images and inverse images of CURRY

Johan G. F. Belinfante
2006 November 23

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

:Package Title: goedel87.16b 2006 November 16 at 2:45 p.m.

It is now: 2006 Nov 23 at 15:36

Loading Simplification Rules

TOOLS.M Revised 2006 November 22

weightlimit = 40

summary

Applying CURRY to a mapping from cart[x, y] to z yields a mapping from x to map[y, z], provided y is not empty.

Applying inverse[CURRY] allows one to go in the opposite direction. An unconditional rewrite rule expressing these facts is derived in this notebook. A variable-free formulation is obtained by using reification. Membership rules for images and inverse images are derived in the final two sections of this notebook.

an image equation

A conditional equation for image[CURRY, map[cart[x, y], z]] is derived in this section by combining two inclusions, one obtained by considering the process of currying, and another by considering the reverse process of uncurrying. For currying, one obtains this inclusion:

In[2] := Map[equal[V, #] & , SubstTest[class, t, 
or[equal[0, y], member[APPLY[funpart[u], setpart[t]], v], not[member[setpart[t], w]],
{u -> CURRY, v -> map[x, map[y, z]], w -> map[cart[x, y], z]]]}

Out[2] := or[equal[0, y],
subclass[map[cart[x,y], z], image[inverse[CURRY], map[x, map[y, z]]]]] = True


One can convert this inclusion for inverse images to one for direct images by using this lemma:

In[4] := SubstTest[implies, subclass[u, image[inverse[funpart[t]], v]],
subclass[image[funpart[t], u], v], t -> CURRY] // Reverse

Out[4] := or[not[subclass[u, image[inverse[CURRY], v]]], subclass[image[CURRY, u], v]] = True
The result of doing so yields this inclusion:

\[
\begin{align*}
\text{Map[not, SubstTest[and, implies[p1, p2],
\quad \text{implies[p2, p3], not}[\text{implies[p1, p3]],
\quad \text{p1} \rightarrow \text{subclass[map[cart[x, y], z], image[inverse[CURRY], map[x, map[y, z]]]]},
\quad \text{p2} \rightarrow \text{subclass[map[cart[x, y], z], image[CURRY, map[cart[x, y], z]]]]],
\quad \text{p3} \rightarrow \text{subclass[image[CURRY, map[cart[x, y], z]], map[x, map[y, z]]]]} & \text{// Reverse} \end{align*}
\]

By considering the reverse process of uncurrying, an inclusion in the other direction is obtained:

\[
\begin{align*}
\text{Map[equal[V, #] \&, SubstTest[class, t,
\quad \text{or}[\text{equal[0, y]}, \text{member}[\text{APPLY[funpart[u], setpart[t]], v]},
\quad \text{not}[\text{member[setpart[t], w]]],
\quad \text{u} \rightarrow \text{inverse[CURRY], v} \rightarrow \text{map[cart[x, y], z]},
\quad \text{w} \rightarrow \text{map[x, map[y, z]]}]]} & \text{// Reverse} \end{align*}
\]

These two inclusions can be combined to yield a conditional equation:

\[
\begin{align*}
\text{SubstTest[and, implies[p, subclass[u, v]],
\quad \text{implies[p, subclass[v, u]],
\quad \text{p} \rightarrow \text{not}[\text{empty[y]]},
\quad \text{u} \rightarrow \text{image[CURRY, map[cart[x, y], z]],
\quad \text{v} \rightarrow \text{map[x, map[y, z]]}]} & \text{// Reverse} \end{align*}
\]

Comment. The condition that \( y \) be not empty cannot simply be omitted. When \( y = 0 \), the equation reduces to this statement:

\[
\begin{align*}
\text{equal[\text{image[CURRY, map[cart[x, y], z]], map[x, map[y, z]]}, \text{map}[x, \text{map}[y, z]]]} & \text{// Reverse} \end{align*}
\]

To get an unconditional equation, the equation must be modified. The condition that \( y \) be nonempty can be eliminated by slightly modifying the equation. This lemma will be needed:

\[
\begin{align*}
\text{SubstTest[implies, not[\text{equal[0, y]]],
\quad \text{equal[\text{image[CURRY, map[cart[t, y], z]], map[t, map[y, z]]},
\quad \text{t} \rightarrow \text{intersection[x, image[V, y]]} & \text{// Reverse} \end{align*}
\]

The following unconditional equation holds, and is made into a rewrite rule:
variable-free formulation

To obtain a variable-free formulation, two simplification rules are needed. The first one is fairly obvious:

\[
\text{Assoc}[\text{id}[\text{FUNS}], \text{composite}[\text{inverse}[\text{DORA}], \text{cross}[\text{Id}, \text{inverse}[\text{S}]]], \text{cross}[\text{x}, \text{Id}]]
\]

