pigeon-hole principle

Johan G. F. Belinfante
2005 August 8

In[1]: = SetDirectory["i: “; << goedel72.07c; << tools.m
:Package Title: goedel72.07c 2005 August 7 at 11:40 p.m.

It is now: 2005 Aug 8 at 10:22
Loading Simplification Rules

TOOLS.M Revised 2005 August 2

weightlimit = 40

summary

This succinct statement of the pigeonhole principle is available in the GOEDEL program:

In[2]: = subclass[FFINITE, DEDEKIND]


Some corollaries of this statement are derived in this notebook to facilitate applications.

derivation

Recall the definition of DEDEKIND:

In[3]: = fix[composite[Q, PS]]


The pigeonhole principle can therefore be restated in this form:

In[4]: = equal[0, fix[composite[Q, PS, id[FFINITE]]]]


When the fixed point set of a composite is empty, the same is true with the factors inter-changed. This yields:
This can be made a new rewrite rule:

```
In[6]:= intersection[omega, fix[compositeCARD, PS, inverseCARD]] := 0
```

The following corollary holds: if a mapping from a finite set to itself is one-to-one, then it is onto:

```
In[7]:= assert[forall[x, 
    implies[and[ONEONE[x], subclass[range[x], domain[x]], member[x, FINITE]], 
        equal/domain[x], range[x]]]]
```

```
Out[7]= True
```

**additional corollaries**

The **GOEDEL** program now recognizes these facts, which can be made into rewrite rules:

```
In[8]:= equal[fix[compositePS, inverseCARD, id[omega], CARD]], 0
```

```
Out[8]= True
```

```
In[9]:= fix[compositePS, inverseCARD, id[omega], CARD] := 0
```

```
In[10]:= equal[fix[compositeinverseCARD, id[omega], CARD, PS]], 0
```

```
Out[10]= True
```

```
In[11]:= fix[compositeinverseCARD, id[omega], CARD, PS] := 0
```

The presence of **fix** makes the above results awkward to apply. This can be remedied as follows:

```
In[12]:= SubstTest[composite, id[range[z]], z, 
    z -> intersection[PS, compositeinverseCARD, id[omega], CARD]] // Reverse
```

```
Out[12]= intersection[PS, compositeinverseCARD, id[omega], CARD] := 0
```

```
In[13]:= intersection[PS, compositeinverseCARD, id[omega], CARD] := 0
```
Introducing variables

Introducing variables makes it easier to understand the results derived above. For example, it follows that if a finite set is a subset of another set with the same cardinality, then the sets are equal.

Corollary. A finite set has no proper subset with the same cardinality.

lemmas about finite bijections

Since results about functions are true in particular for bijections, one has:
pigeon-hole principle

For a one-to-one mapping the domain and range have the same cardinality, so one obtains this further corollary:

\[ \text{In [26]:=} \quad \text{SubstTest}[\text{implies}, \text{member}[\text{funpart}[y], \text{FINITE}], \text{member}[\text{inverse}[\text{funpart}[y]], \text{FINITE}], y \rightarrow \text{inverse}[\text{oopart}[x]]] \]
\[ \text{Out [26]:=} \quad \text{or}[\text{member}[\text{oopart}[x], \text{FINITE}], \text{not}[\text{member}[\text{inverse}[\text{oopart}[x]], \text{FINITE}]]] = \text{True} \]

\[ \text{In [27]:=} \quad \text{or}[\text{equal}[\text{domain}[\text{oopart}[x]], \text{range}[\text{oopart}[x]]], \text{not}[\text{member}[\text{oopart}[x], \text{FINITE}]], \text{not}[\text{subclass}[\text{range}[\text{oopart}[x]], \text{domain}[\text{oopart}[x]]]]] = \text{True} \]

The \text{oopart} wrapper can be removed:

\[ \text{In [28]:=} \quad \text{SubstTest}[\text{implies}, \text{equal}[x, \text{oopart}[y]], \text{or}[\text{equal}[\text{domain}[x], \text{range}[x]], \text{not}[\text{member}[x, \text{FINITE}]], \text{not}[\text{subclass}[\text{range}[x], \text{domain}[x]]]]] \]
\[ \text{Out [28]:=} \quad \text{or}[\text{equal}[\text{domain}[x], \text{range}[x]], \text{not}[\text{FUNCTION}[x]], \text{not}[\text{FUNCTION}[\text{inverse}[x]]], \text{not}[\text{member}[x, \text{FINITE}]], \text{not}[\text{subclass}[\text{range}[x], \text{domain}[x]]]] = \text{True} \]
This says that if a function from a finite set to itself is one-to-one, then it is onto.

\[ \text{In}[29] := \text{or[equal[domain[x_], range[x_]], not[FUNCTION[x_]],}
\]
\[ \text{not[FUNCTION[inverse[x_]]], not[member[x_, FINITE]],}
\]
\[ \text{notsubclass[range[x_], domain[x_]]]} := \text{True} \]