электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Методы построения и верификации математических моделей систем реального времени

 # 12, декабрь 2014

Андреев А. М., Козлов И. А.

УДК: 519.681.3

Россия, МГТУ им. Н.Э. Баумана
arkandreev@gmail.com

Введение

В повседневной жизни мы постоянно взаимодействуем с устройствами, обрабатывающими информацию. И чем чаще мы используем информационные технологии, тем больше нам приходится полагаться на надежность программного обеспечения. Проблема качества ПО особенно важна для ответственных систем, критичных в отношении безопасности, ошибки в которых могут привести к трагическим последствиям. Основным способом решения этой проблемы является верификация программ [1]. Выделяются два основных подхода к верификации: динамический и статический.

Динамический подход подразумевает проверку программы в процессе исполнения. Основным применением этого подхода является тестирование программ. Однако, как показал Э. Дейкстра [2], тестирование не может гарантировать корректность программы. Но это может сделать формальная верификация.

Формальная верификация позволяет доказать соответствие программы своей спецификации. При этом под спецификацией программы понимаются формализованные требования к ее поведению. Одним из основных направлений в рамках этого подхода является верификация модели программы (model checking) [3]. При этом методе для программы строится формальная конечная модель (т.е. с конечным числом состояний), а проверяемые свойства задаются с помощью формул темпоральной логики. Проверка выполнимости темпоральных формул, задающих свойства модели, происходит автоматическим образом.

В настоящее время существует множество программных средств, таких как SPIN[4] и SMV[5], предоставляющих широкие возможности верификации методом проверки модели.

Однако системы реального времени имеют особенности, которые делают невозможной их проверку с помощью верификаторов общего назначения: при создании таких сис-

тем учитываются ограничения на временные характеристики функционирования, а значит и верификатор должен позволять проверять соответствие системы этим ограничениям. Но программные средства верификации классическим методом проверки модели для спецификации и верификации свойств программ используют темпоральные логики (такие, как LTL и CTL), которые фокусируются на временном порядке событий. Этот временной порядок является качественным понятием, так как время не рассматривается в количественном отношении. С помощью LTL и CTL-формул можно специфицировать логику работы программы, но нельзя задать требования к допустимым временным интервалам между событиями. Поэтому для верификации систем реального времени необходим особый подход, включающий подходящие математические модели и алгоритмы их построения и проверки.

В статье рассматриваются математические модели, адекватно отображающие характеристики систем реального времени, а также методы их построения и последующей верификации.

Проверка моделей для темпоральной логики реального времени

Одним из наиболее важных аспектов является выбор семантики временной области. Например, для синхронных систем, в которых компоненты выполняются в пошаговом режиме, предпочтительна дискретная временная область (известным примером дискретной темпоральной логики является RTTL). Но существуют темпоральные логики, которые позволяют осуществлять проверку моделей в терминах непрерывной временной области (R^+ - неотрицательные вещественные числа). Одна из них - ветвящаяся темпоральная логика реального времени Timed CTL [6]. Базовая идея TCTL состоит в использовании временных ограничений в качестве параметров обычных темпоральных операторов CTL. Эта логика не поддерживает использование темпорального оператора X (поскольку для вещественночисленной временной области понятие следующего момента времени некорректно), а остальные операторы снабжаются ограничениями:

- $\mathbf{F_J}$ p - утверждение p станет верным в какой-то момент, принадлежащий временному интервалу J;
- $\mathbf{G_J}$ p - утверждение p будет верным в течение интервала J;
- p $\mathbf{U_J}$ q - утверждение q станет верным в какой-то момент, принадлежащий временному интервалу J, а до этого в любой момент времени будет верным выражение $p \vee q$.

Также при переходе к системам реального времени нужно ввести новую модель представления исходной программы. Для моделирования таких систем используются временные автоматы - конечные автоматы, снабженные конечным множеством вещественнозначных часовых переменных. Временной автомат представляет собой шестёрку

$$A = (Loc, loc_0, \Sigma, X, Inv, E),$$

где

- Loc - конечное множество локаций;
- loc_0 - начальная локация;
- X - конечное множество локальных часов;
- $\sum$ - конечное множество действий (включая пустое действие);
- $Inv : Loc \to \gamma$ - инвариант (ограничение на возможные значения часов, определённое в некоторых локациях);
- E- множество переходов автомата: $E \subseteq Loc \times \Gamma \times \Sigma \times 2^X \times Loc$.

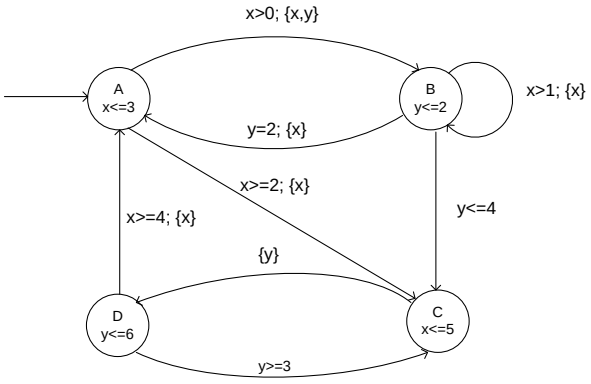

Рисунок 1 - Временной автомат

Основная сложность проверки моделей, в которых временная область непрерывна, состоит в том, что проверяемая модель имеет бесконечно много состояний – для каждого значения времени система может быть в определенном состоянии, а множество этих значений бесконечно. Кажется, что это неразрешимая задача. Фактически, главный вопрос в проверке моделей для темпоральной логики реального времени состоит в том, как справляться с бесконечным множеством состояний. Эта проблема решается путём дискретизации временной области в зависимости от ограничений, накладываемых на автомат, и свойств, которые необходимо проверить.

Построение модели системы реального времени

Состояние временного автомата представляет собой пару $\langle s, v \rangle$, где s - локация (соответствующая позиции в невременном конечном автомате), а v - совокупность значений часов. В связи с этой особенностью используемой модели, построение временного автомата мы будем выполнять в два этапа: на первом этапе по исходной системе будет построена структура Крипке с сохранением инвариантов состояний и ограничений переходов. Затем эта структура будет преобразована во временной автомат, причём его локациями будут состояния модели, полученной на первом этапе, а интерпретации часов должны удовлетворять временным ограничениям.

