DOI 10.37882/2223-2966.2023.08.38

СИНТЕЗ БЕЗОПАСНЫХ КОМПОНЕНТОВ ВЕБ-СЕРВИСОВ НА ОСНОВЕ РЕШЕНИЯ АВТОМАТНЫХ УРАВНЕНИЙ

DERIVING SAFE COMPONENTS OF WEB SERVICES BASED ON THE SOLUTION OF FSM EQUATIONS

E. Shirokova
N. Evtushenko

Summary. A web service is a software system that provides interaction between a service provider and its users. When developing web services, it is important to consider its security, one aspect of which is the absence of livelocks and deadlocks between its components. In this paper we consider the problem of synthesizing a safety web service server application that must interact with various client applications without livelocks.

It is assumed that the behavior of the web service and its known components is described by complete finite state machines (FSMs); the interaction of the web service components is described using the parallel composition operation. Thus, the problem of synthesizing a server application is reduced to solving the corresponding system of parallel FSM equations (inequalities), for which complete progressive solutions are of interest, since the use of such solutions for the synthesis of a server application guarantees the absence of livelocks and deadlocks in the work of a web service.

As is known, the largest (general) solution of a solvable system of equations is the intersection of the largest solutions of all equations of a system. The complexity of deriving the largest progressive solution is higher than the complexity of deriving the «usual» largest solution. In this regard, the problem of the possibility of reducing the solution of a system of FSM equations to the solution of a single equation is of interest. It is shown that, similarly to the case of finding the largest solution of the system, such a reduction is possible for two special cases when it is necessary to synthesize a safety server application that can: 1) work with different client applications and provide the same level of service, and in this case, the corresponding system of FSM equations is considered; 2) or work with one client application and provide different levels of service, and in this case, the corresponding system of FSM inequalities is considered.

Keywords: web service, web service safety, finite state machine, parallel composition of finite state machines, finite state machine equation, system of finite state machine equations, progressive solution.

Широкова Екатерина Владимировна
Национальный исследовательский Томский государственный университет
k@shir.su
Евтушенко Нина Владимировна
доктор технических наук, профессор, Институт системного программирования им. В.П. Иванникова РАН, Национальный исследовательский университет Высшая школа экономики, г. Москва
nyevtush@gmail.com

Аннотация. Веб-сервис представляет собой программную систему, обеспечивающую взаимодействие между поставщиком услуги и ее пользователями. При разработке веб-сервисов важно учитывать его безопасность, одним аспектов которой является отсутствие зацикливаний (livelock) и тупиковых ситуаций (deadlock) между его компонентами. В данной работе рассматривается задача синтеза безопасного серверного приложения веб-сервиса, который должен взаимодействовать с различными клиентскими приложениями без зацикливаний и тупиковых ситуаций.

Предполагается, что поведение веб-сервиса и его известных компонентов описано полностью определенными конечными автоматами; взаимодействие компонентов веб-сервиса описано с помощью операции параллельной композиции. Таким образом, задача синтеза серверного приложения сводится к решению соответствующей системы параллельных автоматных уравнений (неравенств), для которой интерес представляют полностью определенные живые решения, так как использование таких решений для синтеза серверного приложения гарантирует отсутствие зацикливаний и тупиковых ситуаций в работе веб-сервиса.

Как известно, набольшее (общее) решение разрешимой системы уравнений является пересечением наибольших решений всех уравнений системы. Сложность построения наибольшего живого решения более высокая, чем сложность построения «обычного» наибольшего решения. В связи с этим интересной является задача о возможности сведения решения системы автоматных уравнений к решению одного уравнения. Показано, что, подобно случаю нахождения наибольшего решения системы, такое сведение возможно для двух частных случаев, когда необходимо синтезировать безопасное серверное приложение, которое может: 1) работать с разными клиентскими приложениями и предоставлять одинаковый уровень сервиса, и в этом случае рассматривается соответствующая система автоматных уравнений; 2) или работать с одним клиентским приложением и предоставлять разные уровни сервиса, и в этом случае рассматривается соответствующая система автоматных неравенств.

Ключевые слова: веб-сервис, безопасность веб-сервисов, конечный автомат, параллельная композиция автоматов, автоматное уравнение, система автоматных уравнений, живое решение.

Введение

На сегодняшний день все чаще различные услуги предоставляются пользователям посредством сети Интернет, и это способствует появлению большого числа веб-сервисов. Веб-сервис [1] представляет собой программную систему, обеспечивающую взаимодействие между поставщиком услуги и ее пользователями. Для веб-сервисов характерна так называемая клиент-серверная архитектура. Пользователь

услуги посредством клиентского приложения отправляет на сервер запрос, удовлетворяющий правилам протокола передачи данных. После обработки запроса или ряда запросов сервер отправляет клиенту ответ, который является запрашиваемой информацией либо сообщением об ошибке. При разработке веб-сервиса важно учитывать его безопасность. В данной работе мы полагаем, что веб-сервис является *безопасным* [2], если на его компоненты могут поступать только определенные в их спецификациях входные данные. Кроме того, при взаимодействии компонентов веб-сервиса не должно возникать зацикливаний и тупиковых ситуаций. Если взаимодействие компонентов некоторого веб-сервиса не безопасно, то представляет интерес задача синтеза соответствующего безопасного приложения.

