combinatorial intersections of partitions

Johan G. F. Belinfante
2010 January 1

```
In[1]:= SetDirectory["l:"]; << goedel.09dec30a;<< tools.m

        :Package Title: goedel.09dec30a              2009 December 30 at 6:05 a.m.

        It is now:  2010 Jan 1 at 17:28

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 December 17

        weightlimit = 40
```

summary

If **x** and **y** are pairwise disjoint collections of sets, then so is their combinatorial intersection, **image[CAP, cart[x, y]]**. The derivation presented in this notebook is remarkable in that it avoids introducing any additional variables beyond those already in the statement of the theorem.

By contrast, a typical proof that one might give by hand would likely start by considering two typical elements of the combinatorial intersection, say **w** = **u** ⋂ **v** and **w'** = **u'** ⋂ **v'**, where **u, u'** ∈ **x** and **v, v'** ∈ **y**. One would then reason that if **w** and **w'** were not disjoint, then **u** and **u'** can not be disjoint, and also **v** and **v'** can not be disjoint. The hypothesis that **x** and **y** are pairwise disjoint collections then implies that **u** = **u'** and **v** = **v'**, whence **w** = **w'**.

The proof given below avoids all these extra variables and instead relies on the fact that **x** is a pairwise disjoint collection iff **id[x]** ∘ **E** is a function, and the fact that the cross product of two functions is a function. The advantage of not introducing extra variables is that one avoids the task of subsequently having to eliminate all of them.

derivation

Lemma.

```
In[2]:= SubstTest[implies, and[FUNCTION[u], FUNCTION[v]], FUNCTION[composite[cross[u, v], DUP]],
          {u -> composite[id[x], E], v → composite[id[y], E]}] // Reverse

Out[2]= or[FUNCTION[composite[id[cart[x, y]], inverse[CAP], E]],
          not[subclass[cart[x, x], union[DISJOINT, Id]]],
          not[subclass[cart[y, y], union[DISJOINT, Id]]]] == True

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

Lemma.

```
In[4]:= SubstTest[implies, and[FUNCTION[u], FUNCTION[v]], FUNCTION[composite[u, v]],
          {u → CAP, v → composite[id[cart[x, y]], inverse[CAP], E]}] // Reverse

Out[4]= or[not[FUNCTION[composite[id[cart[x, y]], inverse[CAP], E]]], subclass[
          cart[image[CAP, cart[x, y]], image[CAP, cart[x, y]]], union[DISJOINT, Id]]] == True

In[5]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. The combinatorial intersection of two pairwise disjoint collections is pairwise disjoint.

```
In[6]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[p3, p4],
           not[implies[and[p1, p2], p4]], {p1 -> subclass[cart[x, x], union[DISJOINT, Id]],
            p2 -> subclass[cart[y, y], union[DISJOINT, Id]],
            p3 -> FUNCTION[composite[id[cart[x, y]], inverse[CAP], E]],
            p4 -> subclass[cart[image[CAP, cart[x, y]], image[CAP, cart[x, y]]],
              union[DISJOINT, Id]]}]] // Reverse

Out[6]= or[not[subclass[cart[x, x], union[DISJOINT, Id]]],
          not[subclass[cart[y, y], union[DISJOINT, Id]]], subclass[
           cart[image[CAP, cart[x, y]], image[CAP, cart[x, y]]], union[DISJOINT, Id]]] == True

In[7]:= or[not[subclass[cart[x_, x_], union[DISJOINT, Id]]],
          not[subclass[cart[y_, y_], union[DISJOINT, Id]]],
          subclass[cart[image[CAP, cart[x_, y_]], image[CAP, cart[x_, y_]]],
           union[DISJOINT, Id]]] := True
```

A variable-free formulation is possible.

