SIMULATION LANGUAGE PALM

G. JONIN, J. SEDOL
State University, Riga, U. R. S. S.

ABSTRACT

The language PALM is intended for simulation of public service systems, of teletraffic systems in particular. The software for the language is implemented on computers, compatible with IBM/360. The formalized system of concepts, named a mathematical model, gives a possibility for simple and clear description of the system to be simulated and the development of the simulation program.

Basic information is given on the system of the PALM languages, incorporating PALM, a mathematical model is described and the principles and possibilities of PALM are explained. The use of PALM is illustrated.

1. INTRODUCTION

The language PALM (for Russian Программирование — Programming of Simulation Algorithms) was developed at the Latvian State University in Riga. The language is intended for Simulation of public service systems, of teletraffic systems in particular. PALM constitutes part of a system of programming languages called SPALM (System of Programming Algorithms and Models). The software for SPALM is implemented on computers, compatible with IBM/360.

According to the classification [1,2], PALM represents a universal simulation language of the transaction flow type and is based on simple general-purpose language PAL. PALM differs from other simulation languages such as GPSS[3], SIMSCRIPT[4], SIMULA[5], TETRASIM[6] in the following features:

- PALM is of a simple structure and thus easier to master and employ;
- a formalized system of concepts has been provided, named a mathematical model [7], permitting a simple and clear transition from the system to be simulated to a program in PALM;
- algorithm description in PALM is close to conventional mathematical symbolism;
- the amount of software required is comparatively small and ensures high efficiency of the simulation program;
- a mechanism is provided for adding new tools of the language; this makes PALM easily adaptable to simulation tasks of different kinds.

Experience has proved the efficiency of PALM in simulating practical tasks of communication system design in the U.S.S.R.

2. BASIC INFORMATION ON SPA LM

SPALM is a combination of different-purpose languages at different levels for describing algorithms and for programming on computers. Structurally these languages are based on identical principles and the possibility of simultaneous usage is provided for. The major SPALM languages are:

- AL (from Algorithms) — an extended version of the language of mathematical formulas for describing numerical algorithms;
- PALM (from Programming of Algorithms) — a general-purpose programming language;
- PALM — an extended version of PAL for simulation purposes;
- BE (from Basic language for ES computers) — a low-level language analogous to the ASSEMBLER, compatible with PAL and PALM for supplying software.

3. ALGORITHM DESCRIPTION IN AL

AL employs the conventional mathematical denotation for variables, expressions and arithmetical operations (except multiplication and division, which are denoted by * and /, respectively). Two special signs are introduced: "1" for headings and "#" for control transfer. The algorithm is described in blank-separated sentences. The sentences fall into headings, i.e. they contain the "1" sign, and commands (the rest). The headings are subdivided into block headings, e.g. A, and labels, e.g. 2.

The commands are divided into assignment commands, e.g. a=b+c, and control ones, which, in their turn, fall into block addresses, e.g. A, and conditioned transitions, e.g. x<y/3 #5. The examples of control operations contain: address to block A; transition to label "2"; transition to label "5" when x<y/3 holds.

Below is given an algorithm for solving the polynomial \( y = a_0 x^n + a_1 x^{n-1} + \ldots + a_n \), described in AL and containing one block P.

\[
\begin{align*}
\rho: & \quad i=1 \\
1: & \quad y = a_0 \\
2: & \quad y = y + x a_i \\
3: & \quad i = i + 1 \\
4: & \quad i = n + 1
\end{align*}
\]

Commands containing one expression only are abbreviated assignment commands; the result is assigned to the leftmost variable in the expression.

Fig. 1. Block diagram elements.
The description of an algorithm in AL can be illustrated as a block diagram. Fig. 1 shows the elements of a block diagram. The block diagram of algorithm P is given in Fig. 2.

