0 is the only finite acylic unary operation

Johan G. F. Belinfante
2006 February 14

```
In[1]:= SetDirectory["l:"]; << goedel78.14b; << tools.m

        :Package Title: goedel78.14b           2006 February 14 at 3:15 p.m.

        It is now:  2006 Feb 14 at 16:55

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 February 3

        weightlimit = 40
```

summary

The only finite acyclic unary operation is the empty function. A similar result holds for finite acyclic relations

derivation

Lemma.

```
In[2]:= or[ONEONE[inverse[iterate[x, set[y]]]],
          not[equal[0, fix[trv[x]]]], not[FUNCTION[x]]] // NotNotTest

Out[2]= or[and[FUNCTION[inverse[iterate[x, set[y]]]], FUNCTION[iterate[x, set[y]]]],
          not[equal[0, fix[trv[x]]]], not[FUNCTION[x]]] == True

In[3]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Lemma. If the range of a one-to-one function is finite, then so is its domain.

```
In[4]:= Map[implies[FUNCTION[x], #] &, SubstTest[implies,
          and[ONEONE[w], member[domain[w], FINITE]], member[range[w], FINITE], w → inverse[x]]]

Out[4]= or[member[domain[x], FINITE], not[FUNCTION[x]],
          not[FUNCTION[inverse[x]]], not[member[range[x], FINITE]]] == True

In[5]:= or[member[domain[x_], FINITE], not[FUNCTION[x_]],
          not[FUNCTION[inverse[x_]]], not[member[range[x_], FINITE]]] := True
```

Theorem. If **x** has a finite range, then so does **iterate[x, set[y]]**.

```
In[6]:= SubstTest[implies, and[subclass[u, v], member[v, FINITE]],
         member[u, FINITE], {u → range[iterate[x, set[y]]], v → union[range[x], set[y]]}]

Out[6]= or[member[range[iterate[x, set[y]]], FINITE], not[member[range[x], FINITE]]] == True

In[7]:= or[member[range[iterate[x_, set[y_]]], FINITE], not[member[range[x_], FINITE]]] := True
```

Corollary.

```
In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
          {p1 → member[x, FINITE], p2 → member[range[x], FINITE],
           p3 → member[range[iterate[x, set[y]]], FINITE]}]]

Out[8]= or[member[range[iterate[x, set[y]]], FINITE], not[member[x, FINITE]]] == True

In[9]:= or[member[range[iterate[x_, set[y_]]], FINITE], not[member[x_, FINITE]]] := True
```

Lemma.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p5], implies[p1, p6],
           implies[p1, p8], implies[and[p6, p8], p9], implies[p9, not[p5]], p1,
           {p1 → and[FUNCTION[x], member[x, FINITE], subclass[range[x], domain[x]], member[y,
               domain[x]], equal[0, fix[trv[x]]]], p5 → equal[domain[iterate[x, set[y]]], omega],
            p6 → member[range[iterate[x, set[y]]], FINITE], p8 → ONEONE[iterate[x, set[y]]],
            p9 -> member[domain[iterate[x, set[y]]], FINITE]}]]

Out[10]= or[not[equal[0, fix[trv[x]]]], not[FUNCTION[x]], not[member[x, FINITE]],
           not[member[y, domain[x]]], not[subclass[range[x], domain[x]]]] == True

In[11]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Eliminating the variable **y** yields:

```
In[12]:= Map[equal[V, #] &, SubstTest[class, y, or[not[equal[0, u]], not[subclass[P[x], z]],
             not[member[x, v]], not[member[y, w]], not[subclass[t, w]]],
           {u → fix[trv[x]], v → FINITE, w → domain[x], t → range[x], z → FUNS}]] // Reverse

Out[12]= or[equal[0, domain[x]], not[equal[0, fix[trv[x]]]], not[FUNCTION[x]],
           not[member[x, FINITE]], not[subclass[range[x], domain[x]]]] == True

In[13]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. The only function with an empty domain is the empty function.

```
In[14]:= SubstTest[implies, and[equal[x, funpart[y]], equal[0, domain[x]]], equal[0, x], y → x]

Out[14]= or[equal[0, x], not[equal[0, domain[x]]], not[FUNCTION[x]]] == True

In[15]:= or[equal[0, x_], not[equal[0, domain[x_]]], not[FUNCTION[x_]]] := True
```

