compatible collections of functions

Johan G. F. Belinfante
2010 October 1

```
In[1]:= SetDirectory["l:"]; << goedel.10sep30a

        :Package Title: goedel.10sep30a            2010 September 30 at 6:20 p.m.

        It is now:  2010 Oct 1 at 7:43

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

A collection of functions is **compatible** if their union is a function. In this notebook a new criterion for compatibility is derived. It will be shown that a collection of functions is compatible if and only if they are all restrictions of their union.

general results

The new criterion for compatibility can be most succinctly formulated in terms of the class **RS[x]** of restrictions of a class **x**. Recall that the membership rule for the class of restrictions is the following.

```
In[2]:= member[u, RS[x]]

Out[2]= and[equal[u, composite[x, id[domain[u]]]], member[u, V]]
```

An important fact needed in the sequel is that a class is a function if and only if every subset is a restriction. That is, a class is a function if and only if its class of restrictions is equal to its power class.

```
In[3]:= equal[P[x], RS[x]]

Out[3]= FUNCTION[x]
```

If the union of a class of sets is a function, then every member of the class is a function. This fact is also already available in the **GOEDEL** program:

```
In[4]:= implies[FUNCTION[U[x]], subclass[x, FUNS]]

Out[4]= True
```

The following general result will also be needed later.

Theorem. Every member of a class of functions is a function.

```
In[5]:= SubstTest[implies, and[member[x, y], subclass[y, z]], member[x, z], z → FUNS] // Reverse

Out[5]= or[FUNCTION[x], not[member[x, y]], not[subclass[y, FUNS]]] == True

In[6]:= or[FUNCTION[x_], not[member[x_, y_]], not[subclass[y_, FUNS]]] := True
```

derivation

Theorem. If the union of a collection of sets is a function, then every member of the collection is a restriction of their union.

```
In[7]:= SubstTest[implies, and[subclass[u, v], equal[v, w]],
          subclass[u, w], {u → x, v → P[U[x]], w → RS[U[x]]}] // Reverse

Out[7]= or[not[FUNCTION[U[x]]], subclass[x, RS[U[x]]]] == True

In[8]:= or[not[FUNCTION[U[x_]]], subclass[x_, RS[U[x_]]]] := True
```

To derive an implication in the reverse direction, a new variable **u** will be introduced, and later eliminated.

Lemma. If **u ∈ x** and **x ⊂ RS[U[x]]**, then **u** is a restriction of **U[x]**.

```
In[9]:= SubstTest[implies, and[member[u, x], subclass[x, v]],
          member[u, v], v → RS[U[x]]] // Reverse

Out[9]= or[equal[u, composite[U[x], id[domain[u]]]],
          not[member[u, x]], not[subclass[x, RS[U[x]]]]] == True

In[10]:= or[equal[u_, composite[U[x_], id[domain[u_]]]],
           not[member[u_, x_]], not[subclass[x_, RS[U[x_]]]]] := True
```

The **funpart** of any class **x** is its restriction the class of points where its vertical sections are singletons. A class is a function if it is its own funpart.

Lemma. If **x ⊂ FUNS** and **x ⊂ RS[U[x]]**, then every member of **RS[U[x]]** is a subset of **funpart[U[x]]**.

```
In[11]:= Map[not, SubstTest[and, implies[and[p0, p1], p3], implies[and[p0, p2], p4],
            implies[and[p3, p4], p5], not[implies[and[p0, p1, p2], p5]],
            {p0 → member[u, x], p1 → subclass[x, FUNS], p2 → subclass[x, RS[U[x]]],
             p3 → FUNCTION[u], p4 -> equal[u, composite[U[x], id[domain[u]]]],
             p5 → equal[u, composite[funpart[U[x]], id[domain[u]]]]}] // Reverse

Out[11]= or[not[member[u, x]], not[subclass[x, FUNS]],
           not[subclass[x, RS[U[x]]]], subclass[u, funpart[U[x]]]] == True

In[12]:= (% /. {u → u_, x → x_}) /. Equal → SetDelayed
```

Theorem. If every member of a class of functions is a restriction of their union, then their union is a function.

```
In[13]:= Map[equal[V, #] &,
           SubstTest[class, u, implies[and[member[u, x], subclass[x, y]], member[u, v]],
            {v → P[funpart[U[x]]], y → intersection[FUNS, RS[U[x]]]}]]
Out[13]= or[FUNCTION[U[x]], not[subclass[x, FUNS]], not[subclass[x, RS[U[x]]]]] == True

In[14]:= or[FUNCTION[U[x_]], not[subclass[x_, FUNS]], not[subclass[x_, RS[U[x_]]]]] := True
```

Corollary. A condition for a class of functions to be compatible.

```
In[15]:= equiv[and[subclass[x, FUNS], subclass[x, RS[U[x]]]], FUNCTION[U[x]]] // not // not
Out[15]= True

In[16]:= and[subclass[x_, FUNS], subclass[x_, RS[U[x_]]]] := FUNCTION[U[x]]
```

---

serendipity

The following simple fact was encountered in the course of this study, but was not actually needed.

Theorem.

```
In[17]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → P[x], v → RS[x]}] // Reverse
Out[17]= subclass[P[x], RS[x]] == FUNCTION[x]

In[18]:= subclass[P[x_], RS[x_]] := FUNCTION[x]
```