Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Радиофизический факультет
Кафедра информационных технологий в исследовании дискретных структур

ДОПУСТИТЬ К ПРЕДСТАВЛЕНИЮ ГЭК
Руководитель ООП
д-р физ.-мат. наук, профессор
_____ С.П. Моисеева
«10» июня 2019 г.

НАУЧНЫЙ ДОКЛАД
об основных результатах подготовленной научно – квалификационной работы (диссертации)

КОНЕЧНО-АВТОМАТНЫЕ МЕТОДЫ АНАЛИЗА И ОПТИМИЗАЦИИ МНОГОКОМПОНЕНТНЫХ УПРАВЛЯЮЩИХ СИСТЕМ

по основной образовательной программе подготовки научно-педагогических кадров в аспирантуре
направление подготовки 09.06.01 – Информатика и вычислительная техника

Широкова Екатерина Владимировна

Научный руководитель
д-р техн. наук, профессор
_____Н.В. Евтушенко
подпись
«10» июня 2019 г.

Автор работы
аспирант
_____Е.В. Широкова
подпись

Томск-2019

Общая характеристика работы

Актуальность работы. Сложные управляющие системы обычно являются многоэлементными, т.е. представляются в виде композиции взаимодействующих компонент. Соответственно, для такой композиции необходимо решать различные задачи анализа и синтеза, в частности, задачу проверки возможности совместной работы компонент композиции, и задачи оптимизации компонент и композиции в целом относительно различных критериев. Одной из возможностей оптимизации компонент является так называемая пластичность компонент (flexibility), поскольку в качестве одной и той же компоненты можно использовать различные подсистемы, и при этом внешнее поведение всей системы не изменится. Таким образом, возникает вопрос о допустимых изменениях в поведении компоненты или о соответствующем выборе компоненты в зависимости от остальных компонент и известной спецификации всей системы. Проблема известна как задача решения уравнения, создания подмодуля или задача синтеза неизвестной компоненты. Для формального решения задач анализа и синтеза необходимо определить математическую модель, используемую для описания поведения компонент и системы в целом, точность, с которой синтезированная система должна соответствовать спецификации, а также правила совместного функционирования элементов системы. Достаточно часто в качестве таких моделей выступают модели с конечным числом состояний, и в этом случае ряд задач анализа и синтеза многокомпонентных систем можно свести к решению таких задач для автоматных композиций, т.е. воспользоваться конечно-автоматными методами анализа и синтеза многокомпонентных систем. В настоящее время особый интерес представляет решение таких задач для случаев, когда поведение компонент композиции описывается не только классическими автоматами, а их расширениями, в частности, временными системами переходов, и адаптация хорошо зарекомендовавших себя конечно-автоматных методов для таких систем. Таким образом, разработка методов задач синтеза и анализа многомодульных управляющих дискретных систем на основе моделей с конечным числом переходов является актуальной проблемой.

Целью работы является исследование возможности совместной работы совокупности взаимодействующих автоматов и оптимизация компонент автоматных композиций на основе решения (полу)автоматных уравнений и систем таких уравнений.

Методы исследования. Для достижения поставленной цели в работе используется аппарат дискретной математики, в частности, методы теории автоматов.

Научная новизна.

1. Предложен подход для проверки безопасной работы параллельной композиции из временных автоматов без выполнения операции композиции.
2. Предложен подход для решения полуавтоматных уравнений относительно операции конкатенации.
3. Показана сводимость решения системы параллельных автоматных неравенств/уравнений к решению одного неравенства/уравнения для частных случаев систем, а именно при одинаковых спецификациях или при одинаковых контекстах, что позволяет понизить сложность при синтезе/оптимизации компонент многоэлементных дискретных систем.

Основные положения, выносимые на защиту.

1. Критерий безопасной работы параллельной композиции (временных) автоматов на основе проекций автоматов-компонент и алгоритм проверки безопасности

автоматной композиции, позволяющий снизить сложность проверки за счет отсутствия построения глобального автомата, описывающего поведение композиции.

2. Метод построения тестовых последовательностей для проверки поведения параллельной композиции (временных) автоматов на небезопасных входных последовательностях на основе синтеза таких последовательностей для частичных автоматов.

3. Алгоритмы решения (полу)автоматных уравнений относительно операции конкатенации, которые могут быть использованы для проверки безопасности вводимых входных данных.

4. Методы сведения системы параллельных автоматных неравенств к решению одного неравенства при одинаковых спецификациях для всех уравнений системы или при одинаковых контекстах для всех уравнений системы, что позволяет упростить оптимизацию компоненты в многокомпонентных системах определенной топологии.

Достоверность результатов. Все научные положения и выводы, содержащиеся в работе, основаны на утверждениях, доказанных с использованием аппарата дискретной математики, в том числе, с использованием аппарата классической теории автоматов.

Теоретическая и практическая ценность. Предложенные в работе методы могут быть использованы на этапе проектирования для анализа и оптимизации многокомпонентных управляющих систем, в частности, для анализа безопасности работы композиций веб-сервисов, а также для выбора оптимальных компонент веб-сервисов.

Апробация работы. Основные положения и результаты, представленные в работе, обсуждались на следующих международных конференциях:

1. IX российская конференция с международным участием «Новые информационные технологии в исследовании сложных структур» (ICAM-2012), Алтай.
2. IV международная научно-практическая конференция «Актуальные проблемы радиофизики» (АПР-2012), г. Томск.
3. V международная научно-практическая конференция «Актуальные проблемы радиофизики» (АПР-2013), г. Томск.
4. X российская конференция с международным участием «Новые информационные технологии в исследовании сложных структур» (ICAM-2014), Алтай.
5. XI Международная конференция «IEEE Сибирская конференция по управлению и связи» (SIBCON-2015), г. Омск.
6. XI российская конференция с международным участием «Новые информационные технологии в исследовании сложных структур» (ICAM-2016), г. Екатеринбург.
7. Международная конференция «IEEE East-West Design & Test Symposium» (EWDTS-2016), г. Ереван, Армения.
8. XVIII международная конференция «International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices» (EDM-2017), Эрлагол, Алтай.
9. Международная конференция «IEEE East-West Design & Test Symposium» (EWDTS-2018), г. Казань.

Публикации. По результатам работы опубликовано 9 работ, в том числе, две из них в изданиях из перечня ВАК и четыре – в изданиях, индексируемом в базе данных Scopus:

1. Дарусенкова (Широкова) Е.В., Кондратьева О.В. Тестирование робастности веб-сервисов // Тезисы докладов восьмой российской конференции с международным

участием «Новые информационные технологии в исследовании сложных структур». - Томск: Изд-во НТЛ, 2012, с. 93.

2. Дарусенкова (Широкова) Е.В., Кондратьева О.В. Тестирование робастности веб-сервисов на основе композиций конечных автоматов // Известия вузов. Физика. – 2012. - №9/2. – с. 339-340.

3. Дарусенкова (Широкова) Е.В., Кондратьева О.В. Оценка робастности как параметра качества композиции веб-сервисов // Известия вузов. Физика. – 2013. - №9/2. – с. 162-164.

4. Дарусенкова (Широкова) Е.В. Сведение решения системы параллельных автоматных неравенств к решению одного неравенства // Тезисы докладов восьмой российской конференции с международным участием «Новые информационные технологии в исследовании сложных структур». - Томск: Издательский Дом ТГУ, 2014, с. 37-38.

5. E. Darusenkova (Shirokova), N. Yevtushenko, T. Villa Deriving a module of a multiagent system via Finite State Machine equation solving // Труды XI Международной IEEE Сибирской конференции по управлению и связи SIBCON-2015, г. Омск

6. Дарусенкова (Широкова) Е.В. К анализу безопасной работы параллельной композиции частичных автоматов //Новые информационные технологии в исследовании сложных структур : материалы 11-й международной конференции, 6–10 июня 2016 г. Томск: Издательский Дом ТГУ, 2016. С. 49-50.

7. E. Darusenkova (Shirokova), and N. Shabaldina, “Towards parallel composition of partial finite state machines: checking safety property step-by-step”, Proceedings of the IEEE East-West Design & Test Symposium (EWDTS), Yerevan, Armenia, 2016, DOI Bookmark: http://doi. ieeecomputersociety.org/10.1109/EWDTS.2016.7807689.

8. Anton V. Kolomeets, Natalia V. Shabaldina, Ekaterina V. Darusenkova (Shirokova), Nina V. Yevtushenko. Using Models of Finite Transition Systems for Checking Web-Service Security //18th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2017 : proceedings Erlagol, 29 june - 3 july 2017. Novosibirsk: NSTU publisher, 2017. P. 151-154.

9. Ekaterina Shirokova "Checking Robustness of Web Services based on the Parallel Composition of Partial Timed Finite State Machines", Proceedings of IEEE East-West Desine & Test Symposium (EWDTS'2018), Kazan, PP. 256-261.

Содержание работы

Во введении обсуждаются актуальность, цель работы и научная новизна, сформулированы положения, выносимые на защиту, и представлено краткое содержание диссертации.

Первая глава содержит основные определения и обозначения, которые используются в работе.

