(finchar[x] & 0 ∈ x) => Uchains[x] = x

Johan G. F. Belinfante
2008 January 4

```
In[1]:= SetDirectory["l:"]; << goedel.08jan04a; << tools.m

    :Package Title: goedel.08jan04a            2008 January 4 at 4:20 p.m.

    It is now:  2008 Jan 4 at 16:27

    Loading Simplification Rules

    TOOLS.M                                    Revised 2008 January 2

    weightlimit = 40
```

summary

Any class of finite character which holds the empty set is closed under unions of chains. A corollary of this is Tukey's lemma: if the axiom of choice holds, then any class of finite character which holds the empty set has a maximal element.

```
In[2]:= Begin["Goedel`Private`"];

In[3]:= ?? finchar

    finchar[x] is the statement that x is a class of finite character

    finchar[x_] := equal[complement[x], image[S, intersection[FINITE, complement[x]]]]
```

normalization of the predicate finchar

In the course of the derivation several variables are introduced and later eliminated using **class** rules. Each time this happens, the negation of the predicate **finchar** needs to be restored. This normalization is easily accomplished using a double negation, but to save time it helps to have a rewrite rule that automatically takes care of this once and for all.

```
In[4]:= or[not[equal[0, intersection[x, image[S, intersection[FINITE, complement[x]]]]]],
          not[equal[V, union[x, image[S, intersection[FINITE, complement[x]]]]]]] // NotNotTest

Out[4]= or[not[equal[0, intersection[x, image[S, intersection[FINITE, complement[x]]]]]],
          not[equal[V, union[x, image[S, intersection[FINITE, complement[x]]]]]]] ==
         not[equal[complement[x], image[S, intersection[FINITE, complement[x]]]]]

In[5]:= or[not[equal[0, intersection[x_, image[S, intersection[FINITE, complement[x_]]]]]],
          not[equal[V, union[x_, image[S, intersection[FINITE, complement[x_]]]]]]] :=
         not[equal[complement[x], image[S, intersection[FINITE, complement[x]]]]]
```

This reduction of two literals to one is not a minor matter. An important contributor to execution time is the proliferation of possible equality substitutions as the number of literals increases; combinatorial explosion due is always lurking just around the corner. For the same reason, the **equality** flag will be repeatedly cleared and reset as needed to control this problem.

strategy of the derivation

We want to prove that if **finchar[y]** and **member[0,y]**, then **member[x, intersection[chains[S], P[y]]]** implies **member[-U[x], y]**. Because **y** is of finite character, it suffices to show that every finite subset of **U[x]** belongs to **y**. We consider some finite subset **t** of **U[x]** and attempt to show that **t** belongs to **y**. Since **t** is a finite set covered by **x**, for each member of **t** one can select a member of **x** that holds that member of **t**. The set of these selected members is a finite subset of **x** whose union contains **t**. For the formal proof one needs a theorem of finite choice; for this purpose, the following formulation of finite choice will be used:

```
In[6]:= implies[and[member[t, FINITE], subclass[t, U[x]]], not[disjoint[P[E], map[t, x]]]]

Out[6]= True
```

Another variable **z** will be used for this selection function; **z** is a member of the intersection of **P[E]** and **map[t,x]**. After reasoning with all these variables, all of them will be eliminated, one by one.

temporary simplification rules

To avoid introducing yet another variable for **range[z]**, some temporary rewrite rules are derived to undo the effects of a rewrite rule affects the range, namely this one:

```
In[7]:= equal[0, range[z]]

Out[7]= equal[0, domain[z]]
```

Temporary lemma. (A finite nonempty chain has a greatest element.)

```
In[8]:= SubstTest[implies, and[member[t, FINITE], member[t, chains[S]]],
          or[empty[t], member[U[t], t]], t → range[z]] // Reverse

Out[8]= or[equal[0, domain[z]], member[U[range[z]], range[z]], not[member[range[z], FINITE]],
          not[subclass[cart[range[z], range[z]], union[S, inverse[S]]]]] == True

In[9]:= (% /. z → z_) /. Equal → SetDelayed
```

Temporary lemma.

```
In[10]:= SubstTest[implies, and[subclass[t, U[s]], empty[s]], empty[t], s → range[z]] // Reverse

Out[10]= or[equal[0, t], not[equal[0, domain[z]]], not[subclass[t, U[range[z]]]]] == True

In[11]:= (% /. {t → t_, z → z_}) /. Equal → SetDelayed
```

a long series of lemmas

The original layout of the proof of the main theorem involved about twenty statements, far too many to be handled all at once by the **GOEDEL** program. For this reason the derivation was broken down into a series of lemmas, each of which eliminates one of these statements. The original labels for these statements are retained here, and for each lemma, the statement being eliminated is identified.

