# unions of chains of squares

Johan G. F. Belinfante
2011 December 2

```
In[1]:= SetDirectory["l:"]; << goedel.11nov30a
```

```
:Package Title: goedel.11nov30a              2011 November 30 at 12:50 noon

Loading takes about thirteen minutes, half that time due to builtin pauses.

It is now:  2011 Dec 2 at 13:9

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2011 Dec 2 at 13:22
```

## summary

It is shown that the class of unions of chains of cartesian squares of members of a class **x** is the class of cartesian squares of unions of a chains of members of **x**. The following is the simplest of several special cases derived 2007 September 8.

```
In[2]:= Uchains[image[CART, Id]]
```

```
Out[2]= image[CART, Id]
```

## inclusion in one direction

Lemma.

```
In[3]:= SubstTest[implies, and[member[t, P[y]], subclass[P[t], chains[S]]],
          member[U[t], Uchains[y]], t → image[FIX, x]] // Reverse
```

```
Out[3]= or[member[fix[U[x]], Uchains[y]], not[member[fix[U[x]], V]],
          not[subclass[cart[image[IMAGE[inverse[DUP]], x], image[IMAGE[inverse[DUP]], x]],
            union[S, inverse[S]]]], not[subclass[image[IMAGE[inverse[DUP]], x], y]]] == True
```

```
In[4]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. A sethood result.

```
In[6]:= SubstTest[implies, member[t, V], member[fix[t], V], t → U[x]] // Reverse

Out[6]= or[member[fix[U[x]], V], not[member[x, V]]] == True

In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[9]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p1, p2, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 → subclass[y, image[CART, id[x]]],
            p2 → subclass[P[y], chains[S]], p3 -> subclass[y, image[CART, Id]],
            p4 → equal[cart[fix[U[y]], fix[U[y]]], U[y]]}]] // Reverse

Out[9]= or[equal[cart[fix[U[y]], fix[U[y]]], U[y]], not[subclass[y, image[CART, id[x]]]],
          not[subclass[cart[y, y], union[S, inverse[S]]]]] == True

In[10]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[12]:= Map[not, SubstTest[and, (* implies[and[p1,p2],p3],implies[p1,p4],implies[p2,p5],*)
            implies[p0, p6], implies[and[p4, p5, p6], p7], not[implies[and[p0, p1, p2], p7]],
            {p0 → member[x, V], p1 -> subclass[x, image[CART, id[y]]],
             p2 -> subclass[P[x], chains[S]], p3 -> equal[U[x], cartsq[fix[U[x]]]],
             p4 → subclass[image[FIX, x], y], p5 → subclass[P[image[FIX, x]], chains[S]],
             p6 -> member[fix[U[x]], V], p7 → member[U[image[FIX, x]], Uchains[y]]}]] // Reverse

Out[12]= or[member[fix[U[x]], Uchains[y]],
           not[member[x, V]], not[subclass[x, image[CART, id[y]]]],
           not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

In[13]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[17]:= or[and[equal[cart[fix[U[x]], fix[U[x]]], U[x]], member[fix[U[x]], Uchains[y]]],
           not[member[x, V]], not[subclass[x, image[CART, id[y]]]],
           not[subclass[cart[x, x], union[S, inverse[S]]]]] // NotNotTest

Out[17]= or[and[equal[cart[fix[U[x]], fix[U[x]]], U[x]], member[fix[U[x]], Uchains[y]]],
           not[member[x, V]], not[subclass[x, image[CART, id[y]]]],
           not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

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

Lemma. Inclusion in one direction.

```
In[19]:= Map[equal[V, #] &, SubstTest[class, t, implies[member[t, u], member[t, v]],
           {u → intersection[chains[S], P[image[CART, id[x]]]],
            v → image[inverse[BIGCUP], image[CART, id[Uchains[x]]]]}]]

Out[19]= subclass[Uchains[image[CART, id[x]]], image[CART, id[Uchains[x]]]] == True

In[20]:= (% /. x → x_) /. Equal → SetDelayed
```

## inclusion in the opposite direction

Lemma.

```
In[22]:= SubstTest[implies,
           and[subclass[P[y], chains[S]], subclass[y, image[CART, id[x]]], member[y, V]],
           member[U[y], Uchains[image[CART, id[x]]]], y → image[CART, id[t]]] // Reverse

Out[22]= or[member[composite[inverse[E], id[t], E], Uchains[image[CART, id[x]]]],
           not[member[image[CART, id[t]], V]], not[subclass[t, x]],
           not[subclass[cart[t, t], union[S, inverse[S]]]]] == True

In[23]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Lemma.

