Моделирование и анализ информационных систем. Т. 22, №6 (2015), с. 783–794
Modeling and Analysis of Information Systems. Vol. 22, No 6 (2015), pp. 783–794

DOI: 10.18255/1818-1015-2015-6-783-794

UDC 519.711

Teaching Formal Models of Concurrency Specification and Analysis

Shilov N. V.[1]

Received October 26, 2015

There is a widespread and rapidly growing interest to the parallel programming nowadays. This interest is based on availability of supercomputers, computer clusters and powerful graphic processors for computational mathematics and simulation. MPI, OpenMP, CUDA and other technologies provide opportunity to write C and FORTRAN code for parallel speed-up of execution without races for resources. Nevertheless concurrency issues (like races) are still very important for parallel systems in general and *distributed systems* in particular. Due to this reason, there is a need of research, study and teaching of formal models of concurrency and methods of distributed system verification.

The paper presents an individual experience with teaching Formal Models of Concurrency as a graduate elective course for students specializing in high-performance computing. First it sketches course background, objectives, lecture plan and topics. Then the paper presents how to formalize (i.e. specify) a reachability puzzle in semantic, syntactic and logic formal models, namely: in Petri nets, in a dialect of Calculus of Communicating Systems (CCS) and in Computation Tree Logic (CTL). This puzzle is a good educational example to present specifics of different formal notations.

The article is published in the author's wording.

Keywords: concurrency and parallelism, formal methods, formal models, Petri nets, calculi for communicating systems, labeled transition systems, reachability problem, temporal logic, model checking

For citation: Shilov N. V., "Teaching Formal Models of Concurrency Specification and Analysis", *Modeling and Analysis of Information Systems*, **22**:6 (2015), 783–794.

On the authors:
Shilov Nikolay Vyacheslavovich, orcid.org/0000-0001-7515-9647, PhD,
A.P. Ershov Institute of Informatics Systems SD RAS,
Lavrent'ev av., 6, Novosibirsk, 630090, Russia,
e-mail: shilov@iis.nsk.su

Acknowledgments:
[1]Reserch is part of Program IV.39.1 "Theory and applied problems of design and implementation of efficient and faultless program systems and information technologies".

Introduction

One of English-to-Russian technical translation problem is about Russian equivalent for English term *concurrency*. Unfortunately the term is translated as *параллелизм* i.e. by the same word as *parallelism*. To make distinction, let us quote a talk by Dan Grossman at Workshop on *Curricula for Concurrency and Parallelism* (Nevada, Oct. 17, 2010) [6]:

> *By parallelism, I mean using extra computational resources to solve a problem faster. By concurrency, I mean correctly and efficiently managing access to shared resources. While using these terms in this way is not entirely standard, the distinction is paramount.*

Rapid growth of parallel computing power raises questions about correctness (i.e. reliability, safety, liveness, etc.) of parallel system and programs. According to the above citation, concurrency is one of the critical issues related to the correctness of parallel systems and programs. It makes important to introduce formal models and methods of concurrency to Computer Science, Software Engineering and Computer Engineering curricula. But a serious problem for curricula development is diversity of individual notations. Another related problem is right choice of introductory, basic and/or advanced teaching/education level.

An elective topic on *Formal Models of Concurrency* at Department of Information Technology of Novosibirsk State University (IT NSU) and at Department of Applied Mathematics of Novosibirsk State Technical University (AM NSTU) was taught in years 2006-2012 during the second (spring) semester of graduate studies. The number of registered students varied from 8 to 16. Typically these students were affiliated with chairs of High-Performance Computing (IT NSU) and Parallel Computation Technologies (AM NSTU). The primary purpose of this course was to introduce basic concepts and means of semantic, syntactic and logic formal models of concurrency that had (already) become classics of Computer Science.