![Algorithm P](image)

4. THE MATHEMATICAL MODEL

The mathematical model represents a generalized concept of an algorithm and describes the work of systems of discrete events, teletraffic systems in particular. The work of an ordinary algorithm can best be visualized as movement through a block diagram of some object (usually referred to as control), which performs the commands. There can be several such operating objects in the mathematical model and they are referred to as calls. Apart from the commands in AL, calls perform certain specialized ones, namely: generation (breaking up a call into two independent calls), delay (stoppage of call operation for random time with an assigned distribution), wait (stoppage of operation for a time depending on the operation of the other calls) and activation (release from waiting for another call). The above operations are called the major operations and can be combined into derived ones, for example, generation and delay combine into a call source. Fig. 3 is a graphic representation of the major operations and Fig. 4 - a full and a simplified picture of the source.

![Major Operations](image)

![Source](image)

5. THE PAL PROGRAMMING LANGUAGE

PAL is designed for describing algorithms meant to be computer programs. PAL is an analogue of AL differing mainly in that:

- it has an alphabet (a set of symbols, including Latin capitals, figures, signs, \{ \}, <, >, +, - /, \# and blanks. Braces \{ \} are no part of the alphabet and are used to separate the language structures from description text.

- variables are denoted by identifiers in the form of letters only. Identifiers can also stand for arrays of successive variables. If, for instance, LQ is an array of three variables, the latter are designated LQ, LQ1 and LQ2. The indexes are bracketed, e.g. A[1].

- two types of variables - integer and real - are involved.

- besides the headings and the commands a third kind of sentence is introduced called declarations. The declaration defines the variables, assigns their type, mutual arrangement and the initial and/or the initial values. Declarations are usually written in a special
section at the beginning of the program. The declaration section is headed \([\cdot] \) and continues up to the next heading. All the variables and arrays involved in operation are to be declared.
- There are control commands, called addresses, to standard programs \((SP)\). SP, for instance, effects the output (print-out) of the results. The printout command is of the form \(\{#D#\} \), where \(a \) is the list of the print-out information. Here is the above-considered algorithm P as described in PAL:

\[
\begin{align*}
: & \quad N=2 \quad A[N+1]\times1.5 \times 5 \\
:\quad & \quad P: \quad I=1 \quad y=ax \\
1: & \quad y=x^a[A[i]] \quad I=1 \quad I=x+1 \quad #P#Y
\end{align*}
\]

The declaration of array \(A\) and variables \(N, x, y, I\) gives the initial values, which make up the control example, e.g. find

\[y=2x^1.5+3 \quad \text{if} \quad x>5.\]

The presence of the symbol \([\cdot] \) in the declaration indicates the type \(X\) and its absence means the I type. For the rest, the program repeats the algorithm description in \(AL\) (with the addition only of an output command for the variable \(y\) value). In such a form, the program can be fed into computer, which prints out the result \(30.50000\).

6. THE BASIS FOR PALM

The simulation language PALM is obtained by extending PAL, much the same as the mathematical model is built by making additions to \(AL\). Added to PAL are tools for performing the major operations of the model (including source). The tools assume the form of addresses to the SP:

\[\#A#S \quad - \quad \text{source,} \quad \#B#S \quad - \quad \text{delay,} \quad \#C#S \quad - \quad \text{wait,} \quad \#D#S \quad - \quad \text{activation,} \quad \#G#S \quad - \quad \text{generation.}\]

Where \$ stands for added information: flow parameter or the delay time, the queue number, the label for a generated call. Absence of \$ implies \$+1.

In order to make the model work, the translator adds a universal simulation program \((USP)\) to the program written in \(PAL\). Simulation begins by addressing a \(\{#P#P\} \) command to the USP, where \(P\) is the number of parameters assigned to each call. In its turn, the USP addresses the standard blocks of the program written in \(PAL\):

\[
\begin{align*}
\text{MD} & \quad - \quad \text{simulation block corresponding to the operation diagram,} \\
\text{SA} & \quad - \quad \text{build up of the initial state of the model,} \\
\text{RZ} & \quad - \quad \text{processing and output of results,} \\
\text{SB} & \quad - \quad \text{preparation of arrays for collecting information,} \\
\text{NF} & \quad - \quad \text{collection of information.}
\end{align*}
\]

All the enumerated blocks need not be present.

