Specification and simulation of material science problems in ISMA

Yury V Shornikov and Evgeny A Popov

Automated Control Systems Department, Novosibirsk State Technical University, 20 Prospekt K. Marksa, Novosibirsk, 630073, Russia

E-mail: shornikov@inbox.ru

Abstract. The model types supported by a modeling and simulation environment called ISMA are presented. A fragment of the grammar of the general-purpose modeling language LISMA used in ISMA is given. A typical material science and engineering problem, namely modeling and simulation of the heat transfer within a beam, is solved using ISMA. A grammar for specifying chemical kinetics problems is briefly presented in order to demonstrate how the developed mathematical software suite can be extended to a new domain.

1. Introduction
Material science concerns itself with studying and designing materials. Nowadays, engineers often prefer a real experiment to a numerical experiment, simulation, due to the higher cost, danger, or even impossibility of the former. Before simulating a system, an engineer has to mathematically describe it in an unambiguous manner, i.e. to model it. Many problems arising in the field of material science and engineering are modeled by partial differential equations, in general with discontinuities. Since the majority of differential equations cannot be symbolically solved, one has to employ numerical methods in order to obtain the solution. Discretization of a partial differential equation yields a high dimensional system of equations. It is a laborious and error-prone process, that is why it is automated in modern modeling and simulation environments.

2. Supported model types
ISMA (the acronym for “Computer-Aided Analysis Tools” in Russian) is an environment for modeling and simulation of complex dynamical systems [1]. Although, inside ISMA, any system is represented as a hybrid system with continuous modes defined by differential algebraic systems of equations, the environment supports different types of models. The first type is the Cauchy problem for delay differential equations

\[ \begin{align*}
    y' &= f(t, y(t), y(t - \tau)), \\
    t &\in [t_0, t_f], \\
    y(t_0) &= y_0, \\
    y(t) &= \varphi(t), \quad t \in [t_0 - \tau, t_0). 
\end{align*} \tag{1} \]
where $y \in R^N$ is the state vector; $t$ is an independent variable; $\varphi \in R^r$ is the delay vector function, $r \leq N$; $\tau = \{\tau_1, \ldots, \tau_r\}^T$ is the vector of delays; $f : R \times R^N \rightarrow R^N$ is a nonlinear Lipschitz continuous vector function; $y_0 \in R^N$ is the vector of initial conditions.

System (1) can be transformed into the Cauchy problem

$$y' = f(t, y), \quad y(t_0) = y_0,$$

where $y \in R^N$ is the state vector; $f : R \times R^N \rightarrow R^N$ is a nonlinear Lipschitz continuous vector function; $y_0 \in R^N$ is the vector of initial conditions, using the “method of steps” or Pade approximation [2].

This type of models is widely used for representing electromechanical, control systems, etc. However, their application to discrete-continuous or hybrid systems (HS), which are characterized by both discrete and continuous behaviors, is limited [3, 4]. Therefore, a new version of ISMA has been developed. It supports models with events, which make the system to switch to new modes. The behavior of an HS in a certain mode is defined by differential equations of the form

$$y' = f(t, y),$$

$$y(t_0) = y_0,$$  \hspace{1cm} (3)

where $y \in R^N$ is the state vector; $f : R \times R^N \rightarrow R^N$ is a nonlinear vector function that is Lipschitz continuous in the corresponding mode and satisfies the Carathéodory conditions in the simulation interval $[t_0, t_k]$; $y_0 \in R^N$ is the vector of initial conditions; $g(t, x) : R \times R^N \rightarrow R^S, S = 1, 2, \ldots$ is the continuously differentiable nonlinear vector function of constraints.

