intplus[x]

Johan G. F. Belinfante
2012 November 25

```
In[1]:= SetDirectory["l:"]; << goedel.12nov24a
```

```
:Package Title: goedel.12nov24a            2012 November 24 at 12:30 p.m.

Loading takes about sixteen minutes, half that time due to builtin pauses.

It is now:  2012 Nov 25 at 11:19

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2012 Nov 25 at 11:34
```

summary

A new constructor **intplus[x]** is introduced by analogy with **plus[x]** and **ratplus[x]**.

removing four old rules

One first needs to remove several existing rules:

```
In[2]:= composite[INTADD, LEFT[x_]] =.
```

```
In[3]:= composite[INTADD, RIGHT[x_]] =.
```

```
In[4]:= composite[inverse[LEFT[x_]], inverse[INTADD]] =.
```

```
In[5]:= composite[inverse[RIGHT[x_]], inverse[INTADD]] =.
```

definition and replacement rules

The constructor **intplus[x]** is defined as follows.

```
In[6]:= composite[INTADD, LEFT[x_]] := intplus[x]
```

This definition replaces one of the four removed rules. The other three are replaced with the following.

Theorem. An analogous rule.

```
In[7]:= Assoc[INTADD, SWAP, LEFT[x]]

Out[7]= composite[INTADD, RIGHT[x]] == intplus[x]

In[8]:= composite[INTADD, RIGHT[x_]] := intplus[x]
```

Theorem.

```
In[9]:= composite[inverse[LEFT[x]], inverse[INTADD]] // DoubleInverse

Out[9]= composite[inverse[LEFT[x]], inverse[INTADD]] == inverse[intplus[x]]

In[10]:= composite[inverse[LEFT[x_]], inverse[INTADD]] := inverse[intplus[x]]
```

Theorem.

```
In[11]:= composite[inverse[RIGHT[x]], inverse[INTADD]] // DoubleInverse

Out[11]= composite[inverse[RIGHT[x]], inverse[INTADD]] == inverse[intplus[x]]

In[12]:= composite[inverse[RIGHT[x_]], inverse[INTADD]] := inverse[intplus[x]]
```

---

basic properties

Some basic properties of **intplus[x]** are derived here.

Theorem. The class **intplus[x]** is a set.

```
In[13]:= SubstTest[member, composite[binop[t], LEFT[x]], V, t → INTADD] // Reverse

Out[13]= member[intplus[x], V] == True

In[14]:= member[intplus[x_], V] := True
```

Theorem. The set **intplus[x]** is a function.

```
In[15]:= SubstTest[FUNCTION, composite[binop[t], LEFT[x]], t → INTADD] // Reverse

Out[15]= FUNCTION[intplus[x]] == True

In[16]:= FUNCTION[intplus[x_]] := True
```

Theorem. The function **intplus[x]** is one-to-one.

```
In[17]:= Map[FUNCTION,
           SubstTest[composite, inverse[LEFT[x]], inverse[gp[t]], t → INTADD]] // Reverse

Out[17]= FUNCTION[inverse[intplus[x]]] == True

In[18]:= FUNCTION[inverse[intplus[x_]]] := True
```

domain and range

Theorem. Range rule.

```
In[20]:= ImageComp[INTADD, LEFT[x], V]
Out[20]= range[intplus[x]] == intersection[Z, image[V, intersection[Z, set[x]]]]
```

```
In[21]:= range[intplus[x_]] := intersection[Z, image[V, intersection[Z, set[x]]]]
```

Corollary. Simplification rule.

```
In[22]:= Assoc[id[Z], id[range[intplus[x]]], intplus[x]]
Out[22]= composite[id[Z], intplus[x]] == intplus[x]
```

```
In[23]:= composite[id[Z], intplus[x_]] := intplus[x]
```

Corollary. Simplification rule.

```
In[24]:= Assoc[id[P[cart[V, V]]], id[Z], intplus[x]]
Out[24]= composite[id[P[cart[V, V]]], intplus[x]] == intplus[x]
```

```
In[25]:= composite[id[P[cart[V, V]]], intplus[x_]] := intplus[x]
```