Interaction of the program in \(PAL\) and the USP is effected by standard variables making up three arrays \(Q, W, \text{ and } T\), four variables in each, \(W\) being the call count and \(T\) the current time. The declaration \(\{#Q#W\} \) must be present in the declaration section.

Let this be illustrated by a simulation program for a v-line full-access scheme with queue, written in \(PAL\):

\[
\begin{align*}
: & \quad V=3 \quad L=2 \quad I=04 \quad W=4 \quad T=4* \\
M: & \quad \#M#B \\
MD: & \quad A#L \quad W=1 \quad I=x+1 \quad B# \#C# \\
1: & \quad I=1 \quad B# \quad I=1 \quad #D# \\
\end{align*}
\]

Here \(V\) stands for the number of lines, \(L\) for the incoming load, and \(I\) for the number of engaged lines. The program incorporates a control example \(V=3, L=2\). The program does not contain any directions as to when simulation is to be ended. In such a case the USP automatically performs five series of calls, 2000 each.

7. STANDARD STATISTICAL INFORMATION

In the program example considered above there is no operation of information collection. In such a case standard statistical information \((SSI)\) can be obtained by the programmer’s direction. The SSI contains data on the sources, delays and queues: the realized flow intensity, the average delay time and the average number of calls on it, the average waiting time and the average queue content, etc. The SSI can also be obtained by way of supplementary information in cases when the program itself compiles and processes the information.

The basic data concerning the work of the model, such as the series number, the number of calls and the model time, are always printed out without a special demand.

8. THE ADDITIONAL ADVANTAGES OF PALM

The above-considered possibilities of \(PAL\) are supplemented by a number of others. Sources are available of pseudo-random numbers with uniform, exponential and normal distribution. Also available is a SP for realizing the distribution function assigned according to a table. The USP contains a subprogram for building histograms. The latter are printed out as numbers with graphic representations attached beside.

Tools are provided for realizing a disruptible delay, an operation combining elements of both delay and wait.

Provided is also a SP for calculating confidence intervals by statistically processing the simulation results series by series.

9. DEBUGGING

The simulation program can be debugged under the debugging regime of \(PAL\) by \(PAL\). Under the debugging regime, the observer’s commands are inserted at particular program points, which print out the values of several specified variables assigned number of times.

\(PAL\) supplies the tools for the initial and the final debugging. Initial debugging consists in having at the beginning of simulation, some time after each event, information printed out on the state of the model. Final debugging clears the model by switching off the source and the generation. The information printed out before and after each clearing allows to judge
about the correctness of the program.

All the debugging tools are engaged by adding control cards to the program without changing the program itself.

10. BRIEF INFORMATION ON SOFTWARE

The software for SPALM is a unique set of programs of about 44 K (kilobytes) size at present (in 1979). The major part of the program is written in BE language.

The software comprises a translator, common for BE, PAL and PALM and an addenda library. The addenda library contains algorithms of elementary functions and standard programs, including USP. At translation, the user's program is automatically supplemented with the required addenda from the addenda library.

The translator revises the program one time only and works 30-50 times faster than the ASSEMBLER and PL/I translators do.

11. EXTENSION POSSIBILITIES FOR SPALM

The software for SPALM has been designed with a view of continuous improvement of the system. Most tools of adding to the programming possibilities are obtained by extending the SP set. The SP's are combined in complexes referred to as addenda and these represent the constituent parts in program compilation. The most frequently used addenda are included in the addenda library, the others are on the punchcards and are added by hand.

Special-purpose variants can be made of PALM by making up corresponding SP's. Thus an addendum has been developed for simulation of priority systems.

An SP can be written both in BE (thus obtaining more rational programs) and in PAL (thus obtaining easier programs), as well as by combining the two languages.

For realizing additions, which are not connected with the SP, possibilities have been provided for to make comparatively easy changes in the translator.

12. REFERENCES