Первый этап выполняется либо экспертом, осуществляющим построение математической модели, максимально адекватной исходной системе, либо, если это возможно, автоматически. Одним из подходов, обеспечивающих автоматическое построение модели системы, является использование SWITCH-технологии или "автоматного" программирования [7]. Построение структуры Крипке для систем, созданных с использованием такого подхода, описано в работах С.Э. Вельдера и А.А. Шалыто [8].

После получения модели системы реального времени в виде временного автомата $A = (Loc, loc_0, \Sigma, X, Inv, E)$ встает проблема построения графа состояний этого автомата. Графом временного автомата A называется четвёрка $T(A) = (Q, q_0, \Sigma, R)$,

где

- $Q = Loc \times (X \to R^{\geq 0})$ – множество состояний графа, Loc – множество локаций временного автомата, X – множество часов; элементы Q представляют собой пары $\langle loc, v \rangle$, в которой loc - локация, а v -интерпретация часов;
- q_0 - начальное состояние графа $\langle loc_0, v_0 \rangle$, где v_0 - начальная интерпретация всех часов, такая, что для любых $x \in X, v_0(x) = 0$;
- $\sum$ - множество действий. (совпадает с множеством действий автомата A)
- R -множество переходов двух видов:

  а) задержка в локации автомата:

  $\langle loc, v \rangle - \tau \to \langle loc, v' \rangle$ (переход возможен, если значения часов удовлетворяют инварианту состояния);

  б) переход в другую локацию

  $\langle loc, v \rangle \to \langle loc', v' \rangle$ (выполняется при условии удовлетворения значения часов ограничению на переходе; новая интерпретация часов v' получается из v сбрасыванием некоторых часов в 0).

Основной проблемой является бесконечность возможных интерпретаций часов v, которая влечёт за собой бесконечность числа состояний q. Возможным решением этой проблемы является введение отношения эквивалентности [9]. Оно должно быть выбрано так, чтобы эквивалентные состояния были неразличимы с точки зрения проверяемых свойств.

Пусть дана система переходов $T = (Q, q_0, \Sigma, R)$. Введём отношение эквивалентности π, такое, что любые переходы из эквивалентных состояний осуществляются лишь в состояния, которые также эквивалентны относительно данного отношения:

$$s \approx_\pi q \equiv (\forall (s, a, s') \in R)(\exists (q, a, q') \in R) : s' \approx_\pi q'$$

Обозначим новую систему переходов T_π, которая строится следующим образом:

- состояниями T_π являются классы эквивалентности π;
- класс эквивалентности π, включающий q_0, является начальным состоянием T_π ;
- алфавит действий T_π совпадает с алфавитом действий T;

- множество переходов T_π определяется для пар классов эквивалентности π;

Если при бесконечном числе состояний T количество классов эквивалентности π всё же будет конечным, то можно свести задачу верификации исходной системы T к анализу системы переходов T_π, которая называется фактор-системой переходов T по модулю отношения эквивалентности π.

Временные регионы

Можно ли ввести такое отношение эквивалентности, чтобы T_π могла заменить исходную систему переходов при анализе свойств? Простейшим решением является региональное отношение эквивалентности. В его основе лежат следующие соображения:

- $\langle p, v\rangle$ и $\langle q, v'\rangle$ могут принадлежать одному региону, только если p=q;
- $\langle p, v\rangle$ и $\langle q, v'\rangle$ могут быть эквивалентными, только если значения всех часов из v и v' не отличаются целыми частями ([v]=[v']). В обратном случае между этими состояниями могли произойти изменения в значениях временных ограничений.

Если система имеет несколько локальных часов, появляется еще одно условие эквивалентности двух состояний:

- значения всех часов изменяется с одинаковой скоростью , а значит для каждой пары часов отношения между их дробными частями должны совпадать (fr(v)=fr(v')).

На основе этих соображений отношение региональной эквивалентности для нескольких локальных часов определяется так:

Две интерпретации v и v' конечного множества локальных часов X находятся в отношении региональной эквивалентности $\cong$ тогда и только тогда, когда:

- для всех часов $x \in X$,

  то есть $v(x)>C_x \wedge v'(x)>C_x$, то есть в обеих интерпретациях значения всех часов превышают свой потолок, или

- для всех пар часов $(\forall x, y \in X)$, если они не превышают свой потолок $(v(x), v'(x) \le C_x)$ и $(v(y), v'(y) \le C_y)$, то
  - $[v(x)] = [v'(x)] \wedge (fr(v(x)) = 0 \equiv fr(v'(x)) = 0)$, то есть все они имеют совпадающие в обеих интерпретациях целые части, и
  - $(fr(v(x)) \le fr(v(y)) \equiv fr(v'(x)) \le fr(v'(y)))$, то есть для каждой пары часов отношения между их дробными частями в каждой интерпретации совпадают.

Структура регионов для системы с локальными часами x и y и их потолками C_x=2 и C_y=1 изображена на рис. 2.

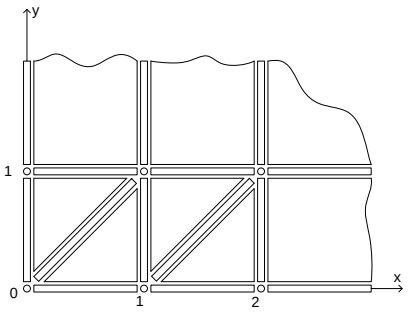

Рисунок 2 - Структура регионов для часов x и y с потолками 2 и 1.

Количество временных регионов определяется следующим соотношением: $|X|!2^{|X|}\prod_{x\in X}(2C_x+2)$. Для каждых часов существует ($2C_x$+2) интервалов; для интерпретаций с ненулевыми дробными частями интервалы отличаются упорядоченностью этих частей, и величина |X|! является верхней границей числа таких перестановок; наконец, третий компонент соотношения ограничивает сверху число часов с совпадающими дробными частями.

