uniform partitions of a finite set

Johan G. F. Belinfante
2011 July 25

```
In[1]:= SetDirectory["l:"]; << goedel.11jul24a
```

```
:Package Title: goedel.11jul24a                2011 July 24 at 9:15 p.m.

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

It is now:  2011 Jul 25 at 16:44

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 25 at 16:55
```

summary

A partition is **uniform** if its members all have the same size. In this notebook it is shown that if **x** is a uniform partition of a finite set, then the number of elements in **U[x]** is the product of the number of elements in **x** with the common size of the elements of **x**. The main ideas in deriving this result are to use a theorem about finite choice and to apply theorems about uncurrying to a certain indexed family of bijections.

f ⊂ inverse[IMAGE[SECOND]]

For each element **t ∈ x**, there is bijection from the natural number **y = card[t]** to **t**. Since the set **x** is finite, one can use a theorem about finite choice to show that there is a function **f: x ⟶ BIJ ⋂ map[y, V]** that selects one of these bjiections for each member of **x**. This function must satisfy **f(t) ∈ bij[y, t]**, and thus **image[f, {t}] ⊂ bij[y, t] ⊂ image[inverse[IMAGE[SECOND], {t}]**. Eliminating the variable **t** one obtains the condition **f ⊂ inverse[IMAGE[SECOND]]**.

In this section, a certain converse theorem is derived. For this, the assumptions that **x** be finite and that **y** be a natural number are not needed. Only the hypotheses **f ∈ map[x, BIJ ⋂ map[y, V]]** and **f ⊂ inverse[IMAGE[SECOND]]** are assumed. To save writing, the abbreviation **hyp** will be used for these two hypotheses in the comments accompanying the theorems derived. It will be shown that if **hyp** and **t ∈ x**, then **f(t) ∈ bij[y, t]**.

Lemma. An inclusion for vertical sections using a **funpart** wrapper.

```
In[2]:= SubstTest[implies, subclass[r, s], subclass[image[r, set[t]], image[s, set[t]]],
          {r → funpart[f], s → inverse[IMAGE[SECOND]]}] // Reverse

Out[2]= or[equal[t, range[APPLY[funpart[f], t]]],
          not[member[t, domain[funpart[f]]]], not[member[range[APPLY[funpart[f], t]], V]],
          not[subclass[funpart[f], inverse[IMAGE[SECOND]]]]] == True

In[3]:= (% /. {f → f_, t → t_}) /. Equal → SetDelayed
```

The preceding lemma has a redundant hypothesis that will now be eliminated.

Lemma. A sufficient condition for **range[APPLY[x, y]]** to be a set.

```
In[4]:= SubstTest[implies, member[w, V], member[range[w], V], w → APPLY[x, y]] // Reverse

Out[4]= or[member[range[APPLY[x, y]], V], not[member[y, domain[x]]]] == True

In[5]:= or[member[range[APPLY[x_, y_]], V], not[member[y_, domain[x_]]]] := True
```

Lemma.

```
In[6]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p1, p2, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 → member[t, domain[funpart[f]]],
            p2 → subclass[funpart[f], inverse[IMAGE[SECOND]]],
            p3 → member[range[APPLY[funpart[f], t]], V],
            p4 → equal[t, range[APPLY[funpart[f], t]]]}]] // Reverse

Out[6]= or[equal[t, range[APPLY[funpart[f], t]]], not[member[t, domain[funpart[f]]]],
          not[subclass[funpart[f], inverse[IMAGE[SECOND]]]]] == True

In[7]:= (% /. {f → f_, t → t_}) /. Equal → SetDelayed
```

Lemma. (Eliminate **funpart** wrapper.)

```
In[8]:= SubstTest[implies, equal[f, funpart[w]],
          or[equal[t, range[APPLY[f, t]]], not[member[t, domain[f]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]]], w → f] // Reverse

Out[8]= or[equal[t, range[APPLY[f, t]]], not[FUNCTION[f]],
          not[member[t, domain[f]]], not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

In[9]:= (% /. {f → f_, t → t_}) /. Equal → SetDelayed
```

Comment. The commenting out in the next theorem reduces execution time from 8 seconds to 2 seconds.

Theorem. If **hyp** and **t ∈ x** then **t = range[APPLY[f, t]]**.

```
In[10]:= Map[not, SubstTest[and, (*implies[p1,p4],implies[and[p1,p2],p5],*)
           implies[and[p3, p4, p5], p6], not[implies[and[p1, p2, p3], p6]],
           {p1 → member[f, map[x, w]], p2 → member[t, x],
            p3 -> subclass[f, inverse[IMAGE[SECOND]]], p4 -> FUNCTION[f],
            p5 -> member[t, domain[f]], p6 -> equal[t, range[APPLY[f, t]]]}]] // Reverse

Out[10]= or[equal[t, range[APPLY[f, t]]], not[member[f, map[x, w]]],
           not[member[t, x]], not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

In[11]:= (% /. {f → f_, w → w_, x → x_, t → t_}) /. Equal → SetDelayed
```

