УДК 004.052.42

# ФОРМИРОВАНИЕ БАЗЫ ЗНАНИЙ НА ОСНОВЕ СТРУКТУРЫ КРИПКЕ И ФОРМУЛ ТЕМПОРАЛЬНОЙ ЛОГИКИ

**Мельцов В.Ю., Чистяков Г.А.**

*ФГБОУ ВПО «Вятский государственный университет», Киров, e-mail: gennadiychistyakov@gmail.com*

В работе предлагается способ формирования базы знаний в логике предикатов первого порядка, отражающей взаимосвязи между состояниями модели некоторого алгоритма. Кроме того, в базу знаний включаются утверждения, описывающие желаемое поведение исследуемого алгоритма. Полученная структура может быть использована для формального доказательства корректности с помощью техники model checking. Способ позволяет применить на заключительном этапе верификации, сопоставлении модели и предъявляемых требований математический аппарат теории логического вывода. Подобное решение направлено на сокращение общего времени проверки за счет высокой степени параллелизма предлагаемого подхода и возможности применения специализированных эвристик, рассчитанных на алгоритмы разной структуры. Способ отличается линейной от числа связей между состояниями модели и числа лексем спецификации асимптотической сложностью и не накладывает дополнительных ограничений на выбор конкретного метода вывода. Однако вследствие высокого быстродействия к использованию может быть рекомендован ускоренный модифицированный метод деления дизъюнктов.

**Ключевые слова: формальная верификация, логический вывод, темпоральная логика, метод деления дизъюнктов, структура Крипке**

# BUILDING THE KNOWLEDGE BASE ON THE BASIS OF KRIPKE STRUCTURES AND TEMPORAL LOGIC FORMULAS

**Meltsov V.Y., Chistyakov G.A.**

*Vyatka State University, Kirov, e-mail: gennadiychistyakov@gmail.com*

This paper proposes a method of building a knowledge base in the first-order predicate logic, which reflects the relationship between the states of a algorithm model. In addition, the knowledge base includes statements that describe the desired behavior of the investigated algorithm. The resulting structure can be used to formally prove the correctness with using the model checking technique. The method allows us to apply on the final stage of verification, comparison of models and requirements, the mathematical apparatus of the inference theory. Such decision is aimed at reducing the total verification time due to the high degree of parallelism of the proposed approach and the possibility of using specialized heuristics, that can be designed for algorithms of different structures. The method is characterized by the linear asymptotic complexity depending of the number of relations between the states of the model and the number of specification tokens and does not impose additional restrictions on the choice of a particular inference method. However, due to the high speed to the use can be recommended accelerated modified disjuncts division method.

**Keywords: formal verification, inference, temporal logic, disjuncts division method, Kripke structure**

На протяжении последнего десятилетия перед специалистами в различных сферах компьютерных технологий остро стоит проблема верификации программного обеспечения. Не обнаруженная на этапе разработки ошибка способна привести к катастрофическим последствиям. В некоторых прикладных областях невозможно своевременно остановить неправильно функционирующую программу, или ее остановка обойдется слишком дорого. Скрытые ошибки рано или поздно могут стать причиной огромных финансовых потерь и даже человеческих жертв, что подтверждает целый ряд примеров [1, 6].

К сожалению, даже самое глубокое тестирование не способно гарантировать корректности выполнения программы. Ситуация значительно усложняется при верификации параллельных процессов, так как в этом случае ошибки могут не проявляться до наступления специфических условий, связанных с их взаимодействием. И если корректность последовательной программы можно доказать, подав на вход всевозможные комбинации исходных данных и затем сравнив выходные данные с ожидаемыми, то для параллельной программы такой подход не дает оснований говорить об отсутствии ошибок. Кроме того, тестирование параллельных программ требует значительных временных затрат. В ряде случаев разработка тестов занимает на несколько порядков больше времени, чем разработка проверяемой программы.

В силу указанных причин в последнее время популярность приобретает формальный подход к верификации алгоритмов и программ, позволяющий однозначно ответить на вопрос корректности объекта анализа [3].

Одним из наиболее эффективных методов формальной проверки является техника model checking, включающая три элемента: способ построения модели исследуемого объекта, способ спецификации предъявля-

емых требований и способ проверки соответствия модели требованиям.

