a pigeonhole principle

Johan G. F. Belinfante
2012 February 21

The following general formulation of the pigeonhole principle is derived: If the domain and range of a finite function are equipollent, then the function is one-to-one. A simple variable-free version of this is also obtained. The main idea of the derivation is to use the theorem of finite choice, which says that any finite set admits a cross-section. The same idea was used by the author in collaboration with Ming Li to prove the special case in which the equipollence relation $Q$ is replaced with the identity relation $Id$. The final result obtained for that special case can now be stated succinctly as follows:

$$\text{In}[2]:= \text{intersection}[\text{FINITE, FUNS, image}\{\text{inverse}[\text{DORA}], \text{Id}\}]$$

$$\text{Out}[2]= \text{intersection}[\text{FINITE, PERMS}]$$

cross-sections of inverses

Statements about cross-sections of $\text{inverse}[x]$ are currently rewritten as mapping statements. This rewriting blocks a direct translation of the proof obtained by the author and Ming Li in 2006. This rewriting is avoided here by leaving out the sethood hypothesis $y \in V$. Later on, when the variable $y$ is eliminated, sethood facts reintroduce the map constructor, but the following simplification rule takes care of that.

Theorem. Simplification rule.

$$\text{In}[3]= X[\text{inverse}[x]] // \text{Normality} // \text{Reverse}$$

$$\text{Out}[3]= \text{intersection}[\text{map}[\text{range}[x], V], P[\text{inverse}[x]]] \Rightarrow X[\text{inverse}[x]]$$
Any cross-section of a finite relation is finite. More generally, any subset of a finite relation is finite. The same is true for any subset of the inverse of a finite relation since the inverse of a finite relation is also finite. An explicit statement of this fact will now be derived.

Lemma.

In[5]:= `SubstTest`[implies, andsubclass[y, t], member[t,FINITE], member[y,FINITE], t -> inverse[fin[x]]] // Reverse

Out[5]= or[member[y,FINITE], notsubclass[y, inverse[fin[x]]]] = True

In[6]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

Theorem. Any subset of the inverse of a finite relation is finite.

In[7]:= `SubstTest`[implies, equal[x,fin[t]], or[member[y,FINITE], notsubclass[y, inverse[x]]], t -> x] // Reverse

Out[7]= or[member[y,FINITE], notmember[x,FINITE], notsubclass[y, inverse[x]]] = True

In[8]:= or[member[y_,FINITE], notmember[x_,FINITE], notsubclass[y_,inverse[x_]]] := True

domain and range of a bijection

Various forms of the following basic fact are available in the GOEDEL program, but the following straightforward statement seems to have been overlooked until now. It is here derived by eliminating an oopart wrapper from an available statement. (The special case y = V is also available and could have been used instead. That special case is obviously subsumed by the current result, and therefore will be removed from the GOEDEL program.)

Theorem. The domain and range of a bijection are equipollent.

In[10]:= `SubstTest`[implies, andmember[x, y], equal[x, oopart[t]]], member[pair[domain[x], range[x]], Q], t -> x] // Reverse

Out[10]= or[member[pair[domain[x], range[x]], Q], notFUNCTION[x], notFUNCTION[inverse[x]], notmember[x, y]] = True

In[11]:= or[member[pair[domain[x_], range[x_]], Q], notFUNCTION[inverse[x_]], notFUNCTION[x_], notmember[x_, y_]] := True

restrictions of functions

Two corollaries of the fact that a subclass of a function is a restriction is derived in this section. They differ only in the inclusion hypothesis.
Theorem.

\begin{verbatim}
In[12]:= SubstTest[or, equal[x, t], not[equal[domain[t], domain[x]]],
    not[FUNCTION[x]], not[subclass[t, x]], t \to inverse[y]] // Reverse
Out[12]= or[equal[x, inverse[y]], not[equal[domain[x], range[y]]],
    not[FUNCTION[x]], not[subclass[inverse[y], x]]] = True
\end{verbatim}

\begin{verbatim}
In[13]:= or[equal[inverse[y_], x_], not[equal[domain[x_], range[y_]]],
    not[FUNCTION[x_]], not[subclass[inverse[y_], x_]]] := True
\end{verbatim}

Corollary.

\begin{verbatim}
In[14]:= Map[not, SubstTest[and, implies[and[p2, p3], p4],
    implies[and[p1, p2, p4], p5], not[implies[and[p1, p2, p3], p5]],
    {p1 \to equal[domain[x], range[y]], p2 \to FUNCTION[x], p3 \to subclass[y, inverse[x]],
    p4 \to subclass[inverse[y], x], p5 \to equal[x, inverse[y]]}]] // Reverse
Out[14]= or[equal[x, inverse[y]], not[equal[domain[x], range[y]]],
    not[FUNCTION[x]], not[subclass[y, inverse[x]]]] = True
\end{verbatim}

\begin{verbatim}
In[15]:= or[equal[inverse[y_], x_], not[equal[domain[x_], range[y_]]],
    not[FUNCTION[x_]], not[subclass[y_, inverse[x_]]]] := True
\end{verbatim}

\section*{a pigeonhole principle}

The main derivation of the pigeon hole principle is done in this section. To speed up the derivation, it is convenient to break it up into a series of lemmas. In each of these lemmas, the proposition \( p_1 \) is the main hypothesis, while \( p_2 \) is a temporary hypothesis that says \( y \) is a cross-section of \( \text{inverse}[x] \).

