 APPLY rules for MONOIDS

Johan G. F. Belinfante
2009 January 1

In[1]:= SetDirectory["l:"]; << goedel.09jan01a; << tools.m

:Package Title: goedel.09jan01a 2009 January 1 at 5:00 p.m.

It is now: 2009 Jan 1 at 17:6

Loading Simplification Rules

TOOLS.M Revised 2008 December 26

weightlimit = 40

---

summary

To facilitate the kind of reasoning about monoids that one finds in the literature, it is convenient to restate various properties of monoids using elements and APPLY. If \( x \) is a monoid operation, then \( \text{range}[x] \) is the underlying set on which this operation acts. If \( u \) and \( v \) are elements of \( \text{range}[x] \), then \( \text{APPLY}[x, \text{PAIR}[u, v]] \) is their product, usually denoted in the literature by \( u \cdot v \). In this notebook, the usual formulation of the definition (axioms) for a monoid are derived.

---

monoids as mappings

Theorem.

In[2]:= Map[not, SubstTest[and, implies[p1, p2],
   implies[p2, p3], implies[and[p1, p3], p4], not[implies[p1, p4]],
   {p1 -> member[x, MONOIDS], p2 -> equal[range[x], fix[domain[x]]], p3 -> equal[
     map[cartsq[range[x]], range[x]], map[cartsq[fix[domain[x]]], fix[domain[x]]]],
   p4 -> member[x, map[cart[range[x], range[x]], range[x]]]]]] // Reverse

Out[2]= or[member[x, map[cart[range[x], range[x]], range[x]]], not[member[x, MONOIDS]]] = True

In[3]:= or[member[x_, map[cart[range[x_], range[x_]], range[x_]]],
   not[member[x_, MONOIDS]]] := True

Corollary.

In[4]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
   {p1 -> member[x, MONOIDS], p2 -> member[x, map[cartsq[range[x]], range[x]]],
   p3 -> equal[cart[range[x], range[x]], domain[x]]]}] // Reverse

Out[4]= or[equal[cart[range[x], range[x]], domain[x]], not[member[x, MONOIDS]]] = True
In[5]:= or[equal[cart[range[x_], range[x_]], domain[x_]], not[member[x_, MONOIDS]]] := True

---

closure

Lemma.

In[6]:= SubstTest[implies, and[member[w, map[t, z]], member[s, t]],
   member[APPLY[w, s], z], {s -> PAIR[u, v], t -> cart[x, y]}] // Reverse

Out[6]= or[member[APPLY[w, PAIR[u, v]], z], not[member[u, x]],
   not[member[v, y]], not[member[w, map[cart[x, y], z]]]] := True

In[7]:= or[member[APPLY[w_, PAIR[u_, v_]], z_], not[member[u_, x_]],
   not[member[v_, y_]], not[member[w_, map[cart[u_, v_], z_]]]] := True

Theorem. (Closure property.) For any monoid \(x\), one has: \(u \in \text{range}[x] \& v \in \text{range}[x] \implies u \cdot v \in \text{range}[x]\).

In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
   not[implies[p1, p3]],
   {p1 -> and[member[x, MONOIDS], member[u, range[x]], member[v, range[x]]],
   p2 -> member[x, map[cart[range[x], range[x]], range[x]]],
   p3 -> member[APPLY[x, PAIR[u, v]], range[x]]}] // Reverse

Out[8]= or[member[APPLY[x, PAIR[u, v]], range[x]], not[member[u, range[x]]],
   not[member[v, range[x]]], not[member[x, MONOIDS]]] := True

In[9]:= or[member[APPLY[x_, PAIR[u_, v_]], range[x_]], not[member[u_, range[x_]]],
   not[member[v_, range[x_]]], not[member[x_, MONOIDS]]] := True

---

associativity

Theorem. Associative law: \((u \cdot v) \cdot w = u \cdot (v \cdot w)\). Note that this rewrite rule does not need explicit hypotheses that the three variables \(u\), \(v\) and \(w\) are members of \(\text{range}[x]\) because if this is not the case the equation reduces to \(V = V\).

