Н. О. Гаранина, И. С. Ануреев, В. Е. Зюбин, С. М. Старолетов, Т. В. Лях, А. С. Розов, С. П. Горлач, Темпоральная логика для программируемых логических контроллеров, *Модел. и анализ информ. систем*, 2020, том 27, номер 4, 412–427

DOI: 10.18255/1818-1015-2020-4-412-427

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 27, NO. 4, 2020
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

Temporal Logic for Programmable Logic Controllers

N. O. Garanina[1], I. S. Anureev[1], V. E. Zyubin[2], S. M. Staroletov[3], T. V. Liakh[2], A. S. Rozov[2], S. P. Gorlatch[4]

DOI: 10.18255/1818-1015-2020-4-412-427

[1]A. P. Ershov Institute of Informatics Systems (IIS), Siberian Branch of the Russian Academy of Sciences, 6, Acad. Lavrentjev ave., Novosibirsk 630090, Russia.
[2]Institute of Automation and Electrometry SB RAS, 1 Academician Koptyug ave., Novosibirsk 630090, Russia.
[3] Polzunov Altai State Technical University, 46 Lenina ave., Barnaul 656038, Russia.
[4]University of Munster, 2 Schlossplatz, Munster 48149, Germany.

MSC2020: 68N30
Research article
Full text in Russian

Received November 12, 2020
After revision December 2, 2020
Accepted December 16, 2020

We address the formal verification of the control software of critical systems, i.e., ensuring the absence of design errors in a system with respect to requirements. Control systems are usually based on industrial controllers, also known as Programmable Logic Controllers (PLCs). A specific feature of a PLC is a scan cycle: 1) the inputs are read, 2) the PLC states change, and 3) the outputs are written. Therefore, in order to formally verify PLC, e.g., by model checking, it is necessary to describe the transition system taking into account this specificity and reason both in terms of state transitions within a cycle and in terms of larger state transitions according to the scan-cyclic semantics. We propose a formal PLC model as a hyperprocess transition system and temporal cycle-LTL logic based on LTL logic for formulating PLC property. A feature of the cycle-LTL logic is the possibility of viewing the scan cycle in two ways: as the effect of the environment (in particular, the control object) on the control system and as the effect of the control system on the environment. For both cases we introduce modified LTL temporal operators. We also define special modified LTL temporal operators to specify inside properties of scan cycles. We describe the translation of formulas of cycle-LTL into formulas of LTL, and prove its correctness. This implies the possibility of model checking requirements expressed in logic cycle-LTL, by using well-known model checking tools with LTL as specification logic, e.g., Spin. We give the illustrative examples of requirements expressed in the cycle-LTL logic.

Keywords: formal verification; temporal logics; transition systems; programmable logic controllers (PLC).

INFORMATION ABOUT THE AUTHORS

Natalia Olegovna Garanina correspondence author	orcid.org/0000-0001-9734-3808. E-mail: garanina@iis.nsk.su Ph.D. in Mathematics, senior research fellow.
Igor Sergeevich Anureev	orcid.org/0000-0001-9574-128X. E-mail: anureev@iis.nsk.su Ph.D. in Mathematics, senior research fellow.
Vladimir Evgenyevich Zyubin	orcid.org/0000-0002-8198-3197. E-mail: zyubin@iae.nsk.su Dr. Sci., head of laboratory.
Sergey Mikhailovich Staroletov	orcid.org/0000-0001-5183-9736. E-mail: serg_soft@mail.ru Ph.D. in Mathematics, lecturer.
Tatiana Victorovna Liakh	orcid.org/0000-0001-9148-946X. E-mail: liakh@iae.nsk.su research engineer.
Andrey Sergeevich Rozov	orcid.org/0000-0002-7468-0411. E-mail: rozov@iae.nsk.su junior researcher.
Sergei Petrovich Gorlatch	orcid.org/0000-0003-3857-9380. E-mail: gorlatch@uni-muenster.de Prof. Dr. Habil.

Funding: The reported study was funded by State assignment No AAAA-A19-119120290056-0.

For citation: N. O. Garanina, I. S. Anureev, V. E. Zyubin, S. M. Staroletov, T. V. Liakh, A. S. Rozov, and S. P. Gorlatch, "Temporal Logic for Programmable Logic Controllers", *Modeling and analysis of information systems*, vol. 27, no. 4, pp. 412-427, 2020.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 27, № 4, 2020
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

Темпоральная логика для программируемых логических контроллеров

Н. О. Гаранина[1], И. С. Анureев[1], В. Е. Зюбин[2], С. М. Старолетов[3], Т. В. Лях[2], А. С. Розов[2], С. П. Горлач[4]

DOI: 10.18255/1818-1015-2020-4-412-427

[1]Институт систем информатики имени А.П. Ершова СО РАН, пр. Лаврентьева, д. 6, г. Новосибирск, 630090 Россия.
[2]Институт автоматики и электрометрии СО РАН, пр. Акад. Коптюга, д. 1, г. Новосибирск, 630090 Россия.
[3]Алтайский государственный технический университет им. И.И. Ползунова, пр. Ленина, д. 46, г. Барнаул, 656038 Россия.
[4]Университет Мюнстера, Шлосплац, д. 2, Мюнстер, 48149 Германия.

УДК 004.822, 681.51
Научная статья
Полный текст на русском языке

Получена 12 ноября 2020 г.
После доработки 2 декабря 2020 г.
Принята к публикации 16 декабря 2020 г.

Мы исследуем формальную верификацию управляющего программного обеспечения критических систем, т. е. проверку соответствия функционирования проектируемой системы предъявленным требованиям. Важнейший класс управляющего программного обеспечения составляют программы для программируемых логических контроллеров (ПЛК). Особенностью программ ПЛК является цикл управления: 1) считываются входы, 2) изменяются состояния ПЛК и 3) записываются выходы. Поэтому для формальной верификации программ ПЛК нужна возможность описывать учитывающие эту специфику системы переходов, а также определять свойства систем, моделирующих программы ПЛК, как относительно переходов внутри цикла, так и относительно более крупных переходов в соответствии с семантикой цикла управления. Мы предлагаем формальную модель программы ПЛК как систему переходов гиперпроцессов и темпоральную логику cycle-LTL для формализации свойств ПЛК. Особенностью логики cycle-LTL является возможность рассматривать свойства систем управления двояким образом: воздействие окружения на систему управления и воздействие системы управления на окружение. Мы определяем модификации стандартных темпоральных операторов логики LTL для каждого из этих случаев, а также для свойств внутри цикла управления. Рассмотрены примеры требований, определенных в нашей логике. Описана трансляция формул cycle-LTL в формулы LTL и показана её корректность. Доказана возможность сведения задачи верификации методом проверки моделей для требований, определенных в логике cycle-LTL, к задаче верификации требований, определенных в стандартной логике LTL.

Ключевые слова: формальная верификация; темпоральная логика; системы переходов; программируемые логические контроллеры (ПЛК).

ИНФОРМАЦИЯ ОБ АВТОРАХ

Наталья Олеговна Гаранина автор для корреспонденции	orcid.org/0000-0001-9734-3808. E-mail: garanina@iis.nsk.su канд. физ.-мат. наук, с.н.с..
Игорь Сергеевич Анureев	orcid.org/0000-0001-9574-128X. E-mail: anureev@iis.nsk.su канд. физ.-мат. наук, с.н.с..
Владимир Евгеньевич Зюбин	orcid.org/0000-0002-8198-3197. E-mail: zyubin@iae.nsk.su д.т.н., заведующий лабораторией.
Сергей Михайлович Старолетов	orcid.org/0000-0001-5183-9736. E-mail: serg_soft@mail.ru канд. физ.-мат. наук, доцент.
Татьяна Викторовна Лях	orcid.org/0000-0001-9148-946X. E-mail: liakh@iae.nsk.su инженер-исследователь.
Андрей Сергеевич Розов	orcid.org/0000-0002-7468-0411. E-mail: rozov@iae.nsk.su м.н.с.
Сергей Петрович Горлач	orcid.org/0000-0003-3857-9380. E-mail: gorlatch@uni-muenster.de профессор, канд. физ.-мат. наук.

Финансирование: Исследование выполнено в рамках государственного задания № АААА-А19-119120290056-0.

Для цитирования: N. O. Garanina, I. S. Anureev, V. E. Zyubin, S. M. Staroletov, T. V. Liakh, A. S. Rozov, and S. P. Gorlatch, “Temporal Logic for Programmable Logic Controllers”, *Modeling and analysis of information systems*, vol. 27, no. 4, pp. 412-427, 2020.

