MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 28, NO. 4, 2021
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

On the Satisfiability and Model Checking for one Parameterized Extension of Linear-time Temporal Logic

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

DOI: 10.18255/1818-1015-2021-4-356-371

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

MSC2020: 68W10
Research article
Full text in Russian

Received November 15, 2021
After revision December 1, 2021
Accepted December 8, 2021

Sequential reactive systems are computer programs or hardware devices which process the flows of input data or control signals and output the streams of instructions or responses. When designing such systems one needs formal specification languages capable of expressing the relationships between the input and output flows. Previously, we introduced a family of such specification languages based on temporal logics *LTL*, *CTL* and CTL^* combined with regular languages. A characteristic feature of these new extensions of conventional temporal logics is that temporal operators and basic predicates are parameterized by regular languages. In our early papers, we estimated the expressive power of the new temporal logic *Reg-LTL* and introduced a model checking algorithm for *Reg-LTL*, *Reg-CTL*, and $Reg\text{-}CTL^*$. The main issue which still remains unclear is the complexity of decision problems for these logics. In the paper, we give a complete solution to satisfiability checking and model checking problems for *Reg-LTL* and prove that both problems are PSPACE-complete. The computational hardness of the problems under consideration is easily proved by reducing to them the intersection emptyness problem for the families of regular languages. The main result of the paper is an algorithm for reducing the satisfiability of checking *Reg-LTL* formulas to the emptiness problem for Buchi automata of relatively small size and a description of a technique that allows one to check the emptiness of the obtained automata within space polynomial of the size of input formulas.

Keywords: temporal logics; regular language; transducer; model checking; satisfiability checking; Buchi automata; emptiness problem

INFORMATION ABOUT THE AUTHORS

Anton Romanovich Gnatenko
correspondence author
orcid.org/0000-0003-1499-2090. E-mail: gnatenko.cmc@gmail.com
Master student.

Vladimir Anatolyevich Zakharov
correspondence author
orcid.org/0000-0002-3794-9565. E-mail: zakh@cs.msu.su
Prof. Dr.Sc.

For citation: A. R. Gnatenko and V. A. Zakharov, "On the Satisfiability and Model Checking for one Parameterized Extension of Linear-time Temporal Logic", *Modeling and analysis of information systems*, vol. 28, no. 4, pp. 356-371, 2021.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 28, № 4, 2021
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

О верификации моделей и проверке выполнимости формул одного параметрического расширения темпоральной логики линейного времени

А. Р. Гнатенко[1], В. А. Захаров[1,2]

DOI: 10.18255/1818-1015-2021-4-356-371

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

УДК 517.9
Научная статья
Полный текст на русском языке

Получена 15 ноября 2021 г.
После доработки 1 декабря 2021 г.
Принята к публикации 8 декабря 2021 г.

К последовательным реагирующим системам относятся компьютерные программы и вычислительные устройства, которые обрабатывают потоки входных данных или сигналов управления и генерируют на выходе последовательности команд или результатов вычислений. Для проектирования таких систем полезно иметь формальные языки спецификаций, способные выражать отношения между входными и выходными потоками данных. В предшествующих работах нами было предложено семейство таких языков спецификаций, представляющих собой расширение темпоральных логик LTL, CTL и CTL^* за счет использования регулярных языков в качестве параметров темпоральных операторов. Мы провели сравнительный анализ выразительных возможностей нового расширения темпоральной логики линейного времени $Reg\text{-}LTL$ и предложили алгоритмы верификации моделей для новых расширений логик $Reg\text{-}LTL$, $Reg\text{-}CTL$, и $Reg\text{-}CTL^*$. Однако вопрос о сложности задач верификации моделей и проверки выполнимости формул указанных логик оставался открытым. В этой статье мы восполняем этот пробел в наших исследованиях и показываем, что для темпоральной логики $Reg\text{-}LTL$ обе задачи являются Pspace-полными. Вычислительная трудность рассматриваемых задач легко доказывается сведением к ним проблемы пустоты пересечения семейств регулярных языков. Основным результатом статьи является алгоритм сведения задачи проверки выполнимости формул логики $Reg\text{-}LTL$ к проблеме пустоты автоматов Бюхи сравнительно небольшого размера и описание стратегии, позволяющей проверять пустоту полученных автоматов с использованием объема памяти, полиномального относительно размера исходных формул.

Ключевые слова: темпоральные логики; регулярный язык; автомат-преобразователь; верификация моделей; проверка выполнимости; автоматы Бюхи; проблема пустоты

ИНФОРМАЦИЯ ОБ АВТОРАХ

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

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

Для цитирования: A. R. Gnatenko and V. A. Zakharov, "On the Satisfiability and Model Checking for one Parameterized Extension of Linear-time Temporal Logic", *Modeling and analysis of information systems*, vol. 28, no. 4, pp. 356-371, 2021.

Введение

Амир Пнуели был, вероятно, первым, кто обратил внимание на возможность использования темпоральных логик для описания поведения последовательных и параллельных программ [1]. Темпоральные формулы хорошо подходят для этой цели, когда поведение вычислительной системы рассматривается как совокупность последовательностей событий. За полвека, прошедшие со времени публикации пионерской работы Пнуели, появилось и было подробно изучено большое разнообразие темпоральных логик, таких как темпоральная логика линейного времени (LTL) [2], логика деревьев вычислений (CTL) и ее обобщение (CTL^*) [3], метрическая темпоральная логика (MTL) [4] и пр. Логика LTL является довольно выразительным формальным языком для рассуждения об устройстве бесконечных последовательностей событий, сигналов, состояний вычисления. Формулы LTL строятся из символов конечного алфавита атомарных высказываний при помощи булевых связок и темпоральных модальностей, или операторов, таких как **F** (оператор происшествия, когда-нибудь), **G** (оператор неограниченной инвариантности, всегда), **U** (оператор ограниченной инвариантности, до тех пор, пока) и некоторых других. Например, для заданного алфавита элементарных событий $\{a, b\}$ формула $\mathbf{F}a$ выполняется для тех и только тех последовательностей событий, которые включают в себя хотя бы одно событие a (т.е., событие a когда-нибудь происходит), а формула $\mathbf{G}b$ требует, чтобы событие b происходило в каждый момент времени (событие b происходит всегда). При помощи формул LTL можно описывать разнообразные свойства вычислений, включая требования живости, безопасности, справедливости [5].

Основываясь на концепциях логики LTL, многие авторы предложили и исследовали ряд ее фрагментов и расширений, к числу которых относятся расширенная LTL [6], квантифицированная LTL [7], динамическая LTL [8, 9], темпоральные дескриптивные логики [10], пространственно-временные логики [11] и целый ряд других. Логики этого семейства представляют двоякий интерес. С одной стороны, они интересны с чисто теоретической точки зрения как некие разумные математические теории, имеющие определенные области применения. В связи с этим возникает необходимость изучить их выразительные возможности, аксиоматизацию, алгоритмические проблемы. С другой стороны, эти логики могут использоваться в качестве языков спецификации для различных типов программного и аппаратного обеспечения, когда приходится иметь дело с задачами верификации, статического анализа, проверкой качества функционирования систем (см. [12, 13]).

Существование большого разнообразия темпоральных логик, которым, казалось бы, недостает строгой систематизации, легко объяснимо. Для верификации разных типов программ, систем обработки информации и вычислительной и коммуникационной аппаратуры требуются модели совершенно разной природы и, следовательно, разные языки спецификации. Рассмотрим, например, последовательные реагирующие системы, представляющие собой компьютерные программы или аппаратные устройства, которые обрабатывают потоки входных данных или управляющих сигналов и выдают на выходе последовательности команд или результатов действий. Эти действия и порядок их выполнения зависят не только от текущего принятого сигнала, но и от всех предыдущих управляющих сигналов. Таким образом, поведение реагирующей системы характеризуется некоторым соответствием (бинарным отношением) между потоком сигналов и потоком действий. Поэтому при разработке формальных методов верификации реагирующих систем необходим, прежде всего, адекватный формальный язык, позволяющий описывать не последовательности событий, а отношения между последовательностями входных и выходных событий.

