reify rules for function constructors

Johan G. F. Belinfante
2010 January 14

In[1]:= SetDirectory["l:"]; << goedel.10jan13a; << tools.m

:Package Title: goedel.10jan13a 2010 January 13 at 1:20 p.m.
It is now: 2010 Jan 14 at 14:28
Loading Simplification Rules
TOOLS.M Revised 2010 January 7
weightlimit = 40

summary

In this notebook the term function constructor is used for any unary class constructor \(f\) such that \(\text{FUNCTION}[f[x]]\) is true without restriction on the class \(x\). The GOEDEL program has numerous examples of function constructors:

In[2]:= Select[UnaryFunctors, FUNCTION[#[x]] &]

Out[2]= {ADJOIN, band, binop, cat, CHAR, clock, const, CORE, eval,
funpart, qp, HULL, id, IMAGE, inttimes, LEFT, list, modulo, oopart,
ordlist, perm, plus, quasigp, RC, RIGHT, semigp, times, unop, VERTSECT}

The reification of any unary class constructor \(f\) is the class \(\text{reify}[x, f[x]] = \text{class}[\text{pair}[x, y], \text{member}[y, f[x]]]\). It should be noted that the variables \(x\) and \(y\) here are both dummy variables. For each unary class constructor \(f\) there is a reify rule that allows one to express the reification of composite constructors of the form \(f[g[x]]\) in terms of the reification of the inner constructor \(g\). These reify rules are an important tool for eliminating set variables because they generally execute much faster than Kurt Gödel’s class rules. A typical example of such a reify rule is the following one for the identity functor constructor \(\text{id}\). (Recall that for any class \(x\), the identity function \(\text{id}[x]\) is the restriction of the global identity function \(\text{Id}\) to the class \(x\).)

In[3]:= reify[x, id[f[x]]]

Out[3]= composite[DUP, reify[x, f[x]]]

A schematic theorem is derived below which states that the rotated inverse of the reification of any function constructor is a function. Consider, for example, the constructor \(f = \text{ADJOIN}\). For any class \(x\) the class \(\text{ADJOIN}[x]\) is a function.

In[4]:= FUNCTION[ADJOIN[x]]

When $x$ is a set, $\text{ADJOIN}[x]$ is the total function which takes any set $y$ to its union with $x$. When $x$ is a proper class, on the other hand, this function is empty: $\text{ADJOIN}[x] = 0$. The rotated inverse of $\text{reify}[x, \text{ADJOIN}[x]]$ is the function $\text{CUP}$ that takes $\text{pair}[x, y]$ to $x \cup y$ for any sets $x, y$.

```
In[5]:= rotate[inverse[reify[x, ADJOIN[x]]]]
Out[5]= CUP
```

Another important example is the function constructor $\text{IMAGE}$. For any class $x$ the function $\text{IMAGE}[x]$ takes a set $y$ to $\text{image}[x, y]$, provided that the latter is a set. The rotated inverse of $\text{reify}[x, \text{IMAGE}[x]]$ is the function $\text{IMG}$ that takes $\text{pair}[x, y]$ to $\text{image}[x, y]$ for any sets $x, y$.

```
In[6]:= rotate[inverse[reify[x, IMAGE[x]]]]
Out[6]= IMG
```

In this notebook new simplification rules are derived for a few particular function constructors. The $\text{GOEDEL}$ program can already deal with many function constructors. The rotated inverse of the reification of several function constructors is a restriction of the rotated membership relation. A typical example is the function wrapper $f = \text{funpart}$.

```
In[7]:= rotate[inverse[reify[x, funpart[x]]]]
Out[7]= \text{composite}[\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{Id, SINGLETON}]]
```

This function is the function part of the rotated membership relation:

```
In[8]:= funpart[rotate[E]]
Out[8]= \text{composite}[\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{Id, SINGLETON}]]
```

It should be noted that the rotated membership relation itself is not a function:

```
In[9]:= FUNCTION[rotate[E]]
Out[9]= False
```