In[10]:= Map[not, SubstTest[and, implies[p1, p2],
   implies[p2, p3], not[implies[p1, p3]],
   {p1 -> member[x, MONOIDS],
   p2 -> member[x, SEMIGPS], p3 -> equal[APPLY[x, PAIR[u, APPLY[x, PAIR[v, w]]]]],
   APPLY[x, PAIR[APPLY[x, PAIR[u, v]], w]]}] // Reverse

Out[10]= or[equal[APPLY[x, PAIR[u, APPLY[x, PAIR[v, w]]]]],
   APPLY[x, PAIR[APPLY[x, PAIR[u, v]], w]], not[member[x, MONOIDS]]] := True

In[11]:= or[equal[APPLY[x_, PAIR[APPLY[x_, PAIR[u_, v_]], w_]],
   APPLY[x_, PAIR[u_, APPLY[x_, PAIR[v_, w_]]]]], not[member[x_, MONOIDS]]] := True
left unital law

Lemma.

\[ \text{In[15]:=} \quad \text{Map}[\text{or}[\#, \text{equal}[z, \text{APPLY}[y, z]]], \text{not}[\text{member}[z, \text{range}[x]]]] \&,
\quad \text{SubstTest}[\text{implies}, \text{equal}[u, v], \text{equal}[\text{APPLY}[u, z], \text{APPLY}[v, z]],
\quad \{u \rightarrow \text{composite}[x, \text{LEFT}[y]], v \rightarrow \text{id}[\text{range}[x]]\}] \quad \text{// Reverse}
\]

\[ \text{Out[15]=} \quad \text{or}[\text{equal}[z, \text{APPLY}[x, \text{PAIR}[y, z]]],
\quad \text{not}[\text{equal}[\text{composite}[x, \text{LEFT}[y]], \text{id}[\text{range}[x]]], \text{not}[\text{member}[z, \text{range}[x]]]] \quad \text{=} \quad \text{True}
\]

\[ \text{In[16]=} \quad (\% \/. \{x \rightarrow x\_\_, y \rightarrow y\_, z \rightarrow z\_\}) \/. \text{Equal} \rightarrow \text{SetDelayed}
\]

Theorem.

\[ \text{In[20]:=} \quad \text{Map}[\text{not}, \text{SubstTest}[\text{and}, \text{implies}[p1, p3], \text{implies}[\text{and}[p2, p3], p4],
\quad \text{not}[\text{implies}[\text{and}[p1, p2], p4]], \{p1 \rightarrow \text{member}[x, \text{MONOIDS}],
\quad p2 \rightarrow \text{member}[w, \text{range}[x]], p3 \rightarrow \text{equal}[\text{composite}[x, \text{LEFT}[e[x]]], \text{id}[\text{range}[x]]],
\quad p4 \rightarrow \text{equal}[w, \text{APPLY}[x, \text{PAIR}[e[x], w]]]]}] \quad \text{// Reverse}
\]

\[ \text{Out[20]=} \quad \text{or}[\text{equal}[w, \text{APPLY}[x, \text{PAIR}[e[x], w]]],
\quad \text{not}[\text{member}[w, \text{range}[x]]], \text{not}[\text{member}[x, \text{MONOIDS]}}] \quad \text{=} \quad \text{True}
\]

\[ \text{In[21]=} \quad \text{or}[\text{equal}[w\_\_, \text{APPLY}[x\_\_, \text{PAIR}[e[x\_\_], w\_]\_]],
\quad \text{not}[\text{member}[w\_\_, \text{range}[x\_\_]]], \text{not}[\text{member}[x\_\_, \text{MONOIDS]}}] \quad \text{=} \quad \text{True}
\]

right unital law

The right unital law will be derived using \text{flip}. This approach requires two lemmas.