Разбиение пространства значений локальных часов на регионы является базовым приёмом при анализе временных автоматов, но оно имеет существенный недостаток: число регионов растёт экспоненциально как с ростом числа локальных часов, так и с увеличением значений их потолков. Поэтому применение такого подхода на практике весьма затруднительно, поскольку размер полученной модели будет гигантским. Поэтому необходимо введение более эффективного отношения эквивалентности.

Временные зоны

Причина огромного числа временных регионов заключается в следующем: разбиение пространства значений локальных часов на регионы производится так, чтобы попавшие в один регион интерпретации часов с точки зрения поведения любого автомата с любыми временными ограничениями были эквивалентны. Этот подход не учитывает специфики конкретного автомата, с точки зрения которого многие из полученных регионов будут неразличимы. Нужно ввести такое отношение эквивалентности, чтобы полученные классы были эквивалентны относительно лишь тех ограничений, которые используются в описании автомата и спецификации. Возможным решением является использование временных зон. Граф зон - это эффективное представление графа регионов, где каждая временная зона является объединением многих регионов, эквивалентных относительно существенных для данного автомата ограничений.

Временная зона задается некоторым ограничением на показания часов g и является классом эквивалентности показаний часов относительно этого ограничения: $[g]=\{v, v|=g\}$. Хотя зоны и являются объединениями регионов, их можно построить и без

предварительного построения регионов, а лишь с использованием операций над временными ограничениями, используемыми в автомате.

Рассмотрим работу с зонами на примере. Пусть в системе переходов автомата, имеющего пару локальных часов x и y, присутствует переход между состояниями a и b, показанный на рис.3. Определим зоны, соответствующие локации a. Пусть при переходе в a автомат может находиться лишь в зоне, определенной ограничениями $1 \le x \le 4, 1 \le y \le 2$. Эту зону назовём исходной зоной (ИЗ)

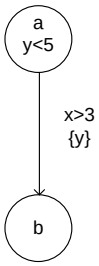

Рисунок 3 - Переход a-b

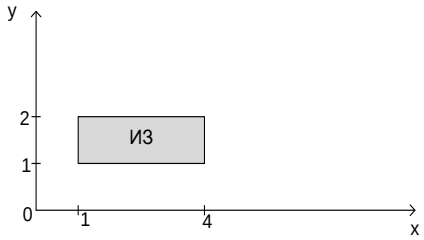

Рисунок 4 - Исходная зона состояния a

С течением времени зона, отображающая возможные интерпретации часов, расширяется. Поскольку часы x и y меняются с одним темпом, множество возможных значений будет ограничено отношением $-1 \le x - y \le 3$. Также зона будет ограничена инвариантом состояния a $y \le 5$. Поэтому зона Z_a, включающая все возможные интерпретации часов до выхода из локации a, выглядит так:

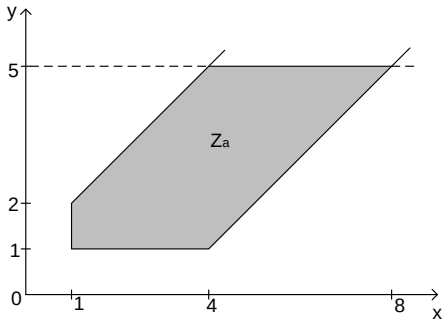

Рисунок 5 - Зона Z_a

Она задаётся следующими временными ограничениями: $Z_a = (x \ge 1) \wedge (1 \le y \le 5) \wedge (-1 \le x - y \le 3)$.

Однако не все интерпретации часов из Z_a будут эквивалентны друг другу. Мы не учли еще одно временное ограничение, а именно условие x>3 на переходе из a в b. Согласно этому условию, если автомат находится в Z_a при x>3, он может перейти в локацию b, иначе - не может. Поэтому зону Z_a следует разделить на 2:

$$Z_1 = (1 \le x \le 3) \wedge (y \ge 1) \wedge (-1 \le x - y \le 3);$$

$$Z_2 = (x > 3) \wedge (1 \le y \le 5) \wedge (-1 \le x - y \le 3).$$

При этом ИЗ Z_1 представляет собой часть ИЗ Z_a, удовлетворяющую условию $x \le 3$:

$$IZ_1 = (1 \le x \le 3) \wedge (1 \le y \le 2).$$

ИЗ Z_2 же помимо оставшейся части IZ_a включает множество интерпретаций часов, для которых x=3, то есть

$$IZ_2 = (x > 3) \wedge (1 \le y \le 2) \vee (x = 3) \wedge (2 \le y \le 4).$$

Заметим, что часть IZ_2, а именно отрезок $(x = 3) \wedge (2 \le y \le 4)$, не входит в Z_2, однако именно ИЗ (вместе с инвариантами) определяет вид зоны.

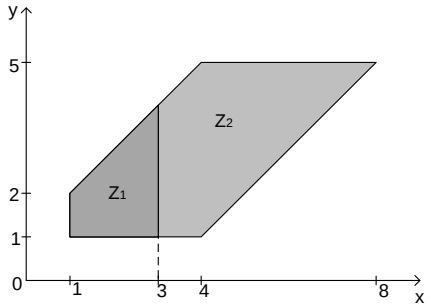

Рисунок 6 - Временные зоны локации a

Из зоны Z_2 возможен переход в локацию b. Определим ИЗ этой локации, исходя из вида Z_2 и условий на переходе. На ребре из a в b происходит сброс часов y, поэтому IZ_b представляет собой проекцию Z_2 на ось x и задается следующим образом: $IZ_b = (3 < x \le 8) \wedge (y = 0)$

При использовании отношения зональной эквивалентности нам нужно ответить на 2 вопроса: каким образом следует представлять зоны и как построить граф временных зон?

