Formal Verification of PLC Programs Using the B Method*

Haniel Barbosa and David Déharbe

Departamento de Informática e Matemática Aplicada, UFRN, Brazil
hanielbbarbosa@gmail.com,
deharbe@dimap.ufrn.br

Abstract. In this paper we propose an approach to verify PLC programs, a common platform to control systems in the industry. Programs written in the languages of the IEC 61131-3 standard are automatically translated to B machines and are then amenable to formal analysis of safety constraints and general structural properties of the application. This approach thus integrates formal methods into existing industrial processes.

Keywords: B method, PLC, safety critical systems, formal methods.

1 Introduction

In many industries, such as mass transport and energy, it is very common to use PLCs in control applications. Those applications are mostly programmed according to IEC 61131-3 [1], an international standard that specifies the five standard PLC programming languages, namely: LD (Ladder Diagram) and FBD (Function Block Diagram), graphical languages; IL (Instruction List) and ST (Structured Text), textual languages; and SFC (Sequential Function Chart), that shows the structure and internal organization of a PLC. It is not rare that a variation of such languages is employed too.

As the complexity of the applications increases, and as various are safety critical, it is important to ensure their reliability. Formal methods are a mean to fulfill this requirement, as testing and simulation (the *de-facto* method in many branches) can left flaws undiscovered. However, it is difficult to integrate formal methods in the industrial process since most control engineers are not familiarized with formal verification.

Some recent works have been trying to integrate formal methods and PLC verification, using different approaches. In [7], the authors created a new language combining ST and Linear Temporal Logic, ST-LTL, to ease the use of formal verification by control engineers. [6] presents a method to verify applications using Safety Function Blocks with timed-automata through model-checking and

* Project supported by ANP. CNPq grants 560014/2010-4 and 573964/2008-4 (National Institute of Science and Technology for Software Engineering—INES, www.ines.org.br).

J. Derrick et al. (Eds.): ABZ 2012, LNCS 7316, pp. 353–356, 2012.

simulation. A model-driven engineering approach is used in [5] to generate models in a FIACRE language from LD programs. To this date, these approaches are concerned only with parts of the IEC 61131-3 standard.

Our approach already handles two of the five languages of the standard, namely SFC and ST, and we are working to extend it to be fully compliant. To do so, we use the PLCopen [3] standard, which provides an interface representing all the IEC 61131-3 languages in an XML-based format. Another goal of our approach is to be capable of verifying legacy programs in numerous different PLCs. We have built an intermediary model based in the PLCopen interface that is loaded from the PLC programs and then is used to *automatically* generate a B model.

B [2] is a formal method that can be used to specify systems and through proof obligations demonstrate its correctness according to the specification, avoiding state-explosion problem. It is practical and competitive to develop safety-critical systems, with the correct methodology and tools. Using the B method we can verify safety constraints through the proof obligations and also to check structural issues, such as *deadlock freedom*, using animation tools such as ProB [4]. Thus, we increase the confidence in the PLC applications and facilitate the use of formal methods in the industry.

Next section presents details on the different phases of our method as well as an example. In the end we have some discussions and future work.

2 The Method

The method we are proposing consists of three main phases:

1. translate the information in the PLC programs into an intermediary model (from now on called "PLC model"), either from a PLC program or from an XML file in the PLCopen standard;
2. generate from it a B model that makes possible to check the structural and safety properties of the project;
3. and at last complete the formal model with these safety properties, derived from the project requirements (manually, for now).

2.1 Towards the PLC Model

The PLC model may be generated either directly from an XML in the PLCopen standard or from the programs in some hybrid language, based on the IEC 61131-3 standard. Such languages are common as adaptations to specific domain PLCs may be necessary.

We projected a compiler to analyze the programs; it deals with the elements of the standard languages and may be customized to any existing differences, to accommodate any new language. This way we can deal with legacy programs that are not strictly standard compliant. To deal with XML, we use a reader module to load the PLC model along with the uncustomized compiler.

Once the PLC model is constructed we are able to work independently from the PLC programs or the PLCopen to generate the B specification.

2.2 Generation of the B Model

A good architecture is essencial to generate a good model, as well as to define which information from which language will be responsible for which elements of the B model, since it is common to have PLCs using more than one language. As so far we are working only with SFC and ST, our architecture can handle just the elements of these two languages. When we deal with the elements of the other languages the architecture will be adapted to include them.

