ISSN 2782-2001
Системы анализа и обработки данных
том 81, № 1, 2021, с. 103–122

http://journals.nstu.ru/vestnik
Analysis and data processing systems
Vol. 81, No. 1, 2021, pp. 103–122

ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

INFORMATICS, COMPPUTER ENGINEERING AND CONTROL

УДК 004.434:004.942 DOI: 10.17212/2782-2001-2021-1-103-122

Язык моделирования гетерогенных динамических систем LISMA_HDS*

Е.А. ПОПОВ[a], Ю.В. ШОРНИКОВ[b]

630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет

[a] *e.popov@corp.nstu.ru* [b] *shornikov@inbox.ru*

Гетерогенные динамические системы (ГДС) описывают одновременно протекающие процессы различной физической природы. Такие системы встречаются в многочисленных приложениях науки и техники. Можно выделить следующие характерные особенности ГДС. Часто такие системы оказываются многорежимными или гибридными. В общем случае их режимы задаются в классе задачи Коши для неявных дифференциально-алгебраических систем уравнений. В связи с наличием нескольких разнородных динамических компонентов или процессов, протекающих как во времени, так и в пространстве, размерность совокупной системы уравнений может быть достаточно высокой. В некоторых случаях система уравнений имеет внутреннюю структуру, например, дифференциально-алгебраическая система уравнений, аппроксимирующая дифференциальное уравнение в частных производных по методу прямых. Тогда имеется возможность компактной алгоритмической записи исходной громоздкой системы уравнений. Также в гетерогенных гибридных динамических системах могут возникать события качественно разных типов. Поэтому появляется необходимость в применении разных численных алгоритмов обнаружения событий.

Сегодня компьютерное моделирование ГДС выполняется в окружении инструментальных средств. Широко используемые инженерами языки моделирования не позволяют в полной мере отразить все свойства систем из этого класса. Например, в них отсутствует возможность типизации событий.

Поэтому был разработан декларативный язык моделирования общего назначения LISMA_HDS инструментальной среды ИСМА, учитывающий вышеуказанные характерные особенности ГДС. Новый язык включает возможности непосредственного или алгоритмического объявления модельных постоянных, задачи Коши для неявной дифференциально-алгебраической системы уравнений, начальных приближений переменных, а также позволяет задавать явные события времени, режимы функционирования и переходы между ними по событиям разных типов, использовать макроподстановки и реализовывать событийное управление.

LISMA_HDS задан с помощью порождающей грамматики в расширенной форме Бэкуса–Науэра и семантических ограничений. Доказана принадлежность порождающей грамматики к подклассу LL(2) контекстно-свободных грамматик.

* *Статья получена 28 октября 2020 г.*

Ключевые слова: гетерогенные динамические системы, гибридные динамические системы, языки моделирования, типизация событий, формальные грамматики, инструментальная среда ИСМА

ВВЕДЕНИЕ

Гетерогенные динамические системы (ГДС) описывают одновременно протекающие процессы различной физической природы. Такие сложные системы находят применение во многих областях науки и техники: в автоматике, робототехнике, электроэнергетике, автомобилестроении, ракетостроении и т. д.

В общем случае ГДС характеризуются *высокой размерностью* в связи с наличием нескольких составляющих элементов. Это свойство особенно проявляется в случае, когда часть из них моделируют процессы, протекающие во времени и пространстве. Соответствующие дифференциальные уравнения в частных производных после пространственной дискретизации становятся системами обыкновенных дифференциальных уравнений высокой размерности [1].

Часто системы уравнений высокой размерности имеют некоторую внутреннюю структуру, позволяющую описать их в компактной форме с использованием *циклов* и *индексной записи* (например, система обыкновенных дифференциальных уравнений, аппроксимирующая дифференциальное уравнение в частных производных по методу прямых) [2, 3].

Дифференциально-алгебраические системы уравнений (ДАУ) являются естественной формой описания процессов самой различной природы. Они возникают, когда на фазовые переменные накладываются ограничения, например, в соответствии с законом сохранения массы [4]. Более того, при составлении совокупной системы уравнений сложной системы из моделей элементов и уравнений связей между ними в общем случае неизбежно возникает *неявная система ДАУ* [3].

С другой стороны, ГДС часто оказываются дискретно-непрерывными или *гибридными*. Гибридные системы (ГС) представляются в виде нескольких непрерывных режимов функционирования, между которыми определены переходы, условия их срабатывания и мгновенные действия при срабатывании (например, изменение значений переменных или изменение системы уравнений, описывающей режимное поведение) [4–6]. Переключение между режимами происходит мгновенно. В момент переключения непрерывное время останавливается, выполняются все переходы, условия которых истинны, и только потом продолжается процесс, описываемый активной системой уравнений.

Если хотя бы один элемент ГДС является гибридным, то и вся совокупная система также характеризуется гибридным поведением. Тогда режим гетерогенной гибридной динамической системы может быть представлен в общем случае в виде задачи Коши для неявной системы ДАУ с ограничением:

$$\mathbf{F}(t,\ \mathbf{y}(t),\quad \mathbf{y}'(t)) = 0\,, \tag{1}$$

$$\mathbf{y}(t_0) = \mathbf{y}_0\,,\quad \mathbf{y}'(t_0) = \mathbf{y}'_0\,,$$

$$pr(t,\ \mathbf{y}(t)) = \text{true}\,,\quad t \in \left[t_0,\ t^*\right),$$

где $\mathbf{F}: \mathbb{R} \times \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^N$ – вектор-функция, удовлетворяющая условиям существования и единственности решения задачи Коши для соответствующей системы ДАУ на интервале $\left[t_0, t^*\right]$ [1]; t – независимая переменная (время); $\mathbf{y}(t) \in \mathbb{R}^N$ – вектор состояния системы; $\mathbf{y}_0$, $\mathbf{y}'_0 \in \mathbb{R}^N$ – согласованные начальные условия, т. е. $\mathbf{F}(t_0, \mathbf{y}_0, \mathbf{y}'_0) = \mathbf{0}$; $pr: \mathbb{R} \times \mathbb{R}^N \to \{\text{false, true}\}$ – режимный предикат; t_0 – момент времени входа в текущий режим; t^* – момент времени переключения в следующий режим.

Пока ГС находится в текущем режиме, режимный предикат остается истинным. В момент времени t^*, когда pr впервые становится ложным, ГС переходит в следующий режим. Режимный предикат составляется из ограничений $g(t, \mathbf{y}(t)) < 0$ на непрерывные событийные функции $g: \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}$, соединенные логическими операторами конъюнкции и дизъюнкции.

