binary closure of EQUIDIFF under vector addition

Johan G. F. Belinfante
2011 October 13

```
In[1]:= SetDirectory["l:"]; << goedel.11oct07a

        :Package Title: goedel.11oct07a                2011 October 7 at 5:00 p.m.

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

        It is now:  2011 Oct 13 at 14:41

        Loading Simplification Rules

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

        weightlimit = 40

        Loading completed.

        It is now:  2011 Oct 13 at 14:53
```

summary

In this notebook it is shown that **EQUIDIFF** is binary closed under the direct square of the direct square of **NATADD**. In fact, an equation holds. The derivation is done entirely without introducing variables. If one were to introduce variables, one would need eight of them. The relation **EQUIDIFF** holds pairs of pairs of natural numbers **((a ,b), (c, d))** satisfying **a + d = b + c**. What needs to be shown is that the vector sum of two such vectors is another. That is, if **a + d = b + c** and **e + h = f + g**, then **(a + e) + (d + h) = (b + f) + (c + g)**. The associative and commutative laws of natural number addition are needed to do this.

temporary abbreviation

To save some writing, the following abbreviation is used occasionally.

```
In[2]:= dirsq[x_] := direct[x, x]
```

inclusion in one direction

An inclusion in one direction is established in this section.

Lemma. The ordered pair of pairs **((0, 0), (0, 0))** belongs to **EQUIDIFF**.

```
In[3]:= subclass[cartsq[cartsq[set[0]]], EQUIDIFF] // AssertTest

Out[3]= subclass[cart[cart[set[0], set[0]], cart[set[0], set[0]]], EQUIDIFF] == True

In[4]:= subclass[cart[cart[set[0], set[0]], cart[set[0], set[0]]], EQUIDIFF] := True
```

Lemma. Simplification rule.

```
In[5]:= composite[FIRST, id[cart[V, cartsq[set[0]]]], inverse[dirsq[NATADD]]] //
         FastReifNormality

Out[5]= composite[FIRST, id[cart[V, cart[set[0], set[0]]]], TWIST,
          cross[inverse[NATADD], inverse[NATADD]]] == id[cart[omega, omega]]

In[6]:= % /. Equal→ SetDelayed
```

Theorem. An inclusion in one direction.

```
In[7]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
          {t → composite[dirsq[dirsq[NATADD]], id[cart[EQUIDIFF, V]], inverse[SECOND]],
           u → cartsq[cartsq[set[0]]], v → EQUIDIFF}] // Reverse

Out[7]= subclass[EQUIDIFF, composite[cross[NATADD, NATADD], TWIST,
           cross[EQUIDIFF, EQUIDIFF], TWIST, cross[inverse[NATADD], inverse[NATADD]]]] == True

In[8]:= % /. Equal→ SetDelayed
```

the medial law

Lemma. Any commutative associative binary operation satisfies the medial law.

```
In[9]:= SubstTest[implies, and[associative[x], equal[flip[x], x]],
          equal[composite[x, dirsq[x]], composite[x, cross[x, x]]], x → dirsq[NATADD]] // Reverse

Out[9]= equal[composite[cross[NATADD, NATADD], TWIST, cross[
            composite[cross[NATADD, NATADD], TWIST], composite[cross[NATADD, NATADD], TWIST]]],
          composite[cross[NATADD, NATADD], TWIST, cross[composite[cross[NATADD, NATADD], TWIST],
            composite[cross[NATADD, NATADD], TWIST]], TWIST]] == True

In[10]:= % /. Equal→ SetDelayed
```