**Применение логического вывода для решения задачи верификации**

Удобным способом представления модели алгоритма (или программы) является структура Крипке, которая отличается простотой и высокой степенью универсальности [2]. Требования к объекту исследования могут быть описаны посредством формул темпоральной логики – расширения классической логики высказываний, позволяющего учитывать порядок событий во времени [1]. Именно эти элементы являются базовыми при выполнении первого и второго этапа формальной верификации с помощью техники model checking.

Классический подход предполагает, что сопоставление модели и спецификации может быть проведено на основе некоторых теорем и структур математического аппарата теории автоматов (сравнение реагирующего и контрольного автоматов Бюхи) [1]. Однако третий этап верификации возможно модифицировать следующим образом.

Ряд существующих методов логического вывода обладает высокой степенью параллелизма, что может быть использовано для ускорения процесса проверки соответствия модели предъявляемым требованиям [5]. Но в таком случае требуется решить задачу построения базы знаний, эквивалентной модели объекта верификации и представленной в форме, подходящей для обработки машиной логического вывода. Кроме того, необходим алгоритм преобразования спецификации в целевое утверждение, выводимость которого будет устанавливаться.

В нашем случае задача формирования базы знаний на языке логики предикатов первого порядка может быть решена с помощью следующего алгоритма.

1. Для установления связей между состояниями структуры и множеством атомарных предикатов введем двухместный предикат Event (α, β), который принимает истинное значение, если состояние структуры α отмечено атомарным предикатом β, и ложное в противном случае.

2. Для определения возможности перехода из одного состояния структуры в другое за один шаг введем двухместный предикат Parent (α, β), который принимает истинное значение, если из состояния α структуры Крипе существует ребро в состояние β.

3. Для определения отсутствия возможности перехода из вершины введем одноместный предикат End (α), который принимает истинное значение, если состояние α структуры Крипке не имеет исходящих дуг.

4. Для определения возможности перехода из одного состояния структуры в другое введем двухместный предикат Path (α, β), который принимает истинное значение, если из состояния α структуры Крипке существует путь в состояние β. Значение предиката для конкретных вершин структуры может быть вычислено через предыдущий как

$$Parent(\alpha,\beta) \vee \exists\chi(Parent(\alpha,\chi) \rightarrow Path(\chi,\beta)) \rightarrow Path(\alpha,\beta).$$

5. Дополнительно необходимо определить предикаты, отражающие требования к объекту верификации. Для этого следует построить дерево грамматического разбора заданной спецификации. Этот шаг может быть выполнен с помощью алгоритма, представленного в работе [4].

6. Последовательно, начиная с листьев дерева разбора, сопоставить каждой вершине отдельный предикат. Правила, по которым формируется сопоставляемый предикат, определяются логикой, используемой для описания требований к объекту верификации. Для спецификации в виде формулы темпоральной логики линейного времени (LTL) необходимо использовать следующий набор правил.

– Если вершина соответствует атомарному предикату, то сопоставить ей предикат вида Event (λ, χ) → $P_i(\lambda)$, где $P_i$ – вводимый предикат, а χ – атомарный предикат.

– Если вершина соответствует операции логического отрицания, то сопоставить ей предикат, получаемый как отрицание предиката-потомка данной вершины $\neg P_j(\lambda) \rightarrow P_i(\lambda)$, где $P_i$ – вводимый предикат, а $P_j$ – предикат, введенный ранее при анализе потомка текущей вершины).

– Если вершина соответствует операции конъюнкции, то сопоставить ей предикат, получаемый с помощью конъюнкции предикатов-потомков данной вершины $P_j(\lambda) \wedge P_k(\lambda) \rightarrow P_i(\lambda)$, где $P_i$ – вводимый предикат, а $P_j$ и $P_k$ – предикаты, введенные ранее при анализе потомков текущей вершины).

– Если вершина соответствует операции дизъюнкции, то сопоставить ей предикат, получаемый с помощью дизъюнкции предикатов-потомков данной вершины $P_j(\lambda) \vee P_k(\lambda) \rightarrow P_i(\lambda)$, где $P_i$ – вводимый предикат, а $P_j$ и $P_k$ – предикаты, введенные ранее при анализе потомков текущей вершины).

– Если вершина соответствует темпоральному оператору $X$, то сопоставить ей предикат вида

$$\forall\chi(\neg Parent(\lambda,\chi) \vee P_j(\chi)) \rightarrow P_i(\lambda),$$

где $P_i$ – вводимый предикат, а $P_j$ – предикат, введенный ранее при анализе потомка текущей вершины.