---

**a schematic theorem**

Lemma.

```
In[10]:= SubstTest[equal, rotate[u], rotate[v], \{u \rightarrow \text{composite}[\text{inverse}[x], \text{id}[\text{cart}[V, V]]], v \rightarrow \text{composite}[\text{rotate}[\text{composite}[\text{funpart}[\text{rotate}[\text{inverse}[x]]]], \text{SWAP}], \text{SWAP}]\}]
Out[10]= equal[\text{composite}[\text{inverse}[x], \text{id}[\text{cart}[V, V]]],
\text{composite}[\text{rotate}[\text{composite}[\text{funpart}[\text{rotate}[\text{inverse}[x]]]], \text{SWAP}], \text{SWAP}] =
\text{FUNCTION}[\text{rotate}[\text{inverse}[x]]]
```

```
In[11]:= equal[\text{composite}[\text{inverse}[x_], \text{id}[\text{cart}[V, V]]],
\text{composite}[\text{rotate}[\text{composite}[\text{funpart}[\text{rotate}[\text{inverse}[x_]]]], \text{SWAP}], \text{SWAP}] :=
\text{FUNCTION}[\text{rotate}[\text{inverse}[x]]]
```
The following schematic theorem becomes an ordinary theorem whenever \( f \) is replaced by any particular unary constructor. The key idea behind the derivation is that the logical statement \( \text{FUNCTION}[f[x]] = \text{True} \) is equivalent to an equation \( \text{funpart}[f[x]] = f[x] \) involving class constructors, and one can therefore apply \text{reify} \ to both sides of that equation.

Schematic Theorem. The rotated inverse of the reification of any function constructor is a function.

\[
\text{FUNCTION}[f[x]] \implies \text{FUNCTION}[(\text{rotate}[\text{inverse}[\text{reify}[x, f[x]]]]].
\]

Comment. One is not free to make substitutions on the formal variable \( x \) in the hypothesis. One is only free to make substitutions on the dummy variable \( f \).

\[
\text{In}[12]:= \text{SubstTest}[\text{implies}, \text{equal}[f[x], g[x]], \text{equal}[\text{rotate}[\text{inverse}[\text{reify}[x, f[x]]], \text{rotate}[\text{inverse}[\text{reify}[x, g[x]]], g[x] \rightarrow \text{funpart}[f[x]]]]] \quad \text{// Reverse}
\]

\[
\text{Out}[12]= \text{or}[[\text{FUNCTION}[\text{rotate}[\text{inverse}[\text{reify}[x, f[x]]]]], \text{not}[\text{FUNCTION}[f[x]]]] = \text{True}
\]

Caution. This schematic theorem cannot be made into a rewrite rule in the \text{GOEDEL} program because the statement \( \text{FUNCTION}[f[x]] \) must be true for every class \( x \), not just for some particular class. The \text{GOEDEL} program has no quantifiers for class variables, but only for set variables.

---

**CORE**

In this section it is shown that \( \text{rotate}[\text{inverse}[\text{reify}[x, \text{CORE}[x]]]] \) is the function \( \text{BIGCUP} \circ \text{CAP} \circ (\text{Id} \otimes \text{POWER}) \).

\[
\text{In}[13]:= \text{FUNCTION}[	ext{CORE}[x]]
\]

\[
\text{Out}[13]= \text{True}
\]

Theorem. A new rewrite rule.

\[
\text{In}[14]:= \text{Map}[\text{composite}[, \text{cross}[\text{Id}, x]], ]
\]

\[
\text{composite}[\text{IMAGE}[, \text{IMAGE}[\text{id}[\text{inverse}[\text{E}]]], \text{CART}, \text{cross}[\text{SINGLETON}, \text{Id}]]] \quad \text{// FastReifTriNormality}
\]

\[
\text{Out}[14]= \text{composite}[\text{IMAGE}[, \text{IMAGE}[\text{id}[\text{inverse}[\text{E}]]], \text{CART}, \text{cross}[\text{SINGLETON}, x]]] =
\]

\[
\text{composite}[\text{CAP}, \text{cross}[\text{Id}, x]]
\]

\[
\text{In}[15]:= \text{composite}[\text{IMAGE}[, \text{IMAGE}[\text{id}[\text{inverse}[\text{E}]]], \text{CART}, \text{cross}[\text{SINGLETON}, x_\_]]] :=
\]

\[
\text{composite}[\text{CAP}, \text{cross}[\text{Id}, x]]
\]

With this new rule in place, one finds:

\[
\text{In}[16]:= \text{rotate}[\text{inverse}[\text{reify}[x, \text{CORE}[x]]]]
\]

\[
\text{Out}[16]= \text{composite}[	ext{BIGCUP}, \text{CAP}, \text{cross}[\text{Id}, \text{POWER}]]
\]
HULL

For the function constructor HULL, the following simplification rule is needed.

Theorem.

```
In[17]:= rotate[composite[inverse[LAMBHULL], E]] // FastReifTriNormality
Out[17]= rotate[composite[inverse[LAMBHULL], E]] =
    composite[inverse[SINGLETON], IMG, cross[LAMBHULL, SINGLETON]]
```

```
In[18]:= rotate[composite[inverse[LAMBHULL], E]] :=
    composite[inverse[SINGLETON], IMG, cross[LAMBHULL, SINGLETON]]
```

oopart

For the constructor oopart, the rotated inverse reification involves the thin part of rotate[E].