Хотя темпоральная логика широко используется уже долгое время для спецификации и верификации программ, не было предпринято никаких серьезных попыток разработать вариант темпорального языка, который подходил бы для работы с реагирующими системами такого рода. Ключевым моментом является то, что темпоральные формулы обычно интерпретируются на последовательностях событий, в то время как поведение реагирующей системы проявляется в парах

последовательностей — входных и выходных — с определенными отношениями между ними. Ни один из традиционных языков темпоральной спецификации вычислений не содержит специальных средств для выражения свойств таких бинарных отношений.

Чтобы преодолеть этот недостаток, мы предложили дополнить временные операторы параметрами, в качестве которых выступают регулярные языки, и получили таким образом *Reg*-варианты темпоральных логик *LTL*, *CTL*, и CTL^*, описанные в работах [14—16]). Например, формулу $\mathbf{G}\varphi$ ("свойство φ всегда выполняется") можно сделать более специфичной, записав ее как $\mathbf{G}_L\varphi$, что означает "свойство φ выходной последовательности выполняется всякий раз, когда входная последовательность представляет собой слово из языка L". Конечно, можно использовать любой тип языков для параметризации временных модальностей, но для того, чтобы традиционные задачи, связанные с применением логик, оставались алгоритмически разрешимыми мы ограничиваемся регулярными языками. В статье [17] было показано, что вопрос о выполнимости даже очень простых формул может оказаться алгоритмически неразрешимым, если позволить использовать в качестве параметров темпоральных операторов контекстно-свободные языки.

Следуя основным положениям теоретико-автоматного подхода к решению задач верификации моделей относительно темпоральных спецификаций (см. [13, 18]), авторы статьи [14] предложили метод трансляции формул логики *Reg-LTL* в автоматы Бюхи; таким образом, задачи верификации моделей и проверки выполнимости формул *Reg-LTL* были сведены к проблеме пустоты для автоматов Бюхи. В статье [16] этот метод трансляции был обобщен и адаптирован для работы с логикой $Reg\text{-}CTL^*$. Однако в обоих случаях размер получаемых в результате трансляции автоматов оценивался двойной экспонентой, зависящей от размера темпоральной формулы; это давало верхнюю оценку ExpSpace вычислительной сложности рассматриваемых задач. С другой стороны, в статье [16] было показано, что задача верификации моделей для логики $Reg\text{-}CTL^*$ является PSpace-трудной. Таким образом, между верхней и нижней оценками вычислительной сложности задач проверки выполнимости формул рассматриваемых параметризованных модификаций темпоральных логик образовался разрыв, и главная цель данной статьи — восполнить этот пробел в наших исследованиях указанных логик.

Мысль о том, что выразительные возможности темпоральной логики можно усилить за счет повышения «чувствительности» темпоральных операторов, не нова. Ограниченность *LTL* стала очевидной, как только было показано, что формулы этой логики позволяют описывать лишь такие регулярные языки, которые можно задавать без привлечения операции итерации Клини [19]. В статье [6] Вольпер привел пример очень простого свойства, которое невозможно охарактеризовать формулами логики *LTL* (в статье [17] приведено другое доказательство этого факта при помощи темпоральной разновидности игр Эренфойхта–Фреше [20]). Чтобы сделать язык логики *LTL* более выразительным, Вольпер предложил ввести новый вид темпоральных операторов, область действия которых определяется регулярными грамматиками [6]. Эта идея получила дальнейшее развитие, в котором для усиления темпоральных операторов вместо грамматик были задействованы конечные автоматы [21] и даже альтернирующие автоматы [22].

В статье [8] Хенриксен и Тиагаранжан предложили альтернативный подход к расширению логики *LTL*. Они использовали регулярные выражения для параметризации темпоральных операторов и получили динамическую темпоральную логику линейного времени (*DLTL*), которая очень похожа на логику *Reg-LTL*. Авторы статьи [8] разработали разрешающую процедуру для этой логики, позволяющую за экспоненциальное время транслировать логические формулы в автоматы Бюхи, размер которых так же экспоненциально зависит от размера формул. Хотя для интерпретации формул *DLTL* используются одиночные последовательности событий, мы воспользовались предложенным в статье [8] методом трансляции, адаптировали его к нашей логике *Reg-LTL* и получили полиномиальные по объему используемой памяти алгоритмы верификации моделей и проверки

выполнимости формул логики *Reg-LTL*. Эти алгоритмы и являются основным результатом нашей статьи. Принимая во внимание нижние оценки сложности указанных задач, полученные ранее в статьях [15—17], мы приходим к выводу о том, что задачи верификации моделей и проверки выполнимости формул темпоральных логик *Reg-CTL*, *Reg-LTL*, и *Reg-CTL** являются Pspace-полными.

Оглавление нашей статьи таково. В разделе 1 описаны автоматы-преобразователи, которые служат формальной моделью последовательных реагирующих систем. Синтаксис и семантика темпоральной логики *Reg-LTL*, используемой для спецификации поведения автоматов-преобразователей, представлены в разделе 2. В разделе 3 рассказывается о новом методе трансляции формул *Reg-LTL* в автоматы Бюхи. Построенному на основе этой трансляции алгоритму проверки выполнимости формул *Reg-LTL* посвящен раздел 4. В нем показано, что для проверки выполнимости формул достаточно использовать объем памяти, полиномиальный относительно размера формул. Далее, в разделе 5 говорится о том, каким образом описанный метод трансляции формул *Reg-LTL* в автоматы Бюхи применять для верификации моделей последовательных реагирующих систем относительно формальных спецификаций их поведения. В заключительном разделе 6 мы подводим итоги проведенных исследований задач спецификации и верификации моделей последовательных реагирующих систем с использованием темпоральных логик.

1. Модель последовательных реагирующих систем

Область применения моделей вычислений с конечным числом состояний (конечных автоматов) обширна. Простота устройства автоматов позволяет использовать их для автоматизации решения задач синтеза и анализа разнообразных управляющих систем. В частности, автоматы-преобразователи (машины с конечным числом состояний) хорошо подходят для моделирования последовательных реагирующих систем, включая интерпретаторы, системные драйверы, контроллеры сетевые коммутаторы. В этой главе мы приводим определения основных понятий, относящихся к этой модели вычислений.

Рассмотрим два конечных алфавита C и $\mathcal{A}$. Элементы множества C мы будем называть *сигналами*, а конечные последовательности этих элементов — *потоками сигналов*. Сигналы можно истолковывать как входные данные, управляющие команды или запросы, которые поступают на вход реагирующей системы из окружающей среды. Множество всех потоков сигналов будем обозначать записью C^*, а конкатенацию потоков $u, v \in C$ записывать как uv. Для обозначения пустого потока сигналов используем символ ε.

Символы алфавита $\mathcal{A}$ будем называть *элементарными действиями*, а конечные слова в алфавите $\mathcal{A}$ — *составными действиями* (далее, просто действиями). Элементарные действия являются абстракциями простейших операций, которые способна совершать реагирующая система, а составные действия являются последовательной композицией простейших операций. В общем случае разные действия могут давать одинаковый эффект, и поэтому их можно интерпретировать на полугруппе $(S, e, \circ)$, порожденной множеством элементарных действий $\mathcal{A}$ с операцией композиции $\circ$ и нейтральным элементом e. Элементы множества S будем называть состояниями данных. В результате применения составного действия $h = a_1 a_2 \ldots a_k$ к состоянию данных s будет получено новое состояние данных $s' = s \circ a_1 \circ a_2 \circ \cdots \circ a_k$.

В этой статье мы ограничимся рассмотрением самой простой интерпретации, когда $(S, e, \circ)$ является свободной полугруппой с множеством образующих $\mathcal{A}$. В этом случае множество состояний данных S совпадает с множеством составных действий $\mathcal{A}^*$, операцией $\circ$ является конкатенация слов в алфавите $\mathcal{A}$, а начальным состоянием данных e является пустое слово в алфавите $\mathcal{A}$. Более общий случай моделей реагирующих систем над произвольными полугруппами рассматривался в работах [23—25].