The rest of the paper is as follows. The syllabus of the course is sketched below in the present section; it comprises course background, objectives and topics & lecture plan. The next section 1 introduces a puzzle that is used in the course to illustrate/study/distiguish all formal models covered in the course. In particular, section 2 sketches how to represent the puzzle in Petri nets (semantical formal model), section 3 — in a dialect of Calculus of Communicating Processes (syntax modal), and section 4 — is Computational tree Logic (logic model). Finally we discuss in brief *paradigm of parallel programming* in the concluding section 5. The latest version (for fall semester of academic year 2012-13) of the recommended reading is presented in the References section of the paper and comprises English and Russian papers and books [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19].

0.1. Course Background

At the transition stage to cluster/multiprocessor/multicore architecture and ubiquitous parallelism, the importance (even urgency) of specifying, developing and validating parallel and concurrent behavior is increasing. Formal methods in Computer Science are mathematical theories and techniques for a sound specification, development and verification of soft/hard-ware. The use of formal methods is motivated by the expectation that rigorous mathematical notation and analysis help to understand better functionality and can improve the reliability and robustness. Different formal methods have different types of formal models: semantic models, syntax models, and logical where both syntax and semantics play important roles.

There are numbers of formal models for concurrency, these models have different types. First we have to point out at Petri nets as a purely semantic model. Next we must refer to Communicated Sequential Processes (CSP), an algebraic formal language with fixed syntax and denotational semantics. There exist several calculi that formalize different aspects of concurrency and parallelism: the Calculus of Communicating Systems (CCS), the Pi-Calculus for Communicating and Mobile Systems, the Ambient Calculus and its variations for mobile agents and security, etc. Various types of dynamic, process, and temporal logics are used for specification and verification of parallelism and concurrency.

The course should help Software Engineers and Applied Mathematicians to overview the spectrum of formal models for concurrency and parallelism, get idea of different reasoning techniques that are available for formal verification of concurrent and parallel systems. Students are expected to be familiar with basic concepts of concurrency and parallelism, elements of set theory and propositional Boolean logic.

0.2. Course Objectives

The course intends to help students to achieve the following objectives:

- to understand why we need formal models for concurrency and to know the concept of syntax, semantic and logic formal models;
- to use different formal models of concurrency for modeling concurrent and parallel systems;
- to comprehend definition of Petri nets and to learn reachability and boundedness properties and algorithms for Petri nets;
- to learn syntax and basic semantics of Communicated Sequential Processes (CSP);
- to become acquainted with the Calculus of Communicating Systems (CCS), the Pi-Calculus, and the Ambient Calculus;
- to learn the concept of Labeled Transition Systems (LTS) and to understand why LTS are used for semantics of different formal models;
- to become acquainted with the concept of bisimulation for LTS and to know about Hennessy-Milner logic and its relations to bisimulation;
- to become familiar with syntax and semantics of Computation Tree Logic (CTL) and its use for specification and verification of properties of LTS;
- to discuss ways of introduction of formal models of concurrency and parallelism into Software Engineering practice.

0.3. Topics and Lecture Plan

Introduction. Concepts of syntax, semantic, algebraic and logic formal models (by study of naive set theory, propositional formulas, Boolean algebras, propositional logic). Why do we need formal models for concurrency?

Basics of Petri nets. Definition of marked Petri net. Firing, small/big-step semantics. Reachability problem and reachability tree/graph. Boundedness problem and checking. Problem solving with aid of Petri nets.

Process Algebras. Communicating Sequential Processes (CSP) by A. Hoare. Process algebra of J. Bergstra and J. Klop.

Calculi for concurrency. Calculus of Communicating Systems (CCS) by R. Milner. Pi-Calculus by R. Milner. Ambient Calculus by L. Cardelli. Modeling with aid of calculi.

Labeled Transition Systems. Definition of LTS. Examples of LTS by Petri nets and calculi. Definition of bisimulation and its properties. Bisimulation and Hennessy-Milner Logic.

