monotone functions

Johan G. F. Belinfante
2010 October 9

```
In[1]:= SetDirectory["l:"]; << goedel.10oct08a

        :Package Title: goedel.10oct08a            2010 October 8 at 9:55 p.m.

        It is now:  2010 Oct 9 at 23:36

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

The concept of monotonicity is not limited to functions, but the most common applications involve functions and inverse of functions. If a function is monotone from one relation **x** to another relation **y**, then for any pair of elements in its domain that are related by **x**, the corresponding values of the function are related by **y**. A similar statement applies to the situation when the inverse of a function is monotone from one relation to another.

introduction

Generally speaking, when dealing with monotone functions, simpler statements are obtained when function application is avoided entirely. Not only are there fewer variables in play, but one does not have to keep mentioning the domain of the function. Nonetheless, there are some occasions when it desirable to be able to reason with explicit arguments and values of functions. This is especially true when reading the mathematical literature because most authors appear to be more comfortable with statements involving function application. In this notebook, basic rewrite rules are derived that make it easy to pass from statements without variables for arguments of functions to statements with those variables present.

The statement that a function **funpart[w]** is monotone from a relation **x** to a relation **y** can be expressed without any need to introduce additional variables as follows:

```
In[2]:= subclass[P[funpart[w]], monotone[x, y]]

Out[2]= subclass[composite[funpart[w], x, inverse[funpart[w]]], y]
```

Note that the function **funpart[w]** itself need not be a set. Although the class **monotone[x, y]** only holds those monotone functions that are sets, one can nevertheless use this class to describe monotonicity for arbitrary functions. The reason that this is possible is because a function is monotone if and only if all its subsets are monotone. Thus, a function is monotone from **x** to **y** if and only if its power class is a subclass of the class **monotone[x, y]**.

The class **monotone[x, y]** can hold sets that are not functions. Its members are only required to be relations that satisfy the inclusion **w ∘ x ∘ inverse[w] ⊂ y**. Explicitly:

```
In[3]:= member[w, monotone[x, y]]

Out[3]= and[member[w, V], subclass[w, cart[V, V]], subclass[composite[w, x, inverse[w]], y]]
```

derivation

The basic idea of the derivation is to use the following rewrite rule about inverse images for functions.

```
In[4]:= subclass[intersection[x, domain[funpart[w]]], image[inverse[funpart[w]], y]]

Out[4]= subclass[image[funpart[w], x], y]
```

The above rewrite rule need not be valid when **funpart[w]** is replaced with an arbitrary relation **w**. An explicit counterexample is easy to find:

```
In[5]:= (equiv[subclass[intersection[x, domain[w]], image[inverse[w], y]],
           subclass[image[w, x], y]]) /. {x → set[0], y → set[0], w → cartsq[V]}

Out[5]= False
```

A corresponding rewrite rule involving function application is already available.

```
In[6]:= implies[and[member[t, x], member[t, domain[funpart[w]]],
          subclass[image[funpart[w], x], y]], member[APPLY[funpart[w], t], y]]

Out[6]= True
```

The following theorem provides the basic tool for introducing function application into statements about function application.

Theorem. If **funpart[w]** is monotone from **x** to **y**, and if elements **u** and **v** in its domain are related by **x**, then the corresponding values **APPLY[funpart[w], u]** and **APPLY[funpart[w], v]** are related by **y**.

```
In[7]:= SubstTest[implies, and[member[t, x], member[t, domain[funpart[v]]],
           subclass[image[funpart[v], x], y]], member[APPLY[funpart[v], t], y],
          {t → pair[u, v], v → cross[funpart[w], funpart[w]]}] // Reverse

Out[7]= or[member[pair[APPLY[funpart[w], u], APPLY[funpart[w], v]], y],
          not[member[u, domain[funpart[w]]]],
          not[member[v, domain[funpart[w]]]], not[member[pair[u, v], x]],
          not[subclass[composite[funpart[w], x, inverse[funpart[w]]], y]] == True

In[8]:= or[member[pair[APPLY[funpart[w_], u_], APPLY[funpart[w_], v_]], y_],
          not[member[pair[u_, v_], x_]], not[member[u_, domain[funpart[w_]]]],
          not[member[v_, domain[funpart[w_]]]],
          not[subclass[composite[funpart[w_], x_, inverse[funpart[w_]]], y_]]] := True
```

The following corollary is obtained when one replaces the relations **x** and **y** by the complement of **y** and the complement of **x**, respectively. Observe that in the monotonicity hypothesis, the function **funpart[w]** and its inverse are interchanged.