Lemma. (Needed only because the **GOEDEL** program has only limited equality substitution rules.)

```
In[12]:= SubstTest[implies, and[member[x, w], FUNCTION[x],
            FUNCTION[inverse[x]], equal[domain[x], y], equal[range[x], z]],
           member[x, bij[y, z]], {x → APPLY[f, t], z → t}] // Reverse

Out[12]= or[member[APPLY[f, t], bij[y, t]], not[equal[t, range[APPLY[f, t]]]],
           not[equal[y, domain[APPLY[f, t]]]], not[FUNCTION[APPLY[f, t]]],
           not[FUNCTION[inverse[APPLY[f, t]]]], not[member[APPLY[f, t], w]]] == True

In[13]:= (% /. {f → f_, t → t_, y → y_, w → w_}) /. Equal → SetDelayed
```

Lemma. A sufficient condition for **y** to belong to **domain[APPLY[f, t]]**.

```
In[14]:= Map[not, SubstTest[and, implies[p3, p5], implies[and[p1, p3], p6],
            implies[and[p5, p6], p9], not[implies[and[p1, p3], p9]],
            {p1 → member[t, x], p3 -> member[f, map[x, intersection[BIJ, map[y, V]]]],
             p5 → subclass[range[f], map[y, V]], p6 → member[APPLY[f, t], range[f]],
             p9 -> equal[y, domain[APPLY[f, t]]]}]] // Reverse

Out[14]= or[equal[y, domain[APPLY[f, t]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]], not[member[t, x]]] == True

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

Theorem. If **hyp** and **t ∈ x**, then **APPLY[f, t] ∈ bij[y, t]**.

```
In[16]:= Map[not, SubstTest[and, (*implies[and[p1,p3],p6],implies[and[p1,p3],p7],
             implies[and[p1,p3],p8],implies[and[p1,p3],p9],implies[and[p1,p2,p3],p10], *)
             implies[and[p6, p7, p8, p9, p10], p11], not[implies[and[p1, p2, p3], p11]],
            {p1 → member[t, x], p2 -> subclass[f, inverse[IMAGE[SECOND]]],
             p3 -> member[f, map[x, intersection[BIJ, map[y, V]]]],
             p6 → member[APPLY[f, t], range[f]], p7 → FUNCTION[APPLY[f, t]],
             p8 → FUNCTION[inverse[APPLY[f, t]]], p9 -> equal[y, domain[APPLY[f, t]]],
             p10 → equal[range[APPLY[f, t]], t], p11 → member[APPLY[f, t], bij[y, t]]}] // Reverse

Out[16]= or[member[APPLY[f, t], bij[y, t]], not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[member[t, x]], not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

In[17]:= or[member[APPLY[f_, t_], bij[y_, t_]],
           not[member[f_, map[x_, intersection[BIJ, map[y_, V]]]]],
           not[member[t_, x_]], not[subclass[f_, inverse[IMAGE[SECOND]]]]] := True
```

f maps x to map[y, U[x]]

In this section it will now be shown that **hyp** implies that **f: x ⟶ map[y, U[x]]**.

Lemma. If **hyp** and **t ∈ x**, then **APPLY[f,t] ∈ map[y, U[x]]**.

```
In[18]:= Map[not, SubstTest[and, implies[and[p0, p1], p2], implies[p1, p3],
           implies[p3, p4], implies[and[p2, p4], p5], not[implies[and[p0, p1], p5]],
           {p0 → and[member[f, map[x, intersection[BIJ, map[y, V]]]], subclass[f,
               inverse[IMAGE[SECOND]]]], p1 → member[t, x], p2 -> member[APPLY[f, t], bij[y, t]],
            p3 → subclass[t, U[x]], p4 → subclass[bij[y, t], map[y, U[x]]],
            p5 -> member[APPLY[f, t], map[y, U[x]]]}]] // Reverse
```

```
Out[18]= or[member[APPLY[f, t], map[y, U[x]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[member[t, x]], not[subclass[f, inverse[IMAGE[SECOND]]]]] == True
```

```
In[19]:= or[member[APPLY[f_, t_], map[y_, U[x_]]],
           not[member[f_, map[x_, intersection[BIJ, map[y_, V]]]]],
           not[member[t_, x_]], not[subclass[f_, inverse[IMAGE[SECOND]]]]] := True
```

To help with eliminating the variable **t**, the preceding result will be restated, replacing function application with an inclusion for a vertical section.

Lemma.

```
In[20]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p1, p2], p4],
           implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]],
           {p1 → and[member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]]], p2 → member[t, x],
            p3 -> FUNCTION[f], p4 -> member[APPLY[f, t], map[y, U[x]]],
            p5 → subclass[image[f, set[t]], map[y, U[x]]]}]] // Reverse
```

```
Out[20]= or[not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[member[t, x]], not[subclass[f, inverse[IMAGE[SECOND]]]],
           subclass[image[f, set[t]], map[y, U[x]]]] == True
```

```
In[21]:= % /. {f → f_, x → x_, y → y_, t → t_} /. Equal → SetDelayed
```

One can now eliminate **t**.

Theorem. **hyp ⟹ image[f, x] ⊂ map[y, U[x]]**.

