AN APPROACH FOR PROGRAM INVESTIGATION OF COMPUTER PROCESSES PRESENTED BY MARKOV MODELS

Radi Romansky *

Technical University of Sofia
Bulgaria

* Corresponding Author, e-mail: rrom@tu-sofia.bg

Abstract: The classical organization of computer processing obeys the hierarchical model of two levels (macro and micro), and the processes of each of them can be described by a functional algorithm. This permits to make formalization by using different mathematical platforms, but the graph theory is very suitable for this purpose. In this reason, the article presents an opportunity for program investigation of computer processes that are modelled by means of Markov chains based on designed program module by using APL2 language in the TryAPL2 operating environment. Presented module is a part of constructed program environment for investigation of computer processes in micro and macro levels.

Key words: APL2 programming, computing formalization, processes modelling, Markov chains, program experiments.

1. INTRODUCTION

The contemporary digital world is built on the basis of continuous communications between users and remote access to information resources, making extensive use of new intelligent [1] and cloud [2] technologies. Of course, this has its advantages, but it requires a careful analysis of the possible dangers, as a discussion of security problems in running an educational organization is presented in [3]. It is known that any computer processing requires strict organization in the provision of computing resources to active applications, which is a basic function of system software. To achieve an effective level of management, it is necessary before a given realization to investigate the processes with a clearly defined goal and an adequate formalization based on abstractions, as one example in this direction is the demonstration of different levels of abstractions presented in [4].

The discussed computational hierarchy is used to illustrate a more general set of criteria in solving the problem posed by applying the capabilities of finite state machines. In principle, the theory of finite state machines (finite automata) is

applicable in the study of computer processes, because during execution each task passes through different states depending on the functional organization. An example for this is the investigation presented in [5] where a method for the synthesis of programmable logic integrated circuits is presented and an algorithm for splitting internal states for the synthesis of fast finite state machines is described. Another research is presented in [6] where software testing is discussed as one of the key stages in the development of a microcontroller-based digital device.

The classical organization of computer processing obeys the hierarchical model of two levels – high (macro-level) and low (micro-level), and the processes of each of them can be described by a functional algorithm. It organizes computations and manages the transitions between the states they enter, with each execution of a high-level application activating a sequence of specified low-level actions (micro-operations). This allows to make a preliminary formalization consistent with Turing Computational Theory [7] by using suitable approach. One possibility is the application of graph theory [8, 9] for the formal description of a given computer process through a directed graph with transitions between states (State Transition Network – STN) and the study of possible implementations based on defined paths "from beginning to end". In addition, a study of temporal and structural parameters of computer processing, more of which have a stochastic character, can be carried out. A procedure for organizing the model study is presented in [10]. Basically, mathematical modelling is widely used in various fields, such as the study of power loss minimization presented in [11]. Modelling as an approach has different directions, as in [12] a view is presented for the application of the deterministic variant when conducting a scientific study of the processes in a heterogeneous e-learning environment.

The purpose of the article is to present an additional opportunity for investigation of computer processes that are modelled by means of Markov chains. An author's program module for Markov model research and calculation of basic parameters is proposed, developed using the APL2 programming language in the TryAPL2 operating environment. The research can be considered as a continuation of the articles [13] and [14], because the proposed module, together with the others presented in the specified articles, build a common program space developed in the specified operating environment.

The article has the following structure – next section presents a brief discussion of selected features of used program environment, section 3 includes an author’s point of view for preliminary formal description of the computer processing organization, and section 4 deals with proposed new program module for execution of analytic Markovian models.

2. BRIEF INTRODUCTION TO THE PROGRAM ENVIRONMENT TryAPL2

TryAPL2 is a programming environment providing the necessary tools to create, maintain and implement applications in the APL2 language. This is a parallel program language for processing arrays with three basic components – definition of virtual

vector (array) for storing items; definition of a relative storage address where the array could be stored; and definition of the virtual distance between subsequent elements [15]. It is the result of the evolution of the APL platform developed for IBM machines several decades ago, the historical development of which is described in [16]. The comment is that there has been a significant evolution of the operating systems and platforms used by APL applications, including the addition of new parallelism capabilities. The article concludes that "the evolution of APL is far from over."

