И. В. АФАНАСЬЕВА
Специальная астрофизическая обсерватория Российской Академии Наук, Нижний Архыз
Ф. А. НОВИКОВ
Санкт-Петербургский политехнический университет Петра Великого, Санкт-Петербург
Л. Н. ФЕДОРЧЕНКО
Санкт-Петербургский Федеральный исследовательский центр Российской Академии Наук, Санкт-Петербург

ВЕРИФИКАЦИЯ СОБЫТИЙНО-УПРАВЛЯЕМЫХ ПРОГРАММНЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА СПЕЦИФИКАЦИИ ВЗАИМОДЕЙСТВУЮЩИХ АВТОМАТНЫХ ОБЪЕКТОВ

Язык спецификации взаимодействующих автоматных объектов CIAO (Cooperative Interaction Automata Objects) предназначен для описания поведения распределенных и параллельных систем, управляемых событиями. К такому классу систем относятся различные программно-аппаратные системы управления, мониторинга, сбора и обработки данных. В докладе предлагаются методы и алгоритмы, позволяющие автоматически проверять соответствие требованиям для программ на языке CIAO и, тем самым, верифицировать семантику разработанной программы.

Введение и постановка задачи. Программы, управляемые событиями, иначе называемые дискретными реагирующими системами [1], часто встречаются в задачах управления, мониторинга, сбора и обработки данных. Событийно-управляемая система реагирует на возникающие события (стимулы), выполняя определенные действия (реакции). Зачастую такие системы относятся к классу ответственных систем [2], для которых сама формулировка требований и проверка соответствия требованиям является нетривиальной задачей. Обычных словесных формулировок и выборочного тестирования для ответственных систем недостаточно, и необходимо применение формальных методов верификации. При этом для событийно-управляемых систем недостаточно указать логическое предусловие, которое должно выполняться до начала работы и логическое постусловие, которое должно выполняться в результате реализации некоторой последовательности событий/действий, поскольку один и тот же набор действий может быть выполнен как в допустимой последовательности, так и в нежелательной, запрещенной последовательности. Таким образом, формальные требования к системам рассматриваемого класса необходимо задавать в форме описания как допустимых, так и недопустимых последовательностей событий/действий. Известны различные методы формального описания множества последовательностей, которые варьируются в зависимости от того, насколько разнообразны элементы последовательностей и как сложно организованы последовательности. В данном случае элементарные события/действия можно трактовать как символы конечного алфавита, а последовательности действий – как слова в этом алфавите, и в таком случае, формальные требования – это некоторый язык в заданном алфавите [3]. Целью данной статьи является представление методов и алгоритмов, которые позволяют для определенного класса реагирующих систем, а именно, для систем, описанных на языке CIAO [4, 5], автоматически строить формальное описание множества последовательностей действий (набор возможных протоколов выполнения) в виде условных регулярных выражений [6].

Применение языка спецификации CIAO для построения событийно-управляемых систем. Практическое применение языка CIAO в области создания систем управления и обработки данных показало неплохие результаты, в частности, высокую степень надежности программного обеспечения, созданного с помощью этого языка [7].

Язык CIAO основан на использовании графов переходов состояний для описания поведения реактивных систем, причем в качестве графов переходов используются диаграммы автомата языка UML [8], расширенные дополнительными конструкциями и соглашениями для повышения выразительной силы языка. Наиболее существенным нововведением языка CIAO является, с одной стороны, множественность взаимодействующих автоматных объектов, а с другой стороны, строгая типизация интерфейсов взаимодействия.

В статье [5] мы рассмотрели применение языка CIAO для построения управляющих систем на примере задачи управления лифтом, описанной Д. Кнутом [9]. В этой задаче указаны действия, которые может выполнять лифт, условия, которые может проверять алгоритм управления, и требования, которым должен удовлетворять алгоритм управления. В результате применения описанной в статье [5] методики получилась спецификация алгоритма управления лифтом на языке CIAO, представленная на рис. 1.

