the class TO of total orderings

Johan G. F. Belinfante
2004 February 10

```
In[1]:= << goedel54.10b; << tools.m

        :Package Title: goedel54.10b      2004 February 10 at 10:10 p.m.

        It is now:  2004 Feb 11 at 13:12

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 January 3

        weightlimit = 40
```

summary

Because the definition of the predicate **TOTALORDER** has been wrapped, the membership rule for the class **TO** of total orderings now looks like this:

```
In[2]:= member[x, TO]

Out[2]= and[member[x, V], TOTALORDER[x]]
```

Basic facts about the class **TO** of all total orderings are derived in this notebook, including two new formulas for this class, either one of which could serve as a suitable starting point for automated reasoning about this class using a program such as **Otter**.

relation of the class TO to other named classes

The class **TO** is contained in the class **PO**, which in turn is the intersection of other named classes. In particular, total orderings are reflexive:

```
In[3]:= subclass[TO, RFX] // AssertTest

Out[3]= subclass[TO, RFX] == True

In[4]:= subclass[TO, RFX] := True
```

The class of total orderings is contained in the class of partial orderings:

```
In[5]:= Map[subclass[#, PO] &, TO // Normality]

Out[5]= subclass[TO, PO] == True

In[6]:= subclass[TO, PO] := True
```

Corollary: **TO** is contained in the class of antisymmetric relations.

```
In[7]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> TO, v -> PO, w -> ANTISYM}]

Out[7]= subclass[TO, ANTISYM] == True

In[8]:= subclass[TO, ANTISYM] := True
```

Another corollary: the class **TO** is contained in the class of transitive relations.

```
In[9]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u -> TO, v -> PO, w -> TRV}]

Out[9]= subclass[TO, TRV] == True

In[10]:= subclass[TO, TRV] := True
```

formula for U[TO]

The following temporary abbreviation is used for the reflexive closure of the singleton of a pair:

```
In[11]:= el[x_, y_] := union[cart[singleton[x], singleton[y]], id[pairset[x, y]]]
```

This is a total ordering:

```
In[12]:= member[el[x, y], TO]

Out[12]= True
```

Since any pair belongs to a total ordering, one derives:

```
In[13]:= Map[implies[member[y, V], #] &,
          SubstTest[implies, and[member[u, v], member[v, w]], member[u, U[w]],
            {u -> pair[x, y], v -> el[x, y], w -> TO}]]

Out[13]= or[member[pair[x, y], U[TO]], not[member[x, V]], not[member[y, V]]] == True

In[14]:= or[member[pair[x_, y_], U[TO]], not[member[x_, V]], not[member[y_, V]]] := True
```

Removing the variables yields a lower bound for **U[TO]**.

```
In[15]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[x, y], or[member[pair[x, y], z],
             not[member[x, V]], not[member[y, V]]], z -> U[TO]]] // Reverse

Out[15]= subclass[cart[V, V], U[TO]] == True

In[16]:= % /. Equal -> SetDelayed
```

That this is also an upper bound is easily established:

```
In[17]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
          subclass[u, w], {u -> TO, v -> RFX, w -> P[cart[V, V]]}]

Out[17]= subclass[U[TO], cart[V, V]] == True
```

```
In[18]:= % /. Equal -> SetDelayed
```

These two inclusions can be combined into an equation:

```
In[19]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> cart[V, V], v -> U[TO]}]

Out[19]= True == equal[cart[V, V], U[TO]]

In[20]:= U[TO] := cart[V, V]
```

other properties of the class TO

Note that:

```
In[21]:= equal[intersection[TO, P[cart[V, V]]], TO]

Out[21]= True
```

This justifies the following rewrite rule:

```
In[22]:= intersection[TO, P[cart[V, V]]] := TO
```

Corollary.

```
In[23]:= ImageComp[IMAGE[id[cart[V, V]]], id[P[cart[V, V]]], TO] // Reverse

Out[23]= image[IMAGE[id[cart[V, V]]], TO] == TO

In[24]:= image[IMAGE[id[cart[V, V]]], TO] := TO
```

Lemma.

```
In[25]:= implies[member[x, TO], member[inverse[x], TO]] // NotNotTest

Out[25]= or[and[member[domain[x], V], member[range[x], V], TOTALORDER[inverse[x]]],
           not[member[x, V]], not[TOTALORDER[x]]] == True

In[26]:= (% /. x -> x_) /. Equal -> SetDelayed

In[27]:= Map[equal[V, #] &,
           SubstTest[class, x, implies[member[x, y], member[inverse[x], y]], y -> TO]] // Reverse

Out[27]= subclass[image[IMAGE[SWAP], TO], TO] == True

In[28]:= % /. Equal -> SetDelayed
```

