УДК 519.767.2

DOI: 10.15827/0236-235X.117.061-066

Дата подачи статьи: 13.12.16

2017. Т. 30. № 1. С. 61–66

ПОСТРОЕНИЕ МОДЕЛЕЙ СИСТЕМ НА БАЗЕ ЭКВАЦИОНАЛЬНОЙ ХАРАКТЕРИСТИКИ ФОРМУЛ LTL

Ю.П. Кораблин, д.т.н., профессор, y.p.k@mail.ru
(Российский государственный социальный университет,
ул. Вильгельма Пика, 4, г. Москва, 129256, Россия);
А.А. Шипов, ведущий инженер, a-j-a-1@yandex.ru
(Московский технологический университет (МИРЭА),
просп. Вернадского, 78, г. Москва, 119454, Россия)

Задача верификации как программных, так и технических систем всегда была и остается одной из самых значимых с момента появления первых вычислительных устройств. Сегодня уже существует достаточно большое количество подходов к решению данной проблемы. Однако именно развитие такого метода формальной верификации, как Model Checking, позволило решить проблему представления верифицируемых систем и унифицировать процесс верификации программных и технических систем. Идея, лежащая в его основе, состоит в приведении исходной системы к некоторой унифицированной форме, то есть для выполнения верификации требуется лишь наличие модели, которая максимально точно описывала бы ее поведение.

В данной статье рассматривается возможность построения моделей систем с помощью RLTL-нотации (Recursive Linear Temporal Logic), которая является рекурсивным представлением формул логики линейного времени. Однако ее использование не ограничивается только этим аспектом.

Преимущество от использования для этих целей именно RLTL состоит в том, что модели, заданные с ее помощью, могут быть верифицированы относительно требований, заданных на основе RLTL, без перехода к другим структурам данных, что, безусловно, будет способствовать упрощению и повышению быстродействия процесса верификации. Кроме того, в работе предложены формальные средства, позволяющие во многих случаях существенно упростить модель, построенную на основе RLTL, за счет сокращения числа ее состояний и переходов.

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

Верификация работы программных и технических систем является неотъемлемой частью их жизненного цикла. Существует целый ряд инструментов и средств верификации, которые являются более удобными и эффективными при тех или иных условиях, на тех или иных этапах проектирования и сопровождения. Однако за последние два десятилетия лучше всего себя зарекомендовали инструменты формальной верификации, работа которых может быть автоматизирована, а корректность функционирования – доказана формально. Наиболее активно используемым инструментом на сегодняшний день является метод формальной верификации на моделях, или Model Checking [1, 2].

Технология процесса верификации программных и технических систем методом проверки на моделях подразумевает создание моделей систем, свойства которых требуется проверить. Так, например, модель системы может быть задана с помощью различных структур данных, в частности таких, как структура Крипке, язык асинхронных процессов Promela, в виде конечного автомата и т.д. Однако независимо от представления модели системы и верифицируемых свойств, заданных на базе логики линейного времени (LTL) [3, 4], необходимо, как правило, процесс их преобразования к автоматам Бюхи. При этом алгоритмы преобразования формул LTL в автоматы Бюхи являются нетривиальными, а принципы их функционирования непонятны на интуитивном уровне.

В данной статье рассматривается новая нотация RLTL (recursive linear temporal logic) для представления формул LTL в виде системы рекурсивных уравнений с целью расширения ее описательной мощности [5]. Кроме того, RLTL может быть использована и для задания моделей верифицируемых систем, что предоставляет следующие важные преимущества:

- задание как модели самой системы, так и проверяемых относительно нее свойств на базе единой структуры данных;
- возможность быстрого и интуитивно понятного преобразования RLTL-структур в автоматы Бюхи для их последующей верификации;
- возможность выполнения верификации без преобразования к автомату Бюхи;
- ускорение процесса верификации, что крайне важно при верификации больших распределенных программных систем.

RLTL-нотация

Рассмотрим рекурсивные представления основных базовых операторов LTL (табл. 1).

