MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 28, NO. 1, 2021
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

LTL-Specification of Counter Machines

E. V. Kuzmin[1]

DOI: 10.18255/1818-1015-2021-1-104-119

[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, 2021
After revision February 12, 2021
Accepted March 12, 2021

The article is written in support of the educational discipline "Non-classical logics". Within the framework of this discipline, the objects of study are the basic principles and constructive elements, with the help of which the formal construction of various non-classical propositional logics takes place. Despite the abstractness of the theory of non-classical logics, in which the main attention is paid to the strict mathematical formalization of logical reasoning, there are real practical areas of application of theoretical results. In particular, languages of temporal modal logics are widely used for modeling, specification, and verification (correctness analysis) of logic control program systems. This article demonstrates, using the linear temporal logic LTL as an example, how abstract concepts of non-classical logics can be reflected in practice in the field of information technology and programming. We show the possibility of representing the behavior of a software system in the form of a set of LTL-formulas and using this representation to verify the satisfiability of program system properties through the procedure of proving the validity of logical inferences, expressed in terms of the linear temporal logic LTL. As program systems, for the specification of the behavior of which the LTL logic will be applied, Minsky counter machines are considered. Minsky counter machines are one of the ways to formalize the intuitive concept of an algorithm. They have the same computing power as Turing machines. A counter machine has the form of a computer program written in a high-level language, since it contains variables called counters, and conditional and unconditional jump operators that allow to build loop constructions. It is known that any algorithm (hypothetically) can be implemented in the form of a Minsky three-counter machine.

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 Counter Machines", *Modeling and analysis of information systems*, vol. 28, no. 1, pp. 104-119, 2021.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 28, № 1, 2021
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

LTL-спецификация счётчиковых машин

Е. В. Кузьмин[1]

DOI: 10.18255/1818-1015-2021-1-104-119

[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 2021 г.
После доработки 12 февраля 2021 г.
Принята к публикации 12 марта 2021 г.

Статья написана в поддержку учебной дисциплины “Неклассические логики”. В рамках этой дисциплины объектами изучения являются базовые принципы и конструктивные элементы, с помощью которых происходит формальное построение различных неклассических логик высказываний. Несмотря на абстрактность теории неклассических логик, в которой основное внимание уделяется строгой математической формализации логических рассуждений, существуют реальные прикладные области применения теоретических результатов. В частности, языки темпоральных модальных логик широко используются для моделирования, спецификации и верификации (анализа корректности) программных систем логического управления. В этой статье на примере линейной темпоральной логики LTL демонстрируется, как абстрактные понятия неклассических логик могут находить отражение на практике в области информационных технологий и программирования. Показывается возможность представления поведения программной системы в виде набора LTL-формул и использования этого представления для проверки выполнимости программных свойств системы через процедуру доказательства справедливости логических выводов, выраженных в терминах линейной темпоральной логики LTL. В качестве программных систем, для спецификации поведения которых будет применяться логика LTL, рассматриваются счётчиковые машины Минского. Счётчиковые машины Минского — один из способов формализации интуитивного понятия алгоритма. Они обладают той же вычислительной мощностью, что и машины Тьюринга. Счётчиковая машина имеет вид компьютерной программы, написанной на языке высокого уровня, поскольку содержит переменные, называемые счётчиками, и операторы условного и безусловного перехода, позволяющие строить конструкции циклов. Известно, что любой алгоритм (гипотетически) может быть реализован в виде трёхсчётчиковой машины Минского.

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 Counter Machines”, *Modeling and analysis of information systems*, vol. 28, no. 1, pp. 104-119, 2021.

Введение

Статья написана в поддержку учебной дисциплины «Неклассические логики» [1, 2]. В рамках этой дисциплины объектами изучения являются базовые принципы и конструктивные элементы, с помощью которых происходит формальное построение различных неклассических логик высказываний. В качестве основных конструктивных элементов рассматриваются возможные миры (как ключевой инструмент выхода за пределы классической логики высказываний), отношение достижимости между мирами и модальные операторы. Логики определяются относительно семантического понятия *логического следования*, отвечающего на вопрос, каким образом из набора предпосылок Σ выводится заключение A.

Несмотря на абстрактность теории неклассических логик, в которой основное внимание уделяется строгой математической формализации логических рассуждений, существуют реальные прикладные области применения теоретических результатов. Например, языки темпоральных модальных логик широко используются для моделирования, спецификации и верификации (анализа корректности) программных систем логического управления.

Цель статьи — наглядно на примере линейной темпоральной логики LTL продемонстрировать, как абстрактные понятия неклассических логик могут находить отражение на практике в области информационных технологий и программирования. В частности, в данной работе будет показана возможность представления поведения программной системы в виде набора LTL-формул и использования этого представления для проверки выполнимости программных свойств системы через процедуру доказательства справедливости логических выводов, выраженных в терминах линейной темпоральной логики LTL.

В статье в качестве программных систем, для спецификации поведения которых будет применяться логика LTL, рассматриваются счётчиковые машины Минского.

Счётчиковые машины Минского [3—5] — это один из способов формализации интуитивного понятия алгоритма. Они обладают той же вычислительной мощностью, что и машины Тьюринга. Однако счётчиковая машина имеет более наглядный программный вид, т. е. вид компьютерной программы, написанной на языке высокого уровня, поскольку содержит переменные, называемые счётчиками, и операторы условного и безусловного перехода, позволяющие строить конструкции циклов. Известно, что произвольная машина Тьюринга моделируется машиной Минского, которая имеет всего лишь три счётчика. Это означает, что для реализации любого алгоритма (гипотетически) достаточно формализма трёхсчётчиковых машин Минского. Более того, для каждой машины Минского может быть построена моделирующая ее работу двухсчётчиковая машина при использовании специальной кодировки входа и выхода [3].

