Запропоновано модифікацію методу верифікації TLA Checker (TLC), спрямовану на зменшення часових витрат, обумовлених процесом перевірки WS-BPEL-описів композитних веб-сервісів на основі відповідних формальних TLA-моделей. Модифікація полягає у серії з BFS- та DFS-обходів

Ключові слова: композитний веб-сервіс, WS-BPEL, специфікація, структура Кріпке, TLC, верифікація, стратифікація

Предложена модификация метода верификации TLA Checker (TLC), направленная на уменьшение временных издержек, обусловленных процессом проверки WS-BPEL-описаний веб-сервисов на основе соответствующих формальных TLA-моделей. Модификация заключается в серии из BFS- и DFS-обходов

Ключевые слова: композитный веб-сервис, WS-BPEL, спецификация, структура Крипке, TLC, верификация, стратификация

УДК 004.053:004.94

WS-BPEL-МОДИФИКАЦИЯ МЕТОДА TLC-ВЕРИФИКАЦИИ

В. В. Шкарупило
Аспирант
Кафедра компьютерных систем и сетей
Запорожский национальный технический университет
ул. Жуковского, 64, г. Запорожье, Украина, 69063
E-mail: vadshkar@yandex.ua

1. Введение

На сегодня использование формальных методов проверки на модели (Model-Checking-методов) в процессе проектирования программных систем можно рассматривать как действенный способ повышения степени доверия к последним [1]. Model-Checking-решения, в отличие от альтернативных направлений (дедуктивной верификации и проверки эквивалентности), могут быть полностью автоматизированы [2]. Это, в свою очередь, можно рассматривать как весомый довод в пользу оптимизации процесса проектирования непосредственно. Отдельно также стоит выделить такое направление как статический анализ кода, которое, в отличие от проверки на модели, сопрягается как с меньшими требованиями к вычислительным ресурсам, так и с меньшим процентом выявляемых ошибок [3].

Рассмотрим в качестве целевой распределенную веб-ориентированную программную систему "композитный веб-сервис" (далее – сервис), а в качестве модели взаимодействия компонентов в составе такой системы – модель централизованного координирования, специфицированную в стандарте WS-BPEL [4]. При создании сервисов на основе этой спецификации допустимо использование набора заданных средств обработки ошибок, которые, впрочем, служат лишь средствами идентификации. С целью же сокращения числа потенциально возможных ошибок разрабатываются подходы к формальной верификации WS-BPEL-спецификаций [5]. Такие подходы направлены в большинстве своем не столько на оптимизацию процесса проектирования, сколько на достижение полноты представления WS-BPEL-концептов в формальных моделях. Подобный характер акцентов можно расценивать как несколько несбалансированный – ввиду "ad-hoc"-режима функционирования таких сервисов [6]. В качестве основных источников временных издержек, сопряженных с верификацией, можно указать асимптотику реализации метода проверки на модели в составе программного инструментария, а также число переменных состояний формальной модели. Асимптотику, в свою очередь, обусловим выбранным методом обхода пространства состояний, а также вычислительной сложностью методов проверки выполнимости булевой формулы. Известно, что вычислительные сложности методов BFS- (Breadth-first Search) и DFS-обходов (Depth-first Search) составляют, соответственно, $O(|V|+|E|)$ и $\Theta(|V|+|E|)$, где V – множество вершин, а E – множество дуг граф-модели системы переходов [7]. Известно также, что для 2-SAT-задачи, где булева формула представляется в 2-КНФ-форме, существует алгоритм решения класса P [8]. Актуальность разработки эффективных методов верификации тем более обоснована по причине экспоненциальной зависимости размера пространства состояний от количества переменных состояний.

2. Анализ литературных данных и постановка проблемы

Среди существующих программных средств автоматизации процедуры формальной верификации отдельно стоит выделить инструментарий UPPAAL – как одно из наиболее производительных решений [9]. Критерием производительности, при этом, является число проверок вида $M,s\models\phi$ за единицу времени, где M – модель системы переходов, s – состояние, $\models$ – оператор выполнимости, а ϕ – формула темпоральной логики, подлежащая проверке в состоянии s модели M.

