s, t], y], not[member[s, U[x]]],
             not[member[t, U[x]]], not[subclass[x, w]], not[subclass[u, v]]],
            {u → cart[x, x], v → union[S, inverse[S]], w → cliques[y]}]] // Reverse

Out[51]= or[not[subclass[x, cliques[y]]],
           not[subclass[cart[x, x], union[S, inverse[S]]]], subclass[cart[U[x], U[x]], y]] == True

In[52]:= or[not[subclass[x_, cliques[y_]]], not[subclass[cart[x_, x_], union[S, inverse[S]]]],
           subclass[cart[U[x_], U[x_]], y_]] := True
```

---

variable-free formulation of the chain condition

```
In[53]:= Map[equal[V, #] &, SubstTest[class, t, implies[member[t, u], member[U[t], v]],
            {u -> intersection[P[cliques[x]], cliques[union[S, inverse[S]]]],
             v -> cliques[x]}]] // Reverse

Out[53]= subclass[image[BIGCUP, intersection[cliques[union[S, inverse[S]]], P[cliques[x]]]],
           cliques[x]] == True

In[54]:= (% /. x → x_) /. Equal → SetDelayed
```

The reverse inclusion holds for any class:

```
In[55]:= Map[equal[V, class[w, #]] &,
           SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
            {t → BIGCUP, u → set[set[w]], v -> intersection[cliques[union[S, inverse[S]]], P[x]]}]]

Out[55]= subclass[x, image[BIGCUP, intersection[cliques[union[S, inverse[S]]], P[x]]]] == True

In[56]:= subclass[x_, image[BIGCUP, intersection[cliques[union[S, inverse[S]]], P[x_]]]] := True
```

This yields an equation:

```
In[58]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[BIGCUP, intersection[cliques[union[S, inverse[S]]], P[cliques[x]]]],
            v -> cliques[x]}]

Out[58]= True == equal[cliques[x],
           image[BIGCUP, intersection[cliques[union[S, inverse[S]]], P[cliques[x]]]]]

In[60]:= image[BIGCUP, intersection[cliques[union[S, inverse[S]]], P[cliques[x_]]]] :=
           cliques[x]
```

The variable **x** can be eliminated using **reify**:

```
In[63]:= Map[VERTSECT, SubstTest[reify, x, image[BIGCUP,
             intersection[cliques[union[S, inverse[S]]], P[f[x]]]], f → cliques]] // Reverse

Out[63]= composite[IMAGE[BIGCUP],
           IMAGE[id[cliques[union[S, inverse[S]]]]], POWER, CLIQUES] == CLIQUES
```

```
In[64]:= composite[IMAGE[BIGCUP],
           IMAGE[id[cliques[union[S, inverse[S]]]]], POWER, CLIQUES] := CLIQUES
```

Lemma.

```
In[67]:= Map[equal[V, #] &, SubstTest[class, x,
            implies[equal[x, z], equal[image[BIGCUP, intersection[w, P[x]]], x]],
            {w -> cliques[union[S, inverse[S]]], z -> cliques[y]}]] // Reverse

Out[67]= or[member[cliques[y],
            fix[composite[IMAGE[BIGCUP], IMAGE[id[cliques[union[S, inverse[S]]]]], POWER]]],
           not[member[fix[y], V]]] == True
```

```
In[68]:= (% /. y → y_) /. Equal → SetDelayed
```

Theorem. Another variable-free formulation of the chain condition.

```
In[69]:= Map[equal[range[CLIQUES], image[CLIQUES, #]] &, SubstTest[class, y,
            or[member[cliques[y], z], not[member[fix[y], V]]], z -> fix[composite[
               IMAGE[BIGCUP], IMAGE[id[cliques[union[S, inverse[S]]]]], POWER]]] // Reverse

Out[69]= subclass[range[CLIQUES], fix[
            composite[IMAGE[BIGCUP], IMAGE[id[cliques[union[S, inverse[S]]]]], POWER]]] == True
```

```
In[70]:= subclass[range[CLIQUES], fix[
            composite[IMAGE[BIGCUP], IMAGE[id[cliques[union[S, inverse[S]]]]], POWER]]] := True
```

---

open questions

It has been shown that the class of cliques of a relation **x** has these properties: every subset of a clique is a clique, and the union of a chain of cliques is a clique. If a class **y** has these two properties, does there exist a relation x such that **y = cliques[x]** ? In other words, can the following inclusion be sharpened to an equation? If so, prove it, and if not, find a counterexample that shows that the inclusion is proper.

```
In[72]:= subclass[range[CLIQUES], intersection[fix[IMAGE[inverse[S]]],
           fix[composite[IMAGE[BIGCUP], IMAGE[id[cliques[union[S, inverse[S]]]]], POWER]]]

Out[72]= True
```

What other properties does the class **range[CLIQUES]** possess, and more generally what properties does **image[-CLIQUES, x]** have? A particularly interesting question is to characterize **image[CLIQUES, EQV]**, the class of cliques of equivalence relations.