Lemma.

```
In[19]:= composite[inverse[OOPART], inverse[FUNPART]] // DoubleInverse
Out[19]= composite[inverse[OOPART], inverse[FUNPART]] = inverse[OOPART]
```

```
In[20]:= composite[inverse[OOPART], inverse[FUNPART]] := inverse[OOPART]
```

Theorem.

```
In[21]:= composite[inverse[SINGLETON], IMG, cross[OOPART, SINGLETON]] // TripleRotate// Reverse
Out[21]= rotate[composite[inverse[OOPART], E]] =
    composite[inverse[SINGLETON], IMG, cross[OOPART, SINGLETON]]
```

```
In[22]:= rotate[composite[inverse[OOPART], E]] :=
    composite[inverse[SINGLETON], IMG, cross[OOPART, SINGLETON]]
```

clock[x]

The clock constructor is treated in this section along the same lines as the oopart constructor.

Lemma.

```
In[23]:= composite[FUNPART, CLOCK] // FastReifNormality
Out[23]= composite[FUNPART, CLOCK] = CLOCK
```
Lemma.

In[25]:= \text{composite}\{\text{inverse}[\text{CLOCK}], \text{inverse}[\text{FUNPART}]\} \equiv \text{inverse}[\text{CLOCK}]

Out[25]= \text{composite}\{\text{inverse}[\text{CLOCK}], \text{inverse}[\text{FUNPART}]\} \equiv \text{inverse}[\text{CLOCK}]

Theorem.

In[26]:= \text{composite}\{\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{CLOCK}, \text{SINGLETON}]\} \equiv \text{TripleRotate} \equiv \text{Reverse}

Out[26]= \text{rotate}\{\text{composite}\{\text{inverse}[\text{CLOCK}], \text{E}\}\} = \text{composite}\{\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{CLOCK}, \text{SINGLETON}]\}

In[27]:= \text{composite}\{\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{CLOCK}, \text{SINGLETON}]\} \equiv \text{TripleRotate} \equiv \text{Reverse}

Out[27]= \text{rotate}\{\text{composite}\{\text{inverse}[\text{CLOCK}], \text{E}\}\} = \text{composite}\{\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{CLOCK}, \text{SINGLETON}]\}

\text{VERTSECT}[x]

Two different formulas can be derived for the rotated inverse of the reification of the \text{VERTSECT} constructor. The only reason to prefer one over the other is resemblance to the expressions obtained for other function constructors.

Lemma.

In[29]:= \text{composite}\{\text{inverse}[\text{VS}], \text{inverse}[\text{FUNPART}]\} \equiv \text{DoubleInverse}

Out[29]= \text{composite}\{\text{inverse}[\text{VS}], \text{inverse}[\text{FUNPART}]\} \equiv \text{inverse}[\text{VS}]

In[30]:= \text{composite}\{\text{inverse}[\text{VS}], \text{inverse}[\text{FUNPART}]\} \equiv \text{inverse}[\text{VS}]

Theorem.

In[31]:= \text{composite}\{\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{VS}, \text{SINGLETON}]\} \equiv \text{TripleRotate} \equiv \text{Reverse}

Out[31]= \text{rotate}\{\text{composite}\{\text{inverse}[\text{VS}], \text{E}\}\} = \text{composite}\{\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{VS}, \text{SINGLETON}]\}

In[32]:= \text{rotate}\{\text{composite}\{\text{inverse}[\text{VS}], \text{E}\}\} = \text{composite}\{\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{VS}, \text{SINGLETON}]\}

Theorem. Another formula for the same function.

In[33]:= \text{rotate}\{\text{composite}\{\text{inverse}[\text{VS}], \text{E}\}\} \equiv \text{FastReifTriNormality} \equiv \text{Reverse}

Out[33]= \text{composite}\{\text{IMG}, \text{cross}[\text{Id}, \text{SINGLETON}], \text{Id}\{\text{composite}\{\text{inverse}[\text{E}], \text{IMAGE}[\text{FIRST}]\}\}\} \equiv \text{composite}\{\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{VS}, \text{SINGLETON}]\}
\textbf{In[34]} := \text{composite}[\text{IMG}, \text{cross}[\text{Id, SINGLETON}], \text{id}[\text{composite}[\text{inverse}[\text{E}], \text{IMAGE}[\text{FIRST}]]]] :=
\text{composite}[\text{inverse}[\text{SINGLETON}], \text{IMG}, \text{cross}[\text{VS, SINGLETON}]]