MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 29, NO. 1, 2022
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

LTL-Specification of Bounded Counter Machines

E. V. Kuzmin[1]

DOI: 10.18255/1818-1015-2022-1-44-59

[1]P. G. Demidov Yaroslavl State University, 14 Sovetskaya str., Yaroslavl 150003, Russia.

MSC2020: 68Q60, 68N30, 68Q04, 03B44, 03B70, 03D10
Research article
Full text in Russian

Received January 11, 2022
After revision February 21, 2022
Accepted March 9, 2022

The article revises the results of the work devoted to the problem of representing the behaviour of a program system as a set of formulas of the linear temporal logic LTL, followed by the use of this representation to verify the satisfiability of the program system properties through the procedure of proving the validity of logical inferences, expressed in terms of the LTL logic. The LTL logic is applied to bounded Minsky counter machines that are considered as program systems of which we need to get the specification of its behaviour. Earlier, when working with the temporal logic LTL as a program logic, a special pseudo-operator was actually introduced to refer to the previous values of variables involved in elementary propositions. Despite the fact that this pseudo-operator is easily implemented in the Cadence SMV verifier when proving the validity of logical LTL-inferences, the classical definition of the LTL logic does not imply its presence. In this article, only binary variables will be used to build an LTL-specification for the behaviour of a bounded counter machine, and tracking of previous values of these variables will be carried out exclusively within the LTL logic itself through the appropriate formulas.

Keywords: non-classical logic; linear temporal logic; counter machines; LTL-specification

INFORMATION ABOUT THE AUTHORS

Egor V. Kuzmin (correspondence author) | orcid.org/0000-0003-0500-306X. E-mail: kuzmin@uniyar.ac.ru
Professor, Doctor of Science.

Funding: This work was supported by P. G. Demidov Yaroslavl State University Project № VIP-016.

For citation: E. V. Kuzmin, “LTL-Specification of Bounded Counter Machines”, *Modeling and analysis of information systems*, vol. 29, no. 1, pp. 44-59, 2022.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 29, № 1, 2022
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

LTL-спецификация ограниченных счётчиковых машин

Е. В. Кузьмин[1]

DOI: 10.18255/1818-1015-2022-1-44-59

[1]Ярославский государственный университет им. П. Г. Демидова, ул. Советская, д. 14, г. Ярославль, 150003 Россия.

УДК 519.7
Научная статья
Полный текст на русском языке

Получена 11 января 2022 г.
После доработки 21 февраля 2022 г.
Принята к публикации 9 марта 2022 г.

В статье пересматриваются результаты работы, посвящённой задаче представления поведения программной системы в виде набора формул линейной темпоральной логики LTL с последующим использованием этого представления для проверки выполнимости программных свойств системы через процедуру доказательства справедливости логических выводов, выраженных в терминах логики LTL. В качестве программных систем, для спецификации поведения которых применяется логика LTL, рассматриваются счётчиковые машины Минского с ограничениями. Ранее при работе с темпоральной логикой LTL как с программной логикой фактически был введён специальный псевдооператор обращения к предыдущим значениям переменных, задействованных в элементарных высказываниях. Несмотря на то что этот псевдооператор легко реализуется в верификаторе Cadence SMV при доказательстве справедливости логических LTL-выводов, классическое определение логики LTL не предполагает его наличия. В данной статье для построения LTL-спецификации поведения ограниченной счётчиковой машины будут использоваться только бинарные переменные, а отслеживание их предыдущих значений будет осуществляться исключительно в рамках самой логики LTL посредством соответствующих формул.

Ключевые слова: неклассическая логика; линейная темпоральная логика; счётчиковые машины; LTL-спецификация

ИНФОРМАЦИЯ ОБ АВТОРАХ

Егор Владимирович Кузьмин
автор для корреспонденции

orcid.org/0000-0003-0500-306X. E-mail: kuzmin@uniyar.ac.ru
профессор, доктор физ.-мат. наук.

Финансирование: Работа выполнена в рамках инициативной НИР ЯрГУ им. П. Г. Демидова № VIP-016.

Для цитирования: E. V. Kuzmin, “LTL-Specification of Bounded Counter Machines”, *Modeling and analysis of information systems*, vol. 29, no. 1, pp. 44-59, 2022.

Введение

В статье пересматриваются результаты работы [1], посвящённой задаче представления поведения программной системы в виде набора формул линейной темпоральной логики LTL с последующим использованием этого представления для проверки выполнимости программных свойств системы через процедуру доказательства справедливости логических выводов, выраженных в терминах логики LTL. В качестве программных систем, для спецификации поведения которых применяется логика LTL, рассматриваются счётчиковые машины Минского [2—4] с ограничениями.

Ранее в [1] при работе с темпоральной логикой LTL как с программной логикой фактически был введён специальный псевдооператор обращения к предыдущим значениям переменных, задействованных в элементарных высказываниях. Несмотря на то что этот псевдооператор легко реализуется в верификаторе Cadence SMV [5] при доказательстве справедливости логических LTL-выводов, классическое определение логики LTL не предполагает его наличия.

В данной статье для построения LTL-спецификации поведения ограниченной счётчиковой машины будут использоваться только бинарные переменные, а отслеживание их предыдущих значений будет осуществляться исключительно в рамках самой логики LTL посредством соответствующих формул.

1. Ограниченные счётчиковые машины

Ограниченная счётчиковая машина представляет собой счётчиковую машину Минского [2—4], дополненную ограничениями на ёмкости счётчиков. Более формально, *ограниченная счётчиковая машина* — это набор из шести элементов $(q_0, q_n, Q, V, \Delta, \mathrm{bnd})$, где $Q = \{q_0, \dots, q_n\}$ — конечное непустое множество состояний машины; $q_0 \in Q$ — начальное состояние; $q_n \in Q$ — финальное состояние; $X = \{x_1, \dots, x_m\}$ — конечное непустое множество счётчиков, которые могут принимать значения из $\mathbb{N} \cup \{0\}$; $\Delta = \{\delta_0, \dots, \delta_{n-1}\}$ — набор правил переходов по состояниям машины; δ_i — правило переходов для состояния q_i; $\mathrm{bnd}\colon X \longrightarrow \mathbb{N}$ — отображение, устанавливающее предельное значение $\mathrm{bnd}x \in \mathbb{N}$ для каждого счётчика $x \in X$. Состояния q_i, $0 \leqslant i \leqslant n-1$, подразделяются на два типа. Состояния первого типа имеют правила переходов вида:

$$(\delta_i)\ \ q_i:\ \ \text{if } x_j < \mathrm{bnd}x_j \text{ then } x_j := x_j + 1;\ \text{goto } q_k,$$

где $1 \leqslant j \leqslant m, 0 \leqslant k \leqslant n$. Для состояний второго типа имеем, $1 \leqslant j \leqslant m, 0 \leqslant k, l \leqslant n$:

$$(\delta_i)\ \ q_i:\ \ \text{if } x_j > 0 \text{ then } (x_j := x_j - 1;\ \text{goto } q_k) \text{ else goto } q_l.$$

Для финального состояния q_n правил переходов не предусмотрено. Это означает, что при попадании в состояние q_n ограниченная счётчиковая машина завершает свою работу. Если при срабатывании правила переходов первого типа оказывается, что значение соответствующего счётчика достигло предельного значения, работа машины также останавливается.

Конфигурация ограниченной счётчиковой машины — это набор $(q_i, c_1, \dots, c_m)$, где q_i — состояние машины, $c_1, \dots, c_m$ — натуральные числа (включая ноль), являющиеся значениями соответствующих счётчиков.

Рассмотрим множество состояний $Q = \{q_0, \dots, q_n\}$ как набор булевых переменных. Будем считать, что некоторая переменная $q_i \in Q$ принимает значение 1, если счётчиковая машина находится в состоянии q_i $(0 \leqslant i \leqslant n)$. В других случаях q_i будет иметь значение 0. Тогда конфигурацию сётчиковой машины можно представить в виде вектора значений соответствующего набора переменных

$$(q_0, \dots, q_n, x_1, \dots, x_m).$$

Исполнением счётчиковой машины называется последовательность конфигураций $s_0\ s_1\ s_2\ s_3\ s_4\ \dots$, индуктивно определяемая в соответствии с правилами переходов. Счётчиковая машина имеет одно

исполнение из начальной конфигурации s_0, так как для каждого состояния предусмотрено не более одного правила переходов. Машина, получив на вход некоторый набор значений счётчиков, стартует из состояния q_0 и либо останавливается в состоянии q_n с выходным набором значений счётчиков, либо прекращает работу в одном из нефинальных состояний из-за достижения счётчиком своего предельного значения, либо зацикливается, реализуя тем самым частичную числовую функцию.

