MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 27, NO. 4, 2020
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

On the Model Checking Problem for Some Extension of CTL*

A. R. Gnatenko[1], V. A. Zakharov[1,2]

DOI: 10.18255/1818-1015-2020-4-428-441

[1]Research University Higher School of Economics, 20 Myasnitskaya, Moscow 101000, Russia.
[2]Ivannikov Institute for System Programming of the RAS, 25 Alexander Solzhenitsyn, Moscow 109004, Russia.

MSC2020: 68Q45
Research article
Full text in Russian

Received November 16, 2020
After revision December 1, 2020
Accepted December 16, 2020

Sequential reactive systems include programs and devices that work with two streams of data and convert input streams of data into output streams. Such information processing systems include controllers, device drivers, computer interpreters. The result of the operation of such computing systems are infinite sequences of pairs of events of the request–response type, and, therefore, finite transducers are most often used as formal models for them. The behavior of transducers is represented by binary relations on infinite sequences, and so, traditional applied temporal logics (like HML, LTL, CTL, mu-calculus) are poorly suited as specification languages, since omega-languages, not binary relations on omega-words are used for interpretation of their formulae. To provide temporal logics with the ability to define properties of transformations that characterize the behavior of reactive systems, we introduced new extensions of these logics, which have two distinctive features: 1) temporal operators are parameterized, and languages in the input alphabet of transducers are used as parameters; 2) languages in the output alphabet of transducers are used as basic predicates. Previously, we studied the expressive power of new extensions Reg-LTL and Reg-CTL of the well-known temporal logics of linear and branching time LTL and CTL, in which it was allowed to use only regular languages for parameterization of temporal operators and basic predicates. We discovered that such a parameterization increases the expressive capabilities of temporal logic, but preserves the decidability of the model checking problem. For the logics mentioned above, we have developed algorithms for the verification of finite transducers. At the next stage of our research on the new extensions of temporal logic designed for the specification and verification of sequential reactive systems, we studied the verification problem for these systems using the temporal logic Reg-CTL*, which is an extension of the Generalized Computational Tree Logics CTL*. In this paper we present an algorithm for checking the satisfiability of Reg-CTL* formulae on models of finite state transducers and show that this problem belongs to the complexity class ExpSpace.

Keywords: reactive system; model checking; finite state transducer; temporal logic; regular language; specification

INFORMATION ABOUT THE AUTHORS

Anton Romanovich Gnatenko	orcid.org/0000-0003-1499-2090. E-mail: gnatenko.cmc@gmail.com student.
Vladimir Anatolyevich Zakharov correspondence author	orcid.org/0000-0002-3794-9565. E-mail: zakh@cs.msu.ru Dr. of Science, professor.

Funding: The reported study was funded by RFBR, project number 18-01-00854.

For citation: A. R. Gnatenko and V. A. Zakharov, "On the Model Checking Problem for Some Extension of CTL*", *Modeling and analysis of information systems*, vol. 27, no. 4, pp. 428-441, 2020.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 27, № 4, 2020
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

О задаче верификации моделей программ для одного расширения логики CTL*

А. Р. Гнатенко[1], В. А. Захаров[1,2]

DOI: 10.18255/1818-1015-2020-4-428-441

[1]Национальный исследовательский университет «Высшая школа экономики», ул. Мясницкая, д. 20, г. Москва, 101000 Россия.
[2]Институт системного программирования имени В. П. Иванникова РАН, А. Солженицына, д. 25, г. Москва, 109004 Россия.

УДК 519.7
Научная статья
Полный текст на русском языке

Получена 16 ноября 2020 г.
После доработки 1 декабря 2020 г.
Принята к публикации 16 декабря 2020 г.

К последовательным реагирующим системам относятся программы и устройства, которые работают с двумя потоками данных и осуществляют преобразование входных потоков данных в выходные потоки. К числу таких систем обработки информации относятся контроллеры, драйверы устройств, компьютерные интерпретаторы. Результатом работы таких вычислительных систем являются бесконечные последовательности пар событий типа запрос–отклик, и поэтому в качестве математических моделей для них наиболее часто используются конечные автоматы-преобразователи. Поведение автоматов-преобразователей представлено бинарными отношениями на бесконечных последовательностях, и традиционные прикладные темпоральные логики (HML, LTL, CTL, mu-исчисление) плохо подходят для этой цели, поскольку для интерпретации их формул используются omega-языки, а не бинарные отношения на omega-словах. Чтобы предоставить темпоральным логикам возможность определять свойства преобразований, которые характеризуют поведение реагирующих систем, мы ввели новые расширения этих логик, имеющие две отличительные особенности: 1) темпоральные операторы в расширениях этих логик параметризованы, и в качестве параметров используются языки в входном алфавите автоматов-преобразователей; 2) в качестве базовых предикатов используются языки в выходном алфавите автоматов-преобразователей. Ранее нами были исследованы выразительные возможности новых расширений Reg-LTL и Reg-CTL известных темпоральных логик линейного и ветвящегося времени LTL и CTL, в которых для параметризации темпоральных операторов и задания базовых предикатов разрешалось использовать только регулярные языки. Мы обнаружили, что такая параметризация увеличивает выразительные возможности темпоральной логики, но сохраняет разрешимость задачи проверки выполнимости формул на конечных моделях. Для указанных выше логик нами были разработаны алгоритмы верификации конечных автоматов-преобразователей. На следующем этапе изучения новых расширений темпоральной логики, предназначенных для спецификации и верификации последовательных реагирующих систем, мы обратились к задаче верификации этих систем с использованием темпоральной логики Reg-CTL*, которая является расширением обобщенной логики деревьев вычислений CTL*. В этой статье описан алгоритм проверки выполнимости формул Reg-CTL* на моделях конечных автоматов-преобразователей и показано, что эта задача принадлежит классу сложности ExpSpace.

Ключевые слова: реагирующая система; верификация моделей программ; конечный автомат-преобразователь; темпоральная логика; регулярный язык; спецификация

ИНФОРМАЦИЯ ОБ АВТОРАХ

Антон Романович Гнатенко | orcid.org/0000-0003-1499-2090. E-mail: gnatenko.cmc@gmail.com
студент.

Владимир Анатольевич Захаров | orcid.org/0000-0002-3794-9565. E-mail: zakh@cs.msu.ru
автор для корреспонденции | доктор физ.-мат. наук, профессор.

Финансирование: Работа выполнена при финансовой поддержке гранта РФФИ N 18-01-00854.

Для цитирования: A. R. Gnatenko and V. A. Zakharov, “On the Model Checking Problem for Some Extension of CTL*”, *Modeling and analysis of information systems*, vol. 27, no. 4, pp. 428-441, 2020.

Введение

Конечные автоматы широко используются в информатике как модели последовательных вычислительных систем. В частности, автоматы-преобразователи с конечным числом состояний служат подходящей формальной моделью для различных программных и аппаратных систем, таких как контроллеры, системные драйверы, сетевые коммутаторы, интерпретаторы программ, которые работают с двумя потоками данных. Эти устройства и программы получают потоки данных на свои входы и преобразуют их в выходные потоки управляющих сигналов (команд, данных). К аппаратным устройствам этого типа относятся адаптеры, сетевые коммутаторы, контроллеры. В программировании автоматы-преобразователи используются в качестве формальных моделей различных программ и протоколов, которые обрабатывают строки символов, потоки команд и данных (см. [1—3]). Такие информационные системы объединяются под общим названием *последовательные реагирующие системы*. Их вычисления проводятся в дискретном времени; на каждом этапе вычислений система получает управляющий сигнал (или часть входных данных) из среды и генерирует (выполняет, выводит) последовательность действий (или часть выходных данных) в ответ. Эти выходные действия и порядок их выполнения зависят от всех предыдущих управляющих сигналов.

