UB[Q]

Johan G. F. Belinfante
2011 July 17

```
In[1]:= SetDirectory["l:"]; << goedel.11jul15a

        :Package Title: goedel.11jul15a               2011 July 15 at 10:40 a.m.

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

        It is now:  2011 Jul 17 at 2:14

        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 Jul 17 at 2:25
```

summary

This notebook is concerned with various ways of stating that all members of a set have the same finite number of elements. In addition, some rewrite rules involving the upper bound relation of the equipollence relation are derived.

invariant classes

A class **x** is **invariant** under **Q** if the inclusion **image[Q, x] ⊂ x** holds. A **final segment** for **Q** is any class of the form **image[Q, x]**.

Theorem. Every class invariant under **Q** is a final segment.

```
In[2]:= Map[implies[#, equal[x, image[Q, x]]] &,
          SubstTest[and, subclass[x, y], subclass[y, x], y -> image[Q, x]]] // Reverse

Out[2]= or[equal[x, image[Q, x]], not[subclass[image[Q, x], x]]] == True

In[3]:= or[equal[x_, image[Q, x_]], not[subclass[image[Q, x_], x_]]] := True
```

Comment. Except for the empty set and the singleton of the empty set, all invariant classes for **Q** are proper classes.

```
In[4]:= invar[Q]

Out[4]= succ[set[0]]
```

Theorem. The complement of any class of upper bounds is a final segment.

```
In[5]:= ImageComp[Q, complement[Q], x] // Reverse
Out[5]= image[Q, complement[ub[Q, x]]] == complement[ub[Q, x]]

In[6]:= image[Q, complement[ub[Q, x_]]] := complement[ub[Q, x]]
```

The complement of a class is invariant under a relation if and only if the class itself is invariant under the inverse relation. This observation yields the following corollaries.

Corollary.

```
In[7]:= SubstTest[invariant, u, complement[v], {u → Q, v → ub[Q, x]}]
Out[7]= subclass[cart[x, image[Q, ub[Q, x]]], Q] == True

In[8]:= subclass[cart[x_, image[Q, ub[Q, x_]]], Q] := True
```

Corollary.

```
In[9]:= SubstTest[subclass, inverse[u], inverse[v], {u -> cart[image[Q, ub[Q, x]], x], v → Q}]
Out[9]= subclass[cart[image[Q, ub[Q, x]], x], Q] == True

In[10]:= subclass[cart[image[Q, ub[Q, x_]], x_], Q] := True
```

Theorem. The class of upper bounds for any subclass **x** is a final segment.

```
In[11]:= SubstTest[implies, invariant[Q, t], equal[image[Q, t], t], t → ub[Q, x]] // Reverse
Out[11]= equal[image[Q, ub[Q, x]], ub[Q, x]] == True

In[12]:= image[Q, ub[Q, x]] := ub[Q, x]
```

One can eliminate the variable **x** in the preceding result by using **reify**.

Theorem. A formula for the composite of an equivalence relation and its corresponding upper bound relation.

```
In[13]:= SubstTest[reify, x, image[t, ub[t, x]], t → Q]
Out[13]= composite[Q, UB[Q]] == UB[Q]

In[14]:= composite[Q, UB[Q]] := UB[Q]
```

uniform classes

A class is **uniform** with respect to some property if all its members satisfy this property. In general, the statement that **t** satisfies some property can be replaced with the statement that **t** belongs to some class **p**. If the property is that each member **t ∈ x** belongs to the class **p**, then the statement that **x** is uniform with respect to **p** is equivalent to the statement that **x ⊂ p**.

```
In[15]:= assert[forall[t, implies[member[t, x], member[t, p]]]]
```

```
Out[15]= subclass[x, p]
```

When the property under consideration is that all members of a class **x** have the same size, the uniformity hypothesis can be stated as the inclusion **x ⊂ image[Q, {y}]** where **y** is any one of the members of the class **x**. The statement that all members of a class have the same size is equivalent to the statement that **x** is a clique of the equipollence relation.

```
In[16]:= assert[forall[y, implies[member[y, x], subclass[x, image[Q, set[y]]]]]]
```

```
Out[16]= subclass[cart[x, x], Q]
```

uniform finite size

If all members of a class **x** have the same finite size **nat[y] ∈ ω**, then the uniformity hypothesis can be stated as **x ⊂ image[Q, {nat[y]}]**.