Тем не менее, производительность верификации не является единственным определяющим фактором при построении формальной модели целевой системы. Существенную роль играют также выразительные

возможности выбранного формализма специфицирования, предопределяющие компактность, наглядность и гибкость реконфигурирования получаемых на его основе моделей. Принимая во внимание "ad-hoc"-сценарии использования композитных сервисов, вопрос выбора приемлемого формализма специфицирования наделим первостепенной значимостью. Таковым (приемлемым) может являться формализм TLA+ темпоральной логики TLA (Temporal Logic of Actions), предложенной Л. Лэмпортом (Leslie Lamport) [10].

Посредством TLA-формул специфицируем функциональные характеристики композитного сервиса (композиции) [11, 12]. Формулы получим на основе концепции "behavior": ϕ как TLA+ формализация последовательности переходов – отношение порядка зададим темпоральным оператором сдвига по времени X (next).

TLA-реализация метода проверки на модели представлена компонентом TLC (TLA Checker) в составе программного инструментария моделирования "TLA Toolbox". С целью оценки производительности TLC были построены формальные TLA- и UPPAAL-модели протокола WS-AT [13]. Назначение протокола – наделение транзакций между компонентами распределенных программных систем ACID-свойствами (Atomicity, Consistency, Isolation, Durability). Измеренные значения временных издержек, сопутствующих верификации моделей, показали, что производительность UPPAAL более чем троекратно превосходит производительность TLC. Полученная TLA-модель, однако, явилась существенно более компактной и наглядной: UPPAAL-синтаксис отождествим с функциональным языком программирования, тогда как синтаксис TLA+ базируется на основе операторов математической логики.

В дополнение к сказанному отметим также, что результаты экспериментальных исследований, связанных с верификацией TLA+ спецификаций, полученных на основе ранее предложенной модели специфицирования, показали, что асимптотическая сложность алгоритма в основе TLC может быть представлена как $\Theta(m^2)$, где m – число атомарных сервисов в составе композиции [14].

3. Цель и задачи исследования

Учитывая результаты TLC-UPPAAL-сравнения, а также ранее приведенные замечания относительно асимптотик методов обхода и задачи 2-SAT, можно утверждать, что актуальной является задача поиска путей оптимизации TLC.

С этой целью в качестве модели системы переходов используем структуру Крипке на множестве литералов AP

$$M = \langle S, \{s_0\}, R, L \rangle, \quad (1)$$

где S – множество состояний, $s_0 \in S$ – начальное состояние, $R \subseteq S^2$ – отношение переходов, а $L: S \to 2^{AP}$ – функция разметки. На модели M будем оперировать понятием "динамики" как конечной последовательностью разметок.

Каждую из функциональных характеристик композиции представим посредством динамики. Теоретически возможное число векторов разметок представим суммой размещений без повторений:

$$n = \sum_{i=1}^{m'} \frac{m!}{(m-i)!},$$

где $m' \le m$ – длина наибольшей из последовательностей.

С целью достижения гибкости реконфигурирования осуществим стратификацию TLA-моделей (рис. 1).

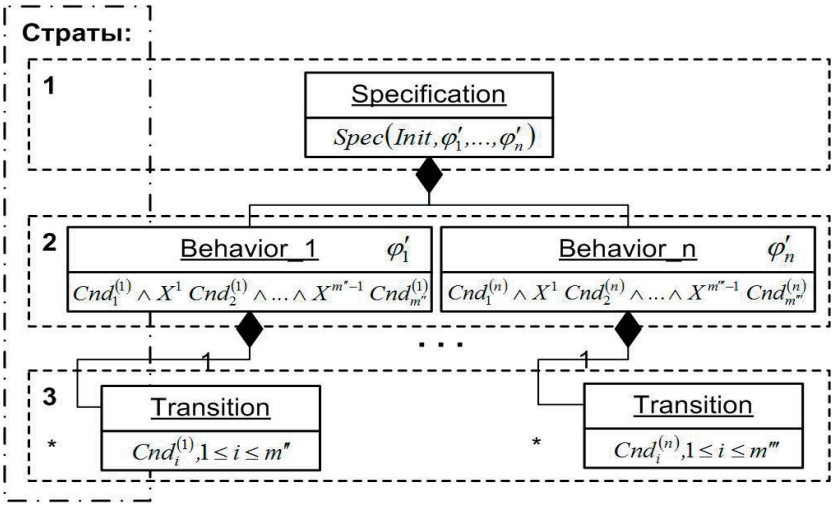

Рис. 1. Иерархические уровни TLA+ описаний

На рис. 1 Spec – целевая TLA-формула, подлежащая TLC-верификации:

$$Spec(Init, \phi'_1, \ldots, \phi'_n) \equiv \begin{cases} Init \wedge G[\phi'_1 \vee \ldots \vee \phi'_n], \\ Init \wedge G[\phi'_1 \wedge \ldots \wedge \phi'_n], \end{cases}$$

где Init – высказывание задания разметки $L(s_0)$ (1), G – темпоральный оператор инвариантности, ϕ'_j $(1 \le j \le n)$ – спецификация динамики. В ϕ'_j формализованное представление $Cnd_{i+1}^{(j)}$ каждого последующего перехода сопровождается оператором X сдвига по времени относительно формализованного представления $Cnd_i^{(j)}$ текущего перехода – верхний индекс j, при этом, указывает на принадлежность к динамике.

Отметим также, что формула $Init \wedge G[\phi'_1 \vee \ldots \vee \phi'_n]$ соответствует нестрогой постановке задачи TLC-верификации, тогда как $Init \wedge G[\phi'_1 \wedge \ldots \wedge \phi'_n]$ – строгой постановке.

Формат представления динамик, приведенный на рис. 1, тем не менее, справедлив лишь для случая последовательных вызовов атомарных сервисов в составе композиции. С целью учета в модели также условных переходов и блоков параллелизма использованию подлежат правила трансляции WS-BPEL-описания в TLA+ спецификацию, приведенные в табл. 1 на основе CSP-формализма Ч. Хоара (C. A. R. Hoare) [15].

Таблица 1

Аксиомы "WS-BPEL-to-TLA"

WS-BPEL-теги	Правила вывода, $1 \le i,k \le m, i \ne k$
<sequence>	$\dfrac{\{Cnd\}Event_i\{Cnd'\},\{Cnd'\}Event_k\{Cnd''\}}{\{Cnd\}Event_i;Event_k\{Cnd''\}}$
<switch> /<case>-fork	$\dfrac{\{Cnd\}Event_i\{Cnd'\},\{Cnd\}Event_k\{Cnd'_{alt}\}}{IF(Cnd)THEN\,Event_i \vee Event_k}$
<switch> /<case>-join	$\dfrac{\{Cnd\}Event_i\{Cnd'\},\{Cnd_{alt}\}Event_k\{Cnd'\}}{IF(Cnd)THEN\,Event_i\,ELSE\,Event_k}$
<flow>-fork	$\dfrac{\{Cnd\}Event_i\{Cnd'\},\{Cnd\}Event_k\{Cnd'_{alt}\}}{\{Cnd\}Event_i \parallel Event_k\{Cnd' \vee Cnd'_{alt}\}}$
<flow>	$\dfrac{\{Cnd\}Event_i\{Cnd'\},\{Cnd_{alt}\}Event_k\{Cnd'_{alt}\}}{\{Cnd \wedge Cnd_{alt}\}Event_i \parallel Event_k\{Cnd' \wedge Cnd'_{alt}\}}$
<flow>-join	$\dfrac{\{Cnd\}Event_i\{Cnd'\},\{Cnd_{alt}\}Event_k\{Cnd'\}}{\{Cnd \wedge Cnd_{alt}\}Event_i \parallel Event_k\{Cnd'\}}$

В табл. 1 $Event_i$ и $Event_k$ есть представления импликаций над элементами множества AP, а Cnd_{alt} и Cnd'_{alt} – альтернативные высказывания задания текущего и последующего переходов.

4. Описание модификации метода и полученные результаты

Предложенная модификация подлежит реализации в два последовательных этапа.

С целью пояснений, при этом, используем пример из [14] – табл. 2.