Символом $\circ$ обозначен оператор продолжения (конкатенация выражений), что позволяет использовать оператор X в неявном виде, а также упростить его восприятие, в частности, запись $\varphi_1 \circ \varphi_2$ будет пониматься как « φ_2 следует за φ_1 ». Под символом Δ , далее именуемым предикатом неопреде-

ленности, будем понимать некоторое неопределенное подмножество символов входного алфавита. Под отрицанием предиката неопределенности будем понимать некоторое другое неопределенное подмножество символов алфавита, а под отрицанием отрицания – некоторое третье неопределенное подмножество. Таким образом, отрицание предиката неопределенности является неполным, а каждое последующее отрицание дает неопределенное подмножество символов. Каждое из этих подмножеств ввиду своей неопределенности также может быть обозначено через Δ .

Таблица 1

Рекурсивные представления операторов LTL

Table 1

Recursive presentations of LTL operators

LTL	RLTL
$F\phi = \phi \vee XF\phi$	$F\phi = \phi \vee \Delta \circ F\phi$
$G\phi = \phi \wedge XG\phi$	$G\phi = \phi \circ G\phi$
$\phi_1 U \phi_2 = \phi_2 \vee \phi_1 \wedge X(\phi_1 U \phi_2)$	$U(\phi_1, \phi_2) = \phi_2 \vee \phi_1 \circ U(\phi_1, \phi_2)$
$\phi_1 \wedge X\phi_2$	$\phi_1 \circ \phi_2$
$X\phi$	$\Delta \circ \phi$

В таблице 1 представлены почти все операторы, используемые в LTL, кроме оператора $X\phi$, у которого отсутствует рекурсивная форма. Таким образом, поскольку все основные операторы LTL могут быть представлены в рекурсивной форме, авторы полагают, что приведение их к некоторому унифицированному виду, который, следуя [5], будем именовать эквациональной характеристикой, позволит существенно расширить описательную способность LTL и упростит процесс верификации ее формул.

Сочетания операторов LTL также могут быть выведены с помощью аксиом в рамках эквациональной характеристики. В таблице 2 приведены некоторые сочетания операторов LTL.

Таблица 2

Сочетания операторов LTL в эквациональной форме

Table 2

Equational combinations of LTL operators

LTL	RLTL
$FG\phi$	$F' = \phi \circ G' \vee \Delta \circ F'$ $G' = \phi \circ G'$
$GF\phi$	$G' = \phi \circ G' \vee \Delta \circ F'$ $F' = \phi \circ G' \vee \Delta \circ F'$

Сформулируем основные аксиомы логики RLTL:

A1.	$FF\phi = F\phi$	(FF)
A2.	$GG\phi = G\phi$	(GG)
A3.	$\neg F\phi = G\neg\phi$	($\neg F$)
A4.	$\neg G\phi = F\neg\phi$	($\neg G$)
A5.	$G\phi = \phi \circ G\phi$	(G)
A6.	$F\phi = \phi \vee \Delta \circ F\phi$	(F)
A7.	$G\phi \vee \neg G\phi = \Delta$	($\vee G$)
A8.	$F\phi \vee \neg F\phi = \Delta$	($\vee F$)
A9.	$G\phi_1 \circ \phi_2 = G\phi_1$	(G $\circ$)
A10.	$F\phi_1 \circ \phi_2 = F(\phi_1 \circ \phi_2)$	(F $\circ$)

A11.	$U(\phi_1, \phi_2) = \phi_2 \vee \phi_1 \circ U(\phi_1, \phi_2)$	(U)
A12.	$\neg(\phi \circ \phi_1) = \neg\phi \vee \Delta \circ \neg\phi_1$	($\neg\phi \circ$)
A13.	$\neg\Delta \rightarrow \Delta$	($\neg\Delta$)
A14.	$\Delta \wedge \phi = \phi$	($\Delta \wedge$)
A15.	$\phi = \phi \circ \Delta$	($\phi \Delta$)

Построение RLTL-моделей

Структуры на базе RLTL-нотации могут быть интерпретированы в качестве недетерминированного конечного автомата, который на вход получает бесконечные цепочки и моделирует поведение реагирующих систем. Так, рассмотрим формальное определение автомата Бюхи. Автомат Бюхи – это пятерка $(S, \Sigma, S_0, \delta, F)$, где Σ – конечное множество символов (алфавит); S – множество состояний; $S_0 \subseteq S$ – множество начальных состояний (содержащее только один элемент для детерминированного случая); $\delta: S \times \Sigma \rightarrow 2^S$ – отношение переходов (для детерминированного автомата $\delta: S \times \Sigma \rightarrow S$); $F \subseteq S$ – множество допускающих (финальных) состояний.