В условиях машинной арифметики точное определение момента переключения t^* не является практически возможным. Поэтому выделяют три типа событий: односторонние, двусторонние, критичные к точности обнаружения [7]. Так как в общем случае невозможно найти момент времени t^* с использованием аналитических методов, применяют численные алгоритмы обнаружения событий. Однако не все алгоритмы гарантируют обнаружение в случае множественных пересечений границы режима в пределах одного шага интегрирования. Поэтому в рамках методологии комплексного обнаружения событий, когда используется несколько одновременно работающих алгоритмов обнаружения, можно также выделить класс труднообнаруживаемых событий [8]. Наличие *событий разных типов* – одна из ключевых особенностей гетерогенных ГС.

1. ЯЗЫК МОДЕЛИРОВАНИЯ LISMA_HDS

В настоящее время сложно представить проектирование или анализ сложной системы без применения компьютерного моделирования. Сегодня инженерам доступны инструментальные среды, автоматизирующие трудоемкий и подверженный ошибкам процесс построения компьютерной модели и выполнения вычислительного эксперимента с ней. Обычно изучаемая система представляется в виде текстовой модели на некотором языке моделирования.

Язык моделирования ГДС должен учитывать их особенности и включать удобные и интуитивно понятные конструкции. Более того, он должен быть простым для предметного пользователя. Большое количество ключевых слов и языковых конструкций, применение таких высокоуровневых парадигм, как объектно-ориентированность, повышают порог вхождения для предметного специалиста без углубленных знаний программирования. Примером может служить язык моделирования Modelica [9], на базе которого построены инструментальные среды Dymola, OpenModelica, Wolfram SystemModeler. В широко распространенном среди инженеров программном комплексе Matlab/Simulink используются базовые текстовый язык программирования и графический язык структурных схем, обладающий известными

недостатками при физическом моделировании сложных динамических систем [3, 5]. С другой стороны, ни один из массово используемых языков моделирования не позволяет описать все свойства ГДС (например, указать типы событий).

Порождающая грамматика языка моделирования должна быть в одном из известных и хорошо изученных подклассов контекстно-свободных грамматик, для которых доказана однозначность, например, LL(k) [10, 11].

Более того, для них могут быть сконструированы эффективные нисходящие безвозвратные синтаксические анализаторы с линейной зависимостью времени работы от длины входной цепочки [12].

Рассмотрим декларативный язык моделирования общего назначения LISMA_HDS инструментальной среды ИСМА [6], учитывающий все вышеуказанные особенности ГДС.

Отличительными особенностями языка LISMA_HDS, как и в случае LISMA_PDE [13], являются запись математических конструкций в близкой к языку математики форме, а также отсутствие повышенных требований к навыкам программирования у предметного пользователя.

Ниже приведены ключевые продукции порождающей грамматики $G[\textit{модель}]$ языка LISMA_HDS в расширенной форме Бэкуса–Науэра в соответствии со стандартом ISO/IEC 14977:1996 [14]. В левых частях правил в скобках приводятся сокращенные имена нетерминальных символов грамматики.

Программные модели (*модель*) на LISMA_HDS являются последовательностями объявлений, возможно, пустыми:

$$\textit{модель}\,(\mathrm{M1}) = \{\textit{объявление}\};$$

$$\textit{объявление}\,(\mathrm{D1}) = \textit{пост\_об} \mid \textit{уравнение} \mid \textit{нач\_усл} \mid \textit{режим}$$

$$\mid \textit{соб\_упр} \mid \textit{макрос} \mid \textit{цикл} \mid \textit{цикл\_соб};$$

Язык LISMA_HDS содержит конструкции для непосредственного или алгоритмического объявления программных постоянных, задачи Коши для неявной системы ДАУ, начальных приближений переменных, а также позволяет задавать явные события времени, режимы функционирования и переходы между ними по событиям разных типов, макроподстановки и реализовывать событийное управление. Поддерживаются как однострочные, так и многострочные комментарии в стиле языка C++.

Рассмотрим разные типы объявлений отдельно в рамках подзадач описания непрерывного и дискретного поведения ГС.

2. СПЕЦИФИКАЦИЯ НЕПРЕРЫВНОГО ПОВЕДЕНИЯ

Непрерывное поведение может быть в общем случае описано системой уравнений вида

$$G(t,\ \mathbf{y}(t),\ \mathbf{y}'(t)) = H(t,\ \mathbf{y}(t),\ \mathbf{y}'(t)), \tag{2}$$

которая может быть легко приведена к виду (1), где $F(t,\ \mathbf{y}(t),\ \mathbf{y}'(t)) = G(t,\ \mathbf{y}(t),\ \mathbf{y}'(t)) - H(t,\ \mathbf{y}(t),\ \mathbf{y}'(t))$.

Уравнения системы (2) описываются на LISMA_HDS в соответствии с грамматикой

$$\textit{уравнение}(\mathrm{E1}) = [\textit{идент}, \text{":"}], \textit{AB}, \text{"="}, \textit{AB}, \text{";"};$$

где нетерминальный символ AB, определенный далее по тексту, представляет собой арифметическое выражение, которое может включать фазовые переменные, их производные (имя переменной с последующим символом '), встроенную переменную модельного времени `time`, числовые литералы, встроенные и объявленные постоянные, а также вызовы встроенных математических функций; терминальный символ *идент* – идентификатор.

Метки перед уравнениями (идентификаторы) используются, когда необходимо указать на конкретные уравнения при изменении совокупной системы в момент входа в новый режим. Несколько уравнений могут иметь одинаковые метки.

Могут быть заданы как точные начальные условия задачи Коши, так и некоторые приближения к ним, которые в дальнейшем будут уточнены в процессе инициализации:

$$\textit{нач\_усл}(\mathrm{I1}) = \textit{перем\_имя}, \text{"("}, \text{"t0"}, \text{")"}, (\text{"="} | \text{"}\sim\text{="}), \textit{AB}, \text{";"};$$

где нетерминальный символ *перем_имя* означает имя переменной.

Если не задано начальное значение переменной, то на этапе инициализации в качестве начального приближения будет использоваться ноль.

При инициализации в исходном режиме `init` могут применяться только арифметические выражения, состоящие из числовых литералов и именованных постоянных величин.

Грамматика объявления постоянных величин имеет вид

$$\textit{пост\_об}(\mathrm{C1}) = \text{"const"}, \textit{цеп\_присв}, \{\text{","}, \textit{цеп\_присв}\}, \text{";"};$$

$$\textit{цеп\_присв}(\mathrm{A1}) = \textit{перем\_имя}, \text{"="}, \textit{цеп\_присв\_хв};$$

$$\textit{цеп\_присв\_хв}(\mathrm{A4}) = \textit{AB} \mid \textit{цеп\_присв};$$

где AB должно быть составлено только из объявленных и встроенных постоянных, а также числовых литералов.

