УДК 519.767.2

DOI: 10.15827/0236-235X.119.456-460

Дата подачи статьи: 05.06.17

2017. Т. 30. № 3. С. 456–460

ВЕРИФИКАЦИЯ МОДЕЛЕЙ СИСТЕМ НА БАЗЕ ЭКВАЦИОНАЛЬНОЙ ХАРАКТЕРИСТИКИ ФОРМУЛ LTL

Ю.П. Кораблин, д.т.н., профессор, y.p.k@mail.ru;

А.С. Кочергин, аспирант, kocherginalexandr@gmail.com;

А.А. Шипов, к.т.н., a-j-a-1@yandex.ru

(Российский государственный социальный университет,
ул. Вильгельма Пика, 4, г. Москва, 129226, Россия)

Метод верификации на моделях Model Checking уже давно получил широкое признание в области, связанной с оценкой качества работы программных и технических систем. Такие ключевые компании в области IT-индустрии, как Intel, Microsoft, Amazon и другие, активно применяют его на этапах разработки и сопровождения своих продуктов. Успех Model Checking, безусловно, не является случайным, поскольку именно его появление и развитие позволили решить множество проблем в области верификации, а именно: проблемы унифицированного представления программных и технических систем, формального задания требований, автоматизации этапов верификации, верификации больших распределенных программных систем и другие.

Однако постоянное развитие современных технологий и темпы роста сложности современных программных систем ставят перед Model Checking все новые проблемы, которые могут стать непреодолимым препятствием на пути эффективной верификации. Поэтому необходимо постоянное совершенствование теории и инструментов данного метода.

В статье авторами подробно рассматривается реализация алгоритма верификации метода Model Checking для формул логики линейного времени LTL на базе новой нотации RLTL (Recursive Linear Temporal Logic), которая является рекурсивным представлением формул логики линейного времени. Поскольку на базе RLTL могут быть заданы как модель верифицируемой системы, так и требования к ней, можно избежать необходимости их предварительного преобразования к автоматам Бюхи и сразу приступить к процессу верификации, что упростит алгоритм метода и повысит его эффективность.

Ключевые слова: верификация, Model Checking, эквивалентная характеристика RLTL, модель Крипке, автомат Бюхи, формула временной логики, LTL, CTL.

Верификация программных и технических систем является сегодня одним из важнейших этапов их жизненного цикла. Для этих целей уже создано и успешно апробировано множество различных средств и методов разного типа сложности и применимости. Однако наиболее часто используемыми и востребованными, как правило, остаются средства формальной верификации, работа которых может быть в той или иной мере автоматизирована в отличие от, например, динамических методов верификации и экспертиз.

Среди формальных методов верификации наибольшую популярность за свою практичность и эффективность получил так называемый метод проверки на моделях Model Checking [1, 2]. Базовая идея алгоритма этого метода для требований к системе, заданных в виде формул логики линейного времени LTL [3, 4], представлена в виде схемы на рисунке 1.

Использование данной схемы в чистом виде при верификации моделей систем больших размеров (более 2^{20} состояний) во многих случаях может приводить к тому, что верификация либо будет выполняться очень долго, либо вообще не будет выполнена. Для решения данных проблем и достижения более высокой производительности сегодня существует ряд специальных средств и методов, которые за счет определенных манипуляций над моделью системы и/или над требованиями к ней позволяют добиться приемлемой производи-

тельности [5–7]. В частности, одним из инструментов повышения эффективности выполнения верификации является предложенный в работе [8] метод абстракции и унификации RLTL-моделей (Recursive Linear Temporal Logic).

В данной статье авторами предложен принципиально иной подход к повышению производительности процесса верификации, который явля-

Рис. 1. Схема работы алгоритма верификации формул LTL методом Model Checking

Fig. 1. The scheme of LTL formulas verification algorithm using the Model Checking method

ется результатом использования предложенной в [9] новой нотации LTL, обозначаемой как RLTL. Показана возможность выполнения на базе RLTL и самого процесса верификации, что позволит упростить алгоритм верификации методом Model Checking и привести его к виду, представленному на рисунке 2.