1. Предварительные сведения

1.1. Счётчиковые машины Минского

Счётчиковая машина Минского M представляет собой набор (q_0, q_n, Q, V, Δ), где $Q = \{q_0, \dots, q_n\}$ — конечное непустое множество состояний машины; $q_0 \in Q$ — начальное состояние; $q_n \in Q$ — финальное состояние; $V = \{x_1, \dots, x_m\}$ — 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. Состояния q_i, $0 \leqslant i \leqslant n-1$, подразделяются на два типа. Состояния первого типа имеют правила переходов вида:

$$(\delta_i)\ \ q_i:\ \ x_j := x_j + 1;\ \text{goto}\ q_k,$$

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 правило перехода не предусмотрено. Это означает, что при попадании в состояние q_n машина Минского M завершает свою работу.

Конфигурация счётчиковой машины Минского — 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 машины Минского можно представить в виде вектора значений соответствующего набора переменных

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

Исполнением машины Минского 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 зацикливается, реализуя тем самым частичную числовую функцию.

Добавим к описанию счётчиковой машины отображение $\mathrm{bnd}\colon V \longrightarrow \mathbb{N}$, устанавливающее предельное значение $\mathrm{bnd}(x)$ для каждого счётчика $x \in V$. Добавив ограничивающее условие, изменим правило переходов первого типа следующим образом, $1 \leqslant j \leqslant m, 0 \leqslant k \leqslant n$:

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

В результате получили счётчиковую машину с ограничениями. Если при срабатывании правила переходов первого типа оказывается, что значение соответствующего счётчика достигло предельного значения, работа машины останавливается. Далее в статье под счётчиковой машиной будет пониматься счётчиковая машина с ограничениями.

1.2. Счётчиковая машина возведения числа в квадрат

Рассмотрим в качестве примера трехсчётчиковую машину Минского $3cM$ с ограничениями, которая реализует функцию возведения числа n в квадрат, где $0 \leqslant n \leqslant 10$. Эта счётчиковая машина имеет восемь состояний $q_0, q_1, \dots, q_7$, где q_0 является начальным состоянием, q_7 — финальное состояние. Множество счётчиков $V = \{a, b, c\}$. 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 $\mathrm{bnd}(a) = 11$, $\mathrm{bnd}(b) = 11$ и $\mathrm{bnd}(c) = 101$. Правила переходов по состояниям машины $3cM$ представлены ниже [4]:

$(\delta_0)\ \ q_0:\ \ \text{if } a > 0 \text{ then } (a := a - 1;\ \text{goto } q_1) \text{ else goto } q_7;$
$(\delta_1)\ \ q_1:\ \ \text{if } c < \mathrm{bnd}(c) \text{ then } c := c + 1;\ \text{goto } q_2;$
$(\delta_2)\ \ q_2:\ \ \text{if } a > 0 \text{ then } (a := a - 1;\ \text{goto } q_3) \text{ else goto } q_5;$
$(\delta_3)\ \ q_3:\ \ \text{if } b < \mathrm{bnd}(b) \text{ then } b := b + 1;\ \text{goto } q_4;$
$(\delta_4)\ \ q_4:\ \ \text{if } c < \mathrm{bnd}(c) \text{ then } c := c + 1;\ \text{goto } q_1;$
$(\delta_5)\ \ q_5:\ \ \text{if } b > 0 \text{ then } (b := b - 1;\ \text{goto } q_6) \text{ else goto } q_0;$
$(\delta_6)\ \ q_6:\ \ \text{if } a < \mathrm{bnd}(a) \text{ then } a := a + 1;\ \text{goto } q_5.$

При построении этой счётчиковой машины использовался тот факт, что $n^2 = (2n-1)+(2n-3)+\dots+3+1$.

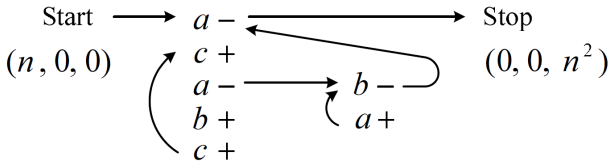

Fig. 1. Graphical representation of the counter machine $3cM$ of squaring a number

Рис. 1. Графическое представление счётчиковой машины $3cM$ возведения числа в квадрат

Правила переходов трехсчётчиковой машины $3cM$ наглядно представлены в графическом виде на рис. 1, где для некоторой переменной $v \in V$ обозначение «$v+$» соответствует увеличению счётчика на единицу, а «$v-$» используется для обозначения условного вычитания единицы с переходом в другое состояние по правосторонней стрелке в случае нулевого значения счётчика v.

2. 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 нашла 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].

Прежде чем рассматривать LTL в качестве программной логики, дадим её определение как некоторой абстрактной модальной темпоральной логики, являющейся одним из представителей внушительного ряда неклассических логик [2].

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), 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 $\neg\varphi$, $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \Longrightarrow \psi)$, $\mathbf{X}\,\varphi$, $(\varphi\,\mathbf{U}\,\psi)$, $\mathbf{F}\,\varphi$ 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 $w \overset{*}{\longrightarrow} w'$ транзитивную достижимость мира w' из мира w за ноль или более шагов (применений функции T). Другими словами $w \overset{*}{\longrightarrow} w'$ означает, что $T^i(w) = w'$, где $T^i(w) = T(T^{i-1}(w))$

и $T^0(w) = w$ при $i \in \mathbb{N} \cup \{0\}$. 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 $\neg$, $\wedge$, $\vee$, $\Longrightarrow$ 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 $\nu_w(\varphi \Longrightarrow \psi) = 0$.

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 $\forall\, w' \in W$ таких, 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 $w \xrightarrow{i} w'$ и $\nu_{w'}(\psi) = 1$, а $\forall\, w'' \in W$ таких, что $w \xrightarrow{j} w''$, имеем $\nu_{w''}(\varphi) = 1$, где $j < i$ и $i \in \mathbb{N} \cup \{0\}$, 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 $\mathbf{F}\varphi = \text{true}\,\mathbf{U}\,\varphi$.

Семантика темпоральных модальных операторов схематично поясняется на рис. 2.

