INTTIMES, part 1

Johan G. F. Belinfante
2006 December 19

```
In[1]:= SetDirectory["l:"]; << goedel88.18b; << tools.m

        :Package Title: goedel88.18b        2006 December 18 at 1:25 p.m.

        It is now:  2006 Dec 19 at 18:53

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 December 17

        weightlimit = 40
```

summary

Each binary endomorphism of **INTADD** corresponds to multiplication of integers by a fixed integer. A **class**-wrapped membership rule for a correspondence **INTTIMES** between integers and members of **binhom[INTADD, INTADD]** has been introduced:

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

In[3]:= FirstMatch[class[t_, member[w_, HoldPattern[INTTIMES]]]]

Out[3]= class[t_, member[w_, INTTIMES]] := ReleaseHold[Module[{u = Unique[], v = Unique[]},
          class[t, exists[u, v, and[equal[w, pair[u, v]], equal[image[v, set[composite[
          id[omega], SUCC]]], set[u]], member[v, binhom[INTADD, INTADD]]]]]]]]
```

The correspondence **INTTIMES** relates endomorphisms of **INTADD** to their values at plus one. In this notebook, only some of the more elementary properties of this correspondence are derived, based primarily on analogy with the previously developed theory of **MIXTIMES**. However, this analogy does not go through for the proof that **INTTIMES** is a function, which will require a fresh approach.

normalization

Lemma.

```
In[4]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → member[x, binhom[INTADD, INTADD]],
            p2 -> equal[domain[x], Z], p3 → member[plus[set[0]], domain[x]]}]] // Reverse

Out[4]= or[member[composite[id[omega], SUCC], domain[x]],
          not[member[x, binhom[INTADD, INTADD]]]] == True
```

```
In[5]:= or[member[composite[id[omega], SUCC], domain[x_]],
           not[member[x_, binhom[INTADD, INTADD]]]] := True
```

Two variable-free rewrite rules are needed to normalize **INTTIMES**. This is the first:

```
In[6]:= Map[equal[V, #] &, SubstTest[class, x, or[member[w, domain[x]], not[member[x, y]]],
           {w → plus[set[0]], y -> binhom[INTADD, INTADD]}]]

Out[6]= equal[0, intersection[binhom[INTADD, INTADD],
            P[complement[cart[set[composite[id[omega], SUCC]], V]]]]] == True

In[7]:= intersection[binhom[INTADD, INTADD],
           P[complement[cart[set[composite[id[omega], SUCC]], V]]]] := 0
```

Corollary. (A second rewrite rule needed to normalize **INTTIMES**.)

```
In[8]:= equal[intersection[binhom[INTADD, INTADD],
           complement[P[complement[cart[set[plus[set[0]]], V]]]]], binhom[INTADD, INTADD]]

Out[8]= True

In[9]:= intersection[binhom[INTADD, INTADD], complement[
            P[complement[cart[set[composite[id[omega], SUCC]], V]]]]] := binhom[INTADD, INTADD]
```

Theorem. This equation for **INTTIMES** suffices for normalization.

```
In[10]:= INTTIMES // Normality // Reverse

Out[10]= composite[id[binhom[INTADD, INTADD]],
            inverse[eval[composite[id[omega], SUCC]]]] == INTTIMES

In[11]:= composite[id[binhom[INTADD, INTADD]],
            inverse[eval[composite[id[omega], SUCC]]]] := INTTIMES
```

---

some properties of INTTIMES

Corollary.

```
In[12]:= Assoc[Id, id[binhom[INTADD, INTADD]], inverse[eval[plus[set[0]]]]]

Out[12]= composite[Id, INTTIMES] == INTTIMES

In[13]:= composite[Id, INTTIMES] := INTTIMES
```

Corollary.

```
In[14]:= SubstTest[subclass, composite[Id, x], cart[V, V], x → INTTIMES] // Reverse

Out[14]= subclass[INTTIMES, cart[V, V]] == True

In[15]:= subclass[INTTIMES, cart[V, V]] := True
```

Corollary.

```
In[16]:= composite[eval[plus[set[0]]], id[binhom[INTADD, INTADD]]] // DoubleInverse

Out[16]= composite[eval[composite[id[omega], SUCC]], id[binhom[INTADD, INTADD]]] ==
          inverse[INTTIMES]

In[17]:= composite[eval[composite[id[omega], SUCC]], id[binhom[INTADD, INTADD]]] :=
          inverse[INTTIMES]
```

Corollary. (One-to-one property.)