В этой статье в центре внимания будут находиться реагирующие системы с ограниченной памятью, которые работают в оперативном (on-line) режиме. В статье [4] мы рассмотрели ряд примеров, которые показывают, что автоматы-преобразователи с конечным числом состояний являются простой и адекватной математической моделью для представления последовательных реагирующих систем во многих приложениях. Поведение таких систем характеризуется не набором последовательностей событий, а взаимосвязью между двумя последовательностями событий. Типичное свойство поведения, требующее проверки, состоит в том, что для каждого входного слова, подпадающего под некоторый шаблон допустимого воздействия внешней среды реагирующая система всегда вычисляет выходное слово, которое подпадает под другой шаблон, определяющий ожидаемый тип отклика системы. Требования такого рода можно сформулировать с помощью темпоральной логики, адаптированной для рассуждений о парах последовательностей событий.

Такие темпоральные логики были введены и изучены в [4—6]. В статье [5] новое расширение $\mathcal{LP}$-LTL темпоральной логики линейного времени LTL было представлено как формальный язык для спецификации поведения последовательных реагирующих систем. В логике $\mathcal{LP}$-LTL темпоральные операторы параметризованы множествами слов (языками), которые представляют допустимые потоки управляющих сигналов, воздействующих на реагирующую систему. Базовые предикаты в $\mathcal{LP}$-LTL также являются языками в алфавите основных действий преобразователя; они представляют собой ожидаемую реакцию преобразователя на указанные воздействия окружающей среды. В статье [5] авторы исследовали задачу проверки модели для регулярного фрагмента *Reg*-LTL этой логики, когда только регулярные языки используются в качестве базовых предикатов и параметров темпоральных операторов. Было показано, что задача проверки выполнимости формул *Reg*-LTL на моделях конечных автоматов-преобразователей разрешима за двойное экспоненциальное время. В статье [4] мы изучили выразительные возможности логики $\mathcal{LP}$-LTL и сравнили ее с другими темпоральными логиками, используемыми для спецификации поведения реагирующих систем. В статье [6] был предложен табличный алгоритм верификации моделей для логики *Reg*-CTL, которая является регулярным расширением темпоральной логики деревьев вычислений CTL.

Здесь мы рассматриваем более общий язык спецификаций последовательных реагирующих систем, являющийся расширением обобщенной логики деревьев вычислений CTL^*, фрагментами которой являются упомянутые выше темпоральные логики LTL и CTL. Как показано в [7], алгоритм решения задачи верификации моделей для логики CTL^* получается комбинацией алгоритмов

решения аналогичной задачи для LTL и CTL. Однако в случае, когда темпоральные операторы параметризованы регулярными выражениями, ситуация оказывается более сложной и требует отдельного исследования. Это исследование проводится в настоящей статье. Мы вводим в рассмотрение расширенную логику $\mathcal{LP}$-CTL^* и её регулярный фрагмент Reg-CTL^* и решаем для этого фрагмента задачу верификации моделей программ. Основные результаты, полученные в данной работе, таковы:

1. Предложено новое расширение темпоральной логики деревьев вычислений $\mathcal{LP}$-CTL^* и определена формальная семантика этой логики на деревьях вычислений конечных автоматов-преобразователей.
2. Для этой логики разработан теоретико-автоматный алгоритм верификации моделей автоматов-преобразователей, использующий объём памяти, экспоненциальный относительно размеров проверяемых автомата и формулы.
3. Показано, что указанная задача верификации автоматов-преобразователей для логики Reg-CTL^* является PSpace-трудной и принадлежит классу сложности ExpSpace.

Полученные результаты основаны на методах, предложенных в работах [5, 6, 8], и продолжают направление исследований, инициированное и развитое в этих статьях. В разделе 1 мы вводим формальное определение автоматов-преобразователей, а также определяем синтаксис и семантику языка темпоральной логики Reg-CTL^*. Далее приведен краткий обзор некоторых ранее известных расширений обычных темпоральных логик. В разделе 3 описан алгоритм решения задачи верификации моделей для формул логики Reg-CTL^* и установлены оценки его сложности. В заключении мы сравниваем полученные в статье результаты с результатами решения аналогичной задачи для других расширений темпоральной логики, а также обсуждаем дальнейшие направления исследований в этой области.

1. Модели реагирующих систем и их спецификация

Последовательные реагирующие системы обработки информации, такие как адаптеры, контроллеры, драйверы устройств, интерпретаторы программ, сетевые коммутаторы работают с двумя потоками данных и преобразуют входные потоки данных (управляющих сигналов, команд) в выходные потоки данных (управляющих сигналов, команд). Для моделирования вычислений таких систем можно использовать конечные автоматы-преобразователи.

Пусть заданы конечные алфавиты $\mathcal{C}$ и $\mathcal{A}$. Элементы множества $\mathcal{C}$ называются *входными сигналами*, а элементы множества $\mathcal{A}$ — *элементарными действиями*. Обозначим записью $\mathcal{A}^*$ множество всех конечных слов над алфавитом $\mathcal{A}$, которые называются *составными действиями*. Для двух составных действий u и v мы пишем uv для их конкатенации и используем обозначение ε для пустого слова. Записи $Reg(\mathcal{C})$ и $Reg(\mathcal{A})$ будут обозначать семейства всех регулярных языков над алфавитами $\mathcal{C}$ и $\mathcal{A}$ соответственно.

Конечный *автомат-преобразователь* (или просто преобразователь) над алфавитами $\mathcal{C}$ и $\mathcal{A}$ — это пятёрка $\pi = (Q, \mathcal{C}, \mathcal{A}, q_{init}, T)$, где Q — конечное множество *управляющих состояний*, $q_{init} \in Q$ — *начальное состояние*, а $T \subseteq Q \times \mathcal{C} \times Q \times \mathcal{A}^*$ — *отношение переходов*. При этом требуется, чтобы отношение T было тотальным, т.е. чтобы для любого состояния $q \in Q$ и любого входного сигнала $c \in \mathcal{C}$ существовали такие $q' \in Q$ и $h \in \mathcal{A}^*$, что $(q, c, q', h) \in T$. *Размером* преобразователя π мы будем называть размер табличного описания его отношения переходов. Эта величина обозначается записью $|\pi|$.

Четвёрка (q, c, q', h) называется *переходом*; он означает, что преобразователь, находясь в состоянии q и получив входной сигнал c, может перейти в состояние q' и выполнить последовательность элементарных действий h. Таким образом, отношение переходов моделирует программу реагирующей системы. *Траектория* преобразователя π из состояния q_0 — это бесконечная последовательность переходов $tr = \{\tau_i\}_{i \geqslant 1}$, где $\tau_i = (q_{i-1}, c_i, q_i, h_i) \in T$ для всех i, $1 \leqslant i \leqslant n$. Последовательность

пар $(c_1, h_1), (c_2, h_2), \ldots$, которыми помечены переходы траектории tr, представляет собой всю информацию о работе реагирующей системы, доступную внешнему наблюдателю. Множество всех траекторий из состояния q обозначается записью $Tr_\pi(q)$. Для краткости вместо $Tr_\pi(q_{init})$ условимся использовать обозначение Tr_π.

Наряду с бесконечными траекториями мы будем также рассматривать конечные траектории. Конечной траекторией из состояния q называется любой конечный префикс некоторой траектории из этого состояния. Множество конечных траекторий из состояния q обозначим записью $Fin\ Tr_\pi(q)$.

