binhom[x, y] ⊂ domain[eval[z]]

Johan G. F. Belinfante
2013 July 22

```
In[1]:= SetDirectory["l:"]; << goedel.13jul20a

        :Package Title: goedel.13jul20a                    2013 July 20 at 1:00 p.m.

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

        It is now:  2013 Jul 22 at 13:17

        Loading Simplification Rules

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

        weightlimit = 40

        Loading completed.

        It is now:  2013 Jul 22 at 13:34
```

summary

A rewrite rule is derived for the inclusion **binhom[x, y] ⊂ domain[eval[z]]** that subsumes several existing rules for special cases.

derivation

Lemma. An implication in one direction.

```
In[2]:= Map[implies[member[z, fix[domain[x]]], #] &,
           SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
            {u → binhom[x, y], v → map[fix[domain[x]], fix[domain[y]]], w → domain[eval[z]]}]] //
          MapNotNot // Reverse

Out[2]= or[not[member[z, fix[domain[x]]]], subclass[binhom[x, y], domain[eval[z]]]] == True

In[3]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

A weaker implication in the reverse direction will next be derived.

Lemma. (Starting point for an implication in the reverse direction.)

```
In[4]:= SubstTest[implies, and[member[t, u], subclass[u, v]], member[t, v],
          {u → binhom[x, y], v → domain[eval[z]]}] // Reverse

Out[4]= or[member[z, domain[funpart[t]]], not[member[t, binhom[x, y]]],
          not[subclass[binhom[x, y], domain[eval[z]]]]] == True

In[5]:= (% /. {t → t_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma. (Eliminating **funpart**.)

```
In[6]:= SubstTest[implies, equal[t, funpart[w]],
           or[member[z, domain[t]], not[member[w, binhom[x, y]]],
            not[subclass[binhom[x, y], domain[eval[z]]]]], w → t] // Reverse // MapNotNot

Out[6]= or[member[z, domain[t]], not[member[t, binhom[x, y]]],
          not[subclass[binhom[x, y], domain[eval[z]]]]] == True

In[7]:= (% /. {t → t_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma. (Eliminating reference to **domain[t]**.)

```
In[8]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[p1, p4],
           implies[and[p3, p4], p5], not[implies[and[p1, p2], p5]], {p1 → member[t, binhom[x, y]],
            p2 -> subclass[binhom[x, y], domain[eval[z]]], p3 -> member[z, domain[t]],
            p4 → equal[domain[t], fix[domain[x]]], p5 → member[z, fix[domain[x]]]}]] // Reverse

Out[8]= or[member[z, fix[domain[x]]], not[member[t, binhom[x, y]]],
          not[subclass[binhom[x, y], domain[eval[z]]]]] == True

In[9]:= (% /. {t → t_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

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

```
In[10]:= Map[equal[V, #] &,
          SubstTest[class, t, or[member[z, u], not[member[t, v]], not[subclass[v, w]]],
           {u → fix[domain[x]], v → binhom[x, y], w → domain[eval[z]]}]]

Out[10]= or[equal[0, binhom[x, y]], member[z, fix[domain[x]]],
           not[subclass[binhom[x, y], domain[eval[z]]]]] == True

In[11]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Corollary. Main result.

```
In[12]:= equiv[subclass[binhom[x, y], domain[eval[z]]],
          or[empty[binhom[x, y]], member[z, fix[domain[x]]]]]

Out[12]= True

In[13]:= subclass[binhom[x_, y_], domain[eval[z_]]] :=
          or[equal[0, binhom[x, y]], member[z, fix[domain[x]]]]
```