# WHO DUNNIT?

**Our series of investigations into the use of LOGO's list processing facilities continues with a look at how to set up a simple database. We use the example of a murder investigation, in which a list of suspects is created and then analysed to ascertain who the murderer was.**

A terrible murder has been committed in a small community in the Ozark Mountains. Zachariah has been viciously attacked with an axe and killed. We know that Matthew and Joshua both own axes, James and Ebenezer own guns, and cousin Jane has a knife. Matthew and James both had blood on their hands when they were questioned by the local sheriff.

Our LOGO database of information about this crime will consist of a list of *facts* — each of which consists of a *relation*, together with one or more nouns. When represented in LOGO, one fact is [OWNS MATTHEW AXE] or, in English, 'Matthew owns an axe'. To represent the fact that James had blood on his hands, we use [BLOODY JAMES].

We begin our investigation with an empty database:

```
TO SETUP
   MAKE "DATABASE []
END
```

We then add facts to our database as we discover them (providing they are not already in the database). For example, we would input ADD [OWNS JANE KNIFE] using the following ADD procedure:

```
TO ADD :FACT
   IF NOT MEMBER? :FACT :DATABASE THEN
      MAKE "DATABASE FPUT :FACT :DATABASE
END
```

The database will eventually fill up:

```
[[BLOODY MATTHEW][BLOODY JAMES][KILLED
   ZACHARIAH AXE] [OWNS MATTHEW AXE]
   [OWNS JOSHUA AXE] [OWNS JAMES GUN]
   [OWNS EBENEZER GUN] [OWNS JANE KNIFE]]
```

To print out the database use SHOW. This can be followed by either "ALL, in which case the whole database will be printed, or by the name of a relation, in which case only the facts for that relation are printed. So, SHOW "OWNS will show us who owns what.

```
TO SHOW :S
   IF :S = "ALL THEN LIST.ALL :DATABASE
   LIST.REL :S :DATABASE
END
```

```
TO LIST.ALL :LIST
   IF EMPTY? :LIST THEN STOP
   PRINT FIRST :LIST
   LIST.ALL BUTFIRST :LIST
END
TO LIST.REL :S :LIST
   IF EMPTY? :LIST THEN STOP
   IF :S = FIRST FIRST :LIST THEN PRINT FIRST
      :LIST
   LIST.REL :S BUTFIRST :LIST
END
```

Now we must devise ways of querying the database. The simplest kind of query we might make of our database is to check whether a fact is known to be true. This we do with a procedure called DOES, which checks whether a fact is in the database. For example, DOES [OWNS JANE KNIFE] should give the answer YES.

```
TO DOES :FACT
   IF MEMBER? :FACT :DATABASE PRINT "YES
      ELSE PRINT "NO
END
```

It would be much more useful for our investigation into this terrible murder if we could ask questions such as 'Who owns an axe?'. The way we will deal with this is to use 'variables'. Any word whose first character is ? will be assumed to be a variable. We can then paraphrase the question as:

```
WHICH [OWNS ?SOMEONE AXE]
```

The reply to this will be a list of all possible values of the variable ?SOMEONE that are consistent with the information in the database.

```
[?SOMEONE MATTHEW]
[?SOMEONE JOSHUA]
NO (MORE) ANSWERS
```

We can have multiple variables. For example:

```
WHICH [KILLED ?MAN ?IMPLEMENT]
```

will give the answer:

```
[?MAN ZACHARIAH] [?IMPLEMENT AXE]
NO (MORE) ANSWERS
```

Let's consider the procedures that enable this analysis of the database, individually. WHICH passes the job over to FIND, indicating DATABASE as the source of facts.

```
TO WHICH :QUERY
   FIND :QUERY :DATABASE
   PRINT [NO (MORE) ANSWERS]
END
```

FIND sets up two global variables, VARS and ANS: VARS is used to hold each possible set of values of the variables in the question, and these are collected together in the list ANS.

```
TO FIND :QUERIES :DATA
   MAKE "VARS []
   MAKE "ANS []
   COMPARE :QUERY :DATA
   PRINTL :ANS
END
```