LISMA_HDS позволяет задавать системы уравнений, начальные условия, постоянные, макроподстановки (*макрос*) и блоки событийного управления (*соб_упр*) алгоритмически с помощью цикла `for` и индексной записи:

$$\textit{цикл}(\mathrm{C2}) = \text{"for"}, \textit{идент}, \text{"="}, \textit{мн\_знач}, \{\text{","}, \textit{мн\_знач}\}, \textit{цикл\_тел};$$

$$\textit{мн\_знач}(\mathrm{V2}) = \textit{ЦБЗ}, [\text{":"}, \textit{ЦБЗ}], [\text{":"}, \textit{ЦБЗ}];$$

$$\textit{цикл\_тел}(\mathrm{C3}) = \text{"{"}, \{\textit{пост\_об} \mid \textit{макрос} \mid \textit{уравнение} \mid \textit{нач\_усл} \mid \textit{соб\_упр}\}, \text{"}"};$$

где терминальный символ *ЦБЗ* означает целое число без знака.

Заголовок цикла включает в себя переменную цикла (идентификатор) и множество ее значений Ω, определенное одной или несколькими тройками разделенных двоеточиями чисел l, s, r (*мн_знач*), означающих начальное значение переменной, шаг приращения и конечное значение соответственно: $\Omega = \bigcup_{i=1}^{N\in\mathbb{Z}_+} \Omega_i$, $\Omega_i = \{\omega_j = l_i + s_i j \mid j \in \mathbb{Z}_+ \bigcup \{0\} \wedge \omega_j \le r_i\}$, l_i, $r_i \in \mathbb{Z}_+ \bigcup \{0\}$, $s_i \in \mathbb{Z}_+$, $l_i \le r_i$. Шаг может быть опущен, тогда он по умолчанию будет установлен равным единице. Вместе с шагом также может быть опущена правая граница, что приведет к автоматической установке $r_i = l_i$ и итоговому множеству Ω_i с единственным элементом l_i.

Переменная цикла может быть использована внутри цикла как обычная переменная в арифметических выражениях и в качестве индекса вместе с обычной переменной:

$$\textit{перем}(\mathrm{V1}) = \textit{перем\_имя}, ["'"];$$

$$\textit{перем\_имя}(\mathrm{V3}) = \textit{перем\_инд} \mid \textit{идент};$$

$$\textit{перем\_инд} = \textit{идент}, "[", \textit{индекс}, "]", [\textit{хвост}];$$

$$\textit{индекс} = [\textit{ЦБЗ}, "*"], \textit{идент}, [\textit{адд\_оп}, \textit{ЦБЗ}];$$

$$\textit{адд\_оп} = "+" \mid "-";$$

$$\textit{хвост} = \textit{идент} \mid \textit{ЦБЗ};$$

где терминальный символ *перем_инд* представляет собой имя переменной с индексом.

Как показано выше, в качестве индекса допускается использование простых арифметических выражений типа `2*i+1`, `i-3`, `i`. Так как индексная запись является фактически манипуляцией с идентификаторами, индекс можно размещать внутри составного имени переменной, например, `u[i]0`, что в случае нулевого значения переменной цикла `i` приведет к использованию переменной с именем `u00`.

Например, задача Коши

$$x_1'(t) = x_2(t), \quad x_1(t_0) = 1,$$

$$x_2'(t) = x_3(t), \quad x_2(t_0) = 2,$$

$$x_3'(t) = x_4(t), \quad x_3(t_0) = 3,$$

$$x_4'(t) = x_5(t), \quad x_4(t_0) = 4,$$

$$x_5'(t) = x_1(t), \quad x_5(t_0) = 5$$

может быть описана на LISMA_HDS в виде программной модели, представленной на рис. 1.

```
1  for i = 1:4
2  {
3       x_[i]' = x_[i+1];
4       x_[i](t0) = i;
5  }
6  x_5' = x_1;
7  x_5(t0) = 5;
```

Рис. 1. Программная модель с алгоритмическим определением задачи Коши

Fig. 1. A program model with an algorithmically defined initial value problem

В язык встроены некоторые часто используемые математические функции, например, модуля (`abs(x)`), экспоненциальная (`exp(x)`), максимума двух чисел (`max(x, y)`), минимума двух чисел (`min(x, y)`), степенная (`pow(x, y)`), квадратного корня (`sqrt(x)`), тригонометрические (`sin(x)`, `cos(x)`, `tg(x)`, `ctg(x)`).

Грамматика вызова встроенной функции имеет вид

$$\textit{вызов}(\text{I3}) = \textit{модуль} \,|\, ... \,|\, \textit{экспонента};$$

$$\textit{модуль}(\text{A2}) = \text{"abs"}, \text{"("}, AB, \text{")"};$$

$$\textit{экспонента}(\text{A3}) = \text{"exp"}, \text{"("}, AB, \text{")"};$$

Рассмотрим простой пример неявной системы ДАУ. Имеется идеальный математический маятник – груз массой m, прикрепленный к подвижному шарниру при помощи жесткого недеформируемого стержня нулевой массы длиной l (рис. 2). Маятник находится в поле тяготения Земли, которое в рассматриваемой области пространства будем считать равномерным. На груз действует сила тяжести $m\mathbf{g}$ и усилие в стержне $\mathbf{T}$. Совместим начало декартовой системы координат и положение шарнира. Пусть в начальный момент времени t_0 груз находится в точке (x_0, y_0), а его скорость равняется (v_{x_0}, v_{y_0}). Пренебрегая силой сопротивления среды и трением в шарнире, получим задачу Коши

$$\begin{gathered}
mv_x'(t) = -T(t)\frac{x(t)}{l}, \quad v_x(t_0) = v_{x_0}, \\
x'(t) = v_x(t), \quad x(t_0) = x_0, \\
mv_y'(t) = -T(t)\frac{y(t)}{l} - mg, \quad v_y(t_0) = v_{y_0}, \\
y'(t) = v_y(t), \quad y(t_0) = y_0, \\
x^2(t) + y^2(t) = l^2.
\end{gathered} \tag{3}$$

Математическая модель (3) может быть представлена на LISMA_HDS, как показано на рис. 3.

Строки 1, 2 отвечают за определение постоянных модели – массы маятника *m* и длины стержня *l*. Ускорение *g* является встроенной постоянной языка.

Строки 4, 6, 8, 10, 11 описывают систему уравнений в соответствии с математической моделью. Производные обозначаются традиционным для языка математики символом '.