Представление временных зон

Любая зона представляет собой класс эквивалентности показаний часов относительно некоторого ограничения g. Следовательно, задав ограничение, мы определим и зону. Ограничения во временном автомате имеют вид $x_i \sim c$ либо $x_i - x_j \sim c$, где x_i и x_j - локальные часы, c - некоторое целое число, а $\sim \in \{<, \le, \ge, >\}$. Любое ограничение можно свести к конъюнкции однотипных выражений. Все неравенства можно свести к отношениям < и $\le$, кроме того, вводятся часы x_0, значение которых всегда равно 0. После этих преобразований описание зоны выглядит следующим образом:

$$Z ::= x_i - x_j < c \mid x_i - x_j \le c \mid Z \wedge Z.$$

Так, зону Z_a в этом случае можно представить так:

$$Z_a = (x_0 - x \le -1) \wedge (x_0 - y \le -1) \wedge (y - x_0 \le 5) \wedge (y - x \le 1) \wedge (x - y \le 3).$$

Зону, описанную с помощью однотипных неравенств, можно представить в виде матрицы $D(Z) = (d_{ij}, < / \le)$, где $d_{ij} = x_i - x_j$. Зона Z_a, описанная с помощью такой матрицы, имеет следующий вид:

	$\mathbf{x_0}$	**x**	**y**
$\mathbf{x_0}$	$(0, \le)$	$(-1, \le)$	$(-1, \le)$
x	$(\infty, <)$	$(0, \le)$	$(3, \le)$
y	$(5, \le)$	$(1, \le)$	$(0, \le)$

На диагонали матрицы всегда находятся пары $(0, \le)$. Одна и та же зона может быть задана различными временными ограничениями и, как следствие, разными матрицами. Например, пару $(\infty, <)$ в матрице, соответствующей Z_a, можно заменить на $(9, \le)$, не изменив зону. Матрица зоны с максимально узкими условиями называется канонической. После выполнения некоторых операций над зонами следует приводить полученную матрицу к каноническому виду.

Построение графа зон

Построение графа зон для временного автомата проведём в несколько этапов.

1) Для каждой локации определяется исходная зона. Для этого используются пометки входных ребер состояния:

- если пометка содержит сброс часов c в 0, то в исходную зону локации войдут только интерпретации часов с c=0;
- если пометка содержит условие вида c>k (и при этом c не сбрасывается в 0), то в исходную зону локации войдут только интерпретации часов с c>k.

Также при определении ИЗ необходимо учитывать инвариант состояния - исходное значение при входе в зону не может превышать максимально допустимое.

Таким образом, ИЗ имеет вид $IZ = z \wedge g \wedge inv$, где z - конъюнкция выражений вида (c=0), g - конъюнкция ограничений на входных рёбрах, inv - инвариант состояния.

Если локация имеет несколько входных рёбер, то для каждого из них создаётся своя ИЗ. Эта зона связывается с конкретным входным ребром, и при её построении учитываются z и g только этого ребра.

Получив ИЗ, нам нужно построить зоны, соответствующие данной локации (по одной на каждое входное ребро). Применительно к выбранной структуре представления зон это осуществляется следующим образом:

- отношения вида $x_i - x_j \sim c,\ i, j \neq 0$ остаются без изменений, поскольку все часы изменяются с одним темпом;
- отношения $x_0 - x_j \sim c$ также остаются в силе - приращение времени не может повлиять на нижнюю границу зоны;
- верхние ограничения (отношения вида $x_i - x_0 \sim c$) ИЗ удаляются (в таблицу записывается пара), но накладываются новые, определяемые инвариантами локации.

В некоторых случаях возможен переход в автомате при таких показаниях часов, которые удовлетворяют условиям на переходе, но не удовлетворяют инварианту состояния, в которое этот переход ведёт. В этом случае автомат будет находиться в недопустимом состоянии. Для предотвращения подобной ситуации необходимо перенести ограничения на время пребывания автомата в неком состоянии на входные дуги этого состояния. После этой операции условия перехода по ребру будет выглядеть так: $g' = g \wedge Inv$, где g - исходное ограничение, а Inv - инвариант состояния, в которое ведёт это ребро. Однако в условие нужно добавлять инварианты лишь тех часов, которые не обнуляются на переходе.

2) На втором шаге каждая из ранее полученных зон разбивается на подзоны в зависимости от временных условий на исходящих дугах. Количество подзон равно $\prod_{x \in X} (C_x + 1)$, где X - множество часов, упоминающихся в ограничениях на исходящих дугах, C_x - количество ограничений часов x, значимых для этой зоны (то есть ограничений, которые делят зону на две непустые подзоны: $(z \wedge g \neq z) \wedge (z \wedge g \neq \varnothing)$). Встаёт вопрос: как построить граф переходов между этими подзонами? Для этого определим все возможные переходы между ними, предварительно разбив множество возможных отношений, встречающихся в ограничениях, на две группы: $\{<, \geq\}$ и $\{>, \leq\}$. Такой выбор пар объясняется тем, что условия $x < 5$ и $x \geq 5$ делят зону на одинаковые подзоны.

В пределах одной зоны возможны следующие переходы:

- из z_1 в z_2, где между z_1 и z_2 одни из часов перешли через ограничение, остальные – нет;
- из z_1 в z_2, где между z_1 и z_2 n часов перешли через свои ограничения, и при этом все n ограничений имеют отношения из одной группы.

Ограничения вида $x_i - x_j \sim c$ не рассматривались, поскольку из-за равномерного изменения всех часов переход между образовавшимися таким образом подзонами невозможен. Если первый тип возможных переходов довольно очевиден, то второй требует некоторого пояснения. Пусть часы x и y перешли через ограничения разных типов, например, x<2 и y>1. Зона z_1 характеризуется показаниями часов $(x < 2 \wedge y \leq 1)$, z_2 - $(x \geq 2 \wedge y > 1)$. Докажем невозможность прямого перехода из z_1 в z_2.

- Если x-y=1, то с течением времени были достигнуты показания часов $(x = 2 \wedge y = 1)$ которые не принадлежат ни z_1, ни z_2.
- Если x-y>1, то часы y достигнут отметки 1 в момент, когда x будет больше 2. $(x > 2 \wedge y = 1)$ также не соответствует зонам z_1 и z_2.
- Если x-y<1, то показания часов в некоторый момент времени будут описываться так: $(x < 2 \wedge y = 1)$. Но при любом значении x, сколь угодно близком к 2, можно выбрать промежуток времени, через который автомат попадёт в зону $(x < 2 \wedge y > 1)$, не являющуюся зонами z_1 и z_2.