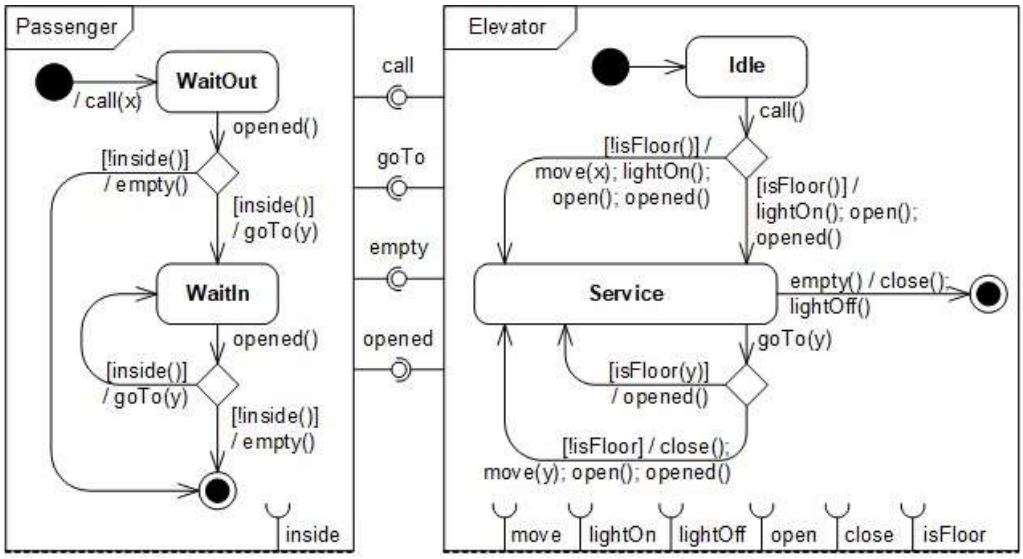

Рис. 1. Спецификация системы управления лифтом на языке CIAO

В этой спецификации присутствуют события/действия call, goTo, empty, opened, move, lightOn, lightOff, open, close и сторожевые условия isFloor, inside. Сокращенные идентификаторы событий/действий и условий, согласно подходу, предложенному в работе [10], приведены в таблице.

Таблица

Расшифровка идентификаторов событий/действий и сторожевых условий

Обозначение	Событие/действие	Обозначение	Действие	Обозначение	Условие
ez1	call	**z1**	move	**c1**	inside
ez2	goTo	**z2**	lightOn	**c2**	isFloor
ez3	empty	**z3**	lightOff		
ez4	opened	**z4**	open		
		z5	close		

В статье [5] требования к системе управления сформулированы следующим образом.

1. Все запросы перемещения на этажи внутри лифта должны быть обслужены.
2. Все запросы вызова лифта от этажей должны быть обслужены.

Также в статье [5] показано, как спецификация на рис. 1 гарантирует выполнение требований 1 и 2. Здесь мы идем далее. При разработке ответственных систем необходимо иметь возможность проверки соответствия требованиям на всех этапах разработки, включая возможность изменения требований при необходимости или включения новых требований к существующей системе. Рассмотрим, например, следующее дополнительное требование.

3. Если в кабине лифта находится пассажир, то свет должен быть включен.

Описание семантики реагирующих систем. Основу предлагаемых методов верификации составляет использование техники формального описания языков. В этих обозначениях и соглашениях элементарные действия – символы некоторого алфавита, протокол выполнения программы – слово в данном алфавите, а все множество протоколов, то есть семантика программы – язык над этим алфавитом. В терминах формальных языков требования к системе выражаются как некоторые утверждения о структуре слов языка, задающего семантику. Наибо-

лее удобно задавать структуру слов с помощью регулярного выражения, и тогда проверка соответствия требованиям сводится к задаче синтаксического анализа [11].