Задача синтеза компонентов многомодульных систем рассматривалась в различных работах. Она известна как задача решения уравнения [3], построение подмодуля [4], задача синтеза неизвестного компонента [5]. В работах [5, 6] рассматривается проблема построения мультиагентной системы, для создания которой необходимо синтезировать так называемого агента, который способен работать в различных контекстах и при работе в каждом контексте предоставлять свой уровень сервиса. Для решения данной задачи авторы предлагают описывать поведение компонентов мультиагентной системы посредством конечного автомата, а взаимодействие компонентов — с помощью операции параллельной композиции автоматов-компонентов. Операция параллельной композиции позволяет описывать взаимодействие компонентов многомодульной системы в режиме диалога, причем в каждый момент времени активным является только один компонент, а внешняя выходная реакция производится только после завершения внутреннего диалога между компонентами. Таким образом, задача синтеза агента при работе в различных контекстах сводится к задаче решения системы параллельных автоматных уравнений. Авторы [5] предлагают решать каждое уравнение отдельно, а затем строить пересечение полученных наибольших (общих) решений. В работе [7] мы показываем, что в некоторых частных случаях систему автоматных параллельных уравнений можно свести к решению одного уравнения, что может снизить сложность нахождения решения по сравнению с решением системы уравнений.

В настоящей работе рассматривается задача синтеза безопасного серверного приложения веб-сервиса, который должен взаимодействовать с различными клиентскими приложениями без зацикливаний и тупиковых ситуаций. Предполагается, что поведение веб-сервиса и его известных компонентов описано полностью определенными конечными автоматами. Взаимодействие компонентов веб-сервиса описано с помощью операции параллельной композиции. Таким образом, задача синтеза серверного приложения сводится к решению соответствующей системы параллельных автоматных уравнений (неравенств).

Для решения данной задачи результаты, полученные в работе [7], расширяются на случай поиска полностью определенных живых решений систем параллельных автоматных уравнений. Живые (compositionally progressive) решения автоматных уравнений [6, 8] представляют особый интерес, поскольку композиция известного компонента автоматной композиции с живым детерминированным решением уравнения является полностью определенным автоматом, то есть композиция не содержит зацикливаний. Таким образом, использование полностью определенных подавтоматов живых решений для синтеза серверного приложения веб-сервиса, который должен взаимодействовать с различными клиентскими приложениями, гарантирует отсутствие зацикливаний и тупиковых ситуаций в работе веб-сервиса.

Основные определения и обозначения

1) Конечный автомат

Конечные автоматы широко используются для описания поведения реактивных систем, которые получают некоторые запросы (входные символы) и производят соответствующие выходные реакции (выходные символы) [9]. Состояния конечного автомата соответствуют памяти о ранее примененных входных действиях и произведенных выходных реакциях. *Конечный автомат* (или просто автомат) [3, 9] A представляет собой пятерку (A, I, O, T_A, a_0), где A — непустое конечное множество состояний с выделенным начальным состоянием a_0; I и O — непустые конечные множества входных и выходных символов (входной и выходной алфавиты), соответственно; $T_A \subseteq A \times I \times O \times A$ — отношение переходов. На основе свойств отношения переходов определяются *подавтоматы* автомата, *полностью определенные* и *частичные* автоматы, *детерминированные* и *недетерминированные* (*наблюдаемые*) автоматы [3, 5, 9]. Конечный автомат является трассовой моделью, представляя конечным способом отображение бесконечного множества входных последовательностей в множество выходных последовательностей. Для входной последовательности α и выходной последовательности β четверка $(\alpha, a, a', \beta) \in T_A$, если найдется последовательность состояний $a_1, \ldots, a_{n+1}$ такая, что $a_1 = a$, $a_{n+1} = a'$ и $(i_j, a_j, a_{j+1}, o_j) \in T_A$ для всех $j = 1, \ldots, n$. Пара α/β называется *входо-выходной* последовательностью автомата A в состоянии a. Множество всех входо-выходных последовательностей автомата A в начальном состоянии a_0 называется *языком* L_A автомата A. Автомат B называется *редукцией* автомата A ($A \leq B$), если $L_B \subseteq L_A$. Если $L_B = L_A$, то автоматы A и B называются *эквивалентными* ($A \approx B$). В детерминированном

полностью определенном автомате для каждой входной последовательности существует единственная допустимая выходная последовательность.