Автомат-преобразователь над множествами сигналов C и элементарных действий $\mathcal{A}$ задается пятеркой $\pi = (Q, C, \mathcal{A}, q_{init}, T)$, где

- Q — конечное множество *состояний управления*;
- $q_{init} \in Q$ — *начальное состояние*;
- $T \subseteq Q \times C \times Q \times \mathcal{A}^*$ — конечное *отношение переходов*.

Четверку $(q', c, q'', h) \in T$ будем называть *переходом*. Каждый такой переход соответствует простейшему шагу вычисления реагирующей системы: если система, пребывающая в состоянии q', получает на входе сигнал c, то она выполняет действие h и переходит в состояние q''. Как обычно, мы будем использовать запись $q' \xrightarrow{c,\ h} q''$ вместо $(q', c, q'', h) \in T$.

Прогон преобразователя π — это бесконечная последовательность переходов

$$\rho = q_1 \xrightarrow{c_1, h_1} q_2 \xrightarrow{c_2, h_2} q_3 \xrightarrow{c_3, h_3} \dots$$

Прогон называется *начальным*, если $q_1 = q_{init}$.

Прогоны автоматов-преобразователей — это и есть абстрактные образы вычислений реагирующих систем. Однако наблюдаемое поведение реагирующей системы выглядит совсем иначе. Стороннему наблюдателю не доступно устройство автомата-преобразователя, и поэтому он может обозревать лишь поступающие на его вход сигналы и регистрировать результаты выполнения действий после каждого входного сигнала. Поэтому при анализе поведения реагирующих систем вместо прогонов мы будем иметь дело с наблюдаемыми трассами, последовательностями пар, состоящими из поступивших на вход сигналов и достигнутых состояний данных.

Трассой над множествами сигналов C и элементарных действий $\mathcal{A}$ называется всякая пара $tr = \langle s_0, \alpha \rangle$, в которой первая компонента $s_0 \in \mathcal{A}^*$, а вторая компонента α является бесконечной последовательностью пар $\alpha = \{(c_i, s_i)\}_{i \geqslant 1}$, $c_i \in C$, $s_i \in \mathcal{A}^*$. Множество всех возможных трасс обозначим записью $Traces$. Мы будем обозначать записью $tr|^i$ суффикс трассы tr, т.е. трассу $tr|^i = (s_i, \alpha|^i)$, в которой $\alpha|^i = (c_{i+1}, s_{i+1}), (c_{i+2}, s_{i+2}), \dots$.

Каждое состояние данных $s_0 \in \mathcal{A}^*$ и прогон $\rho = q_1 \xrightarrow{c_1, h_1} q_2 \xrightarrow{c_2, h_2} \dots$ автомата-преобразователя π порождают трассу $tr = \langle s_0, \alpha \rangle$, в которой последовательность пар $\alpha = (c_1, s_1), (c_2, s_2), \dots, (c_i, s_i), \dots$ удовлетворяет равенству $s_i = s_{i-1} h_i$ для любого $i \geq 1$. Множество $Traces(\pi)$ всех трасс, порожденных начальным состоянием данных e и всеми возможными начальными прогонами ρ_{init} автомата-преобразователя π, формализует понятие наблюдаемого поведения последовательной реагирующей системы, моделируемой автоматом π. Тогда свойством наблюдаемого поведения системы является всякое подмножество $Prop$ множества трасс $Traces$. Реагирующая система, моделируемая автоматом-преобразователем π, удовлетворяет свойству наблюдаемого поведения $Prop$ в том и только том случае, если выполняется отношение $Traces(\pi) \subseteq Prop$.

2. Язык спецификаций поведения реагирующих систем *Reg-LTL*

В основу формального языка спецификаций *Reg-LTL*, предназначенного для описания свойств наблюдаемого поведения реагирующих систем, положен язык темпоральной логики линейного времени *LTL*. Новый логический формализм имеет две отличительные особенности:

- атомарными формулами являются предикаты на множестве состояний данных $\mathcal{A}^*$, т.е. языки в алфавите $\mathcal{A}$;
- все темпоральные операторы параметризованы отдельными сигналами из множества C или множествами (языками) потоков сигналов из совокупности C^*.

В общем случае (см. [14]) базовыми предикатами и параметрами темпоральных операторов могут быть произвольные языки в алфавитах сигналов C и элементарных действий $\mathcal{A}$. Но для построения эффективных разрешающих процедур приходится ограничиваться регулярными языками. Как известно, регулярные языки можно задавать разными способами; один из наиболее простых и эффективных — задавать их при помощи конечных детерминированных автоматов-распознавателей.

В дальнейшем, упоминая о регулярных языках при описании синтаксиса логики *Reg-LTL*, мы будем подразумевать, что каждый регулярный язык представлен минимальным детерминированным автоматом-распознавателем (автоматом Рабина-Скотта), допускающим этот язык.

Формулы логики *Reg-LTL* определяются следующим образом:

1. всякий регулярный язык P в алфавите $\mathcal{A}$ является формулой (базовым предикатом);
2. если выражения φ_1, φ_2 — формулы, то $\neg\varphi_1$ and $\varphi_1 \wedge \varphi_2$ также являются формулами;
3. если выражения φ_1, φ_2 — формулы, $c \in \mathcal{C}$, и L — регулярный язык в алфавите $\mathcal{C}$, то выражения $\mathbf{X}_c\,\varphi_1$ и $\varphi_1 \mathbf{U}_L \varphi_2$ являются формулами.

Семантика языка *Reg-LTL* определяется отношением выполнимости формул на трассах, которые выступают в роли интерпретационных структур. Для заданной трассы $tr = \langle s_0, \alpha \rangle$, где $\alpha = \{(c_i, h_i)\}_{i \geqslant 1}$ мы будем использовать запись $tr \models \psi$ для обозначения того, что формула ψ выполняется на трассе tr. Отношение выполнимости формул $\models$ определяется индуктивно следующим образом:

1. $tr \models P \iff s_0 \in P$ для любого базового предиката P;
2. $tr \models \neg\varphi \iff$ если неверно, что $s \models \varphi$;
3. $tr \models \psi_1 \wedge \psi_2 \iff tr \models \psi_1$ и $tr \models \psi_2$;
4. $tr \models \mathbf{X}_c\varphi \iff c = c_1$ и $tr|^1 \models \varphi$;
5. $tr \models \varphi_1 \mathbf{U}_L \varphi_2 \iff$ существует такое $i \geqslant 0$, что поток сигналов $c_1 c_2 \ldots c_i$ принадлежит языку L и $tr|^i \models \varphi_2$, и при этом для любого j, $0 \leqslant j < i$, всякий раз, когда поток сигналов $c_1 c_2 \ldots c_j$ принадлежит языку L, справедливо соотношение $tr|^j \models \varphi_1$.

Как видно из приведенного определения, темпоральные операторы $\mathbf{X}$ и $\mathbf{U}$ имеют в логике *Reg-LTL* такой же смысл, как и в основополагающей логике *LTL* с той лишь разницей, что теперь при продвижении времени приходится учитывать, принадлежит ли поток сигналов трассы tr, на которой проверяются формулы, тем языкам (в статьях [14, 17] они называются шаблонами поведения окружающей среды), которыми параметризованы эти операторы.

При помощи отрицания и конъюнкции можно выразить формулами другие булевы связки, наподобие дизъюнкции $\vee$ или импликации $\longrightarrow$. Кроме того, точно так же, как и в логике *LTL* можно определить другие темпоральные операторы, такие как $\mathbf{F}_L\varphi \equiv true\,\mathbf{U}_L\varphi$ и $\mathbf{G}_L\varphi \equiv \neg\mathbf{F}_L\neg\varphi$.

Каждая формула φ логики *Reg-LTL* характеризует множество трасс $Traces(\varphi) = \{tr \in Traces \mid tr \models \varphi\}$. Будем говорить, что формула φ *выполнима*, если $Traces(\varphi) \neq \varnothing$; также будем говорить, что автомат-преобразователь π *удовлетворяет формуле* φ (обозначается записью $\pi \models \varphi$) если имеет место включение $Traces(\pi) \subseteq Traces(\varphi)$.