Theorem. The only finite acyclic unary operation is the empty one.

```
In[16]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 -> and[FUNCTION[x], member[x, FINITE], subclass[range[x], domain[x]],
              equal[0, fix[trv[x]]]], p2 → equal[0, domain[x]], p3 → equal[0, x]}]]

Out[16]= or[equal[0, x], not[equal[0, fix[trv[x]]]], not[FUNCTION[x]],
           not[member[x, FINITE]], not[subclass[range[x], domain[x]]]] == True

In[17]:= or[equal[0, x_], not[equal[0, fix[trv[x_]]]], not[FUNCTION[x_]],
           not[member[x_, FINITE]], not[subclass[range[x_], domain[x_]]]] := True
```

Removing the variable **x** yields:

```
In[18]:= Map[equal[V, #] &, SubstTest[class, x, implies[member[x, y], equal[0, x]],
            y -> intersection[FINITE, ACYCLIC, U[image[MAP, Id]]]]] // Reverse

Out[18]= subclass[intersection[ACYCLIC, FINITE, U[image[MAP, Id]]], set[0]] == True

In[19]:= % /. Equal → SetDelayed
```

This can be restated as an equation:

```
In[20]:= equal[intersection[ACYCLIC, FINITE, U[image[MAP, Id]]], set[0]]

Out[20]= True

In[21]:= intersection[ACYCLIC, FINITE, U[image[MAP, Id]]] := set[0]
```

---

applying the theorem of finite choice

In this section the above results are generalized to relations by using the theorem of finite choice.

```
In[22]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p1, p2], p4],
           implies[and[p1, p2], p5], implies[and[p2, p3, p4, p5], p6],
           implies[and[p2, p6], p7], implies[and[p1, p7], p8], not[implies[and[p1, p2], p8]],
           {p1 → member[x, intersection[FINITE, ACYCLIC, image[inverse[DORA], inverse[S]]]],
            p2 → member[y, X[x]], p3 → member[y, FINITE],
            p4 → member[y, ACYCLIC], p5 → subclass[range[y], domain[y]],
            p6 → equal[0, y], p7 → equal[0, domain[x]], p8 → equal[0, x]}]]

Out[22]= or[equal[0, x], not[equal[0, fix[trv[x]]]],
           not[equal[domain[x], domain[y]]], not[FUNCTION[y]],
           not[member[x, FINITE]], not[member[y, V]], not[subclass[x, cart[V, V]]],
           not[subclass[y, x]], not[subclass[range[x], domain[x]]]] == True

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

Removing the variable **y** yields:

```
In[24]:= Map[equal[V, #] &,
           SubstTest[class, y, implies[and[member[x, u], member[y, v]], equal[0, x]],
            {u -> intersection[FINITE, ACYCLIC, image[inverse[DORA], inverse[S]]],
             v → X[x]}]] // Reverse

Out[24]= or[equal[0, x], equal[0, X[x]], not[equal[0, fix[trv[x]]]], not[member[x, FINITE]],
           not[subclass[x, cart[V, V]]], not[subclass[range[x], domain[x]]]] == True

In[25]:= (% /. x → x_) /. Equal → SetDelayed
```

The theorem of finite choice is applied.

```
In[26]:= Map[not, SubstTest[and, implies[and[p1, p2, p3, p4, p5], p6], implies[p4, p5],
           not[implies[and[p1, p2, p3, p4], p6]], {p1 → equal[0, fix[trv[x]]],
            p2 → subclass[x, cart[V, V]], p3 → subclass[range[x], domain[x]],
            p4 → member[x, FINITE], p5 → not[empty[X[x]]], p6 → equal[0, x]}]]

Out[26]= or[equal[0, x], not[equal[0, fix[trv[x]]]], not[member[x, FINITE]],
           not[subclass[x, cart[V, V]]], not[subclass[range[x], domain[x]]]] == True

In[27]:= or[equal[0, x_], not[equal[0, fix[trv[x_]]]], not[member[x_, FINITE]],
           not[subclass[x_, cart[V, V]]], not[subclass[range[x_], domain[x_]]]] := True
```

The variable **x** is removed:

```
In[28]:= Map[equal[V, #] &, SubstTest[class, x, implies[member[x, y], equal[0, x]],
            y -> intersection[FINITE, ACYCLIC, image[inverse[DORA], inverse[S]]]]] // Reverse

Out[28]= subclass[intersection[ACYCLIC, FINITE, image[inverse[DORA], inverse[S]]], set[0]] ==
          True

In[29]:= % /. Equal → SetDelayed
```

Restatement as an equation.

```
In[30]:= equal[intersection[ACYCLIC, FINITE, image[inverse[DORA], inverse[S]]], set[0]]

Out[30]= True

In[31]:= intersection[ACYCLIC, FINITE, image[inverse[DORA], inverse[S]]] := set[0]
```