Contrary to traditional mathematical models (1) and (2), inequalities defining when the mode is active are added to a mode behavior in (3). The presence of such constraints makes symbolic simulation of HSS much more difficult and limits the application of traditional numerical simulation techniques to discrete-continuous systems. However, the switch from (2) to (3) is not enough for modeling of real dynamical phenomena in complex systems in practice. For example, electromechanical and electromagnetic transient phenomena in electric power systems are described by Park-Gorev’s differential-algebraic systems of equations. Taking into account the events, these models can be represented as

$$y' = f(t, x, y), \quad x = \varphi(t, x, y),$$

$$y(t_0) = y_0, \quad t \in [t_0, t_k],$$

$$pr : g(t, y) < 0,$$  \hspace{1cm} (4)

where $y \in R^N$ is the state vector; $f : R \times R^N \times R^{N_x} \rightarrow R^N$ is a nonlinear vector function that is Lipschitz continuous in the corresponding mode and satisfies the Carathéodory conditions in the simulation interval $[t_0, t_k]$; $y_0 \in R^N$ is the vector of initial conditions; $g(t, x) : R \times R^N \rightarrow R^S, S = 1, 2, \ldots$ is the continuously differentiable nonlinear vector function of constraints; $x \in R^{N_x}$ is the vector of algebraic variables; $I$ is an independent variable; $\varphi(t, x, y) : R \times R^{N_x} \times R^N \rightarrow R^{N_x}$ is the vector function of algebraic equations.

By using (1) – (4), one can model a rather wide variety of applications with discrete-continuous phenomena and lumped parameter systems. From the other hand, distributed parameter systems, which are described by partial differential equations (PDEs), are not of less interest in practice. Numerical simulation of phenomena of this kind is difficult due to approximation of PDEs by differential-difference equations of high orders. It is even more difficult to simulate such a system with constraints or a so-
called hybrid system with distributed parameters (HSDP) [5]. We will consider only HSDP modes described by the parabolic PDEs with Dirichlet and Neumann boundary conditions

\[
\frac{\partial z}{\partial t} = \psi(t, z, p, \frac{\partial z}{\partial p}, \frac{\partial^2 z}{\partial p^2}), \quad t \in [t_0, t_k],
\]

\[
z(t_0, p) = z_0, \quad z(t, p_0) = \delta(t, p_0),
\]

\[
z(t, p_m) = \delta(t, p_m),
\]

\[
\frac{\partial z}{\partial p} \big|_{p_0} = \nu(t, p_0), \quad \frac{\partial z}{\partial p} \big|_{p_m} = \nu(t, p_m),
\]

\[
pr(g(t, z)) < 0,
\]

where \( z \in R^{N_z} \) is the state vector; \( \psi : R^{R^{N_z} \times R^{N_p} \times R^{2N_p}} \rightarrow R^{N_z} \) is a nonlinear vector function; \( p \) is the vector of spatial variables; \( z_0 \in R^{N_z} \) is the vector of initial conditions; \( \delta \) is the vector of Dirichlet boundary conditions; \( \nu \) is the vector of Neumann boundary conditions; \( g(t, z) : R^{R^{N_z}} \rightarrow R^5, \ S = 1,2, \ldots \) is the event function used in the predicate \( pr(g(t, z)) < 0 \).

Several modeling languages for specifying HSs of types (1) – (5) were developed for the modeling and simulation environment ISMA. The most flexible and general-purpose one is the textual language LISMA (Language of ISMA) [1, 5]. It includes constructs for describing discrete and continuous behaviors as models (1) – (5), event control, macro definitions, etc. Also LISMA has elements for defining spatial variables.

3. LISMA grammar

The grammar of the LISMA language is given in extended Backus-Naur form (EBNF) [6]. This form is the most laconic and has more comprehensive elements, which allow one to make the description of a grammar simpler and smaller.