Computational Tree Logic. CTL syntax and semantics. Using CTL to specify properties of LTS. CTL Model Checking: algorithmic verification of LTS

Conclusion. Do we need a comprehensive model for concurrency and parallelism?

1. One Puzzle for Many Formalisms

The following puzzle *Four Men and a Boat* was used in the course to illustrate several formal models, namely *Petri nets*, a dialect of *Calculus of Communicated Systems* and *Computational Tree Logic.*

> Four men Albert, Conrad, Donald and Edmund are on the left bank of a river and need to move to the right bank by a boat that has 2 seats and one pair of oars. Sporty Albert can cross the river by the boat without a companion in 5 minutes (in one forth or back direction), regular Conrad can do the same in 10 minutes, fatty Donald – in 20 minutes, and fat Edmund – in 25 minutes. When any two men are crossing the river together the pace of the boat is defined by the fattest man in the pair, ex., Albert and Donald together can cross the river in 20 minutes. Question: do these four men can cross the river in one hour?

This is really a very nice puzzle! Typically 8 in 10 students (in my experience) first “prove” that the four men cannot cross the river in one hour. They usually assume that sporty Albert have to accompany (convoy) other men because he is the fastest and it would be better him to transport the boat back every time; under this assumption transportation of 4 men takes 1 hour and 5 minutes.

The author also made this assumption when heard the puzzle for the first time 17 years ago, and was very much confused when Andrei Sabelfeld (http://www.cse.

chalmers.se/~andrei/, he was a student at the time of the story) told him that it is wrong. In turn Andrey simply gave the following scenario how men can cross the river in one hour:

- first Albert and Conrad cross the river together in 10 minutes, then Albert transports the boat back in 5 minutes;
- next Donald and Edmund cross the river together in 25 minutes and Conrad transports the boat back in 10 minute;
- finally Albert and Conrad cross the river together again in 10 minutes.

So this puzzle gave a good lesson to author to be skeptical about "obvious" assumptions.

2. Puzzle in Petri nets

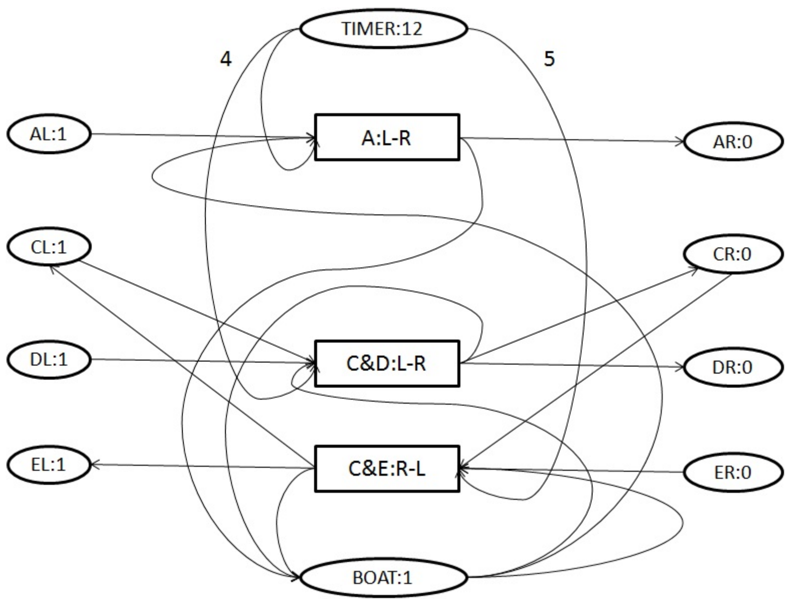

Fig 1. Fragment of net model for Four Men and a Boat Puzzle

A sketch of a marked Petri net that models the puzzle is presented on the figure 1. One can see on this figure

