cross-sections as maximal elements

Johan G. F. Belinfante
2007 November 11

```
In[1]:= SetDirectory["l:"]; << goedel99.10a; << tools.m

        :Package Title: goedel99.10a         2007 November 10 at 5:10 p.m.

        It is now:  2007 Nov 11 at 4:30

        Loading Simplification Rules

        TOOLS.M                                        Revised 2007 September 19

        weightlimit = 40
```

summary

The class **intersection[FUNS, P[x]]** of functions contained in a given class **x** is investigated in this notebook. This class is closed under unions of chains. Cross-sections of **x** are the maximal elements of **intersection[FUNS, P[x]]**.

Uchains[intersection[FUNS, P[x]]]

Theorem.

```
In[2]:= SubstTest[implies, and[equal[Uchains[u], u], equal[Uchains[v], v]],
         equal[Uchains[intersection[u, v]], intersection[u, v]],
         {u → FUNS, v → P[x]}]

Out[2]= True == equal[intersection[FUNS, P[x]], Uchains[intersection[FUNS, P[x]]]]

In[3]:= Uchains[intersection[FUNS, P[x_]]] := intersection[FUNS, P[x]]
```

cross-sections as maximal elements

Lemma.

```
In[4]:= Map[or[subclass[image[inverse[PS], intersection[FUNS, P[x]]], FUNS], #] &,
          SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
           {u → inverse[PS], v → inverse[S], w -> intersection[FUNS, P[x]]}]] // Reverse

Out[4]= subclass[image[inverse[PS], intersection[FUNS, P[x]]], FUNS] == True

In[5]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[6]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
          {u → inverse[PS], v → inverse[S], w -> intersection[FUNS, P[x]]}] // Reverse

Out[6]= subclass[U[image[inverse[PS], intersection[FUNS, P[x]]]], x] == True

In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[8]:= Map[not, SubstTest[and, implies[p4, p5], implies[and[p1, p2, p3, p5], p6],
           not[implies[and[p1, p2, p3, p4], p6]], {p1 → equal[domain[u], domain[x]],
            p2 → FUNCTION[v], p3 → subclass[u, v], p4 → subclass[v, x],
            p5 → subclass[domain[v], domain[x]], p6 → equal[u, v]}]] // Reverse

Out[8]= or[equal[u, v], not[equal[domain[u], domain[x]]],
          not[FUNCTION[v]], not[subclass[u, v]], not[subclass[v, x]]] == True

In[9]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

Theorem.

```
In[10]:= Map[empty[composite[Id, complement[#]]] &,
          SubstTest[class, pair[u, v], implies[and[subclass[u, v], not[equal[u, v]],
             subclass[v, x], FUNCTION[v]], not[member[u, w]]], w -> X[x]]]

Out[10]= equal[0, intersection[FUNS, image[PS, X[x]], P[x]]] == True

In[11]:= intersection[FUNS, image[PS, X[x_]], P[x_]] := 0
```

Restatement: Non-maximal members of **intersection[FUNS, P[x]]** cannot be cross-sections.

```
In[12]:= subclass[image[inverse[PS], intersection[FUNS, P[x]]],
          intersection[FUNS, complement[X[x]], P[x]]]

Out[12]= True
```

An inclusion in the opposite direction will be derived below. We will show that if a function **t** contained in **x** has a domain different from (smaller than) **x** then **t** is not a maximal member of the class of functions contained in **x**. The idea is that if **t** is such a function, one can simply add another point **p** to get a larger function contained in **x**. The following lemma captures this strategy.

```
In[13]:= SubstTest[implies,
           and[member[pair[u, t], composite[Id, w]], member[u, s]], member[t, image[w, s]],
           {w → inverse[PS], s -> intersection[FUNS, P[x]], u → union[t, set[p]]}] // Reverse

Out[13]= or[member[p, t], member[t, image[inverse[PS], intersection[FUNS, P[x]]]],
           not[FUNCTION[union[t, set[p]]]], not[member[p, x]],
           not[member[t, V]], not[subclass[t, x]]] == True

In[14]:= (% /. {p → p_, t → t_, x → x_}) /. Equal → SetDelayed
```

Theorem. (The **FUNCTION** hypothesis in the lemma is replaced by a more convenient one.)