RLTL-нотация

Рассмотрим представления основных базовых конструкций логики LTL, заданных в RLTL-нотации:

LTL	RLTL
$\varphi_1 \vee \varphi_2$	$\varphi_1 + \varphi_2$
$\varphi_1 \wedge \varphi_2$	$\{\varphi_1, \varphi_2\}$
$X\varphi$	$\Delta \circ \varphi$

Запись $\{\varphi_1, \varphi_2\}$ означает множество μ , состоящее из предикатов φ_1 и φ_2 .

В RLTL приняты следующий обозначения.

$\circ$ – оператор продолжения (конкатенация выражений), что позволяет использовать оператор X в неявном виде, а также упростить его восприятие, в частности, запись $\mu_1 \circ \mu_2$ будет пониматься как « μ_2 следует за μ_1 ».

μ – множество выполнимых на конкретном этапе вычислительного процесса атомарных предикатов, задающих те или иные свойства системы.

Δ – предикат неопределенности, задающий неопределенное множество свойств системы μ , выполнимых на конкретном этапе вычислительного процесса. Отрицанием предиката неопределенности является некоторое другое неопределенное подмножество свойств μ' , а отрицанием отрицания – некоторое третье неопределенное подмноже-

ство μ'' . Таким образом, отрицание предиката неопределенности является неполным, а каждое последующее отрицание дает неопределенное подмножество атомарных предикатов. Каждое из этих подмножеств ввиду своей неопределенности также может быть обозначено через Δ .

$\perp$ – оператор остановки вычислительного процесса. Это означает, что, как только данный оператор встречается в конкретный момент времени вычислительного процесса, в дальнейшем ни одно свойство системы не будет выполнено.

Сформулируем основные аксиомы нотации RLTL:

A1.	$FF\varphi = F\varphi$	(FF)
A2.	$GG\varphi = G\varphi$	(GG)
A3.	$\neg F\varphi = G\neg\varphi$	($\neg F$)
A4.	$\neg G\varphi = F\neg\varphi$	($\neg G$)
A5.	$G\varphi = \varphi \circ G\varphi$	(G)
A6.	$F\varphi = \varphi + \Delta \circ F\varphi$	(F)
A7.	$G\varphi + \neg G\varphi = \Delta$	(+ G)
A8.	$F\varphi + \neg F\varphi = \Delta$	(+ F)
A9.	$G\varphi_1 \circ \varphi_2 = G\varphi_1$	(G $\circ$)
A10.	$F\varphi_1 \circ \varphi_2 = F(\varphi_1 \circ \varphi_2)$	(F $\circ$)
A11.	$U(\varphi_1, \varphi_2) = \varphi_2 + \varphi_1 \circ U(\varphi_1, \varphi_2)$	(U)
A12.	$\neg(\mu \circ \mu_1) = \neg\mu + \Delta \circ \neg\mu_1$	($\neg\mu \circ$)
A13.	$\{\mu, \Delta\} = \mu$	(Δ)
A14.	$\{\varphi_1, \Delta \circ \varphi_2\} = \varphi_1 \circ \varphi_2$	($\Delta \circ$)
A15.	$\neg\mu = (\Delta \setminus \mu)$	$\neg\mu$
A16.	$\perp \circ \mu = \mu \circ \perp = \perp$	$\perp \circ$
A17.	$\perp + \mu = \mu$	$\perp +$

Правила вывода в RLTL: R1. $\neg\Delta \rightarrow \Delta$.

Лемма 1. Оператор F логики LTL, рекурсивное представление которого задано как $Fq = q \vee X(Fq)$, может быть представлен в RLTL-нотации в виде $F' = q + \Delta \circ F'$.