The APL2 language version allows the development of applications based on the principles of graph theory and operation research, including its use for programmatic representation of models of computer objects and processes [13]. The developed program modules represent additional functions to the TryAPL2 operating environment, allowing increasing its descriptive capabilities. To conduct experiments with the created modules, the OR (Operations Research) system workspace is used, for which the functions from Table 1 are supported.

Table 1. Main functions supported by OR

Function	***Purpose***
SETUP *matrix*	Defines a specific STN described by a numerical matrix of connections. Defines: *NODES* – list of nodes; *SIZE* – number of nodes; *NETWORK* – copy of matrix; *CM* – matrix of affiliation.
PATHSFROM *node*	Calculation of all paths in the STN form specified *node* to the final node. If STN has 1 start and 1 end node, all paths are calculated by ρPATH←–PATHSFROM 1 ↑PATH – Displays the first path in the structure PATH.
ARCS *path*	Determines the weight of each arc of the *path*. For example: ARCS ↑PATH – displays information about the first of the paths stored in PATH; +/ARCS ↑PATH – determines the length of this path (sum of weights); x/ARCS ↑PATH – calculates the multiplication of the probabilities between nodes in the first path +/⍣⍣ARCS⍣⍣PATH – determines the lengths of all paths; x/⍣⍣ARCS⍣⍣PATH – determines the multiplication of all probabilities in each path in STN.
VALUE *path*	Analogous to the expression +/ARCS *path* ,where *path* can contain one (↑PATH), or all (⍣⍣PATH) possible paths.
MIN←⌈/V	Calculates the minimal probability multiplication for a path
MAX←⌊/V	Calculates the maximal probability multiplication for a path
(V=⌈/V)/PATH	Determines the path in the STN with maximum result of the probabilities multiplication

To conduct an initial quantitative analysis of the created formal description, the standard functions from Table 2 can be used. When each of them is executed, an answer related to the specific study is returned.

Table 2. Standard functions in OR for SETUP MATRIX

SETUP MATRIX	***Comment***
NODES	Determines the numbers of the nodes in the STN
SIZE	Displays dimensions of the connection matrix
NETWORK	Displays defined STN
CM	Displays the connection matrix defined by MATRIX

3. FORMALIZATION OF THE PRELIMINARY STATE FOR COMPUTING ORGANIZATION

Any information processing in a general sense is the implementation of a certain function $f:D\rightarrow R$ to convert elements of a set of input data D into output results forming elements of a set R. It is known that the implementation of these actions in a computer environment is carried out by hardware-program implementation of the functional transformation f^* and binary representation (encoding) of the elements of the two sets, sets D^* and R^*, respectively. Thus, the formal representation of computer processing can be represented in a generalized sense as $f^*:D^*\rightarrow R^*$ with a graphic illustration in Figure 1.

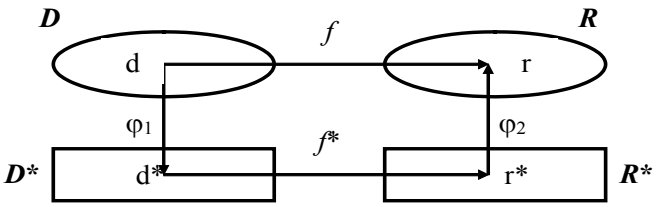


Figure 1. Formal description of the relevance between the information and computer levels

Functions φ_1 and φ_2 perform information conversation and the generalized formal model of the computer processing can be presented as $r=f(d)=\varphi_2\{f^*[\varphi_1(d)]\}=\varphi_2\{f^*[d^*]\}$, where $d^*=\varphi_1(d)$, $r^*=f^*(d^*)$ and $r=\varphi_2(r^*)$. The realization of this functional model takes place in a heterogeneous environment composed of hardware and software components, building a computer system with a specific structure, purpose and behavior. In this reason, the environment for the implementation of computer processing (a set of subsystems), as well as the processes of processing information elements, can be considered as an object for model investigation.

