relations similar to a cartesian product

Johan G. F. Belinfante
2010 May 10

```
In[1]:= SetDirectory["l:"]; << goedel.10may09a;<< tools.m

        :Package Title: goedel.10may09a               2010 May 9 at 9:30 a.m.

        It is now:  2010 May 10 at 13:27

        Loading Simplification Rules

        TOOLS.M                                    Revised 2010 February 26

        weightlimit = 40
```

summary

A relation similar to a cartesian product is a cartesian product.

IMG ∘ (CROSS ⊗ CART)

In this section a variable-free formulation of the following observation is derived.

```
In[2]:= image[cross[u, v], cart[x, y]]

Out[2]= cart[image[u, x], image[v, y]]
```

Lemma. Removal of variables using **reify**.

```
In[3]:= Map[composite[Id, complement[#]] &, SubstTest[reify, x,
          image[t, cart[cart[set[setpart[first[first[x]]]], set[setpart[second[first[x]]]]],
            cart[set[setpart[first[second[x]]]], set[setpart[second[second[x]]]]]],
          t -> complement[dif[composite[IMG, cross[CROSS, CART], TWIST],
             composite[CART, cross[IMG, IMG]]]]]]

Out[3]= composite[intersection[composite[complement[CART], cross[IMG, IMG]],
           composite[IMG, cross[CROSS, CART], TWIST]], id[cart[cart[V, V], cart[V, V]]] == 0

In[4]:= % /. Equal→ SetDelayed
```

Lemma. An inclusion.

```
In[5]:= SubstTest[empty, composite[dif[u, v], id[w]],
         {u -> composite[IMG, cross[CROSS, CART], TWIST],
          v → composite[CART, cross[IMG, IMG]], w -> cart[cart[V, V], cart[V, V]]}]

Out[5]= subclass[composite[IMG, cross[CROSS, CART], TWIST],
          composite[CART, cross[IMG, IMG]]] == True

In[6]:= % /. Equal→ SetDelayed
```

Inclusions of functions can always be replaced by equations.

Theorem. An equation.

```
In[7]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
          equal[u, composite[v, id[domain[u]]]], {u -> composite[IMG, cross[CROSS, CART], TWIST],
           v -> composite[CART, cross[IMG, IMG]]}] // Reverse

Out[7]= equal[composite[CART, cross[IMG, IMG]], composite[IMG, cross[CROSS, CART], TWIST]] == True

In[8]:= composite[IMG, cross[CROSS, CART], TWIST] := composite[CART, cross[IMG, IMG]]
```

---

similarity and cartesian products

Lemma. Composites with cartesian products are cartesian products.

```
In[9]:= ImageComp[IMG, composite[cross[CROSS, CART], TWIST], V] // Reverse

Out[9]= image[IMG, cart[range[CROSS], range[CART]]] == range[CART]

In[10]:= image[IMG, cart[range[CROSS], range[CART]]] := range[CART]
```

Theorem. A weak upper bound for **SIMILAR**.

```
In[11]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]], {t ->
             intersection[composite[inverse[FIRST], SECOND], composite[inverse[SECOND], IMG]],
            u → composite[inverse[S], IMAGE[FIRST], id[image[CROSS, id[BIJ]]]],
            v -> cart[range[CROSS], V]}] // Reverse

Out[11]= subclass[SIMILAR, composite[IMG, id[cart[range[CROSS], V]], inverse[SECOND]]] == True

In[12]:= subclass[SIMILAR, composite[IMG, id[cart[range[CROSS], V]], inverse[SECOND]]] := True
```

Lemma. The class of cartesian products is invariant under **SIMILAR**.

```
In[13]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
           {u → SIMILAR, v -> composite[IMG, id[cart[range[CROSS], V]], inverse[SECOND]],
            w → range[CART]}] // Reverse

Out[13]= subclass[image[SIMILAR, range[CART]], range[CART]] == True

In[14]:= % /. Equal→ SetDelayed
```

Theorem. An equation. Relations similar to cartesian products are cartesian products.

```
In[15]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[SIMILAR, range[CART]], v -> range[CART]}]

Out[15]= equal[image[SIMILAR, range[CART]], range[CART]] == True

In[16]:= image[SIMILAR, range[CART]] := range[CART]
```

A constant function is a function that is also a cartesian product.

```
In[17]:= intersection[FUNS, range[CART]]

Out[17]= CONST
```

Corollary. Relations similar to constant functions are constant functions.

```
In[18]:= SubstTest[implies, and[equal[image[SIMILAR, x], x], equal[image[SIMILAR, y], y]],
           equal[image[SIMILAR, intersection[x, y]], intersection[x, y]],
           {x → FUNS, y → range[CART]}] // Reverse

Out[18]= equal[CONST, image[SIMILAR, CONST]] == True

In[19]:= image[SIMILAR, CONST] := CONST
```