An Extensive Model Checking Framework for Multi-agent Systems

(Demonstration)

Songzheng Song, Yang Liu, Jie Zhang
Nanyang Technological University, Singapore
{songsz, yangliu, zhangj}@ntu.edu.sg

Jun Sun
Singapore University of Technology and Design
sunjun@sutd.edu.sg

ABSTRACT

In this work, we propose a novel probabilistic modeling language PML-MAS to capture the stochastic characteristics of multi-agent systems (MASs). Moreover, we design a model checking framework for MAS, which is highly extensible. It provides powerful modeling editor, interactive simulator and automatic verifier for MASs. In addition, it can support various MAS model languages via extracting their semantic models and verification algorithms.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligent**]: Distributed Artificial Intelligence - Intelligence Agents, Multiagent Systems

General Terms

Languages, Verification

Keywords

Formal Verification, Model Checking Framework, Extensibility

1. MOTIVATION

Multi-agent systems (MASs) are designed for many applications where tasks are difficult or inefficient for an individual agent to perform. However, the existence of multiple autonomous agents in MASs makes it highly nontrivial to precisely analyze the system behaviors. In addition, agents may have unreliable or even random behaviors because of uncertain environments and the potential stochastic dynamics in MASs. For instance, agents in negotiation may behave randomly according to others' execution. This kind of randomization makes MASs analysis even more difficult.

Recently, model checking techniques are widely used in analyzing MASs due to their completeness and automaton. Several model checker are proposed for modelling and verifying critical properties of MASs, e.g., MCMAS [4] and MCK [2]. Besides these stand-alone tools, Hunter *et al.* [3] propose to use Brahms to model MASs and apply existing model checkers to fulfill the verification. These approaches, however, still have some limitations. Specifically, MCMAS and MCK mainly focus on concurrent systems without stochastic behaviors, which limits their application in unreliable environments or agents with random behaviors. The framework in [3] translates the intermediate representation of Brahms model to other model checkers' input languages, which may be inefficient due to the additional translation.

In this demonstration paper, we propose a novel probabilistic modeling language for multi-agent systems (namely PML-MAS), which aims at modeling MASs with stochastic dynamics. The semantic model of PML-MAS is Probabilistic Automata (PA). PA supports full nondeterminism combined with probabilistic choices, therefore randomized behaviors of agents can be analyzed easily. A variety of properties are supported, such as deadlock checking, reachability checking and Linear Temporal Logic (LTL) checking. Moreover, instead of just implementing a model checker for PML-MAS, we design a model checking framework which separates modeling, abstraction techniques, semantic representations of a state space and verification algorithms into four loosely coupled layers. Therefore the most advanced techniques can be integrated into this framework with least effort. Semantic models supported by our framework include Labeled Transition Systems (LTS), Timed Transition Systems (TTS) and PA to cover a wide range of MAS behaviors and their analysis.

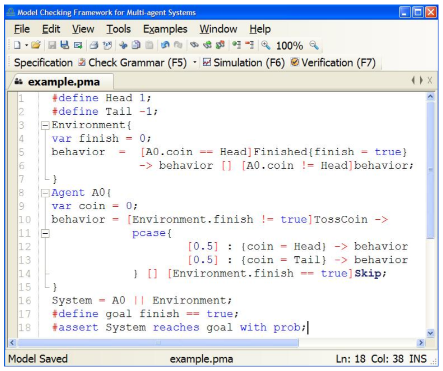
Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

2. PML-MAS

The design of PML-MAS is based on PCSP# language proposed by Sun *et al.* [5] while aiming at MASs, therefore PML-MAS combines the expressiveness of PCSP# with the specific characteristics of MASs. PML-MAS models have two levels: agent level, which describes the dynamics of the environment and the behaviors of each agent, and system level, which regulates the composition relation among the components in the agent level.

Each agent in PML-MAS is modeled by a finite set of finite-domain local variables and its behavior process. The compositional operators supported in the process can easily handle hierarchical control flows in each agent, meanwhile, scenarios with stochastic behaviors can be captured. On system level, we have $Sys = A_0 || \cdots || A_n || Environment$, which means that the participating agents and the environment are executing concurrently. At each round, every agent from A_0 to A_n could pick one action independently according to their own protocols. When agents finish their actions, $Environment$ updates its information accordingly.