Поведение преобразователя $\pi = (Q, \mathcal{C}, \mathcal{A}, q_{init}, T)$ описывается *графом вычислений* — ориентированным графом $\Gamma_\pi = (V_\pi, E_\pi)$ с множествами вершин $V = (Q \times \mathcal{A}^*)$ и помеченных дуг $E \subseteq V \times \mathcal{C} \times V$, согласованных с отношением переходов T следующим образом: для любой пары вершин $u' = (q', s')$ и $u'' = (q'', s'')$, для произвольного входного сигнала c имеет место соотношение

$$(u', c, u'') \in E \iff \exists h \in \mathcal{A}^* : (q', c, q'', h) \in T \wedge s'' = s'h.$$

Вершину графа вычислений (q_{init}, ε) будем называть *начальной* и обозначать записью v_{init}. Между траекториями преобразователя и путями в графе вычислений имеется соответствие, которое важно для последующих построений. Каждой паре, включающей траекторию $tr = \{(q_{i-1}, c_i, q_i, d_i)\}_{i \geqslant 1}$ из состояния q_0 и составное действие s_0, сопоставим путь $\rho = \{(v_{i-1}, c_i, v_i)\}_{i \geqslant 1}$ в графе вычислений Γ_π, где $v_0 = (q_0, s_0)$ и $v_i = (q_i, s_{i-1}h_i)$ для всех $i \geqslant 1$. Для каждой конечной траектории $tr = \{(q_{i-1}, c_i, q_i, h_i)\}_{i=1}^{k} \in Fin\ Tr_\pi$ обозначим записью $v_{init}[tr]$ вершину (q_k, h) графа Γ_π, где $h = h_1 h_2 \ldots h_k$. Мы будем использовать традиционное обозначение $\rho|^k$ для бесконечного суффикса пути $\rho = \{(v_{i-1}, c_i, v_i)\}_{i \geqslant 1}$, начинающегося из вершины v_k.

Задача верификации вычислительной системы состоит в том, чтобы выяснить обладает ли поведение системы требуемыми свойствами. Выбрав конечные преобразователи в качестве моделей последовательных реагирующих систем, мы можем полагать, что поведение преобразователя полностью характеризуется множеством его начальных траекторий Tr_π (или, ввиду описанного выше соответствия, множеством путей в графе Γ_π, начинающихся в вершине v_{init}). Таким образом, «свойством поведения» преобразователя, может считаться любое свойство его траекторий, т.е. произвольное подмножество $Property \subseteq Tr_\pi$.

Хорошо известно, что свойства бесконечных последовательностей состояний системы, также как и свойства систем переходов, наподобие графа вычислений, удобно описывать формулами темпоральных логик (LTL, CTL, CTL^*). Однако следует подчеркнуть, что траектория преобразователя содержит в себе не только последовательность состояний, но также входную и выходную последовательности, связь между которыми осуществляется реагирующей системой. Поэтому для спецификации поведения преобразователя необходим язык спецификаций, позволяющий формально описывать эту связь.

Желаемого эффекта можно достигнуть, снабдив модальные операторы темпоральной логики параметрами; в качестве параметров в общем случае разрешается использовать любой язык в алфавите $\mathcal{C}$. Далее мы определим синтаксис и семантику темпоральной логики $\mathcal{LP}$-CTL^*, которая является расширением обобщенной логики деревьев вычислений CTL^*, и будем исследовать наиболее интересный с практической точки зрения фрагмент Reg-CTL^* введенного расширения. В формулах Reg-CTL^* параметрами темпоральных операторов могут быть только регулярные языки. Логика обобщает Reg-CTL^* языки спецификаций Reg-LTL и Reg-CTL поведения автоматов-преобразователей, изученные в наших предыдущих статьях [4—6, 8].

Поведение преобразователя считается правильным, если реакция системы на определенные сигналы, поступающие из внешней среды, соответствует ожиданию. Желаемый тип этой реакции может быть описан множеством допустимых последовательностей элементарных действий, т.е. некоторым языком в алфавите действий $\mathcal{A}$. Любой такой язык $P \subseteq \mathcal{A}^*$ называется *базовым*

предикатом. При этом нас может интересовать отклик системы на последовательности входных сигналов, подпадающих под некоторый *шаблон поведения окружения*, который задан некоторым языком $L \subseteq C^*$.

Пусть задано семейство $\mathcal{L}$ шаблонов поведения окружения и семейство $\mathcal{P}$ базовых предикатов. Формулы логики $\mathcal{LP}$-CTL^* подразделяются на два типа — *формулы состояния* и *формулы пути*, — и определяются следующим образом:

1) всякий базовый предикат $P \in \mathcal{P}$ — это формула состояния;
2) если $\varphi_1,\ \varphi_2$ — формулы состояния, то $\neg\varphi_1$ и $\varphi_1 \wedge \varphi_2$ — формулы состояния;
3) если ψ — формула пути, то $\mathbf{A}\psi$ и $\mathbf{E}\psi$ — формулы состояния;
4) если φ — формула состояния, то φ — формула пути;
5) если $\psi_1,\ \psi_2$ — формулы пути, то $\neg\psi_1$ и $\psi_1 \wedge \psi_2$ — формулы пути;
6) если ψ, ψ_1, ψ_2 — формулы пути, $c \in C$, и $L \in \mathcal{L}$, то $\mathbf{X}_c\psi$ и $\psi_1\,\mathbf{U}_L\psi_2$ — формулы пути.

Формулы $\mathcal{LP}$-CTL^* интерпретируются на графах вычислений. Пусть Γ_π — граф вычислений преобразователя $\pi = (Q, C, \mathcal{A}, q_{init}, T)$. Запись $\Gamma_\pi, v \models \varphi$ обозначает, что формула состояния φ выполняется в вершине v графа Γ_π. Запись $\Gamma_\pi, \rho \models \psi$, в свою очередь, означает, что формула пути ψ выполняется на пути ρ этого графа.

Пусть $v = (q, s)$ — вершина графа вычислений Γ_π, а $\rho = (v_0, c_1, v_1), (v_1, c_2, v_2), \ldots$ — путь в этом графе. Отношение выполнимости $\models$ определяется индуктивно следующим образом:

1) $\Gamma_\pi, v \models P \iff s \in P$ для всякого базового предиката $P \in \mathcal{P}$;
2) $\Gamma_\pi, v \models \neg\varphi \iff$ неверно, что $\Gamma_\pi, v \models \varphi$;
3) $\Gamma_\pi, v \models \varphi_1 \wedge \varphi_2 \iff \Gamma_\pi, v \models \varphi_1$ и $\Gamma_\pi, v \models \varphi_2$;
4) $\Gamma_\pi, v \models \mathbf{E}\,\varphi \iff$ существует путь ρ_v из вершины v, для которого $\Gamma_\pi, \rho \models \varphi$;
5) для любой формулы состояния $\varphi:\ \Gamma_\pi, \rho \models \varphi \iff \Gamma_\pi, v_0 \models \varphi$;
6) $\Gamma_\pi, \rho \models \neg\psi \iff$ неверно, что $\Gamma_\pi, \rho \models \psi$;
7) $\Gamma_\pi, \rho \models \psi_1 \wedge \psi_2 \iff \Gamma_\pi, \rho \models \psi_1$ и $\Gamma_\pi, \rho \models \psi_2$;
9) $\Gamma_\pi, \rho \models \mathbf{X}_c\varphi_1 \iff c = c_1$ и $\Gamma_\pi, \rho|^1 \models \varphi_1$;
10) $\Gamma_\pi, \rho \models \varphi_1\,\mathbf{U}_L\varphi_2 \iff \exists i \geqslant 0:\ c_1c_2\ldots c_i \in L$, где $\Gamma_\pi, \rho|^i \models \varphi_2$, причём $\forall j,\ 0 \leqslant j < i$, если $c_1c_2\ldots c_j \in L$, то $\Gamma_\pi, \rho|^j \models \varphi_1$.