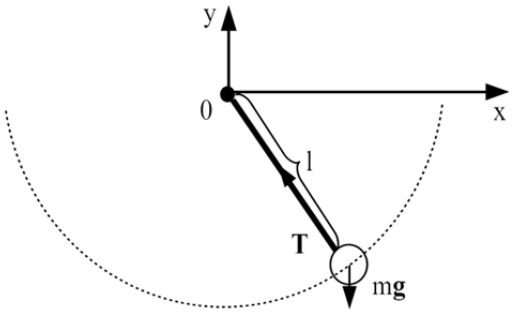

Рис. 2. Идеальный математический маятник

Fig. 2. A simple gravity pendulum

В строках 5 и 7 заданы точные начальные значения переменных `x` и `v_x` соответственно. Для `y` в строке 7 задано начальное приближение; `v_y` и `T` автоматически получат 0 в качестве начальных приближений.

```
1  const l = 5.0;     // длина стержня
2  const m = 1.0;     // масса маятника
3
4  x' = v_x;
5  x(t0) = 4.0;
6  y' = v_y;
7  y(t0) ~= -3.0;
8  m * v_x' = -T * x / l;
9  v_x(t0) = 0.0;
10 m * v_y' = -T * y / l - m * g;
11 x * x + y * y = l * l;
```

Рис. 3. Программная модель идеального математического маятника

Fig. 3. A program model of the simple gravity pendulum

Результаты вычислительного эксперимента с программной моделью на временном интервале $[0;10]$ представлены на рис. 4.

Также LISMA_HDS содержит возможность задания макроподстановок, которые позволяют выделять общие части арифметических выражений в отдельную программную единицу с мнемоническим именем. Макроподстановки объявляются в соответствии с грамматикой

$$\textit{макрос}(\text{M3}) = \text{"macro"}, \textit{перем\_имя}, \text{"="}, \textit{AB}, \text{";"};$$

Макроподстановки не являются уравнениями системы, а подстановка соответствующего арифметического выражения осуществляется во время формирования исполняемой модели.

Например, программная модель маятника, представленная на рис. 3, может быть записана с применением макроподстановок, как показано на рис. 5.

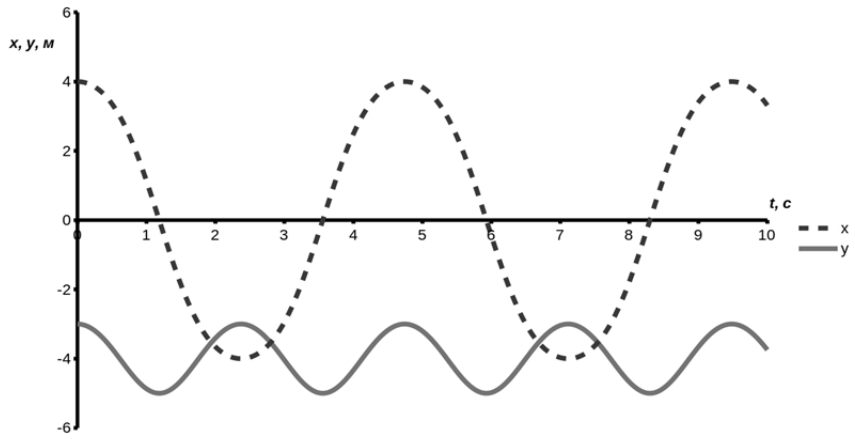

Рис. 4. Результаты расчета идеального маятника

Fig. 4. Simulation results of the simple gravity pendulum calculation

```
1  const l = 5.0;     // длина стержня
2  const m = 1.0;     // масса маятника
3
4  macro TLM = T / (l * m);
5
6  x' = v_x;
7  x(t0) = 4.0;
8  y' = v_y;
9  y(t0) ~= -3.0;
10 v_x' = -TLM * x;
11 v_x(t0) = 0.0;
12 v_y' = -TLM * y - g;
13 x * x + y * y = l * l;
```

Рис. 5. Программная модель идеального математического маятника с макроподстановкой

Fig. 5. A program model of the simple gravity pendulum with a macro

3. СПЕЦИФИКАЦИЯ ДИСКРЕТНОГО ПОВЕДЕНИЯ

Дискретное поведение ГС заключается в смене режимов при наступлении определенных событий.

Грамматика режима ГС на LISMA_HDS имеет вид

$$\textit{режим}(\text{M2}) = \text{"state"}, \textit{идент}, \text{"("}, \textit{ЛВ}, \text{")"}, \textit{реж\_тел}, [\textit{реж\_ист}];$$

Описание режима ГС начинается с ключевого слова `state`. Следом идет имя режима (идентификатор) и логическое выражение (*ЛВ*) в круглых скобках.

После предиката указывается тело режима, состоящее из набора уравнений, начальных условий и операторов удаления уравнений (*ур_удал*):

$$\textit{реж\_тел}\ (\mathrm{M4}) = \text{"\{"}, \{\textit{уравнение} \mid \textit{нач\_усл} \mid \textit{ур\_удал}\}, \text{"\}"};$$

$$\textit{ур\_удал}\ (\mathrm{D2}) = \texttt{"delete"},\ (\text{"*"} \mid \textit{спис\_мет}), \text{";"};$$

$$\textit{спис\_мет}\ (\mathrm{L1}) = \textit{идент}, \{\text{","}, \textit{идент}\};$$

Оператор `delete` позволяет удалить все (`*`) или только указанные путем перечисления их меток уравнения из совокупной системы.

Объявление уравнений с уже существующей меткой приведет к замене всех старых уравнений с этой меткой в совокупной системе на новые с той же самой меткой.

Объявление уравнения без метки или с неиспользованной ранее меткой означает его добавление в совокупную систему.

Добавление или замена уравнения, задание начального значения переменной в режиме приводят к необходимости проведения инициализации.

После тела режима указываются режимы, из которых в него можно перейти, в соответствии с грамматикой

$$\textit{реж\_ист}\ (\mathrm{M5}) = \texttt{"from"}, \textit{реж\_имя}, \{\text{","}, \textit{реж\_имя}\}, \text{";"};$$

$$\textit{реж\_имя}\ (\mathrm{M6}) = \textit{идент} \mid \texttt{"init"};$$

Если опустить конструкцию `from` , режим будет достижим из любого другого.

Логические выражения с типом событий (*ЛВ*) описываются следующей грамматикой:

$$\textit{ЛВ}\,(\mathrm{E3}) = \textit{ЛВТ}, \{\textit{или\_оп}, \textit{ЛВТ}\};$$

$$\textit{или\_оп} = \texttt{"or"} \mid \texttt{"OR"} \mid \text{"| |"};$$

$$\textit{ЛВТ}\,(\mathrm{E4}) = \textit{ЛВР}, \{\textit{и\_оп}, \textit{ЛВР}\};$$

$$\textit{и\_оп} = \texttt{"and"} \mid \texttt{"AND"} \mid \text{"\&\&"};$$

$$\textit{ЛВР}\,(\mathrm{E5}) = \textit{ЛВС}, \{\textit{рав\_оп}, \textit{ЛВС}\};$$

$$\textit{рав\_оп} = \text{"=="} \mid \text{"!="};$$

$$\textit{ЛВС}\,(\mathrm{E6}) = \textit{АВА}, \{\textit{отн\_оп}, \textit{АВА}\};$$

$$\textit{отн\_оп} = \text{"<="} \mid \text{">="} \mid \text{"<"} \mid \text{">"};$$