Based on the expectation that the following quantity will appear in the final result, a normalization result is sought:

\[
\text{composite}[\text{id}[\text{FUNS}], \text{intersection}[\text{composite}[\text{inverse}[\text{IMAGE}[\text{FIRST}]], \text{FIRST}, \text{FIRST}], \text{composite}[\text{inverse}[\text{IMAGE}[\text{SECOND}]], \text{inverse}[\text{S}], \text{MAP}, \text{cross}[\text{SECOND}, \text{Id}]]] = \\
\text{composite}[\text{inverse}[\text{E}], \text{MAP}, \text{cross}[\text{x}, \text{Id}]]
\]

With these two simplification rules in place, reification yields this variable-free result:

\[
\text{Map}[\text{composite}[\text{VERTSECT}[^{\#}], \text{id}[\text{cart}[\text{cart}[\text{V}, \text{V}], \text{V}]]] \&, \text{SubstTest}[\text{reify}, \text{x}, \text{image}[\text{z}, \text{map}[\text{cart}[\text{first}[\text{first}[\text{x}]], \text{second}[\text{first}[\text{x}]]], \text{second}[\text{x}]]], \text{z} \rightarrow \text{CURRY}]
\]

\[
\text{composite}[\text{IMAGE}[\text{CURRY}], \text{MAP}, \text{cross}[\text{CART}, \text{Id}]] = \\
\text{union}[\text{cart}[\text{cart}[\text{cart}[\text{V}, \text{set}[\text{0}]], \text{V}], \text{set}[\text{set}[\text{0}]]], \\
\text{composite}[\text{MAP}, \text{cross}[\text{Id}, \text{MAP}], \text{ASSOC}, \text{id}[\text{cart}[\text{cart}[\text{V}, \text{complement}[\text{set}[\text{0}]]], \text{V}]]]
\]

\[
\text{composite}[\text{IMAGE}[\text{CURRY}], \text{MAP}, \text{cross}[\text{CART}, \text{Id}]] = \\
\text{union}[\text{cart}[\text{cart}[\text{cart}[\text{V}, \text{set}[\text{0}]], \text{V}], \text{set}[\text{set}[\text{0}]]], \\
\text{composite}[\text{MAP}, \text{cross}[\text{Id}, \text{MAP}], \text{ASSOC}, \text{id}[\text{cart}[\text{cart}[\text{V}, \text{complement}[\text{set}[\text{0}]]], \text{V}]]]
\]
additional facts about inverse images

The following implication resembles one derived in an earlier section.

\[ \text{In[23]} := \text{SubstTest[implies, subclass[u, image[inverse[funpart[t]], v]], subclass[image[funpart[t], u], v], t \rightarrow \text{inverse[CURRY]}] // Reverse} \]

\[ \text{Out[23]} = \text{or[not[subclass[u, image[CURRY, v]]], subclass[image[inverse[CURRY], u], v]] = True} \]

\[ \text{In[24]} := \text{or[not[subclass[u_, image[CURRY, v_]]], subclass[image[inverse[CURRY], u_], v_]] := True} \]

Theorem.

\[ \text{In[25]} := \text{SubstTest[implies, member[APPLY[funpart[t], w], x], member[w, image[inverse[funpart[t]], x]], t \rightarrow \text{CURRY] // Reverse} \]

\[ \text{Out[25]} = \text{or[member[w, image[inverse[CURRY], x]], not[member[APPLY[CURRY, w], x]]] = True} \]

\[ \text{In[26]} := \text{or[member[w_, image[inverse[CURRY], x_]], not[member[APPLY[CURRY, w_], x_]]] := True} \]

Converse theorem.

\[ \text{In[27]} := \text{SubstTest[implies, member[w, image[inverse[funpart[t]], x]], member[APPLY[funpart[t], w], x], t \rightarrow \text{CURRY] // Reverse} \]

\[ \text{Out[27]} = \text{or[member[APPLY[CURRY, w], x], not[member[w, image[inverse[CURRY], x]]]] = True} \]

\[ \text{In[28]} := \text{or[member[APPLY[CURRY, w_], x_], not[member[w_, image[inverse[CURRY], x_]]]] := True} \]

The following somewhat trivial general fact is needed as a simplification rule.

\[ \text{In[29]} := \text{Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]], \{p1 \rightarrow member[APPLY[x, y], z], p2 \rightarrow member[y, domain[z]], p3 \rightarrow member[y, V]\]] // Reverse} \]

\[ \text{Out[29]} = \text{or[member[y, V], not[member[APPLY[x, y], z]]] = True} \]

\[ \text{In[30]} := \text{or[member[y_, V], not[member[APPLY[x_, y_], z_]]] := True} \]

Theorem.

\[ \text{In[31]} := \text{SubstTest[implies, member[w, image[u, x]], member[w, range[u]], u \rightarrow \text{inverse[CURRY]] // Reverse // MapNotNot} \]

\[ \text{Out[31]} = \text{or[not[member[w, image[inverse[CURRY], x]]], subclass[w, cart[cart[V, V], V]]] = True} \]

\[ \text{In[32]} := \text{or[not[member[w_, image[inverse[CURRY], x]]], subclass[w_, cart[cart[V, V], V]]] := True} \]

Corollary.
additional facts about images

Theorem.

