Using Meta-modeling Graph Grammars and R-Maude to Process and Simulate LRN Models

Nardjess Dehimi *ndehimi@yahoo.fr*
Computer Science Department/MISC Laboratory
University Mentouri Constantine
Constantine, 25000, Algeria

Allaoua Chaoui *a_chaoui2001@yahoo.com*
Computer Science Department/MISC Laboratory
University Mentouri Constantine
Constantine, 25000, Algeria

Abstract

Currently, code mobility technology is one of the most attractive research areas. Numerous domains are concerned, many platforms are developed and interest applications are realized. However, the poorness of modeling languages to deal with code mobility at requirement phase has motivated researchers to suggest new formalisms. Among these, we find Labeled Reconfigurable Nets (LRN) [9], This new formalism allows explicit modeling of computational environments and processes mobility between them and It allows, in a simple and an intuitive approach, modeling mobile code paradigms (mobile agent, code on demand, remote evaluation). In this paper, we propose an approach based on the combined use of Meta-modeling and Graph Grammars to automatically generate a visual modeling tool for LRN for analysis and simulation purposes. In our approach, the UML Class diagram formalism is used to define a meta-model of LRN. The meta-modeling tool ATOM3 is used to generate a visual modeling tool according to the proposed LRN meta-model. We have also proposed a graph grammar to generate R-Maude [22] specification of the graphically specified LRN models. Then the reconfigurable rewriting logic language R-Maude is used to perform the simulation of the resulted R-Maude specification. Our approach is illustrated through examples.

Keywords: Code Mobility, Modeling Mobility, Labeled Reconfigurable Nets, Mobile Agent, Graph Transformation, R-Maude, ATOM3 Tool, Maude.

1. INTRODUCTION

Indeed, the formal tools which have been used to model and analyze classical systems are unable to deal with code mobility system properties [8]. Many studies on formal tools have attempted to extend classical tools to deal with code mobility properties. Among these studies we can mention process algebra based models (π-calculus [13], join-calculus [7], HOπ-calculus [16], for example) and state transition based models such as Petri nets. Different works tend to propose methodologies and approaches ([6], [14], [17]…), to prevent ad-hoc development for code mobility software. In effect, these approaches are mostly informal, to deal with this problem, some high level Petri Nets have been proposed. The most known are Mobile Nets (variant of colored Petri nets) [1]. To fit mobile Petri nets to specific mobile systems, various extensions have been proposed such as Elementary Object Nets [18], labeled reconfigurable nets [9], Nested Petri Nets [10] [11], HyperPetriNets [4], … In [3], the authors proposed an approach for transforming mobile UML Statechart diagrams to Nested nets models for analysis purposes. It produces highly-structured, graphical, and rigorously-analyzable models that facilitate early detection of errors like deadlock, livelock, etc … Their approach is based on graph transformation since the input and output of the transformation process are graphs. the meta-modeling tool ATOM3 is used. In this study we will deal with the transformation of LRN models into their equivalent R-Maude specification for analysis purpose.

The rest of the paper is organized as follows. In section 2, we introduce the key concepts that are dealt with in our research. In section 3 we describe our approach which consists of transforming labeled reconfigurable nets models into their equivalent R-Maude specifications. In section 4, we illustrate our approach using examples. Section 5 presents the prototype R-Maude using one of the 2 examples given in section 4. In section 6, we will round off our paper by suggesting some concluding remarks and putting forward the future perspectives for further research.

2. Background

In the following subsections we consider the main concepts and tools used in this paper and give further references for the reader.

2.1 Labeled Reconfigurable Nets

The formalism "labeled reconfigurable net" [9] is dedicated to model code mobility systems [8]. The authors proposed to model mobility in an intuitive and explicit way. Mobility of code (a process or an agent) will be directly modeled through a reconfiguration of the net. The formalism allows adding and deleting places, arcs, and transitions at run time. For more details, the reader can refer to [9].

2.2 Rewriting Logic and Maude

Rewriting logic [12] has been introduced by José Meseguer as logic for describing concurrent systems. It is implemented by several languages such as Maude [12]. The latter is a specification and programming language. It is simple, expressive and has a high-performance implementation. Maude offers three types of modules: Functional modules, System modules and Object-Oriented modules. It offers full Maude to support that; furthermore, it has its own model-checker that is used in checking system's properties. For more details the reader can refer to [12].

2.3 Graph Transformation and ATOM3

Graph grammars [15] can be used to describe graph transformation or to generate sets of valid graphs or to specify operations on them. Graph grammars are composed of production rules, each of them, having graphs in their left and right hand sides (LHS and RHS). Several tools for graph transformation have been proposed in the literature. Among these tools, we can cite ATOM3 "A Tool for Multi-formalism and Meta-Modeling" [2]. The two main tasks of ATOM3 are meta-modeling and model transformation. For more details, the reader can refer to [15].

2.4 Reconfigurable Maude

Maude [19] is a high-level language and high performance system supporting executable specifications and declarative programming in rewriting logic [20].

