a twist relation for binary operations

Johan G. F. Belinfante
2012 June 24

```
In[1]:= SetDirectory["l:"]; << goedel.12jun18a

:Package Title: goedel.12jun18a                    2012 June 18 at 2:40 p.m.

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

It is now:  2012 Jun 24 at 6:41

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 Jun 24 at 6:58
```

summary

The equivalence relation **EQUIDIFF** figures prominently in constructing the integers from the natural numbers. In this notebook a small part of that theory is studied in a more general setting. The relation studied here need not in general be an equivalence, and no embedding theorem for general binary operations is derived.

a general twist formula

The general twist formula derived in this section involves two binary operations, as well as their Curried counterparts.

Lemma. Simplification rule.

```
In[2]:= SubstTest[composite, intersection[composite[inverse[FIRST], flip[t]],
          composite[inverse[SECOND], SECOND]], inverse[FIRST], t → flip[x]] // Reverse

Out[2]= composite[
          intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], SECOND]],
          inverse[FIRST]] == inverse[rotate[composite[x, SWAP]]]
```

```
In[3]:= composite[
          intersection[composite[inverse[FIRST], x_], composite[inverse[SECOND], SECOND]],
          inverse[FIRST]] := inverse[rotate[composite[x, SWAP]]]
```

Lemma. Simplification rule.

```
In[4]:= SubstTest[twist, twist[t],
          t -> composite[RIF, cross[inverse[rotate[x]], composite[SWAP, inverse[rotate[y]]]]]]

Out[4]= composite[RIF, cross[inverse[rotate[x]], composite[SWAP, inverse[rotate[y]]]]] ==
          inverse[twist[composite[inverse[y], x]]]

In[5]:= composite[RIF, cross[inverse[rotate[x_]], composite[SWAP, inverse[rotate[y_]]]]] :=
          inverse[twist[composite[inverse[y], x]]]
```

Theorem.   A general twist formula.

```
In[6]:= composite[inverse[E], COMPOSE, cross[composite[INVERSE, APPLY[CURRY, binop[x]]],
            APPLY[CURRY, binop[y]]]] // FastReifTriNormality

Out[6]= composite[inverse[E], COMPOSE,
           cross[composite[INVERSE, APPLY[CURRY, binop[x]]], APPLY[CURRY, binop[y]]]] ==
          composite[SWAP, twist[composite[inverse[binop[y]], binop[x]]]]

In[7]:= composite[inverse[E], COMPOSE,
           cross[composite[INVERSE, APPLY[CURRY, binop[x_]]], APPLY[CURRY, binop[y_]]]] :=
          composite[SWAP, twist[composite[inverse[binop[y]], binop[x]]]]
```

a special case

In this section the special case **x = y = NATADD** is studied.   The Curried version of **NATADD** is **PLUS**.

```
In[8]:= APPLY[CURRY, NATADD]

Out[8]= PLUS
```

The left hand side of the twist formula reduces to **EQUIDIFF**.

```
In[9]:= composite[inverse[E], COMPOSE, cross[composite[INVERSE, PLUS], PLUS]]

Out[9]= EQUIDIFF
```

The right hand side also reduces to **EQUIDIFF**.

```
In[10]:= composite[SWAP, twist[composite[inverse[NATADD], NATADD]]]

Out[10]= EQUIDIFF
```

a simplification rule

In this section it is shown that the function **INVERSE** in twist formula derived in the preceding section could be replaced with the function **IMAGE[SWAP]**. A simplification rule is derived that makes it unnecessary to retain two separate twist formulas.

Theorem. Simplification rule.

```
In[11]:= Assoc[IMAGE[SWAP], id[P[cart[V, V]]], APPLY[CURRY, binop[x]]]

Out[11]= composite[IMAGE[SWAP], APPLY[CURRY, binop[x]]] ==
          composite[INVERSE, APPLY[CURRY, binop[x]]]

In[12]:= composite[IMAGE[SWAP], APPLY[CURRY, binop[x_]]] :=
          composite[INVERSE, APPLY[CURRY, binop[x]]]
```

The same simplification can be made when the binary operation is replaced with its flip.

Corollary. A dual simplification rule.

```
In[13]:= SubstTest[composite, IMAGE[SWAP], APPLY[CURRY, binop[t]], t → flip[binop[x]]] // Reverse

Out[13]= composite[IMAGE[SWAP], APPLY[CURRY, composite[binop[x], SWAP]]] ==
          composite[INVERSE, APPLY[CURRY, composite[binop[x], SWAP]]]

In[14]:= composite[IMAGE[SWAP], APPLY[CURRY, composite[binop[x_], SWAP]]] :=
          composite[INVERSE, APPLY[CURRY, composite[binop[x], SWAP]]]
```

an inclusion

In this section an inclusion is derived for the special case that the two binary operations are related by **flip**.

Lemma. Simplification rule.

```
In[18]:= (fix[composite[SWAP, twist[composite[inverse[binop[x]], binop[y]]]]] //
            RelnNormality) /. y → flip[binop[x]]

Out[18]= fix[composite[SWAP, twist[composite[inverse[binop[x]], binop[x], SWAP]]]] ==
          domain[binop[x]]

In[19]:= fix[composite[SWAP, twist[composite[inverse[binop[x_]], binop[x_], SWAP]]]] :=
          domain[binop[x]]
```

Theorem. An inclusion.

```
In[20]:= SubstTest[subclass, domain[funpart[t]], fix[composite[inverse[E], funpart[t]]],
           t → composite[COMPOSE, cross[composite[INVERSE, APPLY[CURRY, binop[t]]],
              APPLY[CURRY, binop[x]]]]] /. t → flip[binop[x]]

Out[20]= subclass[composite[COMPOSE,
            cross[composite[INVERSE, APPLY[CURRY, composite[binop[x], SWAP]]],
             APPLY[CURRY, binop[x]]]], E] == True

In[21]:= subclass[composite[COMPOSE,
            cross[composite[INVERSE, APPLY[CURRY, composite[binop[x_], SWAP]]],
             APPLY[CURRY, binop[x_]]]], E] := True
```

Corollary. A dual result.

```
In[22]:= SubstTest[subclass, composite[COMPOSE,
           cross[composite[INVERSE, APPLY[CURRY, composite[binop[t], SWAP]]],
            APPLY[CURRY, binop[t]]]], E, t → flip[binop[x]]] // Reverse

Out[22]= subclass[composite[COMPOSE, cross[composite[INVERSE, APPLY[CURRY, binop[x]]],
            APPLY[CURRY, composite[binop[x], SWAP]]]], E] == True

In[23]:= subclass[composite[COMPOSE, cross[composite[INVERSE, APPLY[CURRY, binop[x_]]],
            APPLY[CURRY, composite[binop[x_], SWAP]]]], E] := True
```

Corollary. A special case of interest.

```
In[24]:= SubstTest[subclass, composite[COMPOSE,
           cross[composite[INVERSE, APPLY[CURRY, composite[binop[t], SWAP]]],
            APPLY[CURRY, binop[t]]]], E, t → NATADD] // Reverse

Out[24]= subclass[composite[VERTSECT[EQUIDIFF], id[cart[omega, omega]]], E] == True

In[25]:= subclass[composite[VERTSECT[EQUIDIFF], id[cart[omega, omega]]], E] := True
```

Comment. The above result could also have been deduced from the following.

```
In[26]:= composite[id[Z], E]

Out[26]= composite[VERTSECT[EQUIDIFF], id[cart[omega, omega]]]
```