Нетрудно убедиться, что любая структура на базе RLTL-нотации может быть интерпретирована в качестве автомата Бюхи в соответствии с данным определением. В качестве алфавита в RLTL также выступает некоторое множество символов Σ . В качестве состояний выступает множество метапеременных $S = \{F_1, F_2, \dots, F_n\}$, задающих уравнения системы. По умолчанию в качестве начального состояния выступает метапеременная первого уравнения системы, однако, если существует несколько начальных состояний, их можно задать в виде элементов соответствующего множества. Функция переходов для RLTL остается аналогичной автоматам Бюхи, где в роли переходов выступает оператор продолжения $\circ$, за которым следует метапеременная, обозначающая состояние, в которое осуществляется переход. Что касается множества допустимых или финальных состояний, то в RLTL не существует аналога, поэтому необходимо явное задание множества соответствующих метапеременных.

Для задания с помощью RLTL-нотации моделей систем достаточно использовать лишь тройку значений (S, Σ, δ) , поскольку каждое состояние модели системы является допустимым, а любое множество начальных состояний из более чем одного элемента может быть заменено единственным начальным состоянием, из которого существуют переходы по любому из символов алфавита в каждое из этих состояний. В качестве примера зададим в терминах RLTL простую модель работы светофора на основе ее устного описания: «Горит зеленый сигнал, затем загорается желтый, а после желтого – красный. После красного сигнала снова загорается желтый, за ним зеленый. В такой последовательности сигналы продолжают чередоваться дальше». Опишем модель работы светофора следующей тройкой значений (S, Σ, δ) : $S = \{F_1, F_2, F_3, F_4\}$, $\Sigma = \{g, y, r\}$, $\delta = \{g: F_1 \rightarrow F_2, y: F_2 \rightarrow F_3, r: F_3 \rightarrow F_4, y: F_4 \rightarrow F_1\}$, где g – зеленый сигнал светофора; y – жел-

тый сигнал; r – красный сигнал. В RLTL-нотации модель будет иметь следующий вид: $F_1 = g \circ F_2$, $F_2 = y \circ F_3$, $F_3 = r \circ F_4$, $F_4 = y \circ F_1$.

Использование пятерки значений ($S, \Sigma, S_0, \delta, F$) требуется лишь в тех случаях, когда необходимо превратить данные на базе RLTL в автомат Бюхи. Так, пусть дана формула LTL $GF\phi$ – «всегда в будущем будет наступать ϕ ». В терминах RLTL-нотации она будет выглядеть следующим образом: $F_1 = \phi \circ F_1 \vee \Delta \circ F_2$, $F_2 = \phi \circ F_1 \vee \Delta \circ F_2$.

В данном случае система задана тройкой значений (S, Σ, δ), где $S = \{F_1, F_2\}$, $\Sigma = \{\phi\}$, $\delta = \{\phi: F_1 \rightarrow F_1, \phi: F_2 \rightarrow F_1, \Delta: F_1 \rightarrow F_2, \Delta: F_2 \rightarrow F_2\}$. Для превращения данной формулы в автомат Бюхи достаточно указать множество начальных и множество допустимых состояний: $S_0 = \{F_1\}$ и $F = \{F_1\}$. Аналогично может быть задана и модель любой системы на базе RLTL.