Theorem. Domain rule.

```
In[26]:= IminComp[INTADD, LEFT[x], V]
Out[26]= domain[intplus[x]] == intersection[Z, image[V, intersection[Z, set[x]]]]
```

```
In[27]:= domain[intplus[x_]] := intersection[Z, image[V, intersection[Z, set[x]]]]
```

Corollary. Simplification rule.

```
In[28]:= Assoc[intplus[x], id[domain[intplus[x]]], id[Z]] // Reverse
Out[28]= composite[intplus[x], id[Z]] == intplus[x]
```

```
In[29]:= composite[intplus[x_], id[Z]] := intplus[x]
```

Corollary. Simplification rule.

```
In[30]:= Assoc[intplus[x], id[Z], id[P[cart[V, V]]]] // Reverse
Out[30]= composite[intplus[x], id[P[cart[V, V]]]] == intplus[x]
```

```
In[31]:= composite[intplus[x_], id[P[cart[V, V]]]] := intplus[x]
```

Corollary. The function **intplus[x]** is empty when **x** is not an integer.

```
In[32]:= SubstTest[empty, domain[funpart[t]], t → intplus[x]]

Out[32]= equal[0, intplus[x]] == not[member[x, Z]]

In[33]:= equal[0, intplus[x_]] := not[member[x, Z]]
```

Theorem.

```
In[34]:= SubstTest[member, intplus[x], intersection[u, v],
           {u -> FUNS, v -> image[inverse[DORA], inverse[S]]}] // Reverse

Out[34]= member[intplus[x], UNOPS] == True

In[35]:= member[intplus[x_], UNOPS] := True
```

Theorem.

```
In[37]:= SubstTest[and, member[t, FUNS],
           equal[domain[t], Z], subclass[range[t], Z], t → intplus[x]]

Out[37]= member[intplus[x], map[Z, Z]] == member[x, Z]

In[38]:= member[intplus[x_], map[Z, Z]] := member[x, Z]
```

Theorem.

```
In[40]:= SubstTest[and, member[t, BIJ], equal[domain[t], Z], equal[range[t], Z], t → intplus[x]]

Out[40]= member[intplus[x], bij[Z, Z]] == member[x, Z]

In[41]:= member[intplus[x_], bij[Z, Z]] := member[x, Z]
```

function application and related rules

Theorem. Function application rule.

```
In[42]:= SubstTest[APPLY, composite[binop[t], LEFT[x]], y, t → INTADD] // Reverse

Out[42]= APPLY[intplus[x], y] == intadd[x, y]

In[43]:= APPLY[intplus[x_], y_] := intadd[x, y]
```

Corollary. Vertical section rule.

```
In[44]:= SubstTest[image, funpart[t], set[y], t → intplus[x]] // Reverse

Out[44]= image[intplus[x], set[y]] == set[intadd[x, y]]

In[45]:= image[intplus[x_], set[y_]] := set[intadd[x, y]]
```

Theorem.

```
In[46]:= Assoc[composite[INTADD, cross[Id, INTADD]], ASSOC, RIGHT[x]] // Reverse
Out[46]= composite[intplus[x], INTADD] == composite[INTADD, cross[Id, intplus[x]]]

In[47]:= composite[intplus[x_], INTADD] := composite[INTADD, cross[Id, intplus[x]]]
```

Theorem. Rule for composites.

```
In[48]:= Assoc[intplus[x], INTADD, RIGHT[y]]
Out[48]= composite[intplus[x], intplus[y]] == intplus[intadd[x, y]]

In[49]:= composite[intplus[x_], intplus[y_]] := intplus[intadd[x, y]]
```

Theorem. A special case.

```
In[50]:= intplus[id[omega]] // FastReifNormality
Out[50]= intplus[id[omega]] == id[Z]

In[51]:= intplus[id[omega]] := id[Z]
```

Theorem. Membership rule for pairs.

```
In[52]:= SubstTest[member, y, image[v, set[x]], v → intplus[z]]
Out[52]= member[pair[x, y], intplus[z]] == and[equal[y, intadd[x, z]], member[x, Z], member[z, Z]]

In[53]:= member[pair[x_, y_], intplus[z_]] :=
          and[equal[y, intadd[x, z]], member[x, Z], member[z, Z]]
```