```
In[22]:= Map[equal[V, #] &, SubstTest[class, t, or[subclass[image[f, set[t]], r],
           not[member[f, u]], not[member[t, x]], not[subclass[f, v]]], {r -> map[y, U[x]],
           u -> map[x, intersection[BIJ, map[y, V]]], v -> inverse[IMAGE[SECOND]]}]]
```

```
Out[22]= or[not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]], subclass[image[f, x], map[y, U[x]]]] == True
```

```
In[23]:= % /. {f → f_, x → x_, y → y_} /. Equal → SetDelayed
```

Theorem. **hyp ⟹ f ∈ map[x, map[y, U[x]]]**.

```
In[24]:= SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p1, p4],
           implies[p1, p5], implies[p1, p6], implies[and[p5, p6], p7],
           {p1 -> and[member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]]], p2 → member[f, V], p3 → FUNCTION[f],
            p4 → equal[domain[f], x], p5 → subclass[image[f, x], map[y, U[x]]],
            p6 → equal[range[f], image[f, x]], p7 → subclass[range[f], map[y, U[x]]]}] // MapNotNo
```

```
Out[24]= or[member[f, map[x, map[y, U[x]]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]]] == True
```

```
In[25]:= or[member[f_, map[x_, map[y_, U[x_]]]],
           not[member[f_, map[x_, intersection[BIJ, map[y_, V]]]]],
           not[subclass[f_, inverse[IMAGE[SECOND]]]]] := True
```

---

uncurrying f

This section is devoted to studying the function **APPLY[inverse[CURRY], f]**, which for convenience will be denoted by **b** in the comments. In this section it is shown that that all left-multiplications by **b** are one-to-one, which is equivalent to saying that **rotate[b]** is a function.

Lemma. **hyp ⟹ y = 0 or b ∈ map[x × y, U[x]]**.

```
In[26]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[and[p2, p3], p4], not[implies[and[p1, p3], p4]],
            {p1 → and[member[f, map[x, intersection[BIJ, map[y, V]]]],
               subclass[f, inverse[IMAGE[SECOND]]]],
             p2 -> member[f, map[x, map[y, U[x]]]], p3 → not[empty[y]],
             p4 → member[APPLY[inverse[CURRY], f], map[cart[x, y], U[x]]]}]] // Reverse
```

```
Out[26]= or[equal[0, y], member[APPLY[inverse[CURRY], f], map[cart[x, y], U[x]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]]] == True
```

```
In[27]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

A left multiplication formula holds for currying will now be used. A variable **t** for a member of **x** is introduced here, and will later have to be eliminated again.

Theorem. **t ∈ x & hyp ⟹ y = 0 or APPLY[f, t] = b ∘ LEFT[t]**.

```
In[28]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 → and[member[t, x], member[f, map[x, intersection[BIJ, map[y, V]]]], subclass[f,
               inverse[IMAGE[SECOND]]], not[empty[y]]], p2 -> member[f, map[x, map[y, U[x]]]],
            p3 -> equal[APPLY[f, t], composite[APPLY[inverse[CURRY], f], LEFT[t]]]}]] // Reverse

Out[28]= or[equal[0, y], equal[APPLY[f, t], composite[APPLY[inverse[CURRY], f], LEFT[t]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[member[t, x]], not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

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

Theorem. $\mathbf{t \in x \ \&\ hyp \implies y = 0 \ or \ b \circ LEFT[t] \in bij[y, t]}$.

```
In[30]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
           {p1 -> and[member[t, x], member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]],
            p2 -> equal[APPLY[f, t], composite[APPLY[inverse[CURRY], f], LEFT[t]]],
            p3 → member[APPLY[f, t], bij[y, t]],
            p4 → member[composite[APPLY[inverse[CURRY], f], LEFT[t]], bij[y, t]]}]] // Reverse

Out[30]= or[equal[0, y], member[composite[APPLY[inverse[CURRY], f], LEFT[t]], bij[y, t]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[member[t, x]], not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

In[31]:= (% /. {f → f_, t → t_, x → x_, y → y_}) /. Equal→ SetDelayed
```

Lemma.

```
In[32]:= SubstTest[implies, member[z, bij[x, y]],
           FUNCTION[inverse[z]], z → composite[w, LEFT[t]]] // Reverse

Out[32]= or[FUNCTION[composite[inverse[LEFT[t]], inverse[w]]],
           not[member[composite[w, LEFT[t]], bij[x, y]]]] == True

In[33]:= (% /. {t → t_, w → w_, x → x_, y → y_}) /. Equal→ SetDelayed
```

Theorem. $\mathbf{t \in x \ \&\ hyp \implies y = 0 \ or \ FUNCTION[inverse[LEFT[t]] \circ inverse[b]]}$.

```
In[34]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> and[member[t, x], member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]],
            p2 → member[composite[APPLY[inverse[CURRY], f], LEFT[t]], bij[y, t]], p3 → FUNCTION[
              composite[inverse[LEFT[t]], inverse[APPLY[inverse[CURRY], f]]]]}]] // Reverse

Out[34]= or[equal[0, y],
           FUNCTION[composite[inverse[LEFT[t]], inverse[APPLY[inverse[CURRY], f]]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[member[t, x]], not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

In[35]:= (% /. {t → t_, f → f_, x → x_, y → y_}) /. Equal→ SetDelayed
```

It will next be shown that the literal $\mathbf{t \in x}$ in the above is redundant.

Lemma.

