INTDIV restricted to range[PLUS]

Johan G. F. Belinfante
2012 July 20

In[1]:= **SetDirectory["l:"]; << goedel.12jul19a**

```
:Package Title: goedel.12jul19a                2012 July 19 at 11:40 a.m.

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

It is now:  2012 Jul 20 at 14:46

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 Jul 20 at 15:1
```

summary

The relation **INTDIV** is not antisymmetric, and hence not a partial order, but its restriction to **range[PLUS]** is.

derivation

Theorem. Simplification rule.

In[2]:= **Assoc[PLUS, inverse[PLUS], MIXDIV] // Reverse**

Out[2]= composite[id[range[PLUS]], MIXDIV] == composite[PLUS, DIV]

In[3]:= **composite[id[range[PLUS]], MIXDIV] := composite[PLUS, DIV]**

Theorem. Simplification rule.

In[4]:= **Assoc[INTDIV, PLUS, inverse[PLUS]] // Reverse**

Out[4]= composite[MIXDIV, inverse[PLUS]] == composite[INTDIV, id[range[PLUS]]]

In[5]:= **composite[MIXDIV, inverse[PLUS]] := composite[INTDIV, id[range[PLUS]]]**

It is unclear how best to orient the following rewrite rule. The chosen direction is tentative.

Theorem. Simplification rule.

```
In[6]:= Assoc[id[range[PLUS]], MIXDIV, inverse[PLUS]] // Reverse

Out[6]= composite[PLUS, DIV, inverse[PLUS]] ==
         composite[id[range[PLUS]], INTDIV, id[range[PLUS]]]

In[7]:= composite[PLUS, DIV, inverse[PLUS]] :=
         composite[id[range[PLUS]], INTDIV, id[range[PLUS]]]
```

Corollary.

```
In[28]:= composite[PLUS, inverse[DIV], inverse[PLUS]] // DoubleInverse

Out[28]= composite[PLUS, inverse[DIV], inverse[PLUS]] ==
          composite[id[range[PLUS]], inverse[INTDIV], id[range[PLUS]]]

In[29]:= composite[PLUS, inverse[DIV], inverse[PLUS]] :=
          composite[id[range[PLUS]], inverse[INTDIV], id[range[PLUS]]]
```

a partial order

Theorem.

```
In[8]:= SubstTest[TRANSITIVE, restrict[trv[t], x, x], t → INTDIV] // Reverse

Out[8]= TRANSITIVE[composite[id[x], INTDIV, id[x]]] == True

In[9]:= TRANSITIVE[composite[id[x_], INTDIV, id[x_]]] := True
```

Lemma.

```
In[11]:= SubstTest[ANTISYMMETRIC, composite[PLUS, t, inverse[PLUS]], t → DIV] // Reverse

Out[11]= subclass[composite[id[range[PLUS]],
            intersection[INTDIV, inverse[INTDIV]], id[range[PLUS]]]], Id] == True

In[12]:= % /. Equal → SetDelayed
```

Theorem.

```
In[14]:= PARTIALORDER[composite[id[range[PLUS]], INTDIV, id[range[PLUS]]]] // AssertTest

Out[14]= PARTIALORDER[composite[id[range[PLUS]], INTDIV, id[range[PLUS]]]] == True

In[15]:= PARTIALORDER[composite[id[range[PLUS]], INTDIV, id[range[PLUS]]]] := True
```

Corollary. A temporary rewrite rule.

```
In[33]:= SubstTest[intersection, po[t], inverse[po[t]],
           t -> composite[id[range[PLUS]], INTDIV, id[range[PLUS]]]] // Reverse

Out[33]= composite[id[range[PLUS]], intersection[INTDIV, inverse[INTDIV]], id[range[PLUS]]] ==
          id[range[PLUS]]
```

```
In[34]:= % /. Equal→SetDelayed
```

Later a rewrite rule for **INTDIV** ⋂ **inverse[INTDIV]** will be derived.

divisibility for positive integers

Lemma. Simplification rule.

```
In[17]:= ApComp[inverse[PLUS], PLUS, nat[x]]

Out[17]= APPLY[inverse[PLUS], plus[nat[x]]] == nat[x]

In[18]:= APPLY[inverse[PLUS], plus[nat[x_]]] := nat[x]
```

Theorem.

```
In[19]:= (member[pair[s, t], composite[inverse[funpart[u]], v, funpart[u]]] // AssertTest) /.
          {s → plus[nat[x]], t → plus[nat[y]], u → inverse[PLUS], v → DIV}

Out[19]= member[pair[plus[nat[x]], plus[nat[y]]], INTDIV] == member[pair[nat[x], nat[y]], DIV]

In[20]:= member[pair[plus[nat[x_]], plus[nat[y_]]], INTDIV] := member[pair[nat[x], nat[y]], DIV]
```

a replacement rule

Observation: An old rewrite rule no longer works.

```
In[21]:= union[composite[MIXDIV, inverse[PLUS], INVERSE], composite[MIXDIV, inverse[PLUS]]]

Out[21]= union[composite[INTDIV, id[range[PLUS]]], composite[INTDIV, id[range[PLUS]], INVERSE]]
```