Далее в статье под счётчиковой машиной будет пониматься именно ограниченная счётчиковая машина. При небольшом количестве счётчиков будем обозначать их буквами a, b, c и d.

2. Примеры ограниченных счётчиковых машин

Рассмотрим в качестве примеров счётчиковую машину умножения двух чисел и счётчиковую машину возведения числа в квадрат [3].

Ограниченная счётчиковая машина $4cM$ умножения двух натуральных чисел имеет семь состояний $q_0, q_1, \ldots, q_6$, где q_0 является начальным состоянием, q_6 — финальное состояние. Множество счётчиков $X = \{a, b, c, d\}$. В начальной конфигурации счётчики a и b получают значения n и m соответственно, а начальные значения остальных счётчиков c и d равны нулю. В финальной конфигурации результат вычисления будет содержаться в счётчике c при нулевых значениях счётчиков a и d. Счётчик b будет содержать исходное число m. На значения счётчиков накладываются ограничения bnd$a$, bnd$b$, bnd$c$ и bnd$d$. Правила переходов по состояниям машины $4cM$ представлены ниже:

(δ_0) q_0 : if $a > 0$ then ($a := a - 1$; goto q_1) else goto q_6;
(δ_1) q_1 : if $b > 0$ then ($b := b - 1$; goto q_2) else goto q_4;
(δ_2) q_2 : if $c <$ bnd$c$ then $c := c + 1$; goto q_3;
(δ_3) q_3 : if $d <$ bnd$d$ then $d := d + 1$; goto q_1;
(δ_4) q_4 : if $d > 0$ then ($d := d - 1$; goto q_5) else goto q_0;
(δ_5) q_5 : if $b <$ bnd$b$ then $b := b + 1$; goto q_4.

Четырёхсчётчиковая машина $4cM$ в графическом виде показана на рис. 1, где для переменной $x \in X$ обозначение «$x+$» соответствует увеличению счётчика на единицу с последующим переходом к новому состоянию машины вниз или по стрелке, а «$x-$» используется для обозначения условного вычитания единицы с переходом вниз при $x > 0$ или по правосторонней стрелке в случае нулевого значения счётчика x.

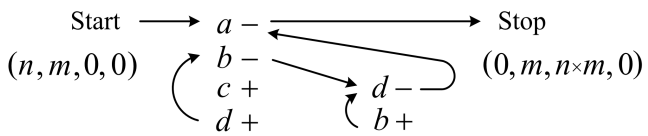

Fig. 1. Graphical representation of the counter machine $4cM$ of multiplying two numbers

Рис. 1. Графическое представление счётчиковой машины $4cM$ умножения двух чисел

Ограниченная счётчиковая машина $3cM$ возведения числа n в квадрат имеет восемь состояний $q_0, q_1, \ldots, q_7$, где q_0 — начальное состояние, q_7 — финальное состояние, и три счётчика a, b и c. В начальной конфигурации счётчик a получает значение n, а начальные значения двух других счётчиков b и c равны нулю. В финальной конфигурации результат вычисления будет содержаться в счётчике c при нулевых значениях остальных счётчиков a и b. На значения счётчиков накладываются ограничения bnd$a$, bnd$b$ и bnd$c$. Машина $3cM$ имеет следующие правила переходов:

(δ_0) q_0 : if $a > 0$ then $(a := a - 1;$ goto $q_1)$ else goto q_7;
(δ_1) q_1 : if $c <$ bnd$c$ then $c := c + 1;$ goto q_2;
(δ_2) q_2 : if $a > 0$ then $(a := a - 1;$ goto $q_3)$ else goto q_5;
(δ_3) q_3 : if $b <$ bnd$b$ then $b := b + 1;$ goto q_4;
(δ_4) q_4 : if $c <$ bnd$c$ then $c := c + 1;$ goto q_1;
(δ_5) q_5 : if $b > 0$ then $(b := b - 1;$ goto $q_6)$ else goto q_0;
(δ_6) q_6 : if $a <$ bnd$a$ then $a := a + 1;$ goto q_5.

Правила переходов счётчиковой машины $3cM$ в графическом виде представлены на рис. 2.

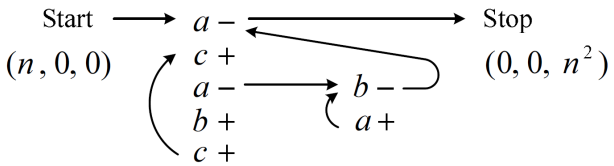

Fig. 2. Graphical representation of the counter machine $3cM$ of squaring a number

Рис. 2. Графическое представление счётчиковой машины $3cM$ возведения числа в квадрат

3. Линейная темпоральная логика

Линейная темпоральная логика LTL (linear temporal logic), или темпоральная логика линейного времени (linear-time temporal logic), представляет собой расширение классической логики высказываний с помощью модальных операторов, позволяющих учитывать временной аспект в последовательностях событий и объектов. Темпоральная логика LTL находит важное применение в области формальной верификации, где она используется для описания требований к аппаратным и программным системам [6]. В данной статье формализм логики LTL будет задействован для описания поведения счётчиковых машин, а также спецификации их свойств, подлежащих дальнейшему анализу на выполнимость методом проверки модели (model checking) [7, 8].

Дадим определение логики LTL, как некоторой абстрактной модальной темпоральной логики, являющейся одним из представителей внушительного ряда неклассических логик [9, 10].

Символами языка логики LTL являются пропозициональные переменные p, p_0, p_1, p_2, ... (элементарные высказывания), константы true и false, логические связки $\neg$ (отрицание), $\wedge$ (конъюнкция), $\vee$ (дизъюнкция), $\Longrightarrow$ (импликация), $\Longleftrightarrow$ (эквивалентность), темпоральные модальные операторы **X** (neXt, непосредственное следование), **U** (Until, условное ожидание), **F** (Future, неизбежность), **G** (Globally, инвариантность) и знаки пунктуации «(» и «)».

Формулами языка логики LTL являются те и только те строки символов, которые могут быть рекурсивно построены из пропозициональных переменных и констант по следующему правилу:

если φ и ψ — формулы, то выражения $\neg\varphi$, $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \Longrightarrow \psi)$, $(\varphi \Longleftrightarrow \psi)$, $\mathbf{X}\,\varphi$, $(\varphi\,\mathbf{U}\,\psi)$, $\mathbf{F}\,\varphi$ и $\mathbf{G}\,\varphi$ также являются формулами.

Значение (истинность или ложность) формул темпоральной логики LTL определяется через понятие интерпретации.

Интерпретация формул логики LTL представляет собой набор (w_0, W, T, v), где W обозначает некоторое непустое множество объектов, интуитивно понимаемое как множество *возможных миров*, $w_0 \in W$ — это начальный мир, а $T : W \longrightarrow W$ — функция переходов между мирами.

Если возможные миры w и w' принадлежат W и связаны функцией переходов $T(w) = w'$, то для простоты будем писать $w \longrightarrow w'$. Запись $w \longrightarrow w'$ читается как «мир w' непосредственно достижим из мира w» и означает, что существует прямой переход из мира w в мир w'.

Обозначим $w \xrightarrow{*} w'$ транзитивную достижимость мира w' из мира w за ноль или более шагов (применений функции T), т. е. $w \xrightarrow{*} w'$ означает, что $\exists\, i \in \mathbb{N} \cup \{0\}$: $T^i(w) = w'$, где $T^i(w) = T(T^{i-1}(w))$ и $T^0(w) = w$. Запись $w \xrightarrow{i} w'$ будет обозначать достижимость w' из w ровно за i шагов. В случае $i = 0$ считается, что мир w транзитивно достигает сам себя за ноль шагов, т. е. $w' = w$.

Функция ν сопоставляет истинное (возвращается результат 1) или ложное (результат 0) значение каждой паре вида (p, w), где p — пропозициональная переменная, а $w \in W$. Запишем это как $\nu_w(p) = 1$ и $\nu_w(p) = 0$ соответственно.

Интуитивно, $\nu_w(p) = 1$ означает, что в мире w высказывание p истинно, а $\nu_w(p) = 0$ — в мире w высказывание p ложно.