Доказательство. Обозначим Fq через F' . Тогда $F' = Fq = q \vee X(Fq) = q \vee X(F')$. Отсюда, в соответствии с правилами представления формул LTL в RLTL, получаем: $F' = q + \Delta \circ F'$. ■

Лемма 2. Оператор G логики LTL, рекурсивное представление которого задано как $Gq = q \wedge X(Gq)$, может быть представлен в RLTL-нотации в виде $F' = q \circ F'$.

Доказательство. Обозначим Gq через F' . Тогда $F' = Gq = q \wedge X(Gq) = q \wedge X(F')$. Отсюда, в соответствии с правилами представления формул LTL в RLTL, получаем

$$F' = \{q, \Delta \circ F'\} = [A14] = q \circ F'. \blacksquare$$

Лемма 3. Оператор U логики LTL, рекурсивное представление которого задано как $pUq = q \vee (p \wedge X(pUq))$, может быть представлен в RLTL-нотации в виде $F' = q + p \circ F'$.

Доказательство. Обозначим pUq через F' . Тогда $F' = pUq = q \vee (p \wedge X(pUq)) = q \vee (p \wedge X(F'))$.

Отсюда, в соответствии с правилами представления формул LTL в RLTL, получаем: $F' = q + \{p, \Delta \circ F'\} = [A14] = q + p \circ F'$. ■

Построение синхронной композиции двух систем рекурсивных уравнений

Идея верификации моделей систем, заданных на базе RLTL-нотации, практически ничем не отличается от верификации систем, заданных в виде автоматов Бюхи. Для того чтобы проверить, выполняется ли целевое требование, также заданное с помощью RLTL, на модели системы или нет, необходимо запустить две системы синхронно (модель и требование к ней) и следить за процессом их синхронной работы. Система, работающая синхронно, осуществляет переход в следующее состояние по некоторому символу p тогда и только тогда, когда вторая система также может выполнить переход из своего текущего состояния в некоторое новое состояние по символу p .

Сформулируем основные аксиомы, необходимые для построения синхронной композиции двух систем рекурсивных уравнений:

- S1. $\mu_1 \otimes \mu_2 = \mu_1$, если $\mu_1 \subseteq \mu_2$,
 μ_2 , если $\mu_2 \subseteq \mu_1$,
 $\perp$ в противном случае,
- S2. $\mu \otimes \perp = \perp$,
- S3. $\mu \otimes \Delta = \mu$,
- S4. $(\mu_1 + \mu_2) \otimes \mu_3 = (\mu_1 \otimes \mu_3) + (\mu_2 \otimes \mu_3)$,
- S5. $(\mu_1 \circ \mu_2) \otimes (\mu_3 \circ \mu_4) = (\mu_1 \otimes \mu_3) \circ (\mu_2 \otimes \mu_4)$.

Под символом $\otimes$ будем понимать оператор синхронной композиции. Все, что стоит справа или слева от данного оператора, должно выполняться синхронно, если это возможно.

Пример построения синхронной композиции

Дано.

Система A:

$$F_{A1} = a \circ F_{A1} + b \circ F_{A2},$$

$$F_{A2} = c \circ F_{A1}.$$

Система B:

$$F_{B1} = c + \Delta \circ F_{B1}.$$

Для построения синхронной композиции расширим систему A следующим образом, полагая, что в случае наступления события «с» система продолжит работать бесконечно долго, то есть заикнется: $F_{B1} = c \circ F_{B2} + \Delta \circ F_{B1}$,

$$F_{B2} = \Delta \circ F_{B2}.$$

Решение.