```
In[18]:= SubstTest[FUNCTION, composite[eval[plus[set[0]]], id[x]],
           x → binhom[INTADD, INTADD]] // Reverse

Out[18]= FUNCTION[inverse[INTTIMES]] == True

In[19]:= FUNCTION[inverse[INTTIMES]] := True
```

Corollary. (This will be needed later in connection with uncurrying.)

```
In[20]:= Assoc[FUNPART, id[binhom[INTADD, INTADD]], inverse[eval[composite[id[omega], SUCC]]]]

Out[20]= composite[FUNPART, INTTIMES] == INTTIMES

In[21]:= composite[FUNPART, INTTIMES] := INTTIMES
```

Theorem. The domain of **INTTIMES** is the set of integers.

```
In[22]:= SubstTest[range, composite[eval[plus[set[0]]], id[x]],
           x → binhom[INTADD, INTADD]] // Reverse

Out[22]= domain[INTTIMES] == Z

In[23]:= domain[INTTIMES] := Z
```

Theorem. **IMAGE[FIRST]** rule

```
In[24]:= Assoc[IMAGE[FIRST], id[binhom[INTADD, INTADD]], inverse[eval[plus[set[0]]]]]

Out[24]= composite[IMAGE[FIRST], INTTIMES] == cart[Z, set[Z]]

In[25]:= composite[IMAGE[FIRST], INTTIMES] := cart[Z, set[Z]]
```

---

range

Lemma.

```
In[26]:= SubstTest[implies, member[w, x],
           subclass[map[x, y], domain[eval[w]]], {w → plus[set[0]], x → Z, y → Z}] // Reverse

Out[26]= subclass[map[Z, Z], domain[eval[composite[id[omega], SUCC]]]] == True
```

```
In[27]:= subclass[map[Z, Z], domain[eval[composite[id[omega], SUCC]]]] := True
```

Lemma.

```
In[28]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → binhom[INTADD, INTADD], v → map[Z, Z], w → domain[eval[plus[set[0]]]]}] // Reverse

Out[28]= subclass[binhom[INTADD, INTADD], domain[eval[composite[id[omega], SUCC]]]] == True

In[29]:= subclass[binhom[INTADD, INTADD], domain[eval[composite[id[omega], SUCC]]]] := True
```

Lemma.

```
In[30]:= equal[intersection[binhom[INTADD, INTADD], domain[eval[composite[id[omega], SUCC]]]],
          binhom[INTADD, INTADD]]

Out[30]= True

In[31]:= intersection[binhom[INTADD, INTADD], domain[eval[composite[id[omega], SUCC]]]] :=
          binhom[INTADD, INTADD]
```

Theorem. The range of **INTTIMES** is the set of all endomorphisms of integer addition.

```
In[32]:= SubstTest[domain, composite[eval[plus[set[0]]], id[x]],
           x → binhom[INTADD, INTADD]] // Reverse

Out[32]= range[INTTIMES] == binhom[INTADD, INTADD]

In[33]:= range[INTTIMES] := binhom[INTADD, INTADD]
```

corollaries

Corollary.

```
In[42]:= SubstTest[subclass, composite[Id, w], cart[x, y], w → INTTIMES] // Reverse

Out[42]= subclass[INTTIMES, cart[x, y]] ==
          and[subclass[Z, x], subclass[binhom[INTADD, INTADD], y]]

In[43]:= subclass[INTTIMES, cart[x_, y_]] :=
          and[subclass[Z, x], subclass[binhom[INTADD, INTADD], y]]
```

Corollary.

```
In[36]:= member[INTTIMES, V] // AssertTest

Out[36]= member[INTTIMES, V] == True

In[37]:= member[INTTIMES, V] := True
```

membership rule

Lemma. (Temporary rewrite rule.)

```
In[38]:= SubstTest[member, pair[t, x], composite[Id, z], {z → INTTIMES, w → V}]

Out[38]= and[member[t, V], member[x, V], member[pair[t, x], INTTIMES]] ==
          member[pair[t, x], INTTIMES]

In[39]:= and[member[t_, V], member[x_, V], member[pair[t_, x_], INTTIMES]] :=
          member[pair[t, x], INTTIMES]
```

Theorem. (A secondary membership rule.)

```
In[40]:= SubstTest[member, pair[t, x],
           composite[eval[plus[set[0]]], id[z]], z -> binhom[INTADD, INTADD]] // Reverse

Out[40]= member[pair[x, t], INTTIMES] ==
          and[equal[image[t, set[composite[id[omega], SUCC]]], set[x]],
           member[t, binhom[INTADD, INTADD]], member[x, V]]

In[41]:= member[pair[x_, t_], INTTIMES] :=
          and[equal[image[t, set[composite[id[omega], SUCC]]], set[x]],
           member[t, binhom[INTADD, INTADD]], member[x, V]]
```