Выразительные возможности темпоральной логики *Reg-LTL* подробно исследованы в статье [17]. В частности, были выделены два фрагмента $\mathcal{LP}$-1-*LTL* и $\mathcal{LP}$-*n*-*LTL*, которые позволили сравнить параметризованную темпоральную логику *Reg-LTL* с темпоральной логикой *LTL* и монодической логикой *S*1*S*. Формулы первого из этих фрагментов $\mathcal{LP}$-1-*LTL* строятся над однобуквенным алфавитом $\mathcal{C} = \{c\}$ входных сигналов и произвольным конечным алфавитом элементарных действий $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$. В качестве параметров темпоральных операторов в формулах этого фрагмента разрешается использовать любые регулярные языки над алфавитом $\{c\}$, а в качестве базовых предикатов используются регулярные языки вида $P_i = \mathcal{A}^* a_i$. При таких ограничениях появляется возможность сравнивать семантики логик *LTL* и $\mathcal{LP}$-1-*LTL* на одном и том же классе моделей (трасс). В [17] было показано, что любая формула *LTL* имеет эквивалентную формулу $\mathcal{LP}$-1-*LTL*, но при этом есть такие формулы $\mathcal{LP}$-1-*LTL*, для которых не существует эквивалентных формул *LTL*. Фрагмент $\mathcal{LP}$-*n*-*LTL* определяется аналогично, с той лишь разницей, что алфавиты $\mathcal{C}$ и $\mathcal{A}$ совпадают, т. е. $\mathcal{C} = \mathcal{A} = \{a_1, a_2, \ldots, a_n\}$. В той же статье [17] удалось установить, что класс моделей, на которых выполняются формулы $\mathcal{LP}$-*n*-*LTL*, в точности совпадает с классом сверхязыков, распознаваемых конечными автоматами Маллера. Отсюда следует, что логики $\mathcal{LP}$-*n*-*LTL* и *S*1*S* имеют одинаковые выразительные возможности.

В следующем разделе мы приступим к описанию нашего метода решения двух основных задач, связанных с описанной здесь логикой *Reg-LTL*. Задача проверки выполнимости формул состоит в том, чтобы для произвольной заданной формулы логики *Reg-LTL* выяснить, является ли эта формула выполнимой. Задача верификации моделей состоит в том, чтобы для произвольного заданного автомата-преобразователя π и произвольной заданной формулы φ логики *Reg-LTL* выяснить, справедливо ли отношение $\pi \models \phi$. Как видно из результатов, представленных в статье [17], обе рассматриваемые задачи являются Pspace-трудными, и поэтому нашей целью является разработка алгоритмов, способных решить указанные задачи с использованием полиномиального объема памяти.

3. Проверка выполнимости формул при помощи автоматов Бюхи

Теоретико-автоматный подход к проверке выполнимости формул темпоральной логики линейного времени, первоначально предложенный в статье [18] и опирающийся на идеи работы [26], предполагает построение для проверяемой формулы φ автомата Бюхи B_φ, который удовлетворяет следующему требованию: φ выполнима тогда и только тогда, когда $Lang(B_\varphi) \neq \emptyset$. Тогда проверка выполнимости формул сводится к задаче проверки пустоты для автоматов Бюхи. Эта задача решается детерминированным алгоритмом за время, линейное относительно размера автомата, или недетерминированным алгоритмом с использованием памяти, объем которой оценивается логарифмом от размера автомата. Обычно автомат Бюхи B_φ строится так, чтобы язык $Lang(B_\varphi)$ состоял из всех ω-слов, представляющих интерпретации (трассы), на которых выполняется проверяемая формула φ. Для темпоральной логики *LTL* трансляция формул в автоматы Бюхи может быть осуществлена за время, экспоненциальное относительно размера формулы и с использованием полиномиального объема памяти. Размер самого автомата B_φ при этом оценивается экспонентой от размера формулы φ, однако проверку пустоты языка $Lang(B_\varphi)$ можно проводить «на лету» без построения автомата в явном виде. Так удается показать, что задача проверки выполнимости формул *LTL* принадлежит классу сложности Pspace.

Авторы статьи [14] применили указанный подход к решению задачи верификации моделей для логики *Reg-LTL*. Однако предложенный в этой статье метод трансляции формул *Reg-LTL* в автоматы Бюхи оказался слишком трудоемким: размер получавшихся автоматов оценивался двойной экспонентой от размера темпоральных формул. Поэтому для проверки выполнимости формул требовался объем памяти, экспоненциально зависящий от размера формул. В данной работе мы модифицируем этот алгоритм, воспользовавшись приемами, которые были предложены в статье [8], и получаем полиномиальные по объему памяти процедуры верификации моделей и проверки выполнимости формул рассматриваемой логики *Reg-LTL*.

3.1. Автоматы Рабина–Скотта и автоматы Бюхи

Конечный автомат Рабина–Скотта (далее автомат-распознаватель) над алфавитом Σ задается пятеркой $A = (Q, \Sigma, q_{init}, \delta, F)$, где Q — конечное множество *состояний*, включающее *начальное состояние* $q_{init} \in Q$ и подмножество *допускающих состояний* $F \subseteq Q$, а $\delta : Q \times \Sigma \mapsto Q$ — *функция переходов*. Эту функцию можно распространить на множество слов Σ^* обычным образом, полагая $\delta(q, \varepsilon) = q$, и $\delta(q, \sigma x) = \delta(\delta(q, \sigma), x)$ для всех состояний $q \in Q$, $\sigma \in \Sigma$ и слов $x \in \Sigma^*$. Размер автомата A — это число его состояний $|A| = |Q|$.

Автомат-распознаватель A *допускает* слово $x \in \Sigma^*$ тогда и только тогда, когда $\delta(q_{init}, x) \in F$. Множество всех слов $Lang(A)$, допускаемых автоматом A, называется *языком* этого автомата. Для произвольного регулярного языка L обозначим записью $|L|$ размер минимального автомата, для которого выполнено равенство $L = Lang(A)$.

Конечные автоматы могут работать не только на конечных словах. Обозначим записью Σ^ω множество всех бесконечных последовательностей букв алфавита Σ, которые будем называть ω-словами.

Недетерминированный *автомат Бюхи* над алфавитом Σ, как и автомат Рабина–Скотта, задается пятеркой $B = (Q, \Sigma, q_{init}, \Delta, F)$, которая включает в себя множество состояний Q, начальное состояние q_{init}, отношение переходов $\Delta \subseteq Q \times \Sigma \times Q$ и подмножество допускающих состояний $F \subseteq Q$. Однако функционирование автоматов Бюхи определяется иначе. Для заданного ω-слова $x = \sigma_0\sigma_1\sigma_2\cdots \in \Sigma^\omega$, *прогоном* автомата Бюхи B на слове x называется такая бесконечная последовательность состояний $q_0, q_1, q_2, \ldots$, в которой $q_0 = q_{init}$, и для каждого $i, i \geqslant 0$, выполняется включение $(q_i, \sigma_i, q_{i+1}) \in \Delta$. Такой прогон считается *допускающим*, если в нем бесконечно много раз встречаются состояния из допускающего подмножества F. Будем говорить, что ω-слово $x \in \Sigma^\omega$ *допускается* автоматом B, если существует допускающий прогон автомата B на слове x. Множество всех ω-слов, которые допускает автомат Бюхи B обозначим записью $Lang(B)$. Размером $|B|$ автомата Бюхи B назовем число его состояний.

