total order for integers

Johan G. F. Belinfante
2006 September 24

```
In[1]:= SetDirectory["l:"]; << goedel85.23b; << tools.m

        :Package Title: goedel85.23b          2006 September 23 at 3:00 p.m.

        It is now:  2006 Sep 24 at 0:10

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 September 23

        weightlimit = 40
```

summary

In this notebook basic properties of the integer arithmetic less-than-or-equal relation relation **INTLEQ** are derived.

definition

The relation **INTLEQ** is defined by the following **class**-wrapped membership rule. The quantities **range[PLUS]** and **INTADD** in this definition are wrapped with **HoldPattern** to expedite normalization tests. (The **HoldPattern** does not show up in the output of **FirstMatch**.)

```
In[2]:= Begin["Goedel`Private`"];

In[3]:= FirstMatch[class[x_, member[y_, HoldPattern[INTLEQ]]]]

Out[3]= class[x_, member[y_, INTLEQ]] := Module[{u = Unique[], v = Unique[], w =
          Unique[]}, ReleaseHold[class[x, exists[u, v, w, and[equal[y, pair[u,
          v]], member[w, range[PLUS]]], member[pair[pair[u, w], v], INTADD]]]]]]
```

normalization

An explicit formula for **INTLEQ** is obtained using **Normality**.

```
In[4]:= INTLEQ // Normality // Reverse

Out[4]= composite[INTADD, id[cart[V, range[PLUS]]], inverse[FIRST]] == INTLEQ

In[5]:= composite[INTADD, id[cart[V, range[PLUS]]], inverse[FIRST]] := INTLEQ
```

This formula provides an easy way to derive the properties of **INTLEQ**.

relation

Theorem. The class **INTLEQ** is a relation.

```
In[6]:= SubstTest[subclass,
          composite[x, id[cart[V, range[PLUS]]], inverse[FIRST]], cart[V, V], x → INTADD]
Out[6]= subclass[INTLEQ, cart[V, V]] == True

In[7]:= subclass[INTLEQ, cart[V, V]] := True
```

Corollary.

```
In[8]:= Assoc[Id, INTADD, composite[id[cart[V, range[PLUS]]], inverse[FIRST]]]
Out[8]= composite[Id, INTLEQ] == INTLEQ

In[9]:= composite[Id, INTLEQ] := INTLEQ
```

domain

The domain of **INTLEQ** is the set **Z** of all integers.

```
In[10]:= SubstTest[domain, composite[x, id[cart[V, range[PLUS]]], inverse[FIRST]], x → INTADD]
Out[10]= domain[INTLEQ] == Z

In[11]:= domain[INTLEQ] := Z
```

Corollary.

```
In[12]:= Assoc[INTLEQ, id[Z], id[P[cart[V, V]]]] // Reverse
Out[12]= composite[INTLEQ, id[P[cart[V, V]]]] == INTLEQ

In[13]:= composite[INTLEQ, id[P[cart[V, V]]]] := INTLEQ
```

range

Lemma.

```
In[14]:= ImageComp[INTADD, inverse[SECOND], range[PLUS]] // Reverse
Out[14]= image[INTADD, cart[V, range[PLUS]]] == Z
```

```
In[15]:= image[INTADD, cart[V, range[PLUS]]] := Z
```

Theorem.

```
In[16]:= SubstTest[range, composite[x, id[cart[V, range[PLUS]]], inverse[FIRST]], x → INTADD]
Out[16]= range[INTLEQ] == Z
```

```
In[17]:= range[INTLEQ] := Z
```

Corollary.

```
In[18]:= Assoc[id[P[cart[V, V]]], id[Z], INTLEQ]
Out[18]= composite[id[P[cart[V, V]]], INTLEQ] == INTLEQ
```

```
In[19]:= composite[id[P[cart[V, V]]], INTLEQ] := INTLEQ
```

Corollary. Since its domain and range are sets, the relation **INTLEQ** is a set.

```
In[20]:= member[INTLEQ, V] // AssertTest
Out[20]= member[INTLEQ, V] == True
```

```
In[21]:= member[INTLEQ, V] := True
```

composite with INVERSE

Lemma. A simplification rule.

```
In[22]:= Assoc[INTADD, id[cart[Z, V]], id[cart[P[cart[V, V]], V]]] // Reverse
Out[22]= composite[INTADD, id[cart[P[cart[V, V]], V]]] == INTADD
```

```
In[23]:= composite[INTADD, id[cart[P[cart[V, V]], V]]] := INTADD
```

Lemma. A simplification rule.

```
In[24]:= IminComp[INTADD, id[cart[P[cart[V, V]], V]], x] // Reverse
Out[24]= composite[image[inverse[INTADD], x], id[P[cart[V, V]]]] == image[inverse[INTADD], x]
```

```
In[25]:= composite[image[inverse[INTADD], x_], id[P[cart[V, V]]]] := image[inverse[INTADD], x]
```

Theorem.

```
In[26]:= Map[composite[#, INVERSE] &,
          SubstTest[image, composite[SWAP, inverse[rotate[z]]], range[PLUS], z → INTADD]]
Out[26]= image[inverse[INTADD], range[PLUS]] == composite[INTLEQ, INVERSE]
```

```
In[27]:= image[inverse[INTADD], range[PLUS]] := composite[INTLEQ, INVERSE]
```