Theorem. If **x ⊂ image[Q, {nat[y]}]**, then every member **t ∈ x** has **nat[y]** members.

```
In[17]:= SubstTest[implies, and[member[t, u], subclass[u, v]],
           member[t, v], {u → x, v → image[Q, set[nat[y]]]}] // Reverse
```

```
Out[17]= or[equal[card[t], nat[y]], not[member[t, x]],
           not[subclass[x, image[Q, set[nat[y]]]]]] == True
```

```
In[18]:= or[equal[card[t_], nat[y_]], not[member[t_, x_]],
           not[subclass[x_, image[Q, set[nat[y_]]]]]] := True
```

Corollary. (Eliminating the **nat** wrapper.)

```
In[19]:= SubstTest[implies, equal[y, nat[w]], or[equal[card[t], y],
            not[member[t, x]], not[subclass[x, image[Q, set[y]]]]], w → y] // Reverse
```

```
Out[19]= or[equal[y, card[t]], not[member[t, x]],
           not[member[y, omega]], not[subclass[x, image[Q, set[y]]]]] == True
```

```
In[20]:= or[equal[y_, card[t_]], not[member[t_, x_]],
           not[member[y_, omega]], not[subclass[x_, image[Q, set[y_]]]]] := True
```

inverse images of UB[Q]

The domain of the upper bound relation for **Q** is the class **cliques[Q]**.

```
In[21]:= domain[UB[Q]]
Out[21]= cliques[Q]
```

Theorem.

```
In[22]:= SubstTest[implies, equal[x, eqv[t]], equal[domain[UB[x]], cliques[x]], t → x] // Reverse
Out[22]= or[equal[cliques[x], domain[UB[x]]], not[EQUIVALENCE[x]]] == True

In[23]:= or[equal[cliques[x_], domain[UB[x_]]], not[EQUIVALENCE[x_]]] := True
```

Theorem.

```
In[24]:= SubstTest[EQUIVALENCE, composite[eqv[t], id[y]], {t → Q, y → image[Q, x]}] // Reverse
Out[24]= EQUIVALENCE[composite[Q, id[image[Q, x]]]] == True

In[25]:= EQUIVALENCE[composite[Q, id[image[Q, x_]]]] := True
```

Theorem.

```
In[26]:= SubstTest[EQUIVALENCE,
           composite[id[y], eqv[t], id[y]], {t → Q, y → image[Q, x]}] // Reverse
Out[26]= EQUIVALENCE[composite[id[image[Q, x]], Q]] == True

In[27]:= EQUIVALENCE[composite[id[image[Q, x_]], Q]] := True
```

Theorem.

```
In[28]:= (SubstTest[domain, UB[eqv[t]], t → composite[id[image[Q, x]], Q]] /. x → omega) // Reverse
Out[28]= image[inverse[UB[Q]], FINITE] == intersection[cliques[Q], P[FINITE]]

In[29]:= image[inverse[UB[Q]], FINITE] := intersection[cliques[Q], P[FINITE]]
```

Corollary.

```
In[30]:= IminComp[Q, UB[Q], omega]
Out[30]= image[inverse[UB[Q]], omega] == intersection[cliques[Q], P[FINITE]]

In[31]:= image[inverse[UB[Q]], omega] := intersection[cliques[Q], P[FINITE]]
```

Observation. The class of all sets with members of the same finite size is **cliques[Q] ⋂ P[FINITE]**.

```
In[32]:= class[x, exists[y, and[member[y, omega], subclass[x, image[Q, set[y]]]]]]

Out[32]= intersection[cliques[Q], P[FINITE]]
```

a disjointness condition

In this section it is shown that the condition that $\mathbf{x \times x \subset Q}$ and $\mathbf{x \subset FINITE}$ is equivalent to the single condition that ω is not disjoint from **ub[Q, x]**. The derivation given here uses quite a few lemmas, clever use of **nat** and **fin** wrappers, and introducing and eliminating an additional variable for a member of **x**. The case that **x** is empty is not really an exception, but at one point produces a redundant literal which is readily removed.

