Zorn's lemma and MAXIMAL[S]

Johan G. F. Belinfante
2007 November 23

```
In[1]:= SetDirectory["l:"]; << goedel99.23a; << tools.m

        :Package Title: goedel99.23a                    2007 November 23 at 5:20 p.m.

        It is now:  2007 Nov 23 at 22:33

        Loading Simplification Rules

        TOOLS.M                                    Revised 2007 September 19

        weightlimit = 40
```

summary

In an earlier notebook, the following variable-free special version of Zorn's lemma was derived:

```
In[2]:= empty[fix[composite[inverse[IMAGE[inverse[PS]]], S, UCHAINS]]]

Out[2]= axch
```

In this notebook, a slightly stronger version of Zorn's lemma is derived. Both the new result and the above result are concerned only with the case of sets partially ordered by inclusion. Let us say that a set **x** is **inductive** if every chain in **x** has an upper bound, that is, if the following condition holds:

```
In[3]:= assert[forall[t, implies[and[member[t, chains[S]], subclass[t, x]],
           exists[u, and[member[u, x], subclass[U[t], u]]]]]]

Out[3]= subclass[Uchains[x], image[inverse[S], x]]
```

The version of Zorn's lemma derived earlier implies that when the axiom of choice holds, any inductive set has a maximal element. The new version to be derived below says a bit more; under the same conditions, not only do maximal elements exist, but every element in an inductive set lies below some maximal element. That is, the conclusion is sharpened to the following:

```
In[4]:= assert[forall[t, implies[member[t, x], exists[u, and[member[u, x], subclass[t, u],
             forall[v, implies[member[v, x], not[member[pair[u, v], PS]]]]]]]]]

Out[4]= subclass[x, image[inverse[S], intersection[x, complement[image[inverse[PS], x]]]]]
```

When the set variable **x** is eliminated from the resulting statement of Zorn's lemma, the class of sets satisfying this sharper conclusion briefly makes an appearance, although it does not appear in the final rewrite rule:

```
In[5]:= class[x, subclass[x,
          image[inverse[S], intersection[x, complement[image[inverse[PS], x]]]]]] // InvertFix

Out[5]= complement[fix[composite[E, complement[composite[inverse[S], MAXIMAL[S]]]]]]
```

Just as for the earlier version, the **GOEDEL** program automatically combines the hypothesis and the conclusion of Zorn's lemma neatly into a single literal, which in this case is a statement that a certain relation is a subclass of another. The above fixed point class is related to the domain of one of these relations. A converse theorem is also derived, resulting in a simple rule that just rewrites this literal to **axch**.

---

membership rule

Lemma. (Membership rule for the relation **composite[inverse[S], MAXIMAL[S]]**.

```
In[6]:= member[pair[x, y], composite[inverse[S], MAXIMAL[S]]] // AssertTest

Out[6]= member[pair[x, y], composite[inverse[S], MAXIMAL[S]]] == and[member[x, V],
          not[subclass[intersection[x, image[S, set[y]]], image[inverse[PS], x]]]]

In[7]:= member[pair[x_, y_], composite[inverse[S], MAXIMAL[S]]] := and[member[x, V],
          not[subclass[intersection[x, image[S, set[y]]], image[inverse[PS], x]]]]
```

Comment. This is just the condition that the set of elements of **x** which contain **y** is not free of maximal elements of **x**; in other words, the condition that **y** is a subset of some maximal element of **x**.

---

a sharper conclusion for Zorn's lemma

Lemma.

```
In[8]:= or[not[subclass[x, image[inverse[PS], x]]],
          not[subclass[Uchains[x], image[inverse[S], x]]],
          subclass[Uchains[x], image[inverse[PS], x]]] // NotNotTest

Out[8]= or[not[subclass[x, image[inverse[PS], x]]],
          not[subclass[Uchains[x], image[inverse[S], x]]],
          subclass[Uchains[x], image[inverse[PS], x]]] == True

In[9]:= (% /. x → x_) /. Equal → SetDelayed
```

The previously derived version of Zorn's lemma was stated in a succinct negative form. To facilitate reasoning it the following positive version of it is convenient.

```
In[10]:= or[not[axch], not[member[x, y]], not[subclass[x, image[inverse[PS], x]]],
           not[subclass[Uchains[x], image[inverse[S], x]]]] // NotNotTest

Out[10]= or[not[axch], not[member[x, y]], not[subclass[x, image[inverse[PS], x]]],
           not[subclass[Uchains[x], image[inverse[S], x]]]] == True
```

```
In[11]:= or[not[axch], not[member[x_, y_]], not[subclass[x_, image[inverse[PS], x_]]],
           not[subclass[Uchains[x_], image[inverse[S], x_]]]] := True
```