Таблица 2

Модель динамик

Спецификации	ϕ'_1	ϕ'_2
События	$Event_1 \wedge X\,Event_2$	$Event_2 \wedge X\,Event_1$
Структура Крипке	$M = \langle S^{(1)} \cup S^{(2)}, \{s_0\}, R^{(1)} \cup R^{(2)}, L \rangle$, где	
	$S^{(1)} = \{s_0, s_1, s_3\}$,	$S^{(2)} = \{s_0, s_2, s_3\}$,
	$R^{(1)} = \{(s_0, s_1), (s_1, s_3)\}$,	$R^{(2)} = \{(s_0, s_2), (s_2, s_3)\}$,
	$L(s_1) = \{(v_1 = 1), (v_2 = 0)\}$,	$L(s_2) = \{(v_1 = 0), (v_2 = 1)\}$,
	$L(s_0) = \{(v_1 = 0), (v_2 = 0)\}$, $L(s_3) = \{(v_1 = 1), (v_2 = 1)\}$.	
Множество литералов	$AP = \bigcup_{i=0}^{2^{\lvert V \rvert}-1} L(s_i)$.	

В табл. 2 $V = \{v_1, v_2\}$ – множество переменных состояний.

На первом этапе осуществим проверку связности иерархической структуры спецификации (рис. 1).

С этой целью построим граф $Gr = \langle Vertices, E \rangle$, где $Vertices$ – множество вершин, а $E \subset Vertices^2$ – множество дуг.

Строки TLA+ спецификации представим множеством $Strs$. Также сформируем множество ключевых слов $Keys = \{"Spec", "Fi", "Cnd", "Init"\}$, где "Fi" – шаблон для ϕ'_j.

Элементы $Vertices$ представим парами вида $(head, Tail)$.

Значения head и Tail получим путем построчного синтаксического разбора спецификации.

Для этого введем функции взятия подстроки –

$$hsubstr : Strs \times Keys \to \{vtc.head_i\}$$

и $$lsubstr : Strs \times Keys \to \{vtc.Tail_i\},$$

где $vtc.head_i$ – подстрока на основе key -элемента в составе str -подстроки, расположенной слева от символьного обозначения (CO) тождества (==),

$vtc.Tail_i$ – множество подстрок на основе key -элементов в составе str -подстроки, расположенной справа; $vtc \in Vertices$, $key \in Keys$, $str \in Strs$, $1 \le i \le |Vertices|$.

Дуги вида $vtc' = E(vtc)$, где $vtc' \in Vertices$, проведем при условии, что $vtc'.head \in vtc.Tail$. Шаблоны элементов E получим на основе предложенного способа стратификации (рис. 1):

- $\langle ("Spec", \{"Init", "Fi_1", ..., "Fi_n"\}), ("Fi_j", \{"Cnd_1", ..., "Cnd_m"\}) \rangle$;

- $\langle ("Spec", \{"Init", "Fi_1", ..., "Fi_n"\}), ("Init", \varnothing) \rangle$;

- $\langle ("Fi_j", \{"Cnd_1", ..., "Cnd_m"\}), ("Cnd_i", \{"Cnd_k"\}) \rangle$,

где $1 \le i,k \le m$, $k \ne i$;

- $\langle ("Cnd_i", \{"Cnd_k"\}), ("Cnd_k", \{"Cnd_l"\}) \rangle$,

где $1 \le l \le m$, $l \ne k \ne i$;

- $\langle ("Cnd_l", \{"Init"\}), ("Init", \varnothing) \rangle$,

где $("Init", \varnothing)$ – терминальная вершина.

На первом этапе обойдем вершины BFS-обходом за $O(|Vertices| + |E|)$. Gr с этой целью представим списком смежности, в котором также сохраним с привязкой к вершинам ранее выделенные str -подстроки. Представим эти подстроки множеством Residuals и используем впоследствии на втором этапе. Отметим также, что в программной реализации метода с вершинами графа ассоциируем тройки, включающие соответствующие элементы $residual \in Residuals$. Представление вершин в виде пар выполнено с целью подчеркнуть действия, осуществляемые на текущем шаге.