Формула $\mathbf{X}_c\varphi_1$ означает, что свойство φ_1 окажется выполненным на следующем шаге, причём этот шаг будет сделан в ответ на получение сигнала c. Формула $\varphi_1\,\mathbf{U}_L\varphi_2$ утверждает, что свойство φ_1 будет выполняться каждый раз, когда поведение окружения соответствует шаблону L, до тех пор, пока не выполнится свойство φ_2. При этом φ_2 также должно выполниться в момент времени, когда последовательность входных сигналов удовлетворяет шаблону. Для удобства записи можно определить другие темпоральные операторы, а также ввести квантор всеобщности для путей:

- Оператор **F**uture: $\mathbf{F}_L\varphi \equiv \mathbf{true}\,\mathbf{U}_L\varphi$: свойство φ выполнилось когда-нибудь при получении последовательности сигналов, удовлетворяющей шаблону L.
- Оператор **G**obally: $\mathbf{G}_L\varphi \equiv \neg\,\mathbf{F}\,\neg\varphi$, двойственный оператору **F**.
- Квантор всеобщности для путей: $\mathbf{A}\psi \equiv \neg\,\mathbf{E}\neg\psi$.

Если $v = v_{init}$, то вместо $\Gamma_\pi, v \models \varphi$ будем использовать запись $\pi \models \varphi$. Задача верификации преобразователя π относительно спецификации $\varphi \in \mathcal{LP}$-CTL^* заключается в проверке соотношения $\pi \models \varphi$.

В общем случае $\mathcal{L}$ и $\mathcal{P}$ могут быть произвольными семействами языков, и эта свобода выбора может приводить к неразрешимости задачи верификации преобразователей. Действительно, пусть $C = \mathcal{A} = \{a, b\}$, а преобразователь π обладает единственным состоянием $q_{init} = q$ и отношением переходов $T = \{(q, c, c, q):\ c \in C\}$. Тогда для любых двух языков $U, V \subseteq C^*$ верно следущее соотношение:

$$\pi \models \mathbf{EF}_U V \iff U \cap V \neq \varnothing.$$

Так как проблема проверки пустоты пересечения двух контекстно-свободных языков неразрешима [9], задача верификации преобразователей оказывается неразрешимой, если в качестве шаблонов поведения окружения и базовых предикатов разрешается использовать произвольные контекстно-свободные языки. Чтобы избежать этой ситуации, в качестве $\mathcal{L}$ и $\mathcal{P}$ мы будем использовать семейства регулярных языков. Выбор этого класса языков оправдан еще и тем, что регулярные языки широко используются в практических приложениях, и для работы с ними существуют эффективные и доступные программные средства.

Полагая $\mathcal{L} = Reg(C)$ и $\mathcal{P} = Reg(\mathcal{A})$, мы получим фрагмент логики $\mathcal{LP}$-CTL^*, который условимся обозначать записью Reg-CTL^*. В следующем разделе мы сравним этот фрагмент введенной нами логики с другими расширениями темпоральных логик, которые были введены и исследованы ранее. Далее для логики Reg-CTL^* будет описан алгоритм решения задачи верификации преобразователей.

2. Другие расширения темпоральных логик

Всесторонний анализ темпоральной логики LTL как формального языка для описания поведения вычислительных систем был впервые проведен в работе [10]. Вскоре после этого было предпринято несколько попыток улучшить выразительные возможности этой темпоральной логики. Модификации проводились в основном по двум направлениям: 1) добавление новых выразительных средств (кванторов, модальностей и т.д.) и 2) изменение семантики темпоральных операторов.

Например, авторы [11] снабдили синтаксис LTL количественной оценкой основных предикатов и обнаружили, что выразительные возможности введенного таким образом количественного расширения значительно превосходят описательные возможности простого LTL. С другой стороны, в статье [12] был предложен метод введения новых темпоральных операторов с использованием праволинейных грамматик. Слова, порождаемые этими грамматиками, определяют шаблоны, по которым проверяется выполнимость формул в рамках временного оператора. Аналогично, в статьях [13—15] было показано, что шаблоны для описания семантики новых темпоральных операторов могут быть определены с помощью конечных автоматов и регулярных выражений. Идея снабжения темпоральных операторов некоторыми параметрами не нова: почти такая же параметризация темпоральных операторов, как в этой статье, была введена в [16] для динамически расширяемого LTL. С тех пор было предпринято несколько попыток такого рода объединить регулярные языки и временные операторы (например, см. [17, 18] среди последних). Почти во всех этих случаях было обнаружено, что выразительные возможности этих расширений увеличиваются и становятся эквивалентными таковым для логики $S1S$, в то время как проблема проверки выполнимости для этих логик остается PSpace-полной.

Вводя новое расширение $\mathcal{LP}$-CTL^* языка темпоральной логики деревьев вычислений, мы не стремились всего лишь улучшить выразительность языка CTL^*. Наша цель состояла в том, чтобы предложить адекватный язык для описания поведения реагирующих систем, моделируемых преобразователями. При вычислении преобразователя выполняется согласованное формирование двух последовательностей — последовательности входных сигналов и последовательности выходных действий. Поэтому отличительной особенностью семантики $\mathcal{LP}$-CTL^* является определенная синхронизация параметров темпоральных операторов (их интерпретация определяется последовательностью входных сигналов) и значений истинности базовых предикатов (они зависят от последовательности выходных действий). Можно сказать, что семантика нашей логики определяется на двухмерных трассах, тогда как во всех ранее известных параметризованных расширениях темпоральной логики использовались только одномерные трассы. Эта семантическая особенность существенно влияет на алгоритмы проверки выполнимости формул.

3. Верификация реагирующих систем относительно формул логики *Reg-CTL**

Задача верификации конечных преобразователей была впервые исследована в работе [5], где был предложен алгоритм проверки выполнимости $\pi \models \varphi$ формул логики *Reg-LTL* на множествах вычислений (траекторий) конечных преобразователей. Предложенная в этой работе логика *Reg-LTL* является линейным фрагментом рассматриваемой здесь логики *Reg-CTL**, который содержит только формулы вида $\mathbf{A}\varphi$, где φ — бескванторная формула пути. Алгоритм верификации, описанный в [5], основан на трансляции пары (π, φ) в такой автомат Бюхи $B(\pi, \varphi)$, что выполнимость $\pi \models \varphi$ имеет место тогда и только тогда, когда автомат $B(\pi, \varphi)$ допускает хотя бы одно слово. Из разрешимости проблемы пустоты для автоматов Бюхи следует разрешимость задачи проверки выполнимости формул логики *Reg-LTL* на конечных моделях.

В этом разделе мы развиваем идею алгоритма, предложенного в работе [5], и представляем более общий алгоритм верификации преобразователей относительно произвольных формул φ логики *Reg-CTL**, основанный на рекурсивном применении алгоритма из [5] к подформулам состояния формулы φ. Для описания этого алгоритма потребуется ввести несколько дополнительных определений.