```
In[9]:= equiv[subclass[P[funpart[w]], monotone[x, y]],
          subclass[P[inverse[funpart[w]]], monotone[complement[y], complement[x]]]]

Out[9]= True
```

Corollary. If the relation **inverse[funpart[w]]** is monotone from **x** to **y**, and if the values **APPLY[funpart[w], u]** and **APPLY[funpart[w], v]** of the function **funpart[w]** at elements **u** and **v** in its domain are related by **x**, then the elements **u** and **v** are related by **y**.

```
In[10]:= SubstTest[implies, and[subclass[P[funpart[w]], monotone[s, t]],
             member[pair[u, v], intersection[s, cartsq[domain[funpart[w]]]]]],
           member[pair[APPLY[funpart[w], u], APPLY[funpart[w], v]], t],
           {s → complement[y], t → complement[x]}] // Reverse

Out[10]= or[member[pair[u, v], y],
           not[member[u, domain[funpart[w]]]], not[member[v, domain[funpart[w]]]],
           not[member[pair[APPLY[funpart[w], u], APPLY[funpart[w], v]], x]],
           not[subclass[composite[inverse[funpart[w]], x, funpart[w]], y]]] == True

In[11]:= or[member[pair[u_, v_], y_],
           not[member[pair[APPLY[funpart[w_], u_], APPLY[funpart[w_], v_]], x_]],
           not[member[u_, domain[funpart[w_]]]], not[member[v_, domain[funpart[w_]]]],
           not[subclass[composite[inverse[funpart[w_]], x_, funpart[w_]], y_]]] := True
```

restatement without funpart wrappers

The two theorems derived in the preceding section can be restated without **funpart** wrappers.

Theorem. If a function **w** is monotone from **x** to **y**, and if elements **u** and **v** in its domain are related by **x**, then the corresponding values **APPLY[w, u]** and **APPLY[w, v]** are related by **y**.

```
In[12]:= SubstTest[implies, equal[w, funpart[t]], or[member[pair[APPLY[w, u], APPLY[w, v]], y],
             not[member[u, domain[w]]], not[member[v, domain[w]]], not[member[pair[u, v], x]],
             not[subclass[composite[w, x, inverse[w]], y]]], t → w] // Reverse

Out[12]= or[member[pair[APPLY[w, u], APPLY[w, v]], y],
           not[FUNCTION[w]], not[member[u, domain[w]]], not[member[v, domain[w]]],
           not[member[pair[u, v], x]], not[subclass[composite[w, x, inverse[w]], y]]] == True

In[13]:= or[member[pair[APPLY[w_, u_], APPLY[w_, v_]], y_],
           not[FUNCTION[w_]], not[member[u_, domain[w_]]],
           not[member[v_, domain[w_]]], not[member[pair[u_, v_], x_]],
           not[subclass[composite[w_, x_, inverse[w_]], y_]]] := True
```

The corresponding statement for the theorem about monotone inverse functions was derived 2010 May 2, and is therefore already available in the **GOEDEL** program. This theorem says that if the inverse of a function **w** is monotone from **x** to

y, and if the values **APPLY[w, u]** and **APPLY[w, v]** of the function **w** at elements **u** and **v** in its domain are related by **x**, then the elements **u** and **v** are related by **y**.

```
In[14]:= implies[FUNCTION[w], or[member[pair[u, v], y], not[member[u, domain[w]]],
           not[member[v, domain[w]]], not[member[pair[APPLY[w, u], APPLY[w, v]], x]],
           not[subclass[composite[inverse[w], x, w], y]]]

Out[14]= True
```

Comment. In special circumstances one or both of the explicit hypotheses that **u** and **v** belong to **domain[w]** may become redundant. For example, if one adds the hypothesis that **x** is a relation, then both of these statements would follow from the hypothesis **pair[APPLY[w,u], APPLY[w,v]] ∈ x ⊂ V × V**. For the special case that both **x** and **y** are the subset relation **S**, one of these statements follows from the other:

```
In[15]:= implies[FUNCTION[w],
          or[subclass[u, v], not[member[v, domain[w]]], not[subclass[APPLY[w, u], APPLY[w, v]]],
           not[subclass[composite[inverse[w], S, w], S]]]

Out[15]= True
```