BFS-обход начнем с

$$s = vtc \in Vertices : vtc.head = hsubstr(str, "Spec"),$$

где $s \in Vertices$ – исток, $str \in Strs$, а $"Spec" \in Keys$. Обход будем производить до тех пор, пока не достигнем терминальной вершины с $vtc'.head = hsubstr(str', "Init")$,

где $str' \in Strs$, а $"Init" \in Keys$. Обозначим такую вершину как t – сток.

При синтаксическом разборе TLA+ спецификации используем множество СО $\{'[]','/\','\/'\} = A$, где $'[]' \in A$ – СО темпорального оператора инвариантности, а $'/\','\/' \in A$ – СО логических операторов $\wedge$ и $\vee$. Элементы A подлежат разбору на основе следующих правил:

- $'[]' \in A$ идентифицируем в качестве признака $s \in Vertices$;

- $\forall vtc.head = hsubstr(str, key) : key \in Keys \setminus \{"Init","Cnd"\}$ суммарное число элементов $'/\','\/' \in A$ в $str \in Strs$ должно составлять значение $|vtc.Tail| - 1$;

- исключение составляет вершина $t \in Vertices$ – на рассматриваемом шаге преследуем цель лишь выявить $"Init" \in Keys$ как один из признаков терминальной вершины в составе пути-листинга BFS-обхода. Полученные пути представим множеством изоморфных подграфов

$$Paths = \left\{ \left\langle Vertices_j, E_j \right\rangle \middle| 0 \le j \le n \right\}, \quad (2)$$

графа Gr, обобщенных следующим свойством:

$$\forall j \; Vertices_j \supseteq \{s,t\}.$$

Введем ограничение на мощность полученного множества – $|Paths| \ge 1$. В составе $Paths$ (2) в обязательном порядке должен присутствовать элемент вида $\langle \{s,t\}, \{(s,t)\} \rangle$ как граф-представление пути $p_0 = s,t$. Будем рассматривать p_0 в качестве средства указания следственно-причинной связи между целевой TLA-формулой $Spec$, подлежащей TLC-верификации, и высказыванием $Init$ задания разметки начального состояния $L(s_0)$ структуры Крипке.

Результат разбора спецификации на основе табл. 2 с использованием правил трансляции (табл. 1) представлен на рис. 2.

Содержимое рис. 2 следует рассматривать как результат приведения бинарного дерева решений к RO-BDD-виду (Reduced Oriented Binary Decision Diagram) путем удаления дублирующих вершин и инцидентных им дуг [16].

На рис. 2 представлены пять путей:

$p_0 = vtc_0, vtc_7$;

$p_1 = vtc_0, vtc_1, vtc_5, vtc_7$;

$p_2 = vtc_0, vtc_1, vtc_3, vtc_5, vtc_7$;

$p_3 = vtc_0, vtc_2, vtc_6, vtc_7$;

$p_4 = vtc_0, vtc_2, vtc_4, vtc_6, vtc_7$.

Два пути из приведенного состава элементов $Paths$ соответствуют специфицированным согласно табл. 2 динамикам – p_2 и p_4. Оставшиеся пути, при этом, подлежат выявлению и удалению на втором (заключительном) этапе рассматриваемой модификации.

Если

$\forall vtc \in Vertices \setminus \{t\}$

выполняется условие $|vtc.Tail| = |E(vtc)|$,

получим связный граф

$Gr = \langle \{s,t\} \subseteq Vertices, E \rangle : s.head = "Spec"$,

$t.head = "Init"$, а $t.Tail = \varnothing$.

В противном случае будем утверждать, что спецификация содержит ошибки.

Поскольку ROBDD-представление является канонической формой представления булевой формулы, а при синтаксическом разборе содержательную нагрузку темпоральных операторов мы не учитывали, такое представление обосновано считать каноническим относительно прообраза $Spec'$ спецификации $Spec$ – булевой формулы как каркаса темпоральной формулы.

На втором этапе осуществим обратный обход посредством DFS-метода.

С этой целью используем пути $p_1,...,p_4$, полученные в следствии осуществления первого этапа. Путь p_0, при этом, рассматривать не будем.