$$\textit{АВА}\,(\mathrm{E7}) = \textit{АВМ}, \{\textit{адд\_оп}, \textit{АВМ}\};$$

$$\textit{АВМ}\,(\mathrm{E8}) = \textit{УАВ}, \{\textit{мул\_оп}, \textit{УАВ}\};$$

$$\textit{мул\_оп} = \text{"*"} \mid \text{"/"};$$

$$УАВ(\text{E9}) = адд\_оп, УАВ | УАВ2;$$

$$УАВ2(\text{E10}) = не\_оп, УАВ | операнд;$$

$$не\_оп = \texttt{"not"}|\texttt{"NOT"}|\texttt{"!"};$$

$$операнд(\text{O1}) = \texttt{"("}, АВ, \texttt{")"} | перем | литерал | вызов | типизация;$$

$$литерал(\text{L2}) = ЦБЗ | вещ\_число;$$

$$типизация(\text{T1}) = тип\_соб, \texttt{"("}, ЛВ, \texttt{")"};$$

$$тип\_соб(\text{T2}) = одн\_соб | дву\_соб | тру\_соб | кри\_соб;$$

$$одн\_соб = \texttt{"unilateral"}|\texttt{"UNILATERAL"};$$

$$дву\_соб = \texttt{"bilateral"}|\texttt{"BILATERAL"};$$

$$тру\_соб = \texttt{"shortliving"}|\texttt{"SHORTLIVING"};$$

$$кри\_соб = \texttt{"ordinary"}|\texttt{"ORDINARY"};$$

$$АВ(\text{E2}) = ЛВ;$$

Таким образом, имеется возможность указать, что все событийные функции, входящие в предикат, описывают односторонние (*одн_соб*), двусторонние (*дву_соб*), труднообнаруживаемые (*тру_соб*), критичные к точности обнаружения (*кри_соб*) события. По умолчанию события являются критичными к точности обнаружения. Тип события может быть указан только в логическом выражении конструкции `state`. В логические выражения не могут входить производные переменных системы.

Если ограничение на событийную функцию попадает под действие нескольких указателей типа события, то его типом считается ближайший, а остальные игнорируются. Например, предикат

```
unilateral(x<0 and bilateral(y<0))
```

эквивалентен предикату без вложенных указателей типа

```
unilateral(x<0) and bilateral(y<0).
```

В качестве примера ГС рассмотрим математический маятник из предыдущего раздела, но уже с горизонтальной недеформируемой пластиной бесконечной длины на уровне $h = 0$, к которой прикреплен шарнир маятника (рис. 6). При столкновении с пластиной происходит абсолютно упругий отскок груза и последующее мгновенное разрушение стержня маятника.

Диаграмма состояний системы представлена на рис. 7, где условие перехода указано над стрелкой. Гибридная система начинает в режиме init, моделирующем движение маятника до разрушения.

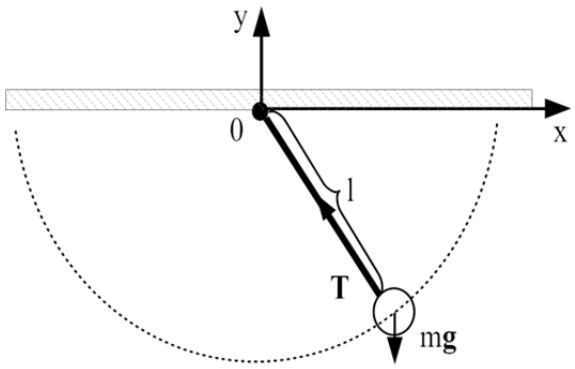

Рис. 6. Идеальный математический маятник с ограничением

Fig. 6. A simple gravity pendulum with an obstacle

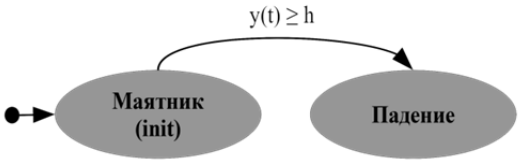

Рис. 7. Диаграмма состояний маятника с ограничением

Fig. 7. A state chart of the simple gravity pendulum with an obstacle

Режим «Маятник» имеет вид (3), но с дополнительным ограничением:

$$pr_1(y(t)) : y(t) - h < 0 .$$

После столкновения в момент времени t^* система переходит в режим «Падение»:

$$v_x'(t) = 0 ,\ v_x(t^*) = -v_x(t^* - 0) ,$$

$$x'(t) = v_x(t) ,\ x(t^*) = x(t^* - 0) ,$$

$$v_y'(t) = -g ,\ v_y(t^*) = -v_y(t^* - 0) ,$$

$$y'(t) = v_y(t) ,\ y(t^*) = y(t^* - 0) ,$$

$$pr_2(t) : \text{true} ,$$

где $a(t^* - 0)$ означает финальное значение переменной a в предыдущем режиме.

Рассмотрим программную модель системы на LISMA_HDS, представленную на рис. 8.

```
1  const l = 5.0;     // длина стержня
2  const m = 1.0;     // масса маятника
3  const h = 0.0;     // высота препятствия
4
5  x' = v_x;
6  x(t0) = 4.0;
7  y' = v_y;
8  y(t0) ~= -3.0;
9  eqs: m * v_x' = -T * x / l;
10 v_x(t0) = -5.0;
11 eqs: m * v_y' = -T * y / l - m * g;
12 cst: x * x + y * y = l * l;
13
14 state Falling(unilateral(y >= h))
15 {
16      x(t0) = -5.0;
17      y(t0) = h;
18      v_x(t0) = 0.0;
19      v_y(t0) = -v_y;
20
21      delete cst;
22 eqs: v_x' = 0.0;
23 eqs: v_y' = -g;
24 }
25 from init;
```

Рис. 8. Программная модель идеального математического маятника с ограничением

Fig. 8. A program model of the simple gravity pendulum with an obstacle

Строки 5–12 определяют локальное поведение ГС в режиме «Маятник» (`init`), где `init` – имя встроенного режима, в котором начинается вычислительный эксперимент.

Как только становится истинным условие `y>=h` с учетом односторонности события (указатель типа `unilateral`), выполняется переход в режим «Падение», описанный строками 14–25. Переход возможен только из режима `init`, что отражено в строке 25. При переходе удаляется уравнение-ограничение `cst`, заменяются дифференциальные уравнения `eqs`, включающие компоненты вектора скорости. Вертикальный компонент скорости меняет свой знак на противоположный в строке 19. Остальным переменным присваиваются новые явно заданные значения в строках 16–18. Это выполняется из-за невозможности идеально точного обнаружения события в связи с ограниченностью разрядной сетки компьютера.

На рис. 9 представлены результаты вычислительного эксперимента с моделью на временном интервале $[0;2]$.