Lemma.

\begin{verbatim}
In[16]:= Map[not, SubstTest[and, (*implies[p2,p3],implies[and[p1,p2],p4],
    implies[and[p1,p2],p5],*) implies[and[p2, p4], p5], p6],
    not[implies[and[p1, p2], p6]], {p1 \to and[member[x, FINITE], FUNCTION[x]],
    p2 \to and[FUNCTION[y], subclass[y, inverse[x]]],
    p3 \to subclass[range[y], domain[x]], p4 \to FUNCTION[inverse[y]],
    p5 \to member[y, FINITE], p6 \to member[domain[y], range[y]], Q}]] // Reverse
Out[16]= or[member[pair[domain[y], range[y]], Q], not[FUNCTION[x]],
    not[FUNCTION[y]], not[member[x, FINITE]], not[subclass[y, inverse[x]]]] = True
\end{verbatim}

\begin{verbatim}
In[17]:= (% /. \{x \to x_\_, y \to y_\_\}) /. Equal \to SetDelayed
\end{verbatim}

Lemma.
Lemma.

The lemmas are combined in the following theorem.

Theorem.

At this point, the variable \( y \) can be eliminated.
Lemma.

In[24]:= Map[equal[V, domain[#]] & , SubstTest[reify, y, 
   case[or[FUNCTION[inverse[x]], not[equal[domain[y], range[x]]]], not[FUNCTION[x]], 
   not[FUNCTION[y]], not[member[x, t]], not[member[pair[domain[x], range[x]], Q]], 
   not[subclass[y, inverse[x]]]], t -> FINITE]]

Out[24]= or[equal[0, X[inverse[x]]], FUNCTION[inverse[x]], not[FUNCTION[x]], 
      not[member[x, FINITE]], not[member[pair[domain[x], range[x]], Q]]] == True

In[25]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

The hypothesis that inverse[x] admits a cross-section is a statement of the theorem of finite choice, and is therefore redundant.

Main Theorem. The pigeonhole principle.

In[26]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]], 
   {p1 -> and[FUNCTION[x], member[x, FINITE], member[pair[domain[x], range[x]], Q]], 
    p2 -> not[equal[0, X[inverse[x]]]], p3 -> FUNCTION[inverse[x]]}] // Reverse

Out[26]= or[FUNCTION[inverse[x]], not[FUNCTION[x]], 
      not[member[x, FINITE]], not[member[pair[domain[x], range[x]], Q]]] == True

In[27]:= or[FUNCTION[inverse[x_]], not[FUNCTION[x_]], 
      not[member[pair[domain[x_], range[x_]], Q]], not[member[x_, FINITE]]] := True

Since any finite set is equipollent to an ordinal, one can replace the equipollence hypothesis with a statement of equality of cardinalities.

Theorem.

In[30]:= SubstTest[implies, equal[x, fin[t]], or[member[pair[domain[x], range[x]], Q], 
   not[equal[card[domain[x]], card[range[x]]]], not[FUNCTION[x]]], t -> x] // Reverse

Out[30]= or[member[pair[domain[x], range[x]], Q], not[equal[card[domain[x]], card[range[x]]]], 
      not[FUNCTION[x]], not[member[x, FINITE]]] = True

In[31]:= or[member[pair[domain[x_], range[x_]], Q], 
      not[equal[card[domain[x_]], card[range[x_]]]], 
      not[FUNCTION[x_]], not[member[x_, FINITE]]] := True

Corollary. Restatement of the pigeonhole principle using equality of cardinality in place of equipollence.

In[34]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]], 
   {p1 -> and[equal[card[domain[x]], card[range[x]]], FUNCTION[x], member[x, FINITE]], 
    p2 -> member[pair[domain[x], range[x]], Q], p3 -> FUNCTION[inverse[x]]}] // Reverse

Out[34]= or[FUNCTION[inverse[x]], not[equal[card[domain[x]], card[range[x]]]], 
      not[FUNCTION[x]], not[member[x, FINITE]]] == True

In[35]:= or[FUNCTION[inverse[x_]], not[equal[card[domain[x_]], card[range[x_]]]], 
      not[FUNCTION[x_]], not[member[x_, FINITE]]] := True
a variable-free statement of a pigeonhole principle

Since there is no \texttt{reify} rule for \texttt{pair}, the variable \(x\) is here eliminated using \texttt{class} rules.

Lemma.

In[36]:= Map[\texttt{equal[V, \# \&, SubstTest[\texttt{class, x, implies\{member[x, V], member[x, t]\}, t \rightarrow complement[dif[intersection[FUNS, \texttt{FINITE, image\{inverse[DORA], Q]\}, BIJ]]]}}

Out[36]= subclass[intersection[\texttt{FINITE, FUNS, image\{inverse[DORA], Q\}], BIJ] \Rightarrow \text{True}

In[37]:= \texttt{\% \! /\! \text{. \texttt{Equal} \rightarrow \texttt{SetDelayed}}}

Theorem. A variable-free restatement of a pigeonhole principle.

In[38]:= \texttt{SubstTest[\texttt{and, subclass[u, v], subclass[v, u],
\{u \rightarrow \text{intersection[\texttt{FINITE, FUNS, image\{inverse[DORA], Q]\},
\v \rightarrow \text{intersection[BIJ, \texttt{FINITE}]}}

Out[38]= \texttt{equal[intersection[BIJ, \texttt{FINITE}],
\text{intersection[\texttt{FINITE, FUNS, image\{inverse[DORA], Q\]}] \Rightarrow \text{True}

In[39]:= \texttt{intersection[\texttt{\texttt{FINITE, FUNS, image\{inverse[DORA], Q\]} \Rightarrow \text{intersection[BIJ, \texttt{FINITE}]]}