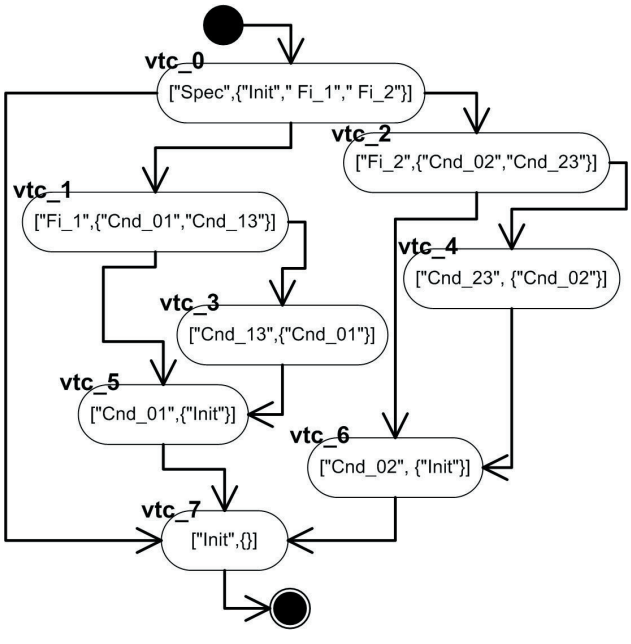

Рис. 2. Граф связности иерархической структуры

На рис. 2 видим два фиктивных пути – не соответствующих допустимым последовательностям событий, предопределенным операторами сдвига по времени (табл. 2) – p_1 и p_3. Выполним проверку непосредственно высказываний-спецификаций переходов – элементов страты 3 (рис. 1).

С этой целью на основе элементов $Paths$ построим граф импликаций $Gr' = \langle Vertices', E' \rangle$, где $Vertices' = \bigcup_{j=1}^{n} Vertices_j$, а $E' = \bigcup_{j=1}^{n} E_j$. Для Gr' DFS-обход запустим с вершины t.

Этап рассмотрим на примере содержимого табл. 2:

- выполним подстановку элементов множества $L(s_0)$ в качестве аргументов

$Init(ap_1, ap_2) \equiv (v_1 = 0) \wedge (v_2 = 0)$,

где $ap_1, ap_2 \in L(s_0)$ –

поскольку $t.head = hsubstr(str, "Init")$;

также создадим список (изначально пустой) выполнимых динамик, который представим множеством $FiList$;

- при условии истинности $Init$ осуществим переходы:

$E'(t) = vtc$,

где $vtc \in \{vtc_3, vtc_4, vtc_5, vtc_6\} \subset Vertices'$; идентифицируем значение $vtc.head$; если соответствующая подстрока $vtc.residual \in Residuals$ содержит в своем составе IF – THEN – ELSE -конструкцию вида:

IF (Init) THEN $Event_i$ ELSE UNCHANGED $<< v_i >>$,

где $Event_i \equiv \neg(v_i = 0) \vee (v_i = 1)$, а модификатор UNCHANGED указывает на неизменность значения переменной состояний v_i, создадим рабочую копию $L(s_0)$ – $L'(s_0)$, в которой заместим элемент $(v_i = 0) \in L(s_0)$ элементом $(v_i = 1)$; на основе $L'(s_0)$ с использованием оператора $\wedge$ получим спецификацию новой разметки; при условии тождественного равенства полученной спецификации IF– аргументу в составе некоторой подстроки $vtc'.residual \in Residuals$: $vtc' = E'(vtc)$, осуществим соответствующий переход; шаг будем повторять итерационно до тех пор, пока <<Eqn1366.eps>>, где $key \in \{"Fi","Spec"\} \subset Keys$; в противном случае осуществим возврат к $t \in Vertices'$ и запустим новый цикл DFS-итераций;

- для случая тождеств вида

$vtc'.head \equiv hsubstr(str,"Fi")$

проверим выполнимость Cnd -высказываний, указанных в качестве $vtc'.Tail$ -элементов – если ϕ_j' впоследствии примет истинное значение, занесем в FiList соответствующий указатель;

- при условии истинности тождества

$s.head \equiv hsubstr(str,"Spec")$, осуществим подстановку элементов, указанных в FiList в качестве аргументов Spec; выполнимость целевой спецификации-формулы, в свою очередь, зависит также и от выбранной постановки задачи TLC-верификации – строгой или нестрогой.