The grammar of LISMA for program models (\( pm\_lisma \)) is

\[
\begin{align*}
pm\_lisma & \rightarrow \text{statement} ' ; ' | \text{statement} ' ; ' pm\_lisma \\
\text{statement} & \rightarrow \text{const} [\text{equation} | \text{macro} | \text{state}] \text{ pseudo\_state} \\
\text{init\_cond} | \text{app\_var} | \text{boundary}
\end{align*}
\]

A program model is a sequence of statements separated by semicolons and includes a description of the continuous and discrete behaviors of an HS. The constructs describing modes (\( state \)) and if-else statements (\( pseudo\_state \)) represent event functions \( g(t, y) < 0 \) and are intended for specifying discrete behaviors. The language elements for defining continuous behaviors are constants (\( const \)), systems of equations (\( equation \)) of types (1) – (5) with expressions in their right-hand sides, initial conditions (\( init\_cond \)), spatial variable definitions (\( app\_var \)), and boundary conditions (\( boundary \)).

The grammar of the \( const \) construct is

\[
\begin{align*}
const & \rightarrow ' const ' const\_body [ ' , ' const\_body ] \\
const\_body & \rightarrow (\text{var\_ident} ' = ' ) \{ \text{var\_ident} ' = ' \} \text{ expression} \\
\text{var\_ident} & \rightarrow \text{Identifier}
\end{align*}
\]

A \( const \) statement consists of the terminal symbol ‘\( const \)’ followed by a sequence of identifiers and equal signs. The last equal sign ends with an \( expression \). So the user can specify a list of constants having the same value.
The grammar of \textit{equation} is

\begin{align}
\text{equation} & \rightarrow \text{variable \,'=''\,expression} \\
\text{variable} & \rightarrow \text{var\_ident \mid \text{derivative\_ident} } \\
\text{derivative\_ident} & \rightarrow \text{var\_ident '}' \{ '''' \} \\
& \mid \text{'der''('\,\text{var\_ident},'\,\text{DecimalLiteral})'}
\end{align}

The grammar of \textit{macro} is

\begin{align}
\text{macro} & \rightarrow \text{macro\,'\,Macro\text{Identifier}'=''\,expression} \\
\text{Macro\text{Identifier}} & \rightarrow \#\,\{(\,\text{Letter\mid IDDigit}\,)(\,\text{Letter\mid IDDigit}\,)\}
\end{align}

Modes and event control are specified in LISMA according to the grammar

\begin{align}
\text{state} & \rightarrow \text{state\,'\,state\_name\,'\,'\,expression\,'\,\}'\,state\_body} \\
& \mid \text{state\_from} \\
\text{state\_body} & \rightarrow \{ \text{\,'\,equation\,'\,setter\,\}'\,\}' \\
\text{state\_from} & \rightarrow \text{from\,'\,Identifier\,'\,'\,Identifier\,} \\
\text{pseudo\_state} & \rightarrow \text{if\,'\,'\,expression\,'\,\}'\,state\_body} \\
& \mid \text{pseudo\_state\_else} \\
\text{pseudo\_state\_else} & \rightarrow \text{else\,'\,state\_body}
\end{align}

The grammar for \textit{app\_var} is

\begin{align}
\text{app\_var} & \rightarrow \text{var\,'\,Identifier\,'\,'\,app\_var\_bound\,'\,\,} \\
& \mid \text{app\_var\_bound\,'\,app\_var\_tail} \\
\text{app\_var\_bound} & \rightarrow \{'-\}'\,\,\text{DecimalLiteral} \\
\text{app\_var\_tail} & \rightarrow \text{apx\,'\,DecimalLiteral\,'\,step'} \\
& \mid \text{( FloatLiteral\mid DecimalLiteral) }
\end{align}

Boundary conditions are specified in LISMA according to the grammar

\begin{align}
\text{boundary} & \rightarrow \text{edge\,'\,edge\_eq\,'\,on\,'\,Identifier\,edge\_side} \\
\text{edge\_eq} & \rightarrow \{ \text{Identifier\mid Partial\,operand\,}'='\} \\
& \mid \text{( FloatLiteral\mid DecimalLiteral )} \\
\text{edge\_side} & \rightarrow \text{left\,'\,'right\,'\,'both'}
\end{align}

Production rules (6) – (12) include the nonterminal symbols Identifier, expression, DecimalLiteral, FloatLiteral, Letter, IDDigit, state\_name, setter, Partial\_operand. The grammar for these constructs is trivial and does not differ from generally used.