3.1. Автоматы распознаватели над конечными и бесконечными словами

Детерминированный конечный *автомат-распознаватель* (или просто автомат) над алфавитом Σ — это пятерка $A = (Q, \Sigma, q_{init}, \delta, F)$, где Q — конечное множество состояний, $q_{init} \in Q$ — начальное состояние, $F \subset Q$ — множество *принимающих состояний*, а $\delta : Q \times \Sigma \longrightarrow Q$ — функция переходов. В отличие от отношения переходов преобразователя, функция переходов δ однозначно определяет следующее состояние по данной паре $(q, \sigma) \in Q \times \Sigma$. Эту функцию можно продолжить на всё множество Σ^* стандартным образом: $\delta(q, \varepsilon) = q$, и $\delta(q, \sigma x) = \delta(\delta(q, \sigma), x)$ для всех $q \in Q, \sigma \in \Sigma$ и $x \in \Sigma^*$. Автомат $A[x] = (Q, \Sigma, \delta(q_{init}, x), \delta, F)$ будем называть *сдвигом* A *на* слово x. Автомат A *принимает* слово x в том и только том случае, когда $\delta(q_{init}, x) \in F$. Говорят, что A *распознает* язык $L(A) = \{x : \delta(q_{init}, x) \in F\}$. Для краткости мы будем обозначать и автомат, и распознаваемый им язык одним и тем же символом A и записывать, например, $x \in A$ и $A = \varnothing$ вместо $x \in L(A)$ и $L(A) = \varnothing$.

Размером $|A|$ автомата A назовем размер табличного описания его функции переходов δ. Размером регулярного языка L мы называем величину $|L|$, равную размеру *минимального* автомата, распознающего этот язык.

Вычисления конечных автоматов можно определить также для бесконечных слов (ω-слов) в алфавите Σ. Множество всех таких слов обозначим записью Σ^ω.

Недетерминированный *обобщенный автомат Бюхи* [19] над алфавитом Σ — это пятерка $B = (Q, \Sigma, q_{init}, \Delta, \mathcal{F})$, где, в отличие от определения конечного автомата, $\Delta \subseteq Q \times \Sigma \times Q$ является *отношением переходов*, а $\mathcal{F} = \{F_1, \ldots, F_m\}$ — *допускающим правилом*, состоящим из компонент $F_i \subseteq Q, 1 \leqslant i \leqslant m$. Для произвольного заданного ω-слова $x = \sigma_0\sigma_1\sigma_2\cdots \in \Sigma^\omega$ *прогоном* автомата B на x назовем бесконечную последовательность состояний $q_0, q_1, q_2, \ldots$, в которой $q_0 = q_{init}$, и для всех $i, i \geqslant 0$, выполняется включение $(q_i, \sigma_i, q_{i+1}) \in \Delta$. Прогон $q_0, q_1, q_2, \ldots$ автомата B называется *допускающим*, если для каждого допускающего правила $F_i \in \mathcal{F}, 1 \leq i \leq m$, этого автомата имеется бесконечно много таких значений j, для которых $q_j \in F_i$. Автомат B *принимает* ω-слово $x \in \Sigma^\omega$ если и только если существует допускающий прогон B на этом слове. Понятие языка автомата B определяется аналогично случаю обычного автомата. Размером автомата B называется размер $|B|$ табличного описания отношения Δ.

Хорошо известны (см., например, [7]) эффективные процедуры проверки языка автомата Бюхи на пустоту, основанные на поиске в графе переходов автомата компонент сильной связности и завершающие вычисления за время, линейное относительно размера автомата $|B|$. Эти процедуры положены в основу теоретико-автоматного подхода к верификации вычислительных систем

для темпоральных логик [20]. Именно этого подхода мы будем придерживаться при построении алгоритма верификации преобразователей для логики $Reg\text{-}CTL^*$.

3.2. Трансляция формул $Reg\text{-}CTL^*$ в автоматы

Перейдем к построению алгоритма верификации. На его вход подается преобразователь $\pi = (Q, C, \mathcal{A}, q_{init}, T)$ и формула $\varphi \in Reg\text{-}CTL^*$. Алгоритм должен проверять, справедливо ли соотношение $\pi \models \varphi$ для этих входных данных.

Предположим, что формула φ имеет вид $\mathbf{E}\psi$, где ψ — бесквантерная формула. Задача верификации для этого частного случая решена в статье [5], где предложен алгоритм трансляции пары (π, φ) в автомат Бюхи $B(\pi, \varphi)$, язык которого непуст тогда и только тогда, когда $\pi \models \varphi$. При этом для каждого базового предиката P, используемого в формуле φ и представляющего собой язык в алфавите $\mathcal{A}$, строится распознающий его автомат A_P. Заметим, что этим способом можно решить задачу и в случае, когда φ имеет вид булевой комбинации формул вида $\mathbf{E}\psi$, т.е. для формул вида $\varphi = \Phi(\mathbf{E}\psi_1, \dots, \mathbf{E}\psi_n)$, где $\Phi(x_1, \dots, x_n)$ — булева формула.

В общем случае формула φ является булевой комбинацией подформул вида $\mathbf{E}\psi(\mathbf{E}\xi_1, \dots, \mathbf{E}\xi_m)$, где $\xi_1, \dots, \xi_m$ — некоторые формулы пути, а $\psi = \psi(P_1, \dots, P_n)$ — *бесквантерная формула пути*, в которой на места вхождения базовых предикатов $P_1, \dots, P_n$ подставлены формулы состояния $\mathbf{E}\xi_1, \dots, \mathbf{E}\xi_m$.

Рассмотрим произвольную конечную траекторию $tr \in Fin\,Tr_\pi$ преобразователя π. Она соответствует некоторому конечному пути $\rho = v_{init}, v_1, v_2, \dots v_t$ в графе вычислений Γ_π. Истинность формул состояния $\mathbf{E}\xi_i, 1 \leqslant i \leqslant m$, в вершине v_t определяется конечной траекторией tr, которую можно рассматривать как конечное слово tr в конечном алфавите переходов T преобразователя π. Предлагаемый нами алгоритм основан на построении для каждой формулы $\mathbf{E}\xi_i$ автомата над алфавитом T, принимающего те и только те конечные траектории, на которых выполняется формула $\mathbf{E}\xi_i$. Построенные автоматы затем используются при конструировании автомата Бюхи для формулы $\mathbf{E}\psi(\mathbf{E}\xi_1, \dots, \mathbf{E}\xi_m)$.

Введем необходимые понятия для описания этого построения. Будем верифицировать преобразователь $\pi = (Q, C, \mathcal{A}, q_{init}, T)$ и полагать, что формула φ имеет вид $\mathbf{E}\psi(\mathbf{E}\xi_1, \dots, \mathbf{E}\xi_m)$. *Проверочным автоматом* для пары (π, φ) называется такой конечный автомат $A(\pi, \varphi) = (Q_A, T, q^A_{init}, \delta_A, F_A)$ над алфавитом T переходов преобразователя π, что для любой конечной траектории $tr \in Fin\,Tr_\pi$ выполняется соотношение

$$\Gamma_\pi, v_{init}[tr] \models \varphi \iff tr \in A(\pi, \varphi).$$

Таким образом, построение автомата $A(\pi, \varphi)$ дает решение задачи верификации: $\pi \models \mathbf{E}\varphi$ тогда и только тогда, когда $\varepsilon \in A(\pi, \varphi)$. Верны следующие утверждения.

Теорема 1. *Существует алгоритм построения для любого конечного преобразователя π и формулы φ вида $\mathbf{E}\psi(\mathbf{E}\xi_1, \dots, \mathbf{E}\xi_m)$ проверочного автомата $A(\pi, \varphi)$ с использованием объема памяти $|\pi| \cdot 2^{\mathrm{poly}(|\varphi|)}$.*

Следствие 1. *Задача верификации конечных преобразователей относительно формул $Reg\text{-}CTL^*$ принадлежит классу сложности* ExpSpace.

Оставшаяся часть раздела посвящена доказательству теоремы 1.

Доказательство. Пусть $\pi = (Q, C, \mathcal{A}, q_{init}, T)$, а φ — формула состояния. Проведём доказательство теоремы индукцией по длине формулы φ.

