positive integers as well-founded relations

Johan G. F. Belinfante
2006 January 29

```
In[1]:= SetDirectory["l:"]; << goedel77.28a; << tools.m

        :Package Title: goedel77.28a             2006 January 28 at 5:40 p.m.

        It is now:  2006 Jan 29 at 19:4

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 January 2

        weightlimit = 40
```

summary

A model of the set **Z** of integers can be constructed from the set **omega** of natural numbers as the set of equivalence classes of a certain equivalence relation **EQUIDIFF** on pairs of natural numbers. In this notebook it is shown that in this model, all positive integers are well founded relations, and the rest are not.

integers as relations

The sum of two natural numbers **x** and **y** is denoted **natadd[x,y]**. The equivalence relation **EQUIDIFF** is defined by the membership relation

```
In[2]:= member[pair[pair[u, v], pair[x, y]], EQUIDIFF]

Out[2]= and[equal[natadd[u, y], natadd[v, x]], member[u, omega],
          member[v, omega], member[x, omega], member[y, omega]]

In[3]:= EQUIVALENCE[EQUIDIFF]

Out[3]= True
```

The construction of the relation **EQUIDIFF** from the function **NATADD** for natural number addition can also be described in variable-free fashion as follows:

```
In[4]:= flip[twist[composite[inverse[NATADD], NATADD]]]

Out[4]= EQUIDIFF
```

The equivalence classes of an equivalence relation are its non-empty vertical sections. In the case of **EQUIDIFF**, this yields the set of integers:

```
In[5]:= dif[range[VERTSECT[EQUIDIFF]], set[0]]
Out[5]= Z
```

Each integer can be viewed as a relation on the set of natural numbers. Since these are readily available examples of relations, it may be of interest to investigate their properties. One of these properties is that each integer is a bijection:

```
In[6]:= subclass[Z, BIJ]
Out[6]= True
```

positive and negative integers

For each natural number **x** there is a corresponding non-negative integer **plus[x]** and a non-positive integer **inverse[plus[x]]**. For the natural number **0**, these are the same. The integer zero is the identity function on the natural numbers:

```
In[7]:= plus[0]
Out[7]= id[omega]
```

The natural number one is **set[0]**, and the corresponding positive integer is the restriction of the successsor function **SUCC** to the set **omega** of natural numbers.

```
In[8]:= plus[set[0]]
Out[8]= composite[id[omega], SUCC]
```

well-founded relations

A relation is well-founded if it does not permit infinite regress. This can be expressed succinctly in terms of subvariance. A class **y** is **subvariant** under **x** if

```
In[9]:= subvariant[x, y]
Out[9]= subclass[y, image[x, y]]
```

The class of all sets **y** subvariant under **x** is denoted **subvar[x]**.

```
In[10]:= class[y, subvariant[x, y]]
Out[10]= subvar[x]
```

A relation **x** is **wellfounded** if the only subvariant set is the empty set.

```
In[11]:= and[subclass[x, cart[V, V]], equal[subvar[x], set[0]]]
Out[11]= WELLFOUNDED[x]
```

For inverse functions, one can describe subvariance in terms of invariance.

```
In[12]:= SubstTest[implies, equal[x, funpart[y]],
          equal[subvar[inverse[x]], intersection[invar[x], P[domain[x]]]], y → x]

Out[12]= or[equal[intersection[invar[x], P[domain[x]]], subvar[inverse[x]]],
           not[FUNCTION[x]]] == True

In[13]:= or[equal[intersection[invar[x_], P[domain[x_]]], subvar[inverse[x_]]],
           not[FUNCTION[x_]]] := True
```

well-foundedness of positive integers

The transitive closure of **plus[set[0]]** is the restriction of the membership relation **E** to the set of natural numbers, which is known to be well-founded.

```
In[14]:= trv[plus[set[0]]]

Out[14]= composite[id[omega], E]

In[15]:= WELLFOUNDED[composite[id[omega], E]]

Out[15]= True
```