---

normalization rules

Theorem. (Normalization rule.)

```
In[54]:= intplus[x] // FastReifNormality// Reverse
Out[54]= composite[id[image[V, intersection[Z, set[x]]]], intplus[x]] == intplus[x]

In[55]:= composite[id[image[V, intersection[Z, set[x_]]]], intplus[x_]] := intplus[x]
```

Theorem. (Normalization rule.)

```
In[56]:= composite[intplus[x], id[image[V, intersection[Z, set[x]]]]] // FastReifNormality
Out[56]= composite[intplus[x], id[image[V, intersection[Z, set[x]]]]] == intplus[x]

In[57]:= composite[intplus[x_], id[image[V, intersection[Z, set[x_]]]]] := intplus[x]
```

Theorem. (Normalization rule.)

```
In[58]:= composite[image[inverse[INTADD], set[x]], INVERSE] // FastReifNormality
Out[58]= composite[image[inverse[INTADD], set[x]], INVERSE] == intplus[x]

In[59]:= composite[image[inverse[INTADD], set[x_]], INVERSE] := intplus[x]
```

Theorem. (Normalization rule.)

```
In[60]:= composite[INVERSE, image[inverse[INTADD], set[x]]] // DoubleInverse
Out[60]= composite[INVERSE, image[inverse[INTADD], set[x]]] == inverse[intplus[x]]

In[61]:= composite[INVERSE, image[inverse[INTADD], set[x_]]] := inverse[intplus[x]]
```

Theorem.

```
In[62]:= Assoc[image[inverse[INTADD], set[x]], INVERSE, INVERSE] // Reverse
Out[62]= composite[intplus[x], INVERSE] == image[inverse[INTADD], set[x]]

In[63]:= composite[intplus[x_], INVERSE] := image[inverse[INTADD], set[x]]
```

Theorem.

```
In[64]:= composite[INVERSE, inverse[intplus[x]]] // DoubleInverse
Out[64]= composite[INVERSE, inverse[intplus[x]]] == image[inverse[INTADD], set[x]]

In[65]:= composite[INVERSE, inverse[intplus[x_]]] := image[inverse[INTADD], set[x]]
```

Theorem.

```
In[66]:= Assoc[INVERSE, image[inverse[INTADD], set[x]], INVERSE] // Reverse
Out[66]= composite[inverse[intplus[x]], INVERSE] == composite[INVERSE, intplus[x]]

In[67]:= composite[inverse[intplus[x_]], INVERSE] := composite[INVERSE, intplus[x]]
```

intplus[inverse[x]]

Theorem.

```
In[68]:= Assoc[intplus[inverse[x]], INVERSE, INVERSE]
Out[68]= intplus[inverse[x]] == inverse[intplus[composite[Id, x]]]

In[69]:= intplus[inverse[x_]] := inverse[intplus[composite[Id, x]]]
```

Theorem.

```
In[70]:= SubstTest[composite, intplus[t], INVERSE, t → inverse[x]]

Out[70]= image[inverse[INTADD], set[inverse[x]]] == composite[INVERSE, intplus[composite[Id, x]]]

In[71]:= image[inverse[INTADD], set[inverse[x_]]] :=
          composite[INVERSE, intplus[composite[Id, x]]]
```

reify rule

Theorem. Reify rule.

```
In[72]:= SubstTest[reify, x, composite[t, LEFT[f[x]]], t → INTADD] // Reverse

Out[72]= reify[x, intplus[f[x]]] ==
          composite[cross[INVERSE, Id], inverse[INTADD], VERTSECT[reify[x, f[x]]]]

In[73]:= reify[x_, intplus[y_]] :=
          composite[cross[INVERSE, Id], inverse[INTADD], VERTSECT[reify[x, y]]]
```

serendipity

The following was discovered, but not used.

Theorem.

```
In[74]:= Assoc[IMAGE[id[cart[V, V]]], id[Z], INTADD]

Out[74]= composite[IMAGE[id[cart[V, V]]], INTADD] == INTADD

In[75]:= composite[IMAGE[id[cart[V, V]]], INTADD] := INTADD
```