Кроме возможности объявления режима, LISMA_HDS также поддерживает событийное управление [15]. Оно позволяет выполнять некоторые действия в начале каждого шага интегрирования, если соответствующее условие истинно. Причем введение в модель событийного управления не приводит к появлению отдельного режима или событийных функций. Поэтому алгоритмы обнаружения событий в таком случае не используются. Событийное управление осуществляется с помощью известной конструкции `if-else`:

соб_упр (I2) = "`if`", "(", *ЛВ*, ")", *реж_тел*, [*иначе*];

иначе (I4) = "`else`", *реж_тел*;

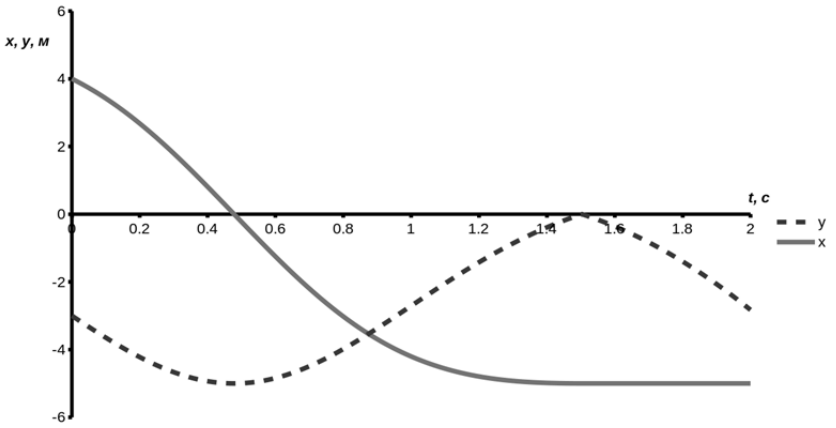

Рис. 9. Результаты расчета математического маятника с ограничением

Fig. 9. Simulation results of the simple gravity pendulum with an obstacle

Например, конструкция событийного управления в программной модели на рис. 10 не позволяет значению переменной `x` превысить 1.

```
1  x' = 1;
2
3  if (x > 1)
4  {
5      x(t0) = 1;
6  }
```

Рис. 10. Пример использования событийного управления

Fig. 10. Example of using event control

Также в LISMA_HDS встроена конструкция для явного задания события времени. Это может быть как единоразовое событие, так и повторяющееся заданное количество раз. Грамматика конструкции имеет вид

цикл_соб (C4) = "`at`", *время*,["`each`", *время*, "`repeat`",(*ЦБЗ* | "*")], *реж_тел*;

время (T3) = *литерал* | *идент*;

Конструкция начинается с ключевого слова `at` и времени возникновения первого события. После ключевого слова `each` указывается период события. В качестве этих двух параметров могут выступать как числовые литералы, так и именованные постоянные (переменные). За ключевым словом `repeat` следует число повторений – неотрицательное целое число. Если опустить необязательную часть `each-repeat`, событие произойдет только один раз – в момент времени, указанный после `at`. `*`, `repeat` означает беско-

нечно генерируемое событие времени. Такую конструкцию удобно использовать для задания, например, генератора импульсов [16]. На рис. 11 показана программная модель с генерацией прямоугольных импульсов с периодом 2 и скважностью 2, начиная с момента времени $t = 0$.

```
1  eq: pulse = 0;
2
3  at 0 each 2 repeat *
4  {
5       eq: pulse = 1;
6  }
7
8  at 1 each 2 repeat *
9  {
10      eq: pulse = 0;
11 }
```

Рис. 11. Генерация прямоугольных импульсов с помощью конструкции `at-each-repeat`

Fig. 11. Generating a square wave using the `at-each-repeat` construct

Объявлены два бесконечно повторяющихся явных события времени. Первое событие генерируется каждые две единицы модельного времени, начиная с $t_{s_1} = 0$, и начинается очередной прямоугольный импульс. Второе событие повторяется с тем же интервалом, но впервые возникает в $t_{s_2} = 1$ и отвечает за окончание импульса.

4. КЛАССИФИКАЦИЯ ГРАММАТИКИ

Для простоты, когда это возможно, будем считать терминальными символами нетерминальные, означающие логические и арифметические операторы, например *add_on*, а также типы событий и имена математических функций. Введем сокращенные обозначения для терминальных символов $G[\textit{модель}]$ (табл. 1).

Представленная грамматика является контекстно-свободной по классификации Хомского, так как все ее продукционные правила имеют вид [11]

$$A \to \alpha,\ \ A \in V_N,\ \ \alpha \in (V_T \cup V_N)^*,$$

где V_T и V_N представляют собой терминальный и нетерминальный словари соответственно, а $*$ означает итерацию.

Необходимым и достаточным условием принадлежности контекстно-свободной грамматики $G(V_T, V_N, P, S)$, где P – множество продукционных правил, S – начальный символ грамматики, классу LL(k) является

$$\forall A \to \beta \in P, A \to \gamma \in P\ (\beta \neq \gamma) : \mathrm{FIRST}(k, \beta\omega) \cap \mathrm{FIRST}(k, \gamma\omega) = \varnothing$$

для все цепочек ω таких, что $S \Rightarrow^* \alpha A\omega$ [12].

Таблица 1

Table 1

Сокращенные наименования терминальных символов

Abbreviated notations of the terminal symbols

Терминал	Сокращение	Терминал	Сокращение	Терминал	Сокращение
":"	c1	"="	e1	идент	i1
"("	l1	")"	r1	"t0"	t0
"~="	e2	";"	s1	"const"	c2
","	c3	"for"	f1	ЦБЗ	n1
вещ_число	n2	"{"	l2	"}"	r2
"'"	d1	"["	l3	"]"	r3
адд_оп	a1	"*"	m1	"/"	m2
"state"	m3	"delete"	d2	"macro"	m4
"from"	f2	или_оп	o1	и_оп	a2
равн_оп	o2	отн_оп	o3	не_оп	n3
тип_соб	t1	имя_мат_функ	a6	перем_инд	a5
"repeat"	a4	"if"	i2	"else"	e3
"init"	i3	"at"	a3	"each"	e4

$\mathrm{FIRST}(k,\alpha)$, $k \in \mathbb{Z}_+$, $\alpha \in (V_T \cup V_N)^*$, представляет собой множество выводимых из α терминальных цепочек, обрезанных до k символов [10, 12].

В табл. 2 представлены множества $\mathrm{FIRST}(1,\alpha)$ и $\mathrm{FIRST}(2,\alpha)$ для нескольких альтернативных продукционных правил G[*модель*] с пересекающимися $\mathrm{FIRST}(1,\alpha)$. Совпадающие элементы множеств выделены полужирным шрифтом.