Corollary.

```
In[28]:= ImincComp[INVERSE, INTADD, range[PLUS]] // Reverse

Out[28]= image[inverse[INTADD], image[INVERSE, range[PLUS]]] == composite[INVERSE, INTLEQ]

In[29]:= image[inverse[INTADD], image[INVERSE, range[PLUS]]] := composite[INVERSE, INTLEQ]
```

fix[INTLEQ]

Lemma.

```
In[30]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → set[id[omega]], v → range[PLUS],
           w → composite[cross[INVERSE, Id], inverse[INTADD]]}]

Out[30]= subclass[Z, fix[INTLEQ]] == True

In[31]:= % /. Equal → SetDelayed
```

The reverse inclusion also holds:

```
In[32]:= SubstTest[subclass, fix[x], domain[x], x → INTLEQ]

Out[32]= subclass[fix[INTLEQ], Z] == True

In[33]:= % /. Equal → SetDelayed
```

These inclusions can be combined into an equation:

```
In[34]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → Z, v -> fix[INTLEQ]}]

Out[34]= True == equal[Z, fix[INTLEQ]]

In[35]:= fix[INTLEQ] := Z
```

Corollary.

```
In[36]:= SubstTest[subclass, x, cart[fix[x], fix[x]], x → INTLEQ] // Reverse

Out[36]= REFLEXIVE[INTLEQ] == True

In[37]:= REFLEXIVE[INTLEQ] := True
```

transitive property

Lemma.

```
In[38]:= ImageComp[HULL[Z], id[Z], range[PLUS]] // Reverse
Out[38]= image[HULL[Z], range[PLUS]] == range[PLUS]

In[39]:= image[HULL[Z], range[PLUS]] := range[PLUS]
```

Theorem.

```
In[40]:= ImageComp[HULL[Z], composite[COMPOSE, id[cart[Z, Z]]], cart[range[PLUS], range[PLUS]]]
Out[40]= image[INTADD, cart[range[PLUS], range[PLUS]]] == range[PLUS]

In[41]:= image[INTADD, cart[range[PLUS], range[PLUS]]] := range[PLUS]

In[42]:= Assoc[composite[INTADD, cross[Id, INTADD]],
           ASSOC, composite[id[cart[cart[V, range[PLUS]], range[PLUS]]],
            inverse[FIRST], inverse[FIRST]]] // Reverse
Out[42]= composite[INTLEQ, INTLEQ] == INTLEQ

In[43]:= composite[INTLEQ, INTLEQ] := INTLEQ

In[44]:= SubstTest[subclass, composite[x, x], x, x → INTLEQ] // Reverse
Out[44]= TRANSITIVE[INTLEQ] == True

In[45]:= TRANSITIVE[INTLEQ] := True
```

inverse

```
In[46]:= SubstTest[inverse, composite[image[inverse[INTADD], x], INVERSE], x → range[PLUS]]
Out[46]= inverse[INTLEQ] == composite[INVERSE, INTLEQ, INVERSE]

In[47]:= inverse[INTLEQ] := composite[INVERSE, INTLEQ, INVERSE]
```

antisymmetry

Theorem. The relation **INTLEQ** is antisymmetric.

```
In[48]:= Map[composite[#, INVERSE] &, SubstTest[intersection, image[inverse[INTADD], x],
           image[inverse[INTADD], y], {x → range[PLUS], y → image[INVERSE, range[PLUS]]}]]
Out[48]= intersection[INTLEQ, composite[INVERSE, INTLEQ, INVERSE]] == id[Z]

In[49]:= intersection[INTLEQ, composite[INVERSE, INTLEQ, INVERSE]] := id[Z]
```

Corollary.

```
In[50]:= SubstTest[and, REFLEXIVE[x], ANTISYMMETRIC[x], TRANSITIVE[x], x → INTLEQ] // Reverse

Out[50]= PARTIALORDER[INTLEQ] == True

In[51]:= PARTIALORDER[INTLEQ] := True
```

Theorem.

```
In[52]:= Map[composite[#, INVERSE] &, SubstTest[image, inverse[INTADD],
           union[x, y], {x → range[PLUS], y → image[INVERSE, range[PLUS]]}]] // Reverse

Out[52]= union[INTLEQ, composite[INVERSE, INTLEQ, INVERSE]] == cart[Z, Z]

In[53]:= union[INTLEQ, composite[INVERSE, INTLEQ, INVERSE]] := cart[Z, Z]
```

Corollary.

```
In[54]:= TOTALORDER[INTLEQ] // AssertTest

Out[54]= TOTALORDER[INTLEQ] == True

In[55]:= TOTALORDER[INTLEQ] := True
```

Corollary.

```
In[56]:= Map[composite[#, INVERSE] &, SubstTest[composite, to[x], inverse[to[x]], x → INTLEQ]]

Out[56]= composite[INTLEQ, INVERSE, INTLEQ] == cart[Z, Z]

In[57]:= composite[INTLEQ, INVERSE, INTLEQ] := cart[Z, Z]
```