applying a curried function

Johan G. F. Belinfante
2006 November 11

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

        :Package Title: goedel87.11a          2006 November 11 at 2:20 a.m.

        It is now:  2006 Nov 11 at 10:35

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 November 5

        weightlimit = 40
```

summary

In this notebook it is shown that if **w** is obtained by currying **x**, then applying **w** to any member **y** of its domain is equal to the composite of **x** with **LEFT[y]**. A corresponding formula for uncurrying is also derived.

the theorem for currying

Lemma.

```
In[2]:= Map[APPLY[A[#], y] &, ImageComp[VS, IMAGE[ASSOC], set[setpart[x]]]]

Out[2]= APPLY[APPLY[CURRY, composite[setpart[x], id[cart[V, V]]]], y] ==
         union[complement[image[V, intersection[domain[domain[setpart[x]]], set[y]]]],
          composite[setpart[x], LEFT[y]]]

In[3]:= APPLY[APPLY[CURRY, composite[setpart[x_], id[cart[V, V]]]], y_] :=
         union[complement[image[V, intersection[domain[domain[setpart[x]]], set[y]]]],
          composite[setpart[x], LEFT[y]]]
```

The **setpart** wrapper can be removed, and the result can be simplified as follows:

```
        (SubstTest[implies, and[equal[v, setpart[x]], equal[u, composite[v, id[cart[V, V]]]]],
            equal[APPLY[APPLY[CURRY, u], y],
             union[complement[image[V, intersection[domain[domain[v]], set[y]]]],
              composite[v, LEFT[y]]]], u → v] /. v → x) // Reverse

        or[equal[APPLY[APPLY[CURRY, x], y], composite[x, LEFT[y]]], not[member[x, V]],
          not[member[y, domain[domain[x]]]], not[subclass[x, cart[cart[V, V], V]]]] == True
```

```
In[4]:= or[equal[APPLY[APPLY[CURRY, x_], y_], composite[x_, LEFT[y_]]], not[member[x_, V]],
          not[member[y_, domain[domain[x_]]]], not[subclass[x_, cart[cart[V, V], V]]]] := True
```

Lemma.

```
In[10]:= SubstTest[implies, and[member[w, u], subclass[u, v]], member[w, v],
           {u → map[cart[x, y], z], v → P[cart[cart[V, V], V]]}] // Reverse

Out[10]= or[not[member[w, map[cart[x, y], z]]], subclass[w, cart[cart[V, V], V]]] == True

In[11]:= or[not[member[w_, map[cart[x_, y_], z_]]], subclass[w_, cart[cart[V, V], V]]] := True
```

Corollary.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
            implies[p1, p4], implies[and[p3, p4, p5], p6], not[implies[p1, p6]],
            {p1 → and[member[w, map[cart[x, y], z]], not[empty[y]], member[s, x]],
             p2 → equal[domain[w], cart[x, y]], p3 → member[s, domain[domain[w]]],
             p4 -> subclass[w, cart[cart[V, V], V]], p5 → member[w, V],
             p6 -> equal[APPLY[APPLY[CURRY, w], s], composite[w, LEFT[s]]]}] // Reverse

Out[15]= or[equal[0, y], equal[APPLY[APPLY[CURRY, w], s], composite[w, LEFT[s]]],
           not[member[s, x]], not[member[w, map[cart[x, y], z]]]] == True

In[16]:= or[equal[0, y_], equal[APPLY[APPLY[CURRY, w_], s_], composite[w_, LEFT[s_]]],
           not[member[s_, x_]], not[member[w_, map[cart[x_, y_], z_]]]] := True
```

---

uncurrying

A temporary abbreviation is introduced:

```
In[17]:= UNCURRY := composite[IMAGE[inverse[ASSOC]], IMAGE[cross[Id, inverse[E]]]]
```

The relation between **UNCURRY** and **inverse[CURRY]** is this:

```
In[19]:= composite[UNCURRY, id[range[CURRY]]]

Out[19]= inverse[CURRY]
```

If **w** belongs to **range[CURRY]**, then applying **inverse[CURRY]** to **w** is equivalent to applying **UNCURRY** to **w**. For sets, the latter yields a simple result.

```
In[21]:= APPLY[UNCURRY, setpart[w]]

Out[21]= rotate[composite[inverse[setpart[w]], E]]
```