Если каждое состояние автомата достижимо из начального состояния по некоторой входо-выходной последовательности, то автомат называется *связным*. *Пересечение* $A \cap B$ [10] автоматов A и B с одинаковыми входными и выходными алфавитами представляет собой наибольший связный подавтомат автомата $(A \times B, I, O, T_{A\cap B}, a_0b_0)$, где $(ab, i, o, a'b') \in T_{A\cap B} \Leftrightarrow (a, i, o, a') \in T_A$ & $(b, i, o, b') \in T_B$. Пересечение полностью определенных автоматов может оказаться частичным. *Объединение* $A \cup B$ [10] автоматов A и B, у которых множество состояний не пересекается, представляет собой автомат $(A \cup B \cup \{a_0b_0\}, I, O, T_{A\cup B}, a_0b_0)$, где $\forall t, t' \in A \cup B$ выполняется: $(a_0b_0, i, o, t') \in T_{A\cup B} \Leftrightarrow (a_0, i, o, t') \in T_A$ или $(b_0, i, o, t') \in T_B$; $(t, i, o, t') \in T_{A\cup B} \Leftrightarrow (t, i, o, t') \in T_A \cup T_B$.

2) Операции над конечно-автоматными языками

Достаточно часто конечно-автоматный язык представляется посредством полуавтомата на основе «растягивания» переходов автомата [3, 5]. В полуавтомат добавляются промежуточные состояния, помечающие пару «состояние, входной символ», в которое есть переход из текущего состояния под действием этого входного символа; все состояния исходного автомата и только они объявляются *финальными*. Из промежуточного состояния есть переход в финальное состояние полуавтомата под действием выходного символа, такого, что соответствующая четверка принадлежит отношению переходов исходного автомата. Дополнение конечно-автоматного языка $Comp(L_A)$ строится на основе соответствующего полуавтомата, и вообще говоря, не является конечно-автоматным языком, ввиду нарушения свойства префикс-замкнутости.

Пусть V и W — некоторые алфавиты и последовательность $\alpha \in V^*$. *U-ограничение* последовательности α (обозначение $\alpha_{\downarrow U}$) строится посредством удаления из α всех символов, принадлежащих множеству $V \backslash U$. *U–расширение* последовательности α (обозначение $\alpha_{\uparrow U}$) представляет собой множество всех последовательностей над алфавитом $(V \cup U)$ с V-ограничением последовательности α. Множество всех последовательностей $L_{\downarrow U}$ есть U–ограничение языка L в алфавите V; и соответственно множество всех последовательностей $L_{\uparrow U}$ есть U–расширение языка L. Данные операции также выполняются над соответствующими полуавтоматами [5, 11].

3) Параллельная композиция конечных автоматов

Если поведение компонентов многомодульной системы описано конечными автоматами, то для описания их взаимодействия в диалоговом режиме может быть использована операция параллельной композиции автоматов-компонентов [5]. В настоящей работе рассматривается бинарная параллельная композиция автоматов, где один из компонентов является встроенным (рисунок 1), не имеющим внешних алфавитов. Такая структура композиции часто применяется для описания взаимодействия компонентов веб-сервисов, представленных конечными автоматами, так как внешние входные символы могут поступать только на клиентское приложение, то есть сервер является встроенным компонентом [12].

Рассмотрим параллельную композицию $S \approx C \lozenge X$ полностью определенных автоматов $C = (C, I \cup V, O \cup U, T_C, c_0)$ и $X = (X, U, V, T_X, x_0)$, представленную на рисунке 1. Алфавиты I, O, V, U попарно не пересекаются; при этом алфавиты I и O являются, соответственно, *внешними* входным и выходным алфавитами композиции, а алфавиты V и U — *внутренними* алфавитами. Мы полагаем, что встроенный компонент X соответствует серверному приложению веб-сервиса, компонент C соответствует клиентскому приложению, с которым должно работать серверное приложение и обеспечивать определенный уровень сервиса S.

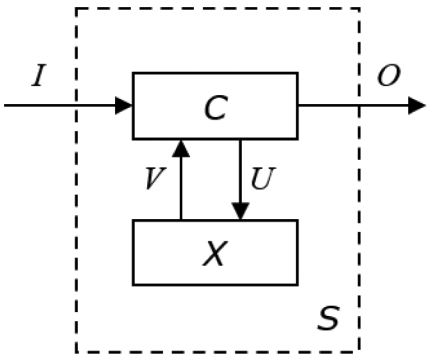

Рис. 1. Параллельная композиция автоматов, где компонент X является встроенным

Компоненты параллельной композиции взаимодействуют при так называемой «медленной» внешней среде. При подаче внешнего входного символа $i \in I$ на компонент C, данный компонент вырабатывает:

1) либо внешний выходной символ $o \in O$ и после этого композиция готова принять следующий внешний входной символ;

2) либо внутренний выходной символ $u \in U$, который поступает на вход компонента X, и компоненты взаимодействуют в режиме диалога до тех пор, пока не компонент C не произведет внешний выходной символ. После этого композиция готова принять следующий внешний входной символ.

В случае, когда при подаче некоторого внешнего входного действия между компонентами композиции

возникает бесконечный диалог, то говорят, что в композиции присутствуют зацикливания (осцилляции). Предполагается, что такие входные действия не поступают на безопасную композицию.