Введение

В последнее десятилетие, в связи с решением задач, предъявляющих повышенные требования по безопасности к системам управления на основе программируемых логических контроллеров (ПЛК), стали разрабатываться решения по формальному моделированию программ для таких контроллеров и доказательству их свойств, в частности, выражаемых в виде формул темпоральных логик.

Долгосрочной целью нашей работы является формальная верификация программ для систем автоматического управления, написанных в рамках процесс-ориентированной парадигмы, и в частности программ, написанных на процесс-ориентированном языке Reflex [1, 2]. Формальная проверка программ для ПЛК, которые являются важными компонентами систем автоматического управления, является актуальной темой теоретических и практических работ [3, 4]. Предлагаемые подходы следуют различным формальным моделям ПЛК [3—7]. В общем случае, функционирование ПЛК состоит из бесконечной последовательности *циклов управления*. Каждый цикл управления включает в себя последовательность из трех фаз: чтение входов, исполнение и запись выходов.

Мы используем процесс-ориентированное моделирование программ для ПЛК на основе *гиперпроцессов* [2]. Этот метод моделирования позволяет естественно определять основные особенности программ для ПЛК, такие как цикл управления и таймеры; он описывает программу для ПЛК как синхронизированную систему взаимодействующих процессов, определяемых набором функциональных состояний и действий в этих состояниях. Согласно классификации [5], гиперпроцесс моделирует программу для ПЛК, которая абстрагируется от времени исполнения цикла управления[1] и использует входное и выходное управление таймерами. Такое моделирование обеспечивает естественную спецификацию для многопроцессорных систем управления, которая за счёт абстрагирования от времени цикла управления позволяет строго упорядочивать последовательность исполнения процессов в цикле управления. Такая сильная синхронизация процессов без чередования позволяет эффективно использовать формальные методы верификации. Гиперпроцесс является основой для предметно-ориентированного языка Reflex, который показал свою продуктивность в ряде промышленных проектов, в частности, в установке для выращивания монокристаллов кремния методом Чохральского [8] и вакуумной системе для большого солнечного вакуумного телескопа [9].

В этой статье мы в качестве метода формальной верификации используем метод проверки моделей. Поэтому гиперпроцесс представлен как система переходов специального вида, а свойства гиперпроцесса — как формулы темпоральной логики. Система переходов гиперпроцессов близка к параллельной многопоточной системе [10], обогащённой синхронизирующим счётчиком, функциональными состояниями процессов, таймерами и примитивами действий для их изменения. Время для программы ПЛК, определённой как гиперпроцесс, тактируется не только внутри фазы исполнения цикла управления (с каждым изменением значений, характеризующих состояния процессов), но и более крупно – относительно последовательности циклов управления, т.е. при чтении входов/записи выходов.

Логика спецификации требований для программы ПЛК должна позволять формулировать высказывания относительно этих двух типов тактирования. В этой статье мы разрабатываем логику *cycle-LTL*, которая обогащает темпоральную логику LTL [11] цикловыми темпоральными операторами для анализа состояний программы ПЛК в начале и в конце цикла управления, а также внутрицикловыми темпоральными операторами для определения свойств программы ПЛК. В этой логике для некоторого абстрактного алгоритма управления можно выразить следующие свойства: “*Если сработал датчик, то устройство включится в следующем цикле управления*” или “*Если температура выше критического значения, то процесс охлаждения всегда включён.*” Заметим, что если

[1] Среда считается достаточно медленной, чтобы считать нулевым время фаз ввода/вывода и исполнения цикла управления.

включение и выключение процесса охлаждения выполняется после действий других процессов в фазе исполнения цикла управления, то последнее свойство может быть нарушенным внутри этой фазы, но может сохраняться вне её. Этот пример иллюстрирует необходимость использования цикловых темпоральных операторов, в частности, оператора “всегда” $\mathbf{G}^{\mathbf{i}}$.

Приведём обзор формализмов, используемых в современных работах по теме исследования. Вначале было разработано расширение временного автомата – PLC-автомат [12] для описания поведения программ для ПЛК в автоматной форме, и далее работы, в основном, концентрировались на применении логических систем, а не автоматов.

В работе [13] сделана оценка методов верификации программ для ПЛК как в текстовой, так и в графической форме с использованием различных инструментов проверки моделей, и приводятся конкретные типы реальных программ. Также описываются текущие проблемы в этой области, в частности, проблема комбинаторного взрыва, моделирование таймеров, определение свойств для проверки и проблема моделирования цикла исполнения таких программ, которой и посвящена настоящая работа.

Известны методы представления программ ПЛК в виде LTL формул. Согласно подходу [14], значение каждой переменной должно изменяться один раз только в одном месте программы за одно полное выполнение при прохождении рабочего цикла ПЛК. Поэтому изменение значения каждой программной переменной представлено двумя явными и одной неявной LTL-формулами. Таким образом, программирование ПЛК предлагается сводить к построению LTL-спецификации для каждой программной переменной. В этом случае доказывается полнота по Тьюрингу языков стандарта IEC 61131-3 с использованием машин Минского.

В работе [15] представлены способы автоматического майнинга инвариантов с помощью причинно-следственного графа связей событий по работающей программе для ПЛК с целью получения TPTL [16] формул для свойств безопасности. Даётся ремарка, что во время выполнения трудно обеспечить соблюдение правила на основе LTL формулы, которое требует, чтобы за одним действием следовало другое, поскольку отсутствие требуемого события в течение ограниченного времени тестирования не предполагает его отсутствия в более позднее время.

В [17] предложен способ автоматического получения спецификаций по заданным шаблонам LTL формул путём прогона циклов исполнения IEC 61131-3 программ. В работе [18] осуществлён перевод кода ПЛК на входной язык верификатора SMV для проверки моделей согласно выявленным инвариантам.

Разработка предметно-ориентированных расширений темпоральных логик для спецификации требований к промышленным контроллерам началась в последнее время. Обзор по этой теме дан в [19], где, в частности, предложена логика ST-LTL [20]. Основные улучшения по сравнению с LTL – это использование предыдущих значений переменных вместо оператора следующего состояния, а также введение оператора *WhileNScanCycles* для работы со значениями управляющих переменных на N-м шаге, что существенно отличается от нашего подхода и приводит к более сложным спецификациям и доказательствам. В [21] вводится темпоральная логика реального времени MITL[a,b], где все темпоральные модальности ограничены интервалом времени [a,b] и далее логика STL (сигнальная темпоральная логика), основанная на фильтрации сигналов и перехода из действительных чисел в натуральные.

Таким образом, наш подход соответствует современным тенденциям развития подходов к формальной спецификации и верификации программ, которые концентрируются на расширении существующих логик в контексте моделирования цикла управления. Первая версия нашего подхода представлена в работе [22].

Оставшаяся часть статьи организована следующим образом. В разделе 1 мы даём определение системы переходов гиперпроцессов. В разделе 2 описаны синтаксис и семантика темпоральной

логики cycle-LTL. В разделе 3 показано, что любую формулу cycle-LTL можно выразить в языке темпоральной логики LTL, и что задача проверки моделей для cycle-LTL и системы переходов гиперпроцессов разрешима. Раздел 3 содержит заключение и план будущих работ.

1. Система переходов гиперпроцессов

Дадим неформальное описание гиперпроцесса [2]. *Гиперпроцесс* — это упорядоченный набор взаимодействующих процессов, которые выполняются последовательно в заданном порядке, образуя *цикл управления*. Этот цикл начинается с чтения входных данных из среды в системные входные переменные гиперпроцесса и заканчивается записью выходных данных (управляющих воздействий) в соответствующие выходные переменные. Значения выходных переменных вырабатываются в фазе исполнения. Мы считаем, что изменения в данных из окружения происходят достаточно медленно, чтобы все процессы успели сработать в фазе исполнения. Таким образом, один цикл управления можно рассматривать как логическую единицу времени гиперпроцесса. Все переменные гиперпроцесса являются глобальными. Особенностью процессов, образующих гиперпроцесс, являются *функциональные состояния* — метки, обозначающие определённую последовательность действий процесса. Среди функциональных состояний выделяются состояния нормального останова и ошибочного останова, в которых процесс не совершает никаких действий. Действия процессов могут изменять значения переменных гиперпроцесса (кроме входных переменных), ограниченно влиять на функциональные состояния других процессов, а также устанавливать и сбрасывать значения таймера. Действия могут иметь охранные условия, зависящие от переменных гиперпроцесса и функциональных состояний других процессов. Наше определение системы переходов гиперпроцессов основано на описании гиперпроцессов и операционной семантике языка Reflex [1, 2].