Maude has been extended to deal with some aspects not considered in former version. Maude [21] is a system to specify and analyze real time and hybrid systems. Mobile Maude [6] is an extension of Maude for mobile systems specification.

An encoding of Labeled Reconfigurable Nets in a Maude-based language has been proposed [22]. The inspired language is called "Reconfigurable Maude" (R-Maude). It profits from the power of Maude (as a meta-language). Maude was extended to support the translation of LRN and their simulation. R-Maude enriches Maude with new kind of rewriting rules. These rules are called Reconfigurable rules (R Rules). The semantic of these rules is similar to that of Reconfigurable transition in LRN. When a rule is executed, the R-Maude specification will be updated in different ways, this will depend on the label associated with this rule.

A specification in R-Maude is a set of Reconfigurable rewrite theories (R-theories). An R-Theory RT is a triple (Ω, E, R) as like a rewrite theory. The difference resides in the set R. R will contain two kinds of rules: standard Rules S-Rules (well known rules of Maude) and Reconfigurable rules R-Rules. An R-Rule rλ is composed of a label λ =<d, RT1, RT2, S> and a rule t□t'. In the label λ, RT1 and RT2 are two R-Theories; S is a segment of a theory, and d a specific parameter. The segment S can be a set of sorts, rules, variables, operators that can be an R-theory or not. When rλ is fired, the specification can be updated in several ways.

Updating specification means that their R-theories will be changed. This change depends on λ. In general, when rλ is fired, the segment S will move from RT1 to RT2 or the inverse. The d parameter can be used to express direction of this move. For more details the reader can refer to [22].

3. OUR APPROACH

The steps of our approach are as follows.

3.1 Meta-Modeling Of LRN (Labeled Reconfigurable Nets)

To build models of LRN formalism in AToM3, we have to define a meta-model for LRN. The meta-formalism used in our work is the UML Class Diagrams and the constraints are expressed in Python code. Since LRN consists of places, transitions, and arcs from places to transitions and from transitions to places, we have proposed to meta-model LRN two Classes to describe Places and Transitions, and two associations for arc-in and arc-out as shown in FIGURE 1. We have also specified the visual representation of each class or association. Given our meta-model, we have used AToM3 tool to generate a visual modeling environment for LRN models. FIGURE 2 shows the generated LRN tool and a dialog box to edit a transition. Each transition has two attributes (label and nom, the attribute label is defined in the case of reconfigurable transition).