Существуют различные способы построения автомата параллельной композиции [5, 13]. В настоящей работе используется подход на основе перехода к полуавтоматам, который для рассматриваемой структуры композиции содержит следующие шаги. Сначала для автоматов-компонентов композиции C и X строятся соответствующие полуавтоматы $Aut(C)$ и $Aut(X)$. Далее полуавтомат $Aut(X)$ расширяется на внешние алфавиты I и O. На следующем шаге необходимо построить пересечение полуавтоматов $Aut(C)$ и $Aut(X)_{\uparrow I\cup O}$ и ограничить полученное пересечение на внешние алфавиты I и O. На последнем шаге полученный результат необходимо пересечь с полуавтоматом $Aut(Max(I, O))$ с языком $(IO)^*$, который моделирует «медленную» внешнюю среду. Полученный полуавтомат преобразуется в соответствующий автомат S путем объединения входных символов с последующими выходными символами.

Автомат бинарной параллельной композиции полностью определенных автоматов может оказаться частичным, если в композиции присутствуют зацикливания. В этом случае I–ограничение полуавтомата $(Aut(C)\cap Aut(X)_{\uparrow I\cup O})_{\downarrow I\cup O}\cap Aut(Max(I,O))$ может не совпадать с I^*. В безопасной композиции зацикливания отсутствуют, и I–ограничение такого полуавтомата совпадает с I^*.

4) Параллельные автоматные уравнения

Задача синтеза серверного приложения веб-сервиса, которое должно взаимодействовать с различными клиентскими приложениями, и при работе с каждым клиентом предоставлять некоторый уровень сервиса может быть сведена к задаче решения соответствующей системы параллельных автоматных уравнений (рисунок 2).

Пусть $C=(C, I\cup V, O\cup U, T_C, c_0)$ и $S=(S, I, O, T_S, s_0)$ — полностью определенные автоматы. Выражение «$C\lozenge X\approx S$» называется *параллельным автоматным уравнением* по отношению к неизвестному автомату X с входным и выходным алфавитами U и V, соответственно. Автомат C называется *контекстом*, автомат S — *спецификацией*. Автоматное уравнение может не иметь решений. Если уравнение разрешимо, то для него существует *набольшее решение Largest*, которое содержит все возможные решения уравнения. Язык наибольшего решения *Largest* определяется следующим образом: $Comp(Aut(C)\lozenge Comp(Aut(S))) = Comp((Aut(C)\cap Comp(Aut(S))_{\uparrow U\cup V})_{\downarrow U\cup V})$ [5]. Если композиция $C\lozenge Largest$ эквивалентна спецификации S, то уравнение $C\lozenge X\approx S$ разрешимо.

Неопределенные входные последовательности наибольшего решения уравнения $C\lozenge X\approx S$ соответствуют входным последовательностям, недопустимым в спецификации. Такие последовательности могут приводить к стрессовым ситуациям в работе веб-сервиса. В связи с этим наибольший интерес представляют полностью определенные решения автоматных уравнений. Если разрешимое автоматное уравнение имеет полностью определенное решение, то оно имеет наибольшее полностью определенное решение. Каждое полностью определенное решение является подавтоматом наибольшего полностью определенного решения, однако не каждый полностью определенный подавтомат наибольшего решения является решением уравнения [3]. Поэтому интерес представляют полностью определенные и *живые* (*compositionally progressive*) решения автоматных уравнений [6, 8, 14, 15], поскольку в этом случае композиция контекста C с таким решением является полностью определенным автоматом, то есть такая композиция не содержит зацикливаний (осцилляций). Если уравнение имеет живое решение, то, как известно, уравнение имеет и наибольшее живое решение. Всякий полностью определенный подавтомат наибольшего живого решения является решением уравнения. Если живого решения уравнения не существует, то нужно либо «запретить» некоторые действия контекста, либо изменить спецификацию совместного поведения компонентов.