Определение 1. *(Система переходов гиперпроцессов, HTS)*
Система переходов гиперпроцессов это пятёрка $H = (P, S, s_{ini}, A, R)$, где

- $P = \{p_1, ..., p_n\}$ — упорядоченное множество процессов;
- S — непустое множество состояний;
- s_{ini} — начальное состояние;
- A — алфавит действий;
- R — отношение помеченных переходов $R : A \longrightarrow 2^{S \times S}$.

Перед определением компонент HTS, опишем элементы гиперпроцессов в целом.

Определение 2. *(Элементы гиперпроцессов)*
Элементами гипероцессов являются переменные, функциональные состояния, действия процессов и таймеры:

Переменные. $V = \{v_1, \dots, v_N\}$ — множество *переменных гиперпроцесса*. Значения переменных в состоянии гиперпроцесса определяются соответствующими функциями $v_i : S \longrightarrow D \cup \{\perp\}$. Мы различаем *входные переменные* V_I, *выходные переменные* V_O и *внутренние переменные* V_L: $V = V_I \cup V_O \cup V_L$.

Функциональные состояния. Для каждого $i \in [1..n]$ множеством *функциональных состояний процесса* p_i является $F_i = \{f_i^1, ..., f_i^{m_i}, stop, err\}$. Выделяются *неактивные* состояния $stop$ и err. Остальные состояния являются *активными*. Значение переменной функционального состояния f_i для каждого процесса p_i в состоянии гиперпроцесса описывается функцией $f_i : S \longrightarrow F_i$.

Действия. В активном функциональном состоянии f_i^j, процесс p_i выполняет действия из множества A. Эти действия формируют *тело функционального состояния*. $L_i^j \in \mathbb{N}$ – это число действий в теле f_i^j. Для каждого процесса a_i – это *счётчик действий*, а его значение – это *позиция*. Значение счётчика действий в состоянии гиперпроцесса является результатом функции $a_i : S \longrightarrow [1..L_i^j] \cup \{\perp\}$. Следующее действие процесса в функциональном состоянии определяется функциями $nxt^j : \mathbb{N} \times S \longrightarrow \mathbb{N} \cup \{\perp\}$. Эти функции неявно задают охранные условия для

действий, поскольку зависят от текущего состояния гиперпроцесса, в частности, от состояния активности других процессов. Если в текущем функциональном состоянии нет следующего действия, т.е. достигнут конец его тела, то $nxt^j(a_i) = \bot$. Функции $an^j : \mathbb{N} \cup \{\bot\} \longrightarrow A$ возвращают имя действия на позиции a_i.

Таймеры. *Таймер* процесса p_i – это переменная t_i, чьи значения в состоянии гиперпроцесса являются результатом функции $t_i : S \longrightarrow \mathbb{N} \cup \{\bot\}$. *Значение таймаута* на позиции a функционального состояния f_i^j – это $N_i^{j_a} \in \mathbb{N}$. Для простоты изложения, здесь мы рассматриваем только процессы, у которых не более одного запуска таймера на каждое функциональное состояние.

Определим компоненты системы переходов гиперпроцессов $H = (P, S, s_{ini}, A, R)$.

Множество состояний S

Состояние $s = (v, sp, pc) \in S$ включает следующие элементы:

- оценка переменных $v = (v_1(s), \dots, v_N(s))$: вектор значений переменных в состоянии s;
- состояние процессов $sp = ((f_1(s), a_1(s), t_1(s)), \dots, (f_n(s), a_n(s), t_n(s)))$, где для каждого процесса p_i в состоянии s, $f_i(s)$ — его текущее функциональное состояние, $a_i(s)$ — счётчик действий в состоянии $f_i(s)$, и $t_i(s)$ — значение его таймера;
- счётчик процессов $pc(s)$ со значениями в $[0..n + 1]$, где 0 зарезервирован для чтения входных данных, а $n + 1$ – для записи выходных данных.

Начальное состояние s_{ini}

$s_{ini} = (v_0, sp_0, pc_0)$, где

- $v_0 = (\bot_1, \dots, \bot_N)$,
- $sp_0 = ((f_1^1, 1, \bot)_1, (stop, \bot, \bot)_2, \dots, (stop, \bot, \bot)_n)$, и
- $pc_0 = 0$.

Алфавит действий A

Алфавит включает действия обновления входных и выходных данных, а также действия процессов: $A = \{inp, out, skip, end, upd, tout, reset, startP, stopP, start, set, next, stop, err\}$.

0. Цикловые действия.

inp – изменение значений входных переменных (чтение входов из окружения).

out – изменение значений выходных переменных (запись выходов в окружение).

1. Служебные действия.

$skip$ – процесс ничего не делает в неактивных состояниях.

end – по завершении действий процесса в активном состоянии, ход передаётся следующему процессу.

2. Действие обновления значений внутренних переменных.

upd – процесс изменяет значения некоторых внутренних переменных.

3. Действия с таймером.

$tout$ – процесс запускает таймер и выполняет действия в текущем состоянии до наступления таймаута;

$reset$ – процесс переустанавливает свой таймер в ноль.

4. Действия с функциональными состояниями.

$startP/stopP$ – процесс переводит целевой процесс p_k в функциональные состояния $f_k^1/stop$;

$start/set$ – процесс переходит в целевое функциональное состояние f_i^1/f;

$stop/err$ – процесс переходит в функциональное состояние $stop/err$;

$next$ – процесс переходит к следующему функциональному состоянию.

Отношение помеченных переходов R

Отношение переходов R задаёт семантику действий процессов и обновления входов. Пусть $i \in [1..n], j \in [1..m_i]$, $a \in [1..L_i^j]$. Мы будем использовать следующую нотацию для действий процессов.

Выражение

$$(f_i = f_i^j, a_i = a, T_i, Tg_i, pc = i) \xrightarrow{act} (y'_1 = new_1, \dots, y'_{m'} = new_{m'})$$

означает, что для всех состояний s, удовлетворяющих предусловию (выражению слева от стрелки), и всех состояний s', удовлетворяющих постусловию (выражению справа от стрелки), верно $R(act) = (s, s')$, и при этом

- $act = an^j(a)$, и $an^j(\perp) \in \{skip, end\}$;
- стартовое состояние s таково, что $pc(s) = i$, $f_i(s) = f_i^j$, $a_i(s) = a$, временное ограничение T_i может быть $t_i(s) \neq \perp$, $t_i(s) = \perp$, $t_i(s) < N_i^j$, или $t_i(s) = N_i^j$, а целевое ограничение Tg_i может быть $tgp_i = k$ или $tgs_i = k$, где tgp_i — это переменная из V_L со значениями в $[1..n]$ для задания целевого процесса, а tgs_i — это переменная в V_L со значениями в $[1..m_i]$ для задания целевого функционального состояния процесса p_i; неупомянутые слева элементы имеют произвольные значения;
- результирующее состояние s' специфицирует изменения элементов гиперпроцесса после действия act: $y_k(s') = new_k$ ($k \in [1..m']$); неупомянутые справа элементы не изменяются.

0. Действия обновления входных и выходных переменных.

Мы используем аналогичную нотацию для определения $R(inp)$ и $R(out)$.

В начале цикла управления значения входных данных считываются во входные переменные из V_I, значение каждого активированного таймера процесса увеличивается на 1, и счётчик процессов смещается к первому процессу:

– $(t_{i_1} \neq \perp, \dots, t_{i_k} \neq \perp, pc = 0) \xrightarrow{inp} (u'_1 = u_1, \dots, u'_K = u_K, t'_{i_1} = t_{i_1} + 1, \dots, t_{i_k} = t_{i_k} + 1, pc' = 1)$.

Действие чтения входных данных — единственное действие в системе, которое, вообще говоря, недетерминировано.

В конце цикла управления значения выходных данных записываются в выходные переменные из V_O, и счётчик процессов смещается на начало цикла управления:

– $(pc = n + 1) \xrightarrow{out} (w'_1 = w_1, \dots, w'_M = w_M, pc' = 0)$

Это действие детерминировано.

В определении отношения переходов R для действий процессов, мы рассматриваем процесс p_i, который выполняет действие номер a_i с именем $an^j(a_i)$ в активном функциональном состоянии $f_i = f_i^j$, или находится в неактивном состоянии.