```
In[36]:= Map[not, SubstTest[and, implies[p1, p3],
           implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 → member[w, map[cart[x, y], z]], p2 → not[empty[y]],
            p3 → equal[domain[w], cart[x, y]], p4 → equal[domain[domain[w]], x]}]] // Reverse

Out[36]= or[equal[0, y], equal[x, domain[domain[w]]], not[member[w, map[cart[x, y], z]]]] == True

In[37]:= or[equal[0, y_], equal[x_, domain[domain[w_]]],
           not[member[w_, map[cart[x_, y_], z_]]]] := True
```

Theorem.

```
In[38]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p2, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 → member[w, map[x, map[y, z]]],
            p2 → not[empty[y]], p3 → member[APPLY[inverse[CURRY], w], map[cart[x, y], z]],
            p4 → equal[domain[domain[APPLY[inverse[CURRY], w]]], x]}]] // Reverse

Out[38]= or[equal[0, y], equal[x, domain[domain[APPLY[inverse[CURRY], w]]]],
           not[member[w, map[x, map[y, z]]]]] == True

In[39]:= or[equal[0, y_], equal[x_, domain[domain[APPLY[inverse[CURRY], w_]]]],
           not[member[w_, map[x_, map[y_, z_]]]]] := True
```

Lemma.

```
In[40]:= SubstTest[implies, empty[w], FUNCTION[w],
           w -> composite[inverse[LEFT[t]], inverse[APPLY[inverse[CURRY], f]]] // Reverse

Out[40]= or[FUNCTION[composite[inverse[LEFT[t]], inverse[APPLY[inverse[CURRY], f]]]],
           member[t, domain[domain[APPLY[inverse[CURRY], f]]]]] == True

In[41]:= (% /. {t → t_, f → f_}) /. Equal → SetDelayed
```

The redundant literal can now be removed.

Theorem. **hyp** $\Longrightarrow$ **y = 0 or FUNCTION[inverse[LEFT[t]] ∘ inverse[b]]**.

```
In[42]:= Map[not,
           SubstTest[and, implies[and[p0, p1], p4], implies[p1, p2], implies[and[p1, p2], p3],
            (*implies[and[not[p0],p3],p4], *) not[implies[p1, p4]], {p0 → member[t, x],
            p1 -> and[member[f, map[x, intersection[BIJ, map[y, V]]]], subclass[f,
               inverse[IMAGE[SECOND]]], not[empty[y]]], p2 -> member[f, map[x, map[y, U[x]]]],
            p3 -> equal[x, domain[domain[APPLY[inverse[CURRY], f]]]], p4 → FUNCTION[
              composite[inverse[LEFT[t]], inverse[APPLY[inverse[CURRY], f]]]]}]] // Reverse

Out[42]= or[equal[0, y],
           FUNCTION[composite[inverse[LEFT[t]], inverse[APPLY[inverse[CURRY], f]]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

In[43]:= (% /. {t → t_, f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

At this point the variable **t** will be eliminated.

Theorem. **hyp ⟹ y = 0 or FUNCTION[rotate[b]]**.

```
In[44]:= Map[equal[V, #] &, SubstTest[class, t,
           or[empty[y], not[member[f, u]], not[subclass[f, v]], not[member[t, z]]],
           {z → domain[fix[composite[inverse[APPLY[inverse[CURRY], f]],
                APPLY[inverse[CURRY], f], cross[Id, Di]]]],
            u → map[x, intersection[BIJ, map[y, V]]], v -> inverse[IMAGE[SECOND]]}]]

Out[44]= or[equal[0, y], FUNCTION[rotate[APPLY[inverse[CURRY], f]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]] == True

In[45]:= (% /. {f → f_, x → x_, y → y_}) /. Equal→ SetDelayed
```

b is one-to-one when x is pairwise disjoint

In this section it is shown that **hyp** implies that **y = 0** or **FIRST ∘ inverse[b] = id[x] ∘ E**. From this it is deduced that **b** is one-to-one when **x** is pairwise disjoint. The variable **t** is reintroduced, and will again have to be eliminated later on.

Lemma.

```
In[46]:= SubstTest[implies, member[z, bij[y, t]],
           equal[range[z], t], {z -> composite[w, LEFT[t]]}] // Reverse

Out[46]= or[equal[t, image[w, cart[set[t], V]]],
           not[member[composite[w, LEFT[t]], bij[y, t]]]] == True

In[47]:= (% /. {t → t_, w → w_, y → y_}) /. Equal→ SetDelayed
```

Theorem. **t ∈ x & hyp ⟹ y = 0 or t = image[b, {t} × V]**.

```
In[48]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
            {p1 -> and[member[t, x], member[f, map[x, intersection[BIJ, map[y, V]]]],
               subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]],
             p2 -> member[composite[APPLY[inverse[CURRY], f], LEFT[t]], bij[y, t]],
             p3 -> equal[t, image[APPLY[inverse[CURRY], f], cart[set[t], V]]]}]] // Reverse

Out[48]= or[equal[0, y], equal[t, image[APPLY[inverse[CURRY], f], cart[set[t], V]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[member[t, x]], not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

In[49]:= (% /. {f → f_, t → t_, x → x_, y → y_}) /. Equal→ SetDelayed
```

Eliminating the variable **t** will be done using two lemmas.