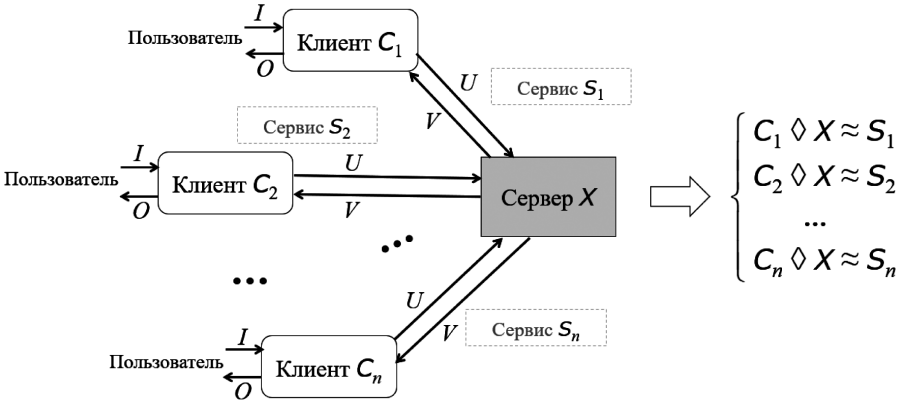

Рис. 2. Многокомпонентная сервисная система

5) Постановка задачи

Пусть задана система из n параллельных уравнений (рисунок 2). Требуется найти наибольшее живое полностью определенное решение системы уравнений, если система разрешима. В работе [5] предлагается способ проверки разрешимости системы и построения для разрешимой системы наибольшего решения на основе таких решений для уравнений системы. Пересечение наибольших решений и является наибольшим решением системы уравнений. В работе [7] показывается, что разрешимость такой системы уравнений может быть проверена на основе одного параллельного уравнения, но в работе отсутствуют доказательства этого факта. Кроме того, возможность построения наибольшего живого решения системы в работах не обсуждается. Процесс нахождения живого наибольшего решения системы уравнений более сложный, чем нахождение «обычного» наибольшего решения, так как полученное пересечение может оказаться решением для каждого уравнения, но не будет являться живым решением [16].

В настоящей работе, рассматриваются два частных случая, когда возможно сведение задачи поиска наибольшего полностью определенного живого решения системы параллельных автоматных уравнений / неравенств к поиску наибольшего полностью определенного живого решения одного уравнения / неравенства, что в некоторых случаях позволяет понизить сложность решения системы. Использование таких решений для синтеза серверного приложения гарантирует отсутствие зацикливаний и тупиковых ситуаций в работе веб-сервиса. Подробные доказательства приводятся для параллельной композиции из двух компонентов (рисунок 1), где неизвестным является встроенный компонент X, соответствующий серверному приложению, которое необходимо синтезировать. Для системы из n уравнений, $n > 2$, можно использовать математическую индукцию.

Синтез безопасного серверного приложения веб-сервиса

1) В первом случае необходимо синтезировать безопасное серверное приложение, которое может взаимодействовать с двумя различными клиентами и при работе с каждым клиентом предоставлять одинаковый уровень сервиса. Для решения данной задачи можно использовать решение следующей системы уравнений.

Пусть $C_i \lozenge X \approx S$ ($i = 1, 2$) — система параллельных автоматных уравнений, где S — детерминированный полностью определенный автомат.

Теорема 1. Пусть $C_i \lozenge X \approx S$ ($i = 1, 2$) — система параллельных автоматных уравнений, где спецификации S есть детерминированный полностью определенный автомат. Наибольшее полностью определенное живое решение системы уравнений эквивалентно наибольшему живому решению уравнения $(C_1 \cup C_2) \lozenge X \approx S$.

Доказательство.

Пусть автомат *Largest* — наибольшее полностью определенное живое решение системы уравнений $C_i \lozenge X \approx S$ ($i = 1, 2$). А значит, он является полностью определенным живым решением каждого уравнения $C_i \lozenge X \approx S$. Поэтому для каждой последовательности α языка автомата C_i и каждой последовательности β языка автомата *Largest*, композиция автоматов, соответствующих последовательностям α и β, эквивалентна автомату S, $i = 1, 2$.

В силу того, что язык автомата объединения $C_1 \cup C_2$ содержит все последовательности языков автоматов C_1 и C_2, то для каждой последовательности α языка автомата объединения $C_1 \cup C_2$ и каждой последовательности β автомата *Largest*, композиция соответствующих последовательностей α и β содержится в языке автомата S.

Таким образом, автомат *Largest* является наибольшим полностью определенным живым решением параллельного автоматного уравнения $(C_1 \cup C_2) \lozenge X \approx S$. □

Данный подход можно расширить на систему из конечного числа параллельных автоматных уравнений.

Следствие 1.1. Пусть $C_i \lozenge X \approx S$ ($i = 1, 2, \ldots n$) — система параллельных автоматных уравнений, где S — детерминированный полностью определенный автомат. Наибольшее полностью определенное живое решение системы уравнений эквивалентно наибольшему полностью определенному живому решению уравнения $(C_1 \cup \ldots \cup C_n) \lozenge X \approx S$.

Проиллюстрируем теорему 1 на следующем примере. Рассмотрим контексты C_1 и C_2, представленные на рисунках 3(а) и 3(б), соответственно. C_1 и C_2 определены над множеством внешних входных символов $I = \{i_1, i_2\}$, множеством внешних выходных символов $O = \{o_1, o_2\}$, множеством внутренних входных символов $V = \{v_1, v_2\}$ и множеством внутренних выходных символов $U = \{u_1, u_2\}$. Спецификация S (рисунок 3(в)) определена над множеством внешних входных символов $I = \{i_1, i_2\}$ и множеством внешних выходных символов $O = \{o_1, o_2\}$.

Решение каждого автоматного уравнения и системы $C_i \lozenge X \approx S$ ($i = 1, 2$) в целом определено над множеством внутренних входных символов $U = \{u_1, u_2\}$ и множеством внутренних выходных символов $V = \{v_1, v_2\}$. Наибольшие полностью определенные живые решения $Largest_1$ и $Largest_2$ уравнений $C_1 \lozenge X \approx S$ и $C_2 \lozenge X \approx S$ приведены на рисунках 4(а) и 4(б), соответственно. Наибольшее полностью определенное живое решение *Largest* системы уравнений показано на рисунке 4(в).