Таким образом, любая RLTL-структура, использующая тройку значений (S, Σ, δ), может быть превращена в автомат Бюхи за счет задания множеств начальных и допустимых состояний. Возможность простого и интуитивно понятного превращения формул RLTL в автоматы Бюхи и наоборот, как и представленные выше примеры, подтверждает возможность задания моделей верифицируемых систем сразу на базе RLTL. Сам процесс задания моделей систем может быть выполнен двумя способами:

- построение модели системы на базе структуры Крипке с ее последующим преобразованием в RLTL; поскольку структуры Крипке могут быть быстро преобразованы в автоматы Бюхи [1], их преобразование в RLTL-формулы также выполняется относительно просто;
- непосредственное построение моделей систем сразу на базе RLTL-нотации, что является более приемлемым, поскольку дальнейшие преобразования для их верификации не требуются.

Абстракция и унификация RLTL-моделей

Для повышения быстродействия процесса верификации больших распределенных систем существует ряд специальных инструментов, средств и алгоритмов:

- редукция частичных порядков [6];
- абстракция данных [6];
- хэширование битовых состояний (вместо полных состояний хранится лишь их хэш, что снижает требования к объему памяти, но уменьшает полноту системы) [7];
- ускорение проверки «корректности в слабом смысле», или Weak Fairness [8].

Использование данных средств позволяет снизить проблему комбинаторного взрыва, то есть быстрого роста числа состояний.

Для упрощения моделей систем введем операцию абстракции и унификации. Операция абстрак-

ции позволяет осуществлять сокращение числа свойств моделей систем, а операция унификации – объединение однотипных элементов. Операцию абстракции обозначим как ABS_H^α , где H – множество заменяемых символов; α – заменяющий символ (тот символ, на который заменяются элементы из множества H); ϕ – атомарный предикат (символ алфавита); μ_i – формула. Аксиомы, описывающие свойства данной операции, представлены далее:

$$AB1 \quad ABS_H^\alpha(\phi) = \phi, \text{ если } \phi \notin H$$

$$AB2 \quad ABS_H^\alpha(\phi) = \alpha, \text{ если } \phi \in H$$

$$AB3 \quad ABS_H^\alpha(\mu_1 \circ \mu_2) = ABS_H^\alpha(\mu_1) \circ ABS_H^\alpha(\mu_2)$$

$$AB4 \quad ABS_H^\alpha(\mu_1 \vee \mu_2) = ABS_H^\alpha(\mu_1) \vee ABS_H^\alpha(\mu_2)$$

$$AB5 \quad ABS_H^\alpha(\mu_1 \wedge \mu_2) = ABS_H^\alpha(\mu_1) \wedge ABS_H^\alpha(\mu_2)$$

Операция унификации базируется на четырех основных правилах:

$$UN1 \quad \begin{aligned} F_i = \phi_1 \circ F_j \\ F_j = \phi_1 \circ F_k \end{aligned} \equiv F_N = \phi_1 \circ F_k, \text{ где } N = \{i, j\}$$

$$UN2 \quad F_N = \phi \circ F_k \equiv F_N = \phi \circ F_N, \text{ где } N = \{i_1, i_2, \dots, i_n\}, k \in N$$

$$UN3 \quad F_N = \phi \circ F_k \equiv F_N = \phi \circ F_k, \text{ где } N = \{i_1, i_2, \dots, i_n\}, k \notin N$$

$$UN4 \quad F_m = \phi \circ F_k \equiv F_m = \phi \circ F_N, \text{ где } N = \{i_1, i_2, \dots, i_n\}, k \in N, m \notin N$$

Их использование позволяет сократить число состояний системы, а также число ее переходов, где N – множество унифицируемых индексов.

В качестве примера возьмем некоторый вариант модели алгоритма «читатели-писатели» для трех «писателей» и двух «читателей». Модель соответствующей системы представлена в виде структуры Крипке на рисунке.

