SysVeritas: A Framework for Verifying IOPT Nets and Execution Semantics within Embedded Systems Design

Paulo Barbosa, João Paulo Barros, Franklin Ramalho, Luís Gomes, Jorge Figueiredo, Filipe Moutinho, Anikó Costa, André Aranha

SysVeritas: A Framework for Verifying IOPT Nets and Execution Semantics within Embedded Systems Design

Paulo Barbosa[1], João Paulo Barros[2,4], Franklin Ramalho[1], Luís Gomes[3,4], Jorge Figueiredo[1], Filipe Moutinho[3,4], Anikó Costa[3,4], and André Aranha[1]

[1] Universidade Federal de Campina Grande, Campina Grande, Brazil
{paulo, franklin, abrantes, andre}@dsc.ufcg.edu.br
[2] Instituto Politécnico de Beja, Escola Superior de Tecnologia e Gestão, Portugal
[3] Universidade Nova de Lisboa, Lisboa, Portugal
{lugo, fcm, jpb, akc}@uninova.pt
[4] UNINOVA, Portugal

Abstract. We present a rewriting logic based technique for defining the formal executable semantics of a non-autonomous Petri net class, named Input-Output Place/Transition nets (IOPT nets), designed for model-based embedded system's development, according to the MDA initiative. For this purpose, we provide model-to-model transformations from ecore IOPT models to a rewriting logic specification in Maude. The transformations are defined as semantic mappings based on the respective metamodels: the IOPT metamodel and the Maude metamodel. Also, we define model to-text transformations for the generation of the model execution code in the rewriting logic framework. Hence, we present a translational semantics composed by two components: (i) the denotational one, considering as semantic domains the operations, equations, and properties that specify the Petri net structure, signals, and events according to the commutative monoid view; and (ii) the operational one, that changes the interleaving semantics of Maude using rewriting rules specified at the Maude metalevel to provide a maximal step semantics for transitions with arcs, test arcs, and priorities. Additionally, this work gives architectural advices in order to compose new semantics specifications by simple component substitution. Due to its simulation and verification capabilities for control systems, the presented work was applied to a domotic project that intends to save energy in residential buildings.

Keywords: Embedded Systems, Petri Nets, Maude, Verification.

1 Introduction

It is well accepted that models offer one of the best choices to deal with the development of complex systems [1]. In particular, models improve the communication between developers and customers. However, much more is expected from a single model. For example, in the embedded systems domain, one has to specify the system in an unambiguously way and enable sophisticated system analysis. Due to this fact, and in order to increase consistency, most of the currently

accepted model-based development techniques are based on formal models [2], these models are able to precisely represent the semantics of computation and of concurrency.

Several distinct modeling formalisms, supporting the model-based development attitude [3], have already proved their adequacy for embedded systems design. With Petri nets [4] as a system specification language we get the advantages of its strong mathematical definition and its well defined and precise semantics, enabling the support for simulation, state space generation, and model-checking techniques for verification purposes.

From another point of view, Petri nets are suitable for the definition of automatic model transformations, in order to obtain models at different levels of abstraction. For example, the FORDESIGN project [3] has obtained several interesting results by reusing the benefits from transformations involving models defined using a non-autonomous Petri nets class entitled Input-Output Place/Transition nets (IOPT nets) and the respective code generation for several languages and platforms. Currently, the MDA-Veritas initiative [5] has been proposed as one alternative in order to reuse the state of the art in model based development provided by the FORDESIGN project, shifting the focus to MDA (Model-Driven Architecture) [6] thus obtaining improvements in the verification of IOPT nets as formal models.

Concerning the aforementioned context, the expected results are highly dependent on the semantics of the IOPT models. Since these models are expected to be built as Platform Independent Models (PIMs), they should be properly transformed into Platform Specific Models (PSMs) for mapping to the corresponding code. However, this automatic generation must take into account the platforms architecture and definition. Thus, an important question arises: how to obtain the verification goals, given the number of existing platforms? Models need to make sense in the corresponding environment they are inserted. Thus, since each specific platform has its own concepts and execution semantics, models should be able to represent these characteristics.

