Modelling concurrent systems with Alvis

Marcin Szpyrka
AGH University of Science and Technology, Department of Automatics

Abstract: Alvis is a new modeling language for developing concurrent (embedded) systems. The language is being developed within the confines of the Alvis project at AGH University of Science and Technology, Department of Automatics. The Alvis language combines hierarchical graphical modelling with a high level programming language. Moreover, a formal verification of a model, based on an LTS graph (Labelled Transition System) is possible. The paper describes selected features of the language and the future plans of the project.

Keywords: Alvis, embedded systems, concurrent systems, modelling and verification

Beginning of the Alvis project dates back to April 2009. The aim of the project is to work out a language suitable for efficient modelling and formal verification of concurrent systems. Especially, we focus on embedded systems. Alvis [1] is a modelling language for real-time concurrent systems. The key concept of Alvis is *agent*. The name has been taken from the CCS process algebra [2] and denotes any distinguished part of the system under consideration with defined identity persisting in time. The Alvis high level programming language is used to describe behaviour of individual agents, and the Alvis graphical language is used to define data and control flow among agents.

Similar to other formal methods, Alvis provides a possibility of formal verification of models. An Alvis model can be transformed into a *labelled transition system* (LTS). After encoding such a graph using the *Binary Coded Graphs* (BCG) format, its properties are verified with the CADP toolbox [3]. The paper presents a survey of main features of the language and describes the state-of-art of the Alvis project.

1. Alvis Code Language

The Alvis Code Language is used to implement the so-called *code layer* of a model. It uses both Haskell functional programming language [4] and original Alvis statements. The layer is used to define data types used in the model under consideration, functions for data manipulation and the behaviour of individual agents. Selected Alvis statements are given in Table 1.

Tab. 1. Selected Alvis statements
Tab. 1. Wybrane konstrukcje języka Alvis

Statement	Description
`cli`	Turns off interrupts handlers.
`critical {...}`	Defines a critical section.
`delay ms`	Delays an agent execution.
`exec x = e`	Assigns the result of the expression to the parameter.
`exit`	Terminates an agent or a procedure.
`if (g1) {...}` `elseif (g2) {...}` `... else {...}`	Conditional statement.
`in p x`	Collects a value via port `p`.
`loop (g) {...}` `loop (every ms) {...}`	Loop statements.
`out p x`	Sends a value via port `p`.
`proc (g) p {...}`	Defines a procedure for port `p`.
`select {` `alt (g1) {...}` `alt (g2) {...}` `... }`	Selects one of alternative choices.
`sti`	Turns on the interrupts handlers.

2. Communication diagrams

Graphical items used in a communication diagram design are shown in Fig. 1. Agents in Alvis are divided into three groups: *active, passive* and *hierarchical ones. Active agents*

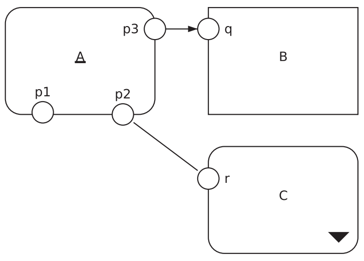

Fig. 1. Elements of Alvis communication diagrams
Rys. 1. Elementy diagramów komunikacji

(agent A) perform some activities and are similar to tasks in the Ada programming language [5]. *Passive agents* (agent B) do not perform any individual activity, and are similar to protected objects (shared variables). Hierarchical agents (agent C) represent submodels (modules). An agent can communicate with other agents through *ports*. Ports are drawn as circles placed at the edges of the corresponding rounded box or rectangle. A *communication channel* is defined explicitly between two agents and connects two ports. Communication channels are drawn as lines (or broken lines). An arrowhead points out the input port for the particular connection (connection $(A.p3, B.q)$). Communication channels without arrowheads represent pairs of connections with opposite directions (connection between ports $A.p2$ and $C.r$).

3. Model verification

From the Alvis point of view, a system is seen as a set of agents that usually run concurrently, communicate one with another, compete for shared resources etc [1]. A state of an Alvis model is represented as a sequence of states of its agents. A state of an agent X is four-tuple $S(X) = (am(X), pc(X), ci(X), pv(X))$, where $am(X)$, $pc(X)$, $ci(X)$ and $pv(X)$ denote mode, program counter, context information list and parameters values of the agent X respectively. In other words, to describe an agent state we provide its mode (e.g. agent is *running* or *waiting*), the index of the current statement, some context information (e.g. the name of called procedure) and current values of its parameters.

