binhom[x,y] ⊂ binclosed[direct[x,y]]

Johan G. F. Belinfante
2011 October 13

```
In[1]:= SetDirectory["l:"]; << goedel.11oct12a

        :Package Title: goedel.11oct12a              2011 October 12 at 10:45 a.m.

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

        It is now:  2011 Oct 13 at 16:33

        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 16:45
```

summary

If **x** is a function, then **binhom[x, y] ⊂ binclosed[direct[x, y]]**.

derivation

Theorem.

```
In[16]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → member[t, binhom[funpart[x], y]],
            p2 → equal[composite[t, funpart[x]], composite[y, cross[t, t]]],
            p3 → subclass[composite[y, cross[t, t], inverse[funpart[x]]], t]}]] // Reverse

Out[16]= or[not[member[t, binhom[funpart[x], y]]],
          subclass[composite[y, cross[t, t], inverse[funpart[x]]], t]] == True

In[17]:= or[not[member[t_, binhom[funpart[x_], y_]]],
          subclass[composite[y_, cross[t_, t_], inverse[funpart[x_]]], t_]] := True
```

Corollary. (Remove the **funpart** wrapper.)

```
In[19]:= SubstTest[implies, equal[x, funpart[w]], or[not[member[t, binhom[x, y]]],
           subclass[composite[y, cross[t, t], inverse[x]], t]], w → x] // Reverse

Out[19]= or[not[FUNCTION[x]], not[member[t, binhom[x, y]]],
           subclass[composite[y, cross[t, t], inverse[x]], t]] == True

In[20]:= or[not[FUNCTION[x_]], not[member[t_, binhom[x_, y_]]],
           subclass[composite[y_, cross[t_, t_], inverse[x_]], t_]] := True
```

Theorem. (Eliminate the variable **t**.)

```
In[22]:= Map[equal[V, #] &,
          dif[binhom[funpart[x], y], binclosed[direct[funpart[x], y]]] // complement //
           Normality]

Out[22]= subclass[binhom[funpart[x], y],
           binclosed[composite[cross[funpart[x], y], TWIST]]] == True

In[23]:= subclass[binhom[funpart[x_], y_],
           binclosed[composite[cross[funpart[x_], y_], TWIST]]] := True
```

Corollary. (Remove the **funpart** wrapper.)

```
In[24]:= SubstTest[implies, equal[x, funpart[w]],
           subclass[binhom[x, y], binclosed[composite[cross[x, y], TWIST]]], w → x] // Reverse

Out[24]= or[not[FUNCTION[x]],
           subclass[binhom[x, y], binclosed[composite[cross[x, y], TWIST]]]] == True

In[25]:= or[not[FUNCTION[x_]],
           subclass[binhom[x_, y_], binclosed[composite[cross[x_, y_], TWIST]]]] := True
```

an example

Theorem.

```
In[27]:= SubstTest[subclass, binhom[funpart[x], y],
           binclosed[composite[cross[funpart[x], y], TWIST]], {x → NATADD, y → NATADD}] // Reverse

Out[27]= subclass[binhom[NATADD, NATADD],
           binclosed[composite[cross[NATADD, NATADD], TWIST]]] == True

In[28]:= subclass[binhom[NATADD, NATADD],
           binclosed[composite[cross[NATADD, NATADD], TWIST]]] := True
```

Corollary.

```
In[30]:= SubstTest[implies, and[member[u, v], subclass[v, w]],
           member[u, w], {u → times[x], v -> binhom[NATADD, NATADD],
            w -> binclosed[composite[cross[NATADD, NATADD], TWIST]]}] // Reverse

Out[30]= or[not[member[x, omega]], subclass[
            composite[NATADD, cross[times[x], times[x]], inverse[NATADD]], times[x]]] == True

In[31]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma.

```
In[34]:= SubstTest[implies, empty[t],
           subclass[composite[NATADD, cross[t, t], inverse[NATADD]], t], t → times[x]] // Reverse

Out[34]= or[member[x, omega], subclass[
            composite[NATADD, cross[times[x], times[x]], inverse[NATADD]], times[x]]] == True

In[35]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem. (Remove redundant literal.)

```
In[37]:= SubstTest[and, implies[p, q], or[p, q], {p -> member[x, omega], q ->
              subclass[composite[NATADD, cross[times[x], times[x]], inverse[NATADD]], times[x]]}]

Out[37]= subclass[composite[NATADD, cross[times[x], times[x]], inverse[NATADD]], times[x]] ==
          True

In[38]:= subclass[
           composite[NATADD, cross[times[x_], times[x_]], inverse[NATADD]], times[x_]] := True
```

A better rewrite rule can be derived.

Theorem.

```
In[41]:= Assoc[times[x], NATADD, inverse[NATADD]] // Reverse

Out[41]= composite[NATADD, cross[times[x], times[x]], inverse[NATADD]] == times[x]

In[42]:= composite[NATADD, cross[times[x_], times[x_]], inverse[NATADD]] := times[x]
```

Corollary. A variable-free statement.

```
In[44]:= Map[rotate[inverse[#]] &,
          SubstTest[reify, x, composite[t, cross[times[x], times[x]], inverse[t]], t → NATADD]]

Out[44]= composite[NATADD, cross[NATMUL, NATMUL], TWIST, cross[DUP, inverse[NATADD]]] == NATMUL

In[45]:= composite[NATADD, cross[NATMUL, NATMUL], TWIST, cross[DUP, inverse[NATADD]]] := NATMUL
```