В первой главе вводится понятие конечного автомата, конечного полуавтомата, временного конечного автомата, параллельной композиции автоматов. (Полу)автоматные модели широко используются для описания поведения систем, которые переходят из состояния в состояние под действием входных воздействий, производя при этом некоторые выходные сигналы. Инициальный конечный автомат рассматривается как преобразователь последовательностей в одном (входном) алфавите в последовательности в

другом (выходном) алфавите. Полуавтоматы часто используются для описания такого отображения как формального языка. Временной автомат позволяет при описании поведения систем учитывать временные аспекты.

Инициальным конечным автоматом или просто *конечным автоматом* называется пятёрка $S = (S, I, O, T_S, s_0)$, где S – непустое конечное множество состояний с выделенным начальным состоянием s_0, I – непустое множество входных символов, называемое входным алфавитом, O – непустое множество выходных символов, называемое выходным алфавитом, $T_S \subseteq I \times S \times S \times O$ – *отношение переходов*. Автомат S называется *полностью определенным*, если для каждой пары $(i, s) \in I \times S$ существует, по крайней мере, одна пара $(o, s') \in O \times S$ такая, что $(i, s, s', o) \in T_S$. В противном случае автомат S называется *частичным*.

Автомат $S = (S, I, O, T_S, s_0)$ называется *детерминированным*, если для любой пары $(i, s) \in I \times S$ существует не более одной пары $(o, s') \in O \times S$ такой, что $(i, s, s', o) \in T_S$, в противном случае автомат называется *недетерминированным*. В детерминированном автомате $S = (S, I, O, T_S, s_0)$ часто вместо отношения переходов используют две функции: функцию *переходов* δ_S: $I \times S \to S$ и функцию *выходов* λ_S: $I \times S \to O$.

Автомат $S = (S, I, O, T_S, s_0)$ называется *наблюдаемым*, если для любой тройки $(s, i, o) \in S \times I \times O$ существует не более одного состояния $s' \in S$ такого, что $(s, i, o, s') \in T_S$.

Описанные выше инициальные автоматы описывают поведение систем с известным начальным состоянием, и такие автоматы часто используются для описания поведения дискретных систем, которые преобразуют последовательности в одном (входном) алфавите I в последовательности в другом (выходном) алфавите O. Отношение переходов распространяется на входные и выходные последовательности естественным образом. Пусть $\alpha = i_1 \ldots i_k$ и $\beta = o_1 \ldots o_k$ – входная и выходная последовательности, соответственно. Четверка $(i_1 \ldots i_k, s, s', o_1 \ldots o_k) \in T_S$, если существует последовательность состояний $s_1, \ldots, s_{k+1}$ такая, что $s_1 = s$ и $(i_j, s_j, s_{j+1}, o_j) \in T_S$ для всех $j = 1, \ldots, k$. В этом случае пара α/β называется *входо-выходной последовательностью* автомата S. Функции переходов и выходов в детерминированном автомате распространяются на последовательности аналогичным образом. В наблюдаемом автомате по любой входо-выходной последовательности из любого состояния достижимо не более одного состояния.

В автомате определяются функции $next\_states_S(i_1 \ldots i_k, s)$ и $out_S(i_1 \ldots i_k, s)$, которые называются *функциями переходов и выходов*. Состояние $s' \in next\_states_S(i_1 \ldots i_k, s)$, если существует четверка $(i_1 \ldots i_k, s, s', o_1 \ldots o_k) \in T_S$; выходная последовательность $(o_1 \ldots o_k) \in out_S(i_1 \ldots i_k, s)$, если существует четверка $(i_1 \ldots i_k, s, s', o_1 \ldots o_k) \in T_S$. В недетерминированном автомате значением функции выходов out_S может быть множество выходных последовательностей. Иными словами, в состоянии s недетерминированного полностью определенного автомата $S = (S, I, O, T_S, s_0)$ функция выходов out_S отображает множество I^* входных последовательностей во множество подмножеств выходных последовательностей. Функция выходов в начальном состоянии называется *функцией выходов автомата* и обозначается out_S. В детерминированном автомате функции переходов и выходов однозначно определяют поведение автомата; поэтому инициальный детерминированный полностью определенный автомат часто определяется как шестерка

$\mathsf{A} = (A, I, O, \delta_A, \lambda_A, a_0)$, где через δ_A: $I \times S \to S$ и λ_A: $I \times S \to O$ обозначаются функции переходов и выходов автомата.

Языком автомата A в состоянии a называется множество входо-выходных последовательностей автомата в этом состоянии, т.е. множество последовательностей входо-выходных пар в алфавите $I \times O$, получаемых при последовательных переходах из состояния a. Формально, язык $L^{\times}_A(a)$ есть подмножество $(I \times O)^*$, и последовательность $(i_1,o_1) \ldots (i_k,o_k) \in L^{\times}_A(a)$, если и только если $o_1 \ldots o_k \in out(a, i_1 \ldots i_k)$. По определению, язык $L^{\times}_A(a)$ содержит пару (ε, ε), где ε – пустая последовательность. Язык $L^{\times}_A(a_0)$ автомата A в начальном состоянии a_0 называется языком автомата и обозначается $L^{\times}_A$.

Если автомат A является частичным, то для некоторых входных последовательностей $i_1 \ldots i_k$ поведение автомата A в состоянии a может быть неопределённым: если $out(a, i_1 \ldots i_k) = \varnothing$, то говорят, что поведение автомата A в состоянии a не определено на входной последовательности $i_1 \ldots i_k$.

Вообще говоря, язык автомата описывает отображение, реализуемое этим автоматом. Последовательность $(i_1, o_1) \ldots (i_k, o_k)$ принадлежит языку $L^{\times}_A(a)$, если и только если поведение автомата определено на последовательности $i_1 \ldots i_k$ в состоянии a, и автомат A в этом состоянии может выдать выходную последовательность $o_1 \ldots o_k$ на входную последовательность $i_1 \ldots i_k$.

Если необходимо подчеркнуть, что в автомате каждый выходной символ появляется как реакция на некоторый входной символ, то язык автомата удобно представлять в виде последовательностей входных и выходных символов. Такой язык $L^{\cup}_A(a)$ есть подмножество $(I \cup O)^*$, и последовательность $i_1o_1 \ldots i_ko_k$ принадлежит $L^{\cup}_A(a)$, если и только если $(i_1, o_1) \ldots (i_k, o_k)$ принадлежит $L^{\times}_A(a)$. По определению, язык $L^{\cup}_A(a)$ содержит пустую последовательность ε.

Автомат $\mathsf{MAX}(I, O) = (\{t_0\}, I, O, T, t_0)$, где $T = \{t_0\} \times I \times O \times \{t_0\}$, называется *максимальным* автоматом над входным алфавитом I и выходным алфавитом O. Автомат $\mathsf{MAX}(I, O)$ имеет язык $(IO)^*$.

Состояние b недетерминированного автомата $\mathsf{B} = (B, I, O, T_B, b_0)$ называется *редукцией* состояния a недетерминированного автомата $\mathsf{A} = (A, I, O, T_A, a_0)$ (обозначение $b \le a$), если $L^{\times}_A(b) \subseteq L^{\times}_A(a)$.

Пусть $\mathsf{A} = (A, I, O, \delta_A, \lambda_A, a_0)$ и $\mathsf{B} = (B, I, O, \delta_B, \lambda_B, b_0)$ – полностью определенные автоматы. Состояние a автомата A называется *эквивалентным* состоянию b автомата B, если $\forall\, \alpha \in I^*$ $(\lambda_A(a, \alpha) = \lambda_B(b, \alpha))$. Автомат, который не содержит эквивалентных состояний, называется *приведенным*. Инициальные автоматы A и B называются *эквивалентными*, если эквивалентны их начальные состояния. Иногда говорят, что эквивалентные автоматы имеют одинаковое *поведение*.

Детерминированный автомат $\mathsf{B} = (B, I, O, \delta_B, \lambda_B, b_0)$ называется *подавтоматом* автомата $\mathsf{S} = (S, I, O, T_S, s_0)$, если $B \subseteq S$, $b_0 = s_0$ и для любой пары $(b, i) \in B \times I$ справедливо $[b, i, \delta_B(b,i), \lambda_B(b,i)] \in T_S$. Детерминированный подавтомат недетерминированного автомата может быть получен при детерминированной фиксации каждого перехода.

Совместное поведение двух автоматов может быть описано посредством пересечения данных автоматов. *Пересечением* $\mathsf{A} \cap \mathsf{B}$ автоматов A и B называется наибольший подавтомат автомата $(A \times B, I, O, T_{A\cap B}, a_0b_0)$, где $T_{A\cap B} = \{(ab, i, o, a'b') \mid (a, i, o, a') \in T_A \ \wedge \ (b, i, o, b') \in T_B\}$. Языком $\mathsf{A} \cap \mathsf{B}$ является

пересечение $L_A \cap L_B$. Пересечение двух наблюдаемых автоматов есть наблюдаемый автомат. Тем не менее, пересечение полностью определенных автоматов может быть частичным автоматом.

Конечным полуавтоматом[1] называется пятёрка $P = (P, J, T_P, p_0, F_P)$, где P – непустое конечное множество состояний с выделенным начальным состоянием p_0 и подмножеством F_P финальных состояний. J – алфавит, $T_P \subseteq J \times P \times P$ – отношение переходов.