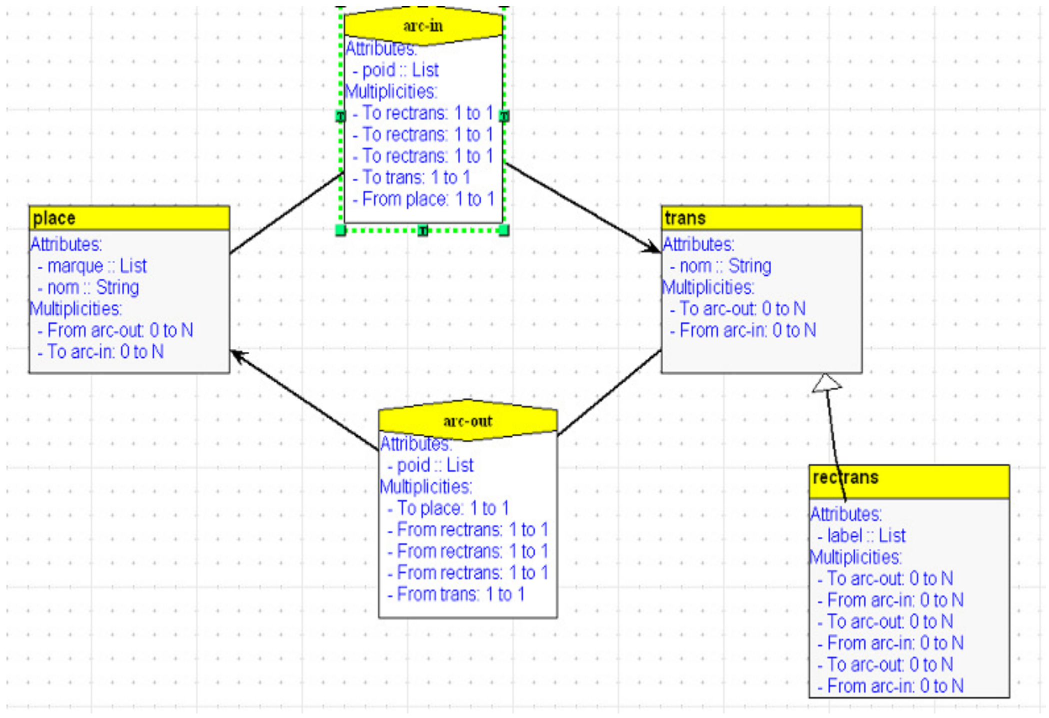


FIGURE 1: LRN Meta-Model

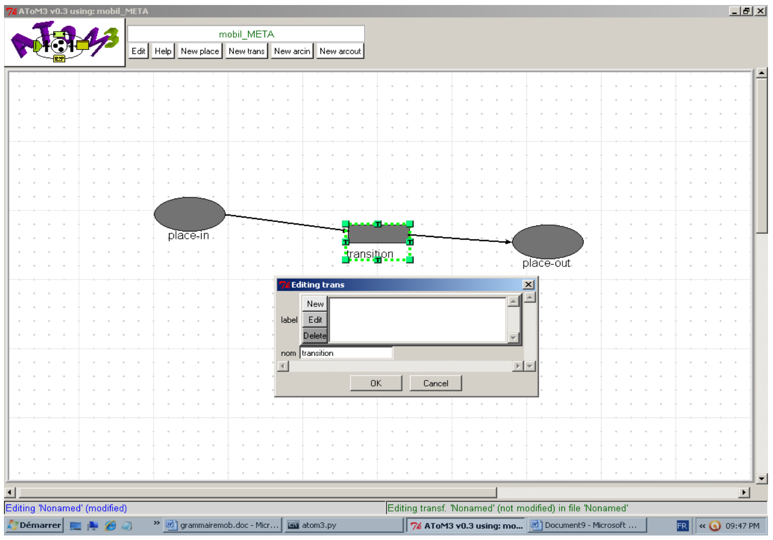


FIGURE 2: Tool bar of the Generated tool to process LRN models

3.2 The Graph Grammar Generating R-Maude Specifications from LRN Models

In order to simulate LRN models, we translate them into their equivalent representations in R-Maude syntax. In this section we show how to use the modeling environment generated in the previous section to generate R-Maude specification. We do this by defining a Graph Grammar to traverse the LRN model and generates the corresponding code in R-Maude. The graph grammar has an initial Action that is used to create the file where the code will be generated. This action decorates also all the Transition and Place elements in the model with temporary attribute according to the conditions specified in the rules. In Transition elements, we use two attributes: current and visited. The current attribute is used to identify the transition in the model whose code has to be generated, whereas the visited attribute is used to indicate whether code for the transition has been generated or not. In Place elements, two attributes are used: fromVisited and toVisited. The fromVisited attribute is used to indicate whether this place is processed as input place whereas the toVisited attribute is used to indicate if this place is processed as output place.

In our graph grammar, we have proposed seven rules (see FIGURE 3) which will be applied in ascending order by the rewriting system until no more rules are applicable. We are concerned here by code generation, so the grammar rules will not change the LRN models. These rules are described as follows:

Rule1: lefthandside (priority 1): is applied to locate a place (not previously processed) related to current transition with an input arc, and generates the corresponding R-Maude specification.
Rule2: separate (priority 2): is applied to generate R-Maude code which separates LHS and RHS of the equivalent rewriting rule.
Rule3: righthandside (priority 3): is applied to locate a place (not previously processed) which is related to current transition with an output arc, and generate the corresponding R-Maude specification.
Rule4: condition (priority 4): is applied to generate the appropriate R-Maude code depending on the condition of the transition, and decorates the transition as visited.
Rule5: InitPlace (priority 5): is applied to locate and initialize temporary attributes in places for processing the next transition.
Rule6: SelectTransition (priority 6): is applied to select an LRN transition not previously processed and generates its equivalent rewriting rule in R-Maude.

Rule7: terminate (priority 7): is applied to perform the writing of the generated R-Maude file and closes it.

The graph grammar has also a final action that erases the temporary attributes from the entities and closes the output file.