Theorem. The class **cliques[union[DISJOINT, Id]]** is binary-closed under **IMAGE[CAP]** ∘ **CART**.

```
In[8]:= Map[empty[domain[complement[#]]] &,
         SubstTest[class, pair[x, y], implies[and[subclass[P[x], z], subclass[P[y], z]],
           subclass[P[image[CAP, cart[x, y]]], z]], z → cliques[union[DISJOINT, Id]]]]

Out[8]= subclass[image[IMAGE[CAP],
           image[CART, cart[cliques[union[DISJOINT, Id]], cliques[union[DISJOINT, Id]]]]],
          cliques[union[DISJOINT, Id]]] == True

In[9]:= subclass[image[IMAGE[CAP],
           image[CART, cart[cliques[union[DISJOINT, Id]], cliques[union[DISJOINT, Id]]]]],
          cliques[union[DISJOINT, Id]]] := True
```

historical and terminological remarks

For each binary constructor, there is another one that A. P. Morse describes as its **combinatorial** counterpart. If **x ∗ y** denotes a binary constructor, his notation for the combinatorial counterpart is **x ∗∗ y**. The latter is defined as the class of all sets **u ∗ v** where **u ∈ x** and **v ∈ y**.

```
In[10]:= "Anthony P. Morse, A Theory of Sets, Second
         Edition, Academic Press, Inc., Orlando, Florida, 1986.";
```

Anthony P. Morse's book on set theory reads as a quasi-computer program. His notations are concise albeit sometimes cryptic, but the **GOEDEL** program is pretty good at deciphering such discourse. Morse's book, first published in 1965, was the basis for an early attempt (by Woody Bledsoe and E. J. Gilbert in 1966) to use computers for automated reasoning in set theory.

Morse developed an elaborate terminology about mathematical notation. For example, he uses the term **verbless binarian** for the symbol $*$. (See his Definitional Schema 0.62 on page 30.)

A typical example of the combinatorial counterpart of binary constructor is the relation between the binary constructors **pair** and **cart**. Morse writes **x , y** for **pair[x, y]**, and **x ,, y** for **cart[x, y]**.

```
In[11]:= class[w, exists[u, v, and[member[u, x], member[v, y], equal[w, pair[u, v]]]]]
Out[11]= cart[x, y]
```

Similarly, Morse writes **x ⋂ y** for the binary constructor **intersection[x, y]**, and **x ⋂⋂ y** for its binary counterpart:

```
In[12]:= class[w, exists[u, v, and[member[u, x], member[v, y], equal[w, intersection[u, v]]]]]
Out[12]= image[CAP, cart[x, y]]
```

For each binary constructor which takes pairs of sets to sets, such as **intersection**, there is a corresponding binary function, in this case **CAP**.

```
In[13]:= class[pair[pair[x, y], z], equal[z, intersection[x, y]]]
Out[13]= CAP
```

The function corresponding to combinatorial intersection is

```
In[14]:= class[pair[pair[x, y], z], equal[z, image[CAP, cart[x, y]]]]
Out[14]= composite[IMAGE[CAP], CART]
```

In general, if **f** is a binary function corresponding to some binary constructor, then the binary function corresponding to the corresponding combinatorial binary constructor is **IMAGE[f] ∘ CART**. For the case of **pair** and **cart**, the corresponding functions are:

```
In[15]:= f == class[pair[pair[x, y], z], equal[z, pair[x, y]]]
Out[15]= f == id[cart[V, V]]
```

and

```
In[16]:= composite[IMAGE[id[cart[V, V]]], CART] ==
          class[pair[pair[x, y], z], equal[z, cart[x, y]]]
Out[16]= True
```

A. P. Morse uses the term **disjointed** class for a pairwise disjoint collection of sets. (See his Definition 2.184.0 on page 147.)

```
In[17]:= disjointed[x] == assert[forall[u, v,
           implies[and[member[u, x], member[v, x]], or[equal[u, v], disjoint[u, v]]]]]

Out[17]= disjointed[x] == subclass[cart[x, x], union[DISJOINT, Id]]
```

Morse denotes the class of all disjointed sets by **dsjn**. (Definition 1.184.1 on page 147.)

```
In[18]:= dsjn == class[x, subclass[cart[x, x], union[DISJOINT, Id]]]

Out[18]= dsjn == cliques[union[DISJOINT, Id]]
```

In his terminology, the theorem proved in this notebook says that if **x** and **y** are disjointed, then so is their combinatorial intersection:

disjointed[x] & disjointed[y] $\Longrightarrow$ **disjointed[x $\bigcap\bigcap$ y]**.