Полуавтомат $P = (P, J, T_P, p_0, F_P)$ называется *детерминированным*, если для любой пары $(p, j) \in P \times J$ существует не более одного состояния $p' \in P$ такого, что $(p, j, p') \in T_P$, в противном случае полуавтомат называется *недетерминированным*.

Языком полуавтомата P в состоянии p называется множество последовательностей, которые переводят полуавтомат из состояния p в одно из финальных состояний. По определению, пустое слово принадлежит языку полуавтомата в состоянии p, если и только если p принадлежит множеству финальных состояний. Два полуавтомата *эквивалентны*, если их языки совпадают. Известно, что для любого полуавтомата $P = (P, J, T_P, p_0, F_P)$ можно построить эквивалентный ему детерминированный полуавтомат $D_P = (D, J, T_D, d_0, F_D)$.

Для представления языка $L^{\cup}{}_S(s_0)$ автомата $S = (S, I, O, T_S, s_0)$ можно построить полуавтомат $P^{\cup}{}_S = (S \cup (S \times I), I \cup O, T_{PS}, s_0, S)$. Множество состояний полуавтомата $P^{\cup}{}_S$ есть объединение множеств $S \cup (S \times I)$; финальными являются состояния из множества S. Для каждого состояния $s \in S$ и входного символа $i \in I$ множество T_{PS} содержит переход $(s, i, (s', i))$, если и только если $\exists\, o \in O$ $((s, i, o, s') \in T_S)$. Для каждого состояния $(s, i) \in S \times I$ и выходного символа $o \in O$ множество T_{PS} содержит переход $((s, i), o, s')$, если и только если $(s, i, o, s') \in T_S$. Язык полуавтомата $P^{\cup}{}_S$ совпадает с языком $L^{\cup}{}_S$ автомата S.

Полуавтомат $P^{\cup}{}_S$ детерминированный, если и только если автомат S наблюдаемый.

Автоматы A и B эквивалентны, если и только если эквивалентны полуавтоматы $P^{\cup}{}_A$ и $P^{\cup}{}_B$.

Пусть $\alpha \in V^*$ – некоторая последовательность и W – некоторый алфавит, *W-проекция* последовательности α (обозначается $\alpha_{\downarrow W}$) получается путем удаления из α всех символов, содержащихся во множестве $V \setminus W$. Пусть $\alpha \in V^*$ – некоторая последовательность и W – некоторый алфавит, *W-расширение* последовательности α (обозначается $\alpha_{\uparrow W}$) есть множество, содержащее каждую последовательность над алфавитом $(V \cup W)$ с V-проекцией последовательности α. Все операторы расширены до операторов над конечными автоматами. Пусть P – полуавтомат над алфавитом V с языком L. *Проекция*($\downarrow$): Пусть U – непустое подмножество множества V, полуавтомат $P_{\downarrow U}$, который принимает язык $L_{\downarrow U}$ над U, получается путем замены в P каждого перехода (s, a, s'), $a \in V \setminus U$, переходом (s, ε, s'). *Расширение*($\uparrow$): Пусть U – некоторый алфавит, полуавтомат $P_{\uparrow U}$ с языком $L_{\uparrow U}$ над $U \cup V$ получается путем добавления перехода (s, a, s) для каждого $a \in U \setminus V$ и каждого состояния s полуавтомата P.

---

1 Рассматриваются только полуавтоматы без ненаблюдаемого действия. В работах авторов Хопкрофта и Ульмана показано, как такое действие можно удалить из полуавтомата без изменения языка полуавтомата.

Максимальный полуавтомат $MAX(J)$ над алфавитом J представляет собой детерминированный полуавтомат $(\{c\}, J, T, c, \{c\})$, где $T = \{t_0\} \times J \times \{t_0\}$. Полуавтомат $MAX(J)$ имеет язык $(J)^*$.

Пусть N – множество натуральных чисел. *Временным автоматом* в диссертационной работе называется конечный автомат с таймаутами, т.е. семёрка $S = (S, I, O, s_0, T_S, \Delta_S, \sigma_S)$, в котором S – непустое конечное множество состояний с выделенным начальным состоянием s_0, I – входной алфавит, O – выходной алфавит, $T_S \subseteq I \times S \times S \times O$ – *отношение переходов*; Δ_S: $S \to S \times (N \cup \{\infty\})$ – функция задержки входного символа, которая для каждого состояния определяет максимальное время ожидания входного символа (входной таймаут); σ_S: $T_S \to (\{\infty\} \cup N)$ – функция задержки выходного символа, которая для каждого перехода определяет необходимый интервал времени для выполнения соответствующего перехода и выработки выходного символа в ответ на поступившее входное воздействие (выходной таймаут), т.е. пятёрка (S, I, O, s_0, T_S) является классическим конечным автоматом.

Для формализации процесса, заключающегося в представлении дискретной системы в виде композиции других, более простых в некотором смысле подсистем, необходимо определить операцию композиции.

Предполагается, что поведение всех подсистем и системы в целом описывается конечными автоматами, и подсистемы взаимодействуют, обмениваясь словами в их языках.

Асинхронное функционирование многокомпонентной системы предполагает отсутствие глобального синхросигнала, позволяющего всем компонентам системы по этому сигналу переключиться из текущего состояния в следующее и произвести выходной символ. В данной работе рассматривается только один из способов асинхронного функционирования, когда компоненты взаимодействуют в режиме диалога, т.е. в каждый момент времени активной является только одна компонента. Такое функционирование можно описать с использованием операции параллельной композиции автоматов-компонент при так называемой «медленной» внешней среде. При подаче внешнего входного сигнала на одну из компонент эта компонента вырабатывает внешний или внутренний выходной сигнал. Если компонента вырабатывает внутренний сигнал, то компоненты работают в режиме диалога, пока компоненты не перестанут обмениваться внутренними сигналами. После этого композиция готова обработать следующий внешний входной сигнал.

Рассмотрим композицию автоматов на рисунке 1.1, в которой

- автомат A имеет входной алфавит $I_1 \cup V$, выходной алфавит $O_1 \cup U$ и отношение переходов T_A;

- автомат B имеет входной алфавит $I_2 \cup U$, выходной алфавит $O_2 \cup V$ и отношение переходов T_B;

- алфавиты I_1, I_2, O_1, O_2, V и U попарно не пересекаются.

Параллельная композиция автоматов A и B имеет

- входной алфавит $I = I_1 \cup I_2$;

- выходной алфавит $O = O_1 \cup O_2$;

- под действием входного символа $i \in I_1 \cup I_2$ композиция «вырабатывает» внешний выходной символ $o \in O_1 \cup O_2$ или внутренний выходной символ, который является входным для другой компоненты;

- следующий входной символ может быть подан на композицию только после того, как компоненты закончили внутренний диалог; если компоненты суть детерминированные полностью определенные автоматы, то это означает, что композиция отреагировала внешним выходным сигналом на предыдущий внешний входной сигнал.

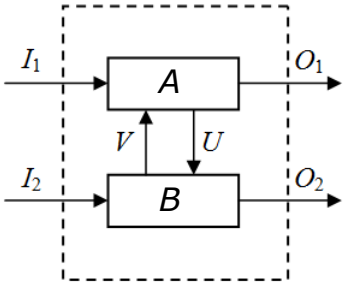


Рисунок 1.1 – Бинарная композиция автоматов

Параллельная композиция полностью определенных детерминированных автоматов A и B может оказаться частичным автоматом. Причиной является тот факт, что после подачи некоторой входной последовательности компоненты начинают вести «бесконечный» диалог, не выдавая никаких внешних сигналов.

Таким образом, *параллельной композицией* конечных автоматов A и B называется приведенный наблюдаемый автомат $A \lozenge B$ с языком $(L^{\cup}_A \lozenge L^{\cup}_B) \cap (IO^*)$, где $L^{\cup}_A \lozenge L^{\cup}_B = [(L^{\cup}_A)_{\uparrow I2 \cup O2} \cap (L^{\cup}_B)_{\uparrow I1 \cup O1}]_{\downarrow I1 \cup I2 \cup O1 \cup O2}$.

В данной главе приводится пример использования автоматных моделей и их композиций для описания поведения веб-сервисов. Функционирование сложного веб-сервиса можно описать с использованием операции параллельной композиции автоматов-компонент.

Во второй главе рассматривается проверка безопасного взаимодействия компонент сложных систем.

В разделе 2.1 предлагается метод проверки безопасной работы композиции автоматов, где одна из компонент является встроенной, т.е. не имеет внешних входного и выходного алфавита. Структура такой композиции приведена на рисунке 2.1. Такие композиции активно используются при анализе клиент-серверных приложений. При проверке безопасного функционирования клиент-серверного приложения предполагается, что для определенных входных действий реализации серверного и клиентского приложений соответствуют их спецификациям. В общем случае это означает, что каждая компонента композиции ведет себя адекватно на последовательностях входных действий, на которых поведение компоненты определено. Поскольку в общем случае сервисная компонента может иметь частичную спецификацию, то поведение компоненты для некоторых входных последовательностей не определено. Для неопределенных входных последовательностей реализация может иметь любое поведение, которое может оказаться опасным для функционирования компоненты или для всей композиции, если такая последовательность будет подана на реализацию компоненты. Поэтому необходимо осуществлять проверку безопасного поведения и/или взаимодействия компонент для последовательностей, которые не определены в их спецификациях.