Lemma.

```
In[50]:= member[t, fix[composite[complement[E], w]]] // AssertTest

Out[50]= member[t, fix[composite[complement[E], w]]] == not[subclass[image[w, set[t]], t]]

In[51]:= member[t_, fix[composite[complement[E], w_]]] := not[subclass[image[w, set[t]], t]]
```

Lemma.

```
In[52]:= member[t, fix[composite[w, inverse[E]]]] // InvertFixTest

Out[52]= member[t, fix[composite[w, inverse[E]]]] == member[t, image[w, t]]

In[53]:= member[t_, fix[composite[w_, inverse[E]]]] := member[t, image[w, t]]
```

The elimination of **t** must be done twice, obtaining two inclusions that can then be combined into an equation.

Theorem. **hyp ⟹ y = 0 or id[x] ∘ E ⊂ FIRST ∘ inverse[b]**.

```
In[54]:= Map[or[equal[V, #], subclass[composite[id[x], E],
             composite[FIRST, inverse[APPLY[inverse[CURRY], f]]]]] &,
           SubstTest[class, t, or[equal[0, y], member[t, w], not[member[f, u]],
             not[member[t, x]], not[subclass[f, v]]],
            {u -> map[x, intersection[BIJ, map[y, V]]], v → inverse[IMAGE[SECOND]],
             w → intersection[complement[fix[composite[complement[composite[FIRST,
                    inverse[APPLY[inverse[CURRY], f]]]], inverse[E]]]], complement[
               fix[composite[complement[E], APPLY[inverse[CURRY], f], inverse[FIRST]]]]]}]]

Out[54]= or[equal[0, y], not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]], subclass[composite[id[x], E],
            composite[FIRST, inverse[APPLY[inverse[CURRY], f]]]] == True

In[55]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

The second elimination of **t** yields an inclusion in the opposite direction. This takes several steps.

Lemma.

```
In[56]:= Map[equal[V, #] &, SubstTest[class, t, or[equal[0, y],
              member[t, w], not[member[f, u]], not[member[t, x]], not[subclass[f, v]]],
             {u -> map[x, intersection[BIJ, map[y, V]]], v → inverse[IMAGE[SECOND]],
              w → intersection[complement[fix[composite[complement[composite[FIRST, inverse[
                      APPLY[inverse[CURRY], f]]]], inverse[E]]]], complement[fix[composite[
                   complement[E], APPLY[inverse[CURRY], f], inverse[FIRST]]]]]}]] // MapNotNot

Out[56]= or[equal[0, y], not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]], subclass[
            composite[FIRST, id[cart[x, V]], inverse[APPLY[inverse[CURRY], f]]], E]] == True

In[57]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[58]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p3, p4],
           not[implies[p1, p4]], {p1 -> and[member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]],
            p2 → member[APPLY[inverse[CURRY], f], map[cart[x, y], U[x]]],
            p3 → equal[domain[APPLY[inverse[CURRY], f]], cart[x, y]],
            p4 -> subclass[domain[APPLY[inverse[CURRY], f]], cart[x, V]]}]] // Reverse

Out[58]= or[equal[0, y], not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]],
           subclass[domain[APPLY[inverse[CURRY], f]], cart[x, V]]] == True

In[59]:= (% /. {f → f_, x → x_, y → y_}) /. Equal→ SetDelayed
```

Lemma.

```
In[60]:= SubstTest[implies,
           equal[t, composite[id[cart[x, V]], inverse[APPLY[inverse[CURRY], f]]]],
           or[equal[0, y], not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
            not[subclass[f, inverse[IMAGE[SECOND]]]], subclass[composite[FIRST, t], E]],
           t -> inverse[APPLY[inverse[CURRY], f]]] // Reverse

Out[60]= or[equal[0, y], not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]],
           not[subclass[domain[APPLY[inverse[CURRY], f]], cart[x, V]]],
           subclass[composite[FIRST, inverse[APPLY[inverse[CURRY], f]]], E]] == True

In[61]:= (% /. {f → f_, x → x_, y → y_}) /. Equal→ SetDelayed
```

Lemma.

```
In[62]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
           not[implies[p1, p3]], {p1 -> and[member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]],
            p2 -> subclass[domain[APPLY[inverse[CURRY], f]], cart[x, V]],
            p3 -> subclass[composite[FIRST, inverse[APPLY[inverse[CURRY], f]]], E]}]] // Reverse

Out[62]= or[equal[0, y], not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]],
           subclass[composite[FIRST, inverse[APPLY[inverse[CURRY], f]]], E]] == True

In[63]:= (% /. {f → f_, x → x_, y → y_}) /. Equal→ SetDelayed
```

Lemma.