1. Служебные действия.

Процесс p_i ничего не делает в неактивных состояниях $stop$ и err, и ход передаётся следующему процессу:

– $(f_i = stop, pc = i) \xrightarrow{skip} (pc' = i + 1)$.

– $(f_i = err, pc = i) \xrightarrow{skip} (pc' = i + 1)$.

Когда процесс p_i достигает конца тела активного функционального состояния f_i^j, он переходит к первому действию этого состояния, которое он будет выполнять в следующем цикле управления (если только другой процесс не переведёт его в иное состояние), и ход передаётся следующему процессу:

– $(f_i = f_i^j, a_i = \perp, pc = i) \xrightarrow{end} (a'_i = 1, pc' = i + 1)$.

2. Действие обновления значений переменных процесса.

С помощью этого действия, процесс p_i изменяет значения некоторых внутренних переменных $\{v_1, \dots, v_m\} \subseteq V_L$, и переходит к следующему действию $nxt^j(a)$ текущего функционального состояния:

$-\ (f_i = f_i^j, pc = i) \xrightarrow{upd} (v_1' = d_1, \dots, v_m' = d_m, a_i' = nxt^j(a))$;

Это действие детерминировано.

3. Действия с таймером.

В следующих случаях, процесс p_i запускает таймер t_i (если $t_i = \bot$), переходит к первому действию текущего функционального состояния f_i^j, и ход передаётся следующему процессу:

$-\ (f_i = f_i^j, t_i = \bot, pc = i) \xrightarrow{tout} (a_i' = 1, t_i' = 0, pc' = i + 1)$;

$-\ (f_i = f_i^j, t_i < N_i^j, pc = i) \xrightarrow{tout} (a_i' = 1, pc' = i + 1)$.

В случае наступления события таймаут, процесс p_i останавливает таймер t_i и переходит к следующим действиям текущего функционального состояния:

$-\ (f_i = f_i^j, t_i = N_i^j, pc = i) \xrightarrow{tout} (a_i' = nxt^j(a), t_i' = \bot)$.

Результат действия $reset$ аналогичен результату первого случая действия $tout$:

$-\ (f_i = f_i^j, t_i \neq \bot, pc = i) \xrightarrow{reset} (a_i' = 1, t_i' = 0, pc' = i + 1))$.

4. Действия с функциональными состояниями.

Следующие два действия процесса p_i переводят процесс p_k ($i \neq k$) в состояние $stop$, err или его первое функциональное состояние. Заметим, что модель гиперпроцесса удовлетворяет принципу ограниченной инкапсуляции: процесс не может перевести другой процесс в произвольное функциональное состояние.

$-\ (f_i = f_i^j, tgp_i = k, pc = i) \xrightarrow{startP} (f_k' = f_k^1, a_k' = 1, t_k' = \bot, a_i' = nxt^j(a))$;

$-\ (f_i = f_i^j, tgp_i = k, pc = i) \xrightarrow{stopP} (f_k' = stop, a_k' = \bot, t_k' = \bot, a_i' = nxt^j(a))$;

$-\ (f_i = f_i^j, tgp_i = k, pc = i) \xrightarrow{errP} (f_k' = err, a_k' = \bot, t_k' = \bot, a_i' = nxt^j(a))$;

С помощью следующих действий, процесс p_i устанавливает, что в следующем цикле управления он начнёт работу с первого действия соответствующего функционального состояния, и останавливает таймер:

$-\ (f_i = f_i^j, tgs_i = k, pc = i) \xrightarrow{set} (f_i' = f_i^k, a_i' = 1, t_i' = \bot)$;

$-\ (f_i = f_i^j, pc = i) \xrightarrow{start} (f_i' = f^1, a_i' = 1, t_i' = \bot)$;

$-\ (f_i = f_i^j, pc = i) \xrightarrow{next} (f_i' = f_i^{j+1}, a_i' = 1, t_i' = \bot)$;

Процесс p_i переходит в состояние нормального или ошибочного останова:

$-\ (f_i = f_i^j, pc = i) \xrightarrow{stop} (f_i' = stop, a_i' = \bot, t_i' = \bot)$;

$-\ (f_i = f_i^j, pc = i) \xrightarrow{err} (f_i' = err, a_i' = \bot, t_i' = \bot)$.

Согласно определению отношения переходов, процессы действуют последовательно в текущем цикле управления в порядке, заданном счётчиком процессов. Отметим, что отношение переходов недетерминировано только для действия чтения входных данных inp, которое не является действием какого-либо процесса. Пусть *выходные состояния* s_{out} — это состояния, в которых $pc(s_{out}) = 0$. В этих состояниях выходные переменные получили новые значения. Таким образом, выходные состояния отражают зависимости выходных данных (реакций системы управления) от входных данных (от объекта управления). Определим *входные состояния* s_{inp} как состояния сразу после чтения входов во входные переменные и перед тем, как процессы начали действовать: $R(inp) = (s_{out}, s_{inp})$. Таким образом, входные состояния отражают возможные зависимости входных данных (реакций объекта управления) от выходных данных системы управления на предыдущем цикле управления. В силу детерминизма действий процессов, по заданному входному состоянию однозначно определяет-

ся ближайшее выходное состояние, однако различные входы всё же могут приводить к одному и тому же выходу: например, определённый диапазон входных значений может обрабатываться одинаково. Пусть S_i – это множество входных состояний, а S_o – это множество выходных состояний.

Определим три вида путей в HTS. *Стандартный путь* $\pi = s_0, s_1, \ldots$ – это последовательность состояний $s_j \in S$ такая, что $\forall j \geq 0\ \exists a \in A : R(a) = (s_j, s_{j+1})$. Пусть $\pi(j)$ — это j-е состояние на пути π. *Входной путь* $\sigma = i_0, i_1, \ldots$ – это бесконечная последовательность входных состояний $i_j \in S_i$ такая, что для каждого $j \geq 0$ существует конечный стандартный путь π_j длины n_j с $\pi_j(0) = i_j$ и $\pi_j(n_j) = i_{j+1}$. *Выходной путь* $\rho = o_0, o_1, \ldots$ – это бесконечная последовательность выходных состояний $o_j \in S_o$ такая, что для каждого $j \geq 0$ существует конечный стандартный путь π_j длины n_j с $\pi_j(0) = o_j$ и $\pi_j(n_j) = o_{j+1}$. Произвольный путь $\chi = x_0, x_1, \ldots$ является *подпутём* пути $\chi' = x'_0, x'_1, \ldots$, если $x_0, x_1 \ldots$ является подпоследовательностью $x'_0, x'_1, \ldots$. Соответственно, χ' – это *надпуть* χ. Если путь χ проходит в H, то пишем $\chi \in H$.

Далее в статье χ — это стандартный, входной или выходной путь, π — стандартный путь, ρ — входной путь, σ — выходной путь. Дадим определение дополнительных путей, которые используются при задании семантики темпоральных операторов cycle-LTL.

Определение 3. *(Пути и их элементы)*

1. $\chi(k)$ — k-е состояние на пути χ;
2. χ^k — путь-суффикс χ, такой что $\chi^k(0) = \chi(k)$;
3. i^π_k — номер k-го входного состояния на пути π;
4. o^π_k — номер k-го выходного состояния на пути π;
5. π_ρ – это путь, содержащий заданный путь ρ так, что их начала совпадают, т.е. ρ подпуть π и $\pi_\rho(0) = \rho(0)$ (этот путь единственный, т.к. процессы обрабатывают входные данные детерминировано);
6. π_σ – это пути, содержащие заданный путь σ так, что их начала совпадают, т.е. σ подпуть π и $\pi_\sigma(0) = \sigma(0)$;
7. ρ^π — это входные пути, которые начинаются раньше заданного π, но содержатся в нём, начиная со своего второго состояния, т.е. $\pi = \pi^k_{\rho^\pi}$, где k такое, что $\rho^\pi(1) = \pi(i^\pi_1)$;
8. ρ^σ — это входные пути, надпути которых содержит заданный путь σ с ближайшего выходного состояния, т.е σ подпуть π_{ρ^σ} и $\sigma(0) = \pi_{\rho^\sigma}(o_1^{\pi_{\rho^\sigma}})$;
9. σ^π — это выходной путь, который начинается позже заданного π, но содержится в нём, и его начало – это первое выходное состояние π, т.е. $\pi_{\sigma^\pi} = \pi^k$, где k такое, что $\sigma^\pi(0) = \pi(o^\pi_1)$ (этот путь единственный, т.к. все его состояния лежат на заданном пути π);
10. σ^ρ — это выходной путь, надпуть которого содержит заданный путь ρ с ближайшего предшествующего входного состояния, т.е. σ подпуть π_ρ и $\sigma(0) = \pi_\rho(o_1^{\pi_\rho})$ (этот путь единственный, т.к. процессы для заданных входных данных вырабатывают детерминированный выход).