При композиции компонент, поведение которых определено не на всех входных последовательностях, композиция называется *безопасной*, если поведение каждой компоненты определено на всех последовательностях, которые могут поступить на эту компоненту при совместной работе с другими компонентами. Если композиция не является безопасной, то необходимо проверить поведение реализации системы на недопустимых входных последовательностях, т.е. убедиться в робастности компоненты.

В данной главе предлагается проверка безопасной работы на основе проекций компонент на соответствующие алфавиты. Если проекция композиции компонент, с которых подаются входные воздействия на проверяемую компоненту, содержится во множестве допустимых входных последовательностей компоненты, то композиция называется *безопасной*.

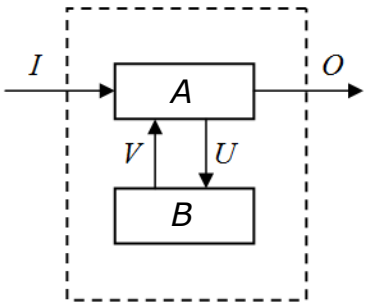


Рисунок 2.1 – Параллельная композиция автоматов, в которой *B* - встроенная компонента

Теорема 2.1. Последовательность α в алфавите U может поступить на компоненту B, если и только если в автомате A существует входо-выходная последовательность γ с U – проекцией α.

Если при проверке безопасности работы клиент-серверного приложения предполагается, что компоненты описываются (временными) частичными автоматами, то необходимо исследовать, какие (временные) входные последовательности безопасны, т.е. не приводят к тупиковым ситуациям (например, к зацикливаниям), т.е. к ситуациям, когда композиция не может выдать внешний выходной сигнал. Другими словами, для безопасной работы композиции поведение каждой компоненты композиции должно быть определено на всех временных последовательностях, которые могут подаваться на эту компоненту при совместной работе с другими компонентами.

Для структуры композиции, представленной на рисунке 2.1, на компоненту B могут поступить только соответствующие проекции тех последовательностей, которые содержатся в языке компоненты A.

На основе теоремы 2.1, предложен алгоритм проверки безопасности работы компонент автоматной композиции на основе проекций автоматных компонент, который включает следующие шаги.

Алгоритм 2.1 Проверка безопасной работы композиции автоматов на основе проекций автоматных компонент.

Вход: Параллельная композиция автоматов A и B.

Выход: Заключение о безопасной или небезопасной работе приложений, реализованных компонентами A и B.

Шаг 1. По автоматам A и B строим соответствующие полуавтоматы P_A и P_B.

Шаг 2. Находим проекцию полуавтомата P_A на алфавит U. Проекция полуавтомата P_A на алфавит U получается посредством «стирания» символа из I на каждом переходе с последующей детерминизацией полученного полуавтомата.

Шаг 3. Проверяем, содержится ли полученная проекция во множестве допустимых входных последовательностей автомата B. Если так, то приложения, реализующие компоненты A и B, могут работать безопасно без возникновения тупиков.

В случае, когда при взаимодействии компонент композиции возникают тупиковые ситуации, можно «ограничить» компоненту A. Это приведет к тому, что компонента A не будет производить внутренние последовательности, которые являются недопустимыми для компоненты B. Такое ограничение может быть получено с помощью следующего алгоритма.

Алгоритм 2.2 «Ограничение» компоненты A для случая, когда композиция автоматов небезопасна.

Вход: Небезопасная параллельная композиция автоматов A и B.

Выход: Безопасная параллельная композиция автоматов A и B, где компонента A не вырабатывает последовательности, которые заводят композицию в тупиковое состояние.

Шаг 1. Для того чтобы определить множество недопустимых входных последовательностей полуавтомата P_B необходимо найти разность множества последовательностей полуавтомата P_A, ограниченного на алфавит U, и множества входных последовательностей полуавтомата P_B.

Шаг 2. Расширяем каждую последовательность полученного множества до алфавита $I \cup O \cup V$. Таким образом, получим множество входо-выходных последовательностей компоненты A, которые заводят в тупик.

Шаг 3. Удаляем полученные последовательности из языка компоненты A.

Таким образом, компонента A становится «более частичной» и не производит внутренние последовательности, которые являются недопустимыми для компоненты B. В главе 3 обсуждается оптимизация компонент относительно робастности, и алгоритм 2.2 и позволяет сделать компоненты «более робастными» за счет ограничений поведения компоненты или выбора другой реализации компоненты.

Для случая, когда композиция, описывающая взаимодействие компонент некоторой системы, не является безопасной, в данной главе обсуждается задача выбора реализаций встроенной компоненты системы. Рассмотрим данную задачу на примере веб-сервисов. Для каждого веб-сервиса существует множество клиентских и серверных приложений, удовлетворяющих спецификации. В этом случае, если композиция автоматов оказалась небезопасной (присутствуют тупиковые ситуации), то для каждого клиентского приложения необходимо найти реализации сервера, с которыми данное приложение может работать совместно. Для решения этой проблемы необходимо выполнить следующие шаги для каждой реализации клиента. Пусть компоненты A и B описывают поведение клиентского и серверного приложений, соответственно.

Алгоритм 2.3 Выбор реализаций встроенной компоненты композиции, которые могут безопасно взаимодействовать с заданной реализацией компоненты A.

Вход: Небезопасная параллельная композиция автоматов A и B.

Выход: Список реализаций компоненты B (список реализаций сервера), с которыми может работать совместно реализация компоненты A (клиентское приложение).

Шаг 1. Построить множество тестовых последовательностей R внешних входных символов, покрывающих все неопределенные в автомате A переходы и позволяющих определить состояния, в которых остановятся автоматы A и B после завершения недопустимого перехода.

Шаг 2. Для каждой последовательности $r \in R$, если r приводит к вызову недопустимых в автомате B переходов, добавить r к множеству T.

Шаг 3. Если T = ∅, то рассматриваемая реализация компоненты A может работать совместно с любой реализацией компоненты B, удовлетворяющей спецификации, в противном случае Шаг 4.

Шаг 4. Для каждой реализации B и каждой последовательности $r \in T$ проверить, обрабатывает ли реализация B корректным образом недопустимые спецификацией переходы.

Шаг 5. На основании анализа выходных последовательней, полученных на шаге 4, для каждой реализации A из списка возможных реализаций B (серверных приложений) оставляются только те, с которыми реализация A (клиентское приложение) может работать совместно.

В данной работе в качестве корректного поведения на недопустимых спецификацией переходах рассматривается только формирование сообщения об ошибке и либо сохранение текущего состояния и ожидание корректного входа, либо переход в начальное состояние. Однако в зависимости от приложения, корректное поведение может определяться различными образами.

В разделе 2.2 рассматривается бинарная композиция временных автоматов и показывается, каким образом полученные в предыдущем разделе результаты могут быть расширены на случай композиции временных автоматов.

В работах авторов О.В. Кондратьевой, Н.В.Евтушенко и А. Кавалли предлагается подход, позволяющий построить для заданного временного конечного автомата соответствующий классический полуавтомат. Для моделирования выходных задержек и таймаутов во входной и выходной алфавиты добавляется новый символ $T = 1$.

В этих же работах предлагаются следующие шаги для построения параллельной композиции двух временных автоматов. Во-первых, для каждого временного автомата необходимо получить соответствующий полуавтомат. Кроме того, необходимо расширить алфавиты каждой компоненты до алфавитов другой компоненты и пересечь полученные полуавтоматы. Таким образом, получается так называемый глобальный полуавтомат, который описывает совместное поведение взаимодействующих компонент. На следующем шаге, чтобы получить параллельную композицию, необходимо ограничить глобальный полуавтомат на внешние алфавиты. Для возвращения к модели временного автомата должно быть выполнено следующее условие: язык L полученного полуавтомата должен содержаться в $(I1^*O)^*$. Если это условие не выполняется, то необходимо пересечь полученный полуавтомат с полуавтоматом, соответствующим языку $(I1^*O)^*$, а затем вернуться к модели временного автомата. Необходимо отметить, что в работах О.В. Кондратьевой, Н.В. Евтушенко и А. Кавалли рассматривается модель временного автомата, в которой значения функций входных таймаутов и задержек выходного символа могут принимать только целочисленные значения. В работе О. Кондратьевой показывается, что если два временных автомата имеют одинаковое поведение на всех временных последовательностях с целочисленным значением временной переменной, то

эти автоматы имеют одинаковое поведение и на всех временных последовательностях с рациональными значениями временной переменной, однако последнее свойство не выполняется для композиции временных автоматов.

В работе авторов А.С. Твардовского и А.В. Лапутенко отмечается, что в общем случае поведение композиции детерминированных временных автоматов не может быть описано детерминированным временным автоматом, если в качестве входных временных последовательностей допустимы последовательности с рациональными значениями временной переменной. Авторы вводят класс систем, для которых такое описание возможно. Параллельная композиция временных автоматов A и B может быть описана детерминированным временным автоматом, если в компоненте A нет перехода (s, i, s', o) $\in \lambda_S$ (нет перехода (p, i, p', o) $\in \lambda_p$ в компоненте B), где i и o являются внешними входными и выходными символами, и в построенной параллельной композиции в каждом состоянии входной таймаут больше, чем выходные задержки для обработки каждого входного символа.

Таким образом, алгоритмы 2.1, 2.2, 2.3 из раздела 2.1 могут быть расширены на случай композиции временных автоматов.