Таким образом, при любых значениях часов x и y прямой переход из z_1 в z_2 невозможен.

Также на разбиение зон влияют временные условия, указанные в формуле. Они не относятся к какому-либо конкретному переходу, а должны проверяться всегда. Каждое элементарное условие имеет вид $x_i \sim c$ или $x_i - x_j \sim c$, поэтому с ними можно поступать аналогично ограничениям на переходах.

Учитывая выбранную модель представления временных зон, принцип определения возможных переходов должен быть немного видоизменён. После введения часов x_0 все ограничения имеют вид $x_i - x_j \sim c$. Чтобы свести построение графа переходов для условий, представленных таким образом, к описанному выше алгоритму, нужно выделить из всего множества условий те, в которых упоминается x_0, а затем разделить их на 2 группы: $\{x_i - x_0 < c, x_0 - x_i \le c\}$ и $\{x_i - x_0 \le c, x_0 - x_i < c\}$.

В целях сокращения затрачиваемого на построение графа времени можно определить часть переходов (а именно все переходы первого типа) уже на стадии разбиения зоны на подзоны. Для этого нужно использовать ряд правил.

- Если подзона разбивается на 2 части ограничением $x_i \sim c$, то назовём образовавшиеся части старшей ($x_i < / \le c$) и младшей ($x_i > / \ge c$). Из младшей подзоны есть переход в старшую. Это правило имеет одно исключение: если описание подзоны содержит ограничение вида $x_i = c$ (с учётом модели представления зон: $(x_i - x_0) \le c \wedge (x_0 - x_i) \le c$), то между частями нет перехода.
- Если подзона z_1 разбивается на 2 части ограничением $x_i \sim c$ и она имеет переход в z_2, которая также разбивается этим ограничением на 2 части, то существует переход из младшей части z_1 в младшую часть z_2 и аналогичный переход между старшими частями. Это правило также имеет исключение: если младшие (старшие) части подзон содержат ограничение вида $x_i = c$, перехода между ними нет.
- Если подзона z_1 разбивается на 2 части ограничением $x_i \sim c$ и она имеет переход в z_2, которая к этому ограничению нечувствительна, то существует переход из старшей части z_1 в z_2.

Доказательство: Если z_2 нечувствительна к ограничению, то она полностью лежит либо ниже, либо выше его. Если она лежит ниже, то из старшей части z_1 невозможно попасть в z_2, что противоречит определению перехода между зонами. Значит, z_2 лежит выше ограничения, то есть она является собственной старшей частью.

- Если нечувствительная к ограничению $x_i \sim c$ подзона z_1 имеет переход в z_2, которая им разбивается на 2 части, то существует переход из z_1 в младшую часть z_2. Доказательство аналогично предыдущему правилу.
- Если подзона z_1 имеет переход в z_2 и обе разбиваются на 2 части ограничением $x_i - x_j \sim c$, то назовём получившиеся части i- и j-ориентированной в зависимости от того, ближе к какой оси (x_i или x_j) относительно прямой $x_i - x_j = c$ находится данная часть. i-(j-)ориентированная часть подзоны z_1 имеет переход в i-(j-) ориентированную часть подзоны z_2 (это объясняется невозможностью перехода через прямую $x_i - x_j = c$).
- Если подзона z_1 разбивается на 2 части ограничением $x_i - x_j \sim c$ и она имеет переход в z_2, которая к этому ограничению нечувствительна, то существует переход из i-ориентированной части z_1 в z_2, если z_1 и z_2 разделены ограничением на часы x_i.
- Если нечувствительная к ограничению $x_i - x_j \sim c$ подзона z_1 имеет переход в z_2, которая им разбивается на 2 части, то существует переход из z_1 в i-ориентированную часть z_2, если z_1 и z_2 разделены ограничением на часы x_j.

После выделения подзон нужно определить для каждой из них ИЗ. ИЗ подзоны z_1 равна $IZ_1 = IZ \cap z_1 \cup \bigcup((x = gl) \cap z_1)$, где IZ - ИЗ зоны z, gl - условия, ограничивающие z_1 снизу. gl можно определять на этапе разбиения на подзоны. При делении подзоны по некоторому ограничению на часы x (x~c) множество gl её младшей части остаётся неизменным, а в gl старшей части добавляется условие x=c, а если подобное условие уже было, то старая константа заменяется новой. Для выполнения этого алгоритма необходима упорядоченность проверки условий на каждые часы.

Разбиение зоны на подзоны на этом этапе имеет цель выделить состояния, из которых возможен переход в другую локацию. И его результатом помимо собственно множества подзон должна стать информация о возможных переходах из этих подзон в другие состояния автомата. Для решения этой задачи исходная зона помечается именами всех локаций, в которые из неё возможен переход. Далее, перед разбиением зоны нужно рассмотреть все условия на исходящих дугах соответствующего состояния. Ограничения на каждые часы упорядочиваются и для каждого из них определяется, какие переходы требуют выполнения ограничения, а какие - его отрицания. После разбиения зоны просматриваются все её подзоны и проверяются на предмет удовлетворения ограничениям. При неудовлетворении подзоны некоторому условию, она теряет соответствующие пометки.

3)Если разбиение зоны по какому-либо ограничению делит её ИЗ на 2 части, каждая из образованных подзон связывается с входным ребром, соответствовавшим зоне до раз-

биения. В этом случае ограничение влияет не только на рассматриваемую зону, но и на состояния, имеющие в неё переход. Ведь если зона Z состояния S, связанная с состоянием S_0, делится (вместе со своей ИЗ) ограничением на 2 части - Z_1 и Z_2, то нужно определить, из какой зоны состояния S_0 возможен переход в Z_1, а из какой - в Z_2. Для этого следует ребро из S_0 в S заменить на n рёбер вида S_0-Z_i, пометить каждое из них соответствующими условиями и произвести описанную выше процедуру упорядочивания ограничений, поделивших ИЗ. Затем с каждым таким ограничением нужно поступить следующим образом:

- переносим условие на все зоны состояния S_0;
- осуществляем разбиение зон состояния S_0 на подзоны по новому ограничению аналогично предыдущему этапу;
- заменяем пометку S совокупностью пометок $\{Z_1, Z_2..Z_n\}$;
- просматриваем все зоны состояния S_0, В зависимости от набора условий, которым удовлетворяет зона, удаляем лишние пометки;
- если ограничение поделило ИЗ подзоны, связанной с состоянием S_i, повторяем алгоритм для этого состояния.