Рисунок 1 иллюстрирует понятия определения 3.

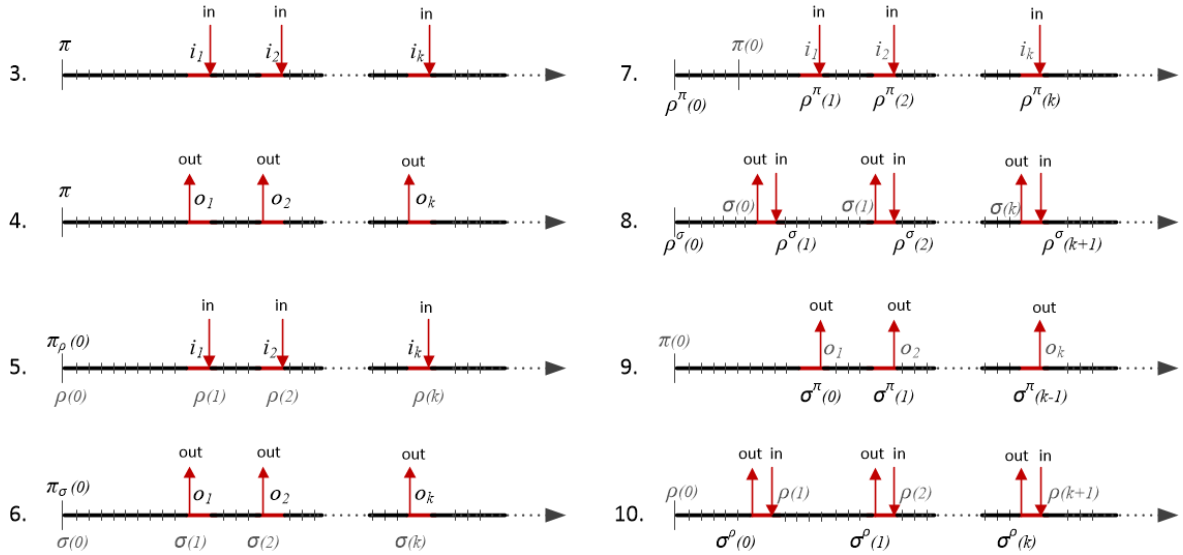

Fig. 1. Paths and their elements

Рис. 1. Пути и их элементы

2. Темпоральная логика cycle-LTL

Синтаксис логики cycle-LTL содержит атомарные высказывания P, булевы связки, стандартные темпоральные операторы LTL, входные, внутренние и выходные цикловые темпоральные операторы:

$$\varphi ::= P \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi \mid \varphi\mathbf{U}\varphi \mid$$
$$\mathbf{X^i}\varphi \mid \mathbf{F^i}\varphi \mid \mathbf{G^i}\varphi \mid \varphi\mathbf{U^i}\varphi \mid \mathbf{X^e}\varphi \mid \mathbf{F^e}\varphi \mid \mathbf{G^e}\varphi \mid \varphi\mathbf{U^e}\varphi \mid \mathbf{X^o}\varphi \mid \mathbf{F^o}\varphi \mid \mathbf{G^o}\varphi \mid \varphi\mathbf{U^o}\varphi$$

Внутренние цикловые темпоральные операторы $\mathbf{X^e}$, $\mathbf{F^e}$, $\mathbf{G^e}$, и $\mathbf{U^e}$ используются для формулирования свойств, которые должны выполняться в течение фазы исполнения циклов управления, тогда как с помощью *входных и выходных цикловых операторов* $\mathbf{X^i}$, $\mathbf{F^i}$, $\mathbf{G^i}$, $\mathbf{U^i}$, $\mathbf{X^o}$, $\mathbf{F^o}$, $\mathbf{G^o}$, и $\mathbf{U^o}$ можно описывать свойства систем управления, которые выполняются в начале и в конце циклов управления.

Пусть Φ^s — это множество формул, начинающихся со стандартных LTL-операторов, Φ^e — множество формул, начинающихся с внутренних операторов, Φ^i — множество формул, начинающихся с входных операторов, Φ^o — множество формул, начинающихся с выходных операторов, и Φ^c — множество всех формул cycle-LTL.

Семантику логики cycle-LTL мы традиционно определяем в терминах отношения выполнимости $\models$ между формулой и путём в HTS: $H, \chi \models \xi$, где H — это система переходов гиперпроцессов, χ — бесконечный стандартный, входной или выходной путь, и $\xi \in \Phi^c$. Мы рассматриваем все виды путей для формул с цикловыми темпоральными операторами, и входные/выходные пути для стандартных LTL-формул. Семантику стандартных LTL-формул на стандартных путях можно посмотреть в [11]. Пусть H — это система переходов гиперпроцессов, π — бесконечный стандартный путь, ρ — входной путь, σ — выходной путь, и $\varphi, \psi \in \Phi^c$ — формулы cycle-LTL.

Семантика формул Φ^s.
Пусть $\xi \in \Phi^s$.

- $H, \rho \models \xi \iff H, \pi_\rho \models \xi$;
- $H, \sigma \models \xi \iff$ для всех π_σ верно $H, \pi_\sigma \models \xi$;

Семантика формул Φ^e.
Пусть $\xi \in \Phi^e$.

- $H, \rho \models \xi \iff H, \pi_\rho \models \xi$;
- $H, \sigma \models \xi \iff$ для всех π_σ верно $H, \pi_\sigma \models \xi$;
- $H, \pi \models \mathbf{X^e}\varphi \iff \pi(1) \notin S_i \cup S_o$ и $H, \pi^1 \models \varphi$;
- $H, \pi \models \mathbf{F^e}\varphi \iff$ существует $0 \le k < o_1^\pi$ такое, что $\pi(k) \notin S_i \cup S_o$ и $H, \pi^k \models \varphi$;
- $H, \pi \models \mathbf{G^e}\varphi \iff$ для всех $0 \le k < o_1^\pi$ $\pi(k) \notin S_i \cup S_o$ и $H, \pi^k \models \varphi$;
- $H, \pi \models \varphi \mathbf{U^e} \psi \iff$ существует $0 \le k < o_1^\pi$ такое что $\pi(k) \notin S_i \cup S_o$ и $H, \pi^k \models \psi$ и для всех $0 \le j < k$ $\pi(j) \notin S_i \cup S_o$ и $H, \pi^j \models \varphi$.

Семантика формул Φ^i.
Пусть $\xi \in \Phi^i$.

- $H, \pi \models \xi \iff$ для всех ρ^π верно $H, \rho^\pi \models \xi$;
- $H, \sigma \models \xi \iff$ для всех ρ^σ верно $H, \rho^\sigma \models \xi$;
- $H, \rho \models \mathbf{X^i}\varphi \iff H, \rho^1 \models \varphi$;
- $H, \rho \models \mathbf{F^i}\varphi \iff$ существует $k \ge 0$ такое, что $H, \rho^k \models \varphi$;
- $H, \rho \models \mathbf{G^i}\varphi \iff$ для всех $k \ge 0$ $H, \rho^k \models \varphi$;
- $H, \rho \models \varphi \mathbf{U^i} \psi \iff$ существует $k \ge 0$ такое что $H, \rho^k \models \psi$ и для всех $0 \le j < k$ $H, \rho^j \models \varphi$.

Семантика формул Φ^o.
Пусть $\xi \in \Phi^o$.

- $H, \pi \models \xi \iff H, \sigma^\pi \models \xi$;
- $H, \rho \models \xi \iff H, \sigma^\rho \models \xi$;
- $H, \sigma \models \mathbf{X^o}\varphi \iff H, \sigma^1 \models \varphi$;
- $H, \sigma \models \mathbf{F^o}\varphi \iff$ существует $k \ge 0$ такое, что $H, \sigma^k \models \varphi$;
- $H, \sigma \models \mathbf{G^o}\varphi \iff$ для всех $k \ge 0$ $H, \sigma^k \models \varphi$;
- $H, \sigma \models \varphi \mathbf{U^o} \psi \iff$ существует $k \ge 0$ такое что $H, \sigma^k \models \psi$ и для всех $0 \le j < k$ $H, \sigma^j \models \varphi$.