Таблица 2

Table 2

Множества $\mathrm{FIRST}(1,\alpha)$ **и** $\mathrm{FIRST}(2,\alpha)$

Sets $\mathrm{FIRST}(1,\alpha)$ **and** $\mathrm{FIRST}(2,\alpha)$

A	α	$\mathrm{FIRST}(1,\alpha)$	$\mathrm{FIRST}(2,\alpha)$
A4	A1	**i1**, **a5**	i1e1, a5e1
	E2	**i1**, a1, n3, l1, a6, t1, **a5**, n1, n2	a1a1, a1n3, n3a1, n3n3, a1l1, a1i1, a1a5, a1n1, a1n2, a1a6, a1t1, n3l1, n3i1, n3a5, n3n1, n3n2, n3a6, n3t1, l1l1, l1i1, l1a5, l1n1, l1n2, l1a6, l1t1, l1a1, l1n3, a6l1, t1l1, i1d1, a5d1, i1a1, i1m1, i1m2, i1o1, i1a2, i1o2, i1o3, a5a1, a5m1, a5m2, a5o1, a5a2, a5o2, a5o3, n1a1, n1m1, n1m2, n1o1, n1a2, n1o2, n1o3, n2a1, n2m1, n2m2, n2o1, n2a2, n2o2, n2o3
E1	i1c1E2e1E2s1	**i1**	i1c1
	E2e1E2s1	**i1**, a1, n3, l1, a6, t1, a5, n1, n2	$\mathrm{FIRST}(2,E2) \cup \{\text{i1e1}, \text{a5e1}, \text{n1e1}, \text{n2e1}\}$

Окончание табл. 2

End of Tab. 2

A	α	FIRST(1,α)	FIRST(2,α)
D1	C1	c2	c2i1, c2a5
	E1	**i1**, a1, n3, l1, a6, t1, **a5**, n1, n2	FIRST(2,i1c1E2e1E2s1) ∪ FIRST(2,E2e1E2s1)
	I1	**i1**, **a5**	i1l1, a5l1
	M2	m3	m3i1
	I2	i2	i2l1
	M3	m4	m4i1, m4a5
	C4	a3	a3i1, a3n1, a3n2
	C2	f1	f1i1

Аналогичным D1 образом строятся множества FIRST(1,α) и FIRST(2,α) для элементов тела цикла и тела режима, из которых так же выводятся цепочки E1 и I1, а поэтому пересекаются FIRST(1,α), но не пересекаются FIRST(2,α).

Таким образом, порождающая грамматика *G*[*модель*] языка LISMA_HDS относится к классу LL(2).

ЗАКЛЮЧЕНИЕ

Представленный в работе декларативный язык моделирования общего назначения LISMA_HDS инструментальной среды ИСМА включает возможности непосредственного или алгоритмического объявления программных постоянных, задачи Коши для неявной системы ДАУ, начальных приближений переменных, а также позволяет задавать явные события времени, режимы функционирования и переходы между ними по событиям разных типов, использовать макроподстановки и реализовывать событийное управление.

Доказано, что порождающая грамматика LISMA_HDS принадлежит к подклассу LL(2) контекстно-свободных грамматик. Для грамматик типа LL(k) возможно построение k-предсказывающего нисходящего синтаксического анализатора, выполняющего однозначный и безвозвратный разбор входной цепочки символов за линейное относительно ее длины время. Так как новая грамматика относится к тому же классу, что и грамматика другого текстового языка моделирования инструментальной среды ИСМА – LISMA_PDE, обеспечивается преемственность методов анализа.

СПИСОК ЛИТЕРАТУРЫ

1. *Brenan K.E., Campbell S.L., Petzold L.R.* Numerical solution of initial-value problems in differential-algebraic equations. – PA, USA: Society for Industrial, Applied Mathematics, 1995. – 251 p.

2. *Mazzia F., Magherini C.* Test set for initial value problem solvers, release 2.4, Report 4/2008. – Bari, Italy: University of Bari, 2008. – URL: http://pitagora.dm.uniba.it/~testset (accessed: 09.03.2021).

3. *Cellier F.E., Kofman E.* Continuous system simulation. – USA: Springer, 2006. – 644 p.

4. *Urquía Moreleda A., Martín Villalba C.* Modeling and simulation in engineering using Modelica. – Madrid, Spain: UNED Editorial, 2018. – 298 p.

5. *Колесов Ю.Б., Сениченков Ю.Б.* Моделирование систем. Динамические и гибридные системы: учебное пособие. – СПб.: БХВ-Петербург, 2012. – 224 с.

6. *Shornikov Yu.V., Dostovalov D.N.* Fundamentals of event-continuous system simulation theory. – Novosibirsk: NSTU Publ., 2018. – 175 p.

7. *Esposito J.M., Kumar V., Pappas G.J.* Accurate event detection for simulating hybrid systems // Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control (HSCC '01). – Berlin, Heidelberg: Springer, 2001. – P. 204–217.

8. *Попов Е.А., Шорников Ю.В.* Детекция событий разного типа в гибридных динамических системах // Научный вестник НГТУ. – 2020. – № 4 (80). – С. 159–176.

9. *Fritzson P.* Principles of object-oriented modeling and simulation with Modelica 3.3: A cyber-physical approach. – Wiley-IEEE Press, 2015. – 1256 p.

10. Compilers: principles, techniques, and tools / A.V. Aho, M.S. Lam, R. Sethi, J.D. Ullman. – Pearson Education, 2007. – 1040 p.

11. *Шорников Ю.В.* Теория и практика языковых процессоров: учебное пособие. – Новосибирск: Изд-во НГТУ, 2004. – 208 с.

12. *Гордеев А.В., Молчанов А.Ю.* Системное программное обеспечение. – СПб.: Питер, 2001. – 736 с.

13. *Бессонов А.В.* Символьная спецификация и анализ программных моделей гибридных систем: дис. ... канд. техн. наук. – Новосибирск, 2016. – 164 с.

14. ISO/IEC 14977:1996 Information technology – Syntactic metalanguage – Extended BNFEBNF Standard. – ISO, 1996. – 12 p.

15. *Томилов И.Н.* Синтаксически ориентированные и графические средства описания и анализа моделей гибридных систем: дис. ... канд. техн. наук. – Новосибирск, 2010. – 175 с.

16. *Shornikov Yu., Popov E.* Modeling and simulation of electronic devices in the ISMA environment // 2019 International Seminar on Electron Devices Design and Production (SED). – IEEE, 2019. – P. 1–4.