Обработка условия заканчивается, если не осталось поделённых им ИЗ. В этом случае переходим к следующему ограничению.

После завершения разбиения необходимо связать между собой подзоны различных состояний. Связь осуществляется на основании пометок. Подзона z_1 может быть помечена:

- состоянием s (пометка осуществляется на 2 этапе). Тогда z_1 связывается с подзоной z_2 состояния s, которая связана с локацией, которой принадлежит z_1. При этом пометка s заменяется на z_2, а z_2 в свою очередь получает входную пометку z_1;
- конкретной подзоной z_2 (пометка осуществляется на 3 этапе). Тогда в граф переходов добавляется переход z_1-z_2, а входная пометка z_2 меняется на подзону z_1.

4) Наконец, последним этапом построения графа временных зон является его коррекция. На первой её итерации рассматриваются все зоны, в которые ведут внешние переходы. Для каждой такой зоны Z выполняются следующие действия:

- определяются зоны, имеющие переход в Z;
- определяется зона $\bigcup(Z_i^u)$, где Z_i^u получаются из найденных на предыдущем этапе зон Z_i путём сброса некоторых часов в ноль, также в состав каждой Z_i^u включаются её границы;
- находится пересечение этой зоны с ИЗ Z. Если $\bigcup(Z_i^u) \wedge IZ = IZ$, то коррекция зоны Z завершается. Иначе:
- обновляется ИЗ Z: $IZ = \bigcup(Z_i^u) \wedge IZ$;
- обновляется зона Z (строится по обновлённой ИЗ и ограничениям);
- зона Z помечается как модифицированная.

Далее механизм коррекции применяется ко всем потомкам модифицированных зон. После обработки пометка сбрасывается, в случае изменения потомка в процессе коррек-

ции он также помечается. Включение границ Z_i^u в её состав сделано для грамотной работы с подзонами, разделенными временными ограничениями. В таком случае зона-предок может не пересекаться с ИЗ зоны-потомка.

Полученный в результате граф ременных зон изображен на рис. 7. На рисунке явно выделено разделение зон по различным локациям.

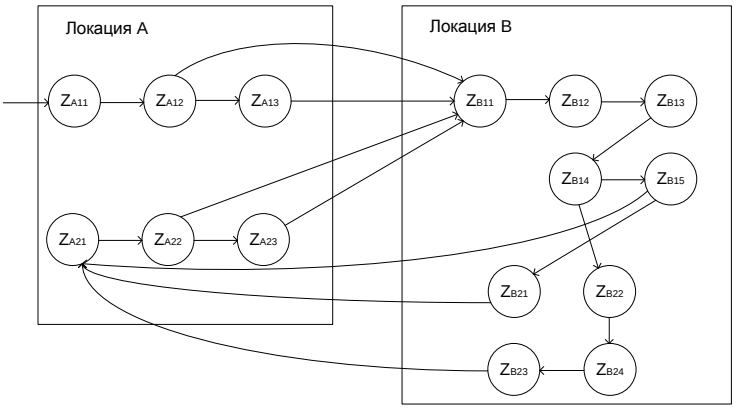

Рисунок 7 - Граф временных зон

Верификация временного автомата

Верификацию временного автомата A при спецификации его свойств с помощью формул логики CTL можно свести к анализу выполнения этих формул для его графа зон R(A). При верификации графа зон используется стандартный для CTL логики алгоритм разметки состояний структуры Крипке выполняющимися в них подформулами. Для спецификации свойств автомата используются обычные CTL-формулы с парами "квантор пути - темпоральный оператор", за исключением пар EX и AX (поскольку для вещественно-численной временной области понятие следующего момента времени некорректно). Множество допустимых атомарных предикатов (ранее содержавшее имена локаций, событий, входных и выходных воздействий) расширяется: в формулах могут присутствовать соотношения между значениями локальных часов, структура которых описывается грамматикой: $\gamma ::= x \sim c \mid x - y \sim c \mid \neg\gamma \mid \gamma \wedge \gamma$, где x и y - локальные часы, c - константа, а $\sim$ – отношение, принадлежащее множеству $\{<, \leq\}$.

При верификации систем реального времени важно иметь возможность задавать свойства, определяющие допустимые интервалы действия темпоральных операторов. В используемой нами логике TCTL операторы U, F и G имеют временные рамки, определяющие такие интервалы. Они указываются в формуле непосредственно у оператора. Формулы состояний ϕ определяется следующей грамматикой:

$$\phi ::= p \mid \gamma \mid \neg\phi \mid \phi \vee \phi \mid E\varphi \mid A\varphi,$$
$$\varphi ::= \phi U_J \phi,$$

где p - атомарный предикат, γ - временное ограничение, φ - формула пути, а $J \subseteq N$ - непрерывный временной интервал.

