triple-application formula for CURRY

Johan G. F. Belinfante
2011 July 13

In[1] :=  SetDirectory["1:"]; << goedel.11jul08b

:Package Title: goedel.11jul08b 2011 July 8 at 10:00 p.m.
Loading takes about eleven minutes, half that time due to built-in pauses.
It is now: 2011 Jul 13 at 13:2
Loading Simplification Rules
TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3
weightlimit = 40
Loading completed.
It is now: 2011 Jul 13 at 13:12

summary

The basic idea of currying is to replace any function \( w \) with two variables with an equivalent function \( \text{APPLY}[\text{CURRY}, w] \) of a single variable, whose values in turn are functions of a single variable. The fundamental idea goes back to a lecture by Moses Schönfinkel (1920) written up for publication in 1924 by Heinrich Behmann, and popularized through the extensive work by Haskell B. Curry in combinatory logic.

An \text{APPLY} formula for curried mappings was derived 2006 November 11 in the posted notebook \text{cur-left.nb}. In this notebook some related results are derived, including a basic equation that involves triple application.

the basic theorem

The connection between a binary function and its curried counterpart is especially simple when the domain of the binary function is a non-empty cartesian product. The following theorem applies only to this special case.

Theorem. A basic triple application equation for curried mappings.
The following companion theorem involves the inverse process of uncurrying.

Theorem. A basic application equation about uncurrying.

In[5]:= Map[not, SubstTest[and, implies[p3, p4],
implies[and[p1, p2, p4], p5], implies[not[implies[and[p1, p2, p3], p6]],
{p1 → member[w, map[cart[x, y], z]], p2 → member[u, x], p3 → member[v, y],
p4 → not[empty[y]], p5 → equal[composite[u, LEFT[u]], APPLY[APPLY[CURRY, w], u]],
p6 → equal[APPLY[w, PAIR[u, v]], APPLY[APPLY[APPLY[CURRY, w], u], v]]]], // Reverse
Out[5]= or[equal[APPLY[w, PAIR[u, v]], APPLY[APPLY[APPLY[CURRY, w], u], v]],
not[member[u, x]], not[member[v, y]], not[member[w, map[cart[x, y], z]]]] := True

In[6]:= or[equal[APPLY[w_, PAIR[u_, v_]], APPLY[APPLY[APPLY[CURRY, w_], u_], v_]],
not[member[u_, x_]], not[member[v_, y_]],
not[member[w_, map[cart[x_, y_], z_]]]] := True

replacing PAIR with pair

In the GOEDEL program pair[u, v] is a primitive ordered pair, which agrees with the defined constructor PAIR[u, v] = A[[u] × [v]] when both u and v are sets.

In[7]:= A[cart[set[x], set[y]]
Out[7]= PAIR[x, y]

In[8]:= equal[PAIR[x, y], pair[x, y]]
Out[8]= and[member[x, V], member[y, V]]

To provide a measure of flexibility about which of the two ordered pairs is used, the counterparts of the results of the preceding section will be restated here with pair in place of PAIR.

Lemma. Replacing PAIR with pair in an APPLY expression.
Corollary. The basic theorem about curried function application with \texttt{pair} in place of \texttt{PAIR}.

\begin{verbatim}
In[11]:= Map[not, SubstTest[and, \texttt{implies[p1,p2]}, \texttt{implies[p1,p3]}],
\texttt{implies[and[p2, p3], p4]}, not[\texttt{implies[p1, p4]}],
\{p1 -> and[member[u, y], member[v, z]], p2 -> member[u, V], p3 -> member[v, V],
p4 -> equal[\texttt{APPLY[x, PAIR[u, v]]}], \texttt{APPLY[x, pair[u, v]]}]\} // Reverse
Out[11]= or[\texttt{equal[APPLY[x, pair[u, v]]}], \texttt{APPLY[x, PAIR[u, v]]}],
not[member[u, y]], not[member[v, z]]] = True
\end{verbatim}

Corollary. The basic theorem about un-curried function application with \texttt{pair} in place of \texttt{PAIR}.

\begin{verbatim}
In[12]:= Map[not, SubstTest[and, \texttt{implies[p1, p2]}, \texttt{implies[and[p1, p2], p3]}],
\texttt{not[implies[p1, p3]]},
\{p1 -> and[\texttt{member[u, x]}, member[v, y], member[w, map[\texttt{cart[x, y], z]]]},
p2 -> equal[\texttt{APPLY[w, PAIR[u, v]]}], \texttt{APPLY[\texttt{APPLY[\texttt{APPLY[\texttt{CURRY, w}, u], v]]}]},
p3 -> equal[\texttt{APPLY[w, pair[u, v]]}], \texttt{APPLY[\texttt{APPLY[\texttt{APPLY[\texttt{CURRY, w}, u], v]]}]}) // Reverse
Out[12]= or[\texttt{equal[APPLY[w, pair[u, v]]}], \texttt{APPLY[\texttt{APPLY[\texttt{CURRY, w}, u], v]]}],
not[\texttt{member[u, x]}], not[\texttt{member[v, y]}], not[\texttt{member[w, map[\texttt{cart[x, y], z]]}]]] = True
\end{verbatim}