Проиллюстрируем спецификации на языке cycle-LTL для систем управления на примере тепловентилятора и устройства поддержания температуры:

1. *Если руки появились под сушилкой, она будет включена в ближайшем цикле управления*:
   $\mathbf{G^i}(hands = on \rightarrow \mathbf{X^i} dryer = on)$.
2. *Если температура выше максимального значения, то процесс охлаждения всегда включён*:
   $\mathbf{G^i}(temp > 95^\circ \rightarrow cooler = on)$.
3. *Нагревательный элемент включён, если температура опустилась ниже минимального значения*:
   $\mathbf{G^o}(temp < 5^\circ \rightarrow heater = on)$.
4. *Если включён нагревательный элемент, то со временем температура поднимется выше минимального значения*:
   $\mathbf{G^o}(heater = on \rightarrow \mathbf{F^i}\ temp \ge 5^\circ)$.

Отметим вариативность синтаксиса cycle-LTL при описании свойств гиперпроцессов. В формулах примеров 1 и 3 можно использовать как оператор $\mathbf{G^i}$ (в комплекте с $\mathbf{X^i}$), так и $\mathbf{G^o}$: в обоих случаях задаются свойства зависимости значений ближайшего выхода от входных данных. Выбор того или иного представления свойства системы зависит от предпочтений пользователя.

Вышеприведённые примеры свойств систем используют только циклические состояния и наблюдаемые входные-выходные переменные системы. Такого рода формулы могут использоваться для задания высокоуровневых свойств систем и алгоритмов управления, независимых от реализации, когда система управления рассматривается как чёрный ящик. Однако, если для некоторой реализации системы управления не выполнилось высокоуровневое свойство, то для дальнейшего анализа может понадобиться проверить низкоуровневые свойства системы, сформулированные в терминах состояний и действий процессов, её реализующих. Эти свойства системы определяются

в виде гипотез, использующих внутрицикловые операторы логики, что позволяет локализовать ошибку внутри исполнительной фазы цикла управления. В ряде случаев проверка таких гипотез менее трудозатратна, чем анализ контрпримера для высокоуровневого свойства. Рассмотрим систему, комбинирующую систему управления освещением и охранную сигнализацию. Следующие два свойства этой системы иллюстрируют причинно-следственную связь между низкоуровневыми гипотезами и высокоуровневыми свойствами: невыполнение низкоуровневой формулы 2 влечёт невыполнение высокоуровневой формулы 1.

1. *Когда обнаружен взлом, все лампы мигают в аварийном режиме*:
   $\mathbf{G^i}(alarm \rightarrow \mathbf{X^i} alarm\_light = on)$.
2. *Если сработал датчик взлома, то процесс внешнего взаимодействия охранной сигнализации в течение ближайшего цикла управления посылает сигнальное сообщение всем связанным системам, включая систему управления освещением*:
   $\mathbf{G^i}(alarm \rightarrow \mathbf{F^e} alarm\_message\_sent)$.

Выбор гипотезы 2 обусловлен предположением, что в случае отправки сигнального сообщения, оно будет вовремя получено и после получения сразу сработает включение аварийного режима работы ламп.

3. Перевод формул логики cycle-LTL в LTL

Мы считаем, что формулы логики cycle-LTL $\xi, \xi' \in \Phi^c$ *эквивалентны* ($\xi \equiv \xi'$), если и только если $H, \chi \vDash \xi \iff H, \chi \vDash \xi'$ для произвольной HTS H и произвольного пути χ. В этом разделе мы покажем, что для формул с цикловыми операторами $\Phi^e \cup \Phi^i \cup \Phi^o$ существуют эквивалентные им стандартные формулы LTL с дополнительными атомарными высказываниями.

Добавим в описание состояний HTS H две служебных булевых переменных $Input$ и $Output$. Пусть переменная $Input$ истинна только во входных состояниях из S_i, а переменная $Output$ истинна только в выходных состояниях из S_o. Тогда для соответствующих атомарных высказываний верно, что $H, \pi \vDash Input$ для всех путей π, таких что $\pi(0) \in S_i$, и $H, \pi \nvDash Input$ для всех путей π, таких что $\pi(0) \notin S_i$, а также $H, \pi \vDash Output$ для всех путей π, таких что $\pi(0) \in S_o$, и $H, \pi \nvDash Output$ для всех путей π, таких что $\pi(0) \notin S_o$. Пусть H — это система переходов гиперпроцессов, π — бесконечный стандартный путь, ρ — входной путь, σ — выходной путь, и $Exe = \neg(Input \vee Output)$. Поскольку доказательства утверждений ниже проводятся индукцией по структуре формулы, то можно считать, что далее подформулы формул с цикловыми операторами φ и ψ являются формулами LTL.

Утверждение 1. *cycle-LTL-формулы с начальными внутренними цикловыми операторами эквивалентны следующим LTL-формулам:*

- $\mathbf{X^e}\varphi \equiv \mathbf{X}(\varphi \wedge Exe)$;
- $\mathbf{F^e}\varphi \equiv (Exe)\mathbf{U}(\varphi \wedge Exe)$;
- $\mathbf{G^e}\varphi \equiv (\varphi \wedge Exe)\mathbf{U}(\neg Exe)$;
- $\varphi \mathbf{U^e} \psi \equiv (\varphi \wedge Exe)\mathbf{U}(\psi \wedge Exe)$.

Доказательство.

Проводится индукцией по структуре формулы. Рассмотрим сначала стандартные пути. Отметим, что $\pi(k) \notin S_i \cup S_o \iff H, \pi^k \vDash \neg(Input \vee Output) \iff H, \pi^k \vDash Exe$.

- $H, \pi \vDash \mathbf{X^e}\varphi \iff (\pi(1) \notin S_i \cup S_o) \bigwedge (H, \pi^1 \vDash \varphi) \iff$
  $(H, \pi^1 \vDash Exe) \bigwedge (H, \pi^1 \vDash \varphi) \iff H, \pi^1 \vDash \varphi \wedge Exe \iff H, \pi \vDash \mathbf{X}(\varphi \wedge Exe)$;
- $H, \pi \vDash \mathbf{F^e}\varphi \iff \exists\, 0 \le k < o_1^\pi : (\pi(k) \notin S_i \cup S_o) \bigwedge (H, \pi^k \vDash \varphi) \iff$
  $\exists\, 0 \le k < o_1^\pi : (H, \pi^k \vDash Exe) \bigwedge (H, \pi^k \vDash \varphi) \overset{def.o_1^\pi}{\iff}$
  $\exists\, 0 \le k < o_1^\pi : (H, \pi^k \vDash \varphi \wedge Exe) \bigwedge \forall\, 0 \le j \le k\ (H, \pi^j \vDash Exe) \iff H, \pi \vDash (Exe)\mathbf{U}(\varphi \wedge Exe)$;
- $H, \pi \vDash \mathbf{G^e}\varphi \iff \forall\, 0 \le k < o_1^\pi : (\pi(k) \notin S_i \cup S_o) \bigwedge (H, \pi^k \vDash \varphi) \iff$
  $\forall\, 0 \le k < o_1^\pi : (H, \pi^k \vDash Exe) \bigwedge (H, \pi^k \vDash \varphi) \bigwedge (H, \pi^{o_1^\pi} \vDash Output) \iff H, \pi \vDash (\varphi \wedge Exe)\mathbf{U}(\neg Exe)$;

- $H, \pi \vDash \varphi \mathbf{U^e} \psi \iff \exists\, 0 \le k < o_1^\pi : (\pi(k) \notin S_i \cup S_o) \bigwedge (H, \pi^k \vDash \psi) \bigwedge$
$\forall\, 0 \le j < k : (\pi(j) \notin S_i \cup S_o) \bigwedge (H, \pi^j \vDash \varphi) \iff$
$\exists\, 0 \le k < o_1^\pi : (H, \pi^k \vDash \psi \wedge Exe) \bigwedge \forall\, 0 \le j < k : (H, \pi^j \vDash \varphi \wedge Exe) \iff H, \pi \vDash (\varphi \wedge Exe)\mathbf{U}(\psi \wedge Exe)$.

Эквивалентность формул Φ^e и вышеприведённым формулам из Φ^s на стандартных путях доказана. Пусть $\xi \in \Phi^e$ и $\xi' \in \Phi^s$ — соответствующая LTL-формула. Покажем эквивалентность этих формул на входных и выходных путях.