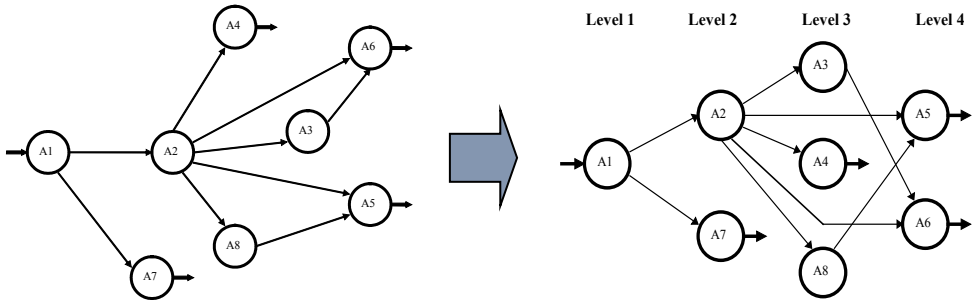
One way to formalize computer processing is by using a directed graph (STN), which is applicable both at the macro level [10] and at the low microprogram level [13]. The nodes of the graph represent the individual processing actions (programs, instructions, microinstructions), and the arcs represent the possible transitions between them. The study is associated with determining the possible paths in STN with calculation of their lengths and the probabilities of their realization under given initial conditions.

In the deterministic approach for investigating computer processing, a Boolean matrix of connections $L=\{l_{ij}\}$ is defined, where $l_{ij}=1$ (if there is a transaction) and $l_{ij}=0$ (if there is no transaction) [10]. In the investigation of high-level processes, one task is to determine the reachability of a given final task (algorithm) from one or more initial ones. For this purpose, a system of algebraic equations describing the presence of arcs a_{ij} between tasks A_i and A_j is compiled:

$$A_j = \left\{ \sum_{i=1}^{n} a_{ij} . A_i \right\}; j = 1,2,...,n$$

Solving these equations allows to construct all directed paths as well as to determine the equivalent paths that lead to the same event in computer processing. For this purpose, it is necessary to transform the complete processing algorithm $G(\alpha A)$, represented by an initial Graph Scheme of Algorithm (GSA), into an Ordered Graph Scheme of Algorithm (OGSA) by numbering the nodes of the graph based on the condition: If $\exists path(A_i \rightarrow A_j\} \Rightarrow A_i < A_j$; for $\forall A_i$, A_j $(i \neq j)$.

An example of the conversion of GSA into OGSA, which will be used in the next section, is presented in Figure 2 [10]. The initial situation GSA (Fig. 2a) describes a set of algorithms (applications) $\{A_1,\ldots, A_8\}$ with communication links for transmitting intermediate results described in the matrix from Table 3.



(a) GSA *(b) ordered GSA*

Figure 2. Example of conversation of GSA to OGSA

Table 3. Matrix of connections between nodes

L	A1	A2	A3	A4	A5	A6	A7	A8
A1		1					1	
A2			1	1	1	1		1
A3						1		
A4								
A5								
A6								
A7								
A8					1			

After applying the ordering procedure, 4 consecutive levels are formed, each including applications with no informational dependency between them. A developed APL2-module "GSA" [10] was used to form OGSA. The procedure performs the following steps: ✓ defining the output Boolean matrix L to describe the GSA; ✓ formation of the transposed matrix LT; ✓ calculation of vectors of connections between elements of LT used to distribute graph nodes (tasks) by layers; ✓ formation of matrix to define OGSA. An additional module "PATHS" allows automated calculation of the number of all possible paths and their lengths representing the possible implementations of information processes in computer processing.

4. PROGRAM MODULE FOR INVESTIGATION MARKOVIAN MODELS

If the elements $\{a_{ij}\}$ in the connection matrix L are replaced by the transition probabilities between the algorithms $p_{ij} = P(A_1 \rightarrow A_j)$ a matrix of transition probabilities describing computer processing as a stochastic Markov chain will be formed. This allows using this approach for investigation developed processes. For this purpose, a program module *"MARKOV"* (Figure 3) for automated execution of defined analytical markovian model has been developed. During the algorithmization, it was assumed that the studied GSA has one initial node, which determines a vector of initial probabilities P_0={1, 0, …, 0}. The assumption is justified, since in the presence of several such it is possible to introduce a common passive initial node. An analogous solution can also be applied to merging the set of final tasks.

