The RAND-ABEL® Programming Language

A new computer-programming language has been developed at The RAND Corporation to improve the interface between policy analysts and computer models and to improve the techniques for writing complex rule-based models to be executed quickly. The new language, RAND-ABEL®, was designed to meet four criteria:

Readability. It was important that policy analysts and other users of models be able to read enough of the computer program to understand the underlying assumptions and logic. When users, modelers, and programmers communicate by means of written documentation, there are usually misunderstandings about what a program or model is actually doing. Making it easier for users to inspect, alter, and enter a program’s parameters does not solve the problem; the user needs to understand what the program is doing with the data.

Writability. It was also important that analysts with general knowledge about programming be able to easily learn the essentials of RAND-ABEL, so they could make substantial changes in the rule-based models (albeit in consultation with professional programmers). Readability took precedence over writability when the tension between objectives was irresolvable.

Speed. Model execution had to be very fast, fast enough so that even large models with thousands of rules could execute in seconds—within a user’s cognitive cycle. This was considered an important step in tightening man-machine interactions.

Portability. The language had to be easily implementable on a number of popular computers and easily portable to new ones.

KEY CHARACTERISTICS

Two characteristics are especially important in enabling RAND-ABEL to meet its design criteria.

RAND-ABEL follows user thought patterns. Decisions regarding language design were based on the ways that RAND-ABEL’s intended users conceived of their models and on the ways in which it was most natural for them to express those models. Also, the language has been designed so that the inferences a reader draws about a statement’s meaning as he begins to read it are not contradicted by information presented later in the statement.

The language structure uses two dimensions. For twenty years, programming-language designers have made languages easy for machines to interpret. Thus, languages were generally one-dimensional. People, however, are very good at absorbing the meaning of two-dimensional constructs, as is clearly evidenced by the widespread popularity of tables in printed matter and of interactive spreadsheets in computer software. Fortunately, we are reaching the stage where machines are fast and cheap enough that we can optimize for people’s two-dimensional cognitive capabilities. RAND-ABEL takes some first steps down that path.

SPECIFIC INNOVATIONS

There are several innovations in RAND-ABEL that are especially important to the language’s effectiveness and that may be of interest to designers of other languages as well. Perhaps the most significant innovation implementable elsewhere is the Table statement. Analysts often organize their thoughts in terms of formatted tables or decision trees. RAND-ABEL allows the modeler to express such structures directly in the computer program (and not merely as data, as in some other

RAND-ABEL was developed for the RAND Strategy Assessment Center, which is sponsored by the Director of Net Assessment, U.S. Department of Defense as part of RAND’s National Defense Research Institute. For further information, see R-3274-NA, The RAND-ABEL® Programming Language: History, Rationale, and Design, Norman Z. Shapiro, H. Edward Hall, Robert H. Anderson, Mark LaCasse, The RAND Corporation, August 1985, or N-2367-NA, The RAND-ABEL® Programming Language: Reference Manual, Norman Z. Shapiro, H. Edward Hall, Robert H. Anderson, Mark LaCasse, The RAND Corporation, October 1985. Inquiries and comments may be directed to Dr. Shapiro at The RAND Corporation, 1700 Main Street, P.O. Box 2138, Santa Monica, California 90406-2138, (213) 393-0411.
languages). To illustrate this, the figure shows a conceptual decision tree (A) with three decision points yielding eight possible decisions. Beside the tree are two ways to express its logic in RAND-ABEL computer code. The first way (B) is understandable line by line—the words and relationships are all familiar—but the second way, involving a decision table (C), is far more comprehensible as a whole. Furthermore, it is easy to scan the table and grasp its global structure. The Table statement can be used for a variety of other purposes, such as allowing the user to relate independent variables with various combinations of “and” and “or.”

EFFECTIVENESS

RAND-ABEL has been successfully used by a multiteam research project with people from diverse backgrounds. It is very fast (requiring less than a millisecond per rule on a VAX 11-780 computer) and has proven quite readable and modifiable. It is the basis for a large system with 120,000 lines of code that combines features of rule-based modeling and traditional simulation.

If \( X = x_1 \)
Then
{ 
  If \( Y = y_1 \)
  Then
  { 
    If \( Z = z_1 \)
    Then Let \( D = d_1 \).
    Else \( [Z = z_2] \)
    Let \( D = d_2 \).
  }
  Else \( [Y = y_2] \)
  { 
    If \( Z = z_1 \)
    Then Let \( D = d_3 \).
    Else
    Let \( D = d_1 \).
  }
}
Else \( [X = x_2] \)
{ 
  If \( Y = y_1 \)
  Then
  { 
    If \( Z = z_1 \)
    Then Let \( D = d_4 \).
    Else \( [Z = z_2] \)
    Let \( D = d_5 \).
  }
  Else \( [Y = y_2] \)
  { 
    If \( Z = z_1 \)
    Then Let \( D = d_2 \).
    Else \( [Z = z_2] \)
    Let \( D = d_6 \).
  }
}

Example: If \( X = x_1 \), \( Y = y_1 \), and \( Z = z_1 \), the decision is \( d_1 \).

### Decision Table

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>y1</td>
<td>z1</td>
<td>d1</td>
</tr>
<tr>
<td>x1</td>
<td>y1</td>
<td>z2</td>
<td>d2</td>
</tr>
<tr>
<td>x1</td>
<td>y2</td>
<td>z1</td>
<td>d3</td>
</tr>
<tr>
<td>x1</td>
<td>y2</td>
<td>z2</td>
<td>d1</td>
</tr>
<tr>
<td>x2</td>
<td>y1</td>
<td>z1</td>
<td>d4</td>
</tr>
<tr>
<td>x2</td>
<td>y1</td>
<td>z2</td>
<td>d5</td>
</tr>
<tr>
<td>x2</td>
<td>y2</td>
<td>z1</td>
<td>d2</td>
</tr>
<tr>
<td>x2</td>
<td>y2</td>
<td>z2</td>
<td>d6</td>
</tr>
</tbody>
</table>