strict numerical dominance

Johan G. F. Belinfante
2004 April 7

```
In[1]:= << goedel56.06a; << tools.m;

        :Package Title: goedel56.06a          2004 April 6 at 9:15 p.m.

        It is now:  2004 Apr 7 at 12:16

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 April 6

        weightlimit = 40
```

summary

A set **x** is **strictly numerically dominated** by a set **y** if there is a bijection from **x** to a subset of **y**, but not in the other direction. For brevity it will be said that **x** is **smaller** than **y**. The following wrapped membership rule for the strict numerical dominance relation has been added to the **GOEDEL** program.

```
In[2]:= Begin["Goedel`Private`"];

In[3]:= InfoMatch[class[w_, HoldPattern[member[z_, SMALLER]]]] // First

Out[3]= {class[w_, member[x_, SMALLER]] := Module[{u = Unique[], v = Unique[], y = Unique[],
           z = Unique[]}, class[w, exists[u, v, and[equal[pair[u, v], x], exists[y,
           and[ONEONE[y], equal[u, domain[y]], subclass[range[y], v]]], not[exists[
           z, and[ONEONE[z], equal[v, domain[z]], subclass[range[z], u]]]]]]]]}
```

Some properties of this relation are derived in this notebook.

normalization

```
In[4]:= SMALLER // Normality // Reverse

Out[4]= intersection[complement[composite[Q, inverse[S]]], composite[Q, S]] == SMALLER

In[5]:= intersection[complement[composite[Q, inverse[S]]], composite[Q, S]] := SMALLER

In[6]:= intersection[composite[Q, inverse[S]], complement[composite[Q, S]]] // DoubleInverse

Out[6]= intersection[complement[composite[Q, S]], composite[Q, inverse[S]]] == inverse[SMALLER]

In[7]:= intersection[complement[composite[Q, S]], composite[Q, inverse[S]]] := inverse[SMALLER]
```

```
In[8]:= SubstTest[subclass, intersection[x, y], x,
          {x -> composite[Q, S], y -> complement[composite[Q, inverse[S]]]}]

Out[8]= subclass[SMALLER, composite[Q, S]] == True

In[9]:= subclass[SMALLER , composite[Q, S]] := True

In[10]:= SubstTest[subclass, intersection[x, y], y,
           {x -> composite[Q, S], y -> complement[composite[Q, inverse[S]]]}]

Out[10]= equal[0, intersection[SMALLER, composite[Q, inverse[S]]]] == True

In[11]:= intersection[SMALLER , composite[Q, inverse[S]]] := 0

In[12]:= SubstTest[subclass, intersection[composite[Q, S], y], cart[V, V],
          y -> complement[composite[Q, inverse[S]]]]

Out[12]= subclass[SMALLER, cart[V, V]] == True

In[13]:= subclass[SMALLER, cart[V, V]] := True

In[14]:= equal[composite[Id, SMALLER], SMALLER]

Out[14]= True

In[15]:= composite[Id, SMALLER] := SMALLER
```

consequences of the Schröder−Bernstein theorem

A consequence of the Schröder−Bernstein theorem is the following simpler formula for **SMALLER**:

```
In[16]:= SubstTest[dif, x, intersection[x, y],
           {x → composite[Q, S], y → composite[Q, inverse[S]]}]

Out[16]= intersection[complement[Q], composite[Q, S]] == SMALLER

In[17]:= intersection[complement[Q], composite[Q, S]] := SMALLER

In[18]:= intersection[complement[Q], composite[Q, inverse[S]]] // DoubleInverse

Out[18]= intersection[complement[Q], composite[Q, inverse[S]]] == inverse[SMALLER]

In[19]:= intersection[complement[Q], composite[Q, inverse[S]]] := inverse[SMALLER]
```

Other consequences:

```
In[20]:= AssInt[SMALLER, composite[Q, inverse[S]], composite[Q, S]]

Out[20]= intersection[Q, SMALLER] == 0

In[21]:= intersection[Q, SMALLER] := 0

In[22]:= SubstTest[union, intersection[x, y], dif[x, y],
           {x -> composite[Q, S], y -> composite[Q, inverse[S]]}]

Out[22]= union[Q, SMALLER] == composite[Q, S]
```

```
In[23]:= union[Q, SMALLER] := composite[Q, S]

In[24]:= intersection[Q, inverse[SMALLER]] // DoubleInverse

Out[24]= intersection[Q, inverse[SMALLER]] == 0

In[25]:= intersection[Q, inverse[SMALLER]] := 0

In[26]:= SubstTest[intersection, dif[x, y], dif[y, x],
          {x -> composite[Q, S], y -> composite[Q, inverse[S]]}]

Out[26]= intersection[SMALLER, inverse[SMALLER]] == 0

In[27]:= intersection[SMALLER, inverse[SMALLER]] := 0
```