Для состояния графа зон $q = \langle loc, z \rangle$, представляющего собой пару "локация-зона", выполнение формул TCTL логики определяется так:

$$q \models p \quad iff \quad p \in L(loc);$$

$$q \models \gamma \quad iff \quad z \models \gamma;$$

$$q \models \neg\phi \quad iff \quad q \not\models \phi;$$

$$q \models \phi_1 \vee \phi_2 \quad iff \quad q \models \phi_1 \ или \ q \models \phi_2;$$

$$q \models E\varphi \quad iff \quad \sigma \models \varphi \quad для\ некоторого\ пути\ \sigma;$$

$$q \models A\varphi \quad iff \quad \sigma \models \varphi \quad для\ всех\ путей\ \sigma.$$

Проверка TCTL формул может быть сведена к проверке формул , заданных в логике CTL. Главная особенность TCTL - наличие ограничений на время действия темпоральных операторов. Если формула проверяется для состояния s, то необходимо отслеживать время, прошедшее с момента прихода автомата в состояние s. Для этого можно ввести дополнительные локальные часы, которые должны быть сброшены в 0 в состоянии s. Для проверки выполнения формулы Ф на вычислении σ в автомат вводятся новые часы f, сбрасываемые в 0 в начальном состоянии вычисления и имеющие единственную цель - отсчёт времени, прошедшего с начала вычисления. Эти часы называются формульными. Они позволяют преобразовать формулы TCTL к виду, принятому в CTL логике. Так, ограничение J темпорального оператора U может быть представлено в виде ограничения на формульные часы f: $f \in J$. Формула пути $\phi_1 U_J \phi_2$ логики TCTL может быть выражена так: $(\phi_1 \vee \phi_2) U (\phi_2 \wedge f \in J)$ при условии, что часы f сброшены в 0 в начальном состоянии этого пути.

С использованием формульных часов выполнимость формула $\Phi = E(\phi_1 U_J \phi_2)$ логики TCTL в состоянии s временного автомата определяется следующим образом:

$$s \models E(\phi_1 U_J \phi_2) \quad iff \quad s\{f = 0\} \models E((\phi_1 \vee \phi_2) U (\phi_2 \wedge f \in J)).$$

Выполнимость формулы $\Phi = A(\phi_1 U_J \phi_2)$ определяется аналогично.

Для формул $F_J \phi$ и $G_J \phi$ также имеются соотношения, позволяющие с использованием формульных часов свести проверку TCTL формулы к анализу соответствующей формулы CTL логики:

$$F_J \phi = true U_J \phi = F(\phi \wedge (f \in J));$$

$$G_J \phi = \neg F_J \neg\phi = \neg F(\neg\phi \wedge (f \in J)) = \neg F\neg((f \in J) \Rightarrow \phi) = G((f \in J) \Rightarrow \phi).$$

Можно осуществить еще одно преобразование, упрощающее проверку формулы. Логика CTL оперирует 8 парами "квантор пути - темпоральный оператор" (в случае верификации графа зон - 6 парами). Для проверки подформулы Pϕ, где ϕ - подформула более

низкого уровня, а P - пара, используется особенный алгоритм для каждой P. Соответственно, количество используемых пар определяет число требуемых алгоритмов. Это количество можно сократить с помощью приведения формулы CTL к базису. Все пары можно выразить через 3 базисных, которыми могут являться, например, AF, EU и EX. Поскольку в случае вещественночисленной временной области оператор X не используется, базис представляет собой набор из двух пар: {AF,EU}. Остальные 4 пары выражаются через базисные следующим образом:

$$EG\varphi \equiv \neg AF\neg\varphi;$$
$$A(\varphi_1 U \varphi_2) \equiv AF\varphi_2 \wedge \neg E(\neg\varphi_2 U(\neg\varphi_1 \wedge \neg\varphi_2));$$
$$EF\varphi \equiv E(TrueU\varphi);$$
$$AG\varphi \equiv \neg E(TrueU\neg\varphi).$$

Алгоритм маркировки состояний

Для каждой подформулы ψ заданной формулы φ алгоритм маркировки должен выполнять следующие шаги:

1) Вычисляется множество Sat_ψ состояний, в которых выполняется ψ.

2) Вводится новый атомарный предикат p_ψ.

3) Этим атомарным предикатом помечаются все состояния из множества Sat_ψ.

Поскольку обработка каждой подформулы завершается введением нового атомарного предиката и пометкой этим предикатом соответствующих состояний модели Крипке, то можно считать, что на каждом шаге элементами подформул являются атомарные предикаты. По завершении алгоритма, если начальное состояние модели помечено атомарным предикатом p_φ, то можно сделать заключение о выполнимости заданной формуле на данном графе зон.

Алгоритм маркировки состояний можно описать так:

```
for all 0<i<=|Φ| do
   for all ψ∈Sub(Φ) with | ψ |=i; для всех подформул Φ с глубиной i
 switch(ψ):
    p    :   Sat_ψ:={s=(loc,z)∈S|p∈L(loc)}
    ¬p   :   Sat_ψ:={s=(loc,z)∈S|p∉L(loc)}
    γ    :   Sat_ψ:={s=(loc,z)∈S|z∧γ=z}
    p∨q  :   Sat_ψ:={s=(loc,z)∈S|p∈L(loc) или q∈L(loc)}
    AF p:    Sat_ψ:=Sat_AF(p)
    E(pUq):  Sat_ψ:=Sat_EU(p,q)
 end switch
   for all s∈ Sat_ψ do L(s):=L(s)∪( p_ψ) od
od
```

В этом варианте алгоритм обрабатывает все формулы, начиная с самых внутренних. | ψ | означает число подформул формулы ψ, поэтому сначала будут обрабатываться под-

формулы нулевой глубины, то есть атомарные предикаты. Учитывая особенности используемого синтаксического анализа, алгоритм может быть несколько изменён: на каждом следующем шаге будем рассматривать не подформулы глубины i, а подформулу с номером i. Учитывая порядок выделения подформул, все подформулы формулы ψ_i будут обработаны алгоритмом маркировки на шагах, предшествующих i.

Если подформула имеет вид AF p или E(pUq), то для вычисления Sat_ψ должна вызываться соответствующая функция, которая возвращает множество состояний, на которых эта подформула выполняется. Эти функции подробно рассмотрены в [9].