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 $\varphi_1, \varphi_2, \ldots, \varphi_m$ (предпосылок), где $m \in \mathbb{N}$, логически выводится формула ψ (заключение), используется металингвистический символ «$\vDash$». 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. $\Sigma \vDash \psi$, тогда и только тогда, когда не существует такой интерпретации, при которой все формулы из Σ в начальном мире являются

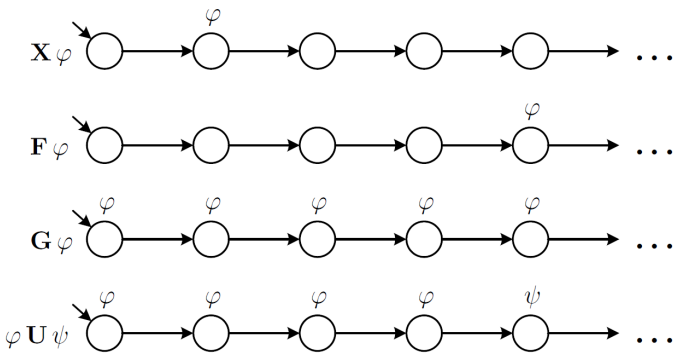

Fig. 2. Semantics of the temporal modal operators **X**, **F**, **G**, and **U**

Рис. 2. Семантика темпоральных модальных операторов **X**, **F**, **G** и **U**

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, ν) для мира $w_0 \in W$ выполняется условие, что если $\nu_{w_0}(\varphi) = 1$ для всех предпосылок $\varphi \in \Sigma$, то $\nu_{w_0}(\psi) = 1$.

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, ν), 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 контрмоделью.

3. LTL как программная логика

Пусть Cl(3cM) — это класс счётчиковых машин с восемью булевыми переменными-состояниями q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, тремя переменными-счётчиками a, b, c и соответствующими ограничениями bnd(a), bnd(b) и bnd(c). Тогда применительно к этому классу трёхсчётчиковых машин Cl(3cM) линейная темпоральная логика LTL может быть рассмотрена следующим образом.

Сопоставим возможному миру $w \in W$ логики LTL конфигурацию некоторой счётчиковой машины из класса Cl(3cM) на некотором шаге её *исполнения*. Тогда каждый возможный мир будет соответствовать вектору значений переменных $(q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, a, b, c)$. При этом разные миры могут иметь одни и те же наборы значений указанных переменных.

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 новый мир w', который будет соответствовать прежней конфигурации счётчиковой машины.

Поскольку интерпретация в логике LTL представляет собой бесконечную последовательность миров, связанных между собой функцией переходов, каждый мир (кроме первого по порядку) в этой последовательности в конкретный момент времени имеет своего непосредственного предшественника. Расширим множество переменных, значения которых будут отслеживаться в возможных мирах. Пусть в интерпретации в некоторый момент времени мир w является непосредственным предшественником мира w'. Для каждой переменной $v \in \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, a, b, c\}$ введём ещё одну переменную $\_v$ для хранения в мире w' предыдущего значения v, т. е. того значения, которое переменная v имела в мире w. Тогда каждый мир может быть представлен в виде вектора значений переменных $(q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, a, b, c, \_q_0, \_q_1, \_q_2, \_q_3, \_q_4, \_q_5, \_q_6, \_q_7, \_a, \_b, \_c)$, который будет

уникальным, т. е. два мира с одним и тем же набором значений соответствующих переменных считаются одним и тем же миром.

Для определённости положим, что в начальном мире любой интерпретации все переменные вида $\_v$ имеют значение 0.

Отметим, что при таком представлении возможных миров в виде векторов значений указанных переменных для класса Cl(3cM) трёхсчётчиковых машин с ограничениями множество миров W в каждой интерпретации (w_0, W, T, ν) логики LTL будет конечным. Этот факт обеспечивает возможность проверки справедливости всех логических выводов в рамках логики LTL, рассмотренной применительно к классу счётчиковых машин Cl(3cM).

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 $\nu_w(p)$ 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 Cl(3cM) 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, ν), это будет означать, что интерпретация (w_0, W, T, ν) 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 $\Sigma = \{\varphi_1, \ldots, \varphi_m\}$, где $m \in \mathbb{N}$, описывает все возможные исполнения некоторой счётчиковой машины M из класса Cl(3cM), у которой q_0 — это начальное состояние, а состояние q_7 является финальным. Рассмотрим LTL-формулу $\psi = «q_0 \Longrightarrow \mathbf{F}\,\mathbf{G}\,q_7»$. 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 $q_7 = 1$. Тогда справедливость логического вывода $\varphi_1, \ldots, \varphi_m \vDash \psi$ будет означать, что счётчиковая машина M, стартуя из состояния q_0 при любых начальных значениях счётчиков, обязательно завершит свою работу в финальном состоянии q_7, т. е. машина M никогда не зацикливается, но при этом всегда срабатывает некоторое правило переходов до тех пор, пока машина не окажется в финальном состоянии. Если же имеет место $\varphi_1, \ldots, \varphi_m \nvDash \psi$, 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 $\varphi_1, \ldots, \varphi_m, \neg\psi$ 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.

4. LTL-спецификация счётчиковой машины с ограничениями

Основная идея представления поведения 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.

Для любой переменной-состояния или переменной-счётчика изменение её значения 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 переменная не меняет своего значения во время работы счётчиковой машины.

Для описания ситуаций, приводящих к увеличению и уменьшению значения целочисленной переменной-счётчика v используются LTL-формулы вида

$$\mathbf{G}\,\mathbf{X}(\ v > \_v \Longrightarrow \mathit{FiringCond}\ \wedge \_v < \mathrm{bnd}(v) \wedge v = \_v + 1\ );$$

$$\mathbf{G}\,\mathbf{X}(\ v < \_v \Longrightarrow \mathit{FiringCond}' \wedge \_v > 0 \qquad \wedge v = \_v - 1\ ).$$

Символ лидирующего подчеркивания «_» в обозначении переменной $\_v$ можно воспринимать как псевдооператор, позволяющий обратиться к значению переменной v, которое она имела в предыдущей конфигурации счётчиковой машины. При этом псевдооператор может использоваться только под действием темпорального оператора **X**.

