mapping properties of APPLY[CURRY, w], part 1.

Johan G. F. Belinfante
2006 November 12

```
In[1]:= SetDirectory["l:"]; << goedel87.11b; << tools.m

        :Package Title: goedel87.11b          2006 November 11 at 7:15 p.m.

        It is now:  2006 Nov 12 at 10:34

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 November 5

        weightlimit = 40
```

summary

It is shown in this notebook that if **w** is a mapping from **cart[x, y]** to **z**, and if **y** is not empty, then **APPLY[CURRY, w]** is a mapping from **x** to **map[y, z]**. Technical comments: Most of the work is concerned with showing that the range of the curried function is a subclass of **map[y,z]**. For several key steps, a lemma is first derived using compound wrappers, and later the wrappers are removed and the result is cleaned up. For some of the more lengthy proofs, a speedup by a factor of ten or twenty is achieved by simply leaving out several simple steps, relying on rewrite rules to fill the missing gaps.

FUNS condition

Lemma.

```
In[2]:= Map[range[A[#]] &, ImageComp[VS, IMAGE[ASSOC], set[funpart[setpart[w]]]]] // Reverse

Out[2]= image[IMAGE[SWAP],
          image[VERTSECT[inverse[rotate[composite[funpart[setpart[w]], SWAP]]],
           domain[domain[funpart[setpart[w]]]]] ==
         range[APPLY[CURRY, composite[funpart[setpart[w]], id[cart[V, V]]]]]

In[3]:= image[IMAGE[SWAP],
          image[VERTSECT[inverse[rotate[composite[funpart[setpart[w_]], SWAP]]],
           domain[domain[funpart[setpart[w_]]]]] :=
         range[APPLY[CURRY, composite[funpart[setpart[w]], id[cart[V, V]]]]]
```

Theorem.

```
In[4]:= Map[empty, dif[domain[domain[funpart[setpart[w]]]],
          image[inverse[VERTSECT[inverse[rotate[composite[funpart[setpart[w]], SWAP]]]]],
           image[inverse[IMAGE[SWAP]], FUNS]]] // Renormality]

Out[4]= subclass[range[APPLY[CURRY, composite[funpart[setpart[w]], id[cart[V, V]]]]], FUNS] ==
         True

In[5]:= (% /. w → w_) /. Equal → SetDelayed
```

U[range[...]] condition

This first lemma, proved using **ReifNormality**, has independent interest.

```
In[6]:= composite[inverse[E], IMAGE[SWAP], VERTSECT[x]] // ReifNormality

Out[6]= composite[inverse[E], IMAGE[SWAP], VERTSECT[x]] == composite[SWAP, thinpart[x]]

In[7]:= composite[inverse[E], IMAGE[SWAP], VERTSECT[x_]] := composite[SWAP, thinpart[x]]
```

Theorem.

```
In[8]:= ImageComp[inverse[E], composite[IMAGE[SWAP],
          VERTSECT[inverse[rotate[composite[funpart[setpart[w]], SWAP]]]],
         domain[domain[funpart[setpart[w]]]]] // Reverse

Out[8]= U[range[APPLY[CURRY, composite[funpart[setpart[w]], id[cart[V, V]]]]] ==
         composite[funpart[setpart[w]],
          id[cart[domain[domain[funpart[setpart[w]]]], V]], inverse[SECOND]]

In[9]:= U[range[APPLY[CURRY, composite[funpart[setpart[w_]], id[cart[V, V]]]]]] :=
         composite[funpart[setpart[w]],
          id[cart[domain[domain[funpart[setpart[w]]]], V]], inverse[SECOND]]
```

domains of members of the range

Lemma. (Here the recently introduced **ReifNormality** test is exploited.)

```
In[10]:= composite[IMAGE[SECOND], VERTSECT[inverse[rotate[w]]]] // ReifNormality

Out[10]= composite[IMAGE[SECOND], VERTSECT[inverse[rotate[w]]]] == composite[
           VERTSECT[inverse[domain[w]]], id[domain[VERTSECT[composite[w, inverse[SECOND]]]]]]

In[11]:= composite[IMAGE[SECOND], VERTSECT[inverse[rotate[w_]]]] := composite[
           VERTSECT[inverse[domain[w]]], id[domain[VERTSECT[composite[w, inverse[SECOND]]]]]]
```