transitive property

```
In[28]:= SubstTest[implies, TRANSITIVE[x], TRANSITIVE[dif[x, inverse[x]]], x -> composite[Q, S]]

Out[28]= TRANSITIVE[SMALLER] == True

In[29]:= TRANSITIVE[SMALLER] := True
```

a consequence of Cantor's theorem

A consequence of Cantor's theorem:

```
In[30]:= SubstTest[subclass, POWER, intersection[x, y],
              {x -> composite[Q, S], y -> complement[composite[Q, inverse[S]]]}]

Out[30]= subclass[POWER, SMALLER] == True

In[31]:= subclass[POWER, SMALLER] := True
```

domain and range, etc.

```
In[32]:= SubstTest[fix, intersection[x, y],
              {x -> composite[Q, S], y -> complement[composite[Q, inverse[S]]]}]

Out[32]= fix[SMALLER] == 0

In[33]:= fix[SMALLER] := 0

In[34]:= SubstTest[image, intersection[x, y], singleton[0],
              {x -> composite[Q, S], y -> complement[composite[Q, inverse[S]]]}]

Out[34]= image[SMALLER, singleton[0]] == complement[singleton[0]]

In[35]:= image[SMALLER, singleton[0]] := complement[singleton[0]]
```

```
In[36]:= SubstTest[image, inverse[intersection[x, y]], singleton[0],
             {x -> composite[Q, S], y -> complement[composite[Q, inverse[S]]]}]

Out[36]= image[inverse[SMALLER], singleton[0]] == 0

In[37]:= image[inverse[SMALLER], singleton[0]] := 0

In[38]:= Map[assert, SubstTest[implies, subclass[u, v],
           subclass[domain[u], domain[v]], {u -> POWER, v -> SMALLER}]]

Out[38]= equal[V, domain[SMALLER]] == True

In[39]:= domain[SMALLER] := V

In[40]:= SubstTest[subclass, image[x, y], range[x], {x -> SMALLER, y -> singleton[0]}]

Out[40]= equal[V, union[range[SMALLER], singleton[0]]] == True

In[41]:= % /. Equal -> SetDelayed

In[42]:= member[0, range[SMALLER]] // AssertTest

Out[42]= member[0, range[SMALLER]] == False

In[43]:= % /. Equal -> SetDelayed

In[44]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> range[SMALLER], v -> complement[singleton[0]]}]

Out[44]= True == equal[complement[singleton[0]], range[SMALLER]]

In[45]:= range[SMALLER] := complement[singleton[0]]
```