- $H, \rho \vDash \xi \iff H, \pi_\rho \vDash \xi \iff H, \pi_\rho \vDash \xi' \iff H, \rho \vDash \xi'$;
- $H, \sigma \vDash \xi \iff \forall \pi_\sigma : H, \pi_\sigma \vDash \xi \iff \forall \pi_\sigma : H, \pi_\sigma \vDash \xi' \iff H, \sigma \vDash \xi'$. ■

Утверждение 2. *cycle-LTL-формулы с начальными входными цикловыми операторами эквивалентны следующим LTL-формулам:*

- $\mathbf{X^i}\varphi \equiv \mathbf{X}(\neg Input \mathbf{U}(Input \wedge \varphi))$;
- $\mathbf{F^i}\varphi \equiv \mathbf{F}(Input \wedge \varphi)$;
- $\mathbf{G^i}\varphi \equiv \mathbf{G}(Input \rightarrow \varphi)$;
- $\varphi \mathbf{U^i} \psi \equiv (Input \rightarrow \varphi)\mathbf{U}(Input \wedge \psi)$.

Доказательство.

Проводится индукцией по структуре формулы. Рассмотрим сначала входные пути.

- $H, \rho \vDash \mathbf{X^i}\varphi \iff H, \rho^1 \vDash \varphi \overset{def.\pi_\rho, Input}{\iff}$
$\forall 1 \le k < i_2^\pi : (H, \pi_\rho^k \vDash \neg Input) \bigwedge (H, \pi_\rho^{i_2^\pi} \vDash Input \wedge \varphi) \iff$
$H, \pi_\rho \vDash \mathbf{X}(\neg Input \mathbf{U}(Input \wedge \varphi)) \iff H, \rho \vDash \mathbf{X}(\neg Input \mathbf{U}(Input \wedge \varphi))$;
- $H, \rho \vDash \mathbf{F^i}\varphi \iff \exists k \ge 0 : H, \rho^k \vDash \varphi \overset{def.\rho}{\iff}$
$\exists k \ge 0 : (H, \rho^k \vDash \varphi) \bigwedge (H, \rho^k \vDash Input) \overset{def.\pi_\rho}{\iff} \exists j \ge 0 : H, \pi_\rho^j \vDash Input \wedge \varphi \iff$
$H, \pi_\rho \vDash \mathbf{F}(Input \wedge \varphi) \iff H, \rho \vDash \mathbf{F}(Input \wedge \varphi)$;
- $H, \rho \vDash \mathbf{G^i}\varphi \iff \forall k \ge 0\ H, \rho^k \vDash \varphi \overset{def.\rho}{\iff}$
$\forall k \ge 0\ (H, \rho^k \vDash \varphi) \bigwedge (H, \rho^k \vDash Input) \overset{def.\pi_\rho, Input}{\iff} \forall j \ge 0\ H, \pi_\rho^j \vDash (Input \rightarrow \varphi)$
$\iff H, \pi_\rho \vDash \mathbf{G}(Input \rightarrow \varphi); \iff H, \rho \vDash \mathbf{G}(Input \rightarrow \varphi)$;
- $H, \rho \vDash \varphi \mathbf{U^i} \psi \iff \exists k \ge 0 : (H, \rho^k \vDash \psi) \bigwedge \forall 0 \le j < k : (H, \rho^j \vDash \varphi) \overset{def.\rho}{\iff}$
$\exists k \ge 0 : (H, \rho^k \vDash Input \wedge \psi) \bigwedge \forall 0 \le j < k : (H, \rho^j \vDash Input \wedge \varphi) \overset{def.\pi_\rho, Input}{\iff}$
$\exists l \ge 0 : (H, \pi_\rho^l \vDash Input \wedge \psi) \bigwedge \forall 0 \le i < l : (H, \pi_\rho^i \vDash Input \rightarrow \varphi) \iff$
$H, \pi_\rho \vDash (Input \rightarrow \varphi)\mathbf{U}(Input \wedge \psi) \iff H, \rho \vDash (Input \rightarrow \varphi)\mathbf{U}(Input \wedge \psi)$.

Эквивалентность формул Φ^i и вышеприведённым формулам из Φ^s на входных путях доказана. Пусть $\xi \in \Phi^i$ и $\xi' \in \Phi^s$ — соответствующая LTL-формула. Покажем эквивалентность этих формул на стандартных и выходных путях. Вывод основан на определении путей π_ρ, ρ^π, σ^π и ρ^σ.

- $H, \pi \vDash \xi \iff \forall \rho^\pi : H, \rho^\pi \vDash \xi \iff \forall \pi'_{\rho^\pi} : H, \pi'_{\rho^\pi} \vDash \xi' \overset{ind.by.struc.\xi'}{\iff} H, \pi \vDash \xi'$;
- $H, \sigma \vDash \xi \iff \forall \rho^\sigma : H, \rho^\sigma \vDash \xi \iff \forall \pi'_{\rho^\sigma} : H, \pi'_{\rho^\sigma} \vDash \xi' \iff \forall \sigma'^{\pi'_{\rho^\sigma}} : H, \sigma'^{\pi'_{\rho^\sigma}} \vDash \xi' \overset{ind.by.struc.\xi'}{\iff} H, \sigma \vDash \xi'$. ■

Утверждение 3. *cycle-LTL-формулы с начальными выходными цикловыми операторами эквивалентны следующим LTL-формулам:*

- $\mathbf{X^o}\varphi \equiv \mathbf{X}(\neg Output \mathbf{U}(Output \wedge \varphi))$;
- $\mathbf{F^o}\varphi \equiv \mathbf{F}(Output \wedge \varphi)$;
- $\mathbf{G^o}\varphi \equiv \mathbf{G}(Output \rightarrow \varphi)$;
- $\varphi \mathbf{U^o} \psi \equiv (Output \rightarrow \varphi)\mathbf{U}(Output \wedge \psi)$.

Доказательство.
Проводится индукцией по структуре формулы. Рассмотрим сначала выходные пути.

- $H, \sigma \models \mathbf{X^i}\varphi \iff H, \sigma^1 \models \varphi \overset{def.\pi_\sigma, Output}{\iff}$
$$\forall\ 1 \le k < o_2^\pi\ (H, \pi_\sigma^k \models \neg Output) \bigwedge (H, \pi_\sigma^{o_2^\pi} \models Output \wedge \varphi) \iff$$
$$H, \pi_\sigma \models \mathbf{X}(\neg Output \mathbf{U}(Output \wedge \varphi)) \iff H, \sigma \models \mathbf{X}(\neg Output \mathbf{U}(Output \wedge \varphi));$$
- $H, \sigma \models \mathbf{F^o}\varphi \iff \exists\ k \ge 0 : H, \sigma^k \models \varphi \overset{def.\sigma}{\iff}$
$$\exists\ k \ge 0 : (H, \sigma^k \models \varphi) \bigwedge (H, \sigma^k \models Output) \overset{def.\pi_\sigma}{\iff}$$
$$\exists\ j \ge 0 : H, \pi_\sigma^j \models Output \wedge \varphi \iff H, \pi_\sigma \models \mathbf{F}(Output \wedge \varphi) \iff$$
$$H, \sigma \models \mathbf{F}(Output \wedge \varphi);$$
- $H, \sigma \models \mathbf{G^o}\varphi \iff \forall\ k \ge 0\ H, \sigma^k \models \varphi \overset{def.\sigma}{\iff}$
$$\forall\ k \ge 0\ (H, \sigma^k \models \varphi) \bigwedge (H, \sigma^k \models Output) \overset{def.\pi_\sigma, Output}{\iff}$$
$$\forall\ j \ge 0\ H, \pi_\sigma^j \models (Output \rightarrow \varphi) \iff H, \pi_\sigma \models \mathbf{G}(Output \rightarrow \varphi) \iff$$
$$H, \sigma \models \mathbf{G}(Output \rightarrow \varphi);$$
- $H, \sigma \models \varphi \mathbf{U^o} \psi \iff \exists\ k \ge 0\ (H, \sigma^k \models \psi) \bigwedge \forall\ 0 \le j < k\ (H, \sigma^j \models \varphi) \overset{def.\sigma}{\iff}$
$$\exists\ k \ge 0 : (H, \sigma^k \models Output \wedge \psi) \bigwedge \forall\ 0 \le j < k\ (H, \sigma^j \models Output \wedge \varphi) \overset{def.\pi_\sigma, Output}{\iff}$$
$$\exists\ l \ge 0 : (H, \pi_\sigma^l \models Output \wedge \psi) \bigwedge \forall\ 0 \le i < l\ (H, \pi_\sigma^i \models Output \rightarrow \varphi) \iff$$
$$H, \pi_\sigma \models (Output \rightarrow \varphi) \mathbf{U}(Output \wedge \psi) \iff H, \sigma \models (Output \rightarrow \varphi) \mathbf{U}(Output \wedge \psi).$$

Эквивалентность формул Φ^o и вышеприведённым формулам из Φ^s на входных путях доказана. Пусть $\xi \in \Phi^o$ и $\xi' \in \Phi^s$ — соответствующая LTL-формула. Покажем эквивалентность этих формул на стандартных и входных путях. Вывод основан на определении путей π_ρ, σ^π, ρ^π и σ^ρ.