```
In[64]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p3, p4],
           not[implies[p1, p4]], {p1 -> and[member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]],
            p2 → member[APPLY[inverse[CURRY], f], map[cart[x, y], U[x]]],
            p3 → equal[domain[APPLY[inverse[CURRY], f]], cart[x, y]],
            p4 → subclass[domain[domain[APPLY[inverse[CURRY], f]]], x]}]] // Reverse

Out[64]= or[equal[0, y], not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]],
           subclass[domain[domain[APPLY[inverse[CURRY], f]]], x]] == True

In[65]:= (% /. {f → f_, x → x_, y → y_}) /. Equal→ SetDelayed
```

The two inclusions are now combined to produce an equation.

Theorem. **hyp** $\Longrightarrow$ **y = 0 or FIRST ∘ inverse[b] = id[x] ∘ E**.

```
In[66]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
           {p -> and[member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]], u -> composite[id[x], E],
            v -> composite[FIRST, inverse[APPLY[inverse[CURRY], f]]]}] // MapNotNot

Out[66]= or[equal[0, y],
           equal[composite[FIRST, inverse[APPLY[inverse[CURRY], f]]], composite[id[x], E]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

In[67]:= or[equal[0, y_], equal[composite[FIRST, inverse[APPLY[inverse[CURRY], f_]]],
            composite[id[x_], E]], not[member[f_, map[x_, intersection[BIJ, map[y_, V]]]]],
           not[subclass[f_, inverse[IMAGE[SECOND]]]]] := True
```

At this point the following two rewrite rules will be used:

```
In[68]:= and[FUNCTION[rotate[b]], FUNCTION[composite[FIRST, inverse[b]]]]

Out[68]= FUNCTION[composite[id[cart[V, V]], inverse[b]]]

In[69]:= FUNCTION[composite[id[x], E]]

Out[69]= subclass[cart[x, x], union[DISJOINT, Id]]
```

Lemma.

```
In[70]:= Map[not, SubstTest[and, implies[p1, p3], implies[and[p2, p3], p4],
           implies[p1, p5], (*implies[and[p4,p5],p6],*) not[implies[and[p1, p2], p6]],
           {p1 -> and[member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]],
            p2 → FUNCTION[composite[id[x], E]], p3 -> equal[
              composite[FIRST, inverse[APPLY[inverse[CURRY], f]]], composite[id[x], E]],
            p4 → FUNCTION[composite[FIRST, inverse[APPLY[inverse[CURRY], f]]]],
            p5 → FUNCTION[rotate[APPLY[inverse[CURRY], f]]], p6 → FUNCTION[
              composite[id[cart[V, V]], inverse[APPLY[inverse[CURRY], f]]]]}]] // Reverse

Out[70]= or[equal[0, y], FUNCTION[composite[id[cart[V, V]], inverse[APPLY[inverse[CURRY], f]]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]],
           not[subclass[cart[x, x], union[DISJOINT, Id]]]] == True

In[71]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma.

```
In[72]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], implies[p3, p4],
           not[implies[p1, p4]], {p1 -> and[member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]],
            p2 → member[APPLY[inverse[CURRY], f], map[cart[x, y], U[x]]],
            p3 → equal[domain[APPLY[inverse[CURRY], f]], cart[x, y]],
            p4 → subclass[domain[APPLY[inverse[CURRY], f]], cart[V, V]]}]] // Reverse

Out[72]= or[equal[0, y], not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]],
           subclass[domain[APPLY[inverse[CURRY], f]], cart[V, V]]] == True

In[73]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. **hyp & $x \times x \subset \text{DISJOINT} \cup \text{Id} \implies y = 0$ or FUNCTION[inverse[b]]**.

```
In[74]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
           {p1 → and[member[f, map[x, intersection[BIJ, map[y, V]]]], subclass[f, inverse[
                IMAGE[SECOND]]], not[empty[y]], subclass[cart[x, x], union[DISJOINT, Id]]],
            p2 -> FUNCTION[composite[id[cart[V, V]], inverse[APPLY[inverse[CURRY], f]]]],
            p3 → subclass[domain[APPLY[inverse[CURRY], f]], cart[V, V]],
            p4 → FUNCTION[inverse[APPLY[inverse[CURRY], f]]]}]] // Reverse

Out[74]= or[equal[0, y], FUNCTION[inverse[APPLY[inverse[CURRY], f]]],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]],
           not[subclass[cart[x, x], union[DISJOINT, Id]]]] == True

In[75]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

an equipollence theorem

In this section it is shown that when **x** is pairwise disjoint, **hyp** implies that the sets **x × y** and **U[x]** are equipollent.