3.2. Трансляция формул *Reg-LTL* в автоматы Бюхи

Рассмотрим (бесконечное) множество пар слов $\mathcal{C} \times \mathcal{A}^*$, где $\mathcal{C}$ — множество сигналов, а $\mathcal{A}$ — множество элементарных действий. Тогда бесконечная последовательность $\beta = \{(c_i, h_i)\}_{i\geqslant 1}, c_i \in \mathcal{C},\ h_i \in \mathcal{A}^*$, может рассматриваться как ω-слово в указанном алфавите. Это слово однозначно характеризует трассу $tr_\beta = \langle e, \alpha\rangle$, где $\alpha = \{(c_i, h_1h_2 \ldots h_i)\}_{i\geqslant 1}$. Наша цель состоит в том, чтобы для произвольной формулы φ логики *Reg-LTL* построить такой автомат Бюхи B_φ, размер которого не более чем экспоненциально зависит от размера формулы B_φ, удовлетворяющий при этом следующему требованию: для любого ω-слова $\beta \in (\mathcal{C} \times \mathcal{A}^*)^\omega$ верно соотношение $\beta \in Lang(B_\varphi) \iff tr_\beta \models \varphi$.

Для заданной формулы φ логики *Reg-LTL* выделим в этой формуле все регулярные языки $L_1, L_2, \ldots, L_k$ и $P_1, P_2, \ldots, P_m$, которые используются в формуле φ в качестве параметров темпоральных операторов и базовых предикатов соответственно. Рассмотрим минимальные по размеру автоматы Рабина-Скотта $A_i = (Q'_i, \mathcal{C}, q'_{init,i}, \delta'_i, F'_i), 1 \leq i \leq k$, и $D_j = (Q''_j, \mathcal{A}, q''_{init,j}, \delta''_j, \mathcal{F}''_j), 1 \leq j \leq m$, распознающие эти регулярные языки. Размером формулы φ будем называть величину $|\varphi| = \ell + \sum\limits_{i=1}^{k} |A_i| + \sum\limits_{j=1}^{m} |D_j|$, где ℓ — количество булевых связок в формуле φ.

При построении автомата Бюхи B_φ будем следовать принципам построения автоматов, характеризующих выполнимость темпоральных формул, которые изложены в монографиях [12, 13] применительно к формулам логики *LTL*.

Замыканием Фишера–Ладнера для формулы φ назовём наименьшее множество формул $Closure(\varphi)$, которое удовлетворяет следующим условиям:

- $\varphi \in Closure(\varphi)$,
- $\psi \in Closure(\varphi) \iff \neg\psi \in Closure(\varphi)$,
- если $\psi' \wedge \psi'' \in Closure(\varphi)$, то $\psi', \psi'' \in Closure(\varphi)$,
- если $\mathbf{X}_c\psi \in Closure(\varphi)$, то $\psi \in Closure(\varphi)$,
- если $\psi'\mathbf{U}_L\psi'' \in Closure(\varphi)$, то $\psi', \psi'' \in Closure(\varphi)$,

Подмножество $K \subseteq Closure(\varphi)$ называется *согласованным*, если K непротиворечиво как множество пропозициональных формул и при этом согласовано с семантикой темпоральных операторов, т. е. удовлетворяет условиям:

- для всякой формулы $\psi \in Closure(\varphi)$ верно $\psi \in K \iff \neg\psi \notin K$ (полагая при этом $\neg\neg\psi = \psi$),
- для всякой формулы $\psi' \wedge \psi'' \in Closure(\varphi)$ верно $\psi' \wedge \psi'' \in K \iff \psi', \psi'' \in K$,
- для всякой формулы $\psi'\mathbf{U}_L\psi'' \in Closure(\varphi)$ если $\psi'' \in K$ и $\varepsilon \in L$, то $\psi'\mathbf{U}_L\psi'' \in K$,

Совокупность всех согласованных подмножеств совокупности формул $Closure(\varphi)$ будем обозначать записью K_φ.

Положим $V = \bigcup_{i=1}^{k} Q'_i$, $W = \bigcup_{j=1}^{m} Q''_j$. Будем говорить, что множество состояний $R \subseteq W$ *достижимо*, если R содержит ровно по одному состоянию каждого из автоматов-распознавателей $D_j, 1 \leq j \leq m$, причем все эти состояния достижимы из соответствующих начальных состояний $q_{init,j}$

этих автоматов по одному и тому же слову, т.е. $R = \{\delta''_j(q''_{init,j}, h) : 1 \le j \le m\}$ для некоторого слова $h \in A^*$.

Автомат Бюхи $B_\varphi = (Q, C \times \mathcal{A}^*, Q_{init}, \Delta, F)$ над алфавитом $C \times \mathcal{A}^*$ определяется следующим образом.

(1) Состояниями этого автомата являются наборы из множества

$$Q \subseteq K_\varphi \times 2^V \times 2^{V \times \{0,1\}} \times 2^W \times \{0,1\} \times \{\circ, \bullet\}.$$

Каждый такой набор (K, C, S, R, b, θ) состоит из согласованного множества формул $K \in K_\varphi$, контрольного множества состояний $C \subseteq V$, выполняющего множества помеченных состояний $S \subseteq V \times \{0,1\}$, предикатного достижимого множества состояний $R \subseteq W$, фазы $b \in \{0,1\}$ и маркера $\theta \in \{\circ, \bullet\}$. Для того, чтобы набор (K, C, S, R, b, θ) принадлежал множеству состояний рассматриваемого автомата Бюхи B_φ необходимо и достаточно обеспечить соблюдение следующих требований:

- для каждого базового предиката P_j, $1 \le j \le m$, верно соотношение $P_j \in K \iff F''_j \cap R \neq \emptyset$;
- для каждой формулы $\psi \mathbf{U}_{L_i} \chi \in Closure(\varphi)$, где $1 \le i \le k$, выполняются условия
  - если $\chi \in K$, то $F'_i \cap C \neq \emptyset$,
  - если $q_{init,i} \in C$ и при этом либо $\varepsilon \notin L_i$, либо $\psi \in K$, то $\psi \mathbf{U}_{L_i} \chi \in K$,
  - если $\psi \mathbf{U}_{L_i} \chi \in K$, то либо верны оба включения $\varepsilon \notin L_i$ и $\psi \in K$, либо $(q_{init,i}, 1-b) \in S$,
  - если $(q, f) \in S$ для некоторого состояния $q \in Q'_i$ и некоторой фазы $f \in \{0,1\}$, то либо $q \in F'_i$ и $\chi \in K$, либо $\psi \in K$.

(2) Отношение переходов $\Delta \subseteq Q \times (C \times \mathcal{A}^*) \times Q$ содержит переход

$$\big((K, C, S, R, b, \theta), (c, h), (K', C', S', R', b', \theta')\big)$$

в том и только том случае, если выполнены следующие требования:

- для каждого j, $1 \le j \le m$, верно соотношение $q \in Q''_j \cap R \implies \delta''_j(q, h) \in R'$,
- для любой формулы $\mathbf{X}_d \psi \in Closure(\varphi)$ верно соотношение $\mathbf{X}_d \psi \in K \implies d = c, \psi \in K'$,
- $\psi \mathbf{U}_{L_i} \chi \in Closure(\varphi)$, где $1 \le i \le k$, выполняются условия
  - если $q' = \delta'_i(q, c) \in Q'_i \cap C'$ и при этом либо $q \notin F'_i$, либо $\psi \in K$, то $q \in C$,
  - если $(q, f) \in S$ для некоторого состояния $q \in Q_i$ и некоторой фазы f, и при этом либо $q \notin F'_i$, либо $\chi \notin K$, то $(\delta'_i(q, c), f) \in S'$,
  - если $\theta = \bullet$, то $b' = 1 - b$ и $\theta' = \circ$,
  - если $\theta = \circ$, то $b' = b$; при этом в случае, если $\chi \notin K$ или множество S содержит пару (q, b), для которой $q \notin F'_i$, то $\theta' = \circ$, а в противном случае $\theta' = \bullet$.

(3) Множество начальных состояний $Q_{init} = \{(K, C, S, R, 0, \bullet) : \varphi \in K, R —$ достижимое множество$\}$.

(4) Множество допускающих состояний $F = \{(K, C, S, R, b, \bullet) : K \in K_\varphi\}$.