In Section 2, we will discuss about the environmental impacts of deploying a semantic model for PSMs, avoiding the early use of electronic devices. We explore the previously defined semantics for IOPTs, briefly recapitulated in Section 3. In Section 4, an executable formal model upon which deploying environmental characteristics will be represented. Moreover, through a running example in Section 5, the analysis for static properties, synthesis of executable sequential implementations, automated distribution and dynamical behaviors following model checking techniques are presented as one of the main points of this work. The approach is fully supported by MDA standards and tools and takes advantage of the suitability of Maude, and its metalanguage capabilities, for reactive systems modeling and execution. Finally, Section 6 discusses our final view about this gap between the concepts available in a design language, in this case the IOPT nets, and those available in platforms, as well as the analysis and verification tools. We expect that designers should be able to work with the domain-specific abstractions such as signals and events, so the knowledge required to map them into a platform must be provided automatically.

2 Contributions to Sustainability

We expect that the greater the effort to tackle the question of representing platform-specific characteristics in verification models, the greater will be the obtained benefits. More specifically, the creation of formal models eases the simulation and verification at several levels, hence decreasing the necessity of dealing with hardware devices at early stages, avoiding the use of additional electronic devices that would hardly be recycled, saving energy, increasing the reliability level, and reducing costs. Hence, our approach constitutes a technological innovation that contributes to sustainability. To that end, we have identified four distinct contributions: (i) the use of more reliable development flows, due to the consolidation of a formal MDA approach, (ii) savings in costs due to hardware, energy, specialized engineer views (for the software developer), more abstract specifications, and correctness, (iii) the use of specific executable semantics specifications available in several kinds of platforms and (iv) at the practical level, the facilitation of rapid prototyping for reliable control and domotic systems.

3 IOPT Nets and MDA-Veritas

Petri nets are a well-known set of formal languages with a common graphical representation, particularly suitable for the visual modeling of concurrent systems. The class of Input Output Place Transition nets extends the class of Place/Transition nets (P/T nets) with non-autonomous constructs allowing the modeling of controllers connected to the environment through signals and events. The respective IOPT semantics that interest us was already presented elsewhere [3]. Next, we briefly present the main characteristics of this semantics.

IOPT nets add several annotations to the P/T nets nodes and net modules. More specifically, transitions can have associated input and output events, as well as a guard that is a function of the input signal values. Additionally, each conflict is resolved through the addition of a priority annotation to each transition. Places have a bound annotation specifying the maximum number of tokens in each place, which can be of major relevance when automatic code generation is considered. They also have a conditional external action on output signals. The respective condition is a function of the place marking. All signals and events are defined at the net module level. Next we briefly present the IOPT nets semantics.

The IOPT nets have *maximal step semantics*: whenever a transition is enabled, and the associated external condition is true (the input event and the input signal guard are both true), the transition is fired. An IOPT net step is maximum when no additional transition can be fired without generating an effective conflict with some transition in the chosen maximal step. Therefore, we define a IOPT net step occurrence and the respective successor marking. The net evolves through the firing of successive maximal steps. The synchronized paradigm, used in this work, also implies that the net evolution is only possible at specific instants in time named *tics*. These are defined by an external *global clock*. An execution step is the period between two tics.

3.1 System Modeling

In [7] we have proposed a formal approach named *semantic equations* to extract formal models from the syntactic constructors, *i.e.* the metamodel. There, we have approached Petri nets models involved in model transformations aiming to ensure semantics preserving properties. Here, we are interested in reusing the *semantic equations* approach but for another purpose: to extract the state space as the semantic representation for the IOPT nets formalism. The understanding of this state space in terms of algebraic structures over graphs has the potential to unify several views of semantics formalisms. This structure can be manipulated by formal methods tools employed in several sorts of analysis and satisfying some requirements that are hard to solve. As an example, several formal tools could handle the state space explosion problem, by providing an on-the-fly way of reasoning over occurrence graph structure of the Petri net model if it is seen as this algebraic structure.

As in [7], this ordinary graph has a set of nodes as a commutative monoid S+ generated by the set S of places and the empty marking as identity. The sum + of transitions represents the parallel firing of transitions, and the sequential firing of transitions is denoted by the operation. By the guarantee of the closure property, we are representing computations through simple transitions.

