antichains[x]

Johan G. F. Belinfante
2010 July 24

```
In[1]:= SetDirectory["l:"]; << goedel.10jul23a; << tools.m

        :Package Title: goedel.10jul23a             2010 July 23 at 2:55 p.m.

        It is now:  2010 Jul 24 at 14:37

        Loading Simplification Rules

        TOOLS.M                                     Revised 2010 February 26

        weightlimit = 40
```

summary

A class **t** is an **antichain** of **x** if the two-sided restriction of **x** to **t** is the identity relation on **t**. See, for example, page 29 in the following reference.

```
In[2]:=  "Egbert Harzheim, Ordered Sets, Advances in Mathematics, volume 7, Springer
         Science+Business Media, Inc., 2005.  ISBN 0387-24219-8.  QA171.48 .H37";
```

Several formulas for the class **antichains[x]** of all small antichains of a class **x** are derived in this notebook, as well as some basic properties of this class. No restriction is placed on the class **x**, but this concept is primarily useful for partial orders.

antichains

Lemma. A normalization rule for the definition of antichain.

```
In[3]:= equal[composite[id[x], y, id[x]], id[x]] // AssertTest // Reverse

Out[3]= and[subclass[x, fix[y]], subclass[composite[id[x], y, id[x]], Id]] ==
         equal[composite[id[x], y, id[x]], id[x]]

In[4]:= and[subclass[x_, fix[y_]], subclass[composite[id[x_], y_, id[x_]], Id]] :=
         equal[composite[id[x], y, id[x]], id[x]]
```

Theorem. If **u** and **v** are members of an antichain **x** of **y,** and if **u** is related to **v** by **y**, then **u = v**.

```
In[5]:= SubstTest[implies, and[member[r, s], equal[s, t]], member[r, t],
          {r → pair[u, v], s -> composite[id[x], y, id[x]], t -> id[x]}] // Reverse

Out[5]= or[equal[u, v], not[equal[composite[id[x], y, id[x]], id[x]]],
          not[member[u, x]], not[member[v, x]], not[member[pair[u, v], y]]] == True

In[6]:= or[equal[u_, v_], not[equal[composite[id[x_], y_, id[x_]], id[x_]]],
          not[member[u_, x_]], not[member[v_, x_]], not[member[pair[u_, v_], y_]]] := True
```

Theorem. If **x** is an antichain of **y**, then **x ⊂ fix[y]**.

```
In[7]:= implies[equal[composite[id[x], y, id[x]], id[x]], subclass[x, fix[y]]] // AssertTest

Out[7]= or[not[equal[composite[id[x], y, id[x]], id[x]]], subclass[x, fix[y]]] == True

In[8]:= or[not[equal[composite[id[x_], y_, id[x_]], id[x_]]], subclass[x_, fix[y_]]] := True
```

the class of antichains

A new constructor **antichains[x]** is defined by the following membership rule, which says that **t ∈ antichains[x]** if and ony if **t** is a set and **id[t] ∘ x ∘ id[t] = id[t]**.

```
In[9]:= member[t_, antichains[x_]] := and[member[t, V], equal[composite[id[t], x, id[t]], id[t]]]
```

Theorem. A normalization rule.

```
In[10]:= antichains[x] // Normality // InvertFix // Reverse

Out[10]= intersection[cliques[union[Id, complement[x]]], P[fix[x]]] == antichains[x]

In[11]:= intersection[cliques[union[Id, complement[x_]]], P[fix[x_]]] := antichains[x]
```

Theorem. The antichains of **x** are the same as those of **inverse[x]**.

```
In[12]:= SubstTest[intersection, cliques[union[Id, complement[t]]], P[fix[t]], t → inverse[x]]

Out[12]= antichains[inverse[x]] == antichains[x]

In[13]:= antichains[inverse[x_]] := antichains[x]
```

Theorem. Every subset of a class **x** is an antichain for the identity relation **id[x]**.

```
In[14]:= antichains[id[x]] // Normality

Out[14]= antichains[id[x]] == P[x]

In[15]:= antichains[id[x_]] := P[x]
```

Corollary. A special case.

```
In[16]:= SubstTest[antichains, id[x], x → 0] // Reverse

Out[16]= antichains[0] == set[0]

In[17]:= antichains[0] := set[0]
```

Corollary. A special case.

```
In[18]:= SubstTest[antichains, id[x], x → V] // Reverse

Out[18]= antichains[Id] == V

In[19]:= antichains[Id] := V
```

Theorem. Another **cliques** formula for the class of antichains.

```
In[20]:= SubstTest[cliques, restrict[t, fix[x], fix[x]],
           t -> union[id[fix[x]], intersection[Di, complement[x]]]]

Out[20]= cliques[union[id[fix[x]], intersection[Di, complement[x]]]] == antichains[x]

In[21]:= cliques[union[id[fix[x_]], intersection[Di, complement[x_]]]] := antichains[x]
```

