Automated Mapping Tool from $\mathcal{M}$oise$^+$ to Colored Petri Nets

Ricardo Machado, Arthur Zelindro, Eder Gonçalves, Diana Adamatti, Giovani Farias
Federal University of Rio Grande – FURG – Center for Computational Sciences – C3 – RS, Brazil
{ricardoarend, zelindroarthur, giovanifarias}@gmail.com, {dianaadamatti, edergoncalves}@furg.br

Abstract

The demand for systems incorporating artificial intelligence, such as multi-agent systems, is continually increasing. Simultaneously, there is a growing need for developing tools to support this field, ensuring better fault tolerance within projects. This is particularly crucial given that these systems possess characteristics that render them non-deterministic, thereby amplifying the challenge of conducting tests. To address this challenge, a mapping tool has been developed. This tool automatically generates a graphical model, facilitating the identification of test paths for a given multi-agent system. It operates by taking an XML file from $\mathcal{M}$oise$^+$, an organizational model for multi-agent systems, and translates it into a colored Petri net. The resulting mapping serves as a foundation for generating test cases essential for validating the $\mathcal{M}$oise$^+$ model, guiding system testing. Automation streamlines the process, enhancing speed, and eliminating the potential for human error.

Introduction

The utilization of Multi-agent Systems (MAS) has become commonplace for addressing various problem types, leveraging the unique characteristics of agents (Padgham and Winikoff 2005; Van Den Broek et al. 2005). However, testing MAS presents a formidable challenge due to their autonomy and deliberation within an open environment, rendering them context-sensitive systems (Houhamdi 2011; Dignum and Dignum 2001). The complexity inherent in MAS, coupled with the imperative to model and analyze their organizational behavior, has spurred considerable interest in Artificial Intelligence (AI). In this context, the organizational model $\mathcal{M}$oise$^+$(Hübner, Sichman, and Boissier 2002; Hübner et al. 2010; Boissier et al. 2020) has emerged as an effective approach for delineating the organizational structure and agent interactions. Conversely, Colored Petri Nets (CPN) have garnered widespread adoption as a graphical language for modeling concurrent systems and analyzing their properties(Jensen 1997; Jensen et al. 2009; Boucherit et al. 2020).

In (Gonçalves et al. 2022), a method for testing in MAS was introduced, involving the mapping of a $\mathcal{M}$oise$^+$ model into a CPN to generate potential test paths for sizing and identifying test cases. The structure of the network is designed based on the functional specification of $\mathcal{M}$oise$^+$, wherein the goal decomposition tree dictates the sequence, choice, or parallelism of goal satisfaction. However, the manual mapping process described in the article can be arduous and prone to errors. To address this challenge, this study proposes the development of a tool to automate the mapping of the $\mathcal{M}$oise$^+$ organizational model into a CPN. This tool is designed to parse $\mathcal{M}$oise$^+$ specifications and automatically generate the corresponding CPN model. By automating this process, researchers can save time and effort otherwise spent on manual translation tasks, ensuring a consistent and precise representation of the MAS's organizational behavior.

Mapping Tool Description

The primary objective of this project is to develop a mapping tool capable of generating a CPN model, utilizing a $\mathcal{M}$oise$^+$ organizational XML file as input. Python was chosen as the main programming language for implementing this tool due to its rich libraries that facilitate the reading and processing of XML files. The libraries employed in this project are *Minidom*[1] and *ElementTree*[2], both serve as parser libraries and play indispensable roles in the development process.

The mapping methodology employed in this project draws inspiration from CPN4M (Gonçalves et al. 2022), where the authors meticulously outline the steps involved in transforming various elements of a MAS, particularly the $\mathcal{M}$oise$^+$ organizational model, such as goals, missions, roles, and specifications, into elements compatible with CPN. We aim to automate the CPN4M mapping process, thereby enhancing efficiency and reducing errors.

Figure 1 provides an overview of the steps involved in our approach. In the initial step, an XML file is read using the *Minidom* library. This choice was made due to its capability to efficiently parse XML files, separating all the required structures within a given XML document. Beyond parsing and structure separation, *Minidom* facilitates navigation through the XML document by offering specific tools

[1]https://docs.python.org/3/library/xml.dom.minidom.html
[2]https://docs.python.org/3/library/xml.etree.elementtree.html

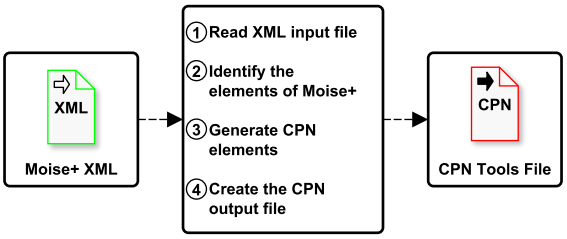

Figure 1: Steps of the proposal approach.

tailored to various use cases.

In the next step, the $\mathcal{M}$oise$^+$ structures are identified and segregated, considering three key elements: (i) *Structural Specification* (SS): this entails extracting roles and their respective groups; (ii) *Functional Specification* (FS): this involves extracting goals, their plan operators, and establishing the correct execution sequence based on the goals decomposition tree. Additionally, it identifies the mission to which each goal belongs and their respective cardinality; (iii) *Deontic Specification* (DS): information is gathered regarding the missions to which each role is affiliated.

At this step, to parse the goal decomposition tree integral to the FS, it was necessary to follow the $\mathcal{M}$oise$^+$ logic: traversing from left to right and bottom to top (Figure 2). The tree is navigated by descending from node to node as long as the current node has children, persisting until a leaf node, such as `g6` in Figure 2, devoid of descendants, is reached. Upon identifying a terminal node, the reading process is completed, and the structure is transcribed into the CPN file.