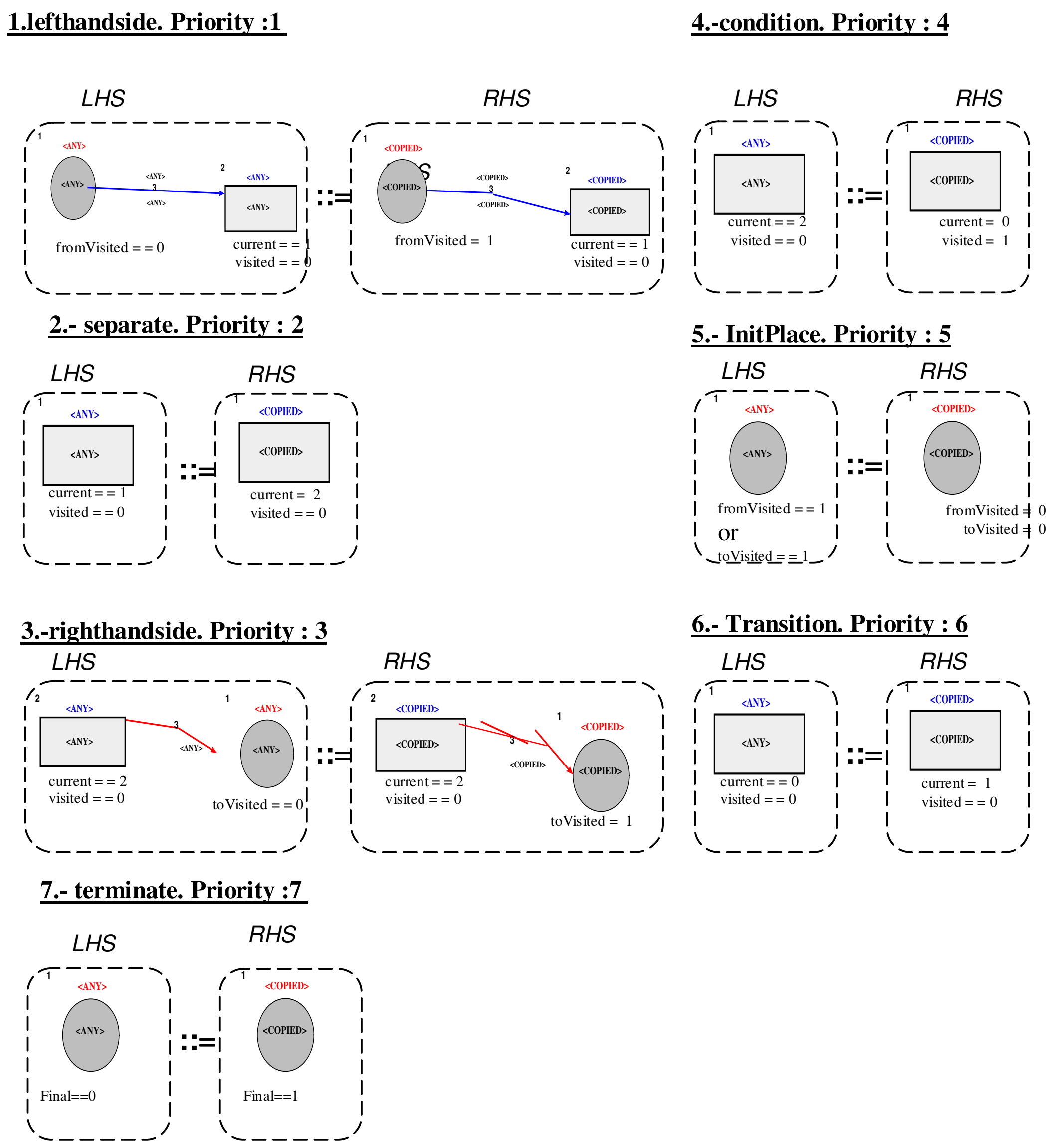


FIGURE 3: The graph grammar rules

4. Examples

4.1 Example 1: LRN With a Static Agent

This example is about a computational environment E1. E1 contains a unique static agent SA1 that execute infinite loop. SA1 requires a non-transferable resource bound by type PNR2 to execute a32. The system will be modeled as a labeled reconfigurable net LRN. FIGURE 4 presents the graphical model of this example created in our tool.

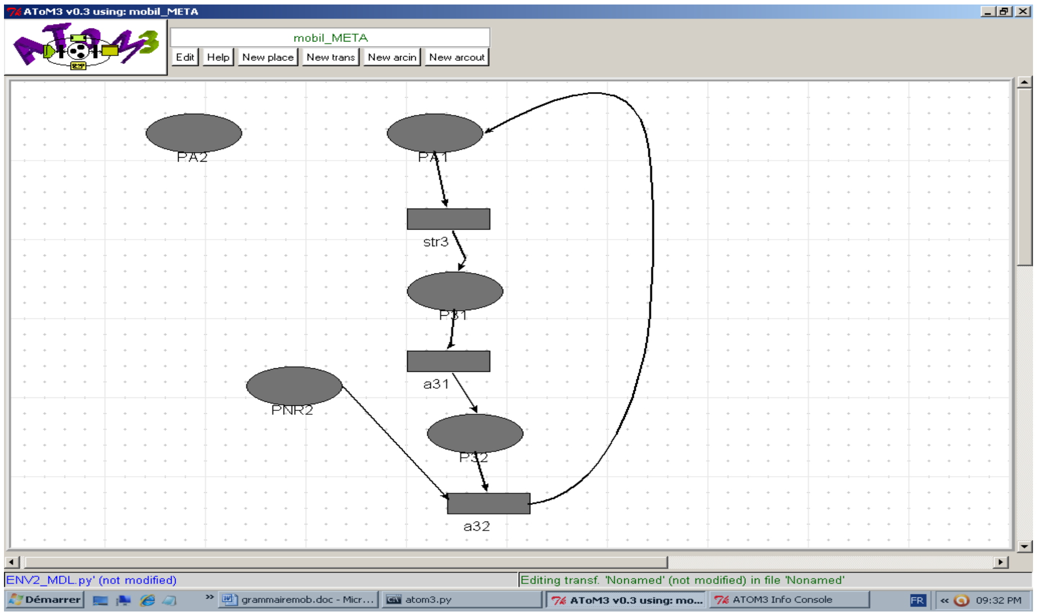
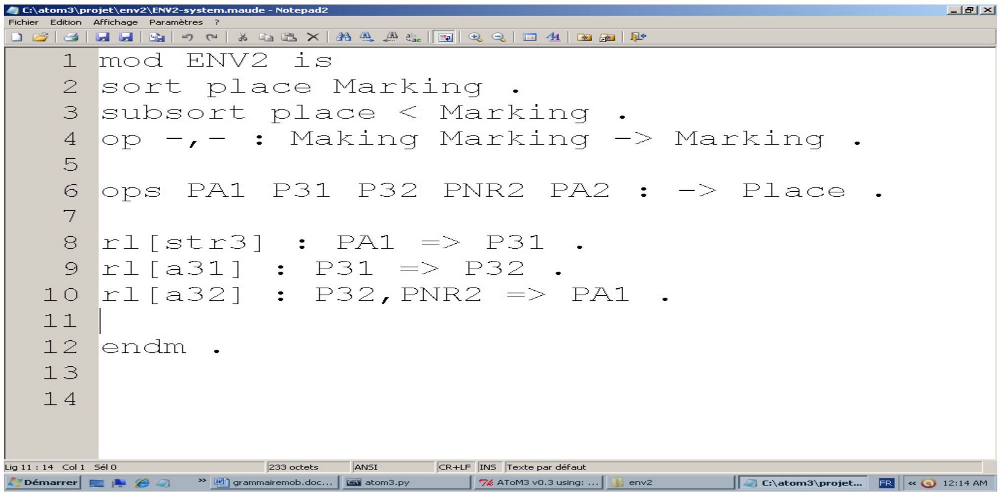