Выражения $\mathit{FiringCond}$ и $\mathit{FiringCond}'$ описывают состояния счётчиковой машины, при которых возникает необходимость изменения значения переменной v, если это, конечно, допускается соответствующим условием $\_v < \mathrm{bnd}(v)$ или $\_v > 0$.

Для спецификации поведения булевой переменной-состояния v будут использоваться LTL-формулы другого вида

$$\mathbf{G}\,\mathbf{X}(\ \neg\_v \wedge\ \ v \Longrightarrow \mathit{FiringCond}\ );$$

$$\mathbf{G}\,\mathbf{X}(\ \ \_v \wedge \neg v \Longrightarrow \mathit{FiringCond}'\ ).$$

Первая формула означает, что всякий раз, когда состояние v счётчиковой машины активируется, из этого следует, что было выполнено условие $\mathit{FiringCond}$ перехода в состояние v, т. е. сработало некоторое правило переходов. Выражение $\mathit{FiringCond}'$ во второй формуле описывает условия выхода из состояния v.

Третья LTL-формула, описывающая сохранение прежнего значения для счётчика v, имеет вид

$$\mathbf{G}\,\mathbf{X}(\ v = \_v\ \Longrightarrow\ \neg(\mathit{FiringCond} \wedge \_v < \mathrm{bnd}(v)) \wedge \neg(\mathit{FiringCond}' \wedge \_v > 0)\ ).$$

Для булевой переменной-состояния v аналогичная LTL-формула выглядит как

$$\mathbf{G}\,\mathbf{X}(\ v = \_v\ \Longrightarrow\ \neg(\neg\_v \wedge \mathit{FiringCond}) \wedge \neg(\_v \wedge \mathit{FiringCond}')\ ).$$

Итак, опираясь на формальное определение счётчиковой машины с ограничениями, запишем в общем виде требуемое поведение каждой её переменной с помощью тройки LTL-формул.

Начнём с описания поведения переменных-счётчиков. LTL-формула, учитывающая ситуации, которые приводят к увеличению значения переменной-счётчика `xj`, имеет следующий вид:

$$\mathbf{G}\,\mathbf{X}(\ \texttt{xj} > \texttt{\_xj}\ \Longrightarrow\ (\texttt{\_qi} \vee \texttt{\_qr} \vee \cdots \vee \texttt{\_qs}) \wedge \texttt{\_xj} < \texttt{bnd(xj)} \wedge \texttt{xj} = \texttt{\_xj+1}\ ),$$

где переменные `_qi`, `_qr`, ... , `_qs` соответствуют таким состояниям первого типа $q_i, q_r, \ldots, q_s$, в правилах переходов которых участвует счётчик x_j.

Например, для состояния первого типа q_i в описании счётчиковой машины должно быть правило переходов (δ_i) q_i : if $x_j < \mathrm{bnd}(x_j)$ then $x_j := x_j + 1$; goto q_k, которое порождает условие LTL-спецификации вида `_qi` ∧ `_xj<bnd(xj)` ∧ `xj=_xj+1`.

Если в описании машины для счётчика x_j правил перехода первого типа нет, то LTL-формула «возрастания» для программной переменной xj следующая:

$$\mathbf{G}\,\mathbf{X}(\ \texttt{xj} > \texttt{\_xj} \implies \text{false}\ ).$$

LTL-формула, учитывающая ситуации, которые приводят к уменьшению значения переменной-счётчика xj, имеет вид

$$\mathbf{G}\,\mathbf{X}(\ \texttt{xj} < \texttt{\_xj} \implies (\texttt{\_qk} \vee \texttt{\_ql} \vee \cdots \vee \texttt{\_qt}) \wedge \texttt{\_xj}{>}0 \wedge \texttt{xj}{=}\texttt{\_xj}{-}1\ ),$$

где переменные _qk, _ql, ..., _qt соответствуют состояниям второго типа q_k, q_l, ..., q_t, в правилах переходов которых участвует счётчик x_j, т. е. для состояния второго типа q_k в описании машины должно быть правило переходов (δ_k) q_k : if $x_j > 0$ then ($x_j := x_j - 1$; goto q_p) else goto q_h, которое порождает условие LTL-спецификации вида _qk ∧ _xj>0 ∧ xj=_xj−1.

Если в описании машины для счётчика x_j правил перехода второго типа нет, то LTL-формула «убывания» для переменной xj строится просто как

$$\mathbf{G}\,\mathbf{X}(\ \texttt{xj} < \texttt{\_xj} \implies \text{false}\ ).$$

Ситуации, не приводящие к изменению значения переменной-счётчика xj, описываются следующей LTL-формулой, которая строится автоматически по уже рассмотренным двум формулам «возрастания» и «убывания»:

$$\mathbf{G}\,\mathbf{X}(\ \texttt{xj} = \texttt{\_xj} \implies \neg((\texttt{\_qi} \vee \texttt{\_qr} \vee \cdots \vee \texttt{\_qs}) \wedge \texttt{\_xj}{<}\texttt{bnd(xj)}) \wedge \neg((\texttt{\_qk} \vee \texttt{\_ql} \vee \cdots \vee \texttt{\_qt}) \wedge \texttt{\_xj}{>}0)\ ),$$

Эта формула справедлива только в том случае, когда счётчик x_j участвует в правилах переходов обоих типов. Если же для счётчика x_j имеются правила переходов только одного типа, то в этой формуле после оператора импликации будет присутствовать только один соответствующий конъюнктивный член с отрицанием.