– Если вершина соответствует темпоральному оператору $G$, то сопоставить ей предикат вида

$$\forall\chi(\neg Path(\lambda,\chi)\vee P_j(\chi))\rightarrow P_i(\lambda),$$

где $P_i$ – вводимый предикат, а $P_j$ – предикат, введенный ранее при анализе потомка текущей вершины.

– Если вершина соответствует темпоральному оператору $F$, то сопоставить ей предикат вида

$$(\forall\chi(\neg Parent(\lambda,\chi)\vee P_i(\chi))\wedge End(\lambda))\vee P_j(\lambda)\rightarrow P_i(\lambda),$$

где $P_i$ – вводимый предикат, а $P_j$ – предикат, введенный ранее при анализе потомка текущей вершины.

– Если вершина соответствует темпоральному оператору $U$, то сопоставить ей предикат вида

$$(\forall\chi(\neg Parent(\lambda,\chi)\vee P_i(\chi))\wedge P_j(\lambda)\wedge\neg End(\lambda))\vee P_k(\lambda)\rightarrow P_i(\lambda),$$

где $P_i$ – вводимый предикат, а $P_j$ и $P_k$ – предикаты, введенные ранее при анализе потомков текущей вершины.

– Если вершина соответствует темпоральному оператору $W$, то сопоставить ей предикат вида

$$(\forall\chi(\neg Parent(\lambda,\chi)\vee P_i(\chi))\wedge P_j(\lambda))\vee P_k(\lambda)\rightarrow P_i(\lambda),$$

где $P_i$ – вводимый предикат, а $P_j$ и $P_k$ – предикаты, введенные ранее при анализе потомков текущей вершины.

– Если вершина соответствует темпоральному оператору $R$, то сопоставить ей предикат вида

$$(\forall\chi(\neg Parent(\lambda,\chi)\vee P_i(\chi))\vee P_j(\lambda))\wedge P_k(\lambda)\rightarrow P_i(\lambda),$$

где $P_i$ – вводимый предикат, а $P_j$ и $P_k$ – предикаты, введенные ранее при анализе потомков текущей вершины.

Так как LTL-формула описывает желаемое поведение объекта верификации, то факт присутствия ошибки может быть подтвержден существованием в структуре Крипке последовательности состояний, нарушающей это поведение. Следовательно, выводимым утверждением будет являться отрицание предиката, сопоставленного корневой вершине дерева грамматического разбора анализируемой спецификации, для начального состояния модели.

### Пример верификации с помощью ускоренного метода логического вывода

Рассмотрим процесс поиска ошибки на примере.

Пусть требуется проверить корректность алгоритма, состоящего из трех параллельно реализуемых процессов. Действия, совершаемые каждым из процессов, могут быть записаны на языке псевдокода следующим образом.

Основной процесс:

1: Загрузка данных // Загрузка из ВЗУ данных, необходимых для работы.

2: Инициализация вспомогательного процесса // Запуск процесса-обработчика запросов.

3: Инициализация семафора // Инициализация семафора, регулирующего доступ к данным.

Вспомогательный процесс:

1: Цикл (пока не получен сигнал завершения).

2: Если (есть успешная попытка успешного подключения), то // Если есть пользовательский запрос.

3: Инициализация процесса обработки данных // Запуск отдельного процесса, обрабатывающего пользовательский запрос.

4: Конец цикла // Метка не является командой.

Процесс обработки данных:

1: Захват семафора.

2: Обработка данных // Выполнение запроса пользователя, модификация данных.

3: Освобождение семафора.

Структуры Крипке, эквивалентные каждому из процессов, представлены на рис. 1.

Вершина $S_i^j$ соответствует команде с номером $i$ $j$-го по порядку процесса, при этом $S_0^j$ и $S_e^j$ соответствуют состояниям начала и завершения процесса. Множество атомарных предикатов состоит из двух элементов {α, β}, первый из которых соответствует операции инициализации семафора, а второй – операциям взаимодействия с семафором.

Асинхронная композиция структур Крипке представлена на рис. 2. Вершина $S_{i|j|k}^l$ соответствует объединению команд $i$, $j$, $k$ и имеет порядковый идентификатор $l$. Следует заметить, что при построении данной композиции состояния $S_1^1$ и $S_2^1$ перво-

го процесса и $S_2^2$ и $S_3^2$ второго были сжаты в одно состояние с меньшим номером. Кроме того, состояние $S_{0|0|0}^0$ соответствует моменту времени, когда ни один процесс не начал свою работу, а $S_{e|e|e}^{48}$ – когда все процессы завершились.