Алгоритм 2.4 Проверка безопасной работы композиции временных автоматов на основе проекций автоматных компонент.

Вход: Параллельная композиция временных автоматов A и B.

Выход: Заключение о безопасной или небезопасной работе приложений, реализованных компонентами A и B.

Шаг 1. По временным автоматам A и B строим соответствующие полуавтоматы P_A и P_B.

Шаг 2. Находим проекцию полуавтомата P_A на алфавит $U \cup T$. Проекция полуавтомата P_A на алфавит $U \cup T$ получается посредством «стирания» символа из I на каждом переходе с последующей детерминизацией полученного полуавтомата.

Шаг 3. Проверяем, содержится ли полученная проекция во множестве допустимых входных временных последовательностей автомата B. Если так, то приложения, реализующие компоненты A и B, могут работать безопасно без возникновения тупиков.

Алгоритм 2.5 «Ограничение» компоненты A для случая, когда композиция временных автоматов небезопасна.

Вход: Небезопасная параллельная композиция временных автоматов A и B.

Выход: Безопасная параллельная композиция временных автоматов A и B, где компонента A не вырабатывает временные последовательности, которые заводят в тупик.

Шаг 1. Для того чтобы определить множество недопустимых входных временных последовательностей полуавтомата P_B необходимо найти разность множества временных последовательностей полуавтомата P_A, ограниченного на алфавит $U \cup T$, и множества входных временных последовательностей полуавтомата P_B.

Шаг 2. Расширяем каждую временную последовательность полученного множества до алфавита $I \cup O \cup V$. Таким образом, получим множество временных входо-выходных последовательностей компоненты A, которые заводят в тупик.

Шаг 3. Удаляем полученные временные последовательности из языка компоненты A.

Таким образом, компонента A становится «более частичной» и не производит внутренние временные последовательности, которые являются недопустимыми для компоненты B.

Алгоритм 2.6 Выбор реализаций встроенной компоненты композиции, которые могут безопасно взаимодействовать с заданной реализацией компоненты A.

Вход: Небезопасная параллельная композиция временных автоматов A и B.

Выход: Список реализаций компоненты B, с которыми может работать совместно реализация компоненты A.

Шаг 1. Построить множество временных тестовых последовательностей R внешних входных символов, покрывающих все неопределенные в автомате A переходы и позволяющих определить состояния, в которых остановятся временные автоматы A и B после завершения недопустимого перехода.

Шаг 2. Для каждой временной последовательности $r \in R$, если r приводит к вызову недопустимых в автомате B переходов, добавить r к множеству T.

Шаг 3. Если $T = \varnothing$, то рассматриваемая реализация компоненты A может работать совместно с любой реализацией компоненты B, удовлетворяющей спецификации, в противном случае Шаг 4.

Шаг 4. Для каждой реализации B и каждой временной последовательности $r \in T$ проверить, обрабатывает ли реализация B корректным образом недопустимые спецификацией переходы.

Шаг 5. На основании анализа выходных последовательностей, полученных на шаге 4, для каждой реализации A формируется список реализаций B, с которыми реализация A может работать совместно.

В разделе 2.3 рассматривается возможность анализа безопасности веб-приложений на основе конкатенации двух полуавтоматов и, при необходимости, «очистка» пользовательских данных на основе решения соответствующих полуавтоматных уравнений.

В работах авторов Ю, Алкалафа, Булгана рассматривается другая композиция систем с конечным числом состояний, а именно полуавтоматов, для проверки возможности атак на веб-сервис. Сначала строится полуавтомат, соответствующий регулярным выражениям шаблонов атак. На следующем этапе строится граф зависимостей на основе анализируемой PHP-программы. Для каждого узла графа строится соответствующий полуавтомат; полуавтомат для всего графа строится с помощью операций объединения, конкатенации и т.д. над построенными базовыми полуавтоматами. Далее необходимо пересечь этот полуавтомат с полуавтоматом, который представляет все возможные шаблоны атак. Если пересечение пусто, то код не имеет проверяемых уязвимостей. Если пересечение не пусто, то атака может произойти, и необходим анализ при обратном проходе по графу зависимостей, чтобы определить множество входных последовательностей, которые могут привести к атакам. При обратном анализе авторы предлагают использовать функции *PreConcatPrefix* и *PreConcatSuffix*. Фактически, использование этих функций соответствует решению полуавтоматных уравнений а) $A.X \approx S$, б) $X.B \approx S$, в) $X.Y \approx S$. В этих уравнениях S соответствует указанному выше непустому пересечению, т.е. S – это полуавтомат, который описывает набор небезопасных действий веб-сервиса, а решение соответствует возможным вредоносным

пользовательским данным. Соответственно, в случаях а) и б) для того чтобы описать такие вредоносные пользовательские входные данные необходимо решить уравнение. Полученное решение может быть использовано для очистки входных данных от вредоносного содержимого, которое является небезопасным для рассматриваемой системы. Случай с) более сложный, так как и префикс, и суффикс могут зависеть от пользовательских данных.

Конкатенацией полуавтоматов $A = (A, V, T_A, a_0, F_A)$ и $B = (B, V, T_B, b_0, F_B)$ называется такой полуавтомат $A.B$, язык которого содержит все строки αβ, где $\alpha \in L_A$ и $\beta \in L_B$. В этом случае, полуавтомат A является префиксом полуавтомата $A.B$, а полуавтомат B – суффиксом полуавтомата $A.B$. Формально, $A.B = (A \cup B, V, T_S, a_0, F_S)$, где $F_S = F_B$, если $b_0 \notin F_B$ либо $F_S = F_A \cup F_B$, если $b_0 \in F_B$. Отношение переходов T_S определяется следующим образом:

1. Для всех $s \in A \setminus F_A$, т.е., для каждого перехода полуавтомата A из нефинального состояния s под воздействием символа v необходимо добавить переход полуавтомата A из состояния s под воздействием символа v в полуавтомат $A.B$.

2. Для всех $s \in F_A$, т.е., для каждого перехода полуавтомата A из финального состояния s под воздействием символа v и для перехода полуавтомата B из начального состояния b_0 под воздействием символа v необходимо добавить переходы в полуавтомат $A.B$: переход полуавтомата A из состояния s под воздействием символа v и переход полуавтомата B из состояния b_0 под воздействием символа v.

3. Для всех $b \in B$, т.е., для каждого перехода полуавтомата B из состояния b под воздействием символа v необходимо добавить переход из состояния b под воздействием символа v в полуавтомат $A.B$.

Таким образом, сначала полуавтомат $A.B$ моделирует поведение полуавтомата A. Когда $A.B$ достигает финальное состояние полуавтомата A, $A.B$ начинает моделировать поведение полуавтомата B и данное финальное состояние полуавтомата A становится начальным состоянием полуавтомата B. Если начальное состояние B является финальным, то финальное состояние A есть финальное состояние конкатенации. Также необходимо отметить, что даже в случае, когда полуавтоматы A и B являются детерминированными, конкатенация $A.B$ может быть недетерминированным полуавтоматом согласно второму правилу построения конкатенации.

Задача нахождения решения уравнений над операцией конкатенации рассматривалась в работах Л. Кари, Г. Тьеррина и П. Линца. Авторы Л. Кари, Г. Тьеррин утверждают, что уравнение $LY = R$ (где Y – неизвестная компонента) имеет единственное наибольшее решение, которое содержит все другие решения уравнения (если уравнение разрешимо). Авторы предлагают формулу для нахождения наибольшего решения и применяют хорошо известные операции над полуавтоматами. В работе П. Линца описана операция левого частного, которую можно использовать для нахождения наибольшего решения. Таким образом, на основе данных работ можно получить следующий алгоритм для нахождения наибольшего решения полуавтоматного уравнения над операцией конкатенацией.

Агоритм 2.7. Нахождение наибольшего решения уравнения $LY = R$.

Вход: Полуавтоматы L и R.

Выход: Наибольшее решение уравнения $LY = R$ (если уравнение имеет решение).

Шаг 1. Необходимо построить дополнение R^c для R: финальные и нефинальные состояния меняем местами, все неопределенные переходы доопределяем переходом в новое финальное состояние ‘Don’t care’ (DNC). Состояние DNC имеет петлю, помеченную всеми действиями.

Шаг 2. Строим левое частное $L\backslash R^c$. Для этого необходимо найти все строки в R^c, которые имеют префикс L и удалить этот префикс из каждой такой строки. Все полученные строки принадлежат левому частному $L\backslash R^c$. Если не удалось найти строк в R^c, которые имеют префикс L, значит, уравнение не имеет решений, Конец.

Шаг 3. Строим дополнение $(L\backslash R^c)^c$.

Аналогично можно построить алгоритм для нахождения решения уравнения вида $YL = R$.

В настоящей работе для нахождения наибольшего решения полуавтоматных уравнений $A.X \approx S$ и $X.B \approx S$ предлагаются следующие алгоритмы.

Алгоритм 2.8. Нахождение наибольшего решения полуавтоматного уравнения $A.X \approx S$.

Вход. Полуавтоматы $A = (A, V, T_A, a_0, F_A)$ и $S = (S, V, T_S, s_0, F_S)$.

Выход. Наибольшее решение уравнения $A.X \approx S$, если уравнение разрешимо или сообщение ‘уравнение неразрешимо’.