The opposite inclusion also holds:

```
In[29]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u -> image[IMAGE[SWAP], TO], v -> TO, w -> IMAGE[SWAP]}]

Out[29]= subclass[TO, image[IMAGE[SWAP], TO]] == True

In[30]:= % /. Equal -> SetDelayed
```

These can be combined into an equation:

```
In[31]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> image[IMAGE[SWAP], TO], v -> TO}]

Out[31]= True == equal[TO, image[IMAGE[SWAP], TO]]

In[32]:= image[IMAGE[SWAP], TO] := TO
```

Corollary.

```
In[33]:= ImageComp[IMAGE[SWAP], id[P[cart[V, V]]], TO]

Out[33]= image[INVERSE, TO] == TO

In[34]:= image[INVERSE, TO] := TO
```

two formulas for the class of total orderings

The fact that any two members of **fix[x]** are comparable for a total ordering **x** can be viewed as the statement that **fix[x]** is a clique in the symmetric hull of **x**. This leads to a simple formula for **TO**, but the derivation of it seems quite tricky.

```
In[35]:= composite[FIRST, id[INVERSE], inverse[CUP]] // DoubleInverse

Out[35]= composite[FIRST, id[INVERSE], inverse[CUP]] == inverse[HULL[SYM]]

In[36]:= composite[FIRST, id[INVERSE], inverse[CUP]] := inverse[HULL[SYM]]
```

A formula for the class of reflexive relations can be derived by using a sequestering trick:

```
In[37]:= (fix[composite[FIRST, id[x], inverse[CUP], inverse[S],
              CART, DUP, IMAGE[inverse[DUP]]]] // Renormality) /. x -> INVERSE

Out[37]= fix[composite[inverse[HULL[SYM]], inverse[S], CART, DUP, IMAGE[inverse[DUP]]]] == RFX

In[38]:= fix[composite[inverse[HULL[SYM]], inverse[S], CART, DUP, IMAGE[inverse[DUP]]]] := RFX
```

A temporary normalization rule is added:

```
In[39]:= (TO // Normality // Reverse) /. Equal -> SetDelayed
```

This temporary rule yields the first of two formulas for the class of total orderings:

```
In[40]:= (Map[intersection[PO, #] &,
            (fix[composite[FIRST, id[funpart[x]], inverse[CUP], S, CART, DUP,
                 IMAGE[inverse[DUP]]]] // Renormality) /. x -> INVERSE]) // InvertFix

Out[40]= intersection[PO,
           fix[composite[inverse[IMAGE[inverse[DUP]]], inverse[E], CLIQUES, HULL[SYM]]]] == TO

In[41]:= intersection[PO,
           fix[composite[inverse[IMAGE[inverse[DUP]]], inverse[E], CLIQUES, HULL[SYM]]]] := TO
```

Lemma. A temporary rewrite rule.

```
In[42]:= fix[composite[inverse[x], S, CART, DUP, y]] // InvertFixTest

Out[42]= fix[composite[inverse[x], S, CART, DUP, y]] ==
          fix[composite[inverse[y], inverse[E], CLIQUES, x]]

In[43]:= fix[composite[inverse[x_], S, CART, DUP, y_]] :=
          fix[composite[inverse[y], inverse[E], CLIQUES, x]]
```

Lemma.

```
In[44]:= SubstTest[intersection, fix[composite[inverse[funpart[x]], S, funpart[y]]],
            fix[composite[inverse[funpart[x]], inverse[S], funpart[y]]],
          {x -> composite[CUP, id[INVERSE], inverse[FIRST]],
                y -> composite[CART, DUP, IMAGE[inverse[DUP]]]}]

Out[44]= intersection[RFX,
           fix[composite[inverse[IMAGE[inverse[DUP]]], inverse[E], CLIQUES, HULL[SYM]]]] ==
          fix[composite[inverse[HULL[SYM]], CART, DUP, IMAGE[inverse[DUP]]]]

In[45]:= intersection[RFX,
           fix[composite[inverse[IMAGE[inverse[DUP]]], inverse[E], CLIQUES, HULL[SYM]]]] :=
          fix[composite[inverse[HULL[SYM]], CART, DUP, IMAGE[inverse[DUP]]]]
```

Lemma.

```
In[46]:= equal[intersection[RFX, TO], TO]

Out[46]= True

In[47]:= intersection[RFX, TO] := TO
```

A second formula for the class of total orderings:

```
In[48]:= AssInt[PO, RFX,
          fix[composite[inverse[IMAGE[inverse[DUP]]], inverse[E], CLIQUES, HULL[SYM]]]]

Out[48]= intersection[PO,
           fix[composite[inverse[HULL[SYM]], CART, DUP, IMAGE[inverse[DUP]]]]] == TO

In[49]:= intersection[PO,
           fix[composite[inverse[HULL[SYM]], CART, DUP, IMAGE[inverse[DUP]]]]] := TO
```