Theorem. The medial law allows one to remove one copy of **TWIST**.

```
In[11]:= SubstTest[implies, equal[u, v], equal[image[inverse[u], Id], image[inverse[v], Id]],
           {u -> composite[cross[NATADD, NATADD], TWIST, cross[composite[
                cross[NATADD, NATADD], TWIST], composite[cross[NATADD, NATADD], TWIST]]],
            v -> composite[cross[NATADD, NATADD], TWIST, cross[composite[cross[NATADD, NATADD],
                TWIST], composite[cross[NATADD, NATADD], TWIST]], TWIST]}] // Reverse

Out[11]= equal[composite[cross[Id, SWAP],
           twist[composite[TWIST, cross[inverse[NATADD], inverse[NATADD]],
             EQUIDIFF, cross[NATADD, NATADD], TWIST]], cross[Id, SWAP]],
          composite[cross[SWAP, SWAP], TWIST, cross[inverse[NATADD], inverse[NATADD]],
           EQUIDIFF, cross[NATADD, NATADD], TWIST]] == True

In[12]:= % /. Equal→ SetDelayed
```

---

main theorem

Lemma.

```
In[13]:= SubstTest[implies, equal[u, v],
           equal[composite[t, u, w], composite[t, v, w]], {t -> cross[SWAP, Id],
            u -> composite[cross[SWAP, SWAP], TWIST, cross[inverse[NATADD], inverse[NATADD]],
              EQUIDIFF, cross[NATADD, NATADD], TWIST], v -> composite[cross[Id, SWAP],
              twist[composite[TWIST, cross[inverse[NATADD], inverse[NATADD]], EQUIDIFF, cross[
                 NATADD, NATADD], TWIST]], cross[Id, SWAP]], w -> cross[Id, SWAP]}] // Reverse

Out[13]= equal[composite[cross[SWAP, SWAP],
           twist[composite[TWIST, cross[inverse[NATADD], inverse[NATADD]],
             EQUIDIFF, cross[NATADD, NATADD], TWIST]], id[cart[V, cart[V, V]]]],
          composite[cross[Id, SWAP], TWIST, cross[inverse[NATADD], inverse[NATADD]],
           EQUIDIFF, cross[NATADD, NATADD], TWIST, cross[Id, SWAP]]] == True

In[14]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[15]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → inverse[cross[dirsq[NATADD], dirsq[NATADD]]],
            u → cartsq[id[omega]], v → EQUIDIFF}] // Reverse

Out[15]= subclass[cart[composite[SWAP, EQUIDIFF], composite[SWAP, EQUIDIFF]],
           composite[TWIST, cross[inverse[NATADD], inverse[NATADD]],
            EQUIDIFF, cross[NATADD, NATADD], TWIST]] == True

In[16]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[17]:= SubstTest[implies, subclass[u, v],
           subclass[image[t, u], image[t, v]], {t → composite[cross[Id, cross[SWAP, SWAP]],
              TWIST, inverse[cross[dirsq[NATADD], dirsq[NATADD]]]],
            u → cartsq[id[omega]], v → EQUIDIFF}] // Reverse

Out[17]= subclass[cross[EQUIDIFF, EQUIDIFF], composite[cross[SWAP, SWAP],
           twist[composite[TWIST, cross[inverse[NATADD], inverse[NATADD]],
             EQUIDIFF, cross[NATADD, NATADD], TWIST]]]] == True

In[18]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[19]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → composite[TWIST, cross[Id, cross[SWAP, SWAP]],
              TWIST, inverse[cross[dirsq[NATADD], dirsq[NATADD]]]],
            u → cartsq[id[omega]], v → EQUIDIFF}] // Reverse

Out[19]= subclass[cart[EQUIDIFF, EQUIDIFF],
           composite[cross[Id, SWAP], TWIST, cross[inverse[NATADD], inverse[NATADD]],
            EQUIDIFF, cross[NATADD, NATADD], TWIST, cross[Id, SWAP]]] == True

In[20]:= % /. Equal→ SetDelayed
```