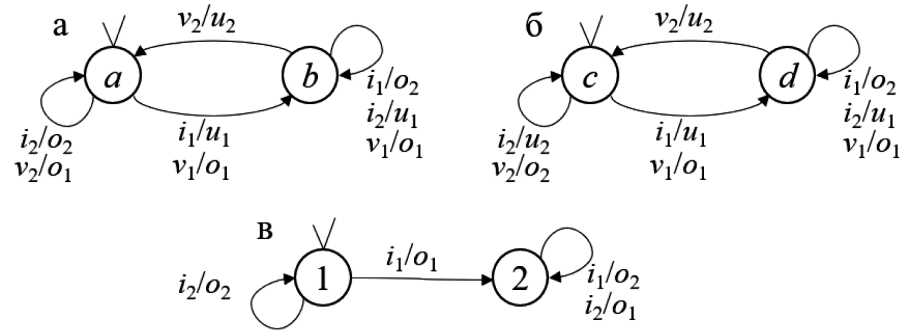

Рис. 3. (а) Контекст C_1; (б) Контекст C_2; (в) Спецификация S

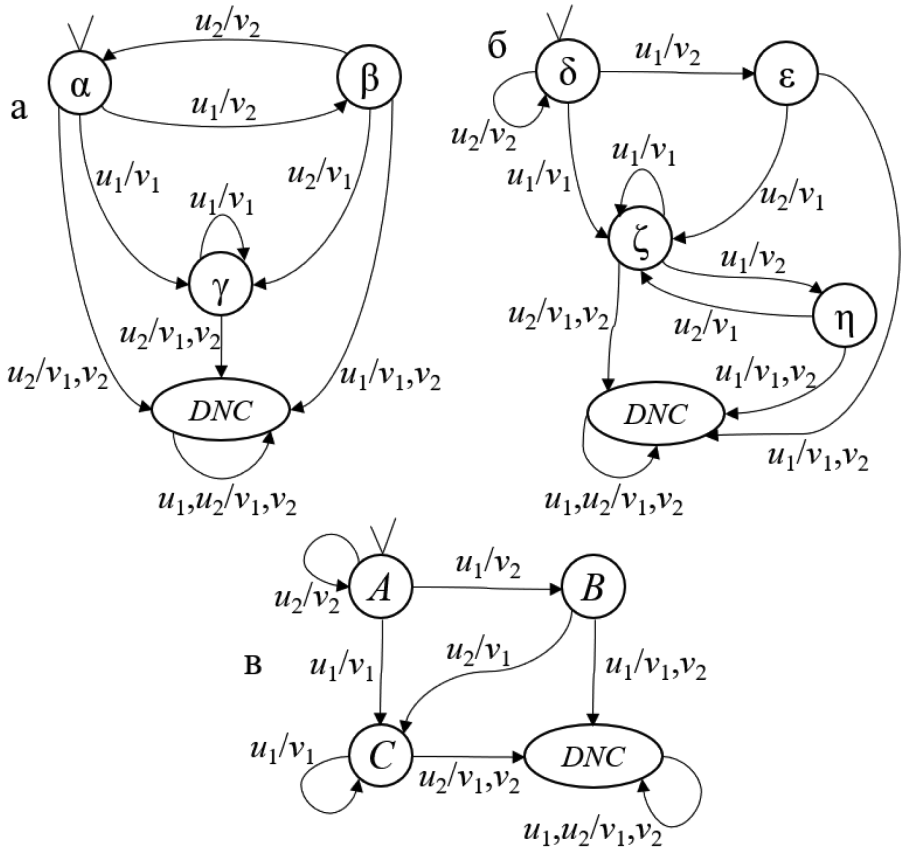

Рис. 4. (а) Наибольшее полностью определенное живое решение $Largest_1$
(б) Наибольшее полностью определенное живое решение $Largest_2$
(в) Наибольшее полностью определенное живое решение *Largest* системы уравнений

Рассмотрим теперь уравнение $(C_1 \cup C_2) \lozenge X \approx S$. Объединение контекстов C_1 и C_2 показано на рисунке 5(а). Наибольшее полностью определенное живое решение *Largest*′ уравнения $(C_1 \cup C_2) \lozenge X \approx S$ приведено на рисунке 5(б).

Таким образом, посредством пересечения полученных решений *Largest* и *Largest*′, можно убедиться, что решение *Largest* системы уравнений эквивалентно решению *Largest*′ уравнения $(C_1 \cup C_2) \lozenge X \approx S$.

2) Рассмотрим второй случай. Необходимо синтезировать безопасное серверное приложение веб-сервиса, которое может предоставлять клиенту различные уровни сервиса. Для решения данной задачи можно использовать решение следующей системы неравенств.

