An Automatic Petri-net Generator for Modeling Multi-agent Systems

Meriem Taibi, Malika Ioualalen, Riad Abdmeziem
LSI - USTHB
Algiers, Algeria
emails: {taibi, ioualalen, abdmeziem}@lsi-usthb.dz

Abstract— A multi-agent system can be studied as a concurrent, asynchronous, stochastic and distributed computer system. These characteristics of multi-agent systems make them also a discrete-event dynamic system; it is, therefore, important to analyze the behavior of such system to ensure that it terminates correctly and satisfies other important properties. Several analytical methodologies have been used to study multi-agent system, particularly Petri nets. Petri nets have a well-defined mathematical structure that can be leveraged to provide formal analysis on discrete-event systems. In this work, we propose an automatic transformation to model multi-agent systems using Colored Petri nets.

Keywords-Multi-agent system; Colored Petri net; Modeling; Description language.

I. INTRODUCTION

Multi-agent systems have been widely studied in the past few decades, where several frameworks have been defined in order to apply the multi-agent system concept to different applications in control and optimization of complex systems [1][2]. An agent is a computer system or computer program that presents several complex characteristics. A Multi-Agent System (MAS) [3] consists of a set of agents, interacting to achieve a common goal. Generally, MAS are known to work properly in a dynamic large-scale complex environment (open environment), thanks to several properties like: autonomy, adaptability, robustness and flexibility. The complexity and capabilities of a multi-agent system are greater than those presented in distributed software systems. In both cases, the study of system properties is becoming more important due to the fact that we are faced more and more to deal with large complex dynamic systems.

Tests and simulations have contributed for a long time to validate such systems. However, these techniques allow to investigate just a part of the global behavior. By that, they differ from the formal verification techniques, which ensure that a property is verified by all possible system executions.

Therefore, the important challenge in this field is the development of analytical methods to assess key properties of such systems. Such methods could be used to impart a preliminary analysis of the multi-agent system, providing design and operation feedback before the development of expensive systems. Many models based on Belief – Desire – Intention (BDI) architecture were proposed in [4] and [5].

Other works include various attempts to deliver a formal model from AUML (Agent Unified Modelling Language) diagrams [6][7]. The advantage of these methods is that most developers are familiar with the (A)UML and an automatic transformation of their diagrams into formal models and model-check them, would greatly simplify the software quality control. The difficulty is that AUML diagrams allow much more freedom for the designer than formal models and the automatic translation is not trivial.

Petri nets have a well-defined mathematical structure that can be leveraged to provide formal analysis on discrete-event systems. In addition, Petri nets have been successfully used in several areas for the modeling and analysis of distributed systems [8].

Several studies have been proposed to model MAS with Petri nets (PN). In [9], a model was proposed for a promotional game of viral marketing on the Internet. Specifically, authors used stochastic Petri nets for modeling a multi-agent wish list. As well, Gazdare [10] used Colored Petri nets (CPN) as a formal method to model a transport system with containers, then, simulate and solve the storage problem. In EL Fallah-Seghrouchni [11], Boukredera [12] and Khosravifar [13], authors also proposed to use the CPN formalism to model interaction protocols.

In this work, we propose an automatic transformation for modeling multi-agent systems. This automation is based on two steps: first, the system is described using a language called MASDL, then, a set of transformation rules are applied to obtain the CPN models.

This document is organized as follows. Motivations and the problem statement are presented next. Section III gives an introduction to MAS. Section IV presents the main aspects of the language we define to specify MAS. In Section V, we present our transformation algorithm allowing to model MAS using CPN. Section VI presents our application, and finally, Section VII discusses the obtained results and presents future work.

We assume that the reader is familiar with Colored Petri net [14].

II. MOTIVATIONS

The Petri nets can be considered as graphic and mathematical tools of modeling and analyzing the discreet system, particularly the competitive, parallel and non-determinist ones. In the field of MAS, the previous works of the Petri nets concentrated on their uses and not on the creation of the new tools and platforms. The goal of our work is to develop a platform which generates automatically models for multi-agent system using CPN. The system in question must be described in an intermediate language. We find in literature two classes of specification languages [15][16]. The first allows the definition of agent and its behavior (e.g., AgentSpeak [17])
and the other describes the system environment (e.g., ELMS [18]). Therefore, the definition of a new language including both aspects is necessary. We propose, then, a new language based on XML. The use of XML has many advantages:

- **Universality**: The adoption of a simple and powerful syntax which allows the representation of the most generic models with hierarchical elements, attributes and textual content.
- **Interoperability**: Thanks to their universal syntax, XML documents are easily transportable and readable between systems.
- **Independence between models and data**: We can write an XML document without resorting ever to a schema. If we need to validate the document, we can build a schema afterward.