В результате осуществления серии шагов заключительного этапа в FiList будут находиться указатели на спецификации динамик, соответствующих путям и p_2 и p_4 – изоморфным подграфам графа Gr'. Gr', в свою очередь, будет состоять только из указанных подграфов.

С целью проведения экспериментальных исследований был использован ранее полученный набор из 10^2 WS-BPEL-описаний для композиций с $m = 2,12,...,42$ [14].

Результаты проведенных экспериментальных исследований показали, что внесение рассмотренной в работе модификации в реализацию метода TLC-верификации оказало положительный эффект применительно к формальным TLA-моделям WS-BPEL-описаний, построенным на основе предложенного способа стратификации (рис. 3).

На основе экспериментальных данных с использованием программного инструментария "TableCurve 5.01" была получена интерполяционная функция

$$y(m) = \frac{1}{a + b \cdot m \cdot \ln(m)},$$

где $a = 1.114$, а $b = -1.34 \cdot 10^{-3}$.

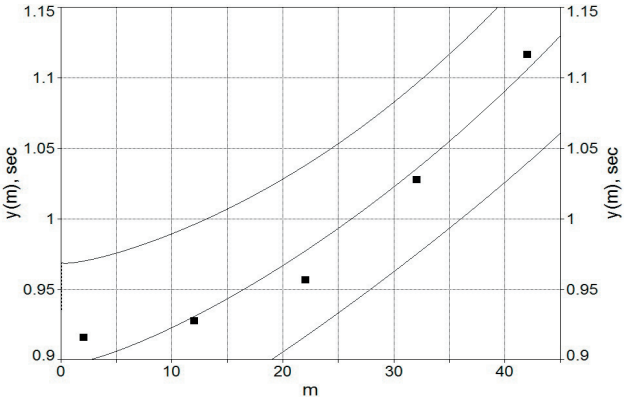

Рис. 3. Результаты проверки предложенной модификации: m – число переменных состояний; $y(m)$ – интерполирующая функция зависимости временных затрат на верификацию от m

Доверительные интервалы проведены для доверительной вероятности 0.95, а значение критерия χ^2 составило 0.968.

Полученные результаты позволяют представить вычислительную сложность алгоритма в основе модифицированного метода как $\Theta(m \cdot \ln(m))$. Это, в свою очередь, можно рассматривать в качестве признака эффективности реализации модифицированного метода TLC-верификации.

5. Выводы

Таким образом, в работе предложена WS-BPEL-модификация метода TLC-верификации, направленная на оптимизацию процесса проектирования композитных веб-сервисов. С этой целью был предложен способ стратификации формальных TLA-моделей, согласно которому выделению подлежат три иерархических уровня – спецификаций переходов, динамик (последовательностей переходов) и целевой TLA-формулы-модели исходного WS-BPEL-описания, подлежащей проверке. Выделение указанных уровней поспособствовало повышению наглядности и гибкости реконфигурирования формальных моделей, а также послужило основой для реализации модификации.

Результаты проведенных экспериментальных исследований, в свою очередь, подтвердили обоснованность внесения предложенной модификации в реализацию метода TLC-верификации при построении формальных TLA-моделей с использованием предложенного способа стратификации. Такая реализация, в свою очередь, может быть охарактеризована как эффективная – вычислительная сложность алгоритма в основе метода составила $\Theta(m \cdot \ln(m))$.

Литература

1. Grumberg, O. 25 Years of Model Checking: History, Achievements, Perspectives [Text] / O. Grumberg, H. Veith. – Berlin: Springer, 2008. – 231 p. – ISBN 10 3-540-69849-3.
2. Тарасюк, О. М. Формальные методы разработки критического программного обеспечения [Текст] : лекционный материал / О. М. Тарасюк, А. В. Горбенко; под ред. В. С. Харченко – МОН Украины, Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», 2009. – 214 с. – ISBN 978-966-96770-8-2.