Q and SMALLER

```
In[46]:= SubstTest[implies, subclass[u, v],
           subclass[composite[u, w], composite[v, w]], {u -> Id, v -> Q, w -> SMALLER}]

Out[46]= subclass[SMALLER, composite[Q, SMALLER]] == True

In[47]:= % /. Equal -> SetDelayed

In[48]:= SubstTest[implies, subclass[u, v],
           subclass[composite[w, u], composite[w, v]], {u -> Id, v -> Q, w -> SMALLER}]

Out[48]= subclass[SMALLER, composite[SMALLER, Q]] == True

In[49]:= % /. Equal -> SetDelayed

In[50]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
             {u -> SMALLER, v -> composite[Q, S], w -> Q}]

Out[50]= subclass[composite[SMALLER, Q], composite[Q, S]] == True

In[51]:= % /. Equal -> SetDelayed
```

```
In[52]:= SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
           {u -> SMALLER, v -> composite[Q, S], w -> Q}]

Out[52]= subclass[composite[Q, SMALLER], composite[Q, S]] == True

In[53]:= % /. Equal -> SetDelayed

In[54]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
           {u -> SMALLER, v -> complement[composite[Q, inverse[S]]], w -> Q}]

Out[54]= subclass[composite[SMALLER, Q],
           composite[complement[composite[Q, inverse[S]]], Q]] == True

In[55]:= % /. Equal -> SetDelayed

In[56]:= SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
           {u -> SMALLER, v -> complement[composite[Q, inverse[S]]], w -> Q}]

Out[56]= subclass[composite[Q, SMALLER],
           composite[Q, complement[composite[Q, inverse[S]]]]] == True

In[57]:= % /. Equal -> SetDelayed

In[58]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u -> composite[SMALLER, Q],
           v -> composite[complement[composite[Q, inverse[S]]], Q],
                w -> complement[composite[Q, inverse[S]]]}]

Out[58]= equal[0, intersection[composite[Q, inverse[S]], composite[SMALLER, Q]]] == True

In[59]:= intersection[composite[Q, inverse[S]], composite[SMALLER, Q]] := 0

In[60]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u -> composite[Q, SMALLER],
           v -> composite[Q, complement[composite[Q, inverse[S]]]],
                w -> complement[composite[Q, inverse[S]]]}]

Out[60]= equal[0, intersection[composite[Q, SMALLER], composite[Q, inverse[S]]]] == True

In[61]:= intersection[composite[Q, SMALLER], composite[Q, inverse[S]]] := 0

In[62]:= SubstTest[subclass, u, intersection[v, w],
           {u -> composite[Q, SMALLER],
           v -> composite[Q, S], w -> complement[composite[Q, inverse[S]]]}]

Out[62]= subclass[composite[Q, SMALLER], SMALLER] == True

In[63]:= % /. Equal -> SetDelayed

In[64]:= SubstTest[subclass, u, intersection[v, w],
           {u -> composite[SMALLER, Q],
           v -> composite[Q, S], w -> complement[composite[Q, inverse[S]]]}]

Out[64]= subclass[composite[SMALLER, Q], SMALLER] == True

In[65]:= % /. Equal -> SetDelayed

In[66]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> composite[Q, SMALLER], v -> SMALLER}]

Out[66]= True == equal[SMALLER, composite[Q, SMALLER]]

In[67]:= composite[Q, SMALLER] := SMALLER
```

```
In[68]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> composite[SMALLER, Q], v -> SMALLER}]

Out[68]= True == equal[SMALLER, composite[SMALLER, Q]]

In[69]:= composite[SMALLER, Q] := SMALLER
```

SMALLER and S

```
In[70]:= SubstTest[implies, subclass[u, v], subclass[composite[S, u], composite[S, v]],
             {u -> Id, v -> Q}]

Out[70]= subclass[S, composite[Q, S]] == True

In[71]:= subclass[S, composite[Q, S]] := True

In[72]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u -> S, v -> composite[Q, S], w -> id[composite[Q, inverse[S]]]}]

Out[72]= subclass[intersection[S, composite[Q, inverse[S]]], Q] == True

In[73]:= % /. Equal -> SetDelayed

In[74]:= SubstTest[subclass, intersection[S, composite[Q, inverse[S]]], intersection[x, y],
             {x -> composite[Q, S], y -> composite[Q, inverse[S]]}] // Reverse

Out[74]= subclass[intersection[S, composite[Q, inverse[S]]], composite[Q, S]] == True

In[75]:= % /. Equal -> SetDelayed
```

A temporary rewrite rule:

```
In[76]:= equal[union[SMALLER, composite[SMALLER, SMALLER]], SMALLER]

Out[76]= True

In[77]:= union[SMALLER, composite[SMALLER, SMALLER]] := SMALLER

In[78]:= SubstTest[implies, and[subclass[u, v], subclass[x, y]],
           subclass[composite[u, x], composite[v, y]],
             {u -> SMALLER, x -> S, v -> SMALLER, y -> union[z, SMALLER]}] /. z -> Q

Out[78]= subclass[composite[SMALLER, S], SMALLER] == True

In[79]:= % /. Equal -> SetDelayed

In[80]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> composite[SMALLER, S], v -> SMALLER}]

Out[80]= True == equal[SMALLER, composite[SMALLER, S]]

In[81]:= composite[SMALLER, S] := SMALLER

In[82]:= SubstTest[implies, subclass[x, y], subclass[composite[x, z], composite[y, z]],
             {x -> S, y -> union[w, SMALLER], z -> SMALLER}] /. w -> Q

Out[82]= subclass[composite[S, SMALLER], SMALLER] == True

In[83]:= % /. Equal -> SetDelayed
```

```
In[84]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> composite[S, SMALLER], v -> SMALLER}]

Out[84]= True == equal[SMALLER, composite[S, SMALLER]]

In[85]:= composite[S, SMALLER] := SMALLER
```