- a place TIMER with initial marking 12 tokens each of which models 5 minutes (because all time values in the puzzle are dividable by 5);
- a place BOAT with initial marking 1 that models the boat;
- places AL, CL, DL and EL with initial marking 1 each; the initial marking models that initially Albert (A), Conrad (C), Donald (D) and Edmund (E) are on the left (L) bank of the river;

- places AR, CR, DR and ER with initial markings 0 each; the initial marking models that initially Albert (A), Conrad (C), Donald (D) and Edmund (E) are absent on the right (R) bank of the river;
- three sample transitions that model situations when Albert moves from left bank to the right (A:L-R), when Bob and Conrad move together from left bank to the right (C&D:L-R), and when Bob and Donald move together from right bank to the left (C&E:R-L) by boat; omitted 17 transitions are similar;
- several edges without tags and two edges tagged by integers 4 and 5: edges without tags have multiplicity 1, edges tagged by integers have multiplicity according to their tags; the multiplicity represents the number of 5 minutes intervals that is required for the corresponding transition;
- firing of each transition models a move from one bank to another by a man or by a pair of men in the boat.

The presented marked net is bounded, it can be checked by construction of Karp-Miller coverage tree. The background (unmarked) net is structurally bounded (i.e. it is bounded for every initial marking), the structural boundedness of the net is obvious[1].

In terms of Petri nets, to solve the puzzle means that the following final marking is reachable from the initial one (TIMER:0, AL:0, CL:0, DL:0, EL:0, AR:1, CR:1, DR:1, ER:1, BOAT:1). Since the net model is bounded, the set of all reachable markings is finite[2]; so the puzzle can be solved by construction of the reachability graph for the marked net. Let us observe that this reachability graph is an example of labeled transition system where nodes are reachable markings of the net and edges are transition firings.

3. Puzzle in CCS

Let us describe a simplified dialect of Calculus of Communicating Systems. Syntax of the dialect is defined in terms of the following three context-free grammars:

$$\begin{array}{r l}
S ::= & \langle empty \rangle \mid N = \ P; S \\
P ::= & 0 \mid A.P \mid (P + P) \mid (P|P) \mid N \\
A ::= & R \mid I \mid O
\end{array}$$

where

- N, R, I and O are disjoint finite alphabets which elements are called *process names*, *regular*, *input* and *output* (*action*) *symbols*,
- words generated from symbol P are called *processes*, and words generated from from symbol S are sets of *definitions* (separated by semicolon).

[1]Sorry for claiming that something is "obvious". This time the claim follows from the net structure: for every marking every firing of a transition reduces the total number of tokens.

[2]It contains not more than 208 reachable states.

Process constructors ".", "+" and "|" are conventionally called *sequential*, *nondeterministic* and *parallel* compositions. A set of definitions is consistent if every name that occurs in the system has single definition in the set.

Input and output alphabets are mutually complementary: they are equipped by a bijection function $(\ )^c$ such that

- for every input symbol i the complement i^c is an output symbol and $(i^c)^c = i$,
- for every output symbol o the complement o^c is an input symbol and $(o^c)^c = o$.

Let us specify the puzzle in terms of our dialect of CCS. Let

$$TIMER = \underbrace{tick.tick.tick.tick.tick.tick.tick.tick.tick.tick.tick.tick.}_{12\ times}0$$

be definition of a process name $TIMER$ where $tick$ is an output symbol (to represent time interval of 5 minutes) with the complementary input symbol $tack$. This input symbol is used in process definitions affiliated with men in the puzzle. For example, in definitions for two process names AL and AR (that correspond to Albert on the left and on the right banks) follow below:

$$\begin{aligned} AL = &\ (acqBL.tack.AR\ +\ ALackCL.AR\ +\ ALackDL.AR\ +\ ALackEL.AR) \\ AR = &\ (done.0\ +\ acqBR.tack.AL\ + \\ &\ +\ ARackCR.AL\ +\ ARackDR.AL\ +\ ARackER.AL) \end{aligned}$$