Функция ν_w для каждого мира $w \in W$ расширяется на все множество формул по рекурсивным правилам. Рекурсивные правила расширения функции ν_w для логических связок $\neg$, $\wedge$, $\vee$, $\Longrightarrow$ и $\Longleftrightarrow$ следующие. Для каждого мира $w \in W$ имеем:

$\nu_w(\neg\varphi) = 1$, если $\nu_w(\varphi) = 0$, иначе $\nu_w(\neg\varphi) = 0$;

$\nu_w(\varphi \wedge \psi) = 1$, если $\nu_w(\varphi) = \nu_w(\psi) = 1$, иначе $\nu_w(\varphi \wedge \psi) = 0$;

$\nu_w(\varphi \vee \psi) = 1$, если $\nu_w(\varphi) = 1$ или $\nu_w(\psi) = 1$, иначе $\nu_w(\varphi \vee \psi) = 0$;

$\nu_w(\varphi \Longrightarrow \psi) = 1$, если $\nu_w(\varphi) = 0$ или $\nu_w(\psi) = 1$, иначе $\nu_w(\varphi \Longrightarrow \psi) = 0$;

$\nu_w(\varphi \Longleftrightarrow \psi) = 1$, если $\nu_w(\varphi) = \nu_w(\psi)$, иначе $\nu_w(\varphi \Longleftrightarrow \psi) = 0$.

Из этих правил видно, что классические логические связки действуют в рамках одного текущего мира w.

Для темпоральных модальных операторов логики LTL рекурсивные правила расширения функции ν_w выглядят следующим образом. Для каждого мира $w \in W$ имеем:

$\nu_w(\mathbf{X}\,\varphi) = 1$, если $\exists\, w' \in W$ такой, что $w \longrightarrow w'$ и $\nu_{w'}(\varphi) = 1$, иначе $\nu_w(\mathbf{X}\,\varphi) = 0$,

т. е. формула φ должна выполняться в следующем достижимом мире;

$\nu_w(\mathbf{F}\,\varphi) = 1$, если $\exists\, w' \in W$ такой, что $w \xrightarrow{*} w'$ и $\nu_{w'}(\varphi) = 1$, иначе $\nu_w(\mathbf{F}\,\varphi) = 0$,

т. е. формула φ должна выполняться в некотором мире w', который транзитивно достижим из текущего мира w;

$\nu_w(\mathbf{G}\,\varphi) = 1$, если $\forall\, w' \in W$ такого, что $w \xrightarrow{*} w'$, имеем $\nu_{w'}(\varphi) = 1$, иначе $\nu_w(\mathbf{G}\,\varphi) = 0$,

т. е. формула φ должна выполняться в текущем мире w и во всех мирах w', которые транзитивно достижимы из мира w;

$\nu_w(\varphi\,\mathbf{U}\,\psi) = 1$, если $\exists\, w' \in W$ такой, что для некоторого $i \in \mathbb{N} \cup \{0\}$ имеем $w \xrightarrow{i} w'$ и $\nu_{w'}(\psi) = 1$, а $\forall\, w'' \in W$ такого, что $\exists\, j, j < i$: $w \xrightarrow{j} w''$, выполняется $\nu_{w''}(\varphi) = 1$, иначе $\nu_w(\varphi\,\mathbf{U}\,\psi) = 0$,

т. е. формула ψ должна выполняться в некотором мире w', который транзитивно достижим из текущего мира w, но при этом во всех мирах, которые предшествуют миру w' на пути из w, должна выполняться формула φ.

В логике LTL операторы **F** и **G** являются двойственными: $\mathbf{G}\,\varphi = \neg\mathbf{F}\neg\varphi$. При этом сам оператор **F** представляет собой специальный случай применения оператора **U**, а именно $\mathbf{F}\varphi = \text{true}\,\mathbf{U}\,\varphi$.

Отметим, что каждую интерпретацию в логике LTL можно представить в виде бесконечной последовательности миров (с начальным миром w_0), связанных между собой функцией переходов, со своей оценочной функцией ν.

Для обозначения того, что из набора LTL-формул φ_1, φ_2, …, φ_m (предпосылок), где $m \in \mathbb{N}$, логически выводится формула ψ (заключение), используется металингвистический символ «$\models$». Вывод в логике LTL является справедливым, если он сохраняет свою истинность в начальных мирах всех интерпретаций. Пусть Σ — произвольное множество формул (набор предпосылок). Тогда заключение ψ будет *логически следовать* из набора формул Σ, т. е. $\Sigma \models \psi$, тогда и только тогда, когда не существует такой интерпретации, при которой все формулы из Σ в начальном мире являются истинными, а формула ψ в нём ложна. Другими словами, каждая интерпретация, обращающая в начальном мире все формулы из Σ в истину, в этом же мире обращает в истину и формулу ψ.

Более формально, логический вывод $\Sigma \models \psi$ справедлив тогда и только тогда, когда при всех возможных интерпретациях (w_0, W, T, v) для мира $w_0 \in W$ выполняется условие, что если $v_{w_0}(\varphi) = 1$ для всех предпосылок $\varphi \in \Sigma$, то $v_{w_0}(\psi) = 1$.

Запись $\Sigma \not\models \psi$ означает, что условие $\Sigma \models \psi$ не выполняется. В этом случае будет существовать хотя бы одна интерпретация (w_0, W, T, v), в рамках которой в начальном мире w_0 все формулы из Σ выполняются, а формула ψ в мире w_0 является ложной. Такая интерпретация называется контрпримером, или контр-моделью.

4. LTL как программная логика

Пусть $\mathrm{Cl}(mcM)$ — это класс ограниченных m-счётчиковых машин с $n+1$ булевыми переменными-состояниями $q_0, q_1, q_2, \ldots, q_n$ и соответствующими ограничениями $\mathrm{bnd}x_1, \mathrm{bnd}x_2, \ldots, \mathrm{bnd}x_m$ для счётчиков $x_1, x_2, \ldots, x_m$. Тогда применительно к этому классу m-счётчиковых машин $\mathrm{Cl}(mcM)$ линейная темпоральная логика LTL может быть рассмотрена следующим образом.

Сопоставим возможному миру $w \in W$ логики LTL конфигурацию некоторой счётчиковой машины из класса $\mathrm{Cl}(mcM)$ на некотором шаге её *исполнения*. Тогда каждый возможный мир будет взаимно однозначно соответствовать вектору значений переменных $(q_0, q_1, q_2, \ldots, q_n, x_1, x_2, \ldots, x_m)$, т. е. миры с одинаковым набором значений указанных переменных считаются одним миром.

Таким образом, для класса $\mathrm{Cl}(mcM)$ ограниченных m-счётчиковых машин множество различных миров W в каждой интерпретации (w_0, W, T, v) логики LTL будет конечным. Этот факт обеспечивает возможность проверки справедливости всех логических выводов в рамках логики LTL, рассмотренной применительно к классу счётчиковых машин $\mathrm{Cl}(mcM)$.

Непосредственную достижимость $w \longrightarrow w'$ возможного мира w' из мира w будем понимать как осуществление перехода из одной конфигурации счётчиковой машины в другую после срабатывания одного из правил переходов. Если для текущей конфигурации ни одно из правил переходов не выполняется, то предполагается, что происходит «пустой» переход в мир w', который будет соответствовать прежней конфигурации счётчиковой машины, т. е. $w' = w$.

В качестве элементарного высказывания логики LTL будем рассматривать выражение над переменными, построенное с использованием операторов сравнения, арифметических операторов и констант (целых неотрицательных чисел). Элементарное высказывание p формулируется таким образом, чтобы в результате оно могло принимать только два логических значения 1 (истина) или 0 (ложь). Результат оценочной функции $v_w(p)$ для элементарного высказывания p и мира w будет зависеть от значений задействованных в p переменных, которые они имеют в мире w.

Примером простого элементарного высказывания является выражение в виде одной булевой переменной-состояния.

Теперь для любой счётчиковой машины M из класса $\mathrm{Cl}(mcM)$ может быть построен такой набор LTL-формул, который будет описывать в точности все возможные исполнения счётчиковой машины M и только их. Всякий раз когда все LTL-формулы из набора будут одновременно обращаться в 1 (иметь значение «истина») при некоторой интерпретации (w_0, W, T, v), это будет означать, что интерпретация (w_0, W, T, v) соответствует одному из возможных исполнений машины M,

т. е. повторяет последовательность конфигураций в исполнении. И наоборот, для любого исполнения счётчиковой машины M будет существовать повторяющая его интерпретация, которая будет приводить к одновременному выполнению всех LTL-формул из рассматриваемого набора.

Пусть набор LTL-формул $\Sigma = \{\varphi_1, \dots, \varphi_k\}$, где $k \in \mathbb{N}$, описывает все возможные исполнения некоторой счётчиковой машины M из класса $\mathrm{Cl}(mcM)$, у которой q_0 — это начальное состояние, а состояние q_n является финальным. Рассмотрим LTL-формулу $\psi = «q_0 \Longrightarrow \mathbf{F}\,\mathbf{G}\,q_n»$. Эта формула является истинной только для тех интерпретаций, которые в начальном мире имеют $q_0 = 0$ или же если в начальном мире выполняется $q_0 = 1$, то в последовательности миров этих интерпретаций рано или поздно появится мир, начиная с которого для всех следующих миров будет выполняться $q_n = 1$. Тогда справедливость логического вывода $\varphi_1, \dots, \varphi_k \vDash \psi$ будет означать, что счётчиковая машина M, стартуя из состояния q_0 при любых начальных значениях счётчиков, обязательно завершит свою работу в финальном состоянии q_n, т. е. в результате конечной последовательности сработавших правил переходов машина окажется в финальном состоянии.