In[35]:= SubstTest[implies, member[APPLY[funpart[t], w], x],
        member[w, image[inverse[funpart[t]], x]], t \[RightArrow] inverse[CURRY]] // Reverse

Out[35]= or[member[w, image[CURRY, x]], not[member[APPLY[inverse[CURRY], w], x]]] = True

In[36]:= or[member[w_, image[CURRY, x_]], not[member[APPLY[inverse[CURRY], w_], x_]]] := True

Converse theorem.

In[37]:= SubstTest[implies, member[APPLY[funpart[t], w], x],
        member[APPLY[funpart[t], w], x], t \[RightArrow] inverse[CURRY]] // Reverse

Out[37]= or[member[APPLY[inverse[CURRY], w], x], not[member[w, image[CURRY, x]]]] = True

In[38]:= or[member[APPLY[inverse[CURRY], w_], x_], not[member[w_, image[CURRY, x_]]]] := True

Theorem.

In[39]:= SubstTest[implies, and[member[w, u], subclass[u, v]],
        member[w, v], {u \[RightArrow] image[CURRY, x], v \[RightArrow] FUNS}] // Reverse // MapNotNot

Out[39]= or[FUNCTION[w], not[member[w, image[CURRY, x]]]] = True

In[40]:= or[FUNCTION[w_], not[member[w_, image[CURRY, x_]]]] := True

Corollary.

In[41]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]]],
        {p1 \[RightArrow] member[APPLY[inverse[CURRY], w], x],
         p2 \[RightArrow] member[w, image[CURRY, x]],
         p3 \[RightArrow] FUNCTION[w]]] // Reverse

Out[41]= or[FUNCTION[w], not[member[APPLY[inverse[CURRY], w], x]]] = True

In[42]:= or[FUNCTION[w_], not[member[APPLY[inverse[CURRY], w_], x_]]] := True

Lemma.
In[43]:= Map[and[#, member[0, range[U[image[CURRY, x]]]]]] & , SubstTest[subclass, 
    image[t, x], range[t, t -> composite[inverse[E], IMAGE[SECOND], CURRY]]]

Out[43]= member[0, range[U[image[CURRY, x]]]] = False

In[44]:= member[0, range[U[image[CURRY, x_]]]] := False

Theorem.

In[45]:= SubstTest[implies, and[member[w, u], subclass[u, v]], member[w, v], 
    {u -> image[CURRY, x], v -> P[complement[cart[V, set[0]]]]}] // Reverse // MapNotNot

Out[45]= or[not[member[0, range[w]]], not[member[0, range[w]]]] = True

In[46]:= or[not[member[0, range[w]]], not[member[0, range[w]]]] := True

Corollary.

In[47]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]], 
    {p1 -> member[APPLY[inverse[CURRY], w], x], 
    p2 -> member[w, image[CURRY, x]], p3 -> not[member[0, range[w]]]]}] // Reverse

Out[47]= or[not[member[0, range[w]]], not[member[0, range[w]]]] = True

In[48]:= or[not[member[0, range[w]]], not[member[0, range[w]]]] := True

Lemma.

In[49]:= SubstTest[subclass, image[t, x], range[t], 
    t -> composite[inverse[E], IMAGE[SECOND], CURRY]] // Reverse


In[50]:= subclass[U[range[U[image[CURRY, x_]]]]], cart[V, V] := True

Theorem.

In[51]:= SubstTest[implies, 
    and[member[w, u], subclass[u, v]], member[w, v], {u -> image[CURRY, x],
    v -> image[inverse[IMAGE[SECOND]], P[cart[V, v]]]} // Reverse // MapNotNot

Out[51]= or[not[member[w, image[CURRY, x]]], subclass[U[range[w]]], cart[V, V]] = True

In[52]:= or[not[member[w, image[CURRY, x]]], subclass[U[range[w]]], cart[V, V]] := True

Corollary.

In[53]:= Map[not, SubstTest[and, implies[p1, p2], not[implies[p1, p3]], 
    {p1 -> member[APPLY[inverse[CURRY], w], x], p2 -> member[w, image[CURRY, x]], 
    p3 -> subclass[U[range[w]]], cart[V, V]]}] // Reverse

Out[53]= or[not[member[APPLY[inverse[CURRY], w], x]], subclass[U[range[w]]], cart[V, V]] = True
In[54]:= or[not[member[APPLY[inverse[CURRY], w_], x_]],
    subclass[U[range[w_]], cart[V, V]]] := True