- $H, \pi \models \xi \iff H, \sigma^\pi \models \xi \iff H, \pi'_{\sigma^\pi} \models \xi' \overset{ind.by.struc.\xi'}{\iff} H, \pi \models \xi';$
- $H, \rho \models \xi \iff H, \sigma^\rho \models \xi; \iff H, \pi'_{\sigma^\rho} \models \xi' \iff H, \rho'^{\pi'_{\sigma^\rho}} \models \xi' \overset{ind.by.struc.\xi'}{\iff} H, \rho \models \xi'.$ ■

Это утверждение является прямым следствием предыдущих:

Утверждение 4. *Для каждой формулы из $\Phi^e \cup \Phi^i \cup \Phi^o$ существует эквивалентная ей LTL-формула линейно большей длины.*

Задача проверки моделей для формул логики cycle-LTL и систем переходов гиперпроцессов состоит в определении семантики формул cycle-LTL в заданной HTS, т.е. в вычислении множества $\{\chi \mid H, \chi \models \xi'$, где H – HTS, $\chi \in H,\ \xi \in \Phi^c\}$. Отметим, что для любой HTS можно за полиномиальное время построить структуру Крипке, содержащую то же самое множество стандартных путей. В силу этого замечания, утверждения 4 и оценки сложности задачи проверки моделей для LTL [11] верна следующая теорема:

Теорема 1. *Существует алгоритм, решающий задачу проверки моделей для формул cycle-LTL и систем переходов гиперпроцессов, сложность которого полиномиально зависит от размера системы переходов гиперпроцессов, и экспоненциально зависит от длины формулы cycle-LTL.*

Заключение

В этой статье мы разработали систему переходов гиперпроцессов (HTS) для моделирования программного обеспечения программируемых логических контроллеров (ПЛК) и новую логику cycle-LTL для формулирования свойств ПЛК. Разработанная HTS-модель естественным образом отражает особенности программ ПЛК, такие как циклы управления и работу с таймерами. Предложенная темпоральная логика cycle-LTL позволяет описывать свойства программ ПЛК как относительно малых временных шагов внутренней фазы исполнения циклов управления, так и больших временных

шагов собственно циклов управления. Мы описали трансляцию формул логики cycle-LTL в формулы логики LTL и доказали её корректность. Эта трансляция демонстрирует, что формулирование свойств систем управления непосредственно в терминах логики LTL оказывается значительно более сложным и запутанным. Мы показали, что в силу корректности этой трансляции, задача проверки моделей для HTS и cycle-LTL является разрешимой.

Мы планируем использовать изложенные в этой статье результаты для обеспечения корректного перевода процесс-ориентированного языка Reflex в язык Promela, используемый верификатором SPIN[23]. Кроме того, мы будем разрабатывать и реализовывать специальный алгоритм проверки моделей для формул cycle-LTL и HTS. Мы ожидаем, что этот алгоритм во многих случаях будет иметь меньшую временную сложность, чем стандартный алгоритм проверки моделей для LTL, за счет использования таких особенностей HTS, как цикличность, строгий порядок действий процессов и детерминированность этих действий. Мы также планируем изучить возможность использования логики cycle-LTL для формулирования свойств систем переходов более общего вида.

References

[1] I. Anureev, «Operacionnaya semantica annotirovannih Reflex programm», *Modelirovanie i analiz informacionnyh sistem*, vol. 26, no. 6, pp. 181–192, 2019.

[2] I. Anureev, N. Garanina, T. Liakh, A. Rozov, and V. Zyubin, «Towards safe cyber-physical systems: the Reflex language and its transformational semantics», in *IEEE International Siberian Conference on Control and Communications*, IEEE, 2019, pp. 18–20.

[3] E. Brinksma and A. Mader, «Verification and Optimization of a PLC Control Schedule», in *SPIN 2000 – SPIN Model Checking and Software Verification*, ser. LNCS, vol. 1885, Springer, 2000, pp. 73–92.

[4] V. Gourcuff, O. de Smet, and J.-M. Faure, «Improving large-sized PLC programs verification using abstractions», *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 5101–5106, 2008.

[5] A. Mader, «A Classification of PLC Models and Applications», in *Discrete Event Systems*, ser. SECS, vol. 569, Springer, 2000, pp. 239–246.

[6] H. Wan, G. Chen, X. Song, and M. Gu, «Formalization and Verification of PLC Timers in Coq», in *Proc. of 33rd Annual IEEE International Computer Software and Applications Conference*, IEEE, 2009, pp. 315–323.

[7] J. Yoo, S. Cha, and E. Jee, «A Verification Framework for FBD Based Software in Nuclear Power Plants», in *Proc. of 15th Asia-Pacific Software Engineering Conference*, IEEE, 2008, pp. 385–392.

[8] D. Bulavskij, V. Zyubin, N. Karlson, V. Krivoruchko, and V. Mironov, «An Automated Control System for a Silicon Single-Crystal Growth Furnace», *Optoelectronics, instrumentation, and data processing*, vol. 32, no. 2, pp. 25–30, 1996.

[9] P. G. Kovadlo, A. Lubkov, A. Bevzov, and et al., «Automation system for the large solar vacuum telescope», *Optoelectronics, Instrumentation and Data processing*, vol. 52, pp. 187–195, 2016.

[10] A. Gupta, V. Kahlon, S. Qadeer, and T. Touili, *Handbook of Model Checking*. Springer International Publishing, 2018, ch. 18. Model Checking Concurrent Programs, pp. 573–577.

[11] E. M. Clarke, T. A. Henzinger, and H. Veith, *Handbook of Model Checking*. Springer International Publishing, 2018, ch. 1. Introduction to Model Checking, pp. 1–13.

[12] H. Dierks, «PLC-automata: A new class of implementable real-time automata», in *International AMAST Workshop on Aspects of Real-Time Systems and Concurrent and Distributed Software*, vol. 1231, Springer, 1997, pp. 111–125.

[13] T. Ovatman, «An overview of model checking practices on verification of PLC software», *Software & Systems Modeling*, vol. 4, no. 15, pp. 937–960, 2016.

[14] E. Kuzmin, D. Ryabukhin, and V. A. Sokolov, «On the expressiveness of the approach to constructing PLC-programs by LTL-specification», *Automatic Control and Computer Sciences*, vol. 7, no. 50, pp. 510–519, 2016.

[15] M. Zhang, «Towards automated safety vetting of PLC code in real-world plants», in *IEEE Symposium on Security and Privacy*, IEEE, 2019, pp. 522–538.

[16] A. Rajeev and T. Henzinger, «A really temporal logic», *Journal of the ACM*, vol. 41, no. 1, pp. 181–203, 1994.

[17] J. Xiong, «A User-Friendly Verification Approach for IEC 61131-3 PLC Programs», *Electronics*, vol. 4, no. 9, 2020.

[18] B. Beckert, «Regression verification for programmable logic controller software», in *International Conference on Formal Engineering Methods*, vol. 9407, Springer, 2015.

[19] O. Ljungkrantz, «An empirical study of control logic specifications for programmable logic controllers», *Empirical Software Engineering*, vol. 3, no. 19, pp. 655–677, 2014.

[20] O. Ljungkrantz, K. Åkesson, M. Fabian, and C. Yuan, «A formal specification language for PLC-based control logic», in *8th IEEE International Conference on Industrial Informatics*, IEEE, 2010, pp. 1067–1072.

[21] O. Maler and D. Nickovic, «Monitoring temporal properties of continuous signals», *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pp. 152–166, 2004.

[22] N. Garanina, I. Anureev, V. Zyubin, A. Rozov, T. Liakh, and S. Gorlatch, «Reasoning about Programmable Logic Controllers», *System Informatics*, vol. 17, pp. 33–42, 2020.

[23] G. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, 2003.