The idea now is to single out some member **y** of **x**, and applies the above weak version to the subset **intersection[x, image[S, set[y]]]** of **x**. If x is inductive, then this subset is also inductive, and one obtains the following statement.

```
In[12]:= Map[not,
           SubstTest[and, implies[p1, p4], implies[and[p2, p3], p5], implies[and[p0, p4, p5], p6],
            not[implies[and[p0, p1, p2, p3], p6]], {p0 → axch, p1 -> member[x, z],
             p2 → member[y, x], p3 → subclass[Uchains[x], image[inverse[S], x]],
             p4 -> member[intersection[x, image[S, set[y]]], V],
             p5 -> subclass[Uchains[intersection[x, image[S, set[y]]]],
               image[inverse[S], intersection[x, image[S, set[y]]]]],
             p6 -> not[subclass[intersection[x, image[S, set[y]]],
                image[inverse[PS], intersection[x, image[S, set[y]]]]]]}]] // Reverse

Out[12]= or[not[axch], not[member[x, z]],
           not[member[y, x]], not[subclass[intersection[x, image[S, set[y]]],
             image[inverse[PS], intersection[x, image[S, set[y]]]]]],
           not[subclass[Uchains[x], image[inverse[S], x]]]] == True

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

This says that the axiom of choice implies that for any element **y** of an inductive set **x**, the set of upper bounds in **x** for **set[y]** holds a maximal element of that subset. It only remains to show that such a maximal element of the subset is also a maximal element of **x** itself. This can be done without introducing any new variables, with a series of lemmas.

```
In[14]:= subclass[composite[id[S], inverse[SECOND], inverse[PS]],
           composite[inverse[FIRST], inverse[S]]] // AssertTest

Out[14]= subclass[composite[id[S], inverse[SECOND], inverse[PS]],
           composite[inverse[FIRST], inverse[S]]] == True

In[15]:= % /. Equal → SetDelayed
```

Lemma.

```
In[16]:= SubstTest[implies, subclass[u, v], subclass[image[u, x], image[v, x]],
           {u -> composite[id[S], inverse[SECOND], inverse[PS]],
            v -> intersection[composite[inverse[FIRST], inverse[S]],
              composite[inverse[SECOND], inverse[PS]]]}] // Reverse

Out[16]= subclass[composite[id[image[inverse[PS], x]], S],
           composite[inverse[PS], id[x], S]] == True

In[17]:= subclass[composite[id[image[inverse[PS], x_]], S],
           composite[inverse[PS], id[x_], S]] := True
```

Lemma.

```
In[18]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
           {u -> composite[id[image[inverse[PS], x]], S],
            v -> composite[inverse[PS], id[x], S], w → set[y]}] // Reverse

Out[18]= subclass[intersection[image[S, set[y]], image[inverse[PS], x]],
           image[inverse[PS], intersection[x, image[S, set[y]]]]] == True

In[19]:= subclass[intersection[image[S, set[y_]], image[inverse[PS], x_]],
           image[inverse[PS], intersection[x_, image[S, set[y_]]]]] := True
```

Lemma.

```
In[20]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u -> intersection[x, image[S, set[y]]],
            v -> intersection[image[S, set[y]], image[inverse[PS], x]],
            w -> image[inverse[PS], intersection[x, image[S, set[y]]]]}] // Reverse

Out[20]= or[not[subclass[intersection[x, image[S, set[y]]], image[inverse[PS], x]]],
           subclass[intersection[x, image[S, set[y]]],
            image[inverse[PS], intersection[x, image[S, set[y]]]]]] == True

In[21]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

One now obtains the following sharper version of Zorn's lemma, which still involves the variable **y**.

```
In[22]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
            {p1 → and[axch, member[x, z],
               member[y, x], subclass[Uchains[x], image[inverse[S], x]]],
             p2 -> not[subclass[intersection[x, image[S, set[y]]],
                image[inverse[PS], intersection[x, image[S, set[y]]]]]],
             p3 -> not[subclass[intersection[x, image[S, set[y]]], image[inverse[PS], x]]]}]] //
          Reverse

Out[22]= or[not[axch], not[member[x, z]], not[member[y, x]],
           not[subclass[intersection[x, image[S, set[y]]], image[inverse[PS], x]]],
           not[subclass[Uchains[x], image[inverse[S], x]]]] == True

In[23]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Main Theorem. Sharper version of Zorn's lemma, obtained by eliminating the variable that refers to the singled-out member of **x**.