```
In[24]:= Map[not, SubstTest[and, implies[p2, p3], implies[p1, p4], implies[p1, p5],
            (*implies[and[p3,p4,p5],p6],*) implies[p2, p7], implies[and[p6, p7], p8],
            not[implies[and[p1, p2], p8]], {p1 → and[member[t, V], subclass[t, x]],
             p2 → subclass[cart[t, t], union[S, inverse[S]]],
             p3 → subclass[cart[t, t], union[S, inverse[S]]],
             p4 → subclass[t, x], p5 → member[image[CART, id[t]], V],
             p6 → member[composite[inverse[E], id[t], E], Uchains[image[CART, id[x]]]],
             p7 → equal[cart[U[t], U[t]], composite[inverse[E], id[t], E]],
             p8 → member[cart[U[t], U[t]], Uchains[image[CART, id[x]]]]}]] // Reverse

Out[24]= or[member[cart[U[t], U[t]], Uchains[image[CART, id[x]]]], not[member[t, V]],
           not[subclass[t, x]], not[subclass[cart[t, t], union[S, inverse[S]]]]] == True

In[25]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Lemma. Inclusion in the opposite direction.

```
In[26]:= Map[equal[V, #] &, SubstTest[class, t,
           implies[member[t, u], member[t, v]], {u → intersection[chains[S], P[x]], v →
             image[inverse[BIGCUP], fix[image[inverse[CART], Uchains[image[CART, id[x]]]]]]}]]

Out[26]= subclass[Uchains[x], fix[image[inverse[CART], Uchains[image[CART, id[x]]]]]] == True

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

## main theorem and corollaries

The main theorem is obtained by combining the two inclusions derived in the preceding sections.

Theorem. Chain unions of squares are squares of chain unions.

```
In[28]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> Uchains[image[CART, id[x]]], v -> image[CART, id[Uchains[x]]]}]

Out[28]= equal[image[CART, id[Uchains[x]]], Uchains[image[CART, id[x]]]] == True

In[29]:= Uchains[image[CART, id[x_]]] := image[CART, id[Uchains[x]]]
```

Corollary. A variable-free restatement of the theorem.

```
In[31]:= Map[VERTSECT, SubstTest[reify, x, Uchains[image[t, id[x]]], t → CART]]

Out[31]= composite[UCHAINS, IMAGE[CART], IMAGE[DUP]] ==
          composite[IMAGE[CART], IMAGE[DUP], UCHAINS]

In[32]:= composite[UCHAINS, IMAGE[CART], IMAGE[DUP]] :=
          composite[IMAGE[CART], IMAGE[DUP], UCHAINS]
```

Lemma.

```
In[38]:= SubstTest[image, t, union[x, set[0]], t → composite[CART, DUP]] // Reverse

Out[38]= image[CART, id[union[x, set[0]]]] == union[image[CART, id[x]], set[0]]

In[39]:= image[CART, id[union[x_, set[0]]]] := union[image[CART, id[x]], set[0]]
```

The following somewhat special application to group theory was the initial motivation for this study.

Corollary.

```
In[41]:= SubstTest[Uchains, image[CART, id[t]],
           t → image[IMAGE[SECOND], intersection[GROUPS, P[gp[x]]]]] // Reverse

Out[41]= Uchains[image[IMAGE[FIRST], intersection[GROUPS, P[gp[x]]]]] ==
          union[image[IMAGE[FIRST], intersection[GROUPS, P[gp[x]]]], set[0]]

In[42]:= Uchains[image[IMAGE[FIRST], intersection[GROUPS, P[gp[x_]]]]] :=
          union[image[IMAGE[FIRST], intersection[GROUPS, P[gp[x]]]], set[0]]
```