Theorem. (The use of **ImageComp** is carefully set up to introduce **CURRY** application.)

```
In[12]:= ImageComp[IMAGE[FIRST], composite[IMAGE[SWAP],
           VERTSECT[inverse[rotate[composite[funpart[setpart[w]], SWAP]]]],
          domain[domain[funpart[setpart[w]]]]] // Reverse

Out[12]= image[IMAGE[FIRST],
          range[APPLY[CURRY, composite[funpart[setpart[w]], id[cart[V, V]]]]] ==
         image[VERTSECT[domain[funpart[setpart[w]]]], domain[domain[funpart[setpart[w]]]]]

In[13]:= image[IMAGE[FIRST],
          range[APPLY[CURRY, composite[funpart[setpart[w_]], id[cart[V, V]]]]] :=
         image[VERTSECT[domain[funpart[setpart[w]]]], domain[domain[funpart[setpart[w]]]]]
```

the complete range condition

Lemma. (This lemma indicates that what is left to be done mainly involves using properties of the domain of **w**.)

```
In[14]:= Map[implies[#,
           subclass[range[APPLY[CURRY, composite[funpart[setpart[w]], id[cart[V, V]]]]],
            map[y, z]]] &, SubstTest[subclass,
          range[APPLY[CURRY, composite[funpart[setpart[w]], id[cart[V, V]]]]],
          intersection[t, u, v], {t -> FUNS, u -> image[inverse[IMAGE[FIRST]], set[y]],
           v -> image[inverse[IMAGE[SECOND]], P[z]]}]]

Out[14]= or[not[subclass[composite[Id, domain[funpart[setpart[w]]]],
            image[inverse[funpart[setpart[w]]], z]]],
          not[subclass[image[VERTSECT[domain[funpart[setpart[w]]]],
             domain[domain[funpart[setpart[w]]]]], set[y]]],
          subclass[range[APPLY[CURRY, composite[funpart[setpart[w]], id[cart[V, V]]]]],
           map[y, z]]] == True

In[15]:= (% /. {w → w_, y → y_, z → z_}) /. Equal -> SetDelayed
```

The wrappers on **w** can be removed, and the domain of **w** specialized to cartesian products as follows:

```
In[16]:= SubstTest[implies,
          and[equal[s, domain[w]], equal[w, composite[funpart[setpart[t]], id[cart[V, V]]]],
           subclass[composite[Id, s], image[inverse[w], z]],
           subclass[image[VERTSECT[s], domain[s]], set[y]]],
          subclass[range[APPLY[CURRY, w]], map[y, z]], {t → w, s → cart[x, y]}] // Reverse

Out[16]= or[not[equal[cart[x, y], domain[w]]], not[FUNCTION[w]],
          not[member[w, V]], not[subclass[cart[x, y], image[inverse[w], z]]],
          not[subclass[domain[w], cart[V, V]]],
          subclass[range[APPLY[CURRY, w]], map[y, z]]] == True

In[17]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal -> SetDelayed
```

Theorem. (This goes ten times faster when some steps are left out, to be filled in by the action of rewrite rules.)

```
In[18]:= Map[not, SubstTest[and, implies[p1, p5], implies[p2, p6], implies[and[p2, p5], p7],
           implies[and[p2, p3, p4, p6, p7], p8], not[implies[p1, p8]],
           {p1 → member[w, map[cart[x, y], z]], p2 -> equal[cart[x, y], domain[w]],
            p3 → FUNCTION[w], p4 → member[w, V], p5 → subclass[range[w], z], p6 ->
             subclass[domain[w], cart[V, V]], p7 -> subclass[cart[x, y], image[inverse[w], z]],
            p8 -> subclass[range[APPLY[CURRY, w]], map[y, z]]}]] // Reverse

Out[18]= or[not[member[w, map[cart[x, y], z]]],
           subclass[range[APPLY[CURRY, w]], map[y, z]]] == True

In[19]:= or[not[member[w_, map[cart[x_, y_], z_]]],
           subclass[range[APPLY[CURRY, w_]], map[y_, z_]]] := True
```