```
In[15]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4],
            implies[and[p0, p4, p5, p6], p7], implies[and[p2, p3], p6],
            not[implies[and[p0, p1, p2, p3, p5], p7]], {p0 → member[t, P[x]], p1 → FUNCTION[t],
             p2 → member[first[p], V], p3 → not[member[first[p], domain[t]]],
             p4 → FUNCTION[union[t, set[p]]], p5 → member[p, x], p6 → not[member[p, t]],
             p7 -> member[t, image[inverse[PS], intersection[FUNS, P[x]]]]}] // Reverse

Out[15]= or[member[t, image[inverse[PS], intersection[FUNS, P[x]]]],
           member[first[p], domain[t]], not[FUNCTION[t]], not[member[p, x]],
           not[member[t, V]], not[member[first[p], V]], not[subclass[t, x]]] == True

In[16]:= (% /. {p → p_, t → t_, x → x_}) /. Equal → SetDelayed
```

Eliminating the variables **p** and **t** yields the following result.

```
In[17]:= Map[empty[domain[complement[#]]] &, SubstTest[class, pair[t, p],
           or[member[t, z], member[first[p], domain[t]], not[FUNCTION[t]], not[member[p, x]],
            not[member[t, V]], not[member[first[p], V]], not[subclass[t, x]]],
           z -> image[inverse[PS], intersection[FUNS, P[x]]]]]

Out[17]= subclass[intersection[FUNS, P[x]], union[image[inverse[PS], intersection[FUNS, P[x]]],
            image[inverse[IMAGE[FIRST]], image[S, set[domain[x]]]]]] == True

In[18]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. (Technical lemma needed to simplify the next result, obtained using **AssertTest**.)

```
In[19]:= Map[not[empty[#]] &,
          symdif[intersection[image[inverse[IMAGE[FIRST]], image[S, set[domain[x]]]], P[x]],
            intersection[image[inverse[IMAGE[FIRST]], set[domain[x]]], P[x]]] // Renormality]

Out[19]= member[domain[x], image[inverse[PS], image[IMAGE[FIRST], P[x]]]] == False

In[20]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. (There are no subsets of **x** whose domain exceeds that of **x**.)

```
In[21]:= equal[intersection[P[x], image[inverse[IMAGE[FIRST]], image[S, set[domain[x]]]]],
            intersection[P[x], image[inverse[IMAGE[FIRST]], set[domain[x]]]] // AssertTest

Out[21]= equal[intersection[image[inverse[IMAGE[FIRST]], image[S, set[domain[x]]], P[x]],
            intersection[image[inverse[IMAGE[FIRST]], set[domain[x]]], P[x]]] == True

In[22]:= intersection[image[inverse[IMAGE[FIRST]], image[S, set[domain[x_]]]], P[x_]] :=
          intersection[image[inverse[IMAGE[FIRST]], set[domain[x]]], P[x]]
```

The preceding rewrite rule is now used to simplify the result obtained when the variables **p** and **t** were eliminated. This provides the desired inclusion in the reverse direction.

```
In[23]:= SubstTest[subclass, u, intersection[v, w], {u -> intersection[FUNS,
             P[x], complement[image[inverse[PS], intersection[FUNS, P[x]]]]],
           v -> image[inverse[IMAGE[FIRST]], image[S, set[domain[x]]]],
           w → intersection[FUNS, P[x]]}] // Reverse

Out[23]= subclass[intersection[FUNS, P[x]],
           union[image[inverse[PS], intersection[FUNS, P[x]]], X[x]]] == True

In[24]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. (The inclusions in both directions are combined into an equation which says that any function contained in a class **x** is either not maximal or else a cross-section.)

```
In[25]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> intersection[FUNS, P[x]],
           v -> union[image[inverse[PS], intersection[FUNS, P[x]]], X[x]]}]

Out[25]= equal[intersection[FUNS, P[x]],
           union[image[inverse[PS], intersection[FUNS, P[x]]], X[x]]] == True

In[26]:= union[image[inverse[PS], intersection[FUNS, P[x_]]], X[x_]] := intersection[FUNS, P[x]]
```

Corollary. The cross-sections of a class **x** are the maximal elements of the class **intersection[FUNS, P[x]]** of functions contained in **x**.

```
In[27]:= Map[equal[X[x], #] &, SubstTest[intersection, u, union[v, w],
             {u → complement[image[inverse[PS], intersection[FUNS, P[x]]]], v → X[x],
              w -> image[inverse[PS], intersection[FUNS, P[x]]]}]] // Reverse

Out[27]= equal[intersection[FUNS,
             complement[image[inverse[PS], intersection[FUNS, P[x]]]], P[x]], X[x]] == True

In[28]:= intersection[FUNS,
           complement[image[inverse[PS], intersection[FUNS, P[x_]]]], P[x_]] := X[x]
```