Теперь опишем построение LTL-спецификации поведения бинарных переменных-состояний. LTL-формула, учитывающая условия, при которых счётчиковая машина переходит из некоторого текущего состояния в новое состояние q_k, т. е. логическая переменная qk получает значение 1, имеет следующий вид:

$$\mathbf{G}\,\mathbf{X}(\ \neg\texttt{\_qk} \wedge \texttt{qk} \implies \texttt{\_qi} \wedge \texttt{\_xj}{>}0 \vee \cdots \vee \texttt{\_qr} \wedge \neg(\texttt{\_xt}{>}0) \vee \cdots \vee \texttt{\_qs} \wedge \texttt{\_xl}{<}\texttt{bnd(xl)}\ ),$$

где условия вида _qi ∧ _xj>0 соответствуют правилам переходов второго типа

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

а условия вида _qr ∧ ¬(_xt>0) — правилам

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

Условия вида _qs ∧ _xl<bnd(xl) соответствуют правилам переходов первого типа

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

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

Если в состояние q_k счётчиковой машины нет ни одного перехода, то для переменной qk LTL-формула «активации» строится как

$$\mathbf{G}\,\mathbf{X}(\ \neg\texttt{\_qk} \wedge \texttt{qk} \implies \text{false}\ ).$$

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

$\mathbf{G}\,\mathbf{X}(\ \_\mathtt{qk} \wedge \neg\mathtt{qk} \implies \_\mathtt{xj}<\mathtt{bnd(xj)}\ )$.

Для правила переходов второго типа без петель получаем LTL-формулу

$\mathbf{G}\,\mathbf{X}(\ \_\mathtt{qk} \wedge \neg\mathtt{qk} \implies \mathrm{true}\ )$,

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

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

$\mathbf{G}\,\mathbf{X}(\ \_\mathtt{qk} \wedge \neg\mathtt{qk} \implies \mathrm{false}\ )$.

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

$\mathbf{G}\,\mathbf{X}(\ \_\mathtt{qk} \wedge \neg\mathtt{qk} \implies \_\mathtt{xj}>0\ )$ или $\mathbf{G}\,\mathbf{X}(\ \_\mathtt{qk} \wedge \neg\mathtt{qk} \implies \neg(\_\mathtt{xj}>0)\ )$.

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

$\mathbf{G}\,\mathbf{X}(\ \_\mathtt{qk} = \mathtt{qk} \implies \neg(\neg\_\mathtt{qk} \wedge \mathit{FiringCondA}) \wedge \neg(\_\mathtt{qk} \wedge \mathit{FiringCondD})\ )$,

где *FiringCondA* — условие, которое стоит после оператора импликации в LTL-формуле «активации» состояния q_k, а *FiringCondD* — соответствующее условие из формулы деактивации этого состояния.

Далее будет приведена LTL-спецификация поведения конкретной счётиковой машины, а также рассмотрены примеры LTL-свойств, требующих проверки их выполнимости для этой счётчиковой машины.

5. LTL-спецификация трёхсчётчиковой машины возведения числа в квадрат

Построим LTL-спецификацию трехсчётчиковой машины $3cM$ (с ограничениями) возведения числа n в квадрат. В этой LTL-спецификации переменная-счётчик a при инициализации получает значение n, $0 \leqslant n \leqslant 10$. Переменная-состояние q0 инициализируется единицей, т. е. изначально имеем q0 = 1. Остальные переменные при инициализации обнуляются. В LTL-формулах в качестве предельных значений переменных-счётчиков будем использовать конкретные числа: bnd(a) = 11, bnd(b) = 11 и bnd(c) = 101.

$\mathtt{Sinit}$: $\mathtt{a}=\mathtt{n} \wedge \mathtt{b}=0 \wedge \mathtt{c}=0 \wedge \mathtt{q0}=1 \wedge \mathtt{q1}=0 \wedge \mathtt{q2}=0 \wedge \mathtt{q3}=0 \wedge \mathtt{q4}=0 \wedge \mathtt{q5}=0 \wedge \mathtt{q6}=0 \wedge \mathtt{q7}=0$;

$\mathtt{Sa}$: $\mathbf{G}\,\mathbf{X}(\ \mathtt{a}>\_\mathtt{a} \implies \_\mathtt{q6} \wedge \_\mathtt{a}<11 \wedge \mathtt{a}=\_\mathtt{a}+1\ ) \wedge$
$\mathbf{G}\,\mathbf{X}(\ \mathtt{a}<\_\mathtt{a} \implies (\_\mathtt{q0} \vee \_\mathtt{q2}) \wedge \_\mathtt{a}>0 \wedge \mathtt{a}=\_\mathtt{a}-1\ ) \wedge$
$\mathbf{G}\,\mathbf{X}(\ \mathtt{a}=\_\mathtt{a} \implies \neg(\_\mathtt{q6} \wedge \_\mathtt{a}<11) \wedge \neg((\_\mathtt{q0} \vee \_\mathtt{q2}) \wedge \_\mathtt{a}>0)\ )$;

$\mathtt{Sb}$: $\mathbf{G}\,\mathbf{X}(\ \mathtt{b}>\_\mathtt{b} \implies \_\mathtt{q3} \wedge \_\mathtt{b}<11 \wedge \mathtt{b}=\_\mathtt{b}+1\ ) \wedge$
$\mathbf{G}\,\mathbf{X}(\ \mathtt{b}<\_\mathtt{b} \implies \_\mathtt{q5} \wedge \_\mathtt{b}>0 \wedge \mathtt{b}=\_\mathtt{b}-1\ ) \wedge$
$\mathbf{G}\,\mathbf{X}(\ \mathtt{b}=\_\mathtt{b} \implies \neg(\_\mathtt{q3} \wedge \_\mathtt{b}<11) \wedge \neg(\_\mathtt{q5} \wedge \_\mathtt{b}>0)\ )$;

$\mathtt{Sc}$: $\mathbf{G}\,\mathbf{X}(\ \mathtt{c}>\_\mathtt{c} \implies (\_\mathtt{q1} \vee \_\mathtt{q4}) \wedge \_\mathtt{c}<101 \wedge \mathtt{c}=\_\mathtt{c}+1\ ) \wedge$
$\mathbf{G}\,\mathbf{X}(\ \mathtt{c}<\_\mathtt{c} \implies \mathtt{false}\ ) \wedge$
$\mathbf{G}\,\mathbf{X}(\ \mathtt{c}=\_\mathtt{c} \implies \neg((\_\mathtt{q1} \vee \_\mathtt{q4}) \wedge \_\mathtt{c}<101)\ )$;