Lemma. (eliminating **p14**)

```
In[12]:= Map[not, SubstTest[and, implies[and[p4a, p4b], p7], implies[and[p13, p10], p14],
           implies[and[p7, p14], p15], not[implies[and[p4a, p4b, p10, p13], p15]],
           {p4a → member[z, map[t, x]], p4b → subclass[z, E], p7 → subclass[t, U[range[z]]],
            p10 → member[U[range[z]], range[z]], p13 → subclass[range[z], y],
            p14 → member[U[range[z]], y], p15 → member[t, image[inverse[S], y]]}]] // Reverse

Out[12]= or[member[t, image[inverse[S], y]],
          not[member[z, map[t, x]]], not[member[U[range[z]], range[z]]],
          not[subclass[z, E]], not[subclass[range[z], y]]] == True

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

Lemma. (eliminating **p7** and **p12**) This lemma just requires a double negation.

```
In[14]:= or[member[t, y], not[equal[0, domain[z]]], not[member[0, y]],
          not[member[z, map[t, x]]], not[subclass[z, E]]] // NotNotTest

Out[14]= or[member[t, y], not[equal[0, domain[z]]],
          not[member[0, y]], not[member[z, map[t, x]]], not[subclass[z, E]]] == True

In[15]:= (% /. {t → t_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma. (eliminating one of two uses of **p8**.)

```
In[16]:= Map[not, SubstTest[and, implies[p4b, p8],
           implies[and[p1c, p8], p9], not[implies[and[p1c, p4b], p9]],
           {p1c -> subclass[cart[x, x], union[S, inverse[S]]], p4b → member[z, map[t, x]],
            p8 -> subclass[range[z], x], p9 -> subclass[P[range[z]], chains[S]]}]] // Reverse

Out[16]= or[not[member[z, map[t, x]]], not[subclass[cart[x, x], union[S, inverse[S]]]],
          subclass[cart[range[z], range[z]], union[S, inverse[S]]]] == True

In[17]:= (% /. {t → t_, x → x_, z → z_}) /. Equal → SetDelayed
```

Lemma. (eliminating a second use of **p8**.)

```
In[18]:= Map[not, SubstTest[and, implies[p4b, p8], implies[and[p1b, p8], p13],
           not[implies[and[p1b, p4b], p13]], {p1b → subclass[x, y], p4b → member[z, map[t, x]],
            p8 → subclass[range[z], x], p13 → subclass[range[z], y]}]] // Reverse

Out[18]= or[not[member[z, map[t, x]]], not[subclass[x, y]], subclass[range[z], y]] == True
```

```
In[19]:= (% /. {t → t_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma. (eliminating **p6** and **p9**.) This lemma only takes a fraction of a minute, but it seems like a long wait.

```
In[20]:= Map[not, SubstTest[and, implies[and[p3a, p4b], p6], implies[and[p1b, p4b], p9],
            implies[and[p6, p9], or[p10, p11]], not[implies[and[p1b, p3a, p4b], or[p10, p11]]],
            {p1b -> subclass[cart[x, x], union[S, inverse[S]]],
             p3a -> member[t, FINITE], p4b -> member[z, map[t, x]],
             p6 → member[range[z], FINITE], p9 → subclass[P[range[z]], chains[S]],
             p10 → member[U[range[z]], range[z]], p11 → empty[range[z]]}]] // Reverse

Out[20]= or[equal[0, domain[z]], member[U[range[z]], range[z]], not[member[t, FINITE]],
           not[member[z, map[t, x]]], not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

In[21]:= (% /. {t → t_, x → x_, z → z_}) /. Equal → SetDelayed
```

Lemma. (eliminating **p13**)

```
In[22]:= Map[not, SubstTest[and, implies[and[p1a, p4b], p13],
            implies[and[p4a, p4b, p10, p13], p15], not[implies[and[p1a, p4a, p4b, p10], p15]],
            {p1a → subclass[x, y], p4a → subclass[z, E], p4b -> member[z, map[t, x]],
             p10 → member[U[range[z]], range[z]], p11 → empty[range[z]],
             p13 → subclass[range[z], y],
             p15 → member[t, image[inverse[S], y]]}]] // Reverse