Сформулируем модель данной системы в терминах RLTL-нотации. Каждому состоянию модели Крипке будет сопоставлена своя метаварiable, определяющая соответствующее уравнение системы. В качестве алфавита будет выступать $\Sigma = \{W_1, W_2, W_3, R_1, R_2, R_{12}\}$. Переходы, ведущие из конкретного состояния, будут помечены символом этого состояния. Поскольку первое состояние си-

стемы не помечено ни одним символом, в RLTL все переходы из данного состояния будут помечены символом неопределенности, или Δ .

$$\begin{aligned} F_M &= \Delta \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{W1} &= W_1 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{W2} &= W_2 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{W3} &= W_3 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{R1} &= R_1 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{R2} &= R_2 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{R12} &= R_{12} \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \end{aligned}$$

Важно учитывать, относительно чего планируется производить упрощение системы. Так, в данном случае пусть проверка системы будет выполняться относительно следующего условия, заданного в терминах LTL: $\phi = GF(W_i)$ – «Всегда в будущем процесс записи для любого из «писателей» будет обязательно завершен». Как видно из формулы, в процессе верификации не будет анализироваться ни один из «читателей», так как в формуле не участвуют предикаты $R1$ и $R2$. В связи с этим можно выполнить абстракцию модели системы относительно всех «читателей».

Применим представленные аксиомы к описанному выше примеру. Поскольку абстракция в этом случае выполняется относительно предикатов $R1$ и $R2$, $H = \{R1, R2, R12\}$, а в качестве символа замены будет использован некоторый символ, например $\alpha = R$. Таким образом, получим:

$$\begin{aligned} \text{ABS}_H^R(F_M) &= F_M = \\ &= \text{ABS}_H^R(\Delta \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12})) \\ \text{ABS}_H^R(F_{W1}) &= F_{W1} = \\ &= \text{ABS}_H^R(W_1 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12})) \\ \text{ABS}_H^R(F_{W2}) &= F_{W2} = \\ &= \text{ABS}_H^R(W_2 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12})) \\ \text{ABS}_H^R(F_{W3}) &= F_{W3} = \\ &= \text{ABS}_H^R(W_3 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12})) \\ \text{ABS}_H^R(F_{R1}) &= F_{R1} = \\ &= \text{ABS}_H^R(R_1 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12})) \\ \text{ABS}_H^R(F_{R2}) &= F_{R2} = \\ &= \text{ABS}_H^R(R_2 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12})) \\ \text{ABS}_H^R(F_{R12}) &= F_{R12} = \\ &= \text{ABS}_H^R(R_{12} \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12})) \end{aligned}$$

После применения аксиом AB1–AB5 система уравнений примет следующий вид:

$$\begin{aligned} F_M &= \Delta \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{W1} &= W_1 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{W2} &= W_2 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{W3} &= W_3 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{R1} &= R \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{R2} &= R \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{R12} &= R \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \end{aligned}$$

Применение представленных выше аксиом позволило сократить общее число символов алфавита (предикатов) системы, однако число ее состояний и переходов осталось прежним. Правило UN1 позволяет сократить число состояний модели верифицируемой системы путем унификации однотипных элементов вычислительного пути. Применим данное правило к системе, полученной на предыдущем шаге, где $N = \{R1, R2, R12\}$:

$$\begin{aligned} F_M &= \Delta \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{W1} &= W_1 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{W2} &= W_2 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_{W3} &= W_3 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \\ F_N &= R \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_{R1} \vee F_{R2} \vee F_{R12}) \end{aligned}$$

Применение правил UN2–UN4 позволяет унифицировать однотипные переходы RLTL-модели. После применения данных правил модель примет вид:

$$\begin{aligned} F_M &= \Delta \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_N \vee F_N \vee F_N) \\ F_{W1} &= W_1 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_N \vee F_N \vee F_N) \\ F_{W2} &= W_2 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_N \vee F_N \vee F_N) \\ F_{W3} &= W_3 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_N \vee F_N \vee F_N) \\ F_N &= R \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_N \vee F_N \vee F_N) \end{aligned}$$

Выполнив правило дизъюнкции, получим упрощенную модель системы «читатели-писатели»:

$$\begin{aligned} F_M &= \Delta \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_N) \\ F_{W1} &= W_1 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_N) \\ F_{W2} &= W_2 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_N) \\ F_{W3} &= W_3 \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_N) \\ F_N &= R \circ (F_{W1} \vee F_{W2} \vee F_{W3} \vee F_N) \end{aligned}$$

Таким образом, на конкретном примере за счет эффективного использования операций абстракции и унификации на RLTL-структурах исходную модель удалось сократить более чем в два раза. Исходная модель – 42 перехода, упрощенная модель – 20 переходов.

