monotonicity of domain[UB[x]]

Johan G. F. Belinfante
2008 July 6

```
In[1]:= SetDirectory["l:"]; << goedel.08jul05a; << tools.m

        :Package Title: goedel.08jul05a                 2008 July 5 at 10:45 p.m.

        It is now:  2008 Jul 6 at 21:48

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 July 5

        weightlimit = 40
```

summary

In this notebook a variable-free formulation of the following monotonicity property of **domain[UB[x]]** is derived.

```
In[2]:= implies[subclass[x, y], subclass[domain[UB[x]], domain[UB[y]]]]

Out[2]= True
```

The same result is derived two ways. The first and simplest method is to use the **APPLY** formulas for **UBD = lambda[x, domain[UB[x]]]** to eliminate the variables **x** and **y** using **class** rules (that is, using the Bernays-Gödel algorithm.). The second method uses **reify** rules in place of the **class** rules to eliminate the variables. For this approach, one needs to use the fact that any statement **p[x,y]** about **x** and **y** in Gödel's class theory can be reformulated as an equation of the form **equal[V, c[x,y]]** for a suitable class **c[x,y]**. Since any two sets **x** and **y** can be combined into a single ordered pair **z = PAIR[x, y]**, one can eliminate both variables at once by reifying the class **c[first[z], second[z]]**.

derivation

Obervation. The monotonicity can be formulated in terms of the function **UBD** as follows:

```
In[3]:= implies[subclass[x, y], subclass[APPLY[UBD, x], APPLY[UBD, y]]]

Out[3]= True
```

Theorem. (Monotonicity property.)

```
In[4]:= Map[subclass[S, #] &, SubstTest[class, pair[x, y],
          implies[subclass[x, y], subclass[APPLY[t, x], APPLY[t, y]]], t → UBD]]

Out[4]= subclass[S, composite[inverse[UBD], S, UBD]] == True
```

alternate derivation

In this section, an alternate derivation of the variable-free statement of monotonicity is presented, using **reify** rules in place of **class** rules. One can only apply **reify** to class expressions, and not to statements, but this poses no real limitation because any statement can be turned into an equation of the form that some class is equal to **V**, and one can then apply **reify** to that class. In the present case, the following observation holds the key to converting the original monotonicity statement into an equation involving a class to which **reify** can be applied. For the monotonicity statement the class in question has the following form:

```
In[5]:= class[t, implies[subclass[x, y], subclass[u, v]]]

Out[5]= union[complement[image[V, intersection[u, complement[v]]]],
         image[V, intersection[x, complement[y]]]]
```

The equation that states that this class is equal to **V** is logically equivalent to the statement **implies[subclass[x,y], subclass[u,v]]**.

```
In[6]:= equal[V, union[complement[image[V, intersection[u, complement[v]]]],
          image[V, intersection[x, complement[y]]]]]

Out[6]= or[not[subclass[x, y]], subclass[u, v]]
```

The equation **V = c[x,y]** need not be stated explicitly, but can be used implicitly in a rewrite rule that rewrites **c[x, y]** to **V**. In the present case, when one replaces the variables **u** and **v** in the class expression considered above respectively by the expressions **domain[UB[x]]** and **domain[UB[y]]**, the **GOEDEL** program automatically uses the monotonicity of **domain[UB[x]]** to rewrite this expression to **V**.

```
In[7]:= union[complement[image[V, intersection[u, complement[v]]]],
          image[V, intersection[x, complement[y]]]] /. {u → domain[UB[x]], v → domain[UB[y]]}

Out[7]= V
```

To eliminate the variables **x** and **y** in the monotonicity statement one now uses **SubstTest** to compare the results obtained by reifying a corresponding expression in which **image[V, -]** has been replaced with **image[w, -]** and then restoring the dummy **w** to **V** with the result obtained by first replacing **w** with **V** and then applying **reify**. One does need to introduce a new rewrite rule to get **UBD** into play:

Lemma.

```
In[8]:= SubstTest[composite, inverse[E], VERTSECT[t],
         t -> composite[FIRST, complement[composite[cross[E, Id], complement[inverse[E]]]]]]

Out[8]= composite[FIRST, complement[composite[cross[E, Id], complement[inverse[E]]]]] ==
         composite[inverse[E], UBD]

In[9]:= composite[FIRST, complement[composite[cross[E, Id], complement[inverse[E]]]]] :=
         composite[inverse[E], UBD]
```

Theorem. Alternate derivation of the monotonicity property of **UBD** using **reify**. (Note the use of **first** and **second** to eliminate both variables at the same time.)

```
In[10]:= Map[subclass[S, #] &, SubstTest[reify, x, union[complement[
             image[w, intersection[complement[domain[UB[second[x]]]], domain[UB[first[x]]]]],
           image[w, intersection[complement[second[x]], first[x]]], w → V]]
```

```
Out[10]= subclass[S, composite[inverse[UBD], S, UBD]] == True
```

```
In[11]:= subclass[S, composite[inverse[UBD], S, UBD]] := True
```

corollaries

In this section, two closely related statements are derived. The first one states that **UBD** subcommutes with **S**:

```
In[12]:= SubstTest[implies, subclass[u, v], subclass[composite[t, u], composite[t, v]],
           {t → UBD, u → S, v -> composite[inverse[UBD], S, UBD]}] // Reverse
```

```
Out[12]= subclass[composite[UBD, S], composite[S, UBD]] == True
```

```
In[13]:= subclass[composite[UBD, S], composite[S, UBD]] := True
```

Corollary.

```
In[14]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
           {u → composite[UBD, S], v -> composite[S, UBD], w → inverse[UBD]}] // Reverse
```

```
Out[14]= subclass[composite[UBD, S, inverse[UBD]], S] == True
```

```
In[15]:= subclass[composite[UBD, S, inverse[UBD]], S] := True
```