Theorem. Another formula for the class of antichains.

```
In[22]:= fix[composite[inverse[IMAGE[DUP]], IMAGE[id[x]], CART, DUP]] // Normality // InvertFix //
           InvertFix

Out[22]= fix[composite[inverse[IMAGE[DUP]], IMAGE[id[x]], CART, DUP]] == antichains[x]

In[23]:= fix[composite[inverse[IMAGE[DUP]], IMAGE[id[x_]], CART, DUP]] := antichains[x]
```

examples of antichains

The empty set is an antichain for any **x**.

```
In[24]:= member[0, antichains[x]]

Out[24]= True
```

Theorem.

```
In[25]:= SubstTest[intersection, range[SINGLETON], intersection[cliques[u], P[v]],
           {u → union[Id, complement[x]], v → fix[x]}] // Reverse

Out[25]= intersection[antichains[x], range[SINGLETON]] == image[SINGLETON, fix[x]]

In[26]:= intersection[antichains[x_], range[SINGLETON]] := image[SINGLETON, fix[x]]
```

Corollary. The singleton of any element of **fix[x]** is an antichain of **x**.

```
In[27]:= SubstTest[subclass, intersection[u, v],
           u, {u → antichains[x], v → range[SINGLETON]}] // Reverse
```

```
Out[27]= subclass[image[SINGLETON, fix[x]], antichains[x]] == True
```

```
In[28]:= subclass[image[SINGLETON, fix[x_]], antichains[x_]] := True
```

Theorem.

```
In[29]:= SubstTest[intersection, cliques[union[Id, complement[t]]], P[fix[t]], t → DISJOINT]
```

```
Out[29]= antichains[DISJOINT] == succ[set[0]]
```

```
In[30]:= antichains[DISJOINT] := succ[set[0]]
```

Theorem. The empty set is the only antichain for an irreflexive relation.

```
In[31]:= SubstTest[intersection, cliques[union[Id, complement[t]]],
          P[fix[t]], t → intersection[Di, x]]
```

```
Out[31]= antichains[intersection[Di, x]] == set[0]
```

```
In[32]:= antichains[intersection[Di, x_]] := set[0]
```

some properties of the class of antichains

Theorem. Any subset of an antichain is an antichain.

```
In[33]:= SubstTest[image, inverse[S], intersection[cliques[u], P[v]],
           {u → union[Id, complement[x]], v → fix[x]}] // Reverse
```

```
Out[33]= image[inverse[S], antichains[x]] == antichains[x]
```

```
In[34]:= image[inverse[S], antichains[x_]] := antichains[x]
```

As a corollary, it follows that the **HULL** function of **antichains[x]** is the identity function on that class.

```
In[46]:= HULL[antichains[x]]
```

```
Out[46]= id[antichains[x]]
```

It also follows that the intersection of any nonempty set of antichains is an antichain.

```
In[36]:= Aclosure[antichains[x]]
```

```
Out[36]= antichains[x]
```

Theorem. The union of all antichains of **x** is **fix[x]**.

```
In[37]:= SubstTest[U, intersection[cliques[u], P[v]],
            {u → union[Id, complement[x]], v → fix[x]}] // Reverse

Out[37]= U[antichains[x]] == fix[x]

In[38]:= U[antichains[x_]] := fix[x]
```

A corollary of this is that any subset of **fix[x]** is an antichain of **x**.

```
In[39]:= Uclosure[antichains[x]]

Out[39]= P[fix[x]]
```

Theorem. The class **antichains[x]** is closed under unions of chains.

```
In[40]:= SubstTest[Uchains, intersection[cliques[u], P[v]],
            {u → union[Id, complement[x]], v → fix[x]}] // Reverse

Out[40]= Uchains[antichains[x]] == antichains[x]

In[41]:= Uchains[antichains[x_]] := antichains[x]
```

Lemma. A normalization rule.

```
In[42]:= SubstTest[fix,
            composite[E, complement[composite[MAXIMAL[t], S]]], t → inverse[x]] // Reverse

Out[42]= fix[composite[E, complement[composite[MINIMAL[x], S]]]] ==
          complement[cliques[union[Id, complement[x]]]]

In[43]:= fix[composite[E, complement[composite[MINIMAL[x_], S]]]] :=
          complement[cliques[union[Id, complement[x]]]]
```

Theorem.

```
In[44]:= antichains[union[x, inverse[x]]] // Normality // InvertFix

Out[44]= antichains[union[x, inverse[x]]] == antichains[x]

In[45]:= antichains[union[x_, inverse[x_]]] := antichains[x]
```

serendipity: cutting cartesian products with Id

The following rewrite rule was discovered in the course of studying alternative expressions for the class of antichains.

Theorem.

```
In[47]:= Assoc[IMAGE[DUP], IMAGE[inverse[DUP]], CART] // Reverse

Out[47]= composite[IMAGE[id[Id]], CART] == composite[CAP, cross[IMAGE[DUP], IMAGE[DUP]]]

In[48]:= composite[IMAGE[id[Id]], CART] := composite[CAP, cross[IMAGE[DUP], IMAGE[DUP]]]
```