Построение синхронной композиции $A \otimes B$:

$$\begin{aligned} P_1 &= F_{A1} \otimes F_{B1} = \\ &= (a \otimes c) \circ (F_{A1} \otimes F_{B2}) + (a \otimes \Delta) \circ (F_{A1} \otimes F_{B1}) + \\ &+ (b \otimes c) \circ (F_{A2} \otimes F_{B2}) + (b \otimes \Delta) \circ (F_{A2} \otimes F_{B1}) = \\ &= \perp \circ (F_{A1} \otimes F_{B2}) + (a \otimes \Delta) \circ P_1 + \perp \circ (F_{A2} \otimes F_{B2}) + \\ &+ (b \otimes \Delta) \circ P_2 = \perp + a \circ P_1 + \perp + b \circ P_2 = a \circ P_1 + b \circ P_2. \\ P_2 &= F_{A2} \otimes F_{B1} = (c \otimes c) \circ (F_{A1} \otimes F_{B2}) + (c \otimes \Delta) \circ (F_{A1} \otimes F_{B1}) = \\ &= (c \otimes c) \circ P_3 + (c \otimes \Delta) \circ P_1 = c \circ P_3 + c \circ P_1. \\ P_3 &= F_{A1} \otimes F_{B2} = (a \otimes \Delta) \circ (F_{A1} \otimes F_{B2}) + (b \otimes \Delta) \circ (F_{A2} \otimes F_{B2}) = \\ &= a \circ P_3 + b \circ P_4. \end{aligned}$$

$$P_4 = F_{A2} \otimes F_{B2} = (c \otimes \Delta) \circ (F_{A1} \otimes F_{B2}) = c \circ P_3.$$

Конечная синхронная композиция:

$$P_1 = a \circ P_1 + b \circ P_2,$$

$$P_2 = c \circ P_3 + c \circ P_1,$$

$$P_3 = a \circ P_3 + b \circ P_4,$$

$$P_4 = c \circ P_3.$$

Теорема 1. Пусть заданы две системы рекурсивных уравнений A и B, где $P_1, P_2, \dots, P_n$ – множество метапеременных системы A; $P_2, P_2, \dots, P_{2m}$ – множество метапеременных системы B.

$$P_{1i} = \sum_{j \in N} \alpha_{ij} \circ P_{1ij}, \text{ где } N = \{1, 2, \dots, n\}.$$

$$P_{2i} = \sum_{k \in M} \alpha_{ik} \circ P_{2ik}, \text{ где } M = \{1, 2, \dots, m\}.$$

$$\forall i \forall j \exists l \in N \text{ такое, что } P_{1ij} = P_{1l}$$

$$\forall i \forall k \exists r \in M \text{ такое, что } P_{2ik} = P_{2r}.$$

Тогда синхронная композиция двух систем уравнений A и B, где $P_1 = P_{11} \otimes P_{21}$, конечна.

Доказательство.

$$\text{Обозначим через } \xi(u, v) = P_{1u} \otimes P_{2v}, \quad (*)$$

где $u = 1, 2, \dots, n$; $v = 1, 2, \dots, m$.

Количество выражений $\xi(u, v)$ конечно. По построению синхронной композиции имеем

$$\begin{aligned} \xi(u, v) &= \sum_{j \in N} \alpha_{uj} \circ P_{1uj} \otimes \sum_{k \in M} \alpha_{vk} \circ P_{2vk} = \\ &= \sum_{\substack{j \in N, \\ k \in M}} (\alpha_{uj} \otimes \alpha_{vk}) \circ (P_{1uj} \otimes P_{2vk}) = \\ &= \sum_{\substack{j \in N, \\ k \in M}} (\alpha_{uj} \otimes \alpha_{vk}) \circ P_{jk}, \text{ где все } P_{ij} \text{ входят в множе-} \end{aligned}$$

ство (*). Таким образом, получается, что система уравнений для синхронной композиции A и B конечна. ■

Верификация

Для верификации с помощью RLTL необходимо наличие модели M на базе RLTL, а также проверяемое относительно нее требование ϕ , заданное на базе LTL. Для проверки выполнимости в системе заданного требования ϕ , как и в алгоритме верификации формул логики линейного времени, будет использовано обратное требование $[\phi]$. Главным и самым важным отличием предлагаемого алгоритма от алгоритма верификации формул LTL является то, что верификация на базе RLTL более производительная, поскольку не требуется преобразовывать модель и обратное к ней требование в автоматы Бюхи. При этом осуществляется лишь преобразование обратного требования, заданного с помощью LTL, в RLTL-представление. Переход от LTL к RLTL является тривиальным и выполняется по описанным выше правилам и аксиомам в отличие от алгоритмов перехода от LTL к автоматам Бюхи, которые, как сказано в [1], зачастую выдают автоматы Бюхи экспоненциальной сложности относительно длины формулы ϕ .