Out[22]= or[member[t, image[inverse[S], y]], not[member[z, map[t, x]]],
           not[member[U[range[z]], range[z]]], not[subclass[x, y]], not[subclass[z, E]]] == True

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

Lemma. (eliminating one use of **p15**)

```
In[24]:= Map[not, SubstTest[and, implies[and[p1a, p4a, p4b, p10], p15],
            implies[and[p15, p16], p17], not[implies[and[p1a, p4a, p4b, p10, p16], p17]],
            {p1a → subclass[x, y], p4a → subclass[z, E], p4b -> member[z, map[t, x]],
             p10 → member[U[range[z]], range[z]],
             p15 → member[t, image[inverse[S], y]], p16 → equal[image[inverse[S], y], y],
             p17 → member[t, y]}]] // Reverse

Out[24]= or[member[t, y], not[equal[y, image[inverse[S], y]]], not[member[z, map[t, x]]],
           not[member[U[range[z]], range[z]]], not[subclass[x, y]], not[subclass[z, E]]] == True

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

Lemma. (Eliminating a second use of **p15**)

```
In[26]:= Map[not, SubstTest[and, implies[and[p1b, p3a, p4b], or[p10, p11]],
            implies[and[p2b, p4a, p4b, p11], p17],
            implies[and[p1a, p4a, p4b, p10, p16], p17],
            not[implies[and[p1a, p1b, p16, p2b, p3a, p4a, p4b], p17]], {p1a → subclass[x, y],
             p1b -> subclass[cart[x, x], union[S, inverse[S]]], p2b -> member[0, y],
             p3a -> member[t, FINITE], p4a → subclass[z, E], p4b -> member[z, map[t, x]],
             p10 → member[U[range[z]], range[z]], p11 → empty[range[z]],
             p13 → subclass[range[z], y],
             p15 → member[t, image[inverse[S], y]], p16 → equal[image[inverse[S], y], y],
             p17 → member[t, y]}]] // Reverse

Out[26]= or[member[t, y], not[equal[y, image[inverse[S], y]]], not[member[0, y]],
           not[member[t, FINITE]], not[member[z, map[t, x]]], not[subclass[x, y]],
           not[subclass[z, E]], not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

In[27]:= (% /. {t → t_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma. (classes of finite character are hereditary.)

```
In[28]:= Map[not, SubstTest[and, implies[p1a, p2], implies[and[p1b, p2, p3, p4, p5], p6],
            not[implies[and[p1a, p1b, p3, p4, p5], p6]], {p1a → finchar[y], p1b -> member[0, y],
             p2 → equal[y, image[inverse[S], y]], p3 → member[z, intersection[P[E], map[t, x]]],
             p4 → member[x, intersection[chains[S], P[y]]],
             p5 -> member[t, FINITE], p6 → member[t, y]}]] // Reverse

Out[28]= or[member[t, y],
           not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
           not[member[0, y]], not[member[t, FINITE]], not[member[x, V]],
           not[member[z, map[t, x]]], not[subclass[x, y]], not[subclass[z, E]],
           not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

In[29]:= (% /. {t → t_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

using wrappers to reduce the number of literals

The next step, to eliminate the variable **z**, was delayed for some time because there were too many literals. This problem was eventially resolved by using wrappers to reduce the number of literals. Three wrappers are introduced at this point: **fin**, **setpart** and **spine**.

```
In[30]:= SubstTest[or, member[t, y],
           not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
           not[member[0, y]], not[member[t, FINITE]], not[member[x, V]],
           not[member[z, map[t, x]]], not[subclass[x, y]], not[subclass[z, E]],
           not[subclass[cart[x, x], union[S, inverse[S]]]],
           {t → fin[u], x → spine[S, setpart[v]]}] // Reverse

Out[30]= or[member[fin[u], y],
           not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
           not[member[0, y]], not[member[z, map[fin[u], spine[S, setpart[v]]]]],
           not[subclass[z, E]], not[subclass[spine[S, setpart[v]], y]]] == True
```

```
In[31]:= (% /. {u → u_, v → v_, y → y_, z → z_}) /. Equal → SetDelayed
```

Before eliminating **z**, some flags are cleared.

```
In[32]:= simplify = False; cond = False; equality = False;
```

The elimination of **z** does not take long now.