---

finishing the job

Lemma.

```
In[20]:= Map[implies[#, member[APPLY[CURRY, w], map[x, map[y, z]]]] &,
          SubstTest[member, APPLY[CURRY, w], intersection[t, u, v],
           {t -> FUNS, u -> image[inverse[IMAGE[FIRST]], set[x]],
            v -> image[inverse[IMAGE[SECOND]], P[map[y, z]]]}]]

Out[20]= or[member[APPLY[CURRY, w], map[x, map[y, z]]],
           not[equal[x, domain[domain[w]]]], not[member[w, V]],
           not[member[domain[domain[w]], V]], not[subclass[w, cart[cart[V, V], V]]],
           not[subclass[range[APPLY[CURRY, w]], map[y, z]]]] == True

In[21]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Main Theorem. (This derivation goes twenty times faster when one leaves out many steps of the complete proof.)

```
In[22]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p3, p4, p5, p6, p7], p8],
           not[implies[p1, p8]], {p1 → and[member[w, map[cart[x, y], z]], not[empty[y]]],
            p2 → equal[domain[w], cart[x, y]],
            p3 → equal[x, domain[domain[w]]], p4 → member[w, V],
            p5 -> member[domain[domain[w]], V], p6 -> subclass[w, cart[cart[V, V], V]],
            p7 -> subclass[range[APPLY[CURRY, w]], map[y, z]],
            p8 -> member[APPLY[CURRY, w], map[x, map[y, z]]]}]] // Reverse

Out[22]= or[equal[0, y], member[APPLY[CURRY, w], map[x, map[y, z]]],
           not[member[w, map[cart[x, y], z]]]] == True

In[23]:= or[equal[0, y_], member[APPLY[CURRY, w_], map[x_, map[y_, z_]]],
           not[member[w_, map[cart[x_, y_], z_]]]] := True
```

U[range[APPLY[CURRY, w]]] theorem

In this final section, the **U[range[...]]** theorem is revisited. It is convenient to first remove all the old wrappers, and then reintroduce a new **funpart** wrapper to clean up the final result.

```
In[24]:= (SubstTest[implies, and[equal[s, composite[t, id[cart[domain[domain[t]], V]]]],
            equal[t, composite[funpart[setpart[w]], id[cart[V, V]]]]],
          equal[U[range[APPLY[CURRY, t]]], composite[s, inverse[SECOND]]],
          {s → w, t → w}] // Reverse) /. w → funpart[r]

Out[24]= or[equal[composite[funpart[r], inverse[SECOND]], U[range[APPLY[CURRY, funpart[r]]]]],
           not[member[funpart[r], V]], not[subclass[domain[funpart[r]], cart[V, V]]]] == True

In[25]:= (% /. r → r_) /. Equal → SetDelayed
```

Removing this newly introduced **funpart** wrapper, one finds:

```
In[26]:= SubstTest[implies, equal[w, funpart[r]],
           or[equal[composite[w, inverse[SECOND]], U[range[APPLY[CURRY, w]]]],
            not[member[w, V]], not[subclass[domain[w], cart[V, V]]], r → w] // Reverse

Out[26]= or[equal[composite[w, inverse[SECOND]], U[range[APPLY[CURRY, w]]]],
           not[FUNCTION[w]], not[member[w, V]], not[subclass[domain[w], cart[V, V]]]] == True

In[27]:= or[equal[composite[w_, inverse[SECOND]], U[range[APPLY[CURRY, w_]]]],
           not[FUNCTION[w_]], not[member[w_, V]], not[subclass[domain[w_], cart[V, V]]]] := True
```

Corollary

```
In[28]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p1, p4],
            implies[p4, p5], implies[and[p2, p3, p5], p6], not[implies[p1, p6]],
            {p1 → member[w, map[cart[x, y], z]], p2 → FUNCTION[w], p3 → member[w, V],
             p4 → equal[domain[w], cart[x, y]], p5 → subclass[domain[w], cart[V, V]],
             p6 -> equal[composite[w, inverse[SECOND]], U[range[APPLY[CURRY, w]]]]}] // Reverse

Out[28]= or[equal[composite[w, inverse[SECOND]], U[range[APPLY[CURRY, w]]]],
           not[member[w, map[cart[x, y], z]]]] == True

In[29]:= or[equal[composite[w_, inverse[SECOND]], U[range[APPLY[CURRY, w_]]]],
           not[member[w_, map[cart[x_, y_], z_]]]] := True
```