3. Vorobyov, K. Comparing Model Checking and Static Program Analysis: A Case Study in Error Detection Approaches [Текст] / K. Vorobyov, P. Krishnan // Proc. 5th Int. Workshop on Systems Software Verification, SSV 2010 (Vancouver, Canada, October 6 – 7, 2010). – P. 1 – 7.
4. Web Services Business Process Execution Language, Version 2.0 [Электронный ресурс] // OASIS Standard, April 11, 2007. – Режим доступа : http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf. – Заголовок с экрана.
5. Cao, T-D. An Approach to Automated Runtime Verification for Timed Systems: Applications to Web Services [Текст] / T-D. Cao, R. Castanet, P. Felix, K. Chiew // Journal of Software. – 2012. – Vol. 7, No. 6. – P. 1338 – 1350.
6. Dhore, S. R. QoS Based Web Services Composition using Ant Colony Optimization: Mobile Agent Approach [Текст] / S. R. Dhore, M. U. Kharat // International Journal of Advanced Research in Computer and Communication Engineering. – 2012. – Vol. 1, No. 7. – P. 519 – 527.
7. Кормен, Т. Х. Алгоритмы: построение и анализ [Текст] : пер. с англ. / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн. – 2-е изд. – М.: Вильямс, 2005. – 1296 с. – ISBN 5-8459-0857-4.
8. Zheng, L. Improving SAT using 2-SAT / L. Zheng, P. J. Stuckey [Текст] // Australian Computer Science Communications. – 2002. – Vol. 24, No. 1. – P. 331 – 340.
9. Larsen, K. Model-based Verification and Analysis for Real-Time Systems [Текст] / K. Larsen // Proc. NATO Advanced Study Institute – Int. Summer School MOD 2012 on Engineering Dependable Software Systems (Marktoberdorf, Germany, July 31 – August 12, 2012). – 155 p.
10. Lamport, L. Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers [Текст] / L. Lamport. – Boston.: Addison-Wesley, 2002. – 364 p. – ISBN 0-321-14306-X.
11. Шкарупило, В. В. Концептуальная модель процесса автоматизированного синтеза композитных веб-сервисов [Текст] / В. В. Шкарупило, Р. К. Кудерметов, Т. А. Паромова // Сборник научных трудов ДонНТУ. Серия : Информатика, кибернетика и вычислительная техника. – Донецк : ДонНТУ, 2012. – Вып. 15 (203). – С. 231 – 238.
12. Shkarupylo, V. V. An Approach to Composite Web Services Formal Verification [Текст] / V. V. Shkarupylo, R. K. Kudermetov // Сборник научных трудов ДонНТУ. Серия : Информатика, кибернетика и вычислительная техника. – Донецк : ДонНТУ, 2012. – Вып. 16 (204). – С. 129 – 133.
13. Ravn, A. P. A formal analysis of the web services atomic transaction protocol with UPPAAL [Текст] / A. P. Ravn, J. Srba, S. Vighio // Proc. 4th Int. Conf. on Leveraging Applications of Formal Methods, Verification and Validation, ISoLA 2010 (Heraklion, Crete, October 18 – 20, 2010). – P. 579 – 593.
14. Шкарупило, В. В. Модель TLA-спецификации композитного веб-сервиса с множеством динамик [Текст] / В. В. Шкарупило // Радіоелектроніка, інформатика, управління. – Запорожье: ЗНТУ, 2013. – Вып. 1 (28). – С. 94 – 100.
15. Хоар, Ч. Взаимодействующие последовательные процессы [Текст] : пер. с англ. / Ч. Хоар. – М.: Мир, 1989. – 264 с. – ISBN 5-03-001043-2.
16. Семенов, А. А. Двоичные диаграммы решений в параллельных алгоритмах обращения дискретных функций [Текст] / А. А. Семенов, А. С. Игнатьев, Д. В. Беспалов // Параллельные вычислительные технологии, ПаВТ 2009 : III междунар. науч. конф., 30 марта – 3 апр. 2009 г. : тезисы докл. – Нижний Новгород : ННГУ им. Н. И. Лобачевского, 2009. – С. 688 – 696.