Пусть $C \lozenge X \leq S_i$ $(i = 1, 2)$ — система параллельных автоматных неравенств, где S_1 и S_2 — полностью определенные, возможно недетерминированные автоматы. Если S_1 и S_2 — детерминированные автоматы, то система неравенств неразрешима, поскольку параллельная композиция автомата C и живого полностью определенного решения определена однозначно. На практике недетерминизм спецификации соответствует опциональности предоставляемого пользователю сервиса.

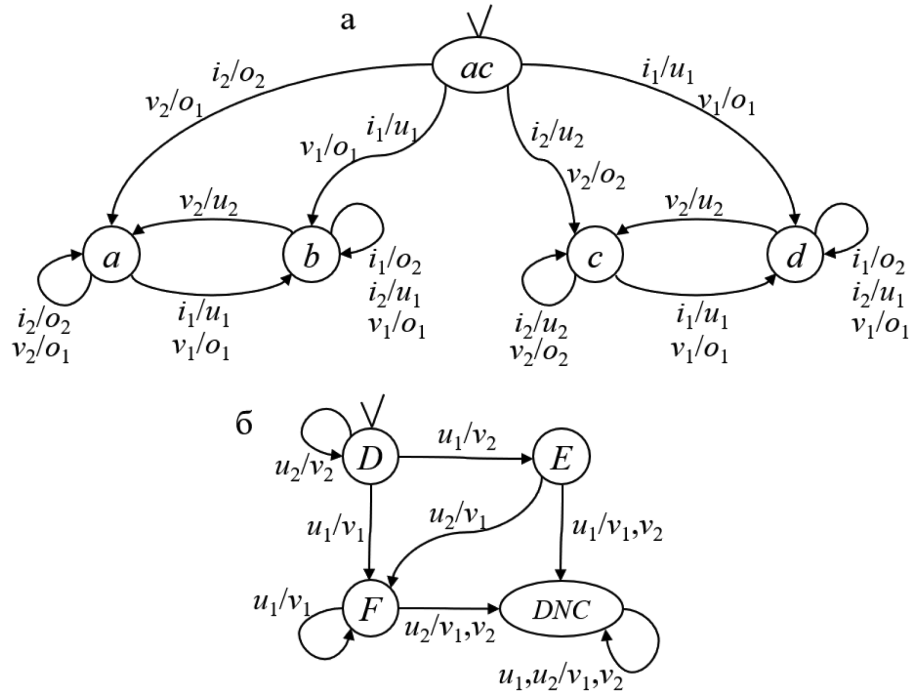

Рис. 5. (а) Автомат объединения контекстов $C_1 \cup C_2$
(б) Наибольшее полностью определенное живое решение *Largest*'

Для понижения сложности нахождения наибольшего полностью определенного живого решения такой системы предлагается следующий подход.

Теорема 2. Пусть $C \lozenge X \leq S_i$ (i = 1, 2) — система параллельных автоматных неравенств, где спецификации S_1 и S_2 определены над одинаковыми алфавитами. Наибольшее полностью определенное живое решение системы неравенств эквивалентно наибольшему живому решению неравенства $C \lozenge X \leq (S_1 \cap S_2)$.

Доказательство.

Пусть автомат *Largest* — наибольшее полностью определенное живое решение системы неравенств $C \lozenge X \leq S_i$ (i = 1, 2). А значит, он является полностью определенным живым решением каждого неравенства $C \lozenge X \leq S_i$. Поэтому для каждой последовательности α языка автомата C и каждой последовательности β языка автомата *Largest*, композиция автоматов, соответствующих последовательностям α и β, является редукцией автомата S_i, i = 1, 2.

В силу того, что язык автомата пересечения $S_1 \cap S_2$ содержит только общие последовательности языков автоматов S_1 и S_2, то для каждой последовательности α языка автомата C и каждой последовательности β языка автомата *Largest*, композиция автоматов, соответствующих последовательностям α и β, является редукцией автомата пересечения $S_1 \cap S_2$.

Таким образом, автомат *Largest* является наибольшим полностью определенным живым решением параллельного автоматного неравенства $C \lozenge X \leq (S_1 \cap S_2)$. □

Данный подход можно расширить на систему из конечного числа параллельных автоматных неравенств.

Следствие 2.1. Пусть $C \lozenge X \leq S_i$ (i = 1, 2, …, n) — система параллельных автоматных неравенств, где S_1, …, S_n определены над одинаковыми алфавитами. Наибольшее полностью определенное живое решение системы неравенств эквивалентно наибольшему полностью определенному живому решению неравенства $C \lozenge X \leq (S_1 \cap \ldots \cap S_n)$.

Заключение

В настоящей работе исследуется задача синтеза безопасного серверного приложения веб-сервиса на основе поиска полностью определенного живого решения соответствующей системы параллельных автоматных уравнений / неравенств. Использование такого решения гарантирует отсутствие зацикливаний и тупиковых ситуаций в работе веб-сервиса.

