Tutorial #1. Goedel’s primitives.

Johan G. F. Belinfante
2001 January 11

```plaintext
<< goedel52.a3; << tests.m
:Package Title: GOEDEL52.A3   2001 January 2 at 8:20 a.m.
It is now:  2001 Jan 11 at 7:28
Loading Simplification Rules
TESTS.M     Revised 2000 December 30
weightlimit = 30

Context switch to 'Goedel' Private is needed for ReplaceTest
Just ignore the error message about Unterminated use of BeginPackage
Get::bebal : Unterminated uses of BeginPackage or Begin in << tests.m.
```

## GOEDEL’s primitives

The following examples explain the meaning of Goedel’s primitives. We begin with the universal class \( V \) and the membership relation \( E \).

```plaintext
class[x, True]

V

class[pair[x, y], member[x, y]]

E
```

There are four primitive unary constructors:

```plaintext
class[w, not[member[w, x]]]
complement[x]

class[u, exists[v, member[pair[u, v], x]]]

domain[x]

class[pair[pair[u, v], w], member[pair[pair[v, u], w], x]] == flip[x]

True
```
```
class[par[pair[u, v], w], member[par[pair[v, w], u], x]]
rotate[x]
```

There are three primitive binary constructors:

```
equal[class[w, or[equal[w, x], equal[w, y]]], pairset[x, y]] // assert
True
class[w, and[member[w, x], member[w, y]]]
intersection[x, y]
class[par[u, v], and[member[u, x], member[v, y]]]
cart[x, y]
```

In the **GOEDEL** program, the constructors **flip** and **pairset** have been de-emphasized. We rewrite **flip** as a composite with the function **SWAP**.

```
flip[x]
composite[x, SWAP]
```

The function **SWAP** interchanges the two members of an ordered pair:

```
class[par[par[u, v], par[x, y]], and[equal[u, y], equal[v, x]]]
SWAP
```

The constructor **pairset** has been largely replaced as a primitive by **singleton**.

```
pairset[x, x]
singleton[x]
```

The general connection between **pairset** and **singleton** is revealed by a **Normality** test:

```
pairset[x, y] // Normality
pairset[x, y] == union[singleton[x], singleton[y]]
```

Goedel also introduced **inverse** as a primitive, but it can be expressed in terms of his other primitives:

```
domain[flip[cart[x, V]]]
inverse[x]
```

The meaning of **inverse** is better understood from its conventional description:

```
class[par[u, v], member[par[v, u], x]]
inverse[x]
```
The rotate constructor.

The constructor rotate is not familiar to most mathematicians, and it could in fact be eliminated in favor of more familiar quantities. To do so, we introduce the constructor composite, which could be defined in terms of the primitives as:

\[
\text{domain}[\text{intersection}[\text{rotate}[, \text{cart}[x, V]], \text{flip}[\text{rotate}[\text{cart}[y, V]]]]]
\]

composite\(x, y\)

The meaning of composite is more clearly explained by its more conventional characterization:

\[
\text{class}[\text{pair}[u, w], \text{exists}[v, \text{and}[\text{member}[\text{pair}[u, v], y], \text{member}[\text{pair}[v, w], x]]]]
\]

composite\(x, y\)

The subclass relation \(S\) and the identity relation \(\text{Id}\) are definable in terms of the membership relation \(E\) using composite:

\[
\text{intersection}[\text{cart}[V, V], \text{complement}[\text{composite}[\text{complement}[E], \text{inverse}[E]]]]
\]

\(S\)

\[
\text{intersection}[S, \text{inverse}[S]]
\]

\(\text{Id}\)

More understandable descriptions of these relations are:

\[
\text{class}[\text{pair}[x, y], \text{subclass}[x, y]]
\]

\(S\)

\[
\text{class}[\text{pair}[x, y], \text{equal}[x, y]]
\]

\(\text{Id}\)

The function SWAP is definable in terms of Goedel’s primitives as

\[
\text{flip}[\text{Id}]
\]

SWAP

It is also convenient to introduce the functions FIRST and SECOND, which can be defined in terms of the primitives in several ways. One can for example use either one of the following as a starting point:

\[
\text{domain}[\text{rotate}[\text{Id}]]
\]

FIRST

\[
\text{rotate}[\text{cart}[\text{Id}, V]]
\]

SECOND

These functions are related to each other by \(\text{flip}\):
flip[\textsc{first}]
elbow{\textsc{second}}

flip[\textsc{second}]

\textsc{first}

The meaning of \textsc{first} and \textsc{second} is made transparent by the following descriptions:

\\begin{verbatim}
class[pair[pair[u, v], w], equal[w, u]]
\textsc{first}
class[pair[pair[u, v], w], equal[w, v]]
\textsc{second}
\\end{verbatim}

Goedel introduced both \texttt{flip} and \texttt{rotate} for the algorithm on which the \texttt{goedel} program is based. For this application one does not actually need to apply \texttt{rotate} to an arbitrary ternary relation; one mainly needs to apply \texttt{rotate} to ternary relations of the form \texttt{cart[x, V]}, where \texttt{x} is a binary relation.

\texttt{rotate[cart[x, V]]}

\texttt{composite[x, second]}

Using \textsc{first} and \textsc{second}, one can in principle eliminate \texttt{rotate} altogether:

\\begin{verbatim}
rotate[x] == composite[second, intersection[composite[inverse[x], first],
composite[inverse[first], second]]]
\\end{verbatim}

True

The discovery of this formula was one of the first applications of the \texttt{goedel} program. Recently we have found that \texttt{rotate} is actually quite useful, so we have stopped using the above formula to eliminate it.