This strategy yields the following lemma:

```
In[23]:= SubstTest[implies,
           and[equal[t, composite[u, id[r]]], member[s, r]], equal[APPLY[t, s], APPLY[u, s]],
           {r → range[CURRY], s → setpart[w], t → inverse[CURRY], u → UNCURRY}] // Reverse

Out[23]= or[
           equal[APPLY[inverse[CURRY], setpart[w]], rotate[composite[inverse[setpart[w]], E]]],
           not[FUNCTION[setpart[w]]], not[subclass[setpart[w],
             cart[V, intersection[complement[set[0]], P[cart[V, V]]]]]]] == True

In[24]:= (% /. w → w_) /. Equal → SetDelayed
```

Lemma.

```
In[25]:= SubstTest[implies, and[member[w, u], subclass[u, v]], member[w, v],
            {u → map[x, map[y, z]], v → range[CURRY]}] // Reverse // MapNotNot

Out[25]= or[equal[0, y], not[member[w, map[x, map[y, z]]]],
           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
```

Theorem.

```
In[27]:= (Map[not, SubstTest[and, implies[and[p1, p2], p4], implies[and[p1, p2, p3], p5],
             implies[and[p1, p4, p5], p6], not[implies[and[p1, p2, p3], p6]],
             {p1 → equal[w, setpart[t]], p2 → member[w, map[x, map[y, z]]],
              p3 → not[empty[y]], p4 → FUNCTION[setpart[t]], p5 -> subclass[setpart[t],
                cart[V, intersection[complement[set[0]], P[cart[V, V]]]]], p6 -> equal[APPLY[
                 inverse[CURRY], w], rotate[composite[inverse[w], E]]]}] // Reverse) /. t → w

Out[27]= or[equal[0, y], equal[APPLY[inverse[CURRY], w], rotate[composite[inverse[w], E]]],
           not[member[w, map[x, map[y, z]]]]] == True

In[28]:= or[equal[0, y_], equal[APPLY[inverse[CURRY], w_], rotate[composite[inverse[w_], E]]],
           not[member[w_, map[x_, map[y_, z_]]]]] := True
```

Lemma.

```
In[29]:= (Map[not, SubstTest[and, implies[p1, p2],
             implies[p2, p3], implies[and[p3, p4], p5], implies[and[p1, p4], p6],
             implies[and[p1, p4, p5, p6], p7], not[implies[and[p1, p4], p7]],
             {p1 → and[member[w, map[x, map[y, z]]], not[equal[0, y]], member[s, x]],
              p2 → equal[APPLY[inverse[CURRY], w], rotate[composite[inverse[w], E]]],
              p3 → equal[composite[Id, U[image[w, set[s]]]],
                composite[APPLY[inverse[CURRY], w], LEFT[s]]], p4 → equal[w, funpart[t]],
              p5 -> equal[composite[Id, U[image[funpart[t], set[s]]]], composite[
                 APPLY[inverse[CURRY], w], LEFT[s]]], p6 → equal[domain[funpart[t]], x],
              p7 -> equal[composite[Id, APPLY[w, s]], composite[
                 APPLY[inverse[CURRY], w], LEFT[s]]]}] // Reverse) /. t → w

Out[29]= or[equal[0, y],
           equal[composite[Id, APPLY[w, s]], composite[APPLY[inverse[CURRY], w], LEFT[s]]],
           not[FUNCTION[w]], not[member[s, x]], not[member[w, map[x, map[y, z]]]]] == True
```

```
In[30]:= (% /. {s → s_, w → w_, x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem.

```
In[31]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], implies[p1, p4],
           implies[p4, p5], implies[and[p3, p5], p6], not[implies[p1, p6]],
           {p1 -> and[member[w, map[x, map[y, z]]], not[equal[0, y]], member[s, x]],
            p2 → FUNCTION[w], p3 -> equal[composite[Id, APPLY[w, s]],
              composite[APPLY[inverse[CURRY], w], LEFT[s]]],
            p4 → member[APPLY[w, s], map[y, z]], p5 → subclass[APPLY[w, s], cart[V, V]],
            p6 -> equal[APPLY[w, s], composite[APPLY[inverse[CURRY], w], LEFT[s]]]}]] // Reverse

Out[31]= or[equal[0, y], equal[APPLY[w, s], composite[APPLY[inverse[CURRY], w], LEFT[s]]],
           not[member[s, x]], not[member[w, map[x, map[y, z]]]]] == True

In[32]:= or[equal[0, y_], equal[APPLY[w_, s_], composite[APPLY[inverse[CURRY], w_], LEFT[s_]]],
           not[member[s_, x_]], not[member[w_, map[x_, map[y_, z_]]]]] := True
```