Шаг 1. Для полуавтомата S с начальным состоянием s_0 необходимо проверить есть ли подавтомат A' с начальным состоянием s_0, который эквивалентен A.

Шаг 2. Если такой полуавтомат A' существует, то необходимо добавить в наибольшее решение $Largest$ каждый подавтомат полуавтомата S с начальным состоянием a, где a есть финальное состояние полуавтомата A'.

Шаг 3. Построить конкатенацию полуавтоматов A и $Largest$. Если конкатенация эквивалентна S, то $Largest$ – наибольшее решение уравнения $A.X \approx S$; иначе, уравнение неразрешимо.

Алгоритм 2.9. Нахождение наибольшего решения полуавтоматного уравнения $X.B \approx S$.

Вход. Полуавтоматы $B = (B, V, T_B, b_0, F_B)$ и $S = (S, V, T_S, s_0, F_S)$.

Выход. Наибольшее решение уравнения $X.B \approx S$, если уравнение разрешимо или сообщение ‘уравнение неразрешимо’.

Шаг 1. Необходимо построить конкатенацию полуавтоматов $MAX(V)$ и B. Если полуавтомат $MAX(V).B$ недетерминированный, построить эквивалентный детерминированный полуавтомат $Det(MAX(V).B)$. Язык полуавтомата $Det(MAX(V).B)$ содержит каждую последовательность $\alpha\beta$, где $\beta \in L_B$ и $\alpha \in L_{MAX(V)}$.

Шаг 2. Пересекаем полуавтоматы $Det(MAX(V).B)$ и S. Язык полуавтомата $D = Det(MAX(V).B) \cap S$ содержит все последовательности $\alpha\beta$, где $\beta \in L_B$ и $\alpha \in L_{MAX(V)}$, такие, что последовательность $\alpha\beta$ принадлежит языку полуавтомата S. Если язык полуавтомата D пустой, то решения нет, Конец.

Шаг 3. Для каждого состояния d полуавтомата D нужно проверить существует ли подавтомат с начальным состоянием d, который эквивалентен B. Добавляем в наибольшее решение $Largest$ каждую последовательность, которая ведёт из начального состояния D до состояния d. Фактически, полуавтомат с начальным состоянием и единственным финальным состоянием d добавляется в наибольшее решение.

Шаг 4. Строим конкатенацию $Largest$ и B. Если конкатенация эквивалентна S, то $Largest$ – наибольшее решение уравнения $X.B \approx S$; иначе, уравнение неразрешимо.

Алгоритмы 2.8 и 2.9 «проще», чем алгоритмы, полученные из формулы нахождения наибольшего решения, представленной в работах авторов Л. Кари, Г. Тьеррина, т.к., в алгоритмах 2.8 и 2.9. не требуется находить дополнения полуавтоматов.

Отметим, что результаты, полученные в главе 2, могут быть использованы для оптимизации веб-сервисов с точки зрения повышения качества, а именно повышения таких параметров качества как робастность и безопасность. Данная задача рассмотрена в разделе 3.2.

В третьей главе рассматривается оптимизация компонент дискретных систем.

В разделе 3.1 рассматривается оптимизация компонент дискретных систем на основе решения (полу)автоматных неравенств/уравнений, поскольку при создании сложной управляющей системы важной задачей является задача синтеза оптимальных в некотором смысле компонент данной системы. В нашем случае рассматривается оптимизация компонент параллельной композиции конечных и временных автоматов (глава 1) и оптимизация полуавтоматных компонент относительно операции конкатенации. Для формального решения задачи необходимо определить математическую модель, используемую для описания поведения компонент и системы в целом, точность, с которой синтезированная система должна соответствовать спецификации, а также правила совместного функционирования элементов системы. В первом случае задача сводится к нахождению общего решения уравнения $C \lozenge X \approx S$ с автоматными компонентами, где X – неизвестная (интересующая нас) компонента, контекст C описывает поведение известной части системы, $\lozenge$ – операция параллельной композиции элементов системы и $\approx$ – отношение конформности (соответствия), в котором должны находиться синтезируемая система и ее спецификация S. Для различных приложений эти параметры определяются по-разному. Если уравнение разрешимо, то решение может быть не единственным. В этом случае желательно попытаться найти общее решение уравнения, которое описывает все возможные изменения в поведении компоненты, не влияющие на точность описания поведения системы. Такое общее решение может, например, быть использовано для выбора оптимальной в некотором смысле реализации компоненты.

В настоящей работе предполагается, что для описания поведения всех систем используются системы с конечным числом переходов, в частности, (полу)автоматные модели, в качестве операции композиции используется параллельная композиция, и соответственно рассматриваются задачи решения автоматных и полуавтоматных уравнений и систем таких уравнений относительно операции параллельной композиции.

Пусть заданы полностью определенные наблюдаемые автоматы $S = (S, I_1 \times I_2, O_1 \times O_2, T_S, s_0)$ и $A = (A, I_1 \times V, O_1 \times U, T_A, a_0)$ (рисунок 1.1).

Выражение $A \lozenge X \approx S$, в котором X есть автомат с входным алфавитом $I_2 \cup U$ и выходным алфавитом $O_2 \cup V$, называется *параллельным автоматным уравнением.* Соответственно, выражение $A \lozenge X \leq S$ называется *параллельным автоматным неравенством.* Автомат B с входным алфавитом $I_2 \cup U$ и выходным алфавитом $O_2 \cup V$ называется *решением неравенства* $A \lozenge X \leq S$, если $A \lozenge B \leq S$. Автомат B называется *решением* $A \lozenge X \approx S$, если $A \lozenge B \approx S$.

Решение $Largest$ называется *наибольшим решением* неравенства $A \lozenge X \leq S$ (уравнения $A \lozenge X \approx S$), если каждое решение неравенства (уравнения) есть редукция автомата $Largest$.

Введенное таким образом автоматное неравенство (уравнение) передает физический смысл его решения. Решение автоматного неравенства $A \lozenge X \leq S$ (уравнения $A \lozenge X \approx S$) соответствует автомату, который при параллельной композиции с автоматом A, определяет поведение, которое содержится в (совпадает с) поведении (-ем) автомата S.

Для введения системы параллельных автоматных уравнений рассмотрим следующую задачу. Предположим, что существует агент, который вынужден работать в различных контекстах, причем при работе в каждом контексте требуется обеспечить свой уровень сервиса. Если поведение контекста, сервиса и агента можно описать соответствующими конечными автоматами, то, решая задачу синтеза такого агента, возникает задача решения системы автоматных уравнений. В мультиагентных системах взаимодействие между компонентами обычно описывается с помощью оператора параллельной композиции, и соответственно получается система параллельных автоматных уравнений.

Пусть даны полностью определенные наблюдаемые автоматы S_j и A_j, $j = 1, .., n$. *Системой параллельных автоматных уравнений* называется совокупность уравнений вида:

$$\begin{cases} A_1 \lozenge X \approx S_1 \\ A_2 \lozenge X \approx S_2 \\ \quad \dots \\ A_n \lozenge X \approx S_n \end{cases} \qquad (1)$$

Пусть каждое уравнение системы соответствует композиции на рисунке 2.1. Неизвестная компонента X является автоматом с входным алфавитом $I_2 \cup U$ и выходным алфавитом $O_2 \cup V$. Автомат B, определенный в алфавитах неизвестной компоненты X, называется *решением системы уравнений* (1), если B является решением каждого уравнения системы.

Решение $Largest$ называется *наибольшим решением системы уравнений* (1), если любое решение системы уравнений является редукцией автомата $Largest$.

Решением системы является редукция пересечения наибольших решений уравнений системы. Это следует из того, что всякое разрешимое автоматное уравнение имеет наибольшее решение $Largest_j$, $j = 1, \ldots, n$. Обычно для нахождения наибольшего решения системы параллельных автоматных уравнений предлагается следующий алгоритм.

Алгоритм 3.1 Нахождение наибольшего решения системы параллельных автоматных уравнений.

Вход: Система параллельных автоматных уравнений (1).

Выход: Наибольшее решение системы параллельных автоматных уравнений (если существует).

Шаг 1. Находим наибольшие решения $Largest_1$, …, $Largest_n$ для каждого уравнения системы. Если хотя бы одно уравнение не разрешимо, то система не имеет решения. Конец. Иначе строим пересечение $Largest = Largest_1 \cap \ldots \cap Largest_n$.

Шаг 2. Если $Largest$ есть решение каждого уравнения, то автомат $Largest$ есть наибольшее решение системы. Если $Largest$ не является решением хотя бы одного уравнения, то система не имеет решения.

В общем случае нахождение наибольшего решения (полу)автоматного уравнения имеет экспоненциальную сложность относительно размеров автоматов, коэффициентов уравнений; общее решение системы уравнений является пересечением наибольших решений уравнений. Соответственно, для упрощения решения таких систем возникает вопрос о возможности сведении системы (полу)автоматных неравенств/уравнений к решению одного неравенства/уравнения.

В настоящей работе рассматриваются композиции автоматов, где одна из компонент является встроенной (рисунок 2.1). Показывается, что в ряде случаев система (полу)автоматных неравенств/уравнений может быть сведена к решению одного неравенства/уравнения; рассматриваются частные случаи, когда во всех неравенствах/уравнениях один и тот же контекст или одна и та же спецификация. Оба условия соответствуют ситуациям, которые достаточно часто встречаются на практике, например, при композиции различных сервисов.

