A Toolbox for the Development of Logic Controllers using Petri Nets

Georg Frey and Florian Wagner

Abstract—This paper presents a toolbox for the development of control algorithms for Programmable Logic Controllers based on Petri Nets. The Toolbox supports the complete development process including design, verification, validation, quality evaluation, simulation and PLC code generation.

I. INTRODUCTION

Usually algorithms for Programmable Logic Controllers (PLCs) are implemented using languages proposed in the standard IEC 61131-3 [1]. The standard has harmonized the way PLCs are programmed. However, the standardized languages do not force programmers to implement the algorithms in a formal way. Hence, formal verification and validation is impossible. To overcome this problem several formal design approaches have been proposed, that allow automatic translation into languages of IEC 61131-3. Many of the proposals use Petri Nets as their basis [2]. One of these proposals is SIPN [3]. SIPN is a graphical programming language based on condition event Petri Nets, enhanced by output signal assignments in places and firing conditions, based on input signals, at the transitions (cf. Fig. 1). To enable modeling of large systems, hierarchical algorithms are also possible. With SIPN, it is possible to realize formal verification as well as validation. Another very important topic for system designers is the changeability of the implemented components. With metrics described in [3], it is possible to qualify the effort to be invested for understanding the algorithm developed in SIPN.

In this paper a Toolbox for designing PLC algorithms in SIPN is described. The next section presents a development process and describes the toolbox support for the individual development steps. Section III gives conclusions and an outlook on future work.

G. Frey and F. Wagner are with the Department of Electrical and Computer Engineering at the University of Kaiserslautern, Erwin-Schrödinger-Str. 12, 67663 Kaiserslautern, Germany (e-mail: {frey; wagner}@eit.uni-kl.de).

II. THE SIPN TOOLBOX

A. Overview

The toolbox was designed as an integrated development environment (IDE) in Microsoft .NET language C#. The complete development process for PLC algorithms, as shown above, can be done in one application.

The main parts of the toolbox are the Editor (ED), the Reachability Analyzing Tool (RAT) and the PLC Implementation Generator (PIG) enhanced by interfaces to Modelica and NuSMV, as shown in Fig. 2. For information interchange with other software dealing with Petri Nets the Editor supports export to PNML (Petri Net Markup Language), a special XML format for Petri Nets [8].

The old Java-based version of the tool (called SIPN Editor [10]), which had the functionality of ED and PIG, is now replaced by the toolbox described in this paper.

B. Formal Development of PLC Algorithms

The formal development process supported by the presented toolbox covers six steps as shown in Fig. 3. The individual steps are discussed in the following sub-sections.
C. Design

The main component of the presented toolbox is the Editor (ED) allowing the graphical design of PLC algorithms with SIPN. ED was created to restrict to the ease of use of modern desktop applications for the Windows XP operating system. The editor enables designers to build an algorithm by drag & drop and assists with online syntax checking for switching conditions and output signal settings.

D. Verification with Reachability Analysis

To ensure that the designed SIPN algorithm fulfills minimum requirements of a valid control algorithm reachability analysis is used. The first step is the calculation of the reachability graph. An interface to the graph visualization software GraphViz [4] is used to get a visual representation of the graph (cf. Fig. 4).

Automatic analysis of the graph provides information about Petri net properties like conflict freeness, reversibility or liveness.

![Reachability Graph generated by RAT for the SIPN of Fig. 1 drawn by GraphViz.](image)

E. Verification and Validation using Model Checking

A disadvantage of the formal verification by reachability analysis is the so called state space explosion which may appear in large systems. One known method to avoid state space explosion is symbolic model checking. In symbolic model checking the reachable state space is not build explicitly. Instead analysis is done by means of BDDs (Binary Decision Diagrams), which provide an efficient way to analyze huge state spaces. For the use of model checking the toolbox provides an interface to the model checker NuSMV [5]. The properties to be checked by NuSMV have to be formulated in Computation Tree Logic (CTL). The CTL expressions for performing verification of basic properties are already provided by the IDE. It is possible to validate specific additional properties by adding CTL formulae inside the IDE before generating the input for NuSMV. The model checking result is presented in the IDE.

F. Test by Simulation

Validation using model checking does not take in account the plant to be controlled. Before applying the designed logic controller to the real plant tests with a model of the plant should be performed. A standard tool for modeling and simulating physical systems is Dymola by Dynasim [6], which uses the sophisticated modeling language Modelica [7] to describe the system. The IDE supports export to Dymola by creating a model of a PLC in Modelica, including interfaces for attaching input and output signals to the plant under control in the simulation environment [9].

G. Transparency Analysis

In many cases a developed PLC algorithm runs several years or even decades with only minor changes. The changes in the algorithm are necessary, since the plant my change. In this case it is important that the design of the algorithm is well readable, hence long lasting code screening for the position where the semantic change is necessary can mean a lot of wasted time and money. The IDE is able to perform transparency analysis which means, that metrics giving information about general readability and redundancy of the SIPN algorithm [3], are evaluated. The evaluation result is shown as a table and in a transparency diagram.

H. Implementation

The SIPN programming language would be useless without the possibility to transfer the designed algorithms to a real PLC. For this purpose the IDE provides an export to the programming language Instruction List (IL) of the IEC 61131-3 in a text file. The text file can be used to import the algorithm to standard software provided by the manufacturer of the PLC which provides the necessary tools for downloading it to the PLC.

III. CONCLUSIONS AND OUTLOOK

A new IDE for designing PLC algorithms in the graphical programming language SIPN has been introduced. The IDE covers the main steps of the PLC software development process (design, verification, validation, quality evaluation, simulative test and implementation). Future extensions to the toolbox will include VHDL export and reachability analysis for timed SIPN.

The toolbox is available at http://www.eit.uni-kl.de/frey.

REFERENCES