```
In[24]:= Map[implies[member[x, y], equal[V, #]] &,
          SubstTest[class, t, implies[and[equal[s, v], member[x, v], member[t, x], member[x, u]],
            not[subclass[intersection[x, image[S, set[t]]], image[inverse[PS], x]]]],
           {s → SELECT, v → V, u -> fix[composite[inverse[IMAGE[inverse[S]]], S, UCHAINS]]}]]

Out[24]= or[not[axch], not[member[x, y]],
           not[subclass[Uchains[x], image[inverse[S], x]]], subclass[x,
            image[inverse[S], intersection[x, complement[image[inverse[PS], x]]]]]] == True
```

```
In[25]:= or[not[axch], not[member[x_, y_]],
           not[subclass[Uchains[x_], image[inverse[S], x_]]], subclass[x_,
            image[inverse[S], intersection[x_, complement[image[inverse[PS], x_]]]]]] := True
```

a variable-free version

Lemma. (Membership rule for the class of elements that do not satisfy the conclusion of the sharper version of Zorn's lemma.)

```
In[26]:= member[x, fix[composite[E, complement[composite[inverse[S], MAXIMAL[S]]]]]] //
          AssertTest

Out[26]= member[x, fix[composite[E, complement[composite[inverse[S], MAXIMAL[S]]]]]] ==
          and[member[x, V], not[subclass[x,
             image[inverse[S], intersection[x, complement[image[inverse[PS], x]]]]]]]

In[27]:= member[x_, fix[composite[E, complement[composite[inverse[S], MAXIMAL[S]]]]]] :=
          and[member[x, V], not[subclass[x,
             image[inverse[S], intersection[x, complement[image[inverse[PS], x]]]]]]]
```

A variable-free version of the main theorem is obtained by eliminating the variable **x**.

```
In[28]:= Map[equal[V, #] &,
          SubstTest[class, x, implies[and[equal[s, V], member[x, u]], member[x, v]],
           {s → SELECT, u -> fix[composite[inverse[IMAGE[inverse[S]]], S, UCHAINS]],
            v -> complement[fix[composite[E, complement[composite[inverse[S], MAXIMAL[S]]]]]]}]]

Out[28]= or[not[axch], subclass[
            composite[inverse[E], id[fix[composite[inverse[IMAGE[inverse[S]]], S, UCHAINS]]]],
            composite[inverse[S], MAXIMAL[S]]]] == True

In[29]:= % /. Equal → SetDelayed
```

a converse theorem

Lemma.

```
In[30]:= Map[empty,
          dif[fix[composite[inverse[IMAGE[inverse[PS]]], S, UCHAINS]], set[0]] // Renormality]

Out[30]= subclass[fix[composite[inverse[IMAGE[inverse[PS]]], S, UCHAINS]], set[0]] == axch

In[31]:= % /. Equal → SetDelayed
```

Converse theorem.

```
In[32]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → FIRST, u ->
             composite[inverse[E], id[fix[composite[inverse[IMAGE[inverse[S]]], S, UCHAINS]]]],
            v -> composite[inverse[S], MAXIMAL[S]]}] // Reverse

Out[32]= or[axch, not[subclass[
             composite[inverse[E], id[fix[composite[inverse[IMAGE[inverse[S]]], S, UCHAINS]]]],
             composite[inverse[S], MAXIMAL[S]]]]] == True

In[33]:= % /. Equal → SetDelayed
```

The sharper variable-free version of Zorn's lemma and its converse can be combined into a fairly simple rewrite rule:

```
In[34]:= equiv[subclass[
           composite[inverse[E], id[fix[composite[inverse[IMAGE[inverse[S]]], S, UCHAINS]]]],
           composite[inverse[S], MAXIMAL[S]]], axch]

Out[34]= True

In[35]:= subclass[
           composite[inverse[E], id[fix[composite[inverse[IMAGE[inverse[S]]], S, UCHAINS]]]],
           composite[inverse[S], MAXIMAL[S]]] := axch
```