The module fulfills the requirement of Markov chain theory that at a given time (step k) the computer processing is in only one of the possible states <S(0), S(1),…, $S(k)$>, starting from an initial state $S(0)=S(k=0)$. Each transition $A_i \rightarrow A_j$ is determined by the transition probabilities $p_{ij}(k)=P[S(k)=A_j \,/\, S(k\text{-}1)=A_i]$ for the successive steps k=1,2,…, as for each step is defined the probability $p_j(k)=P[S(k)=A_j]$ that the system falls into state $S(k)=A_j$. Essentially, these are conditional probabilities that are determined by the full probability formula:

$$p_j(k) = \sum_{i=1}^{n} p_i(k-1).p_{ij} \quad ; \; j = 1,2,...8.$$

A defined vector of initial probabilities $P_0=P(k=0)$ and a matrix of transition probabilities $P=\{p_{ij}\}$ allow to determine the vectors $P(k)$ of the distributed probabilities for the states for each step k=1,2,3,…, as well as to calculate the final probabilities $P(k\rightarrow\infty)$ of falling into a certain state by $\lim_{k\to\infty} p_{ij}(k) = p_j$.

An exemplary execution of module "MARKOV" for investigation of the above discussed GSA (fig. 2) defined as Markov chain is presented in Figure 4.

The model is defined by the parameters:

a) Number of the states N defined in the set $S = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\}$;

b) Matrix P[I,J] of transition probabilities $P = \{p_{ij} / i,j = 1\div8\}$ – Table 4;

c) Vector PO[1÷N] of initial probabilities $P_0 = \{1, 0, 0, 0, 0, 0, 0, 0\}$.

Table 4. Matrix of transition probabilities

P	A1	A2	A3	A4	A5	A6	A7	A8
A1		0.7					0.3	
A2			0.4	0.1	0.2	0.1		0.2
A3						1		
A4								
A5								
A6								
A7								
A8					1			

```
[0]    MARKOV
[1]    '1. NUMBER OF STATES N:'
[2]    N←□
[3]    P←(N N)ρ0
[4]    '2. MATRIX OF PROBABILITIES P(I,J):'
[5]    I←1
[6]    ETI:J←1
[7]    ETJ: 'P(',I,',',J,')='
[8]    P[I;J]←□
[9]    →(N≥J←J+1)/ETJ
[10]   →(N≥I←I+1)/ETI
[11]   '3. INITIAL VECTOR P0(1÷',N,'):'
[12]   PSNEW←P0←□
[13]   K←0
[14]   LOOP:
[15]   'STEP = ',K
[16]   '   VECTOR OF STATES PROBABILITIES:',PSNEW
[17]   '----------------------------------------'
[18]   'NEXT STEP: 1 (FOR YES),  0  (FOR NO)'
[19]   YES←□
[20]   →(YES=0)/OUT
[21]   PSOLD←PSNEW
[22]   J←1
[23]   K←K+1
[24]   LABJ:I←1
[25]   Q←0
[26]   LABI:Q←Q+(PSOLD[I]×P[I;J])
[27]   →(N≥I←I+1)/LABI
[28]   PSNEW[J]←Q
[29]   →(N≥J←J+1)/LABJ
[30]   → LOOP
[31]   OUT: 'END OF MODEL'
```