Lemma. The transitive closure of a relation is the union of all its positive powers. This yields another expression for this transitive closure, involving the function **NATADD**.

```
In[16]:= SubstTest[image, power[x], complement[set[0]], x → plus[set[0]]]

Out[16]= composite[NATADD, id[cart[V, complement[set[0]]], inverse[FIRST]] ==
          composite[id[omega], E]

In[17]:= composite[NATADD, id[cart[V, complement[set[0]]], inverse[FIRST]] :=
          composite[id[omega], E]
```

Each positive integer **plus[x]** is the composite of **x** copies of **plus[set[0]]**. That is, each positive integer is a positive power of the function **plus[set[0]]**, and therefore is a subset of its transitive closure:

```
In[18]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → set[x], v → complement[set[0]], w → power[plus[set[0]]]}]

Out[18]= or[equal[0, x], subclass[plus[x], E]] == True

In[19]:= (% /. x → x_) /. Equal → SetDelayed
```

A better rewrite rule can be derived. To do so, a temporary lemma is needed:

```
In[20]:= subclass[omega, fix[E]] // AssertTest

Out[20]= subclass[omega, fix[E]] == False
```

```
In[21]:= subclass[omega, fix[E]] := False
```

The **GOEDE**L program now recognizes that the implication derived above can be sharpened to a logical equivalence, which is made into a rewrite rule.

```
In[22]:= equiv[subclass[plus[x], E], not[equal[0, x]]]

Out[22]= True

In[23]:= subclass[plus[x_], E] := not[equal[0, x]]
```

Since any subclass of a well-founded relation is well-founded, it follows that

```
In[24]:= SubstTest[implies, and[subclass[u, v], member[v, WF]], member[u, WF],
           {u → plus[x], v → composite[id[omega], E]}]

Out[24]= or[equal[0, x], WELLFOUNDED[plus[x]]] == True

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

This implication can also be sharpened to a logical equivalence.

```
In[26]:= equiv[WELLFOUNDED[plus[x]], not[equal[0, x]]]

Out[26]= True

In[27]:= WELLFOUNDED[plus[x_]] := not[equal[0, x]]
```

negative integers

If one turns **omega** upside down, it becomes an infinitely descending sequence. One can descend this staircase forever either one step at a time or **x** steps at a time.

```
In[28]:= Map[not, SubstTest[implies, and[member[u, v], subclass[v, w]], member[u, w],
             {u → omega, v -> intersection[invar[plus[nat[x]]], P[omega]], w → set[0]}]]

Out[28]= subclass[intersection[invar[plus[nat[x]]], P[omega]], set[0]] == False

In[29]:= (% /. x → x_) /. Equal → SetDelayed
```

It follows immediately that negative integers are not well-founded.

```
In[30]:= SubstTest[equal, subvar[y], set[0], y → inverse[plus[nat[x]]]] // Reverse

Out[30]= WELLFOUNDED[inverse[plus[nat[x]]]] == False

In[31]:= (% /. x → x_) /. Equal → SetDelayed
```

Removing the **nat** wrapper yields:

```
In[32]:= SubstTest[implies, equal[x, nat[y]], not[WELLFOUNDED[inverse[plus[x]]]], y → x]

Out[32]= or[not[member[x, omega]], not[WELLFOUNDED[inverse[plus[x]]]]] == True

In[33]:= (% /. x → x_) /. Equal → SetDelayed
```

In the reverse direction, one has:

```
In[34]:= SubstTest[implies, equal[0, z], WELLFOUNDED[z], z → inverse[plus[x]]]

Out[34]= or[member[x, omega], WELLFOUNDED[inverse[plus[x]]]] == True

In[35]:= (% /. x → x_) /. Equal → SetDelayed
```

These facts can be combined into a logical equivalence, and made into a permanent rewrite rule.

```
In[36]:= equiv[WELLFOUNDED[inverse[plus[x]]], not[member[x, omega]]]

Out[36]= True

In[37]:= WELLFOUNDED[inverse[plus[x_]]] := not[member[x, omega]]
```

Comment. When **x** is not a natural number, **plus[x]** is empty, and so is its inverse. The empty set is a wellfounded relation.