

the equations  $\text{Id} = \text{IMAGE}[x]$  and  $V = \text{fix}[\text{IMAGE}[x]]$

*Johan G. F. Belinfante*  
2004 October 17

```
In[1]:= SetDirectory["i:"]; << goedel62.16a; << tools.m

:Package Title: goedel62.16a          2004 October 16 at 10:30 p.m.

It is now: 2004 Oct 17 at 8:40

Loading Simplification Rules

TOOLS.M          Revised 2004 September 25

weightlimit = 40
```

---

summary

The equations  $\text{Id} = \text{IMAGE}[x]$  and  $V = \text{fix}[\text{IMAGE}[x]]$  are solved in this notebook.

---

the equation  $\text{Id} = \text{IMAGE}[x]$

Lemma.

```
In[2]:= SubstTest[implies, equal[u, v],
               equal[domain[u], domain[v]], {u → Id, v → IMAGE[x]}]

Out[2]= or[equal[V, domain[VERTSECT[x]]], not[equal[Id, IMAGE[x]]]] == True

In[3]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[4]:= SubstTest[implies, equal[u, v], equal[image[w, u], image[w, v]],
               {u → Id, v → IMAGE[x], w → cross[inverse[SINGLETON], inverse[E]]}]

Out[4]= or[equal[Id, thinpart[x]], not[equal[Id, IMAGE[x]]]] == True

In[5]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= SubstTest[implies, and[equal[u, v], equal[v, w]],
  equal[u, w], {u → Id, v → thinpart[x], w → composite[Id, x]}]
Out[6]= or[equal[Id, composite[Id, x]],
  not[equal[Id, thinpart[x]], not[equal[V, domain[VERTSECT[x]]]]] == True
In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

These three lemmas yield a necessary condition for  $x$  to be a solution of the equation  $\mathbf{Id} = \mathbf{IMAGE}[x]$ .

```
In[8]:= Map[not,
  SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
  not[implies[p1, p4]], {p1 → equal[Id, IMAGE[x]], p2 → thin[x],
  p3 → equal[Id, thinpart[x]], p4 → equal[Id, composite[Id, x]]}]
Out[8]= or[equal[Id, composite[Id, x]], not[equal[Id, IMAGE[x]]] == True
In[9]:= (% /. x → x_) /. Equal → SetDelayed
```

This condition is also sufficient:

```
In[10]:= SubstTest[implies, equal[u, v],
  equal[IMAGE[u], IMAGE[v]], {u → Id, v → composite[Id, x]}]
Out[10]= or[equal[Id, IMAGE[x]], not[equal[Id, composite[Id, x]]] == True
In[11]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining these two implications yields a simple rewrite rule for the solution of the equation  $\mathbf{Id} = \mathbf{IMAGE}[x]$ .

```
In[12]:= equiv[equal[Id, IMAGE[x]], equal[Id, composite[Id, x]]]
Out[12]= True
In[13]:= equal[Id, IMAGE[x_]] := equal[Id, composite[Id, x]]
```

---

the equation  $V = \mathbf{fix}[\mathbf{IMAGE}[x]]$

The equation  $V = \mathbf{fix}[\mathbf{IMAGE}[x]]$  can be reduced to the equation  $\mathbf{Id} = \mathbf{IMAGE}[x]$  as follows:

```
In[14]:= SubstTest[implies, and[equal[v, fix[u]], FUNCTION[u]],
  equal[composite[u, id[v]], id[v]], {u → IMAGE[x], v → V}
Out[14]= or[equal[Id, composite[Id, x]], not[equal[V, fix[IMAGE[x]]]] == True
```

```
In[15]:= (% /. x → x_) /. Equal → SetDelayed
```

The reverse implication also holds.

```
In[16]:= SubstTest[implies, equal[u, v],
  equal[fix[IMAGE[u]], fix[IMAGE[v]]], {u → Id, v → composite[Id, x]}]
Out[16]= or[equal[V, fix[IMAGE[x]]], not[equal[Id, composite[Id, x]]]] == True
In[17]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining these two implications yields a rewrite rule for the solution of the equation  $V = \text{fix}[\text{IMAGE}[x]]$ .

```
In[18]:= equiv[equal[Id, composite[Id, x]], equal[V, fix[IMAGE[x]]]]
Out[18]= True
In[19]:= equal[V, fix[IMAGE[x_]]] := equal[Id, composite[Id, x]]
```

## some falsehoods

Generally speaking it does not seem desirable to add rules to the effect that various statements are false because there would be too many such rules, and many of them would not be very interesting. A few such rules are useful to provide counterexamples, and will be added on a case by case basis as interesting applications for them are discovered. In this notebook two basic falsehoods are needed. Both of these should be made permanent. The first one says that the membership relation is not equal to the identity function:

```
In[20]:= equal[E, Id] // AssertTest
Out[20]= equal[E, Id] == False
In[21]:= equal[E, Id] := False
```

The second one says that the subset relation is not equal to the identity function:

```
In[22]:= equal[Id, S] // AssertTest
Out[22]= equal[Id, S] == False
In[23]:= equal[Id, S] := False
```

To avoid having to add similar rewrite rules for the inverses of **E** and **S**, the a single rule for inverses will be derived. A temporary rule is needed:

```
In[24]:= equal[composite[Id, x], Id] // AssertTest // Reverse
Out[24]= and[equal[V, fix[x]], subclass[composite[Id, x], Id]] ==
        equal[Id, composite[Id, x]]

In[25]:= and[equal[V, fix[x_]], subclass[composite[Id, x_], Id]] :=
        equal[Id, composite[Id, x]]
```

The following will be made permanent:

```
In[26]:= equal[Id, inverse[x]] // AssertTest
Out[26]= equal[Id, inverse[x]] == equal[Id, composite[Id, x]]

In[27]:= equal[Id, inverse[x_]] := equal[Id, composite[Id, x]]
```

---

## two examples

In this section, two examples are considered. The first case is now automatic, and does not need a special rule.

```
In[28]:= equal[V, fix[IMAGE[inverse[S]]]]
Out[28]= False
```

Here is another example. This case does require a special rule:

```
In[29]:= SubstTest[equal, V, fix[IMAGE[x]], x → inverse[E]]
Out[29]= equal[V, fix[BIGCUP]] == False

In[30]:= equal[V, fix[BIGCUP]] := False
```