Приведем также пример упрощения модели светофора, описанной ранее в терминах RLTL. Пусть проверка этой системы будет выполняться относительно следующего условия, заданного в терминах LTL: $\phi = GF(g)$ – «Всегда в будущем будет загораться зеленый сигнал светофора». Тогда в качестве множества заменяемых символов можно взять $N = \{y, r\}$, а в качестве символа замены используем просто символ s , который будет означать stop, $\alpha = s$. Таким образом, получим:

$$\begin{aligned} \text{ABS}_H^s(F_1) &= \text{ABS}_H^s(g \circ F_2) \\ \text{ABS}_H^s(F_2) &= \text{ABS}_H^s(y \circ F_3) \\ \text{ABS}_H^s(F_3) &= \text{ABS}_H^s(r \circ F_4) \\ \text{ABS}_H^s(F_4) &= \text{ABS}_H^s(y \circ F_1) \end{aligned}$$

После применения аксиом AB1–AB5 система уравнений примет следующий вид:

$$\begin{aligned} F_1 &= g \circ F_2 \\ F_2 &= s \circ F_3 \\ F_3 &= s \circ F_4 \\ F_4 &= s \circ F_1 \end{aligned}$$

Для сокращения числа состояний системы применим к ней правило UN1, где $N = \{2, 3, 4\}$, и получим: $F_1 = g \circ F_2$, $F_N = s \circ F_1$.

И наконец, применив правила UN2–UN4, получим сокращенную модель светофора: $F_1 = g \circ F_N$, $F_N = s \circ F_1$.

В результате после выполнения операций абстракции и унификации модель светофора сократилась ровно в два раза.

В качестве более содержательного примера использования операций абстракции и унификации рассмотрим следующий случай. Пусть требуется выполнить верификацию множества драйверов и/или некоторых подпрограмм операционной системы, которые используют разделяемые ресурсы, например, оборудование или адреса в оперативной памяти, на предмет наличия в их совместной работе тупиков, гонок и взаимных блокировок. В таком случае последовательность шагов будет следующей.

1. Построить модель для каждого драйвера (подпрограммы) на базе RLTL.

2. Построить верифицируемые свойства на базе RLTL.

3. Для каждой модели выполнить операцию абстракции:

- определить множество H как множество символов, соответствующих свойствам системы, отличным от свойств, задействованных при работе с разделяемыми ресурсами;

- в качестве символа замены α взять любой не использованный ранее символ;

- выполнить абстракцию с помощью аксиом AB1–AB5.

4. Для каждой модели выполнить операцию унификации с помощью аксиом UN1–UN4.

5. Выполнить верификацию абстрагированных и унифицированных моделей систем.

За счет выполнения пункта 3 число свойств каждой модели сократится до числа тех свойств, которые сопоставляются операциям взаимодействия, плюс одно свойство, обозначаемое символом α и характеризующее все отличные от операций взаимодействия свойства. Выполнение пункта 4 позволит на порядок и более сократить размеры каждой модели за счет объединения однотипных блоков состояний, полученных после применения

пункта 3. Таким образом, за счет того, что общая доля операций взаимодействия относительно всех операций драйвера крайне мала, в данном примере удастся добиться многократного сокращения размеров верифицируемых моделей.

Заключение

Авторами была рассмотрена RLTL-нотация в качестве новой структуры данных для задания моделей систем, что позволяет повысить эффективность и быстродействие процесса верификации за счет сокращения числа свойств систем, а также числа состояний и переходов. Было показано, что задание моделей систем с помощью RLTL не сложнее, чем на основе структур Крипке или на базе автоматов Бюхи. Модели, заданные на базе RLTL, как и модели на базе других структур данных, могут быть существенно упрощены за счет использования представленных в статье методов абстракции и унификации, что является важным аспектом для процесса верификации больших распределенных программных систем.