Поясним сказанное на примере. В обозначениях таблицы вводится класс событий/действий **zz**, в который попадают все действия, кроме **z2**, **z3**, **z4** и **z5**. Тогда требование 3 записывается в виде следующего регулярного выражения.

$$\mathbf{(zz)^*, z2, (zz)^*, z4, (zz)^*, z5, (zz)^*, z3, (zz)^*} \quad (1)$$

Фактически требование 3 означает, что действия включения/выключения света и входа/выхода пассажира должны выполняться всегда строго в указанной последовательности относительно друг друга и никак иначе.

Совершенно очевидно, что выполнение описанных таким образом требований легко устанавливается алгоритмом проверки протокола выполнения на принадлежность регулярному языку, задаваемому регулярным выражением (1). Поэтому, мы предполагаем, что требования к системе заданы набором образцов протоколов выполнения, возможно, со сторожевыми условиями, а сама система задана набором графов переходов на языке CIAO, на дугах которых указаны события/действия этих протоколов.

Алгоритм автоматической верификации семантики. Проверка соответствия требованиям для заданной системы на языке CIAO проводится в три этапа.

На первом этапе по заданным графам переходов взаимодействующих автоматных объектов строится единый языковой граф-источник [12]. Узлы в этом графе соответствуют событиям/действиям, а дуги могут быть помечены сторожевыми условиями. Граф получается один, поскольку действие в одном графе переходов через соответствующий интерфейс отождествляется с действием в другом графе переходов. Пути в построенном графе-источнике соответствуют протоколам выполнения системы, то есть словам языка верифицируемой семантики. На рис. 2 показан граф-источник для системы управления лифтом.

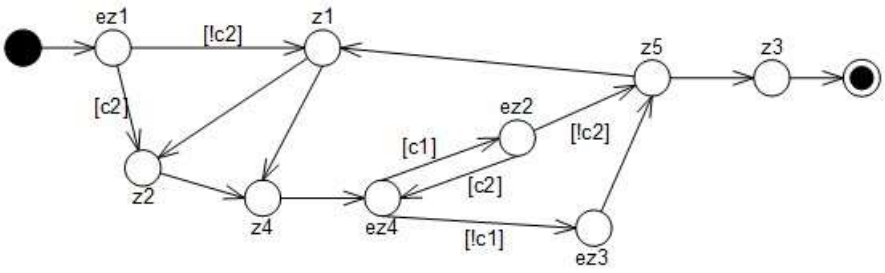

Рис. 2. Граф-источник системы управления лифтом

На втором этапе по графу строятся условные регулярные выражения, описывающие язык семантики верифицируемой системы. Если язык семантики оказывается регулярным (что часто встречается на практике, например, в коммуникационных протоколах [13]), то предлагаемые методы оказываются универсальными и позволяют автоматически извлекать регулярное выражение, описывающее семантику программы, непосредственно из данной программы на языке CIAO. Если же язык семантики проверяемой программы не регулярный, то получить единое регулярное выражение невозможно, но возможно получить некий набор выражений, снабженных сторожевыми условиями. Можно сказать, что поведение программы в целом удается описать как совокупность нескольких описаний поведения программы в особых «режимах», причем каждый режим характеризуется своим сторожевым условием. Разбиение программы на различные режимы работы не автоматическое, а определяется программистом на языке CIAO при задании сторожевых условий. В данном примере получается следующее условное регулярное выражение, соответствующее нижнему пути в графе-источнике на рис. 2.

$$\mathbf{ez1, [c2], z2, z4, ez4, [!c1], ez3, z5, z3} \quad (2)$$

На третьем этапе необходимо сопоставить требования, заданные шаблонами, с регулярными выражениями, которые были получены на втором этапе. Сопоставление заключается в проверке того, что полученные регулярные выражения, описывающие семантику (пути из начальной вершины в заключительную в графе-источнике), действительно имеют структуру, предписан-

ную шаблонами требований. Например, легко проверить, что полученное регулярное выражение (2) действительно является частным случаем шаблона (1), а значит, введенное требование 3 выполнено.

