mapping properties of uncurried functions

Johan G. F. Belinfante
2006 November 16

```
In[1]:= SetDirectory["l:"]; << goedel87.16a; << tools.m

        :Package Title: goedel87.16a          2006 November 16 at 5:40 a.m.

        It is now:  2006 Nov 16 at 14:31

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 November 5

        weightlimit = 40
```

summary

In this notebook it is shown that if **w** is a mapping from **x** to **map[y, z]**, and if **y** is not empty, then **APPLY[inverse[-CURRY], w]** is a mapping from **cart[x, y]** to **z**. The main idea is to use three explicit formulas for **APPLY[inverse[-CURRY], w]** to deduce its properties. Comments: Many lemmas are used. The derivations often omit proof steps, relying on rewrite rules to fill in the gaps.

APPLY formulas

This **APPLY** formula is already available:

```
In[2]:= or[equal[0, y], equal[APPLY[inverse[CURRY], w], rotate[composite[inverse[w], E]]],
          not[member[w, map[x, map[y, z]]]]]

Out[2]= True
```

Triple rotation yields a minor variant:

```
In[3]:= Map[implies[and[member[w, map[x, map[y, z]]], not[empty[y]]],
            equal[APPLY[inverse[CURRY], w], #]] &,
          composite[rotate[E], cross[w, Id]] // TripleRotate]

Out[3]= or[equal[0, y], equal[APPLY[inverse[CURRY], w], composite[rotate[E], cross[w, Id]]],
          not[member[w, map[x, map[y, z]]]]] == True

In[4]:= or[equal[0, y_], equal[APPLY[inverse[CURRY], w_], composite[rotate[E], cross[w_, Id]]],
          not[member[w_, map[x_, map[y_, z_]]]]] := True
```

Lemma.

```
In[5]:= Assoc[rotate[E], cross[FUNPART, Id], cross[w, Id]]

Out[5]= composite[rotate[E], cross[composite[FUNPART, w], Id]] ==
         composite[inverse[SINGLETON], IMG, cross[w, SINGLETON]]

In[6]:= composite[rotate[E], cross[composite[FUNPART, w_], Id]] :=
         composite[inverse[SINGLETON], IMG, cross[w, SINGLETON]]
```

Lemma.

```
In[7]:= SubstTest[implies, equal[w, x], equal[image[y, w], image[y, x]],
          {x → composite[FUNPART, w],
           y -> composite[cross[Id, rotate[E]], id[composite[inverse[SECOND], SECOND]],
             cross[inverse[FIRST], inverse[FIRST]]]}] // Reverse

Out[7]= or[equal[composite[rotate[E], cross[w, Id]],
            composite[inverse[SINGLETON], IMG, cross[w, SINGLETON]]],
          not[subclass[w, cart[V, FUNS]]]] == True

In[8]:= (% /. w → w_) /. Equal → SetDelayed
```

Theorem. A third **APPLY** formula.

```
In[9]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
           implies[p3, p4], implies[p5, p6], not[implies[p1, p7]],
           {p1 → and[member[w, map[x, map[y, z]]], not[empty[y]]],
            p2 → FUNCTION[w], p3 → subclass[range[w], map[y, z]],
            p4 → subclass[range[w], FUNS], p5 -> subclass[w, cart[V, FUNS]],
            p6 -> equal[composite[rotate[E], cross[w, Id]], composite[inverse[SINGLETON],
               IMG, cross[w, SINGLETON]]], p7 → equal[APPLY[inverse[CURRY], w],
              composite[inverse[SINGLETON], IMG, cross[w, SINGLETON]]]}]] // Reverse

Out[9]= or[equal[0, y], equal[APPLY[inverse[CURRY], w],
            composite[inverse[SINGLETON], IMG, cross[w, SINGLETON]]],
          not[member[w, map[x, map[y, z]]]]] == True

In[10]:= or[equal[0, y_], equal[APPLY[inverse[CURRY], w_],
            composite[inverse[SINGLETON], IMG, cross[w_, SINGLETON]]],
           not[member[w_, map[x_, map[y_, z_]]]]] := True
```

FUNCTION lemma

The third **APPLY** formula has this corollary:

```
In[11]:= (Map[not, SubstTest[and, implies[and[p0, p1], p2], implies[p1, p3],
             implies[and[p0, p2], p4], implies[and[p3, p4], p5], not[implies[and[p0, p1], p5]],
             {p0 → equal[w, t], p1 -> and[member[w, map[x, map[y, z]]], not[equal[0, y]]],
              p2 → equal[w, funpart[t]],
              p3 -> equal[APPLY[inverse[CURRY], w], composite[
                 inverse[SINGLETON], IMG, cross[w, SINGLETON]]],
              p4 → FUNCTION[composite[inverse[SINGLETON], IMG, cross[w, SINGLETON]]],
              p5 -> FUNCTION[APPLY[inverse[CURRY], w]]}]] /. t → w) // Reverse

Out[11]= or[equal[0, y], FUNCTION[APPLY[inverse[CURRY], w]],
           not[member[w, map[x, map[y, z]]]]] == True

In[12]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

---

range lemma

The original **APPLY** formula is used here:

