cross-sections and equipollence

Johan G. F. Belinfante
2008 May 22

```
In[1]:= SetDirectory["l:"]; << goedel.08may21a; << tools.m

        :Package Title: goedel.08may21a                 2008 May 21 at 11:55 p.m.

        It is now:  2008 May 22 at 12:52

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 May 17

        weightlimit = 40
```

summary

Cross-sections of **x** are equipollent to the domain of **x**. If a set admits a cross-section, then its domain is equipollent to a subset.

derivation

Lemma.

```
In[2]:= Map[not, SubstTest[and, implies[and[p1, p3], p5], implies[and[p2, p4, p5], p6],
           implies[and[p1, p3, p5, p6], p7], not[implies[and[p1, p2, p3, p4], p7]],
           {p1 → member[x, V], p2 → FUNCTION[t], p3 → subclass[t, x], p4 →
             equal[domain[t], domain[x]], p5 → member[t, V], p6 → member[pair[domain[x], t], Q],
            p7 -> member[pair[domain[x], x], composite[Q, S]]}]] // Reverse

Out[2]= or[member[domain[x], image[Q, P[x]]], not[equal[domain[t], domain[x]]],
          not[FUNCTION[t]], not[member[x, V]], not[subclass[t, x]]] == True

In[3]:= (% /. {x → x_, t → t_}) /. Equal → SetDelayed
```

Theorem. (Eliminating the variable **t**.) If a set admits a cross-section, then its domain is equipollent to a subset.

```
In[4]:= Map[implies[member[x, y], equal[V, #]] &,
         SubstTest[class, t, implies[and[member[x, V], member[t, u]], member[v, w]],
          {u → X[x], v -> domain[x], w -> image[Q, P[x]]}]]

Out[4]= or[equal[0, X[x]], member[domain[x], image[Q, P[x]]], not[member[x, y]]] == True

In[5]:= or[equal[0, X[x_]], member[domain[x_], image[Q, P[x_]]], not[member[x_, y_]]] := True
```

Theorem. (Eliminating the variable **x**.) Variable-free restatement.

```
In[6]:= Map[equal[V, #] &,
          SubstTest[class, x, implies[member[x, s], member[pair[domain[x], x], t]],
           {s → SELECT, t → composite[S, Q]}]]

Out[6]= subclass[SELECT, fix[composite[Q, S, IMAGE[FIRST]]]] == True

In[7]:= subclass[SELECT, fix[composite[Q, S, IMAGE[FIRST]]]] := True
```

Corollary.

```
In[8]:= Map[implies[axch, #] &, SubstTest[and, equal[V, x],
           subclass[x, y], {x → SELECT, y -> fix[composite[Q, S, IMAGE[FIRST]]]}]]

Out[8]= or[equal[V, fix[composite[Q, S, IMAGE[FIRST]]]], not[axch]] == True

In[9]:= or[equal[V, fix[composite[Q, S, IMAGE[FIRST]]]], not[axch]] := True
```