```
In[33]:= Map[equal[V, #] &, SubstTest[class, z,
           implies[and[equal[r, s], member[0, y], subclass[w, y], member[z, u]], member[v, y]],
           {r → complement[y], s → image[S, intersection[FINITE, complement[y]]],
            u → intersection[P[E], map[fin[t], spine[S, setpart[x]]]],
            v → fin[t], w → spine[S, setpart[x]]}]]

Out[33]= or[equal[0, intersection[map[fin[t], spine[S, setpart[x]]], P[E]]], member[fin[t], y],
           not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
           not[member[0, y]], not[subclass[spine[S, setpart[x]], y]]] == True

In[34]:= (% /. {x → x_, y → y_, t → t_}) /. Equal → SetDelayed
```

One needs to restore **equality** flag because it is needed for wrapper removal. The **simplify** and **cond** flags remain cleared at this point.

```
In[35]:= equality = True;
```

Remove the compound **spine[S, setpart[x]]** wrapper.

```
In[36]:= SubstTest[implies, equal[x, spine[S, setpart[u]]],
           or[equal[0, intersection[map[fin[t], x], P[E]]], member[fin[t], y],
            not[finchar[y]], not[member[0, y]], not[subclass[x, y]]], u → x] // Reverse

Out[36]= or[equal[0, intersection[map[fin[t], x], P[E]]], member[fin[t], y],
           not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
           not[member[0, y]], not[member[x, V]], not[subclass[x, y]],
           not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

In[37]:= (% /. {x → x_, t → t_, y → y_}) /. Equal → SetDelayed
```

Remove the **fin** wrappers.

```
In[38]:= SubstTest[implies, equal[t, fin[v]],
           or[equal[0, intersection[map[t, x], P[E]]], member[t, y], not[finchar[y]],
            not[member[0, y]], not[member[x, chains[S]]], not[subclass[x, y]]], v → t] // Reverse

Out[38]= or[equal[0, intersection[map[t, x], P[E]]], member[t, y],
           not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
           not[member[0, y]], not[member[t, FINITE]], not[member[x, V]],
           not[subclass[x, y]], not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

In[39]:= (% /. {x → x_, t → t_, y → y_}) /. Equal → SetDelayed
```

At this point one can eliminate all mention of **P[E]**:

```
In[40]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4],
            implies[p2, p3], not[implies[and[p1, p2], p4]],
            {p1 → and[member[0, y], finchar[y], member[x, chains[S]], subclass[x, y]],
             p2 → and[member[t, FINITE], subclass[t, U[x]]], p3 → not[disjoint[P[E], map[t, x]]],
             p4 → member[t, y]}]] // Reverse

Out[40]= or[member[t, y],
          not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
          not[member[0, y]], not[member[t, FINITE]], not[member[x, V]], not[subclass[t, U[x]]],
          not[subclass[x, y]], not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

In[41]:= (% /. {x → x_, t → t_, y → y_}) /. Equal → SetDelayed
```

Before eliminating the variable **t** the equality flag is cleared.

```
In[42]:= equality = False;

In[43]:= Map[equal[V, #] &, SubstTest[class, t,
          implies[and[finchar[y], member[0, y], member[x, u], member[t, v]], member[t, y]],
          {u -> intersection[chains[S], P[y]], v -> intersection[FINITE, P[U[x]]]}]]

Out[43]= or[not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
          not[member[0, y]], not[member[x, V]], not[subclass[x, y]],
          not[subclass[cart[x, x], union[S, inverse[S]]]],
          subclass[intersection[FINITE, P[U[x]]], y]] == True

In[44]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. (A set belongs to a class of finite character iff all its finite subsets are members.)

```
In[45]:= SubstTest[or, member[t, y], not[member[t, V]], not[finchar[y]],
          not[subclass[intersection[FINITE, P[t]], y]], t → U[x]] // Reverse

Out[45]= or[member[U[x], y],
          not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
          not[member[x, V]], not[subclass[intersection[FINITE, P[U[x]]], y]]] == True

In[46]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[47]:= Map[not, SubstTest[and, implies[and[p1, p2], p5],
            implies[and[p1, p2, p5], p6], not[implies[and[p1, p2], p6]],
            {p1 → finchar[y], p2 → and[member[0, y], member[x, intersection[chains[S], P[y]]]],
             p5 -> subclass[intersection[FINITE, P[U[x]]], y],
             p6 → member[U[x], y]}]] // Reverse

Out[47]= or[member[U[x], y],
          not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
          not[member[0, y]], not[member[x, V]], not[subclass[x, y]],
          not[subclass[cart[x, x], union[S, inverse[S]]]]] == True

In[48]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Now the variable **x** needs to be removed.