4. Heat transfer

Let us consider a heat transfer model for a beam in one dimension. Assume that the temperatures at the ends of the beam are given, and the beam side surface is heat-insulated. Let the x-axis be directed along the beam axis, and the beam’s ends be at the points $x = 0$ and $x = L$. Then the problem consists in finding the dependence of the temperature $u$ on time at different points along the beam, i.e. a function $u(x,t)$ of two variables. The function has to satisfy the heat equation
\[ \rho c \frac{\partial u}{\partial t} = \lambda \frac{\partial^2 u}{\partial x^2}, \]

where \( 0 < x < L; \lambda \) is the thermal conductivity, \( \lambda > 0; \rho \) is the beam density; \( c \) is the heat capacity.

Also, in order to obtain a correct problem statement, the initial and boundary conditions have to be defined:

\[ u(x,0) = f(x), \quad u(0,t) = \varphi_1(t), \quad u(L,t) = \varphi_2(t), \]

where \( f(x) \) is the initial condition, a known function defined along the body; \( \varphi_1, \varphi_2 \) are the Dirichlet boundary conditions, known functions of time.

Considering the uniform grid of size \( J \) for \( x \), we obtain the explicit differential-algebraic system of equations

\[ \frac{du_j}{dt} = \frac{\lambda}{\Delta x^2 \rho c} (u_{j+1} - 2u_j + u_{j-1}), \quad 2 \leq j \leq J-1, \]

\[ u_j = \varphi_2(t), \]

where \( \Delta x = L/(J-1); u_j \) approximates \( u(x_j,t), \quad x_j = (j-1)\Delta x, \quad 1 \leq j \leq J \).

Since the manual obtaining of (13) is quite a laborious and error-prone task, the process is automated in ISMA.

The following parameter values will be used in all the experiments: \( L = 0.2m, \lambda = 384\frac{W}{m\cdot K}, \rho = 8800 \frac{kg}{m^3}, \quad c = 381\frac{J}{kg\cdot K} \), the duration of each simulation run is 30 s.

### 4.1. One-mode heat equation

Let us consider the behavior of the model without an external influence. The one-mode computer model is shown in figure 1a. In this experiment, the temperature on the left end is 0 K, i.e. \( \varphi_1(t) = 0 \), and the temperature on the right end is 50 K, i.e. \( \varphi_2(t) = 50 \). The initial condition is \( f(x) = 323 \).

```plaintext
1 // spatial variable
2 var x0, 0.2L apx 21;
3 // parameters
4 const lambda = 384, c = 381, rho = 8800;
5 // equations
6 u' = lambda * D(u, x, 2) / (rho * c);
7 bc = 800;
8 // initial conditions
9 u(0) = 323;
10 // boundary conditions
11 edge u = bc on x both;
12 // discrete behavior
13 state st1 (TIME > 15) {
14   bc = 100;
15 } from init;
```

Figure 1. Models in LISMA.

The simulation results are shown in figure 2. ISMA’s library of numerical methods contains original explicit one-step methods with accuracy and stability control, which can be used to solve moderately stiff systems and can be more efficient than implicit ones when solving high dimensional systems [7, 8].
4.2. Stepwise temperature change

Let us consider the case, when the boundary conditions at both ends of the body instantly change from 600 K to 100 K at \( t = 15 \), i.e.

\[
\varphi_1(t) = \varphi_2(t) = \begin{cases} 
600, & t < 15, \\
100, & t \geq 15.
\end{cases}
\]

The initial condition is \( f(x) = 323 \). The computer model is shown in figure 1b.