The architecture of the model is depicted in figure 1. This model represents a PLC that process signals required by a control application, having them as *inputs*. The PLC *outputs* are treated as local variables; it is no loss of generality to deal with them like that since we are dealing with the PLCs only as independent components. The safety requirements will concern mostly these *outputs*.

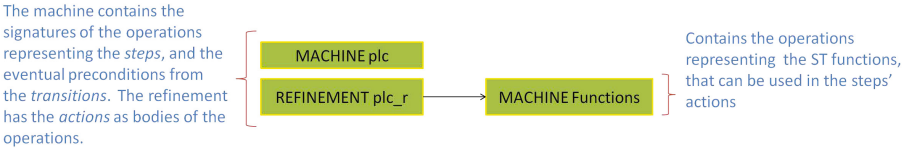

Fig. 1. Shows the architecture of the B model generated by a SFC + ST PLC

For the PLC component, the **operations** are derived from the SFC steps, as whether or not they have **preconditions** is based in the SFC transitions. The body of these **operations** and the content of their **preconditions** are the result of the translation of ST statements. The **operations** of the *Functions* **machine** are also constructed with the translated ST statements.

Figure 2 shows a little example of the generation of the B model. Due to space limitations, we do not present the whole process.

The next step is to add safety requirements. Since the PLC programs do not represent such constraints explicitly, they are manually extracted from the project requirements and inserted into the model as **invariants** of the

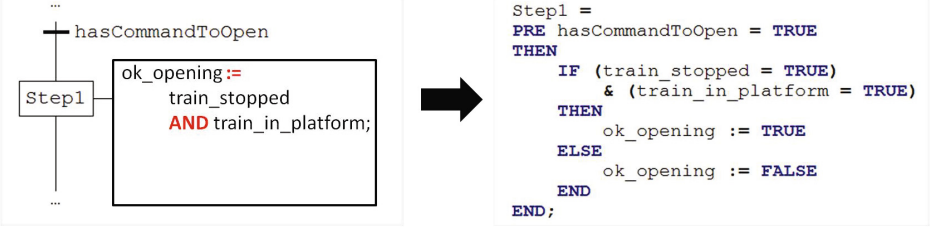

Fig. 2. Example shows a SFC step, its transition and its action (in ST) as base for the generation of a B **operation**

refinement, conditions that must always hold as the PLC actions are performed. For example, the requirement "a train must be stopped and in the platform to open its doors" is inserted as the invariant: $(ok\_opening = \text{TRUE}) \Rightarrow ((train\_stopped = \text{TRUE}) \ \& \ (train\_in\_platform = \text{TRUE}))$. Tools like AtelierB can perform automatic verification of their consistency and point out where lies any problem, guiding its treatment.

The formal model can also be evaluated with an animation tool like ProB, making possible to *model check* the model to verify structural properties (*deadlock*, *liveness*, LTL conditions...).

3 Discussions and Future Work

We have overviewed a method to carry out formal verification of the languages of the IEC 61131-3 standard for PLC programming through the automatic generation of a B specification. There is still much work to be done, but the results so far are quite satisfactory.

Future work lies most in expanding the generation of the B model to the other language. The safety constraints are still manually derived from the requirements, but we plan to automatize this process. We are about to start a case study with the company ClearSy, strongly involved with the B method and safety critical systems engineering, in a real project in the railway field to execute problem diagnosis in high speed trains.

References

1. IEC: IEC 61131-3 - Programmable controllers. International Electrotechnical Comission Standards (2003)
2. Abrial, Jr.: The B-book: assigning programs to meanings. Cambridge University Press, Cambridge (2005)
3. PLCopen: XML Formats for IEC 61131-3. PLCopen Technical Committee 6 (2009)
4. Leuschel, M., Butler, M.: ProB: A Model Checker for B. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) FME 2003. LNCS, vol. 2805, pp. 855–874. Springer, Heidelberg (2003)
5. Farines, J., de Queiroz, M.H., da Rocha, V.G., Carpes, A.A.M., Vernadat, F., Crégut, X.: A model-driven engineering approach to formal verification of PLC programs. In: IEEE EFTA (2011)
6. Ljungkrantz, O., Åkesson, K., Fabian, M., Yuan, C.: A Formal Specification language for PLC-based Control Logic. In: Proc. of 8th IEEE International Conference on Industrial Informatics, pp. 1067–1072 (2010)
7. Soliman, D., Frey, G.: Verification and Validation of Safety Applications based on PLcopen Safety Function Blocks using Timed Automata in Uppaal. In: Proceedings of the Second IDAC Workshop on Dependable Control of Discrete Systems (DCDS), pp. 39–44 (2009)