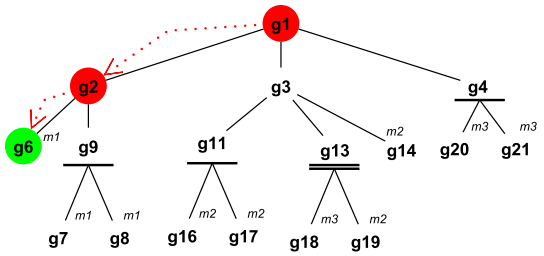

Figure 2: Example of functional specification reading

The reading done with the pattern of reading the first node without children, presented itself as an initial problem. This was due to the need to constantly store the current child number that was being read. When the reading ended and it was necessary to enter another node, this value was lost. So, the solution to this problem was *"to consume"* the node after the read was complete. This ensures that reading is always done on the first node without children, and if the current node has children, the process is repeated. Finally, the stopping condition is met when reaching the root node. In the example in Figure 2, this would be the node `g1`, which lacks any children as they were all read and duly consumed.

The third step involves generating an equivalent CPN element for each $\mathcal{M}$oise$^+$ element. In CPN, connections between different places are established through arcs, and transitions are depicted by rectangles. The $\mathcal{M}$oise$^+$ plan specifies how these connections are established, utilizing sequential, parallel, or choice operators. Figure 3 illustrates these connections for plan operators translated into CPN. In the sequence operator, the place representing goal `g1` leads to `g2`. In the choice operator, starting from `g0`, there is the option to transition to either `g1` or `g2`. Lastly, in parallel, the path from `g0` leads simultaneously to both `g1` and `g2`.

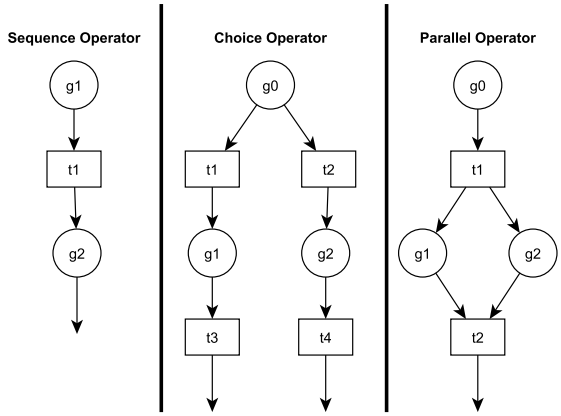

Figure 3: Plan operators converted to CPN

In the FS of $\mathcal{M}$oise$^+$, each goal is linked to a specific mission. This association is depicted within the ellipse representing the goal, situated beneath its name and enclosed within parentheses. For goals lacking an assigned mission, only the name of the goal is presented within the ellipse.

Roles and groups are symbolized by tokens (colors) and are incorporated into the CPN through declarations as a set of colors. The allocation of roles to their respective goals is accomplished via guard functions associated with transitions. These guard functions ensure that only the appropriate tokens progress to the subsequent place, maintaining the integrity of the system's representation.

Conclusion

Multi-agent systems are distributed systems comprising autonomous agents interacting in an environment to execute tasks and accomplish goals. The testing phase is pivotal in ensuring the correction and reliability of the entire software system, offering users assurances of functionality and safety.

The overarching objective of this work is realized through the development of a translation tool. This tool facilitates the direct conversion from a $\mathcal{M}$oise$^+$ file to a file that generates a CPN. By automating the generation of test cases for the organizational model of a MAS, the tool significantly saves time and resources for researchers, who can leverage it in future endeavors. In summary, the tool furnishes a more efficient and precise solution, mitigating errors inherent in manual translation, and thereby enhancing the generation of test cases for MAS.

References

Boissier, O.; Bordini, R. H.; Hubner, J.; and Ricci, A. 2020. *Multi-agent oriented programming: programming multi-agent systems using JaCaMo*. Mit Press.

Boucherit, A.; Castro, L. M.; Khababa, A.; and Hasan, O. 2020. Petri net and rewriting logic based formal analysis of multi-agent based safety-critical systems. *Multiagent and Grid Systems* 16(1):47–66.

Dignum, V., and Dignum, F. 2001. Modelling agent societies: Co-ordination frameworks and institutions. In *Portuguese Conference on Artificial Intelligence*, 191–204. Springer.

Gonçalves, E. M. N.; Machado, R. A.; Rodrigues, B. C.; and Adamatti, D. 2022. Cpn4m: Testing multi-agent systems under organizational model m oise+ using colored petri nets. *Applied Sciences* 12(12):5857.

Houhamdi, Z. 2011. Multi-agent system testing: A survey. *International Journal of Advanced Computer*.

Hübner, J. F.; Boissier, O.; Kitio, R.; and Ricci, A. 2010. Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous agents and multi-agent systems* 20(3):369–400.

Hübner, J. F.; Sichman, J. S.; and Boissier, O. 2002. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Brazilian Symposium on Artificial Intelligence*, 118–128. Springer.

Jensen, K.; Kristensen, L. M.; Jensen, K.; and Kristensen, L. M. 2009. Formal definition of timed coloured petri nets. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems* 257–271.

Jensen, K. 1997. *Coloured Petri nets: basic concepts, analysis methods and practical use*, volume 1. Springer Science & Business Media.

Padgham, L., and Winikoff, M. 2005. *Developing intelligent agent systems: A practical guide*, volume 13. John Wiley & Sons.

Van Den Broek, E. L.; Jonker, C. M.; Sharpanskykh, A.; Treur, J.; et al. 2005. Formal modelling and analysis of organizations. In *International Conference on Autonomous Agents and Multiagent Systems*, 18–34. Springer.