Пусть $C_i \lozenge X \leq S_i$ (i = 1, 2) – система параллельных автоматных неравенств, где $S_1 = S_2 = S$.

Теорема 3.1. Пусть $C_i \lozenge X \leq S$ (i = 1, 2) – система параллельных автоматных неравенств, где контексты C_1 и C_2 определены над одинаковыми алфавитами. Наибольшее решение системы неравенств эквивалентно наибольшему решению неравенства $(C_1 \cup C_2) \lozenge X \leq S$.

Следствие 1. Пусть $C_i \lozenge X \leq S$ (i = 1, 2, …, k) – система параллельных автоматных неравенств, где C_1 , …, C_k определены над одинаковыми алфавитами. Наибольшее решение системы неравенств эквивалентно наибольшему решению неравенства $(C_1 \cup \ldots \cup C_k) \lozenge X \leq S$.

Следствие 2. Пусть $C_i \lozenge X \leq S$ (i = 1, 2, …, k) – система параллельных автоматных неравенств, где C_1 , …, C_k определены над одинаковыми алфавитами и S – детерминированный конечный автомат. Наибольшее решение системы неравенств эквивалентно наибольшему решению неравенства $(C_1 \cup \ldots \cup C_k) \lozenge X \cong S$.

Пусть $C_i \lozenge X \leq S_i$ (i = 1, 2) – система параллельных автоматных неравенств, где $C_1 = C_2 = C$.

Теорема 3.2. 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 решение системы неравенств эквивалентно наибольшему решению неравенства $C \lozenge X \leq (S_1 \cap S_2)$.

Следствие 1. Пусть $C \lozenge X \leq S_i$ (i = 1, 2, …, k) – система параллельных автоматных неравенств, где S_1 , …, S_k определены над одинаковыми алфавитами. Наибольшее решение системы неравенств эквивалентно наибольшему решению неравенства $C \lozenge X \leq (S_1 \cap \ldots \cap S_k)$.

Следствие 2. Пусть $C \lozenge X \cong S_i$ (i = 1, 2, …, k) – система параллельных автоматных неравенств, где S_1 , …, S_k – детерминированные конечные автоматы и определены над одинаковыми алфавитами. Наибольшее решение системы неравенств эквивалентно наибольшему решению неравенства $C \lozenge X \cong (S_1 \cap \ldots \cap S_k)$.

В разделе 3.2 рассматривается оптимизация компонент композиции веб-сервисов с целью повышения их качества с точки зрения повышения робастности и безопасности компонент, возможно с точки зрения повышения качества для конечного пользователя.

Развитие интернет-технологий и предоставление различных услуг с использованием сети интернет способствует появлению большого числа различных веб-сервисов. Для веб-сервисов характерна так называемая сервис-ориентированная архитектура, одной из отличительных особенностей которой является возможность построения сервиса из компонент, предоставляемых различными разработчиками. При этом разработчики даже могут не знать, где размещаются некоторые компоненты системы.

Для получения нового веб-сервиса с нужным функционалом можно построить множество комбинаций различных имеющихся в сети Интернет сервисов. Результат такой комбинации называется композицией веб-сервисов. Композиции веб-сервисов могут быть организованы методами оркестровки или хореографии. Они определяют взаимодействие различных составляющих композиции. В ряде случаев такое взаимодействие может быть описано с использованием операции параллельной композиции.

Оркестровка веб-сервисов (англ. orchestration) – способ композиции веб-сервисов, при котором сервисы находятся под контролем единственного центрального процесса – дирижера. Этот процесс координирует выполнение различных операций веб-сервисов, участвующих в композиции. Вызванные веб-сервисы «не знают», что они участвуют в композиции.

Хореография веб-сервисов (англ. choreography) – децентрализованный способ композиции веб-сервисов, при котором сервисы взаимодействуют между собой. Каждый веб-сервис, участвующий в хореографии, должен точно «знать» когда начинать активность и с кем он должен взаимодействовать.

В случае, когда поведение компонент композиции веб-сервисов можно описать конечными (временными) автоматами, и компоненты взаимодействуют в режиме диалога через последовательный обмен запросами и ответами, для описания поведения композиции веб-сервисов можно использовать параллельную композицию автоматов-компонент. Таким образом, для анализа и оптимизации такой композиции сервисов могут быть использованы методы, описанные в разделах 2.1, 2.2 и 3.1.

Важным этапом построения композиции веб-сервисов является выбор оптимальных компонент. Оптимизация компонент композиции веб-сервисов может определяться различными критериями. Помимо функциональных критериев важно рассматривать качество полученной композиции, т.к. она должна удовлетворять потребностям конечного пользователя.

Качество веб-сервиса определяется следующим образом. *QoS* (англ. Quality of Service – качество веб-сервиса) – способность сервиса обеспечивать избирательный подход к различным клиентам наиболее эффективным способом. Обычно, QoS представляют в виде вектора различных атрибутов (параметров) качества, соответствующих данному веб-сервису. Можно привести следующие примеры параметров качества:

- стоимость (execution cost) – денежные затраты, связанные с использованием сервиса;
- время реакции (response time) – отрезок времени между отправкой запроса и получением результата;
- доступность (availability)– вероятность того, что веб-сервис доступен для использования в течение определенного периода времени;
- коэффициент успешного выполнения (successful execution rate) – способность поставщика сервиса доставить вызванный веб-сервис в течение ожидаемого времени;
- репутация (reputation) – средняя оценка, которая дается потребителями веб-сервиса, которые имеют опыт работы с этим сервисом;
- статистика использования (usage frequency) – количество вызовов данного сервиса потребителями;
- безопасность (security) – свойство обеспечения конфиденциальности, целостности и доступности информации.

В работе авторов О.В. Кондратьевой, Н.Г. Кушик, А. Кавалли и Н.В. Евтушенко отмечается, что оценка качества композиции веб-сервисов осуществляется, как правило, на основе структуры композиции с использованием функций агрегации известных параметров качества каждой компоненты.

Таким образом, важным этапом оптимизации композиции веб-сервисов является оценка качества каждой компоненты, в том числе, независимо от других компонент.

В настоящей работе в виде параметра одного из параметров качества веб-сервисов предлагается рассматривать робастность. Под робастностью понимается устойчивость системы к «неожиданным» входным последовательностям. Система должна иметь возможность обрабатывать недопустимые входные данные путем создания соответствующих сообщений. Для того чтобы при оценке качества учитывалась робастность, робастность можно представить как отношение «робастно» обработанных запросов к общему количеству запросов. Формирование запросов для оценки робастности может быть осуществлено на основе различных методов тестирования робастности для программных компонент.

Для повышения качества с точки зрения повышения робастности и безопасности веб-сервисов предполагается использовать результаты, полученные во 2 главе. Для ограничения поведения неробастной компоненты может быть использован алгоритм 2.2. Для очистки входных данных от вредоносного содержимого, которое является небезопасным для рассматриваемой системы, предполагается использовать алгоритмы 2.8 и 2.9 нахождения решения полуавтоматных уравнений.

Для завершения настоящей работы предполагается провести эксперименты с различными системами, представленными автоматными сетями, и оценить их качество относительно различных критериев. При оптимизации аппаратных реализаций автоматных сетей совместно с В.А. Шварцкопом экспериментально показано, что использование минимальной формы каждого компонента при реализации снижает количество аппаратных модулей АЛМ (адаптивный логический модуль) в реализации компонента. Кроме того, рассматривается минимизация компонент относительно «безразличных» входных последовательностей компоненты. Экспериментально показано, что предложенный подход к покомпонентной итеративной оптимизации представляется

достаточно эффективным для автоматных композиций, близких по структуре к последовательной композиции.

Также предполагается провести эксперименты с двумя веб-сервисами построения тестов для дискретных систем на основе автоматных моделей и на основе логических схем, разработанных сотрудниками научной группы кафедры ИТИДиС ТГУ в рамках проекта РНФ № 16-49-03012 (веб-сервисы и их описания доступны по ссылкам http://fsmtestonline.ru/ и http://lctester.online, соответственно). Первый сервис, строящий тесты на основе автоматных моделей, состоит из двух пакетов: пакет построения тестов для конечных детерминированных автоматов и пакет построения тестов для расширенных автоматов. Тесты строятся различными методами для моделей «белого» и «черного ящика». Пакет по расширенным автоматам также содержит графический редактор для создания диаграмм переходов расширенного автомата, заданного пользователем. Второй сервис в некотором смысле дополняет предыдущий, расширяя возможности пользователя при задании формальных спецификаций. В частности, основополагающей моделью в данном случае выступает комбинационная или последовательностная логическая схема. Тесты строятся на основе модели «белого ящика». Такие веб-сервисы представляют большую практическую ценность, т.к. могут быть использованы как в образовательных целях, так и для тестирования реальных программных и аппаратных систем.

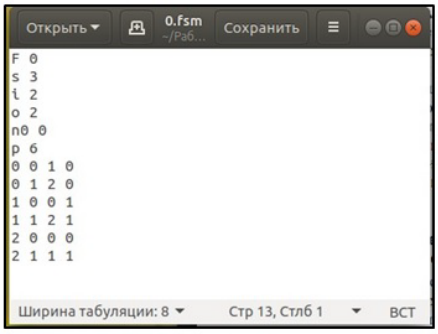
Для оценки качества каждого веб-сервиса предполагается рассмотреть следующий набор параметров: репутация, время генерации тестовых последовательностей, робастность.