Lemma.

```
In[76]:= SubstTest[implies, equal[u, v], equal[domain[u], domain[v]], {u → composite[id[x], E],
           v → composite[FIRST, inverse[APPLY[inverse[CURRY], f]]]}] // Reverse

Out[76]= or[equal[image[APPLY[inverse[CURRY], f], cart[V, V]], U[x]], not[equal[
            composite[FIRST, inverse[APPLY[inverse[CURRY], f]]], composite[id[x], E]]]] == True

In[77]:= (% /. {f → f_, x → x_}) /. Equal → SetDelayed
```

Lemma.

```
In[78]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           implies[p1, p4], implies[and[p3, p4], p5], not[implies[p1, p5]],
           {p1 -> and[member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]], p2 ->
             equal[composite[FIRST, inverse[APPLY[inverse[CURRY], f]]], composite[id[x], E]],
            p3 -> equal[image[APPLY[inverse[CURRY], f], cart[V, V]], U[x]],
            p4 -> subclass[domain[APPLY[inverse[CURRY], f]], cart[V, V]],
            p5 → equal[range[APPLY[inverse[CURRY], f]], U[x]]}]] // Reverse

Out[78]= or[equal[0, y], equal[range[APPLY[inverse[CURRY], f]], U[x]],
          not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
          not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

In[79]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

Technical Lemma.

```
In[80]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → and[member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]]], p2 → subclass[range[f], map[y, U[x]]],
            p3 → implies[not[empty[y]], not[member[0, range[f]]]]}]] // Reverse

Out[80]= or[equal[0, y], not[member[0, range[f]]],
          not[member[f, map[x, intersection[BIJ, map[y, U[x]]]]]],
          not[subclass[f, inverse[IMAGE[SECOND]]]]] == True

In[81]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

The next lemma exposes what is needed to prove equipollence.

Lemma.

```
In[82]:= Map[implies[member[f, z], #] &, SubstTest[implies, member[t, BIJ],
           member[pair[domain[t], range[t]], Q], t → APPLY[inverse[CURRY], f]]] // Reverse

Out[82]= or[member[0, range[f]],
          member[pair[domain[APPLY[inverse[CURRY], f]], range[APPLY[inverse[CURRY], f]]], Q],
          not[FUNCTION[f]], not[FUNCTION[APPLY[inverse[CURRY], f]]],
          not[FUNCTION[inverse[APPLY[inverse[CURRY], f]]]],
          not[member[f, z]], not[subclass[U[range[f]], cart[V, V]]] == True

In[83]:= (% /. {f → f_, z → z_}) /. Equal → SetDelayed
```

Technical lemma.

```
In[84]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → member[f, map[x, intersection[BIJ, map[y, V]]]], p2 →
             subclass[range[f], map[y, V]], p3 → subclass[U[range[f]], cart[V, V]]}]] // Reverse

Out[84]= or[not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
          subclass[U[range[f]], cart[V, V]] == True

In[85]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. (This takes a long time.)

```
In[86]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
           (* implies[p1,p4],implies[p2,p5], implies[p1,p6], implies[p1,p7], *)
           implies[and[p1, p3, p4, p5, p6, p7], p8] , not[implies[p1, p8]],
           {p1 -> and[subclass[cart[x, x], union[DISJOINT, Id]],
              member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]],
            p2 → member[APPLY[inverse[CURRY], f], map[cart[x, y], U[x]]],
            p3 → not[member[0, range[f]]], p4 -> FUNCTION[f],
            p5 -> FUNCTION[APPLY[inverse[CURRY], f]],
            p6 -> FUNCTION[inverse[APPLY[inverse[CURRY], f]]],
            p7 -> subclass[U[range[f]], cart[V, V]],
            p8 -> member[pair[domain[APPLY[inverse[CURRY], f]],
               range[APPLY[inverse[CURRY], f]]], Q]}]] // Reverse

Out[86]= or[equal[0, y],
          member[pair[domain[APPLY[inverse[CURRY], f]], range[APPLY[inverse[CURRY], f]]], Q],
          not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
          not[subclass[f, inverse[IMAGE[SECOND]]]],
          not[subclass[cart[x, x], union[DISJOINT, Id]]] == True

In[87]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

Major theorem. **hyp & $x \times x \subset \mathrm{DISJOINT} \cup \mathrm{Id} \implies y = 0$ or pair[x × y, U[x]] ∈ Q.**

```
In[88]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[p3, p4], implies[p1, p5], implies[and[p2, p4, p5], p6],
           not[implies[p1, p6]], {p1 -> and[subclass[cart[x, x], union[DISJOINT, Id]],
              member[f, map[x, intersection[BIJ, map[y, V]]]],
              subclass[f, inverse[IMAGE[SECOND]]], not[empty[y]]], p2 -> member[
              pair[domain[APPLY[inverse[CURRY], f]], range[APPLY[inverse[CURRY], f]]], Q],
            p3 → member[APPLY[inverse[CURRY], f], map[cart[x, y], U[x]]],
            p4 → equal[domain[APPLY[inverse[CURRY], f]], cart[x, y]],
            p5 → equal[range[APPLY[inverse[CURRY], f]], U[x]],
            p6 → member[pair[cart[x, y], U[x]], Q]}]] // Reverse

Out[88]= or[equal[0, y], member[pair[cart[x, y], U[x]], Q],
           not[member[f, map[x, intersection[BIJ, map[y, V]]]]],
           not[subclass[f, inverse[IMAGE[SECOND]]]],
           not[subclass[cart[x, x], union[DISJOINT, Id]]]] == True

In[89]:= (% /. {f → f_, x → x_, y → y_}) /. Equal → SetDelayed
```

One can now eliminate the variable **f**.