FIGURE 4: The LRN model of the example

Translating LRN Model to R-Maude Specification

This step has graphical representation of an LRN model as input. It consists of translating this graphical representation into its equivalent R-Maude specification using the graph grammar defined in the previous section. To realise this translation, the user have to execute the graph grammar previously defined.

The result of this translation given in FIGURE 5, is the file env2-system.Maude which contains the specification of the LRN model of FIGURE 4.


C:\atom3\projet\env2\ENV2-system.maude - Notepad2
Fichier Edition Affichage Paramètres ?
Lig 11 : 14 Col 1 Sél 0
233 octets
ANSI
CR+LF
INS
Texte par défaut
Démarrer
12:14 AM

```
mod ENV2 is
sort place Marking .
subsort place < Marking .
op -,- : Making Marking -> Marking .

ops PA1 P31 P32 PNR2 PA2 : -> Place .

rl[str3] : PA1 => P31 .
rl[a31] : P31 => P32 .
rl[a32] : P32,PNR2 => PA1 .

endm .
```

FIGURE 5: Generated R-Maude specification env2-system.maude

4.2 Example 2: LRN With a Mobile Agent

In this example, E1 and E2 are two computational environments. E1 contains two agents, a mobile agent MA and a static agent SA1; E2 contains a unique static agent SA2. The three agents execute infinite loops. MA executes actions {a11, a12, a13 }, SA1 executes actions {a21, a22, a23}, and SA2 executes actions {a33, a32}. To be executed, a11 require a transferable resource TR1 and a nontransferable resource bound by type PNR1 which is shared with a21. a12 and a22 share a transferable resource bound by value, and a13 and a23 share a non-transferable resource NR1. In E2, SA2 requires a non-transferable resource bound by type PNR2 to execute a32. PNR2 has the same type of PNR1. The system will be modeled as a labeled reconfigurable net LRN. LRN contains two environments E1 and E2 that model the two computational environments. In this case the unit A that models the mobile agent A will contain a reconfigure transition rt < A, E1, E2, ψ, β >; such that:

1. E1 =(RP1, GP1, U1, A1); RP1 contains at least four places that model the four resources. Let TR1, NR1, PNR1 and VTR1 be these places. GP1 contains at least a free place PA1 modeling that A can be received, and U1={A}.
2. E2=(RP2,GP2, U2, A2); RP2={PNR2}, GP2={PA2}.
3. ψr={TR1}, ψc={VTR1};
4. β={(PA2, str1), (PNR2, a11), (NR1, a13)}.