where

- $done$ is a regular action symbol;
- $acqBL$ and $acqBR$ are output symbols (to represent that a man acquires the boat located on the left/right back respectively) with complementary input symbols $relBL$ and $relBR$ (to represent boat release);
- $ALackCL$, $ALackDL$, $ALackEL$ and $ARackCR$, $ARackDR$, $ARackER$ are input symbols (to represent that Albert acknowledges an invitation of Conrad, Donald or Edmund with the same location to cross the river together) with complimentary symbols $CLaskAL$, $DLaskAL$, $ELaskAL$ and $CRaskAR$, $DRaskAR$, $ERaskAR$ respectively (to represent invitations).

The intended semantics of these two definitions is behavior of Albert on the left and on the right banks.

- AL specifies that on the left bank he has the following 4 disjoint options:
  - acquire the boat ($acqBL$), cross the river (synchronizing $tack$ with $tick$) and then proceed further on the right back according to AR;
  - acknowledge invitation to use the boat acquired by Conrad ($ALackCL$) to cross the river and then proceed further on the right back according to AR;
  - acknowledge invitation to use the boat acquired by Donald ($ALackDL$) to cross the river and then proceed further on the right back according to AR;

  - acknowledge invitation to use the boat acquired by Edmund ($ALackEL$) to cross the river and then proceed further on the right back according to AR.

- AR specifies that on the right bank he has the following 5 disjoint options:
  - stop any further activities and remain on the right bank ($done$);
  - acquire the boat ($acqBR$), cross the river (synchronizing $tack$ with $tick$) and then proceed further on the left back according to AL;
  - acknowledge invitation to use the boat acquired by Conrad ($ARackCR$) to cross the river and then proceed further on the left back according to AL;
  - acknowledge invitation to use the boat acquired by Donald ($ARackDR$) to cross the river and then proceed further on the left back according to AL;
  - acknowledge invitation to use the boat acquired by Edmund ($ARackER$) to cross the river and then proceed further on the left back according to AL.

Definitions for names CL and CR (that correspond to Conrad on the left and on the right banks), DL and DR (that correspond to Donald on the left and on the right banks), and EL and ER (that correspond to Edmund on the left and on the right banks) are presented in the table 1.

Table 1. Definitions that model Conrad, Donald and Edmund behavior

$$\begin{aligned}
CL &= acqBL.tack.tack.(CR + CLaskAL.CR) + CLackDL.CR + CLackEL.CR\\
CR &= done.0 + acqBR.tack.tack.(CL + CRackAR.CL) + CRackDR.CL + CRackER.CL\\
DL &= acqBL.tack.tack.tack.tack.(DR + DLaskAL.DR + DLaskCL.DR) + CLackEL.CR\\
DR &= done.0 + acqBR.tack.tack.tack.tack(DL + DRaskAR.DLDRaskCR.DL)+\\
&\qquad +DRackER.DL\\
EL &= acqBL.tack.tack.tack.tack.tack.\\
&\qquad (ER + ELaskAL.ER + ELaskCL.ER + ELaskDL.ER)\\
ER &= done.0 + acqBR.tack.tack.tack.tack.tack.\\
&\qquad (EL + ERaskAR.EL + ERaskCR.EL + ERaskDR.EL)
\end{aligned}$$

The last two definitions for names BL and BR correspond to the boat on the left and on the right banks:

$$BL = relBL.BR \text{ and } BR = (done.0 + relBR.BL).$$

The initial configuration of the puzzle can be represented as the following process

$$(TIMER|AL|CL|DL|EL|BL).$$

Reduction rules are very standard for CCS:

- if r is a regular action symbol then $r.\alpha \to \alpha$,
- if $\alpha \to \gamma$ then $(\alpha + \beta) \to \gamma$ and $(\alpha|\beta) \to (\gamma|\beta)$,
- if s is an input/output symbol then $(s.\alpha|s^c.\beta|\gamma) \to (\alpha|\beta|\gamma)$,

as well as the following congruencies:

- $\alpha + \beta \equiv \beta + \alpha$ and $\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$,
- $\alpha|0 \equiv \alpha$, $\alpha|\beta \equiv \beta|\alpha$, and $\alpha|(\beta|\gamma) \equiv (\alpha|\beta)|\gamma$,
- if S is a set of name definitions and $n = \alpha$ is one of them then[3] $n \overset{S}{\equiv} \alpha$.