Theorem. Eliminating a **nat** wrapper.

```
In[33]:= SubstTest[implies, equal[y, nat[t]],
           implies[subclass[x, image[Q, set[y]]], subclass[x, FINITE]], t → y] // Reverse

Out[33]= or[not[member[y, omega]],
           not[subclass[x, image[Q, set[y]]]], subclass[x, FINITE]] == True

In[34]:= or[not[member[y_, omega]],
           not[subclass[x_, image[Q, set[y_]]]], subclass[x_, FINITE]] := True
```

Corollary. (Eliminating the variable **y**.)

```
In[35]:= Map[equal[V, #] &, SubstTest[class, y,
           or[not[member[y, omega]], not[subclass[x, image[inverse[q], set[y]]]],
            subclass[x, FINITE]], {q → Q, w → omega, t → FINITE}]]

Out[35]= or[equal[0, intersection[omega, ub[Q, x]]], subclass[x, FINITE]] == True

In[36]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[37]:= Map[equal[V, #] &, SubstTest[class, t, implies[member[t, u], subclass[P[x], v]],
           {u -> intersection[omega, ub[Q, x]], v → cliques[Q]}]]

Out[37]= or[equal[0, intersection[omega, ub[Q, x]]], subclass[cart[x, x], Q]] == True

In[38]:= (% /. x → x_) /. Equal → SetDelayed
```

For the final steps, the **fin** wrapper is used onthe variable for an element of **x**.

Lemma.

```
In[44]:= SubstTest[implies, member[u, v], not[empty[v]],
           {u → card[fin[t]], v → intersection[omega, ub[Q, x]]}] // Reverse

Out[44]= or[not[equal[0, intersection[omega, ub[Q, x]]]],
           not[subclass[x, image[Q, set[card[fin[t]]]]]]] == True
```

```
In[45]:= (% /. {t → t_, x → x_}) /. Equal → SetDelayed
```

Lemma.

```
In[49]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[p3, p4],
           implies[p4, p5], not[implies[and[p1, p2], p5]], {p1 → member[fin[t], x],
            p2 → subclass[cart[x, x], Q], p3 → subclass[x, image[Q, set[fin[t]]]],
            p4 → subclass[x, image[Q, set[card[fin[t]]]]],
            p5 → not[equal[0, intersection[omega, ub[Q, x]]]]}]] // Reverse

Out[49]= or[not[equal[0, intersection[omega, ub[Q, x]]]],
          not[member[fin[t], x]], not[subclass[cart[x, x], Q]]] == True

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

The **fin** wrapper does not need to be removed prior to eliminating the temporary variable **t**. In addition, the finiteness condition is shifted from the condition $\mathbf{t} \in \mathbf{FINITE}$ to the condition $\mathbf{x} \subset \mathbf{FINITE}$. Remarkably, all this can be accomplished at the same time.

Lemma. (Eliminate **t**.)

```
In[57]:= Map[implies[subclass[x, FINITE], #] &, Map[equal[V, #] &,
           SubstTest[class, t, implies[and[member[fin[t], x], subclass[u, v]], not[empty[w]]],
            {u → cart[x, x], v → Q, w -> intersection[omega, ub[Q, x]]}]]] // MapNotNot

Out[57]= or[equal[0, x], not[equal[0, intersection[omega, ub[Q, x]]]],
          not[subclass[x, FINITE]], not[subclass[cart[x, x], Q]]] == True

In[58]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. Removing a redundant literal.

```
In[65]:= SubstTest[and, implies[p, q], or[p, q],
          {p → equal[0, x], q -> or[not[equal[0, intersection[omega, ub[Q, x]]]],
             not[subclass[x, FINITE]], not[subclass[cart[x, x], Q]]]}]

Out[65]= or[not[equal[0, intersection[omega, ub[Q, x]]]],
          not[subclass[x, FINITE]], not[subclass[cart[x, x], Q]]] == True

In[66]:= (% /. x → x_) /. Equal → SetDelayed
```

The preceding results can finally be combined into a single rewrite rule.

Theorem.

```
In[68]:= equiv[equal[0, intersection[omega, ub[Q, x]]],
           or[not[subclass[x, FINITE]], not[subclass[cart[x, x], Q]]]] // not // not

Out[68]= True

In[70]:= equal[0, intersection[omega, ub[Q, x_]]] :=
          or[not[subclass[x, FINITE]], not[subclass[cart[x, x], Q]]]
```