Таким образом, использование RLTL-нотации является важным и необходимым шагом, поскольку задание моделей верифицируемых систем и их требований сразу на базе единой унифицированной структуры позволяет сделать процесс верификации более гибким, быстрым и удобным.

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. 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. Springer-Verlag, NY, 1992, 427 p.
5. Шипов А.А., Кораблин Ю.П. Эквационная характеристика формул LTL // Программные продукты и системы. 2015. № 4. С. 175–179.
6. Кларк Э.М., Грамберг О., Пелед Д. Верификация моделей программ. Model Checking; [пер. с англ.]. М.: Изд-во МЦНМО, 2002. 416 с.
7. Holzman G.J. An Analysis of Bitstate Hashing. Proc. 15th Int. Conf. on Protocol Specification, Testing, and Verification, 1998, pp. 301–314.
8. Olderog E.-R., Apt K.R. Fairness in Parallel Programs: The Transformational Approach. ACM Transactions on Programming Languages and Systems, July 1988, vol. 10, no. 3, pp. 420–455.

SYSTEM MODELS CONSTRUCTION BASED ON LTL FORMULA EQUATIONAL CHARACTERISTICS

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

A.A. Shipov², Leading Engineer, a-j-a-1@yandex.ru

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

² Moscow Technological University (MIREA), Vernadsky Ave. 78, Moscow, 119454, Russian Federation

Abstract. Verification of software and technical systems has always been and still remains one of the most significant tasks since appearance of first computing devices. Today there are quite a lot of approaches to solve this problem. However, development of such formal verification method as Model Checking helped to solve the problem of verifying systems representation and to unify verification process for software and technical systems. Its main idea is to transform an original system into a unified form. It means that a verification process requires only a model that would most precisely describe system's behavior.

The article considers the possibility of system model construction using RLTL-notations (Recursive Linear Temporal Logic), which are a recursive representation of formulas of linear temporal logic. However, its usage is not limited to this aspect. The advantage of using RLTL for these purposes is that models based on it might be verified in respect to requirements which are also RLTL-based without casting to any another data structure. It will certainly help to simplify and improve the performance of a verification process. Furthermore, the article describes the formal tools, which allow simplifying RLTL-based models in many cases reducing the number of their states and transitions.

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

References

1. Karpov Yu.G. *Model Checking. Verifikatsiya paralelnykh i raspredelennykh programmnykh sistem* [Model Checking. Parallel and Distributed Software System Verification]. St. Petersburg, BHV-Peterburg Publ., 2010, 552 p.
2. Korablin Yu.P. *Semantika yazykov raspredelennogo programmirovaniya* [Distributed Programming Language Semantics]. Study guide. 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-Verlag, NY, 1992, 427 p.
5. Shipov A.A., Korablin Yu.P. LTL formula equational characteristics. *Programmnye produkty i sistemy* [Software & Systems]. 2015, no. 4 (112), pp. 175–179.
6. Clarke E.M., Grumberg O., Peled D. *Model Checking*. MIT Press, 1999, 330 p. (Russ.ed.: Moscow, MTsNMO Publ., 2002, 416 p.).
7. Holzman G.J. An Analysis of Bitstate Hashing. *Proc. 15th Int. Conf. Protocol Specification, Testing, and Verification*. 1998, pp. 301–314.
8. Olderog E.-R., Apt K.R. Fairness in Parallel Programs: The Transformational Approach. *ACM Trans. Programming Languages and Systems*. 1988, vol. 10, no. 3, pp. 420–455.

Примеры библиографического описания статьи

1. Кораблин Ю.П., Шипов А.А. Построение моделей систем на базе эквациональной характеристики формул LTL // Программные продукты и системы. 2017. Т. 30. № 1. С. 61–66; DOI: 10.15827/0236-235X.117.061-066.
2. Korablin Yu.P., Shipov A.A. System models construction based on LTL formula equational characteristics. *Programmnye produkty i sistemy* [Software & Systems]. 2017, vol. 30, no. 1, pp. 61–66 (in Russ.); DOI: 10.15827/0236-235X.117.061-066.