Если же имеет место $\varphi_1, \dots, \varphi_k \nvDash \psi$, то существует интерпретация (контрпример), соответствующая такому исполнению счётчиковой машины, которое для некоторых входных данных не приводит к требуемому результату вычислений, поскольку не попадает в финальное состояние. При этом набор формул $\varphi_1, \dots, \varphi_k, \neg\psi$ будет описывать все интерпретации, являющиеся контрпримерами, т. е. те интерпретации, для которых все формулы из этого набора одновременно являются истинными.

Получили, что задача проверки выполнимости для счётчиковой машины некоторого свойства, заданного LTL-формулой, может быть сведена к задаче доказательства справедливости логического вывода в рамках линейной темпоральной логики LTL.

В следующих разделах будет показано, каким образом поведение счётчиковых машин может быть представлено в виде набора LTL-формулы, а также будут рассмотрены примеры LTL-свойств счётчиковых машин.

5. LTL-спецификация ограниченной счётчиковой машины

Основная идея представления поведения ограниченной счётчиковой машины в виде набора LTL-формул состоит в описании (на языке логики LTL) того, каким образом происходит изменение значения каждой переменной счётчиковой машины на каждом шаге её исполнения.

При этом каждый счётчик $x_j \in X$, $1 \leqslant j \leqslant m$, задаётся своим набором из k бинарных переменных $x_j^{k-1}, x_j^{k-2}, \dots, x_j^1, x_j^0$ следующим образом:

$$x_j = \sum_{i=0}^{k-1} x_j^i 2^i = (x_j^{k-1} x_j^{k-2} \dots x_j^1 x_j^0)_2, \quad \text{где} \quad k = \lceil \log_2(\mathrm{bnd} x_j + 1) \rceil,$$

$\lceil\ \rceil$ — функция округления до большего целого числа.

Изменение значения бинарной переменной, соответствующей состоянию машины или одной из переменных счётчика, задаётся с помощью трёх LTL-формул. Первая LTL-формула описывает ситуации, при которых происходит возрастание значения соответствующей переменной, вторая LTL-формула определяет условия, приводящие к уменьшению значения переменной. Третья LTL-формула имеет жёсткую конструкцию, составленную из элементов первых двух формул. Она описывает условия, при которых рассматриваемая переменная не меняет своего значения во время работы счётчиковой машины.

Для спецификации поведения состояния q используются LTL-формулы вида:

$\mathbf{G}\,(\,\neg q \wedge\ \mathbf{X}(q)\ \Longrightarrow\ \mathit{FiringCondA}\,)$;

$\mathbf{G}\,(\ \ q \wedge \neg\mathbf{X}(q)\ \Longrightarrow\ \mathit{FiringCondD}\,)$.

Первая формула означает, что всякий раз, когда состояние q счётчиковой машины активируется, из этого следует, что было выполнено условие $FiringCondA$ перехода в состояние q, т. е. сработало некоторое правило переходов. Выражение $FiringCondD$ во второй формуле описывает условия выхода из состояния q.

Третья LTL-формула, описывающая условия сохранения прежнего состояния q, имеет вид

$$\mathbf{G}\,(\,(q \Longleftrightarrow \mathbf{X}(q)) \implies \neg(\neg q \wedge FiringCondA) \wedge \neg(q \wedge FiringCondD)\,).$$

Для спецификации поведения переменной x_j^i представления счётчика x_j используются LTL-формулы следующего вида:

$$\begin{aligned}
&\mathbf{G}\,(\,\neg x_j^i \wedge \ \mathbf{X}(x_j^i) \implies FiringCondInc \wedge x_j < \mathrm{bnd}x_j \wedge \ x_j^{i-1} \wedge \ x_j^{i-2} \wedge \cdots \wedge \ x_j^1 \wedge \ x_j^0 \ \vee \\
&\qquad\qquad FiringCondDec \wedge x_j > 0 \qquad \wedge \neg x_j^{i-1} \wedge \neg x_j^{i-2} \wedge \cdots \wedge \neg x_j^1 \wedge \neg x_j^0\ );\\
&\mathbf{G}\,(\ \ x_j^i \wedge \neg\mathbf{X}(x_j^i) \implies FiringCondInc \wedge x_j < \mathrm{bnd}x_j \wedge \ x_j^{i-1} \wedge \ x_j^{i-2} \wedge \cdots \wedge \ x_j^1 \wedge \ x_j^0 \ \vee \\
&\qquad\qquad FiringCondDec \wedge x_j > 0 \qquad \wedge \neg x_j^{i-1} \wedge \neg x_j^{i-2} \wedge \cdots \wedge \neg x_j^1 \wedge \neg x_j^0\ );\\
&\mathbf{G}\,(\,(x_j^i \Longleftrightarrow \mathbf{X}(x_j^i)) \implies \neg(FiringCondInc \wedge x_j < \mathrm{bnd}x_j \wedge \ x_j^{i-1} \wedge \ x_j^{i-2} \wedge \cdots \wedge \ x_j^1 \wedge \ x_j^0 \ \vee \\
&\qquad\qquad FiringCondDec \wedge x_j > 0 \qquad \wedge \neg x_j^{i-1} \wedge \neg x_j^{i-2} \wedge \cdots \wedge \neg x_j^1 \wedge \neg x_j^0)\ ).
\end{aligned}$$

Здесь выражения $FiringCondInc$ и $FiringCondDec$ представляют собой дизъюнкции состояний счётчиковой машины, при которых возникает необходимость изменения значения переменной x_j, если, конечно, это допускается соответствующими условиями $x_j < \mathrm{bnd}x_j$ и $x_j > 0$, где $x_j = \sum_{i=0}^{k-1} x_j^i 2^i$ и $k = \lceil \log_2(\mathrm{bnd}x_j + 1) \rceil$. А именно, $FiringCondInc = (q_p \vee \cdots \vee q_r)$, где $q_p, \ldots, q_r$ — состояния первого типа, в правилах переходов которых участвует счётчик x_j, т. е. происходит увеличение счётчика x_j на единицу при условии, что $x_j < \mathrm{bnd}x_j$. И $FiringCondDec = (q_s \vee \cdots \vee q_t)$, где $q_s, \ldots, q_t$ — состояния второго типа с правилами переходов, в которых значение счётчика x_j уменьшается на единицу при условии $x_j > 0$.

Если в описании машины для счётчика x_j правил переходов первого типа нет, то для переменной x_j^i имеем $FiringCondInc = \mathrm{false}$. Аналогичным образом, если для счётчика x_j нет правил переходов второго типа, то $FiringCondDec = \mathrm{false}$.

Заметим, что структура LTL-формул спецификации поведения переменной x_j^i позволяет заменить их одной LTL-формулой вида

$$\begin{aligned}
&\mathbf{G}\,(\ \mathbf{X}(x_j^i) \ \Longleftrightarrow \ \neg(\,x_j^i \Longleftrightarrow (FiringCondInc \wedge x_j < \mathrm{bnd}x_j \wedge \ x_j^{i-1} \wedge \ x_j^{i-2} \wedge \cdots \wedge \ x_j^1 \wedge \ x_j^0 \ \vee \\
&\qquad\qquad FiringCondDec \wedge x_j > 0 \qquad \wedge \neg x_j^{i-1} \wedge \neg x_j^{i-2} \wedge \cdots \wedge \neg x_j^1 \wedge \neg x_j^0))\ ).
\end{aligned}$$

Опираясь на формальное определение ограниченной счётчиковой машины, опишем более подробно схему построения LTL-спецификации изменения значения переменной-состояния.