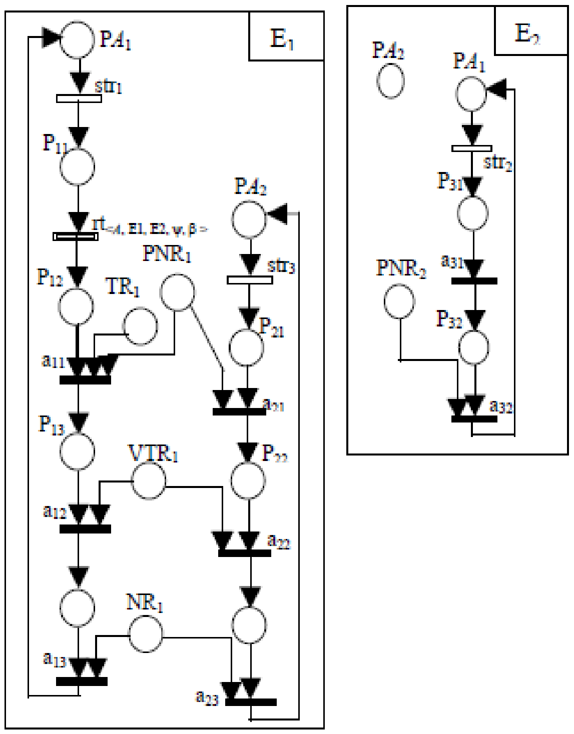


FIGURE 6: LRN MA-model before firing rt

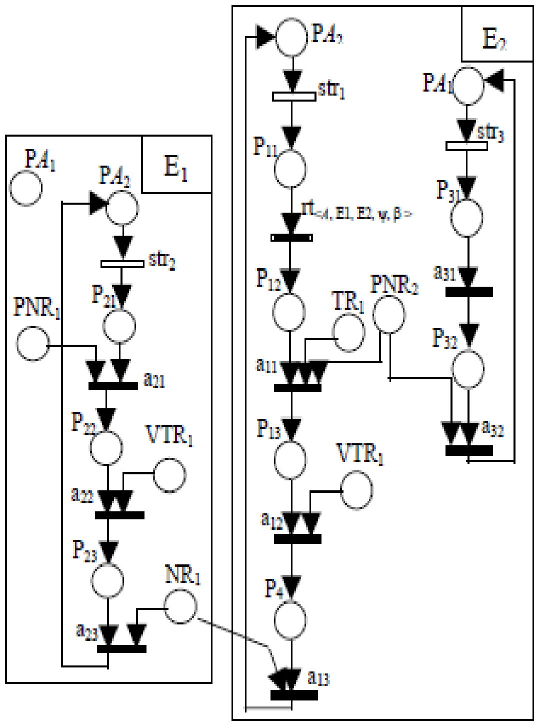
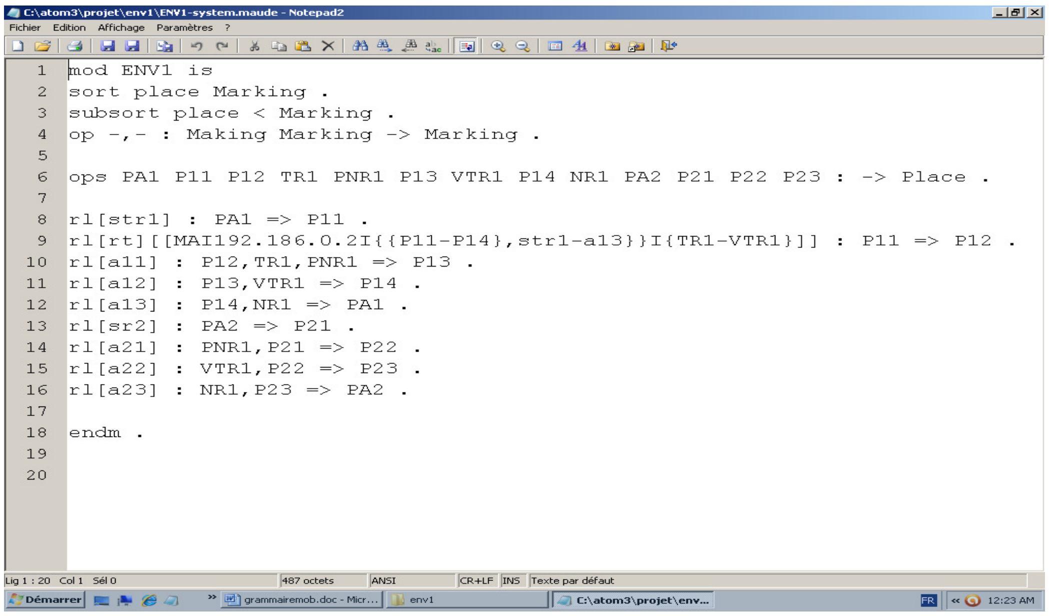


FIGURE 7: LRN MA-model after firing rt

Translating LRN Model to R-Maude Specification

The model of the LRN associated to E1 of FIGURE 6 is too huge in ATOM3, so because of the lack of space, only its R-Maude specification generated by the execution of the graph grammar on the ATOM3 tool and saved in the file env1-system.Maude,is given by FIGURE 8.



```
mod ENV1 is
sort place Marking .
subsort place < Marking .
op -,- : Making Marking -> Marking .

ops PA1 P11 P12 TR1 PNR1 P13 VTR1 P14 NR1 PA2 P21 P22 P23 : -> Place .

rl[str1] : PA1 => P11 .
rl[rt][[MAI192.186.0.2I{{P11-P14},str1-a13}}I{TR1-VTR1}]] : P11 => P12 .
rl[a11] : P12,TR1,PNR1 => P13 .
rl[a12] : P13,VTR1 => P14 .
rl[a13] : P14,NR1 => PA1 .
rl[sr2] : PA2 => P21 .
rl[a21] : PNR1,P21 => P22 .
rl[a22] : VTR1,P22 => P23 .
rl[a23] : NR1,P23 => PA2 .

endm .
```

FIGURE 8: Generated R-Maude specification env1-system.maude

5. PROTOTYPING R-MAUDE

R-Maude has been prototyped [22]. The prototype is a system composed from a text editor and an interpreter. The editor is used to enter the specification and commands which have been automatically generated by the execution of the graph grammar on the models to experiment, using ATOM3 tool. The interpreter executes commands and updates specifications. The system was experimented on a LAN (Local Area Network), consisting of a few machines. The system is installed on all hosts. So the specifications are edited everywhere. On every host, commands can be executed. The execution of commands will create the system dynamic. This dynamic can be shown as migration of specification's part (or else the whole) through the LAN.