```
In[13]:= Map[not, SubstTest[and, implies[p0, p1], implies[p0, p2], implies[p1, p3],
            not[implies[p0, p4]], {p0 → and[member[w, map[x, map[y, z]]], not[equal[0, y]]],
             p1 → subclass[range[w], map[y, z]],
             p2 → equal[APPLY[inverse[CURRY], w], rotate[composite[inverse[w], E]]],
             p3 → subclass[range[U[range[w]]], z],
             p4 -> subclass[range[APPLY[inverse[CURRY], w]], z]}]] // Reverse

Out[13]= or[equal[0, y], not[member[w, map[x, map[y, z]]]],
           subclass[range[APPLY[inverse[CURRY], w]], z]] == True

In[14]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

---

domain theorem

Theorem. (Formula for the domain.)

```
In[15]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p0, p2]],
            {p0 → and[member[w, map[x, map[y, z]]], not[equal[0, y]]],
             p1 → equal[APPLY[inverse[CURRY], w], rotate[composite[inverse[w], E]]],
             p2 → equal[domain[APPLY[inverse[CURRY], w]],
               composite[inverse[E], IMAGE[FIRST], w]]}]] // Reverse

Out[15]= or[equal[0, y], equal[composite[inverse[E], IMAGE[FIRST], w],
           domain[APPLY[inverse[CURRY], w]]], not[member[w, map[x, map[y, z]]]]] == True

In[16]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. (Upper bound for the domain.)

```
In[17]:= Map[not, SubstTest[and, implies[p2, p5], implies[p0, p1],
            implies[p0, p2], implies[p0, p4], implies[and[p1, p4, p5], p6],
            not[implies[p0, p6]], {p0 → and[member[w, map[x, map[y, z]]], not[equal[0, y]]],
             p1 → equal[domain[w], x], p2 → subclass[range[w], map[y, z]],
             p4 → equal[APPLY[inverse[CURRY], w], rotate[composite[inverse[w], E]]],
             p5 → subclass[domain[U[range[w]]], y],
             p6 -> subclass[domain[APPLY[inverse[CURRY], w]], cart[x, y]]}]] // Reverse

Out[17]= or[equal[0, y], not[member[w, map[x, map[y, z]]]],
           subclass[domain[APPLY[inverse[CURRY], w]], cart[x, y]]] == True

In[18]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[19]:= SubstTest[implies, subclass[u, v], subclass[composite[u, w], composite[v, w]],
           {u → cart[range[w], y], v -> composite[inverse[E], IMAGE[FIRST]]}] // Reverse

Out[19]= or[not[subclass[cart[range[w], y], composite[inverse[E], IMAGE[FIRST]]]],
           subclass[cart[domain[w], y], composite[inverse[E], IMAGE[FIRST], w]]] == True

In[20]:= (% /. {w → w_, y → y_}) /. Equal → SetDelayed
```

Theorem. (Lower bound for the domain.)

```
In[21]:= Map[not, SubstTest[and, implies[p0, p1], implies[p2, p3],
            implies[p0, p4], not[implies[p0, p5]],
            {p0 → and[member[w, map[x, map[y, z]]], not[equal[0, y]]],
             p1 → equal[domain[w], x], p2 → subclass[range[w], map[y, z]],
             p3 → subclass[cart[range[w], y], composite[inverse[E], IMAGE[FIRST]]],
             p4 →
              equal[domain[APPLY[inverse[CURRY], w]], composite[inverse[E], IMAGE[FIRST], w]],
             p5 → subclass[cart[x, y], domain[APPLY[inverse[CURRY], w]]]}]] // Reverse

Out[21]= or[equal[0, y], not[member[w, map[x, map[y, z]]]],
           subclass[cart[x, y], domain[APPLY[inverse[CURRY], w]]]] == True

In[22]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. (Equation for the domain.)

```
In[23]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
           {p -> and[member[w, map[x, map[y, z]]], not[equal[0, y]]],
            u -> domain[APPLY[inverse[CURRY], w]], v -> cart[x, y]}]

Out[23]= or[equal[0, y], equal[cart[x, y], domain[APPLY[inverse[CURRY], w]]],
           not[member[w, map[x, map[y, z]]]]] == True

In[24]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

main theorem

The first lemma provides everything except for the sethood condition on **t = APPLY[inverse[CURRY],w]**.