Figure 3. APL2-module "MARKOV"

```
STEP=0
  VECTOR OF STATES PROBABILITIES: 1 0 0 0 0 0 0 0
----------------------------------------
NEXT STEP: 1 (FOR YES),  0  (FOR NO)
□:
  1
STEP=1
  VECTOR OF STATES PROBABILITIES: 0 0.7 0 0 0 0 0.3 0
----------------------------------------
NEXT STEP: 1 (FOR YES),  0  (FOR NO)
□:
  1
STEP=2
  VECTOR OF STATES PROBABILITIES: 0 0 0.28 0.07 0.14 0.07 0 0.14
----------------------------------------
NEXT STEP: 1 (FOR YES),  0  (FOR NO)
□:
  1
STEP=3
  VECTOR OF STATES PROBABILITIES: 0 0 0 0 0.14 0.28 0 0
----------------------------------------
NEXT STEP: 1 (FOR YES),  0  (FOR NO)
□:
  1
STEP=4
  VECTOR OF STATES PROBABILITIES: 0 0 0 0 0 0 0 0
----------------------------------------
NEXT STEP: 1 (FOR YES),  0  (FOR NO)
□:
  0
END OF MODEL
```

Figure 4. Program execution of the module "MARKOV"

The initial state of the investigation is represented at STEP=0 with the elements of the vector of initial probabilities, and the next steps reflects the development of the process $P(k)=\{p_1(k), \ldots, p_n(k)\}$. Obtaining a zero vector for the state probabilities determines the end of the experiment (reaching the final task) – in the case of performing STEP=4. Verification of the research results by means of Tables 1 and 2 is shown in Figure 5 and Figure 6.

```
   SETUP P
   NODES
1 2 3 4 5 6 7 8
   SIZE
8
   NETWORK
0  0.7  0    0    0    0    0.3  0
0  0    0.4  0.1  0.2  0.1  0    0.2
0  0    0    0    0    1    0    0
0  0    0    0    0    0    0    0
0  0    0    0    0    0    0    0
0  0    0    0    0    0    0    0
0  0    0    0    0    0    0    0
0  0    0    0    1    0    0    0
```

Figure 5. Overview of the initial conditions for conducting the experiment

```
  PATHSFROM  1                          List of determined paths in S(A)
1 2 3 6   1 2 4   1 2 5   1 2 6   1 2 8 5   1 7
  ρPATH←PATHSFROM  1                    Count of all paths in S(A)
6
  ↑PATH                                 Fist path in the list
1 2 3 6
  ARCS ↑PATH                            Probabilities for this first path
0.7  0.4  1
  x/ARCS ↑PATH                  Multiplication of probabilities for the first path
0.28
  x/¨ARCS¨PATH                  Multiplications for all paths in S(A)
0.28  0.07  0.14  0.07  0.14  0.3
  VALUE¨PATH                    Sum  of the weights for each path in S(A)
2.1  0.8  0.9  0.8  1.9  0.3
  □←V←x/¨ARCS¨PATH              Multiplications for all paths in S(A)
0.28  0.07  0.14  0.07  0.14  0.3
  □←MIN←⌈/V                     Minimum probability of realization of S(A)
0.07
  □←MAX←⌊/V                     Maximum probability of realization of S(A)
0.3
  (V=⌈/V)/PATH                  Path with maximal probability for realization
1  7
  (V=⌊/V)/PATH                  Path with minimal probability for realization
1 2 4   1 2 6
```

Figure 6. Stochastic investigation by using APL2-tool OR

5. CONCLUSION

The developed program functions “GSA”, “PATHS” [10] and “ESTIMATES” [14], supplemented with “MARKOV”, create a common program space for conducting analytical experiments in an APL2-environment based on the graph formalization of computer processing. For the purpose of the study, processing is considered as a set of independent processes realized in the execution of separate tasks. This allows the proposed research procedures to be applied to both multiprogramming and parallel computing. In addition to studying the stochastic characteristics of algorithmic structures, the developed program space can also be used for modelling and investigation dispatchers for system resource allocation and task scheduling in parallel and pseudo-parallel computer systems. For this, it is only necessary to determine the volume of the granule (program atom for processing), which will be represented by A_j in the generalized graph.

REFERENCES

[1] Kasabov, N. From Multilayer Perceptrons and neuro-fuzzy systems to deep learning machines: Which method to use? – A survey. *International Journal on Information Technologies and Security*, vol. 9, no. 2, 2017, pp. 3-24.

[2] Kravets, O.Ja., Atlasov, I.V., Aksenov, I.A., Molchan, A.S., Frantsisko, O.Yu., Rahman, P.A. Increasing efficiency of routing in transient modes of computer network operation. *International Journal on Information Technologies and Security*, vol. 13, no. 2, 2021, pp. 3-14.