LTL-формула, учитывающая условия, при которых счётчиковая машина переходит из некоторого текущего состояния в новое состояние q_k, т. е. логическая переменная q_k получает значение 1, имеет следующий вид:

$$\mathbf{G}\,(\ \neg q_k \wedge \mathbf{X}(q_k) \implies q_i \wedge x_j > 0 \vee \cdots \vee q_r \wedge \neg(x_t > 0) \vee \cdots \vee q_s \wedge x_l < \mathrm{bnd}x_l\ ),$$

где условия вида $q_i \wedge x_j > 0$ соответствуют правилам переходов второго типа

$$(\delta_i)\ \ q_i : \ \text{if } x_j > 0 \text{ then } (x_j := x_j - 1;\ \text{goto } \mathbf{q_k}) \text{ else goto } q_h,$$

а условия вида $q_r \wedge \neg(x_t > 0)$ — правилам

$$(\delta_r)\ \ q_r : \ \text{if } x_t > 0 \text{ then } (x_t := x_t - 1;\ \text{goto } q_p) \text{ else goto } \mathbf{q_k}.$$

Условия вида $q_s \wedge x_l < \mathrm{bnd} x_l$ соответствуют правилам переходов первого типа

$(\delta_s)\ \ q_s :\ \ \mathsf{if}\ x_l < \mathrm{bnd} x_l\ \mathsf{then}\ x_l := x_l + 1;\ \mathsf{goto}\ \mathbf{q_k}.$

Важно отметить, что все переменные приведённой LTL-формулы, стоящие в условиях после оператора импликации, должны быть отличными от переменной-состояния q_k, т. е. переход в то же самое состояние здесь не специфицируется.

Если счётчиковая машина не имеет правил переходов, приводящих в состояние q_k, то для этого состояния LTL-формула «активации» строится как

$$\mathbf{G}(\ \neg q_k \wedge \mathbf{X}(q_k) \implies \mathrm{false}\ ).$$

LTL-формула «деактивации» (выхода из) состояния q_k, которому соответствует правило переходов первого типа со счётчиком x_j, имеет вид:

$$\mathbf{G}(\ q_k \wedge \neg\mathbf{X}(q_k) \implies x_j < \mathrm{bnd} x_j\ ).$$

Для правила переходов второго типа без петель получаем LTL-формулу

$$\mathbf{G}(\ q_k \wedge \neg\mathbf{X}(q_k) \implies \mathrm{true}\ ),$$

где логическая константа true означает, что переменная q_k должна быть обязательно сброшена в ноль уже на следующем шаге после её активации, т. е. после присваивания значения 1.

Несмотря на то что импликация внутри последней LTL-формулы является тавтологией, построение этой формулы имеет смысл, поскольку условие срабатывания в виде константы true участвует в LTL-формуле, описывающей ситуации, при которых переменная q_k сохраняет своё значение. По сути здесь налагается запрет иметь переменной q_k значение 1 дольше одного шага исполнения счётчиковой машины.

Если правило переходов для q_k имеет вид петли $(\delta_k)\ \ \mathbf{q_k} :\ \mathsf{if}\ x_j < \mathrm{bnd} x_j\ \mathsf{then}\ x_j := x_j + 1;\ \mathsf{goto}\ \mathbf{q_k}$, состояние q_k соответствует двум петлям $(\delta_k)\ \ \mathbf{q_k} :\ \mathsf{if}\ x_l > 0\ \mathsf{then}\ (x_j := x_j - 1;\ \mathsf{goto}\ \mathbf{q_k})\ \mathsf{else}\ \mathsf{goto}\ \mathbf{q_k}$ или же q_k является финальным состоянием, то переменная q_k будет иметь следующую LTL-формулу:

$$\mathbf{G}(\ q_k \wedge \neg\mathbf{X}(q_k) \implies \mathrm{false}\ ).$$

Если состояние второго типа q_k имеет в правиле переходов со счётчиком x_j только одну петлю, то для переменной q_k выбирается соответствующий вариант LTL-формулы

$$\mathbf{G}(\ q_k \wedge \neg\mathbf{X}(q_k) \implies x_j > 0\ )\quad \text{или}\quad \mathbf{G}(\ q_k \wedge \neg\mathbf{X}(q_k) \implies \neg(x_j > 0)\ ).$$

LTL-формула, описывающая ситуации, при которых переменная-состояние q_k сохраняет своё значение после применения некоторого правила переходов, строится автоматически по уже рассмотренным формулам «активации» и «деактивации»:

$$\mathbf{G}(\ (q_k \iff \mathbf{X}(q_k)) \implies \neg(\neg q_k \wedge \mathit{FiringCondA}) \wedge \neg(q_k \wedge \mathit{FiringCondD})\ ),$$

где $\mathit{FiringCondA}$ — условие, которое стоит после оператора импликации в LTL-формуле активации состояния q_k, а $\mathit{FiringCondD}$ — соответствующее условие из формулы деактивации состояния q_k.

И наконец, во всех рассмотренных LTL-формулах заменим условные выражения $x_j < \mathrm{bnd} x_j$ и $x_j > 0$ на соответствующие булевы функции $\mathrm{lt}(x_j^{k-1}, \ldots, x_j^0)$ и $\mathrm{gt}(x_j^{k-1}, \ldots, x_j^0)$, реализующие эти выражения:

$$x_j < \mathrm{bnd} x_j \ \equiv\ \mathrm{lt}(x_j^{k-1}, \ldots, x_j^0) \stackrel{\mathrm{def}}{=} \neg x_j^{k-1}\ op_{k-1}\ (\neg x_j^{k-2}\ op_{k-2}\ (\ldots(\neg x_j^1\ op_1\ (\neg x_j^0\ op_0\ 0))\ldots)),$$

$$x_j > 0 \ \equiv\ \mathrm{gt}(x_j^{k-1}, \ldots, x_j^0) \stackrel{\mathrm{def}}{=} x_j^{k-1} \vee x_j^{k-2} \vee \cdots \vee x_j^1 \vee x_j^0,$$

где $x_j = \sum_{i=0}^{k-1} x_j^i\, 2^i$, $\mathrm{bnd}x_j = \sum_{i=0}^{k-1} b_j^i\, 2^i$, $b_j^i \in \{0,1\}$, $k = \lceil \log_2(\mathrm{bnd}x_j + 1) \rceil$, $op_i \in \{\vee, \wedge\}$, $0 \leqslant i \leqslant k-1$. При этом $op_i = \vee$, если $b_j^i = 1$, и $op_i = \wedge$, если $b_j^i = 0$.

Отметим, что эти замены условных выражений могут быть выполнены через бинарные переменные $\mathrm{lt}x_j$ и $\mathrm{gt}x_j$ и LTL-формулы

$$\mathbf{G}\,(\ \mathrm{lt}x_j \iff \neg x_j^{k-1}\ op_{k-1}\ (\neg x_j^{k-2}\ op_{k-2}\ (\ldots(\neg x_j^1\ op_1\ (\neg x_j^0\ op_0\ 0))\ldots))\ ),$$

$$\mathbf{G}\,(\ \mathrm{gt}x_j \iff x_j^{k-1} \vee x_j^{k-2} \vee \cdots \vee x_j^1 \vee x_j^0\ ).$$

Таким образом, во всех формулах LTL-спецификации ограниченной счётчиковой машины будут использоваться только логические операторы и двоичные переменные.

6. LTL-спецификация счётчиковой машины возведения числа в квадрат

Построим LTL-спецификацию поведения ограниченной трехсчётчиковой машины $3cM$ возведения числа n в квадрат для всех значений n из диапазона 0 до 15 включительно. В этой LTL-спецификации переменная-счётчик a при инициализации получает значение n, $0 \leqslant n \leqslant 15$. Переменная-состояние q_0 инициализируется единицей, т. е. изначально имеем $q_0 = 1$. Остальные переменные при инициализации выставляются в ноль. В LTL-формулах в качестве предельных значений переменных-счётчиков будем использовать конкретные числа: $\mathrm{bnd}a = 15$, $\mathrm{bnd}b = 15$ и $\mathrm{bnd}c = 225$.

Целочисленные переменные-счётчики a, b и c, а также число n, которое подаётся на вход счётчиковой машины, будем представлять с помощью соответствующих наборов бинарных переменных. При этом формула Sn используется для фиксации значения n на протяжении одного исполнения счётчиковой машины $3cM$, стартующей из начального состояния q_0 при $a = n$, $b = 0$ и $c = 0$.