$\mathtt{Sq0}$: $\mathbf{G}\,\mathbf{X}(\ \neg\_\mathtt{q0} \wedge \ \mathtt{q0} \implies \_\mathtt{q5} \wedge \neg(\_\mathtt{b}>0)\ ) \wedge$
$\mathbf{G}\,\mathbf{X}(\ \ \_\mathtt{q0} \wedge \neg\mathtt{q0} \implies \mathtt{true}\ ) \wedge$
$\mathbf{G}\,\mathbf{X}(\ \ \_\mathtt{q0} = \ \mathtt{q0} \implies \neg(\neg\_\mathtt{q0} \wedge \_\mathtt{q5} \wedge \neg(\_\mathtt{b}>0)) \wedge \neg\_\mathtt{q0}\ )$;

$$
\begin{aligned}
\texttt{Sq1}:\ & \mathbf{G}\mathbf{X}(\ \neg\texttt{\_q1} \wedge\ \texttt{q1}\ \Rightarrow\ \texttt{\_q0} \wedge \texttt{\_a} > 0 \vee \texttt{\_q4} \wedge \texttt{\_c} < 101\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q1} \wedge \neg\texttt{q1}\ \Rightarrow\ \texttt{\_c} < 101\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q1} =\ \texttt{q1}\ \Rightarrow\ \neg(\neg\texttt{\_q1} \wedge (\texttt{\_q0} \wedge \texttt{\_a} > 0 \vee \texttt{\_q4} \wedge \texttt{\_c} < 101)) \wedge \neg(\texttt{\_q1} \wedge \texttt{\_c} < 101)\ ); \\
\texttt{Sq2}:\ & \mathbf{G}\mathbf{X}(\ \neg\texttt{\_q2} \wedge\ \texttt{q2}\ \Rightarrow\ \texttt{\_q1} \wedge \texttt{\_c} < 101) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q2} \wedge \neg\texttt{q2}\ \Rightarrow\ \texttt{true}\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q2} =\ \texttt{q2}\ \Rightarrow\ \neg(\neg\texttt{\_q2} \wedge \texttt{\_q1} \wedge \texttt{\_c} < 101) \wedge \neg\texttt{\_q2}\ ); \\
\texttt{Sq3}:\ & \mathbf{G}\mathbf{X}(\ \neg\texttt{\_q3} \wedge\ \texttt{q3}\ \Rightarrow\ \texttt{\_q2} \wedge \texttt{\_a} > 0\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q3} \wedge \neg\texttt{q3}\ \Rightarrow\ \texttt{\_b} < 11\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q3} = \neg\texttt{q3}\ \Rightarrow\ \neg(\neg\texttt{\_q3} \wedge \texttt{\_q2} \wedge \texttt{\_a} > 0) \wedge \neg(\texttt{\_q3} \wedge \texttt{\_b} < 11)\ ); \\
\texttt{Sq4}:\ & \mathbf{G}\mathbf{X}(\ \neg\texttt{\_q4} \wedge\ \texttt{q4}\ \Rightarrow\ \texttt{\_q3} \wedge \texttt{\_b} < 11\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q4} \wedge \neg\texttt{q4}\ \Rightarrow\ \texttt{\_c} < 101\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q4} =\ \texttt{q4}\ \Rightarrow\ \neg(\neg\texttt{\_q4} \wedge \texttt{\_q3} \wedge \texttt{\_b} < 11) \wedge \neg(\texttt{\_q4} \wedge \texttt{\_c} < 101)\ ); \\
\texttt{Sq5}:\ & \mathbf{G}\mathbf{X}(\ \neg\texttt{\_q5} \wedge\ \texttt{q5}\ \Rightarrow\ \texttt{\_q2} \wedge \neg(\texttt{\_a} > 0) \vee \texttt{\_q6} \wedge \texttt{\_a} < 11\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q5} \wedge \neg\texttt{q5}\ \Rightarrow\ \texttt{true}\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q5} = \neg\texttt{q5}\ \Rightarrow\ \neg(\neg\texttt{\_q5} \wedge (\texttt{\_q2} \wedge \neg(\texttt{\_a} > 0) \vee \texttt{\_q6} \wedge \texttt{\_a} < 11)) \wedge \neg\texttt{\_q5}\ ); \\
\texttt{Sq6}:\ & \mathbf{G}\mathbf{X}(\ \neg\texttt{\_q6} \wedge\ \texttt{q6}\ \Rightarrow\ \texttt{\_q5} \wedge \texttt{\_b} > 0\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q6} \wedge \neg\texttt{q6}\ \Rightarrow\ \texttt{\_a} < 11\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q6} = \neg\texttt{q6}\ \Rightarrow\ \neg(\neg\texttt{\_q6} \wedge \texttt{\_q5} \wedge \texttt{\_b} > 0) \wedge \neg(\texttt{\_q6} \wedge \texttt{\_a} < 11)\ ); \\
\texttt{Sq7}:\ & \mathbf{G}\mathbf{X}(\ \neg\texttt{\_q7} \wedge\ \texttt{q7}\ \Rightarrow\ \texttt{\_q0} \wedge \neg(\texttt{\_a} > 0)\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q7} \wedge \neg\texttt{q7}\ \Rightarrow\ \texttt{false}\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \ \texttt{\_q7} = \neg\texttt{q7}\ \Rightarrow\ \neg(\neg\texttt{\_q7} \wedge \texttt{\_q0} \wedge \neg(\texttt{\_a} > 0))\ ).
\end{aligned}
$$

Рассмотрим LTL-свойства счётчиковой машины $3cM$, которые обязательно должны выполняться для любого её исполнения.

$\texttt{P1}:\ \mathbf{G}(\ \texttt{q7}\ \Rightarrow\ \texttt{c} = \texttt{n} * \texttt{n} \wedge \texttt{a} = 0 \wedge \texttt{b} = 0\ ).$

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.