Теорема 1. *Формула φ выполнима тогда и только тогда, когда $Lang(B_\varphi) \neq \emptyset$.*

Доказательство. (эскиз) Справедливость утверждения теоремы устанавливается, следуя той же схеме рассуждений, которая применяется при обосновании аналогичных теорем для темпоральных логик *LTL* [12, 13], *DLTL* [8], и, наконец, самой логики *Reg-LTL* [14].

Для каждого допускающего прогона $\rho = z_0, z_1, \ldots, z_i, \ldots$ автомата Бюхи B_φ на ω-слове β, где $z_i = (K_i, C_i, S_i, R_i, b_i, \theta_i)$, $i \ge 0$, индукцией по структуре подформул $\psi \in Closure(\varphi)$ можно показать, что для любого $i \ge 0$ имеет место соотношение $\psi \in K_i \iff tr_\beta|^i \models \psi$. Кроме того, для каждой трассы tr, на которой выполняется формула φ, можно показать, что для ω-слова β, порождающего эту трассу, автомат Бюхи B_φ имеет допускающее вычисление ρ, в котором для каждого состояния $z_i = (K_i, C_i, S_i, R_i, b_i, \theta_i)$ верно $K_i = \{\psi : \psi \in Closure(\varphi), tr|^i \models \psi\}$. Поскольку для каждого начального состояния $z_0 = (K_0, C_0, S_0, R_0, b_0, \theta_0)$ верно включение $\varphi \in K_0$, отсюда следует утверждение теоремы.

Вместо того, чтобы приводить технические детали этого обоснования, мы ограничимся объяснением той роли, которую играют в функционировании автомата B_φ все компоненты его состояний $z = (K, C, S, R, b, \theta)$.

Ясно, что K — это множество всех тех формул $\psi \in Closure(\varphi)$, совместная выполнимость которых должна быть подтверждена на какой-либо трассе, т.е. $tr \models \psi$. Остальные компоненты состояний автомата совместно с определением отношения переходов Δ призваны обеспечить соблюдение этого требования. Выполнимость базовых предикатов контролируют множества R_i, содержащие ровно по одному состоянию для каждого автомата $D_j, 1 \le j \le m$. Определение отношения Δ гарантирует, что в прогоне ρ на ω-слове $\{(c_i, h_i) : i \ge 0\}$ эти множества R_i изменяются в соответствии с действиями h_i. В этом же определении обеспечены условия выполнимости формул вида $\mathbf{X}_c\psi$, входящих в множество K.

Автомат следит за выполнимостью формул вида $\psi \mathbf{U}_{L_j}\chi$ при помощи аппарата фаз $b \in \{0, 1\}$; в каждом допускающем прогоне фазы чередуются. Маркер θ призван отслеживать завершение фаз. Равенство $\theta = \bullet$ означает, что текущая фаза успешно завершена. Информация о том, что в текущем состоянии автомат обеспечивает семантические условия выполнимости формулы $\psi \mathbf{U}_{L_j}\chi$ во время фазы b, хранится в виде пары $(q', b), q' \in Q'_j$, в выполняющем множестве S. Фаза b не будет завершена, пока хотя бы одна пара (q, b) находится в выполняющем множестве. В определении отношения переходов указано условие, при котором из выполняющего множества S удаляется пара (q, b); это происходит либо в том случае, когда поступивший поток сигналов не принадлежит языку L_j, который используется в качестве параметра темпорального оператора $\mathbf{U}_{L_j}$ (эта альтернатива проявляется в отношении $q \notin L_j$), либо когда в текущем состоянии z возникает необходимость обеспечить выполнимость граничного условия χ (эта альтернатива проявляется в отношении $\chi \in K$). Во время фазы b могут появляться новые формулы с оператором $\mathbf{U}$, выполнение которых также нужно проверить. Для этих формул информация о необходимости проверки их выполнения записывается в множество S с фазой $1 - b$. При этом проверка выполнимости этих формул начинается сразу же, но на ее завершение отводится время до конца следующей фазы.

Определение множества допускающих состояний гарантирует, что в том случае, если автомат B_φ имеет допускающий прогон на слове β, все фазы бесконечно часто завершаются, и, значит, выполнимость всех формул вида $\psi \mathbf{U}_{L_j}\chi$, появляющихся в множествах K, постоянно подтверждается на трассе tr_β. Так как для допускающего прогона верно включение $\varphi \in K_0$, описанный выше принцип функционирования автомата B_φ гарантирует, что в случае непустоты ω-языка $Lang(B_\varphi)$ существует такое ω-слово β, для которого $tr_\beta \models \varphi$. □

Как видно из приведенного описания автомата Бюхи B_φ, число его состояний оценивается сверху величиной $2^{poly(|\varphi|)}$. Ранее, в статье [14], теоретико-автоматный подход был применен для демонстрации того, что задача верификации моделей разрешима для логики *Reg-LTL*. Однако метод, который был предложен в этой статье, приводил к построению автоматов Бюхи, размер которых оценивался двойной экспонентой от размера проверяемой формулы. То устройство автомата B_φ, которое описано в этом разделе, оказывается существенно более компактным за счет использования в структуре состояний (K, C, S, R, b, θ) автоматов компонентов фазы и маркера. С их помощью оказалось возможным хранить подмножества C, S, R множеств состояний V и W автоматов-распознавателей параметров темпоральных операторов и базовых предикатов формулы φ; в работе [14] для тех же самых целей приходилось использовать семейства подмножеств V и W, что и приводило к «двойному экспоненциальному взрыву».

4. Сложность задачи проверки выполнимости формул *Reg-LTL*

Используя конструкцию автомата B_φ из предыдущего раздела, построим алгоритм проверки выполнимости формул φ логики *Reg-LTL*, которому требуется полиномиальный относительно раз-

мера формул объем памяти. Автомат B_φ, в том виде, в котором он описан в разделе 3, не вполне пригоден для алгоритмических целей, поскольку он работает над бесконечным алфавитом $C \times \mathcal{A}^*$. Но мы покажем, что для проверки пустоты достаточно исследовать лишь конечный фрагмент этого автомата, который, впрочем, может иметь большое число состояний. Чтобы обнаружить в этом фрагменте допускающий прогон автомата, используя при этом ограниченный объем памяти, мы воспользуемся приемом недетерминированного угадывания нужного прогона, а затем применим теорему Савича [27] и сделаем процесс угадывания детерминированным перебором, при котором происходит лишь квадратичное увеличение объема используемой памяти.

Первая трудность, которую нужно преодолеть, состоит в том, что переходы автомата B_φ происходят по элементам бесконечного алфавита $(c, h) \in C \times \mathcal{A}^*$, в котором действия h могут иметь произвольно большую длину. Однако как будет видно из последующей леммы, бесконечный алфавит $C \times \mathcal{A}^*$ можно разбить на конечное число классов так, что для элементов из одного и того же класса отношение переходов автомата определено одинаково. Это обусловлено тем, что, как это видно из определения B_φ, действия h оказывают влияние в переходах из одних состояний в другие только на компоненты предикатных множеств достижимых состояний R, а количество различных множеств такого вида конечно.

Лемма 1. *Пусть z и z' — два состояния автомата Бюхи $B_\varphi = (Q, C \times \mathcal{A}^*, Q_{init}, \Delta, F)$ над алфавитом $C \times \mathcal{A}^*$, и при этом для некоторого элемента (c, h) из множества $C \times \mathcal{A}^*$ существует переход $(z, (c, h), z') \in \Delta$. Тогда существует такое действие $g \in \mathcal{A}^*$ длины, не превосходящей $2^{|\varphi| \log |\varphi|}$, для которого также есть переход $(z, (c, g), z') \in \Delta$.*

Доказательство. Рассмотрим переход $(z, (c, h), z') \in \Delta$ между состояниями $z = (K, C, S, R, b, \theta)$ и $z' = (K', C', S', R', b', \theta')$ в автомате B_φ по элементу $(c, h) \in C \times \mathcal{A}^*$, и пусть $R = \{q''_1, q''_2, \ldots, q''_m\}$ — предикатное достижимое множество, в котором $q''_j, 1 \le j \le m$, — состояния автоматов-распознавателей базовых предикатов формулы φ. Тогда согласно определению отношения переходов Δ автомата Бюхи B_φ имеем $R' = \{\delta''_1(q''_1, h), \delta''_2(q''_2, h), \ldots, \delta''_m(q''_m, h)\}$.