Базис индукции состоит в доказательстве утверждения теоремы для случая $\varphi = P$, где P — базовый предикат. Заметим, что согласно определению семантики формул, $P \equiv \mathbf{E}P$. Поэтому мы будем считать, что $\varphi \equiv \mathbf{E}P$. Пусть $A_P = (Q_P, \mathcal{A}, p_{init}, \delta_P, F_P)$ — автомат, распознающий язык P. Положим

$$A(\pi, \mathbf{E}\varphi) = (Q \times Q_P, T, (q_{init}, p_{init}), v, Q \times F_P),$$

где функция переходов $\upsilon: Q \times Q_P \times T \longrightarrow Q \times Q_P$ такова, что для всех состояний $q \in Q$, $p \in Q_P$ и любого перехода преобразователя $\tau = (q, c, q', h) \in T$ выполняется равенство $\upsilon(q, p, \tau) = (q', \delta_P(p, h))$.

Базис индукции, таким образом, обоснован. Заметим, что множество состояний проверочного автомата $A(\pi, \varphi)$ имеет вид $Q \times Q_P$.

Перейдем теперь к обоснованию индуктивного перехода. Любая формула состояния является булевой комбинацией формул вида $\mathbf{E}\psi(\mathbf{E}\xi_1, \ldots, \mathbf{E}\xi_n)$, где $\psi(P_1, \ldots, P_n)$ — бесквантерная формула пути с базовыми предикатами $P_1, \ldots, P_n$. Достаточно, таким образом, рассмотреть случай $\varphi = \mathbf{E}\psi(\mathbf{E}\xi_1, \ldots, \mathbf{E}\xi_n)$. В силу отмеченной выше эквивалентности $P \equiv \mathbf{E}P$ все базовые предикаты P в формуле ψ можно заменить на $\mathbf{E}P$. Поэтому можно считать, что список $\mathbf{E}\xi_1, \ldots, \mathbf{E}\xi_n$ исчерпывает *все* подформулы состояния, встречающиеся в φ (не считая тех подформул состояния, которые являются подформулами $\xi_i, 1 \leqslant i \leqslant n$).

Предположение индукции, которое мы будем использовать, таково: для формул состояния $\psi_i, 1 \leqslant i \leqslant n$, существуют и могут быть эффективно построены проверочные автоматы $A(\pi, \xi_1), \ldots, A(\pi, \xi_n)$, причём для каждого $i, 1 \leqslant i \leqslant n$, множество состояний $A(\pi, \xi_i)$ имеет вид $Q \times Q_{P_{i_1}} \times \ldots \times Q_{P_{i_m}}$, где $P_{i_1}, \ldots, P_{i_m}$ — все базовые предикаты, встречающиеся в формуле ξ_i.

Каждый автомат $A(\pi, \xi_i)$ распознает язык над алфавитом T. Таким образом, формула $\mathbf{E}\xi_i$ является аналогом базового предиката над этим алфавитом. С этой точки зрения φ — бесквантерная формула, к которой можно применить алгоритм, разработанный в статье [5]. Этот алгоритм построит автомат Бюхи $B(\pi, \varphi)$, работающий в алфавите T и обладающий следующим свойством: для всякой конечной траектории $tr \in Fin\,Tr_\pi: \Gamma_\pi, \upsilon_{init}[tr] \models \varphi$ тогда и только тогда, когда у автомата $B(\pi, \varphi)$ есть допускающий прогон на бесконечной траектории tr_{inf}, для которой tr является префиксом. Мы приводим здесь краткое описание процедуры построения автомата $B(\pi, \varphi)$.

Замыканием Фишера-Ладнера формулы $\varphi = \mathbf{E}\psi(\mathbf{E}\xi_1, \ldots, \mathbf{E}\xi_n)$ называется минимальное по включению множество формул $\mathrm{FL}(\varphi)$, для которого выполняются следующие свойства:

- $\psi \in \mathrm{FL}(\varphi)$;
- если $\neg\zeta \in \mathrm{FL}(\varphi)$, то $\zeta \in \mathrm{FL}(\varphi)$;
- если $\zeta \in \mathrm{FL}(\varphi)$, то $\neg\zeta \in \mathrm{FL}(\varphi)$ (мы отождествляем $\neg\neg\zeta$ и ζ);
- если $\zeta \wedge \chi \in \mathrm{FL}(\varphi)$, то $\zeta, \chi \in \mathrm{FL}(\varphi)$;
- если $\mathbf{X}_c\zeta \in \mathrm{FL}(\varphi)$, то $\zeta \in \mathrm{FL}(\varphi)$;
- если $\zeta\,\mathbf{U}_L\chi \in \mathrm{FL}(\varphi)$, то $\zeta, \chi \in \mathrm{FL}(\varphi)$.

Мы добавляем к классическому определению замыкания ещё одно требование:

- если $\zeta\,\mathbf{U}_L\chi \in \mathrm{FL}(\varphi)$, то для всех $\tau = (q, c, q', h) \in T: \zeta\,\mathbf{U}_{L[c]}\chi \in \mathrm{FL}(\varphi)$.

Множество формул $K \subseteq \mathrm{FL}(\varphi)$ называется *атомом*, если:

- K непротиворечиво с точки зрения пропозициональной логики, т.е.
  - если $\xi \wedge \chi \in K$, то $\xi \in K$ и $\chi \in K$;
  - $\xi \in K$ тогда и только тогда, когда $\neg\xi \notin K$;
  - если $\mathrm{true} \in \mathrm{FL}(\varphi)$, то $\mathrm{true} \in K$;
- K подчиняется семантике оператора $\mathbf{U}$, то есть для всех $\zeta\,\mathbf{U}_L\chi \in \mathrm{FL}(\varphi)$
  - если $\chi \in K$ и $\varepsilon \in L$ то $\zeta\,\mathbf{U}_L\chi \in K$;
  - если $\zeta\,\mathbf{U}_L\chi \in K$, и $\chi \notin K$, и $\varepsilon \in L$, то $\zeta \in K$.

Атом K, содержащий формулу ψ, называется *начальным*. Далее под формулами будут подразумеваться элементы множества $\mathrm{FL}(\varphi)$.

Пусть $\{P_i\}_{i=1}^m$ — это все базовые предикаты, встречающиеся в формуле $\varphi = \mathbf{E}\psi(\mathbf{E}\varphi_1, \ldots, \mathbf{E}\varphi_n)$, и пусть им соответствуют автоматы $A_{P_i} = (Q_{P_i}, \mathcal{A}, p_i^{init}, \delta_{P_i}, F_{P_i})$. Каждый базовый предикат встречается в одной из формул ξ_i. По предположению индукции для каждой из этих формул существует проверочный автомат $A(\pi, \xi_i) = (Q_i, T, q_i^{init}, \delta_i, F_i),\ 1 \leqslant i \leqslant n$. Каждое состояние автомата $A(\pi, \xi_i)$ представляет собой кортеж вида $(q, p_{j_1}, \ldots, p_{j_r}), q \in Q, p_{j_k} \in Q_{P_{j_k}}$. Это состояние можно получить из кортежа

$(q, p_1, \dots, p_m), p_j \in Q_{P_j}$, опустив все те элементы p_j, которые соответствуют базовым предикатам P_j, не встречающимся в формуле в ξ_i. Будем обозначать полученное таким образом состояние записью $q_i(q, p_1, \dots, p_m)$.