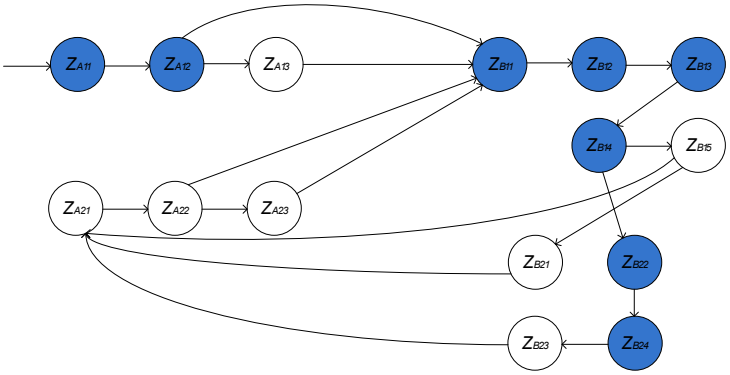

Рисунок 8 - Маркировка состояний графа.

Построение контрпримера

После завершения алгоритма верификации система должна привести путь, подтверждающий результат верификации. При работе с CTL логикой дело с контрпримерами обстоит несколько сложнее, чем в случае с LTL формулами. Скажем, при успешном выполнении проверки формулы $\phi=E(\psi_1 U \psi_2)$ легко построить подтверждающий пример - нужно всего лишь привести любой путь в графе, для которого справедливо $(\psi_1 U \psi_2)$. Но что, если проверка такой формулы даёт отрицательный результат? Для подтверждения его нужно показать, что нет ни одного пути, на котором выполняется $(\psi_1 U \psi_2)$. К сожалению, доказать это с помощью демонстрации путей невозможно, поскольку для этого потребуется предоставление пользователю всех возможных вычислений автомата. Поэтому ограничимся одним примером, на котором заданная формула не выполняется.

Построение контрпримера зависит от результата верификации. При отрицательном результате проверки формулы Ф для графа зон R(A) нужно провести дополнительную маркировку состояний R(A) формулой ¬Ф (чтобы в любом случае в начальном состоянии выполнялась конечная формула и можно было использовать одинаковый алгоритм). В результате получено множество Sat_φ, где $\varphi=Ф$ или $\varphi=¬Ф$ в зависимости от результата верификации. Затем выбирается путь по следующему принципу.

- Путь начинается из начального состояния графа.
- На каждом шаге следующее состояние пути выбирается из потомков текущего состояния:

- если до текущего момента все состояния в пути принадлежали Sat_φ, то при наличии среди потомков состояний, принадлежащих Sat_φ, следующее состояние пути недетерминировано выбирается из них;
- если до текущего момента все состояния в пути принадлежали Sat_φ, то при отсутствии среди потомков состояний, принадлежащих Sat_φ, следующее состояние пути недетерминировано выбирается из всех потомков;
- если к текущему моменту в пути уже присутствуют состояния, не принадлежащие Sat_φ, то следующее состояние пути недетерминировано выбирается из всех потомков;
- если текущее состояние не имеет потомков, путь завершается.

- Если очередное состояние уже встречалось ранее в пути, построение вычисления на этом завершается с пометкой об обнаружении цикла.

Таким образом, результатом верификации является заключение о выполнимости формулы для данной системы реального времени, а также подтверждающий пример или контрпример.

Заключение

В статье предложен подход к верификации систем реального времени на основе метода Model Checking. Данный подход может применяться для выявления ошибок в программах на этапе детализированного проектирования систем реального времени.

В первой части статьи рассмотрен математический аппарат моделирования системы и формализации требований к ней. В качестве структуры, используемой для построения моделей систем реального времени, выбран временной автомат, а в качестве формализма для представления спецификации - TCTL - логика.

Во второй части рассмотрены методы упрощения графа состояний временного автомата - введение отношения эквивалентности и последовательный переход к графам регионов и временных зон. Описан алгоритм построения графа временных зон.

Третья часть статьи посвящена процессу проверки соответствия системы своей спецификации. Для осуществления проверки используется метод маркировки состояний. Описаны преобразования, позволяющие свести задачу к существующим методам анализа CTL-формул.

К недостаткам подхода можно отнести сложность его применения для больших программных систем. Даже верификаторы общего назначения зачастую плохо справляются с проблемой комбинаторного взрыва в пространстве состояний, возникающего с ростом размера входной модели. Для систем реального времени эта проблема стоит особенно остро, ведь помимо выделения локаций, выполняемого в любом верификаторе, нужно произвести разбиение этих локаций на отдельные временные зоны. Этот этап усугубляет проблему роста числа состояний, а также значительно увеличивает время выполнения верификации. Описанный в статье подход позволяет отчасти решить первую из этих про-

блем: итоговое число состояний значительно меньше, чем может быть получено другими методами (такими, как построение графа регионов).

Список литературы

1. Кузьмин Е. В., Соколов В. А. О дисциплине специализации «Верификация программ» // Доклады II научно-методической конференции «Преподавание математики в компьютерных науках» Ярославль: ЯрГУ. 2007. С. 91-101
2. Дейкстра Э. Дисциплина программирования / Дал У., Дейкстра Э., Хоор К. Структурное программирование. М.: Мир. 1975. –247 с.
3. Кларк Э.М., Грамберг О., Пелед Д. Верификация моделей программ: Model checking. М.: МНЦМО, 2002. –416 с.
4. SPIN – http://spinroot.com/spin/whatispin.html
5. Symbolic Model Verificator. Carnegie Mellon University.– http://www.cs.cmu.edu/~modelcheck/smv.html
6. Alur R., Courcoubetis C., Dill D. Model-checking in dense real-time // Information and Computation. 104: 2–34, 1993.
7. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998. –55 с.
8. 8. Вельдер С.Э, Лукин М.А., Шалыто А.А., Яминов Б.Р. Верификация автоматных программ. СПбГУ ИТМО, 2011. – 240 с.
9. Карпов Ю. Г. MODEL CHECKING. Верификация параллельных и распределённых программных систем. –Спб.: БХВ-Петербург, 2010. –560 с.
10. Шалыто А. А. Программная реализация управляющих автоматов // Судостроительная промышленность. Серия «Автоматика и телемеханика». 1991. Вып. 13.
11. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. 2008. — 167 с.
12. Кузьмин Е.В., Соколов В.А. О некоторых подходах к верификации автоматных программ.- Сборник докладов семинара Go4IT – шаг к новым технологиям Интернета. Москва, Институт системного программирования, 2007. С. 43-48.