The concept of *semantic domain* and *semantic equations* gives us a guarantee that the metamodel presented in [8] will provide models able to satisfy the desired executability requirement.

4 From IOPT Models to Algebraic Specifications

Fig. 1 shows the main flow for the use of this solution. It is inserted in a major flow of the MDA-Veritas solution. We have two roles: (i) the modeler that starts editing the model until, throughout MDA transformations, generating the Maude code; and (ii) the verifier, that employs the Maude model in concrete syntax until the final analysis of formulae. The formulae can be proved at the object level by using the Maude LTL model-checker through the definition of predicates.

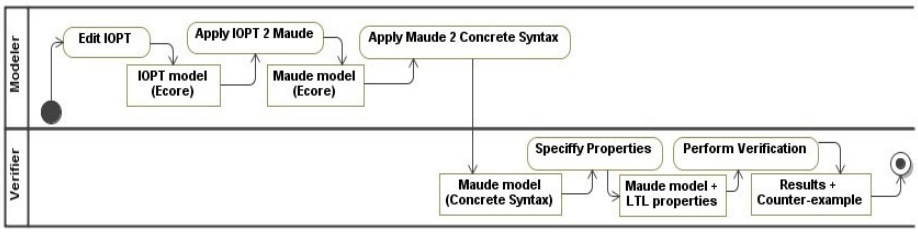

Fig. 1. Modeling and verification process

Fig. 2 presents a very simple example illustrating how to create and derive the *semantic model* for an IOPT. The model reuses the basic Petri nets graph structure. The following generated code has this graph representation plus the new features,

such as events (OFF, BRIGHT, and DARK) and signals (SW8, SW9, and SW10). However, the most important features cannot be seen syntactically, because they are in the semantic domain. Examples of this are the specific execution semantics adopted for the model, the priority decision that decides what transition will fire in a conflict situation, or the occurrence of signals and events that affect the guards of the Petri net.

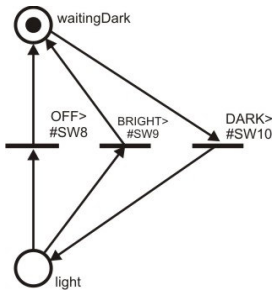

Fig. 2. Fragment of a domotics model

For example, the following Maude code represents the net of Fig. 2. It describes the existing sorts (line 2) for this algebraic representation, existing events (line 3), existing signals (line 4), and existing places (line 5). Other constructions responsible for the soundness of the definition will be discussed next. Finally, we have the representation of one transition (line 6), called turnLightOn, which has as precondition the DARK event, the SW10 signal, the waitingDark token, and produces no event (idle), no signal (noSignal), and one token light.

```
1 mod DOMOTICS-IOPT-NET-CONFIGURATION is
2 sorts Event EventSet Signal SignalSet PlaceMarking NetMarking IOPT .
3 ops OFF BRIGHT DARK : -> Event . op idle : -> Event .
4 ops SW8 SW9 SW10 : -> Signal . op noSignal : -> Signal .
5 ops waitingDark light : -> PlaceMarking . op empty : -> PlaceMarking
.
...
6 rl
[turnLghtOn]:{DARK}+[SW10]+(waitngDark)=>{idle}+[noSignal]+(light).
7 endm
```

The following code fragments represent a basic template for the translation of an IOPT model to a rewriting logic specification in Maude. We use regular grammar constructors for specifying the possible number of elements in a model. Line 1 is the declaration of the module able to represent the IOPT net. Line 2 defines the existing sorts for this specification. From line 3 up to 5 we have the declaration of the names for sorts Event, Signal, and PlaceMarking respectively, and the corresponding identity operations, *i.e.* idle, noSignal and empty. Finally, line 6 is the basic operation of combination of PlaceMarkings that follows the principles of commutative monoid Petri nets representation with the corresponding properties.

```
1 mod name-IOPT-NET-CONFIGURATION is
2 sorts Event EventSet Signal SignalSet PlaceMarking NetMarking IOPT .
3 ops (Event_names) : -> Event . op idle : -> Event .
4 ops (Signal_names)* : -> Signal . op noSignal : -> Signal .
```

```
5 ops (PlaceMarking_names)* :->PlaceMarking . op empty:-> PlaceMarking
.
6 op __:PlaceMarking PlaceMarking ->PlaceMarking [assoc comm id:
empty].
```