Sn : $\mathbf{G}\,(\ (\mathbf{X}(n_3) \iff n_3) \wedge (\mathbf{X}(n_2) \iff n_2) \wedge (\mathbf{X}(n_1) \iff n_1) \wedge (\mathbf{X}(n_0) \iff n_0)\ )$;

Sinit : $q_0 \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4 \wedge \neg q_5 \wedge \neg q_6 \wedge \neg q_7 \wedge (a_3 \iff n_3) \wedge (a_2 \iff n_2) \wedge (a_1 \iff n_1) \wedge (a_0 \iff n_0) \wedge \neg b_3 \wedge \neg b_2 \wedge \neg b_1 \wedge \neg b_0 \wedge \neg c_7 \wedge \neg c_6 \wedge \neg c_5 \wedge \neg c_4 \wedge \neg c_3 \wedge \neg c_2 \wedge \neg c_1 \wedge \neg c_0$;

Sgl : $\mathbf{G}\,(\ \mathrm{gt}a \iff a_3 \vee a_2 \vee a_1 \vee a_0\ ) \wedge$
$\mathbf{G}\,(\ \mathrm{gt}b \iff b_3 \vee b_2 \vee b_1 \vee b_0\ ) \wedge$
$\mathbf{G}\,(\ \mathrm{lt}a \iff \neg a_3 \vee \neg a_2 \vee \neg a_1 \vee \neg a_0\ ) \wedge$
$\mathbf{G}\,(\ \mathrm{lt}b \iff \neg b_3 \vee \neg b_2 \vee \neg b_1 \vee \neg b_0\ ) \wedge$
$\mathbf{G}\,(\ \mathrm{lt}c \iff \neg c_7 \vee \neg c_6 \vee \neg c_5 \vee \neg c_4 \wedge \neg c_3 \wedge \neg c_2 \wedge \neg c_1 \wedge \neg c_0\ )$;

Sa : $\mathbf{G}\,(\ \mathbf{X}(a_3) \iff \neg(a_3 \iff (q_6 \wedge \mathrm{lt}a \wedge a_2 \wedge a_1 \wedge a_0 \vee (q_0 \vee q_2) \wedge \mathrm{gt}a \wedge \neg a_2 \wedge \neg a_1 \wedge \neg a_0))\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(a_2) \iff \neg(a_2 \iff (q_6 \wedge \mathrm{lt}a \wedge a_1 \wedge a_0 \quad \vee (q_0 \vee q_2) \wedge \mathrm{gt}a \wedge \neg a_1 \wedge \neg a_0))\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(a_1) \iff \neg(a_1 \iff (q_6 \wedge \mathrm{lt}a \wedge a_0 \quad \vee (q_0 \vee q_2) \wedge \mathrm{gt}a \wedge \neg a_0))\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(a_0) \iff \neg(a_0 \iff (q_6 \wedge \mathrm{lt}a \quad \vee (q_0 \vee q_2) \wedge \mathrm{gt}a))\ )$;

Sb : $\mathbf{G}\,(\ \mathbf{X}(b_3) \iff \neg(b_3 \iff (q_3 \wedge \mathrm{lt}b \wedge b_2 \wedge b_1 \wedge b_0 \vee q_5 \wedge \mathrm{gt}b \wedge \neg b_2 \wedge \neg b_1 \wedge \neg b_0))\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(b_2) \iff \neg(b_2 \iff (q_3 \wedge \mathrm{lt}b \wedge b_1 \wedge b_0 \quad \vee q_5 \wedge \mathrm{gt}b \wedge \neg b_1 \wedge \neg b_0))\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(b_1) \iff \neg(b_1 \iff (q_3 \wedge \mathrm{lt}b \wedge b_0 \quad \vee q_5 \wedge \mathrm{gt}b \wedge \neg b_0))\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(b_0) \iff \neg(b_0 \iff (q_3 \wedge \mathrm{lt}b \quad \vee q_5 \wedge \mathrm{gt}b))\ )$;

Sc : $\mathbf{G}\,(\ \mathbf{X}(c_7) \iff \neg(c_7 \iff (q_1 \vee q_4) \wedge \mathrm{lt}c \wedge c_6 \wedge c_5 \wedge c_4 \wedge c_3 \wedge c_2 \wedge c_1 \wedge c_0)\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(c_6) \iff \neg(c_6 \iff (q_1 \vee q_4) \wedge \mathrm{lt}c \wedge c_5 \wedge c_4 \wedge c_3 \wedge c_2 \wedge c_1 \wedge c_0)\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(c_5) \iff \neg(c_5 \iff (q_1 \vee q_4) \wedge \mathrm{lt}c \wedge c_4 \wedge c_3 \wedge c_2 \wedge c_1 \wedge c_0)\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(c_4) \iff \neg(c_4 \iff (q_1 \vee q_4) \wedge \mathrm{lt}c \wedge c_3 \wedge c_2 \wedge c_1 \wedge c_0)\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(c_3) \iff \neg(c_3 \iff (q_1 \vee q_4) \wedge \mathrm{lt}c \wedge c_2 \wedge c_1 \wedge c_0)\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(c_2) \iff \neg(c_2 \iff (q_1 \vee q_4) \wedge \mathrm{lt}c \wedge c_1 \wedge c_0)\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(c_1) \iff \neg(c_1 \iff (q_1 \vee q_4) \wedge \mathrm{lt}c \wedge c_0)\ ) \wedge$
$\mathbf{G}\,(\ \mathbf{X}(c_0) \iff \neg(c_0 \iff (q_1 \vee q_4) \wedge \mathrm{lt}c)\ )$;

$$
\begin{aligned}
\text{Sq0}: \quad & \mathbf{G}(\ \neg q_0 \wedge \ \mathbf{X}(q_0) \implies q_5 \wedge \neg \mathrm{gt}b\ ) \wedge \\
& \mathbf{G}(\ \ q_0 \wedge \neg\mathbf{X}(q_0) \implies \mathrm{true}\ ) \wedge \\
& \mathbf{G}(\ (q_0 \iff \mathbf{X}(q_0)) \implies \neg(\neg q_0 \wedge q_5 \wedge \neg \mathrm{gt}b) \wedge \neg q_0\ ); \\
\text{Sq1}: \quad & \mathbf{G}(\ \neg q_1 \wedge \ \mathbf{X}(q_1) \implies q_0 \wedge \mathrm{gt}a \vee q_4 \wedge \mathrm{lt}c\ ) \wedge \\
& \mathbf{G}(\ \ q_1 \wedge \neg\mathbf{X}(q_1) \implies \mathrm{lt}c\ ) \wedge \\
& \mathbf{G}(\ (q_1 \iff \mathbf{X}(q_1)) \implies \neg(\neg q_1 \wedge (q_0 \wedge \mathrm{gt}a \vee q_4 \wedge \mathrm{lt}c)) \wedge \neg(q_1 \wedge \mathrm{lt}c)\ ); \\
\text{Sq2}: \quad & \mathbf{G}(\ \neg q_2 \wedge \ \mathbf{X}(q_2) \implies q_1 \wedge \mathrm{lt}c\ ) \wedge \\
& \mathbf{G}(\ \ q_2 \wedge \neg\mathbf{X}(q_2) \implies \mathrm{true}\ ) \wedge \\
& \mathbf{G}(\ (q_2 \iff \mathbf{X}(q_2)) \implies \neg(\neg q_2 \wedge q_1 \wedge \mathrm{lt}c) \wedge \neg q_2\ ); \\
\text{Sq3}: \quad & \mathbf{G}(\ \neg q_3 \wedge \ \mathbf{X}(q_3) \implies q_2 \wedge \mathrm{gt}a\ ) \wedge \\
& \mathbf{G}(\ \ q_3 \wedge \neg\mathbf{X}(q_3) \implies \mathrm{lt}b\ ) \wedge \\
& \mathbf{G}(\ (q_3 \iff \mathbf{X}(q_3)) \implies \neg(\neg q_3 \wedge q_2 \wedge \mathrm{gt}a) \wedge \neg(q_3 \wedge \mathrm{lt}b)\ ); \\
\text{Sq4}: \quad & \mathbf{G}(\ \neg q_4 \wedge \ \mathbf{X}(q_4) \implies q_3 \wedge \mathrm{lt}b\ ) \wedge \\
& \mathbf{G}(\ \ q_4 \wedge \neg\mathbf{X}(q_4) \implies \mathrm{lt}c\ ) \wedge \\
& \mathbf{G}(\ (q_4 \iff \mathbf{X}(q_4)) \implies \neg(\neg q_4 \wedge q_3 \wedge \mathrm{lt}b) \wedge \neg(q_4 \wedge \mathrm{lt}c)\ ); \\
\text{Sq5}: \quad & \mathbf{G}(\ \neg q_5 \wedge \ \mathbf{X}(q_5) \implies q_2 \wedge \neg\mathrm{gt}a \vee q_6 \wedge \mathrm{lt}a\ ) \wedge \\
& \mathbf{G}(\ \ q_5 \wedge \neg\mathbf{X}(q_5) \implies \mathrm{true}\ ) \wedge \\
& \mathbf{G}(\ (q_5 \iff \mathbf{X}(q_5)) \implies \neg(\neg q_5 \wedge (q_2 \wedge \neg\mathrm{gt}a \vee q_6 \wedge \mathrm{lt}a)) \wedge \neg q_5\ ); \\
\text{Sq6}: \quad & \mathbf{G}(\ \neg q_6 \wedge \ \mathbf{X}(q_6) \implies q_5 \wedge \mathrm{gt}b\ ) \wedge \\
& \mathbf{G}(\ \ q_6 \wedge \neg\mathbf{X}(q_6) \implies \mathrm{lt}a\ ) \wedge \\
& \mathbf{G}(\ (q_6 \iff \mathbf{X}(q_6)) \implies \neg(\neg q_6 \wedge q_5 \wedge \mathrm{gt}b) \wedge \neg(q_6 \wedge \mathrm{lt}a)\ ); \\
\text{Sq7}: \quad & \mathbf{G}(\ \neg q_7 \wedge \ \mathbf{X}(q_7) \implies q_0 \wedge \neg\mathrm{gt}a\ ) \wedge \\
& \mathbf{G}(\ \ q_7 \wedge \neg\mathbf{X}(q_7) \implies \mathrm{false}\ ) \wedge \\
& \mathbf{G}(\ (q_7 \iff \mathbf{X}(q_7)) \implies \neg(\neg q_7 \wedge q_0 \wedge \neg\mathrm{gt}a)\ ).
\end{aligned}
$$