Рассмотрим кратчайшее по длине действие $g = a_1 a_2 \ldots a_N \in \mathcal{A}^*$, для которого равенства $\delta''_j(q''_j, h) = \delta''_j(q''_j, g)$ выполняются для любого $j, 1 \le j \le m$, и покажем, что $N = O(2^{|\varphi| \log |\varphi|})$. Для этого достаточно заметить, что в силу минимальности g среди предикатных множеств состояний

$$R_\ell = \{\delta''_1(q''_1, a_1 a_2 \ldots a_\ell), \delta''_2(q''_2, a_1 a_2 \ldots a_\ell), \ldots, \delta''_m(q''_m, a_1 a_2 \ldots a_\ell)\},$$

где $0 \le \ell \le N$, не может быть двух одинаковых. Поэтому длина N действия g не превосходит количества всех возможных предикатных множеств, которые образованы состояниями автоматов $D_j, 1 \le j \le m$, распознающих базовые предикаты $P_j, 1 \le j \le m$, формулы φ. Поскольку число состояний $|Q''_j|$ каждого такого автомата-преобразователя $D_j, 1 \le j \le m$, а также и число m самих автоматов, не превосходит размера формулы $|\varphi|$, приходим к выводу о том, что $N \le 2^{|\varphi| \log |\varphi|}$. □

Лемма 2. *Подмножество состояний $R = \{q''_1, q''_2, \ldots, q''_m\}$ автоматов $D_j, 1 \le j \le m$, распознающих базовые предикаты $P_j, 1 \le j \le m$, формулы φ является достижимым множеством в том и только том случае, если существует такое действие $g \in \mathbf{A}^*$ длины, не превосходящей $2^{|\varphi| \log |\varphi|}$, для которого $q''_j = \delta''_j(q''_{init,j}, g)$ для всех $j, 1 \le j \le m$.*

Доказательство. Проводится по той же схеме, что и доказательство леммы 1. □

Опираясь на теорему 1 и леммы 1, 2, приходим к основному утверждению этого раздела.

Теорема 2. *Задача проверки выполнимости формул логики Reg-LTL принадлежит классу сложности* Pspace.

Доказательство. Из теоремы 1 следует, что для проверки выполнимости произвольной формулы φ логики *Reg-LTL* достаточно проверить, допускает ли хотя бы одно ω-слово автомат Бюхи $B_\varphi = (Q, \mathcal{C} \times \mathcal{A}^*, Q_{init}, \Delta, F)$, описанный в разделе 3. Как было отмечено в этом же разделе, число состояний автомата B_φ оценивается сверху величиной $2^{poly(|\varphi|)}$. Для проверки непустоты языка $Lang(B_\varphi)$ достаточно проверить (см. [12, 13]), существуют ли такие начальное состояние z_{init} и допускающее состояние z_{acc} автомата B_φ, что состояние z_{acc} достижимо как из z_{init}, так и из z_{acc}. Для этого, в свою очередь, достаточно проверить, существует ли такой прогон $\rho = z_0, z_1, \ldots, z_{\ell-1}, (z_\ell, \ldots, z_{\ell+r})^\omega$ автомата Бюхи B_φ, в котором z_0— начальное состояние, а z_ℓ — допускающее состояние, и при этом число $\ell + r$ различных состояний в этом прогоне не превосходит общего числа состояний автомата B_φ. При помощи леммы 2 можно проверить, используя лишь полиномиальный объем памяти, является ли наугад выбранное состояние z_0 начальным состоянием автомата B_φ. Недетерминированный алгоритм способен отыскать такой прогон ρ, выбирая наугад состояния автомата B_φ, проверяя выполнимость отношения перехода $(z_i, (c, h), z_{i+1}) \in \Delta$ между всеми парами последовательно выбранных состояний z_i и z_{i+1} хотя бы для какой-нибудь пары $(c, h) \in \mathcal{C} \times \mathcal{A}^*$, и проводя учет числа выбранных для прогона состояний. Лемма 1 показывает, что для проверки существования перехода в автомате B_φ из состояния z_i в состояние z_{i+1} достаточно ограничиться рассмотрением лишь таких пар (c, h), в которых длина N действия $h = a_1 a_2 \ldots a_N$ не превосходит величины $2^{|\varphi| \log |\varphi|}$. В этом случае проверку существования перехода можно также провести при помощи недетерминированной вспомогательной процедуры, которая выбирает наугад элементарные действия и подает их на входы всех тех автоматов-распознавателей $D_1, D_2, \ldots, D_m$, которые специфицируют базовые предикаты $P_1, P_2, \ldots, P_m$ формулы φ. На каждом шаге работы эта процедура проводит учет количества выбранных действий и хранит в памяти набор тех состояний $(q_1'', \ldots, q_m'')$, в которых пребывают автоматы $D_1, D_2, \ldots, D_m$. Очевидно, что такой процедуре требуется объем памяти, полиномиальный относительно размера формулы φ. Но тогда и сам алгоритм проверки существования допускающего прогона автомата Бюхи B_φ также может быть выполнен с использованием полиномиального относительно размера формулы φ объема памяти. Таким образом, показано, что задача проверки выполнимости формул логики *Reg-LTL* принадлежит классу сложности NPspace, который, как показано в статье [27], совпадает с классом сложности Pspace. □

Теорема 3. *Задача проверки выполнимости формул логики Reg-LTL является* Pspace*-трудной.*

Доказательство. Нижнюю оценку сложности задачи проверки выполнимости можно получить путем сведения к ней проблемы пустоты пересечения семейства регулярных языков, распознаваемых заданными детерминированными автоматами Рабин-Скотта. Pspace-полнота этой задачи была установлена в статье [28].

Пусть задано произвольное семейство детерминированных автоматов-распознавателей $M = \{A_1, A_2, \ldots, A_n\}$ и пусть $P_i = Lang(A_i)$ для каждого $i, 1 \le i \le n$. Рассмотрим формулу $\varphi_M = \mathbf{F}_{C^*}(P_1 \wedge P_2 \wedge \cdots \wedge P_n)$. Из определения отношения выполнимости для темпорального оператора **F** видно, что формула φ_M выполнима тогда и только тогда, когда $\bigcap_{i=1}^{n} P_i \neq \emptyset$. □

Следствие 1. *Задача проверки выполнимости формул логики Reg-LTL является* Pspace*-полной.*

5. Сложность задачи верификации моделей для логики *Reg-LTL*

Сложность задачи верификации моделей реагирующих систем относительно *Reg-LTL* спецификаций мы установим с помощью модификации методов, использованных в предыдущем разделе. Эта задача состоит в том, чтобы проверить выполнимость соотношения $\pi \models \varphi$ для произвольного заданного автомата-преобразователя π и произвольной заданной формулы φ логики *Reg-LTL*.

Теорема 4. *Задача верификации автоматов-преобразователей относительно формул логики Reg-LTL принадлежит классу сложности* Pspace.