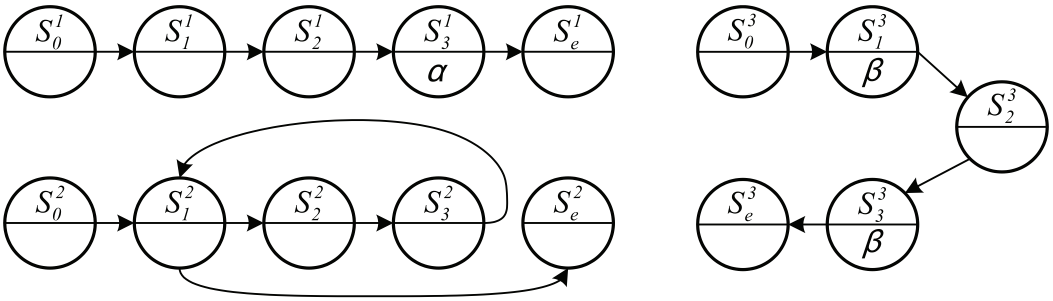


*Рис. 1. Структуры Крипке отдельных процессов*

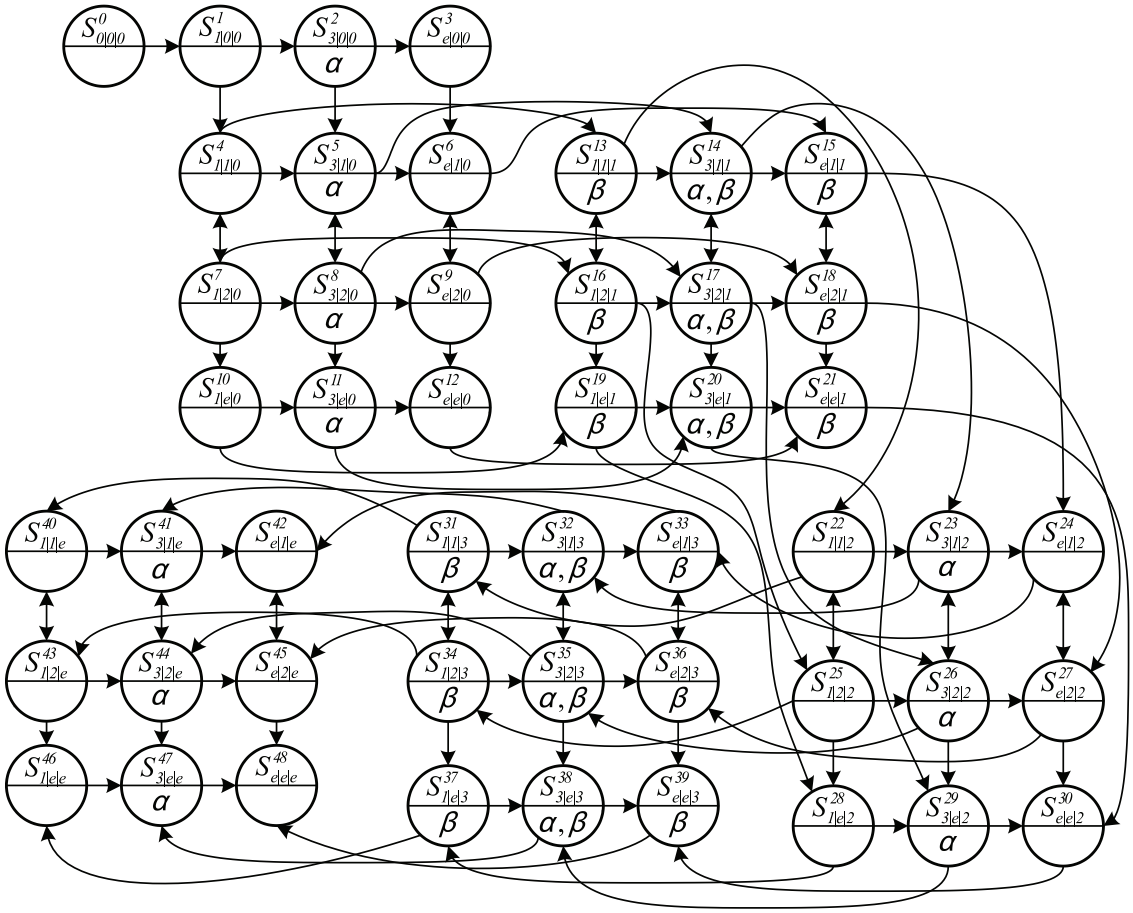


*Рис. 2. Асинхронная композиция структур Крипке*

Пусть необходимо проверить, удовлетворяет ли представленный алгоритм свойству: «Любому обращению к семафору предшествует операция его инициализации». Данное требование может быть перефразировано следующим образом: «Когда-либо в будущем выполнится операция инициализация семафора, а до тех пор не будет ни одного обращения к семафору, или же обращение к семафору не произойдет никогда». На языке темпоральной логики линейного времени свойство будет иметь вид $(\neg\beta)U\alpha \vee G(\neg\beta)$.

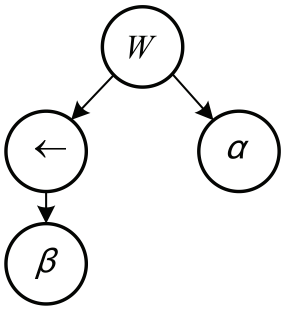


*Рис. 3. Оптимизированное дерево грамматического разбора формулы темпоральной логики линейного времени*

На этапе построения дерева грамматического разбора с помощью предложенного в работе [4] алгоритма к спецификации могут быть применены стандартные преобразования, после чего будет получена оптимизированная структура, представленная на рис. 3.

Использование предложенного выше алгоритма приведет к построению базы знаний, содержащей двести шестнадцать фактов. Ниже представлены факты, порождаемые вершинами композиции с порядковыми номерами 0–9 и 48. Остальные элементы имеют аналогичный вид.

1) $1 \to Parent(s_0, s_1) \wedge \neg Event(s_0, \alpha) \wedge \neg Event(s_0, \beta)$.

2) $1 \to Parent(s_1, s_2) \wedge Parent(s_1, s_4) \wedge \neg Event(s_1, \alpha) \wedge \neg Event(s_1, \beta)$.