In terms of the presented CCS-dialect, to solve the puzzle means that the final process 0 is reachable (assuming system of name definitions that comprises all definitions listed above are provided) in the reduction graph for the initial process $(TIMER \mid AL \mid CL| DL \mid EL \mid BL)$. This graph is finite because every nondeterministic branch with recursion in any of our definitions contains at least one *tack* that must be synchronize with *tick*, but the total amount of ticks is 12 (according to definition of $TIMER$). So the puzzle can be solved in terms of CCS by construction of the reduction graph. Let us observe that this graph is also an example of labeled transition system where nodes are process configurations (that are reducts of the initial process) and edges are reductions.

4. Puzzle in CTL

As follows from sections 2 and 3, the puzzle can be reduced to the reachability problem for finite graphs (label transition systems in particular) with aid of Petri nets (semantic model) or CCS (syntactic model). Moreover, the puzzle can be solved on the fly at time of constructing the corresponding labeled transition system.

But let us recall that students who try to solve the puzzle usually believe that Albert have to accompany other men for accelerating transportation. In other words they make the following *belief assumption*:

If a positive solution exists,
then there exists a solution where Albert convoys other men
until all are on the right bank.

There are two ways how to refute this belief: human-oriented or computer-aided.

- Human-oriented way in this case comprises two steps: first someone should solve the puzzle (it was Andrey Sabelfeld in my case), then prove manually impossibility of a solution where Albert convoys other men (that is very easy).
- Computer-aided approach needs to build the corresponding labeled transition system first, then to modify a reachability algorithm to check whether there exists a path where Albert is always on the move.

The first step of a computer-aided approach can be carried out in many ways: for example one may construct reachability graph for the Petri net (that model the puzzle), or reduction graph for the corresponding CCS specification. The next step of the approach may be generalized: it makes sense to build an efficiently decidable graph *query language* so that the standard reachability (and many its modifications) becomes just a special query of this language.

[3]Supscript S may be omitted when the system is implicit.

There are many query languages of this sort indeed, Computation Tree Logic (CTL), Linear Temporal Logic, μ-Calculus for instance. For example, in CTL the belief assumption can be represented by CTL formula presented in the table 2. Here $Albert\_at\_Left$, $Conrad\_at\_Left$, …, $Edmund\_at\_Right$ and $Albert\_on\_Move$ are propositional variables with natural interpretation in the labeled transition system.

Table 2. CTL specification for the *belief assumption* about the puzzle

$$\begin{aligned}
&(Albert\_at\_Left \ \& \ Conrad\_at\_Left \ \& \\
&\& \ Donald\_at\_Left \ \& \ Edmund\_at\_Left \ \& \\
&\& \ Boat\_at\_Left \ \& \ Timer\_is\_Set) \ \rightarrow \\
&\qquad \rightarrow \ (\mathbf{EF}(Albert\_at\_Right \ \& Conrad\_at\_Right \ \& \\
&\qquad \& \ Donald\_at\_Right \ \& \ Edmund\_at\_Right) \ \rightarrow \\
&\qquad \rightarrow \mathbf{E}(Albert\_on\_Move \ \mathbf{U} \ (Albert\_at\_Right \ \& \ Conrad\_at\_Right \ \& \\
&\qquad\qquad \& \ Donald\_at\_Right \ \& \ Edmund\_at\_Right)))
\end{aligned}$$