Corollary.

```
In[90]:= Map[equal[V, #] &, SubstTest[class, f, or[equal[0, y], member[p, q],
            not[member[f, u]], not[subclass[f, v]], not[subclass[P[x], z]]],
           {p -> pair[cart[x, y], U[x]], q → Q, u -> map[x, intersection[BIJ, map[y, V]]],
            v -> inverse[IMAGE[SECOND]], w → x, z → cliques[union[DISJOINT, Id]]}]]

Out[90]= or[equal[0, y], equal[0, intersection[map[x, intersection[BIJ, map[y, V]]],
             P[inverse[IMAGE[SECOND]]]]], member[pair[cart[x, y], U[x]], Q],
           not[subclass[cart[x, x], union[DISJOINT, Id]]]] == True

In[91]:= or[equal[0,
            intersection[map[x_, intersection[BIJ, map[y_, V]]], P[inverse[IMAGE[SECOND]]]]],
           equal[0, y_], member[pair[cart[x_, y_], U[x_]], Q],
           not[subclass[cart[x_, x_], union[DISJOINT, Id]]]] := True
```

finite uniform partitions

Lemma. (Application of a theorem about finite choice to uniform partitions.)

```
In[92]:= SubstTest[or, not[equal[0, intersection[map[x, t], P[z]]]],
           not[member[x, FINITE]], not[subclass[x, image[inverse[z], t]]],
           {t → intersection[BIJ, map[y, V]], z → inverse[IMAGE[SECOND]]}] // Reverse

Out[92]= or[not[equal[0,
             intersection[map[x, intersection[BIJ, map[y, V]]], P[inverse[IMAGE[SECOND]]]]]],
           not[member[x, FINITE]], not[subclass[x, image[Q, set[y]]]]] == True
```

```
In[93]:= or[not[equal[0,
             intersection[map[x_, intersection[BIJ, map[y_, V]]], P[inverse[IMAGE[SECOND]]]]]],
           not[member[x_, FINITE]], not[subclass[x_, image[Q, set[y_]]]]] := True
```

Main Theorem. If each member of a finite set **x** of pairwise disjoint sets is equipollent to **y**, then **x × y** is equipollent to **U[x]**.

```
In[94]:= Map[not, SubstTest[and, implies[and[p1, p2], p4], implies[and[p0, p2, p3, p4], p5],
            implies[p2, or[p0, p5]], not[implies[and[p1, p2, p3], p5]],
            {p0 → not[empty[y]], p1 → member[x, FINITE], p2 -> subclass[x, image[Q, set[y]]],
             p3 → subclass[cart[x, x], union[DISJOINT, Id]], p4 → not[equal[0, intersection[
                 map[x, intersection[BIJ, map[y, V]]], P[inverse[IMAGE[SECOND]]]]]],
             p5 -> member[pair[cart[x, y], U[x]], Q]}]] // Reverse
```

```
Out[94]= or[member[pair[cart[x, y], U[x]], Q],
           not[member[x, FINITE]], not[subclass[x, image[Q, set[y]]]],
           not[subclass[cart[x, x], union[DISJOINT, Id]]]] == True
```

```
In[95]:= or[member[pair[cart[x_, y_], U[x_]], Q],
           not[member[x_, FINITE]], not[subclass[x_, image[Q, set[y_]]]],
           not[subclass[cart[x_, x_], union[DISJOINT, Id]]]] := True
```

Theorem.

```
In[96]:= SubstTest[or, member[pair[cart[u, v], U[u]], Q],
           not[member[u, FINITE]], not[subclass[u, image[Q, set[v]]]],
           not[subclass[cart[u, u], union[DISJOINT, Id]]], {u → fin[x], v → nat[y]}] // Reverse
```

```
Out[96]= or[equal[card[U[fin[x]]], natmul[card[fin[x]], nat[y]]],
           not[subclass[cart[fin[x], fin[x]], union[DISJOINT, Id]]],
           not[subclass[fin[x], image[Q, set[nat[y]]]]]] == True
```

```
In[97]:= or[equal[card[U[fin[x_]]], natmul[card[fin[x_]], nat[y_]]],
           not[subclass[cart[fin[x_], fin[x_]], union[DISJOINT, Id]]],
           not[subclass[fin[x_], image[Q, set[nat[y_]]]]]] := True
```

Corollary.

```
In[98]:= SubstTest[implies, and[equal[x, fin[u]], equal[y, nat[v]]],
           or[equal[card[U[x]], natmul[card[x], y]],
            not[subclass[cart[x, x], union[DISJOINT, Id]]],
            not[subclass[x, image[Q, set[y]]]]], {u → x, v → y}] // Reverse
```

```
Out[98]= or[equal[card[U[x]], natmul[y, card[x]]], not[member[x, FINITE]],
           not[member[y, omega]], not[subclass[x, image[Q, set[y]]]],
           not[subclass[cart[x, x], union[DISJOINT, Id]]]] == True
```

```
In[99]:= or[equal[card[U[x_]], natmul[card[x_], y_]], not[member[x_, FINITE]],
           not[member[y_, omega]], not[subclass[cart[x_, x_], union[DISJOINT, Id]]],
           not[subclass[x_, image[Q, set[y_]]]]] := True
```