Попов Евгений Александрович, ассистент кафедры автоматизированных систем управления Новосибирского государственного технического университета. Основное направление научных исследований – языки моделирования и алгоритмы численного анализа гибридных динамических систем. Имеет 30 публикаций. E-mail: e.popov@corp.nstu.ru

Шорников Юрий Владимирович, доктор технических наук, профессор, профессор кафедры автоматизированных систем управления Новосибирского государственного технического университета. Основное направление научных исследований – математическое, лингвистическое и программное обеспечение моделирования гибридных динамических систем. Имеет около 200 публикаций. E-mail: shornikov@inbox.ru

Popov Evgeny A., teaching assistant at the Automated Control Systems Department of Novosibirsk State Technical University. The main area of his research is modeling languages and simulation algorithms for hybrid dynamical systems. He has 30 publications. E-mail: e.popov@corp.nstu.ru

Shornikov Yury V., D.Sc. (Eng.), professor, professor at the Automated Control Systems Department of Novosibirsk State Technical University. The main area of his research is mathematical and linguistic software for modeling and simulation of hybrid dynamical systems. He has about 200 publications. E-mail: shornikov@inbox.ru

DOI: 10.17212/2782-2001-2021-1-103-122

LISMA_HDS language for modeling heterogeneous dynamic systems[*]

E.A. POPOV[a], *YU.V. SHORNIKOV*[b]

Novosibirsk State Technical University, 20, K. Marx Prospekt, Novosibirsk, 630073, Russian Federation

[a] *e.popov@corp.nstu.ru* [b] *shornikov@inbox.ru*

Abstract

Heterogeneous dynamic systems (HDS) simultaneously describe processes of different physical nature. Systems of this kind are typical for numerous applications. HDSs are characterized by the following features. They are often multimode or hybrid systems. In general, their modes are defined as initial value problems (Cauchy problems) for implicit differential-algebraic systems of equations. Due to the presence of heterogeneous dynamic components or processes evolving in both time and space, the dimension of the complete system of equations may be pretty high. In some cases, the system of equations has an internal structure, for instance, the differential-algebraic system of equations approximating a partial differential equation by the method of lines. An original huge system of equations can then be algorithmically rewritten in a compact form. Moreover, heterogeneous hybrid dynamical systems can generate events of qualitatively different types. Therefore one has to use different numerical event detection algorithms.

Nowadays, HDSs are modeled and simulated in computer environments. The modeling languages widely used by engineers do not allow them to fully specify all the properties of the systems of this class. For instance, they do not include event typing constructs.

That is why a declarative general-purpose modeling language named LISMA_HDS has been developed for the computer-aided modeling and ISMA simulation environment. The language takes into account all of the characteristic features of HDSs. It includes constructs for plain or algorithmic declaration of model constants, initial value problems for explicit differential-algebraic systems of equations, and initial guesses for variables. It also allows researchers to define explicit time events, modes and transitions between them upon the occurrence of events of different types, to use macros and implement event control.

LISMA_HDS is defined by a generative grammar in an extended Backus-Naur form and semantic constraints. It is proved that the grammar belongs to the LL(2) subclass of context-free grammars.

Keywords: heterogeneous dynamic systems, hybrid dynamical systems, modeling languages, event typing, formal grammars, ISMA modeling and simulation environment

REFERENCES

1. Brenan K.E., Campbell S.L., Petzold L.R. *Numerical solution of initial-value problems in differential-algebraic equations*. PA, USA, Society for Industrial, Applied Mathematics, 1995. 251 p.

2. Mazzia F., Magherini C. *Test set for initial value problem solvers, release 2.4, Report 4/2008*. Bari, Italy, University of Bari, 2008. Available at: http://pitagora.dm.uniba.it/~testset (accessed 09.03.2021).

3. Cellier F.E., Kofman E. *Continuous system simulation*. USA, Springer, 2006. 644 p.

4. Urquía Moreleda A., Martín Villalba C. *Modeling and simulation in engineering using Modelica*. Madrid, Spain, UNED Editorial, 2018. 298 p.

[*] *Received 28 October 2020.*

5. Kolesov Yu.B., Senichenkov Yu.B. *Modelirovanie sistem. Dinamicheskie i gibridnye sistemy* [System modeling and simulation. Dynamical and hybrid systems]. St. Petersburg, BHV-Peterburg Publ., 2012. 224 p.

6. Shornikov Yu.V., Dostovalov D.N. *Fundamentals of event-continuous system simulation theory*. Novosibirsk, NSTU Publ., 2018. 175 p.

7. Esposito J.M., Kumar V., Pappas G.J. Accurate event detection for simulating hybrid systems. *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control (HSCC '01)*. Berlin, Heidelberg, Springer, 2001, pp. 204–217.

8. Popov E.A., Shornikov Yu.V. Detektsiya sobytii raznogo tipa v gibridnykh dinamicheskikh sistemakh [Detection of different type events in hybrid dynamical systems]. *Nauchnyi vestnik Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta = Science bulletin of the Novosibirsk state technical university*, 2020, no. 4 (80), pp. 159–176.

9. Fritzson P. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A cyber-physical approach*. Wiley-IEEE Press, 2015. 1256 p.

10. Aho A.V., Lam M.S., Sethi R., Ullman J.D. *Compilers: principles, techniques, and tools*. Pearson Education, 2007. 1040 p.

11. Shornikov Yu.V. *Teoriya i praktika yazykovykh protsessorov* [Theory and practice of language processors]. Novosibirsk, NSTU Publ., 2004. 208 p.

12. Gordeev A.V., Molchanov A.Yu. *Sistemnoe programmnoe obespechenie* [System software]. St. Petersburg, Piter Publ., 2001. 736 p.

13. Bessonov A.V. *Simvol'naya spetsifikatsiya i analiz programmnykh modelei gibridnykh sistem*. Diss. kand. tekhn. nauk [Textual specification and analysis of hybrid system program models. PhD eng. sci. diss.]. Novosibirsk, 2016. 164 p.

14. *ISO/IEC 14977:1996 Information technology – Syntactic metalanguage – Extended BNFEBNF Standard*. ISO, 1996. 12 p.

15. Tomilov I.N. *Sintaksicheski orientirovannye i graficheskie sredstva opisaniya i analiza modelei gibridnykh sistem*. Diss. kand. tekhn. nauk [Syntax-oriented and visual tools for describing and analyzing hybrid system models. PhD eng. sci. diss.]. Novosibirsk, 2010. 175 p.

16. Shornikov Yu., Popov E. Modeling and simulation of electronic devices in the ISMA environment. *2019 International Seminar on Electron Devices Design and Production (SED)*. IEEE, 2019, pp. 1–4.

Для цитирования:

Попов Е.А., Шорников Ю.В. Язык моделирования гетерогенных динамических систем LISMA_HDS // Системы анализа и обработки данных. – 2021. – № 1 (81). – С. 103–122. – DOI: 10.17212/2782-2001-2021-1-103-122.

For citation:

Popov E.A., Shornikov Yu.V. Yazyk modelirovaniya geterogennykh dinamicheskikh sistem LISMA_HDS [LISMA_HDS language for modeling heterogeneous dynamic systems]. *Sistemy analiza i obrabotki dannykh = Analysis and data processing systems*, 2021, no. 1 (81), pp. 103–122. DOI: 10.17212/2782-2001-2021-1-103-122.