The simulation results are depicted in figure 3 and allow one to study the temperature dynamics along the whole beam. The change of the temperature can be clearly seen at the beam’s edges and is hardly distinguishable in the middle part.
5. Unification
Unification of the software suite consists in applying the base language LISMA with minor changes in the generative grammar to problems from different domains that can be modeled as (1) – (5). As an example, let us consider chemical kinetics problems, which belong in type (2). A language called LISMA+ has been developed for specifying problems of this kind [5]. The language extends the general language for specifying dynamical and hybrid systems, LISMA.

Let us represent the generative grammar of chemical reactions $G[C]$ as $C \rightarrow SC \mid S$ and $S \rightarrow E_S aE_S$, where $S$ are elementary reactions; $E_S$ is a subset of arithmetic expressions; $a \in \Sigma^*$ is an element of the Kleene star of the terminal alphabet of the base grammar $G[expression]$. This element is unambiguously defied by $a \rightarrow 'id'=\varepsilon \mid c$, where the $id$ identifier denotes the elementary reaction speed. $E_S$ is selected $G[E_S] \subseteq G[expression]$ to hold. Hence, the production rules of the new grammar have to be of the form $E_S \rightarrow T \mid T + E_S$, $T \rightarrow O \mid O^* T$, and $O \rightarrow id \mid c$, where $id$ is the chemical reaction reagent identifier, which is a variable written without a subscript, so it is in accord with the rules of writing simple variables in $G[expression]$; $c$ is a constant or an unsigned integer, which stands for the number of reagents. Due to the inclusion of $G[E_S]$, the unambiguous analysis methods employed for the base grammar $G[expression]$ are inherited and used. Therefore, ISMA’s language processor does not have to be vastly changed.

This example shows how the important problem of unifying the mathematical software components of the developed modeling and simulation environment can be solved.

Acknowledgments
This work was supported by the grant of the Russian Foundation for Basic Research (RFBR grant 17-07-01513) and by the grant of the Education, Audiovisual and Culture Agence (EU), programme ERASMUS+ Capacity building in higher education, project 573751-EPP-1-2016-1-DE-EPPKA2-CBHE-JP, Innovative teaching and learning strategies in open modelling and simulation environment for student-centered engineering education.

References
[1] Shornikov Y V and Dostovalov D N 2017 Features of the ISMA modeling and simulation environment Proc. Int. Multi-Conf. on Engineering, Computer and Information Sciences SIBIRCON-2017 (Novosibirsk, Russia) (New York, USA: IEEE) p 332–7
[2] Hairer E, Norsett S P and Wanner G 1993 Solving Ordinary Differential Equations I. Nonstiff Problems (Berlin, Germany: Springer-Verlag Berlin Heidelberg)
[3] Schaft A J van der and Schumacher H 2000 An Introduction to Hybrid Dynamical Systems (London, UK: Springer-Verlag London)
[4] Goebel R, Sanfelice R G and Teel A R 2012 Hybrid Dynamical Systems: Modeling, Stability, and Robustness (Princeton, USA: Princeton University Press)
[5] Shornikov Y V and Dostovalov D N 2018 Fundamentals of Event-Continuous System Simulation Theory (Novosibirsk, Russia: NSTU Publisher)
[6] Aho A V, Lam M S, Sethi R and Ullman J 2006 Compilers: Principles, Techniques, and Tools (Boston, USA: Addison Wesley)
[7] Levykin A I, Novikov A E and Novikov E A 2018 Third Order (m, k)-Method for Solving Stiff Systems of ODEs and DAEs XIV Int. Scientific-Technical Conference APEIE-2018 (Novosibirsk, Russia) (New York, USA: IEEE) p 158–63
[8] Novikov A E, Novikov E A and Rybkov M V 2017 An algorithm of variable structure based on three-stage explicit-implicit methods Sib. Elec. Math. Rep. 14 433–42