Automatic Generation of DistAlgo Programs from Event-B Models

Alexis Grall^{1,2}(✉)

¹ LORIA UMR 7503, Vandœuvre-lès-Nancy, France
alexis.grall@loria.fr

² Université de Lorraine, Vandœuvre-lès-Nancy, France

1 Motivations

The development of distributed algorithms offers challenges in verifying that they meet their specifications. The correct-by-construction approach consists in developing a model of the algorithm before transforming this model into a program. This transformation can introduce errors that were not present in the model. Our objective is to develop an automatic transformation of distributed algorithm Event-B [2] models into DistAlgo [7] programs. The Event-B language combines refinement techniques and state based modelling and is adapted to the verification of distributed systems [3, 12]. The DistAlgo language is a high-level programming language for distributed algorithms. Its high-levelness makes DistAlgo closer to the mathematical notations of Event-B and improves the clarity of DistAlgo programs. A verified automatic transformation ensures that the properties proved in the model still hold in the program and facilitates the developing process.

2 Related Works

Code generation from Event-B models has been a subject of interest in the B community. The B0 [11] language defines constraints on classical B for code generation and an equivalence between B types and usual programming types such as arrays, integers *etc.* B0 can be translated to Ada, C, C++ using the AtelierB [4] tools. In Event-B, several plugins have been developed for the Rodin [10] software. The EB2ALL [8] framework provides a list of transformations of Event-B models into classical programming languages (C, C++, Java, ...) and this work can be considered as adding a new target programming language but with the target of a distributed program. Automatic generation of distributed programs was proposed in ViSiDiA [1, 9] together with Event-B with the plugin B2VISIDIA [12] relating the local Event-B model and a ViSiDiA program. A Tasking Event-B [6] plugin for Rodin extends the Event-B language to provide features for specifying concurrent multi-tasking systems. The plugin enables the decomposition of a model into several machines performing tasks and provides a tool support for translating a tasking specification into Ada code. However, a global state is preferable for verifying global properties on distributed algorithms.

Fig. 1. The global methodology for correct-by-construction distributed algorithms.

3 General Approach

The general methodology, presented in Fig. 1, starts by stating the *problem* to solve by listing the requirements attached to the problem. One has then to specify the abstract EVENT-B machine AM translating the main requirements for the given problem. The process is progressing by a list of formal EVENT-B refined machines leading to a final concrete EVENT-B machine and context, CM and CM-CONTEXT. Finally, the translations of these final context and machine into DistAlgo components and programs are generated in two main steps: the automatic transformation of CM and CM-CONTEXT into a DistAlgo program and the possible tuning of the obtained DistAlgo components using some auxiliary configurations (*e.g.* the number of processes).

The final pair of machine and context, CM and CM-CONTEXT are supposed to be specified in a language LB (Local Event-B) which restricts the Event-B language to models of distributed algorithms that are local, which means that the constants and variables are local to processes (they are associated to processes.) and that events only use local variables and constants of one process. Transformations of this LB model into a DistAlgo program are based on the extraction of information concerning the network and the process classes from the context CONTEXT-CM, and on the analysis of the localization of the different variables and events of the machine CM. Constants whose values are not defined in the context are instantiated during the configuration phase.

4 Development of the Transformation

The LB language [5] defines a distributed algorithm in Event-B as a collection of processes with a specific local algorithm for each process. We derive the general architecture of the distributed algorithm from the context CONTEXT-CM which specifies the topology of the network, the processes, the local constants of the processes and the communication operations. We derive the local algorithms from the machine CM which specifies the local variables of the processes and the variable specifying the state of the channels. These variables define the global state of the distributed algorithm while the local state of a process is defined by the values of its local variables and the states of the channels involving this process. The transition relation between the local states of a process are then defined by the local events (of this process) in the machine CM. An additional variable *pc* in the machine helps specifying the flow of the algorithm and can be used for constructing control flow graphs describing the local algorithms.

The transformation rules add constraints on LB that are specific to DistAlgo and that ensure that the expressions used in the model can be translated into DistAlgo by specifying a correspondence between Event-B and DistAlgo types. The transformation into DistAlgo generates programs based on the local control flow graphs, therefore giving a tool to help understanding the program.

Refined models of distributed algorithms have been manually and successfully transformed into DistAlgo programs following the given transformation rules. These models describe a simple communication between two processes, a simple communication algorithm between processes in a star network and the election in a connected acyclic network.

We need to ensure that any safety property that was proved with invariants in an Event-B model still holds in the DistAlgo program obtained from the transformation of the model, *i.e.* the soundness of the transformation. We have already shown the soundness of the translation for an earlier version of LB, *i.e.* a program obtained by the transformation refines the machine it is obtained from. We have thus, that for any possible execution of the program, an equivalent behaviour can be observed for the model machine. However, the completeness of the transformation is not guaranteed when no restrictions are imposed on the corresponding programs.

5 Future Work

The implementation of the transformation as a Rodin plugin is currently a work in progress with so far encouraging results on simple cases. The proof of the soundness of the transformation remains to be completed and modified with respect to the current version of the transformation. Other future work includes the extension of LB to enable more possibilities in modelling, such as enabling the broadcast of messages, the use of user defined functions/libraries in the resulting programs or the addition of configurable timeout delays. Additional case studies of parallel computing are to be refined into LB and transformed. These case studies will be an occasion to compare the performance of the automatically generated program with an hand written version of this program. Long term future work includes the derivation of other transformations into other programming languages from the transformation to DistAlgo.

References

1. Abdou, W., Abdallah, N.O., Mosbah, M.: Visidia: A java framework for designing, simulating, and visualizing distributed algorithms. In: 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications, pp. 43–46. IEEE (2014)
2. Abrial, J.R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press, Cambridge (2010)
3. Abrial, J.R., Cansell, D., Méry, D.: A mechanically proved and incremental development of IEEE 1394 tree identify protocol. *Formal Aspects Comput.* **14**(3), 215–227 (2003)
4. Atelier, B.: The Industrial Tool to Efficiently Deploy the B Method. <http://www.atelierb.eu/en/>
5. Cirstea, H., Grall, A., Méry, D.: Generating Distributed Programs from Event-B Models. Technical report, Loria & Inria Grand Est, March 2020. <https://hal.inria.fr/hal-02496623>
6. Edmunds, A., Butler, M.: Tasking Event-B: an extension to Event-B for generating concurrent code. In: PLACES 2011, February 2011
7. Liu, Y.A., Stoller, S.D., Lin, B., Gorbovitski, M.: From clarity to efficiency for distributed algorithms. In: ACM SIGPLAN Notices, vol. 47, pp. 395–410. ACM (2012)
8. Méry, D., Singh, N.K.: EB2C : A Tool for Event-B to C Conversion Support (2011–2019). <http://eb2all.loria.fr>
9. Mosbah, M.: VISIDIA (2009). <http://visidia.labri.fr>
10. Project RODIN: Rigorous open development environment for complex systems (2004). <http://rodin-b-sharp.sourceforge.net/>. 2004–2007
11. Tatibouët, B., Requet, A., Voisinnet, J.-C., Hammad, A.: Java card code generation from B specifications. In: Dong, J.S., Woodcock, J. (eds.) ICFEM 2003. LNCS, vol. 2885, pp. 306–318. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39893-6_18
12. Tounsi, M., Mosbah, M., Méry, D.: From event-b specifications to programs for distributed algorithms. *IJAACS* **9**(3/4), 223–242 (2016). <https://doi.org/10.1504/IJAACS.2016.079623>