3) $1 \to Parent(s_2, s_3) \wedge Parent(s_2, s_5) \wedge Event(s_2, \alpha) \wedge \neg Event(s_2, \beta)$.

4) $1 \to Parent(s_3, s_6) \wedge \neg Event(s_3, \alpha) \wedge \neg Event(s_3, \beta)$.

5) $1 \to Parent(s_4, s_5) \wedge Parent(s_4, s_7) \wedge Parent(s_4, s_{13}) \wedge \neg Event(s_4, \alpha) \wedge \neg Event(s_4, \beta)$.

6) $1 \to Parent(s_5, s_6) \wedge Parent(s_5, s_8) \wedge Parent(s_5, s_{14}) \wedge Event(s_5, \alpha) \wedge \neg Event(s_5, \beta)$.

7) $1 \to Parent(s_6, s_9) \wedge Parent(s_6, s_{15}) \wedge \neg Event(s_6, \alpha) \wedge \neg Event(s_6, \beta)$.

8) $1 \to Parent(s_7, s_8) \wedge Parent(s_7, s_{10}) \wedge Parent(s_7, s_{16}) \wedge \neg Event(s_7, \alpha) \wedge \neg Event(s_7, \beta)$.

9) $1 \to Parent(s_8, s_9) \wedge Parent(s_8, s_{11}) \wedge Parent(s_8, s_{17}) \wedge Event(s_8, \alpha) \wedge \neg Event(s_8, \beta)$.

10) $1 \to Parent(s_9, s_{12}) \wedge Parent(s_9, s_{18}) \wedge \neg Event(s_9, \alpha) \wedge \neg Event(s_9, \beta)$.

11) $1 \to Parent(s_7, s_4) \wedge Parent(s_8, s_5) \wedge Parent(s_9, s_6)$.

12) $1 \to \neg Event(s_{48}, \alpha) \wedge \neg Event(s_{48}, \beta) \wedge End(s_{48})$.

Кроме того, в результате анализа дерева разбора спецификации (с учетом необходимости проверки выводимости отрицания требования) в базу знаний будут добавлены следующие утверждения.

1) $1 \to \neg P_1(\chi) \vee Event(\chi, \beta)$.

2) $1 \to \neg P_2(\chi) \vee \neg Event(\chi, \alpha)$.

3) $1 \to \neg P_3(\chi) \vee P_1(\chi)$.

4) $1 \to P_4(\chi) \vee P_2(\chi) \vee P_3(\chi)$.