Опишем теперь конструкцию автомата Бюхи $B(\pi, \varphi)$. Каждое состояние этого автомата содержит состояние преобразователя π, набор состояний автоматов A_{P_i} и атом $K \subseteq \mathrm{FL}(\varphi)$. Устройство автомата $B(\pi, \varphi) = (S, T, \varnothing, \Delta, \mathcal{F})$ таково:

- $S = \left\{ \langle q, p_1, \dots, p_m, K \rangle \mid q \in Q, p_j \in Q_{P_j}, \text{а } K - \text{атом} \right\}$;
- $\mathcal{F} = \{F_{\alpha \mathbf{U}_L \beta} \mid \alpha \mathbf{U}_L \beta \in \mathrm{FL}(\varphi)\}$, где $F_{\alpha \mathbf{U}_L \beta}$ — множество всех тех состояний, которые удовлетворяют следующему требованию: для всех сдвигов $L' = L[x]$, где $x \in C^*$, если $\alpha \mathbf{U}_L \beta \in K$, то $\beta \in K$ и $\varepsilon \in L'$.
- отношение $\Delta \subseteq S \times T \times S$ подчиняется следующему требованию: для всякого состояния $s = \langle q, p_1, \dots, p_n, K \rangle \in S$ и перехода $\tau = (q, c, q', h) \in T$:
  - если $\mathbf{E}\varphi_i \in K$ для некоторого $i, 1 \leqslant i \leqslant n$, причем $q_i(s) \notin F_i$, или если существует $\mathbf{X}_a \xi \in K$, для которого $a \neq c$, то *не существует ни одного перехода* из s по τ, т.е. $(s, \tau, s') \notin \Delta$ для любого состояния $s' \in S$;
  - если для всех формул $\mathbf{E}\varphi_i \in K$ выполняется $q_i(s) \in F_i$, и для всех $\mathbf{X}_a \xi \in K$ выполняется $a = c$, то $(s, \tau, s') \in \Delta$ для каждого такого состояния $s' = \langle q', p'_1, \dots, p'_m, K' \rangle \in S$, которое удовлетворяет набору условий:
    1. $p'_i = \delta_{P_i}(p_i, h), 1 \leqslant i \leqslant n$;
    2. для всех $\mathbf{X}_c \xi : \mathbf{X}_c \xi \in K$ если и только если $\xi \in K'$;
    3. для всех $\zeta \mathbf{U}_L \chi$: $\zeta \mathbf{U}_L \chi \in K$ тогда и только тогда, когда либо $\varepsilon \in L$ и $\chi \in K$, либо $(\zeta \mathbf{U}_{L[c]} \chi) \in K'$ и при условии $\varepsilon \in L$ верно, что $\zeta \in K$.

Приведенная здесь конструкция автомата Бюхи $B(\pi, \varphi)$ в общих чертах воспроизводит известное построение автомата Бюхи по формуле линейной темпоральной логики LTL, описанное [7], с той лишь разницей, что при работе с формулами языка $Reg\text{-}CTL^*$ используются дополнительные конечные автоматы для базовых предикатов, а также проверочные автоматы для подформул состояния. И поэтому нетрудно заметить, что, следуя тому же ходу рассуждений с применением индукции по длине формулы φ и длине траектории $tr \in Fin\, Tr_\pi$, который подробно описан в книге [7], можно доказать следующее вспомогательное утверждение:

Лемма 1. *Для любого конечного преобразователя π, всякой формулы состояния $\varphi = \mathbf{E}\psi(\mathbf{E}\varphi_1, \dots, \mathbf{E}\varphi_n)$ и произвольной конечной траектории $tr \in Fin\, Tr_\pi$ выполняется соотношение: $\Gamma_\pi, q_{init}[tr] \models \varphi$ тогда и только тогда, когда автомат Бюхи $B(\pi, \varphi)$ имеет допускающий прогон из состояния $\langle q_{init}[tr], p_1^{init}[tr], \dots, p_m^{init}[tr], K \rangle$, для которого $\psi \in K$.*

Опираясь на лемму 1, можно предложить следующий способ построения проверочного автомата $A(\pi, \varphi)$. Этот автомат должен иметь следующее устройство $A(\pi, \varphi) = (Q_A, T, q_A^{init}, \delta_A, F_A)$, где

- $Q_A = Q \times Q_{P_1} \times \dots \times Q_{P_m}$;
- $q_A^{init} = (q_{init}, p_1^{init}, \dots, p_m^{init})$;
- значение функции переходов $\delta_A : Q_A \times T \longrightarrow Q_A$ определяется следующим образом: для любого состояния $q_A = (q', p_1, \dots, p_m)$ и перехода $\tau = (q', c, q'', h)$

$$\delta_A(q_A, \tau) \quad = \quad (q'', \delta_{P_1}(p_1, \tau), \dots, \delta_{P_m}(p_m, \tau));$$

- множество $F_A \subseteq Q_A$ состоит из всех состояний $q_A = (q, p_1, \dots, p_m)$, для которых существуют начальный атом $K \subseteq \mathrm{FL}(\varphi)$ и допускающий прогон $B(\pi, \varphi)$ из состояния $\langle q, p_1, \dots, p_m, K \rangle$.

Согласно лемме 1 автомат $A(\pi, \varphi)$ удовлетворяет определению проверочного автомата:

$$\Gamma_\pi, v_{init}[tr] \models \varphi \iff tr \in A(\pi, \varphi).$$

Оценим размер полученного проверочного автомата. Как видно из описания устройства проверочного автомата, его размер можно оценить сверху

$$|A(\pi,\varphi)| = \mathrm{poly}(|\pi| \cdot \prod_{i=1}^{m} |P_i|) \lesssim |\pi| \cdot 2^{\mathrm{poly}(|\varphi|)}.$$

Для вычисления множества принимающих состояний автомата $A(\pi,\varphi)$ достаточно для каждого состояния q_A проверить, существует ли такой начальный атом $K \subseteq \mathrm{FL}(\varphi)$, что автомат $B(\pi,\varphi)$ имеет допускающий прогон из (q_A, K).

Поскольку $|K| \leqslant |\mathrm{FL}(\varphi)| \leqslant 2^{\mathrm{poly}(|\varphi|)}$, для хранения описания одного состояния автомата $B(\pi,\varphi)$ требуется объем памяти, не превышающий величины $2^{\mathrm{poly}(|\varphi|)}$. Таким образом, существование допускающего прогона может быть проверено *недетерминированно* переборным поиском с использованием объема памяти, ограниченного величиной $2^{\mathrm{poly}(|\varphi|)}$. По теореме Савича [21] детерминизация алгоритма поиска приводит не более чем к квадратичному увеличению объема используемой памяти. Поэтому сложность по объему памяти предложенного алгоритма проверки выполнимости $\pi \vDash \varphi$ ограничена сверху величиной $|\pi| 2^{\mathrm{poly}(|\varphi|)}$. □

Нижняя оценка сложности задачи верификации

Теорема 2. *Задача верификации конечных преобразователей относительно формул логики Reg-CTL* является* PSpace*-трудной.*

Доказательство. Покажем, как можно свести известную (см. [22]) PSpace-полную проблему проверки непустоты пересечения языков, распознаваемых семейством конечных автоматов, к указанной задаче верификации. Пусть задано семейство конечных автоматов $A_1, A_2, \ldots, A_n$ над алфавитом Σ. Рассмотрим преобразователь $\pi = (q, \Sigma, \Sigma, q, T)$ с одним состоянием управления и множеством переходов $T = \{(q,\sigma,\sigma,q) : \sigma \in \Sigma\}$. Будем использовать автоматы A_i для представления базовых предикатов (регулярных языков) в формуле $\varphi = \mathbf{EF}_{\Sigma^*} \bigwedge_{i=1}^{n} A_i$. Эта формула выражает следующее требование: из начальной вершины графа Γ_π достижима вершина, в которой истинны все базовые предикаты $A_1, \ldots, A_n$. Таким образом, имеет место равносильность

$$\bigcap_{i=1}^{n} A_i \neq \varnothing \quad \Longleftrightarrow \quad \pi \vDash \mathbf{EF}_{\Sigma^*} \bigwedge_{i=1}^{n} A_i,$$

показывающая, что для проверки непустоты пересечения языков, распознаваемых семейством автоматов $A_1, A_2, \ldots, A_n$, достаточно проверить соотношение $\pi \vDash \varphi$. □

4. Заключение

В этой статье мы определили синтаксис и семантику нового расширения *Reg-CTL** темпоральной логики *CTL**, которую можно использовать для спецификаций поведения последовательных реагирующих систем. Для новой темпоральной логики была исследована и решена задача верификации моделей конечных автоматов-преобразователей. Решение получено с помощью теоретико-автоматного метода, следуя схемам построения алгоритмов верификации автоматов-преобразователей, предложенных в работах [5, 8] для других расширений темпоральных логик.