III. **MULTI-AGENT SYSTEMS (MAS)**

According to Weiss [19], agents are computational systems situated in some environment, and are capable of autonomous action in this environment in order to meet their design objectives. Agents perceive and interact with each other via the environment, and they act upon it, so that it reaches a certain state where their goals are achieved. Consequently, the MAS environment consists of a set of states \( S = \{s_1, s_2, \ldots \} \), where an agent can undertake a set of actions \( A = \{a_1, a_2, \ldots \} \) and perceive a set of percepts \( P = \{p_1, p_2, \ldots \} \). Therefore, environment modelling is an important issue in the development of multi-agent systems. Although, some multi-agent systems may be situated in an existing environment, in agent-based simulations, the environment is necessarily a computational process too, so modelling multi-agent environments is always an important issue. For this objective, we present in the next section Multi-Agent System Description Language, a language used for the specification of multi-agent environments.

IV. **MULTI-AGENT SYSTEM DESCRIPTION LANGUAGE**

In this section, we introduce the main aspects of the language we defined for the specification of the multi-agent system and its environment. The language is called Multi-Agent System Description Language (MASDL). MASDL is inspired from the Environment Description Language for Multi Agent Simulation (ELMS) language [18], which is an XML-based language that provides the ability to describe multi-agents. The syntax and the various components of our language are given below. The validation of the syntax is done using World Wide Web Consortium (W3C) scheme which is a grammar defined in XML formalism.

- **MAS general structure**: MAS specification contains the name of the system, a list of agents, a set of objects (system environment), a list of states (agents states and objects states) and finally a list of actions may be performed by agents. The code sample

    ```xml
    Listing 1 gives the general structure of the system.
    Listing 1: MAS description structure
    <MAS NAME = "">
    <AGENTS_LIST>
        <AGENT NAME = "">
        </AGENT>
    </AGENTS_LIST>
    <OBJECTS_LIST>
        <OBJECT NAME = "">
        </OBJECT>
    </OBJECTS_LIST>
    <STATES_LIST>
        <STATE/>
    </STATES_LIST>
    <ACTIONS_LIST>
        <ACTION NAME = "/>
    </ACTIONS_LIST>
    </MAS>
    
    - **Agent description**: This description contains the name of the agent, a list of its attributes (agent properties), the current state and list of actions. The following example in Listing 2, defines an agent named agent1 which has an attribute prop1 of type type1 with a value val1. The agent1 has state_agent1 like initial state and can perform action1 and action2.

    ```xml
    Listing 2: Agent description example
    <AGENT NAME = "agent1">
    <ATTRIBUTES>
        <ATT NAME= "prop1" TYPE="type1" VALUE="val1"/>
    </ATTRIBUTES>
    <CURRENTSTATE>
        <ITEM NAME = "state_agent1"/>
    </CURRENTSTATE>
    <ACTIONS>
        <ITEM NAME = "action1"/>
        <ITEM NAME = "action2"/>
    </ACTIONS>
    </AGENT>
    
    - **Resources description**: This concept allows the specification of the different objects in the MAS environment (all the entities in the environment that are not agent). An object class includes its name, the current state of the object, its identifier and the available quantity (a negative amount is used in case where the amount is unlimited).

    ```xml
    Listing 3: Example of object description
    <OBJECT NAME = "Object1">
    <ATTRIBUTES>
        <ATT NAME= "quantity" TYPE= "int" VALUE="100"/>
    </ATTRIBUTES>
    <CURRENTSTATE>
        <ITEM NAME = "state_res"/>
    </CURRENTSTATE>
    </OBJECT>
    </OBJECTS_LIST>
    
    The code sample Listing 3 above, defines a resource called object1. This object has an integer attribute representing the number of units available in the system and current state state_res.
**State description** The state is defined in the tag `<AGENT_STATE_LIST>` when it relates to agents and in the tag `<OBJECT_STATE_LIST>` when it concerns resources. A state is described by its name and an informal description given by the designer, it corresponds to the semantics of the state. A state represents a situation in which an agent or a resource can be, during the running of the system. In [Listing 4] the code exemplifies definition of two states (one for agent state and the second for resource one).

**Action description** The description of the action includes its name, its content and an informal description. *Content* specifies the agents that are involved in the execution of the action and also the potential resources (objects) needed. The agents specification includes their name, input and output states. An action can be executed by an agent, if it is in the defined input state. These states represent the preconditions of the action. Resources can also be instantiated or removed by action. The specification of resources including their type, input and output states and the number of units to subtract (sub_quantity_entry) or to add (add_quantity_exit).