Отметим также, что, если формулировать требования к системе сразу на базе RLTL-нотации,

можно в принципе избавиться от всех промежуточных этапов превращения и выполнять процесс верификации сразу после создания модели системы и требований к ней. Указанные выше отличия позволяют существенно повысить производительность процесса верификации по сравнению с верификацией на основе автоматов Бюхи и получить более наглядные представления как модели системы, так и задаваемых требований в виде системы рекурсивных уравнений.

Ключевым процессом в RLTL-верификации, как и в алгоритме верификации формул LTL, является процесс построения синхронной композиции. После того как будет построена композиция двух систем в соответствии с методом, представленным ранее, необходимо выполнить оценку результатов данного процесса. Чтобы исходное требование φ оказалось невыполнимым для модели M , достаточно найти любую ветку вычислительного процесса M , на которой выполняется обратное требование $\neg\varphi$. Если такая ветка существует, она и будет искомым контрпримером. Обратное требование $\neg\varphi$ выполнимо для композиции $M \otimes \varphi$ на некотором ее вычислительном пути тогда и только тогда, когда в $M \otimes \varphi$ присутствуют все уравнения из $\neg\varphi$.

Пример 2. Верификация модели.

Дано.

Модель системы (M):

$$FA_1 = a \circ FA_1 + b \circ FA_2,$$

$$FA_2 = c \circ FA_1.$$

Проверяемое требование (φ) LTL:

$$\varphi = Fc \text{ — когда-нибудь наступит «с»}.$$

Обратное требование ($\neg\varphi$) LTL:

$$\neg\varphi = G\neg c \text{ — всегда будет «с»}.$$

Обратное требование ($\neg\varphi$) RLTL:

$$F_{\neg\varphi} = \neg c \circ F_{\neg\varphi}.$$

Решение.

Построение синхронной композиции $M \otimes \neg\varphi$:

$$\begin{aligned} P_1 &= FA_1 \otimes F_{\neg\varphi} = \\ &= (a \otimes \neg c) \circ (FA_1 \otimes F_{\neg\varphi}) + (b \otimes \neg c) \circ (FA_2 \otimes F_{\neg\varphi}) = \\ &= (a \otimes (\neg c)) \circ (FA_1 \otimes F_{\neg\varphi}) + \\ &+ (b \otimes (\neg c)) \circ (FA_2 \otimes F_{\neg\varphi}) = \\ &= a \circ (FA_1 \otimes F_{\neg\varphi}) + b \circ (FA_2 \otimes F_{\neg\varphi}) = a \circ P_1 + b \circ P_2. \end{aligned}$$

$$P_2 = FA_2 \otimes F_{\neg\varphi} = (c \otimes \neg c) \circ (FA_1 \otimes F_{\neg\varphi}) =$$

$$= \perp \circ FA_1 \otimes F_{\neg\varphi} = \perp.$$

Конечная синхронная композиция:

$$P_1 = a \circ P_1 + b \circ P_2,$$

$$P_2 = \perp.$$

Результат: $P_1 = a \circ P_1$.

Проверяемое свойство $\neg\varphi$ выполняется на M , поскольку в $M \otimes \neg\varphi$ в полной мере было реализовано свойство $\neg\varphi$, а именно, его уравнение на одном из путей вычислительного процесса. Из этого следует, что на этом пути не реализуется φ . Значит, путь $P_1 \rightarrow a \rightarrow P_1 \rightarrow a \rightarrow P_1 \dots \rightarrow a \rightarrow P_1$ является контрпримером.