Thus, the Event, Signal and PlaceMarking elements are composed, in lines 7 up to 9, in EventSet, SignalSet and NetMarking respectively. The composition of these major structures is called a IOPT type at line 11.

```
7 op {(_)*} : (Event)* -> EventSet [ctor] .
8 op [(_)*] : (Signal)* -> SignalSet [ctor] .
9 op (())* : (Place)* -> NetMarking .
10 op noState : -> [IOPT] .
11 op _+_+_ : EventSet SignalSet NetMarking -> IOPT .
```

Finally, the transitions are represented as rewrite rules from one IOPT configuration (line 12) to another IOPT configuration (line 13).

```
(
12 rl [Rule_name]:{(Event_names)*}+[(Signal_names)*]+((Place_names)*)
13 => {(Event_names)*}+[(Signal_names)*]+((Place_names)*) .
)*
```

4.1 Maximal Step Semantics

In order to give a semantic representation of Petri nets having the maximal step semantics in a translational way, we need to change the Maude's original interleaving semantics. The main structure of the modules produced in this activity is depicted in Fig. 3. Starting from an IOPT_SYSTEM_CONFIGURATION, our choice relies on the fact that the Maude system contains a predefined module called CONFIGURATION for defining a denotational semantics and a META-LEVEL for redefining the operational semantics. The components from the denotational view were presented in the previous section, through the definition of an IOPT_NET_CONFIGURATION. Here, we focus on the operational view. A the META-LEVEL, we can find operators that represent terms and modules. From this level, the module also supplies efficient descent operations reducing the computations from the metalevel to the object level. Thus, operations such as *metaApply*, that matches the term with lefthand side of the rule, apply the rule at the top of the term and returns the metarepresentation of the term, can be used in order to take the application of all enabled transitions at a single step. Inheriting from the META-LEVEL, we defined a module META-PETRI-NET able to represent the main structural operations and rules from a Petri net. Finally, the MAXIMAL_STEP redefines the execution semantics of the Maude system for this domain, according to our specification. An excerpt of this module is shown in the following code fragment.

```
1 rl [maximal-step] : T:Term =>
2 maxStep(T:Term, applicableRules(T:Term, rules, module)) .
...
3 eq maxStep(T:Term,(rl X:Term => Y:Term [label(Q:Qid)] .) RS:RuleSet)
=
```

```
4 maxStep(getTerm(metaApply(module, T:Term, Q:Qid, none, 0)),
RS:RuleSet)
5 eq maxStep(T:Term, none) = T:Term .
```

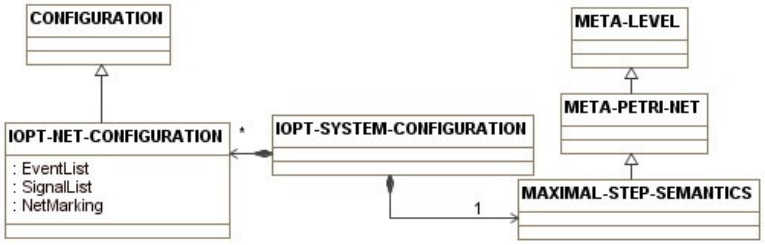

Fig. 3. The modules infrastructure

Lines 1 and 2 of this excerpt present the basic definition of the maximal-step rule for IOPT nets. A Term t is rewritten of the application of all applicable rules for the given module. Lines 3 and 4 detail the match of a given rule if applicable and the call of the built-in operation metaApply from the Maude META-LEVEL module. Line 5 represents the can in which all applicable rules were already applied, returning the current term.

5 Example

We have applied this formal specification in different scenarios. In order to illustrate it in practice, we show in Fig. 4 an application under development that supports domotic control systems. For simplification purposes, we have chosen a model with sensors that controls: (i) the arrive and leaving of strange people; (ii) the detection fire and its disabling; (iii) the alert for an unexpected situation and its stop; (iv) the detection of darkness and turning on the lights; and (v) a central enabler that establishes the priorities of these independent actuators.