Behaviour of an Alvis model is considered at the level of detail of single steps (single instructions), but agents can perform its steps concurrently. States of an Alvis model and transitions among them are represented using a labelled transition system (LST graph for short [6]). Such graphs also provide a formal semantic for Alvis models and make a formal verification possible. For example, the CADP toolbox [3] and model checking techniques can be used to check whether a given model satisfies its requirements. Moreover, CADP offers a wide set of functionalities, ranging from step-by-step simulation to massively parallel model-checking.

4. Alvis Toolkit

The Alvis language is still under development. Nowadays, we focus on developing a computer software supporting the language called *Alvis Toolkit*. The current version of *Alvis Editor* – a tool to the design of Alvis models can be downloaded from the project website (http://fm.ia.agh.edu.pl). *Alvis Editor* provides a full-featured graphical editor for the design of hierarchical communication diagrams and a code editor for the implementation of the code layer. The second part of the toolkit is *Alvis Translator* used to generate an LTS graph for an Alvis model automatically. The translator builds a Java representation of a model, generates LTS graphs for individual agents and finally joins them into the final LTS graph.

5. Future plans

Our future plans focus on developing computer tools for Alvis and system layers for the models. The latter represent the hardware environment the developed embedded system is included into. For example, a designer can choose a multiprocessor environment with an unlimited number of microprocessors that makes Alvis a formal language for modelling concurrent systems. On the other hand, a single microprocessor environment can be chosen and agents will compete for the processor. The *Alvis Toolkit* is expected to include (finally): a translator from Alvis to Haskell language, a graphical simulator and *Alvis Components Library* with predefined componets to be used in models.

Bibliography

1. Szpyrka M., Matyasik P., Mrówka R.: *Alvis – modelling language for concurrent systems*, [in:] Bouvry P., Gonzalez-Velez H., Kołodziej J. (eds.): *Intelligent Decision Systems in Large-Scale Distributed Environments*, Volume 362 of SCI, Springer-Verlag (2011), 315–342.
2. Milner R.: *Communication and Concurrency*, Prentice-Hall (1989).
3. Garavel H., Lang F., Mateescu R., Serwe W.: CADP 2006: *A toolbox for the construction and analysis of distributed processes*, [in:] Computer Aided Verification (CAV'2007), Volume 4590 of LNCS, Berlin, Germany, Springer (2007), 158–163.
4. O'Sullivan B., Goerzen J., Stewart D.: *Real World Haskell*, O'Reilly Media, Sebastopol, CA, USA (2008).
5. Barnes J.: *Programming in Ada 2005*, Addison Wesley (2006).
6. Szpyrka M., Kotulski L.: *Snapshot reachability graphs for alvis models*, [in:] König A., Dengel A., Hinkelmann K., Kise K., Howlett R., Jain L. (eds.): Knowledge-Based and Intelligent Information and Engineering Systems – KES 2011, Volume 6881 of LNCS, Springer-Verlag, Berlin, Heidelberg (2011), 190–199. ■

Modelowanie systemów współbieżnych w języku Alvis

Streszczenie: Alvis jest nowym językiem modelowania przeznaczonym do rozwijania systemów współbieżnych, zwłaszcza systemów wbudowanych. Język jest rozwijany w Katedrze Automatyki AGH w ramach projektu o tej samej nazwie. Język Alvis łączy w sobie cechy języków programowania wysokiego poziomu z hierarchicznym językiem modelowania połączeń między agentami. Ponadto umożliwia on formalną weryfikację systemu wbudowanego bazującego na grafie LTS, stanowiącego formalną reprezentację przestrzeni stanów modelu. Artykuł zawiera przegląd podstawowych informacji na temat języka i projektu.

Słowa kluczowe: Alvis, systemy wbudowane, systemy współbieżne, modelowanie i weryfikacja

Marcin Szpyrka, PhD, Prof. of AGH

Prof. Marcin Szpyrka holds a position of associate professor in AGH UST in Krakow, Poland, Department of Automatics. He has a MSc in Mathematics and PhD and DSc (habilitation) in Computer Science. He is the author of over 70 publications, from the domains of formal methods, software engineering and knowledge engineering. Among other things, he is author of 3 books on Petri nets. His fields of interest also include theory of concurrency and functional programming. He is currently leader of the Alvis project.

e-mail: mszpyrka@agh.edu.pl