5) $1 \to P_4(\chi) \vee P_2(\chi) \vee \neg Parent(\chi, \gamma) \vee \neg P_4(\gamma)$.

Выводимым утверждением в таком случае будет посылка вида $1 \to P_4(s_0)$.

Применение метода деления дизъюнктов [5] позволит установить выводимость целевого утверждения за четыре итерации. Более того, найденная траектория вывода ($s_0 \to s_1 \to s_4 \to s_{13}$) будет являться контрпримером – последовательностью состояний алгоритма, приводящей к нарушению спецификации. Анализ трассы показывает, что возможной является ситуация, когда из-за возникновения задержки после второй операции основного процесса (например, вследствие длительной инициализации нескольких обработчиков запросов) экземпляр третьего процесса попытается использовать непроинициализированный семафор. Последствия такой ситуации непредсказуемы, поэтому ее следует считать недопустимой.

**Заключение**

Таким образом, предлагаемый способ формирования базы знаний на основе структуры Крипке, эквивалентной исследуемому алгоритму, и спецификации, описывающей предъявляемые требования, позволяет использовать на заключительном этапе верификации математический аппарат теории логического вывода. Данное решение за счет высокой степени параллелизма этого аппарата позволяет значительно сократить общее время формальной проверки корректности алгоритмов и программ.

Асимптотическая сложность способа определяется как $O(M + K)$, где $M$ – число связей между состояниями модели (в большинстве случаев сравнимо с числом состояний модели), а $K$ – число лексем спецификации. Кроме того, предлагаемый способ не зависит от конкретных методов логического вывода. Вследствие универсальности полу-

ченной базы знаний рекомендуемый к применению модифицированный ускоренный метод деления дизъюнктов может быть заменен любым подходящим методом, оперирующим знаниями в логике предикатов.

Представленный способ был использован при разработке программной системы для верификации параллельных алгоритмов с помощью ускоренного метода логического вывода.

## Список литературы

1. Карпов Ю.Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010. – 560 с.

2. Кларк Э. Верификация моделей программ. Model Checking / Э. Кларк, О. Грамберг, Д. Пелед. – М.: МЦНМО, 2002. – 416 с.

3. Кулямин В.В. Методы верификации программного обеспечения / Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы», 2008. – 117 с.

4. Мельцов В.Ю., Чистяков Г.А. Эффективный метод построения оптимизированного дерева грамматического разбора формул темпоральной логики линейного времени // Вестник ТГТУ. – 2012. – № 4. – С. 819–828.

5. Страбыкин Д.А. Логический вывод в системе обработки знаний. – СПб: Изд-во СПбГЭТУ, 1998. – 164 с.

6. Baier C. Principles of Model Checking / C. Baier, J.-P. Katoen. – Cambridge: The MIT Press, 2008. – 994 p.

## References

1. Karpov Yu.G. *Model Checking: verifikatsia parallelnyh i raspredelennyh system* [Model Checking: Verification of Parallel and Distributed Systems]. St.. Petersburg, BHV-Petersburg, 2010. 560 p.

2. Klark E., Gramberg O., Peled D. *Verificatsia modeley programm. Model Checking* [Verification of Program Models. Model Checking]. Moscow, MCNMO Press, 2002. 416 p.

3. Kulyamin V.V. *Metody verificatsii programmnogo obespechenia* [Methods for Software Verification]. All-Russian competitive selection of an overview and analytical articles on priority «Information systems and telecommunication», 2008. 117 p.

4. Meltsov V.Yu., Chistyakov G.A. Effective Method for Constructing Optimized Parse Tree of Temporal Logic Formulas of Linear Time. *Vestnic TSTU*, 2012, no. 4, pp. 819–828.

5. Strabykin D.A. *Logicheskiy vivod v sisteme obrabotki znaniy* [The Logical Inference in the Knowledge Processing]. St. Petersburg, SPbGETU Press, 1998. 164 p.

6. Baier C., Katoen J.-P. *Principles of Model Checking*. Cambridge, The MIT Press, 2008. 994 p.

**Рецензенты:**

Пономарев В.И., д.т.н., профессор, директор закрытого акционерного общества «НПП «Знак», г. Киров;

Вечтомов Е.М., д.ф.-м.н., профессор, зав. кафедрой «Алгебра и дискретная математика», ФГБОУ ВПО «ВятГГУ», г. Киров.

Работа поступила в редакцию 11.07.2013.