Lemma.

\[ \text{In[23]:=} \quad \text{SubstTest}[\text{implies}, \text{and}[\text{member}[w, \text{range}[t]]], \text{member}[t, \text{MONOIDS}],
\quad \text{equal}[w, \text{APPLY}[t, \text{PAIR}[e[t], w]]], t \rightarrow \text{flip}[x]] \quad \text{// Reverse}
\]

\[ \text{Out[23]=} \quad \text{or}[\text{equal}[w, \text{APPLY}[x, \text{PAIR}[w, e[x]]]], \text{not}[\text{member}[w, \text{image}[x, \text{cart}[V, V]]]],
\quad \text{not}[\text{member}[\text{composite}[x, \text{SWAP}], \text{MONOIDS]}}] \quad \text{=} \quad \text{True}
\]

\[ \text{In[24]=} \quad (\% \/. \{w \rightarrow w\_, x \rightarrow x\_\}) \/. \text{Equal} \rightarrow \text{SetDelayed}
\]

Lemma.

\[ \text{In[26]:=} \quad \text{SubstTest}[\text{implies}, \text{equal}[x, \text{binop}[t]],
\quad \text{equal}[\text{image}[x, \text{cart}[V, V]], \text{range}[x]], t \rightarrow x] \quad \text{// Reverse}
\]

\[ \text{Out[26]=} \quad \text{or}[\text{equal}[\text{image}[x, \text{cart}[V, V]], \text{range}[x]], \text{not}[\text{member}[x, \text{BINOPS]}}] \quad \text{=} \quad \text{True}
\]

\[ \text{In[27]=} \quad \text{or}[\text{equal}[\text{image}[x\_\_, \text{cart}[V, V]], \text{range}[x\_\_]], \text{not}[\text{member}[x\_\_, \text{BINOPS]}}] \quad \text{=} \quad \text{True}
\]
Theorem.

\texttt{In[29]} := \texttt{Map[not, SubstTest[and, implies[p1, p3], implies[p1, p5],}
\texttt{implies[p3, p4], implies[and[p2, p4, p5], p6], not[implies[and[p1, p2], p6]],}
\texttt{p1 \rightarrow member[x, MONOIDS], p2 \rightarrow member[w, range[x]], p3 \rightarrow member[x, BINOPS], p4 \rightarrow}
\texttt{equal[image[x, cart[V, V]], range[x]], p5 \rightarrow member[composite[x, SWAP], MONOIDS],}
\texttt{p6 \rightarrow equal[w, APPLY[x, PAIR[w, e[x]]]]] // Reverse}

\texttt{Out[29]} = \texttt{or[equal[w, APPLY[x, PAIR[w, e[x]]]],}
\texttt{not[member[w, range[x]]], not[member[x, MONOIDS]]] := True}

\texttt{In[31]} := \texttt{or[equal[APPLY[x_, PAIR[w_, e[x_]]], w_],}
\texttt{not[member[w_, range[x_]]], not[member[x_, MONOIDS]]] := True}

---

a corollary

Theorem.

\texttt{In[36]} := \texttt{Map[not, SubstTest[and, implies[p1, p2],}
\texttt{implies[and[p1, p2], p3], not[implies[p1, p3]], p1 \rightarrow member[x, MONOIDS],}
\texttt{p2 \rightarrow member[e[x], ids[x]], p3 \rightarrow member[e[x], range[x]]] // Reverse}

\texttt{Out[36]} = \texttt{or[member[e[x], range[x]], not[member[x, MONOIDS]]] := True}

\texttt{In[37]} := \texttt{or[member[e[x_], range[x_]], not[member[x_, MONOIDS]]] := True}

Corollary. If \( x \) is a monoid, then \( e[x] \cdot e[x] = e[x] \).

\texttt{In[39]} := \texttt{Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],}
\texttt{not[implies[p1, p3]], p1 \rightarrow member[x, MONOIDS], p2 \rightarrow member[e[x], range[x]],}
\texttt{p3 \rightarrow equal[APPLY[x, PAIR[e[x], e[x]]], e[x]]] // Reverse}

\texttt{Out[39]} = \texttt{or[equal[APPLY[x, PAIR[e[x], e[x]]], e[x]], not[member[x, MONOIDS]]] := True}

\texttt{In[41]} := \texttt{or[equal[APPLY[x_, PAIR[e[x_], e[x_]]], e[x_]], not[member[x_, MONOIDS]]] := True}