FXML/Jahuel: A Formal Framework for Software Synthesis

Sergio Yovine
VERIMAG/CRNS
Sergio.Yovine@imag.fr
http://www-verimag.imag.fr/~yovine

1 Introduction

The high demand of performance driven by today's mass-market real-time embedded applications, such us HDTV and video streaming, is pushing industry to make use of multi-core and multi-processor hardware. Moreover, many different customizations of the same application are produced with significant variations in their functional and non-functional characteristics.

The intrinsic complexity of programming concurrent real-time software for parallel hardware, adds on the difficulty to deal with quantitative requirements about timing, power consumption, etc., to make design, deployment, and customization extremely difficult. To overcome this problem, there is a need for IDE for embedded-software product lines based on appropriate formal specification languages, and supported by tools for producing executable code which is guaranteed to meet non-functional requirements. For this, two issues must be addressed: (2) mapping software concurrency onto hardware physical parallelism, and (2) ensuring software-level timing requirements with architecture-level resources.

For this reason, in 2004, VERIMAG with the support of STMicroelectronics, started the design and development of a language, called FXML, and its associated compilation chain, called Jahuel, for coping with concurrency and real-time properties from high-level modeling all the way down to code.

2 Description

Basic FXML is an algebraic language which provides a minimal set of platform-independent primitives for programming concurrent real-time software at a high-level of abstraction. This basic language provides both semantic and syntactic support for expressing application software behavior on an abstract execution model.

FXML can be regarded as a formal coordination language for managing dependencies and interactions between activities. The main difference with other coordination languages is that FXML can express rich control and data relationships, and it can be extended with more concrete synchronization, communication, and scheduling mechanisms.

Moreover, FXML has been designed to be used both as a programming language to express software structure, functionality, requirements and constraints, and as a compiler's intermediate representation which can be syntactically manipulated to perform program analyses and transformations.

The code-generation approach implemented by FXML's compilation-chain, Jahuel, relies on the existence of an abstract model of the target execution platform onto which an FXML program is to be translated to. Jahuel performs successive syntactic transformations whose correction is proved formally inside the FXML's semantic world. The underlying assumption is that the concrete platform is indeed an implementation of the model, and that this relationship can be proved by some other means such as theorem proving or model checking.

Jahuel is implemented in Java, using the Java Architecture for XML Binding (JAXB) API, to manipulate XML documents. FXML and its extensions are defined by XML schemes. Using JAXB, each language is bound to a Java class which provides the appropriate data representation and manipulation methods. Transformations are implemented on top of these Java classes.

Currently, Jahuel provides some general transformations which can be customized for different execution and simulation platforms. We have instantiated them to generate code for, e.g., Java, C with pthreads, C with sockets, etc. Jahuel also generates BIP models which enables component-based formal verification.

In summary, FXML and its code-generation tool-suite Jahuel provide an extensible and customizable IDE for software product lines oriented towards generating code for multiple platforms via domain-specific semantics-preserving syntactic transformations.

3 Conclusions and Future

Ongoing work includes applying FXML and Jahuel in industrial applications, strengthening the integration into an end-to-end industrial design flow, and generating code for dynamic schedulers and multi-core platforms.

A first attempt of integration in an industrial IDE has been carried out in collaboration with STMicroelectronics, in the context of the FlexCC2 compilation technology. We are currently evaluating this prototype on an industrial test-bed.

We are also studying the use of FXML a semantic framework for other languages. In this case, FXML programs are generated by a compiler. More recently, we have proposed an FXML-based semantics of a subset of the language StreamIt. We are currently working on providing semantics to the full language.

4 Links and References

URL: http://www-verimag.imag.fr/~yovine/jahuel

1. S. Yovine, I. Assayad, F.-X. Defaut, M. Zanconi, A. Basu. A formal approach to derivation of concurrent implementations in software product lines. Process Algebra for Parallel and Distributed Processing: Algebraic Languages in Specification-Based Software Development, M. Alexander and B. Gardner, Eds., Computational Science Series, Chapman and Hall/CRC Press, Taylor and Francis Group LLC, 2008.

2. A. Basu, S. Yovine, M. Zanconi. An approach to derivation of component-based implementations from data-oriented specifications. APGES 2007, Oct. 4th 2007, Salzburg, Austria.

3. I. Assayad, V. Bertin, F-X. Defaut, Ph. Gerner, O. Quevreux, S. Yovine. Jahuel: A formal framework for software synthesis. In Proceedings of ICFEM 2005 Seventh International Conference on Formal Engineering Methods". 1-4 November 2005, Manchester, UK. LNCS 3785, Pages: 204-218. Springer, 2005.

4. I. Assayad, Ph. Gerner, S. Yovine, V. Bertin. Modelling, Analysis and Implementation of an On-line Video Encoder. In Proceedings of "The First International Conference on Distributed Frameworks for Multimedia Applications (DFMA'2005)". Besancon, France, February 6-9, 2005. Pages: 295 - 302. IEEE Computer Society. 2005.