Рассмотрим LTL-свойства счётчиковой машины $3cM$, которые обязательно должны выполняться для любого её исполнения. При записи формул будем использовать исходные названия счётчиков a, b и c, а также вспомогательную переменную n, предполагая, что $a = \sum_{i=0}^{3} a_i 2^i$, $b = \sum_{i=0}^{3} b_i 2^i$, $c = \sum_{i=0}^{7} c_i 2^i$ и $n = \sum_{i=0}^{3} n_i 2^i$.

P1 : $\mathbf{G}(\ q_7 \implies c = n^2 \wedge a = 0 \wedge b = 0\ )$.
Это свойство означает, что если счётчиковая машина $3cM$ оказывается в финальном состоянии q_7, то значения счётчиков a и b будут равны 0, а счётчик c будет содержать искомый результат n^2.

P2 : $\mathbf{G}(\ a + b \leqslant n\ )$.
Это свойство требует, чтобы суммарное значение счётчиков a и b было ограничено константой n на протяжении всего исполнения счётчиковой машины $3cM$.

P3 : $\mathbf{G}(\ c \leqslant n^2\ )$.
Требуется, чтобы значение счётчика c на протяжении всего исполнения машины $3cM$ не выходило за пределы n^2.

P4 : $\mathbf{G}\mathbf{F}(q_7)$.
Счётчиковая машина $3cM$ из любого состояния (в том числе и из начального q_0) всегда рано или поздно перейдёт в финальное состояние q_7.

P5 : $\mathbf{G}(\ q_7 \implies \mathbf{X}(q_7)\ )$.
Если счётчиковая машина $3cM$ попадает в финальное состояние q_7, то она навсегда остаётся в этом состоянии.

P6 : $\mathbf{G}(\ q_0 + q_1 + q_2 + q_3 + q_4 + q_5 + q_6 + q_7 = 1\ )$.
Всегда активно только одно состояние счётчиковой машины $3cM$.

Тогда справедливость логического вывода (в рамках логики LTL)

$$\mathrm{Sn}, \mathrm{Sinit}, \mathrm{Sa}, \mathrm{Sb}, \mathrm{Sc}, \mathrm{Sq0}, \mathrm{Sq1}, \mathrm{Sq2}, \mathrm{Sq3}, \mathrm{Sq4}, \mathrm{Sq5}, \mathrm{Sq6}, \mathrm{Sq7} \models \mathrm{P1}, \mathrm{P2}, \mathrm{P3}, \mathrm{P4}, \mathrm{P5}, \mathrm{P6}$$

будет означать, что счётчиковая машина $3cM$ для любого n, $0 \leqslant n \leqslant 15$, стартуя из начального состояния q_0 при $a = n$, $b = 0$ и $c = 0$, обязательно завершит работу в финальном состоянии q_7 с результатом $a = 0$, $b = 0$ и $c = n^2$. При этом на протяжении всего исполнения будет выполняться $a + b \leqslant n$ и $c \leqslant n^2$. Действительно, левая часть логического вывода описывает те и только те интерпретации, которые соответствуют всем возможным исполнениям счётчиковой машины $3cM$ при условии $0 \leqslant n \leqslant 15$, а правая часть вывода обязывает, чтобы эти интерпретации/исполнения удовлетворяли требуемым свойствам машины $3cM$.

Рассмотрим пример логического вывода, который не будет являться справедливым в логике LTL применительно к счётчиковой машине $3cM$. Потребуем, чтобы в каждом возможном исполнении машины $3cM$ итоговый результат её работы в состоянии q_7 был отличным от $c = 2n$ при $n > 0$:

$$\mathrm{P7}: \quad \mathbf{G}\,(\, q_7 \wedge n > 0 \implies \neg(c = 2n)\,).$$

В этом случае получим, что формула P7 не будет выводиться из приведённых выше предпосылок

$$\mathrm{Sn}, \mathrm{Sinit}, \mathrm{Sa}, \mathrm{Sb}, \mathrm{Sc}, \mathrm{Sq0}, \mathrm{Sq1}, \mathrm{Sq2}, \mathrm{Sq3}, \mathrm{Sq4}, \mathrm{Sq5}, \mathrm{Sq6}, \mathrm{Sq7} \not\models \mathrm{P7},$$

так как существует контрпример — интерпретация, при которой все формулы левой части логического вывода являются истинными, а правая часть в виде формулы-заключения P7 становится ложной. Эта интерпретация представляет собой цепочку из 16 различных миров, последний из которых имеет петлю, т. е. достижим сам из себя. Контрпример соответствует исполнению счётчиковой машины $3cM$ при $n = 2$.

Проверку справедливости рассмотренных логических выводов можно выполнить, например, с помощью программного средства верификации Cadence SMV [5, 7, 11].

7. Проверка справедливости логических выводов

Ниже приводится код на языке программного средства верификации Cadence SMV [5], позволяющий проверить выполнимость свойств счётчиковой машины $3cM$ с использованием линейной темпоральной логики LTL.

На вход верификатора Cadence SMV подаются описания переменных-состояний, переменных-счётчиков и «константы» n, а также описания вспомогательных переменных, применяющихся для реализации условных функций. Указываются области допустимых значений этих переменных. Запись формул логики LTL происходит через ключевое слово `assert`.

Проверка выполнимости каждого LTL-свойства `P1`, `P2`, `P3`, `P4`, `P5`, `P6` и `P7` проводится по отдельности посредством вызова соответствующей команды главного меню программы Cadence SMV.

Поскольку имена свойств задействованы в конструкции `using ... prove`, а точнее, указаны в разделе `prove`, то проверка выполнимости выбранного свойства будет проводиться только для тех интерпретаций логики LTL, которые одновременно удовлетворяют всем LTL-формулам, перечисленным в разделе `using`.

Контрпример для свойства `P7` выводится в виде таблицы, каждый столбец которой содержит вектор значений всех переменных в соответствующем мире найденной интерпретации.