Рассмотрим более содержательный пример, в котором будет выполняться проверка работы дымового извещателя. Для наглядности на рисунке 3 представлена его модель в виде автомата Бюхи, где

Рис. 3. Модель дымового извещателя на базе автомата Бюхи

Fig. 3. A smoke detector model based on Buchi automaton

a – режим готовности; b – срабатывание датчика дыма; c – опрос соседних датчиков; d – режим тревоги.

Модель дымового извещателя на базе нотации RLTL будет иметь следующий вид:

$$FA_1 = a \circ FA_1 + b \circ FA_2,$$

$$FA_2 = c \circ FA_3,$$

$$FA_3 = d \circ FA_4 + a \circ FA_1,$$

$$FA_4 = d \circ FA_4 + a \circ FA_1.$$

Проверяемое требование (φ) LTL: $\varphi = GF(b \rightarrow Xc)$ – всегда, в случае наступления в будущем в некоторый момент времени события b , на следующем шаге будет выполнено событие c .

Обратное требование ($\neg\varphi$) LTL: $\neg\varphi = FG(b \wedge \neg Xc)$ – когда-нибудь наступит такой момент, что всегда будет наступать b на текущем шаге и на следующем шаге будет $\neg c$.

Обратное требование ($\neg\varphi$) RLTL:

$$FB_1 = b \circ FB_3 + \Delta \circ FB_1,$$

$$FB_2 = b \circ FB_3,$$

$$FB_3 = \neg c \circ FB_2.$$

Построение синхронной композиции $M \otimes \neg\varphi$:

$$\begin{aligned} P_1 &= FA_1 \otimes FB_1 = (a \otimes b) \circ (FA_1 \otimes FB_3) + (b \otimes b) \circ (FA_2 \otimes FB_3) + \\ &+ (a \otimes \Delta) \circ (FA_1 \otimes FB_1) + (b \otimes \Delta) \circ (FA_2 \otimes FB_1) = \\ &= \perp \circ (FA_1 \otimes FB_3) + b \circ (FA_2 \otimes FB_3) + a \circ (FA_1 \otimes FB_1) + \\ &+ b \circ (FA_2 \otimes FB_1) = \\ &= b \circ (FA_2 \otimes FB_3) + a \circ (FA_1 \otimes FB_1) + b \circ (FA_2 \otimes FB_1) = \\ &= b \circ P_2 + a \circ P_1 + b \circ P_3. \end{aligned}$$

$$P_2 = (FA_2 \otimes FB_3) = (c \otimes \neg c) \circ (FA_3 \otimes FB_2) = \perp \circ (FA_3 \otimes FB_2) = \perp.$$

$$P_3 = (FA_2 \otimes FB_1) = (c \otimes b) \circ (FA_3 \otimes FB_3) +$$

$$+ (c \otimes \Delta) \circ (FA_3 \otimes FB_1) =$$

$$= \perp \circ (FA_3 \otimes FB_3) + c \circ (FA_3 \otimes FB_1) =$$

$$= c \circ (FA_3 \otimes FB_1) = c \circ P_4.$$

$$P_4 = (FA_3 \otimes FB_1) = (d \otimes b) \circ (FA_4 \otimes FB_3) +$$

$$+ (d \otimes \Delta) \circ (FA_4 \otimes FB_1) + (a \otimes b) \circ (FA_1 \otimes FB_3) +$$

$$+ (a \otimes \Delta) \circ (FA_1 \otimes FB_1) =$$

$$= \perp \circ (FA_4 \otimes FB_3) + d \circ (FA_4 \otimes FB_1) + \perp \circ (FA_1 \otimes FB_3) +$$