```
In[49]:= Map[equal[V, #] &, SubstTest[class, x,
           implies[and[finchar[y], member[0, y], member[x, t]], member[U[x], y]],
           t -> intersection[chains[S], P[y]]]]]

Out[49]= or[not[equal[complement[y], image[S, intersection[FINITE, complement[y]]]]],
           not[member[0, y]], subclass[Uchains[y], y]] == True

In[50]:= (% /. y → y_) /. Equal → SetDelayed
```

This can be cleaned up using this lemma:

```
In[51]:= or[equal[y, Uchains[y]], not[subclass[Uchains[y], y]]] // AssertTest

Out[51]= or[equal[y, Uchains[y]], not[subclass[Uchains[y], y]]] == True

In[52]:= (% /. y → y_) /. Equal → SetDelayed
```

Main Theorem. A class of finite character which holds the empty set is closed under unions of chains.

```
In[53]:= Map[not, SubstTest[and, implies[and[p0, p1], p4], implies[p4, p5],
             not[implies[and[p0, p1], p5]], {p0 → member[0, x], p1 → finchar[x],
              p4 -> subclass[Uchains[x], x], p5 → equal[Uchains[x], x]}]] // Reverse

Out[53]= or[equal[x, Uchains[x]],
           not[equal[complement[x], image[S, intersection[FINITE, complement[x]]]]],
           not[member[0, x]]] == True

In[54]:= or[equal[x_, Uchains[x_]],
           not[equal[complement[x_], image[S, intersection[FINITE, complement[x_]]]]],
           not[member[0, x_]]] := True
```

Corollary. A completely variable-free statement is possible. This weaker statement only captures the case of sets of finite character, not proper classes.

```
In[55]:= Map[equal[V, #] &,
          dif[intersection[FINCHAR, image[E, set[0]]], fix[UCHAINS]] // complement //
           Renormality]

Out[55]= subclass[FINCHAR, union[fix[UCHAINS], P[complement[set[0]]]]] == True

In[56]:= subclass[FINCHAR, union[fix[UCHAINS], P[complement[set[0]]]]] := True
```

Tukey's lemma

As a corollary of Zorn's lemma, one can derive a famous result due to Tukey.

```
In[57]:= "J. W. Tukey, Convergence and Uniformity
             in Topology, Ann. Math. Studies, No. 2, Princeton, 1940.";
```

Tukey's lemma.

```
In[58]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → intersection[FINCHAR, image[E, set[0]]],
            v → fix[UCHAINS], w → domain[MAXIMAL[S]]}] // Reverse

Out[58]= or[not[axch], not[member[0, U[intersection[FINCHAR, subvar[inverse[PS]]]]]]] == True

In[59]:= % /. Equal → SetDelayed
```

The converse is also true:

```
In[60]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u -> range[CLIQUES],
            v -> intersection[FINCHAR, image[E, set[0]]], w → domain[MAXIMAL[S]]}] // Reverse

Out[60]= or[axch, member[0, U[intersection[FINCHAR, subvar[inverse[PS]]]]]] == True

In[61]:= % /. Equal → SetDelayed
```

These results can be combined into a single rewrite rule:

```
In[62]:= equiv[member[0, U[intersection[FINCHAR, subvar[inverse[PS]]]]], not[axch]]

Out[62]= True

In[63]:= member[0, U[intersection[FINCHAR, subvar[inverse[PS]]]]] := not[axch]
```

a counterexample

One cannot omit the hypothesis **member[0, x]** in the main theorem. In general, a class of finite character need not be closed under unions of chains. This is simply because the union of the empty chain need not belong to a class of finite character.

```
In[64]:= Map[not, SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
            {u → FINCHAR, v → fix[UCHAINS], w → image[E, set[0]]}]] // Reverse

Out[64]= subclass[FINCHAR, fix[UCHAINS]] == False

In[65]:= subclass[FINCHAR, fix[UCHAINS]] := False
```

Comment. Jean Rubin explicitly also adds an analog of the hypothesis **member[0, x]** to her version of Tukey's lemma. Her version of Tukey's lemma is presented as a theorem-schema with a ‘variable predicate’ in place of a free class variable.

```
In[66]:=   "Jean E. Rubin, Set Theory for the
            Mathematician, Holden Day, San Francisco, 1967.  (see page 254.)";
```