Theorem.

```
In[21]:= SubstTest[implies, and[subclass[u, v], equal[v, w]], subclass[u, w],
           {u -> cross[EQUIDIFF, EQUIDIFF], v → composite[cross[SWAP, SWAP],
              twist[composite[TWIST, cross[inverse[NATADD], inverse[NATADD]],
                EQUIDIFF, cross[NATADD, NATADD], TWIST]], id[cart[V, cart[V, V]]]],
            w -> composite[cross[Id, SWAP], TWIST, cross[inverse[NATADD], inverse[NATADD]],
              EQUIDIFF, cross[NATADD, NATADD], TWIST, cross[Id, SWAP]]}] // Reverse

Out[21]= subclass[cross[EQUIDIFF, EQUIDIFF],
           composite[cross[Id, SWAP], TWIST, cross[inverse[NATADD], inverse[NATADD]],
            EQUIDIFF, cross[NATADD, NATADD], TWIST, cross[Id, SWAP]]] == True

In[22]:= % /. Equal→ SetDelayed
```

Theorem.

```
In[23]:= Assoc[composite[TWIST, cross[inverse[NATADD], inverse[NATADD]], EQUIDIFF,
            cross[NATADD, NATADD]], SWAP, composite[TWIST, cross[SWAP, SWAP]]] // Reverse

Out[23]= composite[cross[SWAP, SWAP], TWIST, cross[inverse[NATADD], inverse[NATADD]],
           EQUIDIFF, cross[NATADD, NATADD], TWIST, cross[SWAP, SWAP]] == composite[TWIST,
           cross[inverse[NATADD], inverse[NATADD]], EQUIDIFF, cross[NATADD, NATADD], TWIST]

In[24]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[25]:= SubstTest[implies, subclass[u, v], subclass[composite[t, u, w], composite[t, v, w]],
           {t -> cross[SWAP, Id], u → cross[EQUIDIFF, EQUIDIFF],
            v -> composite[cross[Id, SWAP], TWIST, cross[inverse[NATADD], inverse[NATADD]],
              EQUIDIFF, cross[NATADD, NATADD], TWIST, cross[Id, SWAP]],
            w -> cross[SWAP, Id]}] // Reverse

Out[25]= subclass[cross[EQUIDIFF, EQUIDIFF],
           composite[TWIST, cross[inverse[NATADD], inverse[NATADD]],
            EQUIDIFF, cross[NATADD, NATADD], TWIST]] == True

In[26]:= % /. Equal→ SetDelayed
```

Theorem.

```
In[27]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → cross[dirsq[NATADD], dirsq[NATADD]], u -> cross[EQUIDIFF, EQUIDIFF],
            v -> composite[TWIST, cross[inverse[NATADD], inverse[NATADD]],
              EQUIDIFF, cross[NATADD, NATADD], TWIST]}] // Reverse

Out[27]= subclass[composite[cross[NATADD, NATADD], TWIST, cross[EQUIDIFF, EQUIDIFF],
            TWIST, cross[inverse[NATADD], inverse[NATADD]]], EQUIDIFF] == True

In[28]:= % /. Equal→ SetDelayed
```

Main Theorem. An equation that says **EQUIDIFF** + **EQUIDIFF** = **EQUIDIFF**, where + denotes vector addition of sets.

```
In[29]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> composite[cross[NATADD, NATADD], TWIST, cross[EQUIDIFF, EQUIDIFF],
              TWIST, cross[inverse[NATADD], inverse[NATADD]]], v -> EQUIDIFF}]

Out[29]= equal[EQUIDIFF, composite[cross[NATADD, NATADD], TWIST,
            cross[EQUIDIFF, EQUIDIFF], TWIST, cross[inverse[NATADD], inverse[NATADD]]]] == True

In[30]:= composite[cross[NATADD, NATADD], TWIST, cross[EQUIDIFF, EQUIDIFF],
           TWIST, cross[inverse[NATADD], inverse[NATADD]]] := EQUIDIFF
```

It follows from this that **EQUIDIFF** is binary closed under the direct square of the direct square of **NATADD**.

```
In[31]:= member[EQUIDIFF, binclosed[dirsq[dirsq[NATADD]]]]

Out[31]= True
```