The specifications (or their parts) are transferred in messages between machines, using
UDP protocol. The interpreter realized for R-Maude can be used to interpret Maude specifications. The major different is that in this newest interpreter, the interpretation of R-Rules is added. The label of an R-Rule precedes the rule, and it has the form [MT|L| IP@| S]. Semantics of these parameters is : MT: mobility type (MA, COD, REV, …), L: a multi-set of operations and rules to be moved, cloned or removed from or to the local host, IP@: IP address of distant host, S: sources to move or to remove from or to the local host. When specifications (or part of them) are moved, some resources (R) necessary to firing some rules become far (on another host). IP address of the far host appears with the concerned resource in the form: R[IP@].

The newest is that R-transition will be translated in R-Rules.
We consider that the two environments E1, E2 are specified as two specifications on two hosts (Host1 and Host2). Host1 has the IP address: 192.168.0.1, and Host2 has the IP address: 192.168.0.2.
On Host1, the specification is given by FIGURE 8.
and on Host2, we have the specification :

```
mod E2
sort Place Marking .
subsort Place << Marking .
op _,_ : Marking Marking ->Marking .
ops PA1,PA2,P31,P32,PNR2 : -> Place.
rl [str3] : PA2=>P31 .
rl [a31] : P31=>P32 .
rl [a32] : P32, PNR2=>PA2 .
endmod
```

As an example of a command, we have "rw PA1" on Host1. The execution of this command will produce respectively on Host1, and Host2 the two specifications:

```
mod E1
sort Place Marking .
subsort Place << Marking .
op _,_ : Marking Marking ->Marking .
ops PA1,PA2,P21,P22,P23,VTR1,PNR1,NR1:->Place.
rl [str2]: PA2=>P21 .
rl [a21] : P21, PNR1=>P22 .
rl [a22] : P22, VTR1=>P23 .
rl [a23] : P23, NR1=>PA2 .
endmod.
```

And in Host2 the produced specification s

```
mod E2
sort Place Marking .
subsort Place << Marking .
op _,_ : Marking Marking ->Marking .
ops PA1,PA2,P31,P32,PNR2 : -> Place.
ops VTR1, TR1:-> Place.
ops P11,P12,P13,P14:->Place.
rl [str3] : PA2=>P31 .
rl [a31] : P31=>P32 .
rl [a32] : P32, PNR2=>PA2 .
rl [str1] : PA1=>P11 .
rl [rt][MA|192.186.0.2|{{P11-P14},{str1-a13}}|{TR1,VTR1}] :P11=>P12 .
rl [a11] : P12, TR1, PNR2=>P13 .
rl [a12] : P13, VTR1=>P14 .
rl [a13]: P14,NR1[192.168.0.1]=>PA1.
Endmod
```

Finally, the state of the marking will be : "P12" on the Host2. At this point, the two specifications continue their execution on the two hosts where they reside.

6. CONCLUSION

In this paper, we have proposed an approach and a tool in dealing with the transformation of LRN models to their equivalent R-Maude specifications automatically. This transformation aims to make it possible to achieve the verification of properties of systems modeled using LRN, since the latter do not have tools for analysis and verification. Our approach is based on the combined use of meta-modeling and graph grammars wherein ATOM3 is used as a graph transformation tool.

This study opens up new perspectives for future research. In our forthcoming studies, we plan to hide the steps of the Simulation and thereby spare the user from invoking R-Maude language manually and manipulating the textual version of the simulation result. Thus, the latter will be returned in graphical way to LRN model structure. We plan also, to focus on modeling and analyzing aspects. In the modeling aspects, our concern will be much more on handling problems such as those of modeling multi-hops mobility, process states during travel, birth places and locations. As for the analysis aspect, we are currently working on a denotational semantics for LRN. It is to be underlined that the current R-Maude can be used only to simulate Models. Future works will handle specification analyzing, so we plan to integrate the LTL Maude Model checker in our tool to perform some verification of mobile systems properties, so that Maude model-checker will be adapted to reconfigurable Maude. For LRN, many extensions have been proposed, in [23], authors have proposed "Temporal LRN" and in [24] they have suggested "Coloured LRN". In line with these suggestions, we focus on using R-Maude to simulate models of these extensions,

7. REFERENCES

[1] A. Asperti, N. Busi. "Mobile Petri Nets". Technical Report UBLCS-96-10, Department of Computer Science University of Bologna, May 1996.

[2] AToM3 Home page, version 3.00, http://atom3.cs.mcgill.ca.