Описанный здесь алгоритм верификации автоматов-преобразователей для логики *Reg-CTL** имеет значительно большую сложность по объему памяти, нежели аналогичный алгоритм для *CTL** (см. [7]). Однако нам удалось показать, что рассматриваемая нами задача верификации является PSpace-трудной. Таким образом, пока существует экспоненциальный разрыв между верхней и нижней оценками сложности этой задачи. Мы рассчитываем построить более эффективный (полиномиальный по памяти) алгоритм верификации, основанный на результатах статьи [16], где

исследовали свойства логики, семантически близкие рассматриваемой нами логике *Reg-LTL*. Эти результаты дают основание предполагать, что задача верификации для *Reg-CTL**, по-видимому, является PSpace-полной. Доказательство этого предположения является одной из целей дальнейших исследований в области верификации реагирующих систем.

Разумеется, интересно также проверить практическую применимость предложенного в статьях [4—6, 8] формализма для спецификации поведения последовательных реагирующих систем. По-видимому, для этой цели можно обратиться к задаче верификации программируемых логических контроллеров (ПЛК) [23, 24] и сравнить выразительные возможности предложенного нами расширения темпоральных логик с другими формальными средствами спецификации и верификации ПЛК [25—28].

Авторы выражают признательность рецензенту за ценные замечания, позволившие улучшить облик этой статьи.

References

[1] R. Alur and P. Cerny, «Streaming transducers for algorithmic verification of single-pass list-processing programs», in *Proceedings of the 38th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 2011, pp. 599–610.

[2] Q. Hu and L. D'Antoni, «Automatic program inversion using symbolic transducers», in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2017, pp. 376–389.

[3] M. Veanes, P. Hooimeijer, B. Livshits, D. Molnar, and N. Bjorner, «Symbolic finite state transducers: Algorithms and applications», in *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 2012, pp. 137–150.

[4] A. R. Gnatenko and V. A. Zakharov, «On the Expressive Power of Some Extensions of Linear Temporal Logic», *Automatic Control and Computer Sciences*, vol. 53, no. 7, pp. 663–675, 2019.

[5] D. G. Zakharov, V. A. Kozlova, and D. Kozlova, «On the model checking of sequential reactive systems», in *Proceedings of the 25-th International Workshop on Concurrency, Specification and Programming, Rostock, Germany, September 28-30*, vol. 1698, 2016, pp. 233–244.

[6] A. R. Gnatenko and V. A. Zakharov, «On the model checking of finite state transducers over semigroups», *Proceedings of ISP RAS*, vol. 30, no. 3, pp. 303–324, 2018.

[7] E. M. Clarke Jr, O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Model checking*. MIT press, 1999.

[8] A. R. Gnatenko, «On the complexity of model checking problem for finite state transducers over free semigroups», in *Proceedings of the Student Session of European Summer School on Logic, Language and Information, Riga*, 2019.

[9] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[10] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi, «On the temporal analysis of fairness», in *Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 1980, pp. 163–173.

[11] Z. Manna and P. Wolper, «Synthesis of communicating processes from temporal logic specifications», in *Workshop on Logic of Programs*, Springer, vol. 131, 1981, pp. 253–281.

[12] P. Wolper, «Temporal Logic Can Be More Expressive», *Information and control*, vol. 56, no. 1-2, pp. 72–99, 1983.

[13] O. Kupferman, N. Piterman, and M. Y. Vardi, «Extended temporal logic revisited», in *Proceedings of 12-th International Conference on Concurrency Theory*, Springer, 2001, pp. 519–535.

[14] M. Y. Vardi and P. Wolper, «Yet another process logic», in *Lecture Notes in Computer Science*, Springer, vol. 14, 1983, pp. 501–512.

[15] M. Y. Vardi, «A Temporal Fixpoint Calculus», in *Proceedings of the 15-th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 1988, pp. 250–259.

[16] J. G. Henriksen and P. S. Thiagarajan, «Dynamic linear time temporal logic», *Annals of Pure and Applied logic*, vol. 96, no. 1-3, pp. 187–207, 1999.

[17] M. Leucker and C. Sanchez, «Regular Linear Temporal Logic», in *Proceedings of the 4-th International colloquium on theoretical aspects of computing*, Springer, 2007, pp. 291–305.

[18] R. Mateescu, P. T. Monteiro, E. Dumas, and H. De Jong, «CTRL: Extension of CTL with regular expressions and fairness operators to verify genetic regulatory networks», *Theoretical Computer Science*, vol. 412, no. 26, pp. 2854–2883, 2011.

[19] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper, «Simple on-the-fly automatic verification of linear temporal logic», in *International Conference on Protocol Specification, Testing and Verification*, Springer, 1995, pp. 3–18.

[20] M. Y. Vardi and P. Wolper, «An automata-theoretic approach to automatic program verification», in *Proceedings of the First Symposium on Logic in Computer Science*, IEEE Computer Society, 1986, pp. 322–331.

[21] W. J. Savitch, «Relationships between nondeterministic and deterministic tape complexities», *Journal of computer and system sciences*, vol. 4, no. 2, pp. 177–192, 1970.

[22] D. Kozen, «Lower bounds for natural proof systems», in *Proceedings of the 18-th Symp. on the Foundations of Computer Science*, IEEE, 1977, pp. 254–266.

[23] A. Mader, «A classification of PLC models and applications», in *Discrete Event Systems*, vol. 569, Springer, 2000, pp. 239–246.

[24] T. Ovatman, A. Aral, D. Polat, and A. O. Unver, «An overview of model checking practices on verification of PLC software», *Software & Systems Modeling*, vol. 15, no. 4, pp. 937–960, 2016.

[25] N. Garanina, I. Anureev, V. Zyubin, A. Rozov, T. Liakh, and S. Gorlatch, «Reasoning about Programmable Logic Controllers», *System Informatics*, vol. 17, pp. 33–42, 2020.

[26] E. V. Kuzmin, D. A. Ryabukhin, and V. A. Sokolov, «On the expressiveness of the approach to constructing PLC-programs by LTL-specification», *Automatic Control and Computer Sciences*, vol. 50, no. 7, pp. 510–519, 2016.

[27] O. Ljungkrantz, K. Akesson, M. Fabian, and C. Yuan, «A formal specification language for PLC-based control logic», in *Proceedings of the 8th IEEE International Conference on Industrial Informatics*, IEEE, 2010, pp. 1067–1072.

[28] O. Ljungkrantz, K. Akesson, M. Fabian, and A. H. Ebrahimi, «An empirical study of control logic specifications for programmable logic controllers», *Empirical Software Engineering*, vol. 19, no. 3, pp. 655–677, 2014.