$$+ a \circ (FA_1 \otimes FB_1) = d \circ (FA_4 \otimes FB_1) + a \circ (FA_1 \otimes FB_1) =$$

$$= d \circ P_5 + a \circ P_1.$$

$$P_5 = (FA_4 \otimes FB_1) = (d \otimes b) \circ (FA_4 \otimes FB_3) +$$

$$+ (d \otimes \Delta) \circ (FA_4 \otimes FB_1) + (a \otimes b) \circ (FA_1 \otimes FB_3) +$$

$$+ (a \otimes \Delta) \circ (FA_1 \otimes FB_1) =$$

$$= \perp \circ (FA_4 \otimes FB_3) + d \circ (FA_4 \otimes FB_1) + \perp \circ (FA_1 \otimes FB_3) +$$

$$+ a \circ (FA_1 \otimes FB_1) = d \circ P_5 + a \circ P_1.$$

Конечная синхронная композиция:

$$P_1 = FA_1 \otimes FB_1 = b \circ \perp + a \circ (FA_1 \otimes FB_1) + b \circ (FA_2 \otimes FB_1) =$$

$$= a \circ P_1 + b \circ P_3.$$

$$P_3 = (FA_2 \otimes FB_1) = c \circ (FA_3 \otimes FB_1) = c \circ P_4.$$

$$P_4 = (FA_3 \otimes FB_1) = d \circ (FA_4 \otimes FB_1) + a \circ (FA_1 \otimes FB_1) =$$

$$= d \circ P_5 + a \circ P_1.$$

$$P_5 = (FA_4 \otimes FB_1) = d \circ (FA_4 \otimes FB_1) + a \circ (FA_1 \otimes FB_1) =$$

$$= d \circ P_5 + a \circ P_1.$$

Результат. Проверяемое свойство φ не выполняется на M , поскольку в композиции $M \otimes S$ присутствуют не все уравнения из φ , а именно, нет уравнений F_{B2} и F_{B3} . Таким образом, исходное требование φ выполнимо на M .

Заключение

В статье предложен и подробно рассмотрен новый алгоритм верификации для метода Model Checking на базе нотации RLTL для верификации формул логики линейного времени. Данный алгоритм во многом схож с алгоритмом верификации формул LTL, поскольку использует аналогичные принципы и подход к решению задачи верификации. Однако он имеет особенность – позволяет сократить число подготовительных этапов, необходимых для верификации, тем самым способствуя повышению производительности метода.

Работа предложенного алгоритма подробно рассмотрена на конкретных примерах с описанием всех этапов и их теоретической базы. Авторы считают, что он может выступить в качестве более производительной альтернативы существующему, а RLTL-нотация может использоваться в качестве основной структуры данных, на базе которой будут задаваться как модели верифицируемых систем,

так и требования к ним. В целом же такой подход должен способствовать общему повышению быстродействия метода Model Checking за счет снижения сложности алгоритма верификации и использования единой структуры данных.

Литература

1. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем. СПб: БХВ-Петербург, 2010. 552 с.
2. Кораблин Ю.П. Семантика языков распределенного программирования; [под ред. В.П. Кутепова]. М.: Изд-во МЭИ, 1996. 102 с.
3. Kroger F., Merz S. Temporal logic and state systems. Springer, 2008, 436 p.
4. Manna Z., Pnueli A. The temporal logic of reactive and concurrent systems: specification. Springer, 1992, 427 p.
5. Кларк Э.М., Грамберг О., Пелед Д. Верификация моделей программ. Model Checking. М.: Изд-во МЦНМО, 2002. 416 с.
6. Gerard J. Holzman. An analysis of bitstate hashing. Proc. 15th Int. Conf. on Protocol Specification, Testing, and Verification, 1998, pp. 301–314.
7. Ernst-Rudiger Olderog, Krzysztof R. Apt. Fairness in parallel programs: the transformational approach. ACM Trans. Program. Lang. Syst., 1988, vol. 10, no. 3, pp. 420–455.
8. Шипов А.А., Кораблин Ю.П. Построение моделей систем на базе эквациональной характеристики формул LTL // Программные продукты и системы. 2017. № 1. С. 61–66.
9. Шипов А.А., Кораблин Ю.П. Эквационная характеристика формул LTL // Программные продукты и системы. 2015. № 4. С. 175–179.