[3] M. R. Bahri, A. Hettab, A. Chaoui, E. Kerkouche. “Transforming Mobile UML Statecharts Models to Nested Nets Models using Graph Grammars: An Approach for Modeling and Analysis of Mobile Agent-Based Software Systems“. In Proccedings of IEEE SEEFM2009, the 2009 Fourth South-East European Workshop on Formal Methods. Thessaloniki, Greece,

[4] Dec 5th, 2009. pp. 33-39,

[5] M.A. Bednarczyk, L. Bernardinello, W. Pawlowski, L. Pomello. “Modeling Mobility with Petri Hypernets”. 17th Int. Conf. on Recent Trends in Algebraic Development Techniques, WADT’04. LNCS vol. 3423, Springer-Verlag, 2004.

[6] D. Xu, Y. Deng. “Modeling Mobile Agent Systems with High Level Petri Nets”. 0-7803-6583-6/00/ © 2000 IEEE.

[7] F. Dur, N. Steven, E. P. Lincoln, J. Meseguer. “principles of mobile maude”. In D.Kotz and F.Mattern, editors, Agent systems, mobile agents and applications, second international symposium on agent systems and applications and fourth international symposium on mobile agents, ASA/MA 2000 LNCS 1882, Springer Verlag. Sep 2000.

[8] C. Fournet, G. Gonthier. “The Join Calculus: a Language for Distributed Mobile Programming”. In Applied Semantics. International Summer School, APPSEM 2000, Caminha, Portugal, Sep00, LNCS 2395, Springer-Verlag. Aug 2002, pp. 268-332.

[9] A. Fuggetta, G. P. Picco, G. Vigna. “Understanding Code Mobility”. IEEE transactions on software engineering, vol. 24, no. 5, may 1998.

[10] L. Kahloul, A. Chaoui. “Labeled reconfigurable nets For modeling code mobility“. In proceedings of ACIT 2007, Lattakia, Syria.

[11] K. M. van Hee, I. A. Lomazova, O. Oanea, A. Serebrenik, N. Sidorova, M. Voorhoeve. “Nested Nets for Adaptive Systems”. 14 EE. ICATPN, 2006, pp. 241-260.

[12] 11, I.A. Lomazova. “Nested Petri Nets”; Multilevel and Recursive Systems. Fundamenta Informaticae vol.47, pp.283-293. IOS Press, 2002.

[13] J. Meseguer. “A Logical Theory of Concurrent Objects and its Realization in the Maude Language”. Agha G., Wegner P. and Yonezawa A., Editors, Research Directions in Object-Based Concurrency. MIT Press, 1992, pp. 314-390.

[14] R. Milner, J. Parrow, D. Walker. “A calculus of mobile processes”. Information and Computation, 100:1–77, 1992.

[15] R. Berger, I. Dori, S. Katz.”Modeling code mobility and migration: an OPM/Web approach”, Int. J. Web Engineering and Technology, Vol. 2, No. 1, pp.6–28, 2005

[16] G. Rozengerg, "Handbook of Graph Grammar and computing Graph Transformation", World Scientific, 1999.

[17] D. Sangiorgi, D. Walker. “The π-Calculus: A Theory of Mobile Processes”. Cambridge University Press, 2001.

[18] L. Athie, S. A. DeLoach. “Designing and Specifying Mobility within the Multiagent Systems Engineering methodology ” Special Track on Agents, Interactions, Mobility, and Systems (AIMS) at the 18th ACM Symposium on Applied Computing (SAC 2003). Melbourne, Florida, USA, 2003.

[19] R. Valk. “Petri Nets as Token Objects: An Introduction to Elementary Object Nets”. Applications and Theory of Petri Nets, LNCS vol.1420, pp.1-25, Springer-Verlag, 1998.

[20] M. Clavel, F.Durán, S.Eker, P.Lincoln, N. Marti-Oliet, J.Meseguer, J. Quesada. “Maude:specification and programming in rewriting logic”.SRI International, http://maude,.csl.sri.com, Januray 1999.

[21] J. Meseguer: “Conditional rewriting logic as a unified model of concurrency”. Theoretical Computer Science, 96 (1):73-155, 1992.

[22] P. C. Ölveczky, J. Meseguer:” Real-Time Maude : A tool for simulating and analyzing real-time and hybrid systems”. In K. Futatsugi, editor, Third International Workshop on Rewriting Logic and its Applications, volume 36 of Electronic Notes in Theoretical Computer Science. Elsevier, 2000.

[23] http://www.elsevier.nl/locate.entcs/volume36.html.

[24] L. Kahloul, Allaoua Chaoui. ”LRN/R-Maude Based Approach For Modeling And Simulation Of Mobile Code Systems”. Ubiquitous Computing and Communication Journal, vol. 3, No. 6, Dec. 2008.

[25] L. Kahloul, Allaoua Chaoui. ”Temporal Labeled Reconfigurable Nets for Code Mobility Modeling”. The International Workshop on (Trustworthy Ubiquitous Computing (TwUC 2007)

[26] associated to the iiWAS2007 conference.

[27] L. Kahloul, Allaoua Chaoui. “Coloured reconfigurable nets for code mobility modeling”. In Proc World Academy of Science, Engineering and Technology. Vol. 25, International Conference Venice, Italy. Nov 2007.