$\texttt{P2}:\ \mathbf{G}(\ \texttt{a} + \texttt{b} \leqslant \texttt{n}\ ).$

Это свойство требует, чтобы суммарное значение счётчиков a и b было ограничено константой n на протяжении всего исполнения счётчиковой машины $3cM$.

$\texttt{P3}:\ \mathbf{G}(\ \texttt{c} \leqslant \texttt{n} * \texttt{n}\ ).$

Требуется, чтобы значение счётчика c на протяжении всего исполнения машины $3cM$ не выходило за пределы n^2.

$\texttt{P4}:\ \texttt{q0}\ \Rightarrow\ \mathbf{F}\mathbf{G}(\ \texttt{q7}\ ).$

Если счётчиковая машина $3cM$ стартует из состояния q_0, то она обязательно должна остановиться в своём финальном состоянии q_7.

Добавим возможность обращаться в каждом мире логики LTL к значению ещё одной переменной n. Переменная n будет использоваться как константа на протяжении одного исполнения машины $3cM$, т. е. значение n будет одним и тем же во всех мирах одной интерпретации, соответствующей этому исполнению. Чтобы зафиксировать значение переменной n, которое она будет иметь при инициализации счётчиковой машины $3cM$, построим следующую LTL-формулу:

$$
\begin{aligned}
\texttt{Sn}:\ & 0 \leqslant \texttt{n} \wedge \texttt{n} \leqslant 10 \wedge \\
& \mathbf{G}\mathbf{X}(\ \texttt{n} > \texttt{\_n}\ \Rightarrow\ \texttt{false}\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \texttt{n} < \texttt{\_n}\ \Rightarrow\ \texttt{false}\ ) \wedge \\
& \mathbf{G}\mathbf{X}(\ \texttt{n} = \texttt{\_n}\ \Rightarrow\ \texttt{true}\ ).
\end{aligned}
$$

Тогда справедливость логического вывода (в рамках логики LTL)

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

будет означать, что счётчиковая машина $3cM$ для любого n, $0 \leqslant n \leqslant 10$, 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 $0 \leqslant n \leqslant 10$, 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 $c = 2 * n$ при $n > 0$:

$$\texttt{P5} : \ \mathbf{G}(\ \texttt{q7} \wedge \texttt{n} > 0 \implies \neg(\texttt{c} = 2 * \texttt{n})\ ).$$

В этом случае получим, что формула P5 не будет выводиться из приведённых выше посылок

$$\texttt{Sn}, \texttt{Sinit}, \texttt{Sa}, \texttt{Sb}, \texttt{Sc}, \texttt{Sq0}, \texttt{Sq1}, \texttt{Sq2}, \texttt{Sq3}, \texttt{Sq4}, \texttt{Sq5}, \texttt{Sq6}, \texttt{Sq7} \not\models \texttt{P5},$$

так как существует контрпример — интерпретация, при которой все формулы левой части логического вывода являются истинными, а правая часть в виде формулы-заключения P5 становится ложной. Эта интерпретация представляет собой цепочку из 17 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 [7, 9, 10].

6. Проверка справедливости логических выводов

Ниже приводится код на языке программного средства верификации Cadence SMV [9], 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 запоминания предыдущих значений состояний и счётчиков машины $3cM$. С помощью оператора `init` проводится инициализация вспомогательных переменных для начальных миров всех интерпретаций, а оператор `next` позволяет явно задать значение переменной вида $\_v$ в каждом следующем мире текущей интерпретации. Эти операторы устанавливают базовые правила переходов между мирами в рамках интерпретаций логики LTL. Запись формул логики LTL происходит через ключевое слово `assert`.

Проверка выполнимости каждого LTL-свойства P1, P2, P3, P4 и P5 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 P5 выводится в виде таблицы, каждый столбец которой содержит вектор значений всех переменных в соответствующем мире найденной интерпретации.