From this model, we can automatically extract a *translational semantic* representation able to be employed in the Maude system. The following excerpt illustrates the single transition ***disFireAlert*** in Fig. 4 generated automatically from the previously described model.

```
1 mod DOMOTICS-IOPT-NET is
...
2 rl [disFireAlert] : {DISABLE} + [SW4] + (enabled firing) =>
3 {noEvent} + [noSignal] + (waitingForFire waitingToEnable) .
...
```

This means that the transition disFireAlert, which represents disabling the alert of fire, has dependencies in its firing from the existence of the event DISABLE and the signal SW4 (representing the action of pressing the switch of number 4 in the board that contains a PIC microcontroller) and having tokens in the places enabled and firing. As result, because of the act of consuming, no event and no signal are available in the system, the matched tokens were removed and new tokens for the

places waitingForFire and waitingToEnable were produced. This represents a Platform-Specific Model (PSM) according to the MDA view.

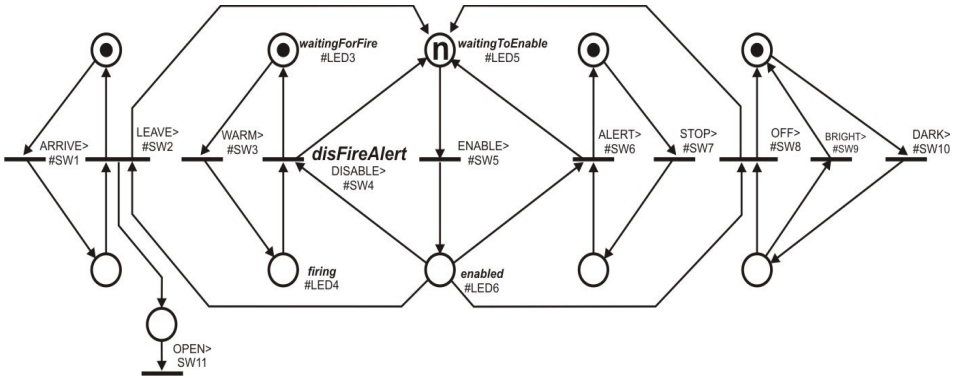

Fig. 4. Simplified domotics model

5.1 Simulation and Verification

The system can be simulated through common rewrite commands or generate the state space as an excerpt shown in Fig. 5. We developed a plugin to produce graphical visualizations of IOPT states after simulation and verification by using the GraphViz solution [9]. This represents the case where the system just allows the firing of one actuator that disables according to the priorities one enabled module from many (defined as n=1 in the place waitingToEnable) when sensors were fired concurrently. It shows that from the initial state, it is possible, through the maximal-step semantics, the activation of the events ARRIVE, WARM, ENABLE, STOP and DARK. From this state, excepting the luminosity module that is completely independent, only the ALERT event and its corresponding module, sensors and actuators are able to return to the initial state, depending from the ENABLE actuator.

The developers of the domotic model were interested in the verification of properties such as deadlock freeness, ensuring the correct application of the priorities and logical implications that given the firing of a sensor, the corresponding actuator will take the control and after solving will go back to the waiting state. These kind of primitive properties are automatically derived for the Maude LTL model-checker syntax from the initial system marking as the example that follows:

```
1 eq initial = {noEvent} + [noSignal] + (waitingForPresence
waitingForFire
2 waitingForDarkness waitingAlert waitingToEnable waitingToEnable)
...
3 search in DOMOTICS-IOPT-NET : upTerm(initial) =>! Any:Term .
```