[3] Zaslavskaya, O.Yu., Zaslavskiy, A.A., Bolnokin, V.B., Kravets, O.Ja. Features of ensuring information security when using cloud technologies in educational institutions. *International Journal on Information Technologies and Securit*y, vol. 10, no. 3, 2018, pp. 93-102.

[4] Hanson, J.R., Walker, S.I. Formalizing falsification for theories of consciousness across computational hierarchies. *Neuroscience of Consciousness*, vol. 2021, no. 2, 2021, niab014 (https://doi.org/10.1093/nc/niab014).

[5] Solov’ev, V.V. Synthesis of fast finite state machines on programmable logic integrated circuits by splitting internal states. *Journal of Computer and Systems Science International,* vol. 61, June 2022, pp. 360–371 (https://doi.org/10.1134/S1064230722030133).

[6] Zozulya, M.M., Kravets, O.Ja., Atlasov, I. V., Aksenov, I.A., Bozhko, L.M., Rahman, P.A. Algorithmization of the software testing system based on finite automata. *International Journal on Information Technologies and Security*, vol. 14, no. 1, 2022, pp. 77-86.

[7] Ramos, T.M.F., Almeida, A.A. & Ayala-Rincón, M. Formalization of the computational theory of a Turing complete functional language model. *Journal of Automated Reasoning,* 30 Jan 2022 (https://doi.org/10.1007/s10817-021-09615-x).

[8] Gowda1, D.V., Shashidhara, K.S., Ramesha, M., Sridhara, S.B., Manoj Kumar, S.B. Recent advance in graph theory and its applications. *Advances in Mathematics: Scientific Journal*, vol. 10, no. 3, 2021, pp. 1407–1412 (https://doi.org/10.37418/amsj.10.3.29).

[9] Munir, S., Jami, S.I., Wasi, S. Knowledge graph based semantic modelling for profiling in Industry 4.0, *International Journal on Information Technologies and Security*, vol. 12, no. 1, 2020, pp. 37-50.

[10] Romansky, R. An Approach for mathematical modelling and investigation of computer processes at a macro level. *Mathematics*, vol. 8, no. 10, Oct 2020, 1838 (DOI: 10.3390/math8101838).

[11] Digalovski, M., Rafajlovski, G. Distribution transformer mathematical model for power losses minimization. *International Journal on Information Technologies and Security*, vol.12, no. 2, 2020, pp. 57-68.

[12] Romansky, R., Noninska, I. Deterministic model investigation of processes in a heterogeneous e-learning environment. *International Journal of Human Capital and Information Technology Professionals (IJHCITP)*, vol. 13, no 1, 2022, article 28, pp. 1-16 (DOI: 10.4018/IJHCITP.293228).

[13] Romansky, R. Program environment for investigation of micro-level computer processing. *International Journal on Information Technologies and Security*, vol. 13, no. 1, March 2021, pp.83-92.

[14] Romansky, R. Mathematical modelling and study of stochastic parameters of computer data processing. *Mathematics*, vol. 9, no. 18, Sep 2021, art. 2240 (DOI: 10.3390/math9182240)

[15] Engel, S. Writing circuit histories. *Fast Capitalism*, vol. 15, 2018, pp. 19-30; doi:10.32855/fcapital.201801.004

[16] Hui, R.K.W., Kroberg, M.J. APL since 1978. *Proceeding of the ACM on Programming Languages*, vol. 4, no. HOPL, art. 69, June 2020, pp. 1-108 (https://doi.org/10.1145/3386319).

Information about the author:

Radi Romansky is a full professor at Technical University of Sofia, Doctor (Dr) in Computer Engineering and Doctor of Science (D.Sc.) in Informatics and Computer Science; Full member of European Network of Excellence on High Performance and Embedded Architectures and Compilation (HiPEAC). He has over 215 scientific publications and over 25 books. Areas of scientific interests: ICT, informatics, computer architectures, computer modelling, privacy and data protection, etc.

Manuscript received on 16 September 2022