Для двух частных случаев показано, что возможно сведение поиска наибольшего полностью определенного живого решения системы параллельных автоматных уравнений / неравенств к поиску наибольшего полностью определенного живого решения одного уравнения / неравенства, что в ряде случаев позволяет понизить

сложность решения системы. Рассмотрены ситуации, когда необходимо синтезировать безопасное серверное приложение, которое может работать с разными клиентскими приложениями и предоставлять одинаковый уровень сервиса, или необходимо синтезировать безопасное серверное приложение, которое может работать с одним клиентским приложением, предоставляя разные уровни сервиса.

В дальнейшем предполагается исследовать общий случай, когда контексты и спецификации разных уравнений не совпадают, а также рассмотреть структуру параллельной композиции, когда контекст и неизвестный компонент имеют внешние входные и выходные алфавиты. Также планируется провести эксперименты с реальными веб-сервисами. Отметим, что все операции в предложенном подходе выполняются на конечных автоматах и полуавтоматах. При числе состояний рассматриваемых автоматов не более 100, эти операции могут быть эффективно выполнены с использованием программного инструмента BALM II [17].

ЛИТЕРАТУРА

1. World Wide Web Consortium // Web Services Activity URL: http://www.w3.org/2002/ws/ (дата обращения: 10.06.2023).
2. Kondratyeva O. Timed FSM strategy for optimizing web service compositions w.r.t. the quality and safety issues: Doctoral thesis, Paris Saclais. — 2015.
3. Евтушенко Н.В. Недетерминированные автоматы: анализ и синтез. Ч. 2. Решение автоматных уравнений: Учебное пособие / Н.В. Евтушенко, М.В. Рекун, С.В. Тихомирова // Томск: Томский государственный университет. — 2009. — 111 с.
4. Bochmann G.V. Using logic to solve the submodule construction problem // Discrete Event Dynamic Systems. — 2013. — P. 27–59.
5. Villa T. The Unknown Component Problem. Theory and Applications / T. Villa, N. Yevtushenko, R.K. Brayton, A. Mishchenko, A. Petrenko, A. Sangiovanni-Vincentelli // Berlin: Springer. — 2012. — 312 p.
6. K. El-Fakih, N. Yevtushenko Progressive solutions to FSM equations // URL: http://www-higashi.ist.osaka-u.ac.jp/~kelfakih/EY08-CIAA08.pdf (дата обращения: 03.06.2023).
7. Darusenkova E. Deriving a module of a multiagent system via Finite State Machine equation solving / E. Darusenkova, N. Yevtushenko, T. Villa // International Siberian Conference on Control and Communications (SIBCON–2015). Proceedings. — Omsk: The Tomsk IEEE Chapter & Student Branch. Russia, Omsk. — 2015. — DOI: 10.1109/SIBCON.2015.7147035.
8. Жарикова С.В. Метод нахождения прогрессивного решения системы автоматных уравнений // Доклады V Сибирской научной школы-семинара с международным участием «Компьютерная безопасность и криптография» — SYBECRUPT'06. — 2006. — С. 20–24.
9. Гилл А. Введение в теорию конечных автоматов / А. Гилл. — М: Издательство Наука. — 1966. — 272 с.
10. Евтушенко Н.В. Недетерминированные автоматы: анализ и синтез. Ч. 1. Отношения и операции: Учебное пособие / Н.В. Евтушенко, А.Ф. Петренко, М.В. Ветрова // Томск: Томский государственный университет. — 2006. — 142 с.
11. Aho A.V. The Theory of Parsing, Translation, and Compiling, Volume I: Parsing / A.V. Aho, J.D. Ullman // Prentice Hall. — 1972. — p. 1002.
12. Никитин А.Л. Тестирование робастности многомодульных систем на основе автоматной модели // Труды 7-й и 8-й конференций студенческого научно-исследовательского инкубатора. — Томск: Томское университетское издательство. — 2011. — С. 121-124.
13. El-Fakih K. FSM Based Interoperability Testing Methods / K. El-Fakih, V. Trenkaev, N. Spitsyna, N. Yevtushenko // Lecture notes in Computer Science 2978. — 2004. — P. 60–75.
14. Бушков В.Г. К описанию прогрессивных решений параллельного автоматного уравнения // Прикладная дискретная математика. — 2008. — № 1(1). — С. 120–125.
15. Buffalov S. Progressive solutions to a parallel automata equation / S. Buffalov, K. El-Fakih, N. Yevtushenko, G. Bochmann // Proceedings of the international conference FORTE2003, Germany. — 2003. — P. 367–382.
16. Тихомирова С.В. Оптимизация многокомпонентных дискретных систем на основе решения автоматных уравнений: Дис. канд. техн. наук. — Томск. — 2008. — 145 с.
17. Castagnetti G. Solving parallel equations with BALM-II / G. Castagnetti, M. Piccolo, T. Villa, N. Yevtushenko, A. Mishchenko, R. Brayton // Technical Report No UCB/EECS-2012-181, Electrical Engineering and Computer Sciences University of California at Berkeley. — 2012. — p. 32.

Журнал «Современная наука: актуальные проблемы теории и практики»