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

```
module main()
{ /* ------- описание переменных ------- */
   q0,  q1,  q2,  q3,  q4,  q5,  q6,  q7: 0..1;
  _q0, _q1, _q2, _q3, _q4, _q5, _q6, _q7: 0..1;
  a, _a, b, _b, n, _n: 0..15;
  c, _c: 0..127;
  /* ------- инициализация вспомогательных переменных ------- */
  init(_q0):= 0; init(_q1):= 0; init(_q2):= 0; init(_q3):= 0;
  init(_q4):= 0; init(_q5):= 0; init(_q6):= 0; init(_q7):= 0;
  init(_a):= 0; init(_b):= 0; init(_c):= 0; init(_n):= 0;
  /* ------- запоминание предыдущих значений переменных ------- */
  next(_q0):= q0; next(_q1):= q1; next(_q2):= q2; next(_q3):= q3;
  next(_q4):= q4; next(_q5):= q5; next(_q6):= q6; next(_q7):= q7;
  next(_a):= a; next(_b):= b; next(_c):= c; next(_n):= n;
  /* ------- LTL-спецификация ------- */
  Sn: assert /* формулы для константы n */
      0 <= n & n <= 10 &
      G X( n > _n  ->  0 ) &
      G X( n < _n  ->  0 ) &
      G X( n = _n  ->  1 );
  Sinit: assert  /* формула инициализации */
         (a = n) & (b = 0) & (c = 0) &
         (q0 = 1) & (q1 = 0) & (q2 = 0) & (q3 = 0) &
         (q4 = 0) & (q5 = 0) & (q6 = 0) & (q7 = 0);
  Sa: assert  /* формулы для переменной a */
      G X( a > _a  ->  _q6 & _a<11 & a=_a+1 ) &
      G X( a < _a  ->  (_q0 | _q2) & _a>0 & a=_a-1 ) &
      G X( a = _a  ->  ~(_q6 & _a<11) & ~((_q0 | _q2) & _a>0) );
  Sb: assert  /* формулы для переменной b */
      G X( b > _b  ->  _q3 & _b<11 & b=_b+1 ) &
      G X( b < _b  ->  _q5 & _b>0 & b=_b-1 ) &
      G X( b = _b  ->  ~(_q3 & _b<11) & ~(_q5 & _b>0) );
  Sc: assert  /* формулы для переменной c */
      G X( c > _c  ->  (_q1 | _q4) & _c<101 & c=_c+1 ) &
      G X( c < _c  ->  0 ) &
      G X( c = _c  ->  ~((_q1 | _q4) & _c<101) );
  Sq0: assert  /* формулы для переменной q0 */
       G X( ~_q0 &  q0  ->  _q5 & ~(_b>0) ) &
       G X(  _q0 & ~q0  ->  1 ) &
       G X(  _q0 =  q0  ->  ~(~_q0 & _q5 & ~(_b>0)) & ~_q0 );
  Sq1: assert  /* формулы для переменной q1 */
       G X( ~_q1 &  q1  ->  _q0 & _a>0 | _q4 & _c<101 ) &
       G X(  _q1 & ~q1  ->  _c<101 ) &
       G X(  _q1 =  q1  ->  ~(~_q1 & (_q0 & _a>0 | _q4 & _c<101)) & ~(_q1 & _c<101) );
  Sq2: assert  /* формулы для переменной q2 */
       G X( ~_q2 &  q2  ->  _q1 & _c<101 ) &
       G X(  _q2 & ~q2  ->  1 ) &
       G X(  _q2 =  q2  ->  ~(~_q2 & _q1 & _c<101) & ~_q2 );
  Sq3: assert  /* формулы для переменной q3 */
       G X( ~_q3 &  q3  ->  _q2 & _a>0 ) &
       G X(  _q3 & ~q3  ->  _b<11 ) &
       G X(  _q3 =  q3  ->  ~(~_q3 & _q2 & _a>0) & ~(_q3 & _b<11) );
  Sq4: assert  /* формулы для переменной q4 */
       G X( ~_q4 &  q4  ->  _q3 & _b<11 ) &
```

```
      G X(  _q4 & ~q4  ->  _c<101 ) &
      G X(  _q4 =  q4  ->  ~(~_q4 & _q3 & _b<11) & ~(_q4 & _c<101) );
  Sq5: assert  /* формулы для переменной q5 */
      G X( ~_q5 &  q5  ->  _q2 & ~(_a>0) | _q6 & _a<11 ) &
      G X(  _q5 & ~q5  ->  1 ) &
      G X(  _q5 =  q5  ->  ~(~_q5 & (_q2 & ~(_a>0) | _q6 & _a<11)) & ~_q5 );
  Sq6: assert  /* формулы для переменной q6 */
      G X( ~_q6 &  q6  ->  _q5 & _b>0 ) &
      G X(  _q6 & ~q6  ->  _a<11 ) &
      G X(  _q6 =  q6  ->  ~(~_q6 & _q5 & _b>0) & ~(_q6 & _a<11) );
  Sq7: assert  /* формулы для переменной q7 */
      G X( ~_q7 &  q7  ->  _q0 & ~(_a>0) ) &
      G X(  _q7 & ~q7  ->  0 ) &
      G X(  _q7 =  q7  ->  ~(~_q7 & _q0 & ~(_a>0)) );
  /* ------- 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 q0  ->  F G(q7);
  P5: assert G( q7 & n>0  ->  ~(c = 2*n) );
  /* ------- предпосылки и заключения в логических выводах ------- */
  using  /* используя предпосылки */
    Sn, Sinit,
    Sa, Sb, Sc,
    Sq0, Sq1, Sq2, Sq3, Sq4, Sq5, Sq6, Sq7
  prove  /* доказать заключения */
    P1, P2, P3, P4, P5;
}
```

Заключение

В данной статье описывается способ LTL-спецификации поведения уже построенных счётчиковых машин с ограничениями. Однако применяемый для этого подход может быть задействован и для создания новых программ. Действительно, после задания с помощью LTL-формул требуемого поведения программы и проверки на выполнимость ряда ключевых программных свойств по полученной LTL-спецификации может быть построен соответствующий код на языке высокого уровня. Такой подход, например, был применён в работах [11—16] для спецификации, построения и верификации программ логических контроллеров (ПЛК). При этом кроме LTL-спецификации исполнений программы ПЛК и проверяемых программных свойств, на языке логики LTL проводилось описание согласованного поведения датчиков, т. е. накладывались ограничения на возможные входные данные ПЛК-программы.

References

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

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

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

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

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

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

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

[11] E. V. Kuzmin and V. A. Sokolov, “On Construction and Verification of PLC Programs”, *Automatic Control and Computer Sciences*, vol. 47, no. 7, pp. 443–451, 2013.

[12] E. V. Kuzmin and V. A. Sokolov, “Modeling, Specification and Construction of PLC-programs”, *Automatic Control and Computer Sciences*, vol. 48, no. 7, pp. 554–563, 2014.

[13] E. V. Kuzmin, V. A. Sokolov, and D. A. Ryabukhin, “Construction and Verification of PLC-programs by LTL-specification”, *Automatic Control and Computer Sciences*, vol. 49, no. 7, pp. 453–465, 2015.

[14] E. V. Kuzmin, V. A. Sokolov, and D. A. Ryabukhin, “Construction and Verification of PLC LD Programs by the LTL Specification”, *Automatic Control and Computer Sciences*, vol. 48, no. 7, pp. 424–436, 2014.

[15] E. V. Kuzmin, D. A. Ryabukhin, and V. A. Sokolov, “Modeling a Consistent Behavior of PLC-Sensors”, *Automatic Control and Computer Sciences*, vol. 48, no. 7, pp. 602–614, 2014.

[16] E. V. Kuzmin, D. A. Ryabukhin, and V. A. Sokolov, “On the Expressiveness of the Approach to Constructing PLC-programs by LTL-Specification”, *Automatic Control and Computer Sciences*, vol. 50, no. 7, pp. 510–519, 2016.