Заключение. Таким образом, если техническое задание на ответственную реагирующую систему описано в терминах допустимых и недопустимых последовательностей элементарных действий с помощью условных регулярных выражений, то есть указано желаемое и нежелательное поведение, то предлагаемые методы позволяют без тестирования, математически строго автоматически проверять соответствие разработанной системы на языке CIAO требованиям технического задания.

ЛИТЕРАТУРА

1. **Карпов Ю.Г.** *Model Checking. Верификация параллельных и распределенных программных систем.* СПб. : БХВ-Петербург, 2010. 560 с.
2. **Sommerville I.** *Software engineering, 10th ed.* Boston: Pearson, 2020.
3. **Fedorchenko L., Baranov S.** Equivalent Transformations and Regularization in Context-Free Grammars. *Cybernetics and Information Technologies*. 2015. Vol. 14. No. 4. Pp. 29–44. DOI: 10.1515/cait-2014-0003
4. **Новиков Ф.А., Афанасьева И.В.** Кооперативное взаимодействие автоматных объектов. *Информационно-управляющие системы.* 2016. №6. С. 50–63. DOI: 10.15217/issn1684-8853.2016.6.50
5. **Афанасьева И.В., Новиков Ф.А., Федорченко Л.Н.** Методика построения событийно-управляемых программных систем с использованием языка спецификации CIAO. *Труды СПИИРАН*. 2020.Т. 19. № 3. С. 481–514. DOI: 10.15622/sp.2020.19.3.1
6. **Aho A.V., Lam M.S., Sethi R., Ullman J.D.** *Compilers: principles, techniques, and tools, 2nd ed.* Boston : Pearson/Addison-Wesley, 2007.
7. **Афанасьева И.В., Новиков Ф.А.** Архитектура программного обеспечения систем оптической регистрации. *Информационно-управляющие системы.* 2016. № 3. С. 51–63. DOI: 10.15217/issn1684-8853.2016.3.51
8. **Новиков Ф.А., Иванов Д.Ю.** *Моделирование на UML. Теория, практика, видеокурс.* СПб.: Наука и Техника. 2010. 640 с.
9. **Knuth D.E.** *The Art of Computer Programming: Fundamental Algorithms, 3rd ed.* Addison-Wesley Professional, 1998. Vol. 1. 652 p.
10. **Поликарпова Н.И., Шалыто А.А.** *Автоматное программирование.* СПб. : Питер, 2011. 176 с.
11. **Fan W., Li J., Ma S., Tang N., Wu Y., Wu Y.** Graph pattern matching: From intractable to polynomial time. Proceedings of the VLDB Endowment, 2010. Vol. 3(1-2). Pp. 264-275. DOI: 10.14778/1920841.1920878
12. **Авдошин С.М., Набебин А.А.** *Дискретная математика. Формально-логические системы и языки.* М. : ДМК Пресс, 2018. 390 с.
13. **Levonevskiy D., Novikov F., Fedorchenko L., Afanasieva I.** Verification of Internet Protocol Properties Using Cooperating Automaton Objects. Proceedings of the 12th International Conference on Security of Information and Networks (SIN'19). ACM, 2019. Pp. 1–4. DOI: 10.1145/3357613.3357639

I.V.Afanasieva (Special Astrophysical Observatory of the Russian Academy of Sciences (SAO RAS), F.A.Novikov (Peter the Great St. Petersburg Polytechnic University (SPbPU), St. Petersburg), Nizhny Arkhyz), L.N.Fedorchenko (St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS), St. Petersburg)

Verification of the event-driven software systems using the specification language of cooperating automaton objects

The CIAO (Cooperative Interaction Automata Objects) specification language is intended to describe the behavior of the distributed and concurrent event-driven systems. This class of systems includes various software and hardware systems for control, monitoring, data collection and processing. The paper proposes methods and algorithms that allow one to automatically check compliance with the requirements for programs in the CIAO language and, thereby, verify the semantics of the developed program.

Авторы готовы представить текст на английском языке для сборника материалов мультиконференции, который будет подан для индексирования в Scopus.