Software & Systems

DOI: 10.15827/0236-235X.119.456-460

Received 05.06.17

2017, vol. 30, no. 3, pp. 456–460

SYSTEM MODEL VERIFICATION BASED ON EQUATIONAL CHARACTERISTICS OF LTL FORMULAS

Yu.P. Korablin¹, Dr.Sc. (Engineering), Professor, y.p.k@mail.ru

A.S. Kochergin¹, Postgraduate Student, kocherginaalexandr@gmail.com

A.A. Shipov¹, Ph.D (Engineering), a-j-a-1@yandex.ru

¹ Russian State Social University, Vilgelm Pik St. 4, Moscow, 129226, Russian Federation

Abstract. For a long time, the Model Checking method has been widely used in the field related with software and technical systems quality evaluation. Such top IT companies as Intel, Microsoft, Amazon, etc. actively use it in the processes of development and maintenance of their products. Such a success of this method is certainly not accidental. It helped to solve a lot of problems in the field of verification, namely: the problems of unified representation of software and technical systems, the problem of formal requirements representation, the automation of verification phases, the verification of large distributed software systems, etc.

However, such challenges as the continuous development of modern technologies and the growth rates of modern software systems complexity can become an insurmountable obstacle for effective verification with Model Checking. Therefore, it is necessary to make permanent improvements of its theory and tools.

The article demonstrates a new verification algorithm of Linear Temporal Logic formulas by means of Model Checking based on the new RLTL notation (Recursive Linear Temporal Logic), which is a recursive representation of LTL formulas. This algorithm can avoid necessary conversions into Büchi automata of the system model and verifying statements since RLTL can be used to define both of them. This allows beginning a verification process immediately and increases Model Checking efficiency.

Keywords: verification, Model Checking, RLTL the equational characteristics of LTL, Kripke structure, Buchi automaton, temporal logic formula, LTL, CTL.

References

1. Karpov Yu.G. Model Checking. Verifikatsiya paralelnykh i raspredelennykh programmnykh sistem [Model Checking. Verification of Parallel and Distributed Software Systems]. BHV-Peterburg Publ., 2010, 552 p.
2. Korablin Yu.P. Semantika yazykov raspredelennogo programmirovaniya [Semantics of Distributed Programming Languages]. V.P. Kutepov (Ed.). Moscow, MEI Publ., 1996, 102 p.
3. Kroger F., Merz S. Temporal Logic and State Systems. Springer Publ., 2008, 436 p.
4. Manna Z., Pnueli A. The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer Publ., 1992, 427 p.
5. Klark E.M., Gramberg O., Peled D. Verifikatsiya modeley programm. Model Checking [Verification of Program Models. Model Checking]. MTsNMO Publ., 2002, 416 p.
6. Holzman G.J. An Analysis of Bitstate Hashing. Proc. 15th Int. Conf. Protocol Specification, Testing, and Verification. 1998, pp. 301–314.
7. Ernst-Rudiger Olderog, Krzysztof R. Apt. Fairness in Parallel Programs: The Transformational Approach. ACM Trans. on Programming Languages and Systems. 1988, vol. 10, no. 3, pp. 420–455.
8. Shipov A.A., Korablin Yu.P. System models construction based on LTL formula equational characteristics. Programmnye produkty i sistemy [Software & Systems]. 2017, no. 1, pp. 61–66 (in Russ.).
9. Shipov A.A., Korablin Yu.P. Equational characteristics of LTL formulas. Programmnye produkty i sistemy [Software & Systems]. 2015, no. 4 (112), pp. 175–179 (in Russ.).