Для оценки первого параметра, репутации каждого веб-сервиса, необходимо провести опрос пользователей данных сервисов (например, сотрудников и студентов РФФ и ФПМК ТГУ и других университетов, использующих данные сервисы). Для этого необходимо создать опросники, например, с помощью инструмента Google Forms. Пользователь должен оценить каждый веб-сервис по различным критериям: функциональность, скорость работы, интерфейс и.т.д. Полученные в ходе опроса пользователей оценки будут использованы для различных оценок качества для каждого сервиса.

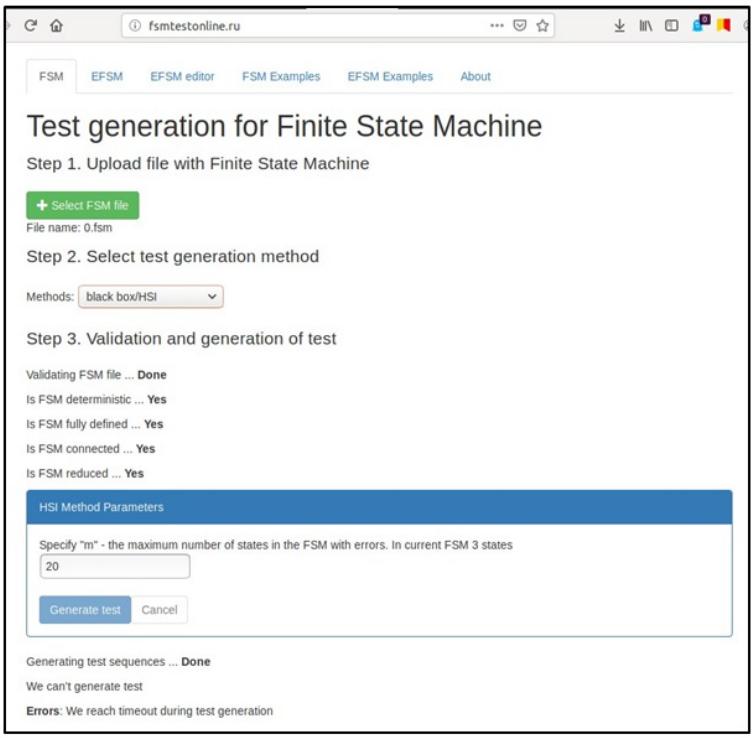
Второй параметр, время генерации тестов, выводится на экран вместе с тестовыми последовательностями при использовании сервиса, т.е. может быть непосредственно использован для оценки качества сервиса.

Для оценки робастности веб-сервисов необходимо проверить поведение каждого веб-сервиса при подаче на вход недопустимых данных и определить отношение числа «робастно» обработанных запросов к общему числу сделанных запросов.

Рассмотрим примеры обработки некорректных запросов первым веб-сервисом, а именно, пакетом построения тестов для конечных детерминированных автоматов. Подадим на вход файл с расширением .fsm с описанием детерминированного конечного автомата (рисунок 3.1(а)). Далее выберем метод построения теста «black box/HSI». После этого необходимо указать максимальное число состояний в реализации. Если времени для обработки запроса недостаточно, например, задано слишком большое число состояний реализации, то сервис не прекращает работу, а по истечении некоторого времени выдает сообщение о том, что тест не может быть сгенерирован, т.к. истекло время ожидания (рисунок 3.1(б)). Таким образом, сервис «робастно» обрабатывает такой запрос.



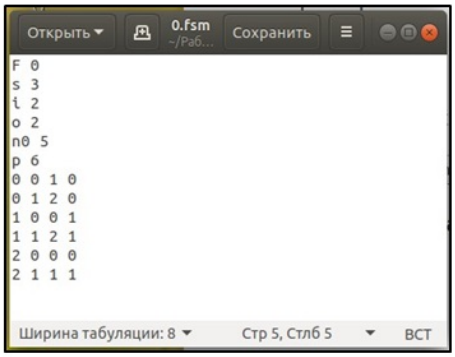
а



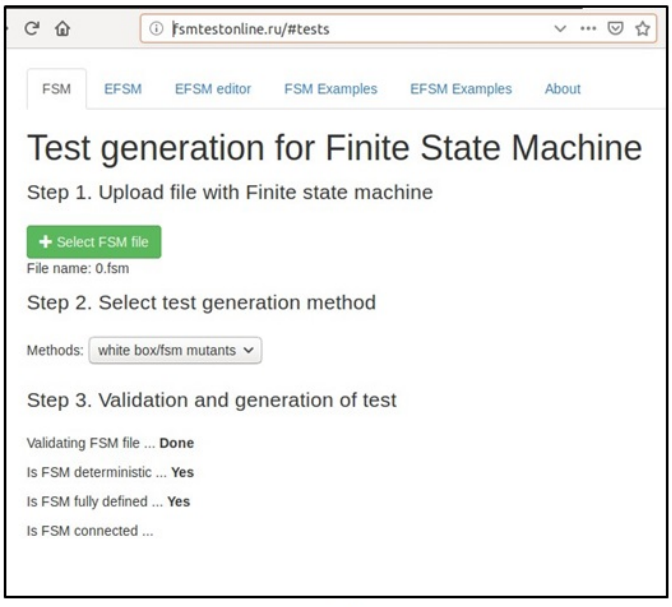
б

Рисунок 3.1 – Пример «робастной» обработки некорректного запроса

Рассмотрим другой пример некорректного запроса. Подадим на вход тот же файл с расширением .fsm. Но, в пятой строке, указывающей на начальное состояние автомата, заменим значение «0» на значение «5» (рисунок 3.2(а)). Так как в данном автомате нет состояния «5», то такой запрос будет недопустимым. При выборе метода генерации теста «white box/fsm mutants» веб-сервис «зависает» на этапе проверки связности автомата (рисунок 3.2(б)). Сообщение об ошибке не выдаётся. Таким образом, сервис не может обработать такой запрос, т.е. этот запрос не относится к запросам, которые «робастно» обрабатываются данным сервисом.



а



б

Рисунок 3.2 – Пример «не робастной» обработки некорректного запроса

Предполагается провести эксперименты по оценке робастности разработанных сервисов, выбрав следующий список некорректных запросов: подача на вход файлов с различными ошибками в описании автомата; подача на вход файлов с описаниями автоматов с очень большим числом состояний; ввод некорректных данных, например текста, в специальное поле для указания максимального числа состояний в реализации; ввод очень большого числа состояний в поле для указания максимального числа состояний в реализации; одновременный запуск веб-сервиса большим числом пользователей. Кроме того, в рабочем режиме предполагается фиксировать отказы сервисов и запросы, которые приводят к такому отказу.

Также, в рамках настоящей работы, предполагается проведение экспериментов по оценке робастности, безопасности и качества веб-сервиса «ТГУ.Расписание» (доступен по ссылке http://schedule.tsu.ru/), начатую при выполнении работ по проекту РНФ № 16-49-03012. Данный сервис предоставляет информацию о расписании групп, аудиторий и преподавателей Томского государственного университета. Для оценки качества можно выбрать следующие параметры: время отклика на запросы, репутация, популярность, робастность.

В заключении кратко перечисляются полученные в диссертации результаты, которые изложены в пунктах научной новизны, теоретической и практической ценности. Также обсуждаются перспективы дальнейших научных исследований.

АНТИПЛАГИАТ
ТВОРИТЕ СОБСТВЕННЫМ УМОМ

Отчет о проверке на заимствования №1

Автор: k@shir.su / ID: 3558155
Проверяющий: (k@shir.su / ID: 3558155)

Отчет предоставлен сервисом «Антиплагиат»- http://users.antiplagiat.ru

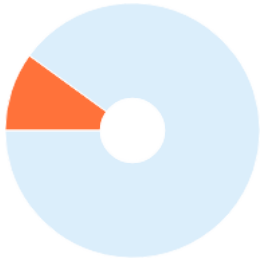
ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 16
Начало загрузки: 13.06.2019 20:46:30
Длительность загрузки: 00:00:01
Имя исходного файла: ВР
Размер текста: 752 кБ
Символов в тексте: 64650
Слов в тексте: 8280
Число предложений: 525

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 13.06.2019 20:46:32
Длительность проверки: 00:00:03
Комментарии: не указано
Модули поиска: Модуль поиска Интернет

ЗАИМСТВОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
10,42%	0%	89,58%



Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Доля в тексте	Источник	Ссылка	Актуален на	Модуль поиска	Блоков в отчете	Блоков в тексте
[01]	0,88%	3,32%	Посмотреть автореферат	http://sun.tsu.ru	16 Ноя 2012	Модуль поиска Интернет	11	30
[02]	0,29%	3,21%	ОПТИМИЗАЦИЯ МНОГОКО...	http://diss.seluk.ru	13 Сен 2018	Модуль поиска Интернет	1	28
[03]	2,05%	2,78%	Метод нахождения прогрес...	http://lib.exdat.com	18 Апр 2016	Модуль поиска Интернет	12	20

Еще источников: 13
Еще заимствований: 7,22%