**Algorithm assumptions:** We assume that our algorithm has as input and output data the following sets, described in the [Fig. 2] which are calculated from the MASDL specification.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG</td>
<td>Set of agent</td>
</tr>
<tr>
<td>RE</td>
<td>Set of resources</td>
</tr>
<tr>
<td>SA</td>
<td>Set of agents states</td>
</tr>
<tr>
<td>SR</td>
<td>Set of resources states</td>
</tr>
<tr>
<td>AC</td>
<td>Set of actions</td>
</tr>
<tr>
<td>P</td>
<td>Set of places</td>
</tr>
<tr>
<td>T</td>
<td>Set of transitions</td>
</tr>
<tr>
<td>Arc</td>
<td>Set of Arcs</td>
</tr>
<tr>
<td>$C_1$</td>
<td>Color with the structure &lt;id,state&gt;</td>
</tr>
<tr>
<td>$C_2$</td>
<td>Color with the structure &lt;id,state,quantity&gt;</td>
</tr>
</tbody>
</table>

**Algorithm assumptions:** We assume that our algorithm has as input and output data the following sets, described in the [Fig. 2] which are calculated from the MASDL specification.

### V. Automatic Multi-Agent Modeling using CPN

The objective of this section is to give an algorithm allowing to transform a description of multi-agent system to Colored Petri net models. The CPN models obtained are written in a XML based language with a specific syntax which we call *Petri Net Description Language (PNDL)*.

#### A. Transformation algorithm

The transformation algorithm [1] allows to generate automatically CPN models of the described system. The important steps of the algorithm are given in [Fig. 1] Based on MASDL language, the system is defined by a set of states $S = \{s_1, s_2, \ldots\}$ and agent is able to perform a set of actions $A = \{a_1, a_2, \ldots\}$. The execution of an action causes changes in the environment.

![Fig. 1: Schema of the modeling process.](image)

In the example above [Listing 5] an action named *action1* is defined and has as a precondition: the agent Ag must be in the state state1_Ag and the object Res in the state state1_Res. As a result of the execution of this action, the agent Ag will be in the output state state2_Ag and the resource Res in state2_Res state with the production of three units of this resource.
1) If an agent \( a \) is initially in the state \( sa \), we put one token of color \(< a, sa > \in C_1\), in the place \( p_{sa} \);
2) If an resource \( r \) is initially in the state \( sr \), we put one token of color \(< r, sr, quantity > \in C_2\), in the place \( p_{sr} \).

B. Output format

We propose an XML-based language for the description of the models generated by the algorithm. We entitle our language Petri Net Description Language, which is based on the tags \(< \text{PLACES}>\), \(< \text{TRANSITIONS}>\) and \(< \text{ARCS}>\) to describe the model and on \(< \text{COLORS}>\) and \(< \text{TOKENS}>\) to give its marking. The general structure of the language is presented in the following Listing

Listing 6: Description of CPN Model

```xml
<RDPC NAME = "RdP_Example">
  <PLACES>
    <PLACE NAME = "p1"/>
    <PLACE NAME = "p2"/>
  </PLACES>
  <TRANSITIONS>
    <TRANSITION NAME = "t1"/>
  </TRANSITIONS>
  <ARCS>
    <ARC FROM="p1" TO="t1"/>
    <ARC FROM="t1" TO="p2"/>
  </ARCS>
  <COLORS>
    <COLOR NAME = "c1"/>
  </COLORS>
  <TOKENS>
    <TOKEN COLOR = "c1" PLACE = "p1"/>
  </TOKENS>
</RDPC>
```

VI. RUNNING MASDL ENVIRONMENT

The use of XML provides various advantages, wide range of XML tools are currently available and it can be useful for the future development. The validation of the description is done using W3C scheme. For the implementation of our tool, we chose Java, which allows us to use Java Architecture for XML Binding (JAXB) and Application Programming Interface (API) to create XML application data. The global architecture of our application is shown in Fig.

Our tool allows users to introduce environment specifications from a graphical interface, as shown in Fig. With this interface, users do not need to deal with the language syntax but just fill the different fields.

Filled fields will be checked and compiled to generate the corresponding XML file, as shown in Fig. on which the transformation algorithm will be applied. To generate a
VII. Conclusion and Future Works

In this paper, we introduced the MASDL language, used for the specification of the agents and their environment. The language is based on XML and is independent of the agent runtime platform and implementation language. We defined also transformation rules to obtain formal models from the system specification to analyze and verify the described multi-agent system. We plan in our further work to connect our tool to another verification tool, as CPN Tool or GreatSPN for the general properties verification (deadlock, boundedness, etc.). We will focus mainly on extending our model by introducing the temporal dimension in order to perform a quantitative analysis and compute MAS system performances (average waiting time, average resources available, etc.).

REFERENCES