Доказательство. Эта задача равносильна задаче проверки пустоты пересечения множества всех трасс $Traces(\pi)$, порождаемых автоматом-преобразователем π, и множеством всех трасс $Traces(\neg\varphi)$, на которых выполняется формула $\neg\varphi$. Для такой проверки построим автомат Бюхи $B_{\pi,\varphi}$, который удовлетворяет следующему требованию: $\pi \models \varphi \iff Lang(B_{\pi,\varphi}) = \emptyset$. Как было показано в теореме 1, для каждой формулы $\neg\varphi$ можно построить автомат Бюхи $\hat{B}_{\neg\varphi}$, удовлетворяющий требованию $Lang(\hat{B}_{\neg\varphi}) = \{tr = \langle e, \alpha\rangle : tr \in Traces, tr \notin Traces(\varphi)\}$ и имеющий $2^{poly(|\varphi|)}$ состояний. Отличие автомата $\hat{B}_{\neg\varphi}$ от автоматов Бюхи, описанных в разделе 3, состоит в том, что начальными состояниями в нем являются лишь такие наборы $z = (K, C, S, R_0, b, \theta)$, в которых предикатное достижимое множество $R_0 = \{q''_{init,j} : 1 \le j \le m\}$ состоит из начальных состояний всех автоматов, распознающих базовые предикаты формулы φ. Для каждого автомата-преобразователя $\pi = (Q, \mathcal{C}, \mathcal{A}, q_{init}, T)$ множество трасс этого автомата распознается автоматом Бюхи $B_\pi = (Q, \mathcal{C} \times \mathcal{A}^*, q_{init}, \Delta_T, Q)$, отношение переходов Δ_T которого удовлетворяет условию $(q, (c, h), q') \in \Delta_T \iff (q, c, q', h) \in T$. Тогда, как известно из теории автоматов (см. [12, 13]), множество трасс $Traces(\pi) \cap Traces(\neg\varphi)$ также распознается автоматом Бюхи $B_{\pi,\varphi}$, число состояний которого не превосходит произведения числа состояний автоматов B_π и $B_{\neg\varphi}$. Для проверки пустоты языка $Lang(B_{\pi,\varphi})$ применяется недетерминированный алгоритм поиска допускающего прогона, аналогичный алгоритму, описанному в доказательстве теоремы 1. □

Pspace-трудность задачи верификации моделей реагирующих систем относительно *Reg-LTL* спецификаций можно установить путем сведения к ней все той же проблемы пустоты пересечения семейства регулярных языков, подобно тому, как это было показано в теореме 3.

Следствие 2. *Задача верификации автоматов-преобразователей относительно формул логики Reg-LTL является* Pspace*-полной.*

6. Заключение

В данной работе дано полное решение задач проверки выполнимости формул темпоральной логики *Reg-LTL* и верификации автоматов-преобразователей относительно формул этой логики: 1) построены полиномиальные по объему используемой памяти разрешающие алгоритмы и 2) доказана Pspace-трудность обеих задач.

Можно обозначить также некоторые направления дальнейших исследований, представляющие теоретический и прикладной интерес. Для логики *Reg-LTL* задачи проверки выполнимости и верификации моделей оказались PSpace-полными, а в случае использования в качестве параметров произвольных контекстно-свободных языков — неразрешимыми. Однако остался неисследованным случай, когда в качестве языков для параметризации темпоральных операторов и базовых предикатов выбираются контекстно-свободные языки некоторых специальных семейств, для которых проблема проверки пустоты пересечения имеет эффективное решение. Интерес представляет вопрос о том, где проходит граница между разрешимостью и неразрешимостью и как изменяется при этом сложность указанных задач.

Помимо сложности проверки выполнимости существуют и другие математические вопросы, традиционно исследуемые для языков спецификаций: выразительность, аксиоматизируемость и другие. Выразительность некоторых фрагментов языка *Reg-LTL* изучалась в статье [17], однако в ней формулы этого языка интерпретировались над обычными, а не двойными трассами. В то же время, для двойных трасс существует своя теория регулярных множеств, и вопрос об отношении *Reg-LTL* с этими множествами заслуживает внимания.

References

[1] A. Pnueli, «The temporal logic of programs», in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, 1977, pp. 46–57.

[2] D. Gabbay, A. Pnueli, S. Shelach, and J. Stavi, «The temporal analysis of fairness», in *Proceedings of 7-th ACM Symposium on Principles of Programming Languages*, 1980, pp. 163–173.

[3] M. Ben-Ari, Z. Manna, and A. Pnueli, «The temporal logic of branching time», in *Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 1981, pp. 164–176.

[4] J. Ouaknine and J. Worrell, «Some recent results in Metric Temporal Logic», in *Proceedings of the 6th International Conference Formal Modeling and Analysis of Timed Systems*, 2008, pp. 1–13.

[5] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems*. Springer, 1992.

[6] P. Wolper, «Temporal Logic Can Be More Expressive», *Information and Control*, vol. 56, pp. 72–99, 1983.

[7] T. French, «Quantified propositional temporal logic with repeating states», in *Proceedings of the 10th International Symposium on Temporal Representation and Reasoning*, 2003, pp. 155–165.

[8] J. Henriksen and P. S. Thiagarajan, «Dynamic linear time temporal logic», *Annals of Pure and Applied Logic*, vol. 96, pp. 187–207, 1999.

[9] G. De Giacomo and M. Y. Vardi, «Linear temporal logic and linear dynamic logic on finite traces», in *Proceedings of the 23th International Joint Conference on Artificial Intelligence*, 2013, pp. 854–860.

[10] A. Artale, R. Konchakov, V. Ryzhikov, and M. Zakharyaschev, «Tractable interval temporal propositional and description logics», in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015, pp. 1417–1423.

[11] S. Merz, M. Wirsing, and J. Zappe, «A Spatio-Temporal Logic for the Specification and Refinement of Mobile Systems», in *Proceedings of the 6th International Conference on Fundamental Approaches to Software Engineering*, 2003, pp. 87–101.

[12] C. Baier and J. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[13] E. M. Clarke, O. Gramberg, and D. A. Peled, *Model Checking*. MIT Press, 1999.

[14] D. G. Kozlova and V. A. Zakharov, «On the model checking of sequential reactive systems», in *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming (CS&P 2016), CEUR Workshop Proceedings*, vol. 1698, 2016, pp. 376–389.

[15] A. R. Gnatenko and V. A. Zakharov, «On the verification of finite transducers over semigroups», *Proceedings of ISP RAS*, vol. 30, pp. 303–324, 2018.

[16] A. Gnatenko and V. Zakharov, «On the Model Checking Problem for Some Extension of CTL*», *Modeling and Analysis of Information Systems*, vol. 27, pp. 428–441, 2020.

[17] A. Gnatenko and V. Zakharov, «On the Expressive Power of Some Extensions of Linear Temporal Logic», *Modeling and Analysis of Information Systems*, vol. 25, pp. 506–524, 2018.

[18] M. Vardi and P. Wolper, «An Automata-Theoretic Approach to Automatic Program Verification», in *Proceedings of the First Symposium on Logic in Computer Science*, 1986, pp. 322–331.

[19] J. A. W. Kamp, *Tense Logic and the Theory of Linear Order, PhD thesis*. University of California, Los Angeles, 1968.

[20] K. Etessami and T. Wilke, «An Until Hierarchy and Other Applications of an Ehrenfeucht-Fraisse Game for Temporal Logic», *Information and Computation*, vol. 160, pp. 88–108, 2000.

[21] M. Leucker and C. Sanchez, «Regular Linear Temporal Logic», in *Proceedings of the 4-th International Colloquium on Theoretical Aspects of Computing*, 2007, pp. 291–305.

[22] O. Kupferman, N. Piterman, and M. Y. Vardi, «Extended Temporal Logic Revisited», in *Proceedings of 12-th International Conference on Concurrency Theory*, 2001, pp. 519–535.

[23] A. R. Gnatenko and V. A. Zakharov, «O slozhnosti verifikatsii avtomatov-preobrazovateley nad kommutativnymi polugruppami», in *Trudy 18 Mezhdunarodnoj konferentsii Problemy teoreticheskoy kibernetiki*, 2017, pp. 68–70.

[24] V. Zakharov and G. Temerbekova, «On the Minimization of Finite State Transducers over Semigroups», *Modeling and Analysis of Information Systems*, vol. 23, pp. 741–753, 2016.

[25] V. Zakharov, «Equivalence checking problem for finite state transducers over semigroups», in *Proceedings of the 6-th International Conference on Algebraic Informatics*, 2015, pp. 208–221.

[26] J. M. Fisher and R. E. Ladner, «Propositional Dynamic Logic of Regular Programs», *Journal of Computer and System Sciences*, vol. 18, pp. 194–211, 1979.

[27] W. J. Savitch, «Relationships between nondeterministic and deterministic tape complexities», *Journal of Computer and System Science*, vol. 4, pp. 177–192, 1970.

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