Theorem. Replacement rule.

```
In[22]:= union[composite[INTDIV, id[range[PLUS]]],
          composite[INTDIV, id[range[PLUS]], INVERSE]] // FastReifNormality

Out[22]= union[composite[INTDIV, id[range[PLUS]]],
          composite[INTDIV, id[range[PLUS]], INVERSE]] == INTDIV

In[23]:= union[composite[INTDIV, id[range[PLUS]]],
          composite[INTDIV, id[range[PLUS]], INVERSE]] := INTDIV
```

Corollary.

```
In[24]:= union[composite[id[range[PLUS]], inverse[INTDIV]],
          composite[INVERSE, id[range[PLUS]], inverse[INTDIV]]] // DoubleInverse

Out[24]= union[composite[id[range[PLUS]], inverse[INTDIV]],
          composite[INVERSE, id[range[PLUS]], inverse[INTDIV]]] == inverse[INTDIV]
```

```
In[25]:= union[composite[id[range[PLUS]], inverse[INTDIV]],
           composite[INVERSE, id[range[PLUS]], inverse[INTDIV]]] := inverse[INTDIV]
```

INTDIV ⋂ inverse[INTDIV]

Lemma. Simplification rule.

```
In[36]:= Assoc[id[P[cart[V, V]]], id[Z], inverse[INTDIV]]

Out[36]= composite[id[P[cart[V, V]]], inverse[INTDIV]] == inverse[INTDIV]

In[37]:= composite[id[P[cart[V, V]]], inverse[INTDIV]] := inverse[INTDIV]
```

Theorem. Simplification rule.

```
In[38]:= AssInt[composite[INTDIV, IMAGE[SWAP]], cart[Z, V], inverse[INTDIV]]

Out[38]= intersection[composite[INTDIV, IMAGE[SWAP]], inverse[INTDIV]] ==
          intersection[INTDIV, inverse[INTDIV]]

In[39]:= % /. Equal → SetDelayed
```

Lemma.

```
In[40]:= intersection[composite[INTDIV, id[range[PLUS]], INVERSE], inverse[INTDIV]] //
          FastReifNormality

Out[40]= intersection[composite[INTDIV, id[range[PLUS]], INVERSE], inverse[INTDIV]] ==
          composite[intersection[INTDIV, inverse[INTDIV]], id[image[INVERSE, range[PLUS]]]]

In[41]:= % /. Equal → SetDelayed
```

Lemma.

```
In[43]:= SubstTest[intersection, composite[INVERSE, u],
          composite[INVERSE, v], {u → INTDIV, v → inverse[INTDIV]}]

Out[43]= composite[INVERSE, intersection[INTDIV, inverse[INTDIV]]] ==
          intersection[INTDIV, inverse[INTDIV]]

In[44]:= % /. Equal → SetDelayed
```

Corollary.

```
In[45]:= composite[intersection[INTDIV, inverse[INTDIV]], INVERSE] // DoubleInverse

Out[45]= composite[intersection[INTDIV, inverse[INTDIV]], INVERSE] ==
          intersection[INTDIV, inverse[INTDIV]]

In[46]:= % /. Equal → SetDelayed
```

Theorem. Simplification rule.

```
In[47]:= Map[composite[INVERSE, inverse[#], INVERSE] &,
           Map[composite[INVERSE, inverse[#], INVERSE] &, SubstTest[intersection,
             union[u, v], inverse[union[u, v]], {u -> composite[INTDIV, id[range[PLUS]]],
              v -> composite[INTDIV, id[range[PLUS]], INVERSE]}]]] // Reverse

Out[47]= intersection[INTDIV, inverse[INTDIV]] == union[composite[id[Z], INVERSE], id[Z]]

In[48]:= intersection[INTDIV, inverse[INTDIV]] := union[composite[id[Z], INVERSE], id[Z]]
```

INTDIV restricted to negative integers

Lemma. Simplification rule.

```
In[54]:= Assoc[composite[INVERSE, id[x]], INVERSE, INTDIV]

Out[54]= composite[INVERSE, id[x], INTDIV] == composite[id[image[INVERSE, x]], INTDIV]

In[55]:= composite[INVERSE, id[x_], INTDIV] := composite[id[image[INVERSE, x]], INTDIV]
```

Lemma. Simplification rule.

```
In[60]:= Assoc[INTDIV, INVERSE, composite[id[x], INVERSE]] // Reverse

Out[60]= composite[INTDIV, id[x], INVERSE] == composite[INTDIV, id[image[INVERSE, x]]]

In[61]:= composite[INTDIV, id[x_], INVERSE] := composite[INTDIV, id[image[INVERSE, x]]]
```

Theorem. Divisibility for negative integers is a partial order.

```
In[62]:= SubstTest[PARTIALORDER, composite[oopart[t], po[x], inverse[oopart[t]]],
           {t → INVERSE, x -> composite[id[range[PLUS]], INTDIV, id[range[PLUS]]]}] // Reverse

Out[62]= PARTIALORDER[composite[id[image[INVERSE, range[PLUS]]],
            INTDIV, id[image[INVERSE, range[PLUS]]]]] == True

In[63]:= PARTIALORDER[composite[id[image[INVERSE, range[PLUS]]],
            INTDIV, id[image[INVERSE, range[PLUS]]]]] := True
```