```
In[25]:= (Map[not, SubstTest[and, implies[p0, p1],
              implies[p0, p2], implies[p0, p3], implies[and[p1, p2, p3, p4], p5],
              not[implies[and[p0, p4], p5]], {p0 → and[member[w, map[x, map[y, z]]],
                 not[equal[0, y]], equal[t, APPLY[inverse[CURRY], w]]],
               p1 → FUNCTION[t], p2 -> subclass[range[t], z],
               p3 → equal[domain[t], cart[x, y]], p4 → member[t, V],
               p5 -> member[t, map[cart[x, y], z]]}]] // Reverse) /. t → APPLY[inverse[CURRY], w]

Out[25]= or[equal[0, y], member[APPLY[inverse[CURRY], w], map[cart[x, y], z]],
           not[FUNCTION[w]], not[member[w, map[x, map[y, z]]]],
           not[subclass[w, cart[V, intersection[complement[set[0]], P[cart[V, V]]]]]] == True

In[26]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[27]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p0, p3]],
            {p0 → and[member[w, map[x, map[y, z]]], not[equal[0, y]]],
             p1 → subclass[range[w], map[y, z]], p2 → subclass[U[range[w]], cart[y, z]],
             p3 -> subclass[U[range[w]], cart[V, V]]}]] // Reverse

Out[27]= or[equal[0, y], not[member[w, map[x, map[y, z]]]],
           subclass[U[range[w]], cart[V, V]]] == True

In[28]:= (% /. {w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Temporary lemma.

```
In[29]:= Map[implies[#, subclass[w, cart[V, P[x]]]] &,
           SubstTest[and, subclass[w, cart[V, V]], subclass[range[w], z], z → P[x]]] // Reverse

Out[29]= or[not[subclass[w, cart[V, V]]],
           not[subclass[U[range[w]], x]], subclass[w, cart[V, P[x]]]] == True

In[30]:= (% /. {w → w_, x → x_}) /. Equal → SetDelayed
```

Temporary lemma.

```
In[31]:= Map[implies[and[not[member[0, range[w]]], subclass[w, cart[V, V]],
              not[member[0, range[w]]], subclass[U[range[w]], cart[V, V]]], #] &,
           subclass[w, cart[V, intersection[complement[set[0]], P[cart[V, V]]]]] // AssertTest]

Out[31]= or[member[0, range[w]],
           not[subclass[w, cart[V, V]]], not[subclass[U[range[w]], cart[V, V]]],
           subclass[w, cart[V, intersection[complement[set[0]], P[cart[V, V]]]]]] == True

In[32]:= (% /. w → w_) /. Equal → SetDelayed
```

Combining these lemmas yields:

```
In[33]:= Map[not, SubstTest[and, implies[p1, p4],
           implies[and[p2, p3, p4], p5], not[implies[and[p1, p2, p3], p5]],
           {p1 -> FUNCTION[w], p2 -> subclass[U[range[w]], cart[V, V]],
            p3 -> not[member[0, range[w]]], p4 → subclass[w, cart[V, V]], p5 -> subclass[w,
              cart[V, intersection[complement[set[0]], P[cart[V, V]]]]]}]] // Reverse

Out[33]= or[member[0, range[w]], not[FUNCTION[w]], not[subclass[U[range[w]], cart[V, V]]],
           subclass[w, cart[V, intersection[complement[set[0]], P[cart[V, V]]]]]] == True

In[34]:= (% /. w → w_) /. Equal → SetDelayed

Main Theorem.

In[35]:= Map[not, SubstTest[and, implies[p0, p1], implies[p0, p2], implies[p0, p3],
           implies[p0, p4], implies[and[p0, p4], p5], implies[and[p0, p2, p6], p7],
           not[implies[p0, p7]], {p0 → and[member[w, map[x, map[y, z]]], not[equal[0, y]]],
            p1 → member[w, V], p2 → FUNCTION[w], p3 -> subclass[U[range[w]], cart[V, V]],
            p4 -> subclass[range[w], map[y, z]], p5 → not[member[0, range[w]]],
            p6 -> subclass[w, cart[V, intersection[complement[set[0]], P[cart[V, V]]]]],
            p7 -> member[APPLY[inverse[CURRY], w], map[cart[x, y], z]]}]] // Reverse

Out[35]= or[equal[0, y], member[APPLY[inverse[CURRY], w], map[cart[x, y], z]],
           not[member[w, map[x, map[y, z]]]]] == True

In[36]:= or[equal[0, y_], member[APPLY[inverse[CURRY], w_], map[cart[x_, y_], z_]],
           not[member[w_, map[x_, map[y_, z_]]]]] := True
```