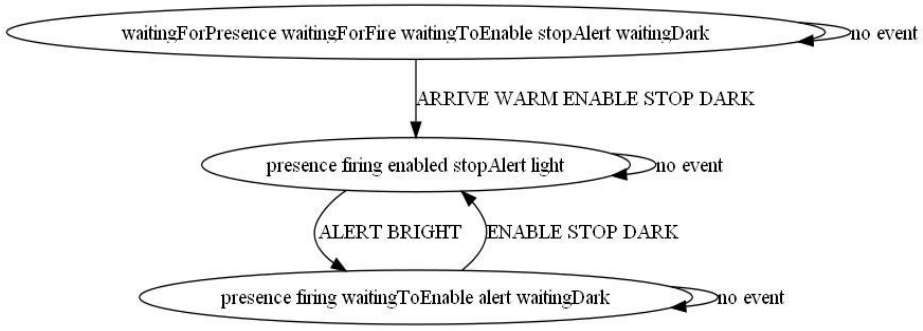

Fig. 5. Excerpt of the generated state space with the maximal step semantics

There, we have a search for a deadlocked state according to the maximal step semantics. The generated verification code starts by translating the initial marking specified in the ecore IOPT model to the lines 1-2. This initial marking specifies that there are no events and signals and the initial tokens. Finally, line 3 represents the excerpt where the search Maude command is performed by translating the initial marking to its representation at the meta-level (upTerm) and try to find any term with no successors (through the command =>!). After successive refinements during the modeling phase, having several improvements, we get a running model with no deadlocks as show the following Maude output.

```
1 No solution .
2 states: 36 rewrites: 66 in 0ms cpu (0ms real) (~ rewrites/second)
```

6 Conclusions

For the embedded system's development, our contribution comes from the fact that safety and economic concerns require high levels of assurance that the designed model will work as expected in a physical environment. In this sense, the move to the physical implementation represents a huge gap of abstraction. Formal models enable us to solve this problem in an elegant fashion. The gap is filled with an artifact that is a representation of the designed model but can also represent logically most of conditions generated by the environment. Therefore we continue in the MDA lifecycle because this technique also represents the conceptual transformations of PIMs to PSMs according to the MDA view.

The present solution still has some limitations concerning the rigorous formalization of some execution semantics regarding the checking of semantics preservation in model transformations, establishing an equivalence relation between the models. More specifically, and although this does not affect the simulation and verification purposes initially established, the state space in the *maximal-step* semantics case cannot be fully explored for all partial subsets of events generated by the environment. As future work, we intend to extend the solution for more specific platforms, producing a semantic framework that will bring several benefits for the system's designer before producing a device from a chip layout or deploying code in an embedded platform.

Acknowledgment. This work is supported by the cooperation project funded by Portuguese FCT through the project ref. 4.4.1.00-CAPES, and by Brazilian CAPES through the project ref. 236/09.

References

1. Gomes, L., Fernandes, J.M.: Behavioral Modeling for Embedded Systems and Technologies. Information Science Reference (2009).
2. Sgroi, M., Lavagno, L., Sangiovanni-Vincentelli, A.: Formal models for embedded system design. IEEE Des. Test 17 (2000) 14–27.
3. Gomes, L., Barros, J. P., Costa, A., Pais, R., Moutinho, F.: Formal Methods for Embedded Systems Co design: the FORDESIGN Project. In: ReCoSoC'05-Reconfigurable Communication-centric Systems-on Chip - Workshop Proceedings. (2005).
4. Girault, Claude, V.R.: Petri Nets for Systems Engineering. In: XV, 607 p., Hardcover. (2003).
5. Barbosa, P., Costa, A., Ramalho, F., Figueiredo, J., Gomes, L., Junior, A.: Checking Semantics Equivalence of MDA Transformations in Concurrent Systems. Journal of Universal Computer Science (J.UCS) (to appear) (2009) http://www.jucs.org/jucs.
6. OMG: Model-Driven Architecture (2008) http://www.omg.org/mda/.
7. Barbosa, P., Ramalho, F., Figueiredo, J., Costa, A., Gomes, L., Junior, A.: Semantic equations for formal models in the model-driven architecture. In: Emerging Trends in Technological Innovation. (2010).
8. "Ecore Representation for Extending PNML for Input-Output Place-Transition Nets"; Filipe Moutinho, Luis Gomes, Franklin Ramalho, Jorge Figueiredo, João Paulo Barros, Paulo Barbosa, Rui Pais, Anikó Costa; IECON'2010 - 36th Annual Conference of the IEEE Industrial Electronics Society, November 7-10, 2010; Glendale, AZ, USA.
9. Ellson, J., Gansner, E., Koutsofios, L., North, S., Woodhull, G., Description, S., Technologies, L.: Graphviz. open source graph drawing tools. In: Lecture Notes in Computer Science, Springer-Verlag (2001) 483–484.