5. Parallel Programming Paradigm

Paradigm is an approach to problem formulation/formalization and the ways to solve them. The term comes from Greek and means *pattern*, *example* (nouns), *exhibit*, *represent* (verbs). A contemporary meaning of the term is due to well-known book by Tomas Kuhn [11]. Robert Floyd was the first who had explicitly used the term in the Computer Science context. In particular, he addressed *Paradigms of Programming* in his Turing Award Lecture in 1968 [5]. Unfortunately, R. Floyd had not defined explicitly this concept. Peter Van Roy has published in 2009 a taxonomy *The principal programming paradigms* (at `http://www.info.ucl.ac.be/~pvr/paradigms.html`) with 27 different paradigms. He also suggested the following definition [16]:

> *A programming paradigm is an approach to programming a computer based on a mathematical theory or a coherent set of principles. Each paradigm supports a set of concepts that makes it the best for a certain kind of problem.*

The definition suggested by Peter Van Roy has been refined in [18] as follows.

- Programming paradigms are alternative approaches (patterns) to formalization of information problems (problem statements), data presentation, handling, and processing.
- They are fixed in a form of formal (mathematical) theory and accumulated in programming languages.
- Programming value of a paradigm may be characterized by a set of programming problems/application areas that the paradigm fits better than the other ones.
- Educational value of programming paradigms is to teach to think different about programming problems and to select the best paradigm to solve any particular problem.

This definition assumes that teaching/learning formal models/theories is obligatory for mastering/comprehending any programming paradigm, the parallel programming in particular. And the puzzle about four men and a boat helps to teach and learn some formal models and theories of concurrency.

References

[1] *Handbook of Process Algebra*, eds. J. A. Bergstra, A. Ponse, S. A. Smolka, Elsevier, Amsterdam, 2001.

[2] L. Cardelli, A. D. Gordon, "Mobile ambient", Lecture Notes in Computer Science, **1378**, Springer-Verlag, Berlin, Heidelberg, 1998, 140–155.

[3] L. Cardelli, "Mobility and Security.", *Foundations of Secure Computation*, Proceedings of the NATO Advanced Study Institute on Foundations of Secure Computation, IOS Press, Amsterdam, 2000, 3–37.

[4] E. M. Jr. Clarke, O. Grumberg, D. A. Peled, *Model Checking*, MIT Press, Cambridge, Massachusetts, 1999.

[5] R. W. Floyd, "The paradigms of Programming", *Communications of ACM*, **22** (1979), 455–460.

[6] D. Grossman, "Ready-For-Use: 3 Weeks of Parallelism and Concurrency in a Required Second-Year Data-Structures Course", SPLASH 2010 Workshop on Curricula for Concurrency and Parallelism (Reno, Nevada, USA, Oct. 17, 2010), https://homes.cs.washington.edu/~djg/papers/grossmanSPAC_position2010.pdf.

[7] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice Hall, Upper Saddle River, New Jersey, 1985, (This book was updated by Jim Davies at the Oxford University Computing Laboratory in 2004 and the new edition is available at the http://www.usingcsp.com/).

[8] Yu. G. Karpov, *Model Checking. Verificaciya parallelnyh i raspredelennyh programmnyh system*, BHV-Peterburg, Saint Petersburg, 2010, (In Russian).

[9] V. E. Kotov, *Seti Petri*, Nauka, Moscow, Russia, 1987, (In Russian).

[10] Z. Manna, A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems: Specification*, Springer-Verlag,, New York, 1992.

[11] T. S. Kuhn, *The structure of Scientific Revolutions*, Univ. of Chicago Press, Chicago and London, 1996, (3rd Edition).

[12] R. Milner, *Communicating and Mobile Systems: the Pi-Calculus*, Cambridge University Press, Cambridge, England, 1999.

[13] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, **92**, Springer-Verlag, Berlin, Heidelberg, 1980.