На языке программного средства Cadence SMV символы «&», «|», «~», «->» и «<->» означают логические операторы «$\wedge$», «$\vee$», «$\neg$», «$\Longrightarrow$» и «$\Longleftrightarrow$» соответственно. А константы «true» и «false» имеют вид «1» и «0».

```
module main()
{ /* ------- описание переменных ------- */
  q0, q1, q2, q3, q4, q5, q6, q7: 0..1;
  a, b, n: 0..15;
  c: 0..255;
  c0, c1, c2, c3, c4, c5, c6, c7: 0..1;
  a0, a1, a2, a3: 0..1;
  b0, b1, b2, b3: 0..1;
  n0, n1, n2, n3: 0..1;
  gta, lta, gtb, ltb, ltc: 0..1;
  /* ------- представления счётчиков a, b и c ------- */
  a :=  a0 + a1*2 + a2*4 + a3*8;
  b :=  b0 + b1*2 + b2*4 + b3*8;
  n :=  n0 + n1*2 + n2*4 + n3*8;
  c :=  c0 + c1*2 + c2*4 + c3*8 + c4*16 + c5*32 + c6*64 + c7*128;
  /* ------- LTL-спецификация ------- */
  Sn: assert /* формулы для константы n */
      G( (X(n3) <-> n3) & (X(n2) <-> n2) & (X(n1) <-> n1) & (X(n0) <-> n0) );
  Sinit: assert /* формула инициализации */
          q0 & ~q1 & ~q2 & ~q3 & ~q4 & ~q5 & ~q6 & ~q7 &
         (a3 <-> n3) & (a2 <-> n2) & (a1 <-> n1) & (a0 <-> n0) &
         ~b3 & ~b2 & ~b1 & ~b0 & ~c7 & ~c6 & ~c5 & ~c4 & ~c3 & ~c2 & ~c1 & ~c0;
  Sgl: assert /* формулы для функций gt и lt */
       G( gta <->  a3 |  a2 |  a1 |  a0 ) &
       G( gtb <->  b3 |  b2 |  b1 |  b0 ) &
       G( lta <-> ~a3 | ~a2 | ~a1 | ~a0 ) &
       G( ltb <-> ~b3 | ~b2 | ~b1 | ~b0 ) &
       G( ltc <-> ~c7 | ~c6 | ~c5 | ~c4 & ~c3 & ~c2 & ~c1 & ~c0 );
  Sa: assert /* формулы для переменных a3, a2, a1, a0 представления счётчика a */
      G( X(a3) <->  ~(a3 <-> (q6 & lta & a2 & a1 & a0 | (q0 | q2) & gta & ~a2 & ~a1 & ~a0)) ) &
      G( X(a2) <->  ~(a2 <-> (q6 & lta & a1 & a0      | (q0 | q2) & gta & ~a1 & ~a0      )) ) &
      G( X(a1) <->  ~(a1 <-> (q6 & lta & a0           | (q0 | q2) & gta & ~a0           )) ) &
      G( X(a0) <->  ~(a0 <-> (q6 & lta                | (q0 | q2) & gta                )) );
  Sb: assert /* формулы для переменных b3, b2, b1, b0 представления счётчика b */
      G( X(b3) <->  ~(b3 <-> (q3 & ltb & b2 & b1 & b0 | q5 & gtb & ~b2 & ~b1 & ~b0)) ) &
      G( X(b2) <->  ~(b2 <-> (q3 & ltb & b1 & b0      | q5 & gtb & ~b1 & ~b0      )) ) &
      G( X(b1) <->  ~(b1 <-> (q3 & ltb & b0           | q5 & gtb & ~b0           )) ) &
      G( X(b0) <->  ~(b0 <-> (q3 & ltb                | q5 & gtb                )) );
  Sc: assert /* формулы для переменных c7, c6, c5, c4, c3, c2, c1, c0 представления счётчика c */
      G( X(c7) <->  ~(c7 <-> (q1 | q4) & ltc & c6 & c5 & c4 & c3 & c2 & c1 & c0) ) &
      G( X(c6) <->  ~(c6 <-> (q1 | q4) & ltc & c5 & c4 & c3 & c2 & c1 & c0     ) ) &
      G( X(c5) <->  ~(c5 <-> (q1 | q4) & ltc & c4 & c3 & c2 & c1 & c0          ) ) &
      G( X(c4) <->  ~(c4 <-> (q1 | q4) & ltc & c3 & c2 & c1 & c0               ) ) &
      G( X(c3) <->  ~(c3 <-> (q1 | q4) & ltc & c2 & c1 & c0                    ) ) &
      G( X(c2) <->  ~(c2 <-> (q1 | q4) & ltc & c1 & c0                         ) ) &
      G( X(c1) <->  ~(c1 <-> (q1 | q4) & ltc & c0                              ) ) &
      G( X(c0) <->  ~(c0 <-> (q1 | q4) & ltc                                   ) );
  Sq0: assert /* формулы для переменной q0 */
       G( ~q0  &  X(q0)  ->  q5 & ~gtb ) &
       G(  q0  & ~X(q0)  ->  1 ) &
       G( (q0 <-> X(q0)) ->  ~(~q0 & q5 & ~gtb) & ~q0 );
  Sq1: assert /* формулы для переменной q1 */
       G( ~q1  &  X(q1)  ->  q0 & gta | q4 & ltc ) &
       G(  q1  & ~X(q1)  ->  ltc ) &
       G( (q1 <-> X(q1)) ->  ~(~q1 & (q0 & gta | q4 & ltc)) & ~(q1 & ltc) );
```

```
  Sq2: assert /* формулы для переменной q2 */
       G( ~q2  &  X(q2)  ->  q1 & ltc ) &
       G(  q2  & ~X(q2)  ->  1 ) &
       G( (q2 <-> X(q2)) ->  ~(~q2 & q1 & ltc) & ~q2 );
  Sq3: assert /* формулы для переменной q3 */
       G( ~q3  &  X(q3)  ->  q2 & gta ) &
       G(  q3  & ~X(q3)  ->  ltb ) &
       G( (q3 <-> X(q3)) ->  ~(~q3 & q2 & gta) & ~(q3 & ltb) );
  Sq4: assert /* формулы для переменной q4 */
       G( ~q4  &  X(q4)  ->  q3 & ltb ) &
       G(  q4  & ~X(q4)  ->  ltc ) &
       G( (q4 <-> X(q4)) ->  ~(~q4 & q3 & ltb) & ~(q4 & ltc) );
  Sq5: assert /* формулы для переменной q5 */
       G( ~q5  &  X(q5)  ->  q2 & ~gta | q6 & lta ) &
       G(  q5  & ~X(q5)  ->  1 ) &
       G( (q5 <-> X(q5)) ->  ~(~q5 & (q2 & ~gta | q6 & lta)) & ~q5 );
  Sq6: assert /* формулы для переменной q6 */
       G( ~q6  &  X(q6)  ->  q5 & gtb ) &
       G(  q6  & ~X(q6)  ->  lta ) &
       G( (q6 <-> X(q6)) ->  ~(~q6 & q5 & gtb) & ~(q6 & lta) );
  Sq7: assert /* формулы для переменной q7 */
       G( ~q7  &  X(q7)  ->  q0 & ~gta ) &
       G(  q7  & ~X(q7)  ->  0 ) &
       G( (q7 <-> X(q7)) ->  ~(~q7 & q0 & ~gta) );
  /* ------- проверяемые свойства счётчиковой машины ------- */
  P1: assert G( q7 -> c=n*n & a=0 & b=0 );
  P2: assert G( a+b <= n );
  P3: assert G( c <= n*n );
  P4: assert G F(q7);
  P5: assert G( q7 -> X(q7) );
  P6: assert G( q0+q1+q2+q3+q4+q5+q6+q7 = 1 );
  P7: assert G( q7 & n>0 -> ~(c = 2*n) );
  /* ------- предпосылки и заключения в логических выводах ------- */
  using  /* используя предпосылки */
    Sn, Sinit, Sgl, Sa, Sb, Sc, Sq0, Sq1, Sq2, Sq3, Sq4, Sq5, Sq6, Sq7
  prove  /* доказать заключения */
    P1, P2, P3, P4, P5, P6, P7;
}
```

Заключение

В статье предложен способ описания поведения ограниченной счётчиковой машины с помощью набора формул линейной темпоральной логики LTL. В LTL-спецификации используются только двоичные переменные в виде элементарных высказываний. При этом обращение к предыдущим значениям переменных осуществляется исключительно в терминах самой логики LTL посредством соответствующих формул. Введённые ограничения на допустимые значения счётчиков играют роль ограничений на размер памяти программной системы и позволяют заранее определить количество двоичных переменных, необходимых для представления счётчиков машины Минского.

Ранее в работе [1] для хранения предыдущих значений счётчиков и состояний машины Минского отводились специальные дополнительные переменные. Однако при доказательстве справедливости логических LTL-выводов в программе Cadence SMV каждая переменная преобразуется в набор двоичных переменных. Таким образом, представление счётчиков и состояний машины сразу в виде двоичных переменных, не требующее выделения дублирующих переменных для хранения предыдущих значений, в два раза уменьшает общее число переменных, задействованных в описании поведения машины, что значительно сокращает время на проверку справедливости логических выводов.

References

[1] E. V. Kuzmin, "LTL-Specification of Counter Machines", in Russian, *Modeling and Analysis of Information Systems*, vol. 28, no. 1, pp. 104–119, 2021.

[2] M. Minsky, *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., 1967.

[3] R. Schroeppel, "A Two Counter Machine Cannot Calculate 2^N", Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Artificial Intelligence Memo #257, 1972, p. 32.

[4] E. V. Kuzmin, *Counter Machines*, in Russian. Yaroslavl: Yaroslavl State University, 2010, p. 128.

[5] *Cadence SMV*. [Online]. Available: http://www.kenmcmil.com/smv.html.

[6] A. Pnueli, "The Temporal Logic of Programs", in *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, IEEE Computer Society Press, 1977, pp. 46–57.

[7] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. The MIT Press, 2001.

[8] C. Baier and J.-P. Katoen, *Principles of Model Checking*. The MIT Press, 2008.

[9] G. Priest, *An Introduction to Non-Classical Logic. From if to is.* Cambridge University Press, 2008, p. 648.

[10] E. V. Kuzmin, *Non-Classical Propositional Logics*, in Russian. Yaroslavl: P.G. Demidov Yaroslavl State University, 2016, p. 160.

[11] E. Clarke, O. Grumberg, and K. Hamaguchi, "Another Look at LTL Model Checking", Carnegie Mellon University, Tech. Rep. CMU-CS-94-114, 1994.