[14] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice Hall, Upper Saddle River, New Jersey, 1981.

[15] W. Reisig, *A Primer in Petri Net Design*, Springer-Verlag, Berlin, Heidelberg, 1992.

[16] P. van Roy, "Programming Paradigms for Dummies: What Every Programmer Should Know", *New Computational Paradigms for Computer Music,*, eds. G. Assayag, A. Gerzso, IRCAM/Delatour, France, 2009, 9–38.

[17] N. V. Shilov, K. Yi, "How to Find a Coin: Propositional Program Logics Made Easy", *Current Trends in Theoretical Computer Science.* V. 2, World Scientific, Singapore, 2004, 181–214.

[18] N. V. Shilov et al., "Development of the Computer Language Classification Knowledge Portal", *Ershov Informatics Conference PSI'11*, Lecture Notes in Computer Science, **7162**, Springer-Verlag, Berlin, Heidelberg, 2012, 340–348.

[19] C. Stirling, *Modal and Temporal Properties of Processes*, Springer-Verlag, New York, 2001.

DOI: 10.18255/1818-1015-2015-6-783-794

О преподавании формальных моделей и алгоритмов анализа параллельных систем

Шилов Н. В.[1]

получена 26 октября 2015

В настоящее время наблюдается огромный практический интерес к параллельному программированию. Этот интерес обусловлен доступностью супер-ЭВМ, компьютерных кластеров и мощных графических процессоров для массового использования в вычислительной математике и компьютерном моделировании. Кроме того, такие технологии параллельного программирования, как MPI, OpenMP и CUDA, позволяют использовать безопасным образом опыт программирования на классических языках Си и FORTRAN для ускорения вычислений, избегая конфликтов ("гонок") из-за ресурсов. Однако такой прогресс параллельного программирования не означает, что конкуренция из-за ресурсов не может возникать в параллельных общего вида, в так называемых *распределенных системах* в частности. Поэтому остается актуальным изучение и преподавание формальных моделей параллелизма и средств верификации поведенческих свойств параллельных (распределенных) систем.

В статье представлен опыт преподавания специального курса по формальным моделям параллелизма для магистрантов и аспирантов, специализирующихся в области высокопроизводительных вычислений. Сначала в статье дан обзор курса в целом, предварительных знаний, необходимых для этого курса, целей и задач курса, представлен план лекций и список рекомендованной литературы. Затем представлен пример одной поучительной головоломки (на достижимость в пространстве состояний) и ее формализации средствами семантических, синтаксических и логических моделей, как-то: сетями Петри, средствами исчисления параллельных взаимодействующих процессов (CCS) и темпоральной логики CTL. Эта головоломка — хороший пример для того, чтобы показать специфику и пользу каждого из рассмотренных формализмов.

Статья представляет собой расширенную версию доклада на VI Международном семинаре "Program Semantics, Specification and Verification: Theory and Applications", Казань, 2015.

Статья публикуется в авторской редакции.

Ключевые слова: системы с параллелизмом, формальные методы, формальные модели, сети Петри, исчисления взаимодействующих процессов, помеченные системы переходов, проблема достижимости, темпоральная логика, верификация моделей

Для цитирования: Шилов Н. В., "О преподавании формальных моделей и алгоритмов анализа параллельных систем", *Моделирование и анализ информационных систем*, **22**:6 (2015), 783–794.

Об авторах:
Шилов Николай Вячеславович, orcid.org/0000-0001-7515-9647,
канд. физ.-мат. наук, старший научный сотрудник,
Институт систем информатики им. А. П. Ершова СО РАН
630090 Россия, г. Новосибирск, пр. Лаврентьева, 6,
e-mail: shilov@iis.nsk.su

Благодарности:
[1]Работа выполнена в рамках программы IV.39.1 «Теоретические и прикладные проблемы создания эффективных надежных программных систем и информационных технологий».