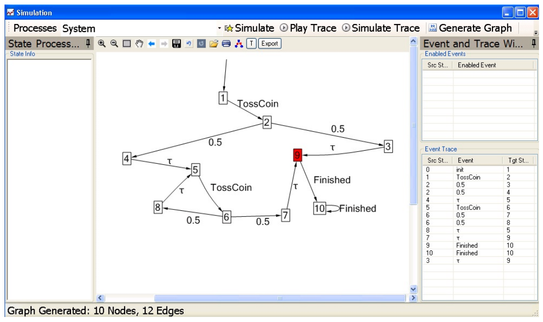
A simple PML-MAS model is shown in Fig. 1, which is in the built-in editor of our model checking framework. Line 1-2 define 2 constants $Head$ and $Tail$. *Environment* is defined in line 3-7, which has a variable *finish*. Line 8-15 are the definition of an agent A_0, and *coin* is declared as a local variable in A_0, which affects the behaviors of *Environment*. Vice versa, A_0's behavior is also affected by *finish*. At each round, as long as $finish$ is not $true$, A_0 executes $TossCoin$ to toss a coin, otherwise it will terminate. The outcome of tossing should follow the uniform distribution, which is described by $pcase$. After A_0's execution, $Environment$ could


Model Checking Framework for Multi-agent Systems
File Edit View Tools Examples Window Help
100%
Specification Check Grammar (F5) Simulation (F6) Verification (F7)
example.pma

```
1   #define Head 1;
2   #define Tail -1;
3  Environment{
4   var finish = 0;
5   behavior  =  [A0.coin == Head]Finished{finish = true}
6                -> behavior [] [A0.coin != Head]behavior;
7  }
8  Agent A0{
9   var coin = 0;
10  behavior = [Environment.finish != true]TossCoin ->
11              pcase{
12                      [0.5] : {coin = Head} -> behavior
13                      [0.5] : {coin = Tail} -> behavior
14              } [] [Environment.finish == true]Skip;
15 }
16  System = A0 || Environment;
17  #define goal finish == true;
18  #assert System reaches goal with prob;
```

Model Saved   example.pma   Ln: 18 Col: 38 INS

Figure 1: Editor of the Model Checker Framework



Src St...	Enabled Event

Event Trace

Src St...	Event	Tgt St...
0	init	1
1	TossCoin	2
2	0.5	3
2	0.5	4
4	τ	5
5	TossCoin	6
6	0.5	7
6	0.5	8
8	τ	5
7	τ	9
9	Finished	10
10	Finished	10
3	τ	9

Graph Generated: 10 Nodes, 12 Edges

Figure 2: Simulator of the Model Checker Framework

choose its action accordingly. Lines 17-18 define one desired property: *what is the probability that the system can reach the goal?* Here *goal* represents that $finish$ is $true$.

3. MODEL CHECKING FRAMEWORK

Besides the powerful editor shown in Fig. 1, an interactive simulator and a verifier are also integrated to our framework. In the discrete-event simulator, users can interactively and visually simulate system behaviors step by step. The state space representing the above example $System$ can automatically generated by our simulator, as shown in Fig. 2. The transitions are labeled by probabilities and actions. If the labeling is an action, it indicates the probability for this transition is 1. τ represents the invisible actions in the system, i.e., the variable updates. The defined property can be verified in the verifier to tell uses what the desired probability is. Detailed demonstration can be found at [1].

Moreover, our framework has four layers to increase its extensibility: modeling layer, abstraction layer, semantic model layer and verification layer, as shown in Fig. 3.

Multiple languages are potentially supported in the modeling layer, each of which is an independent module, therefore users can choose their familiar languages. PML-MAS has been integrated in our framework, and other popular languages such as ISPL (modeling language of MCMAS) and MCK are our ongoing work.

Various abstraction techniques aim at hiding unnecessary information from the models and reducing the size of semantic models. For instance, symmetry reduction is used to abstract the behaviors of a system with identical sub-components. According to users' requirements, different abstractions can be applied.

In the semantic model layer, according to different characteristic of the system and the operational semantics of modeling languages, three semantic models are supported. We use explicit model checking approach so that the explicit state information is stored in the memory during the verification. LTS are used to represent

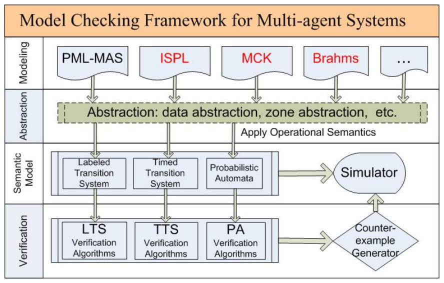

Model Checking Framework for Multi-agent Systems
Modeling
PML-MAS
ISPL
MCK
Brahms
...
Abstraction
Abstraction: data abstraction, zone abstraction, etc.
Apply Operational Semantics
Semantic Model
Labeled Transition System
Timed Transition System
Probabilistic Automata
Simulator
Verification
LTS Verification Algorithms
TTS Verification Algorithms
PA Verification Algorithms
Counter-example Generator

Figure 3: Architecture of the Model Checker Framework

the concurrent systems, such as models in MCMAS. TTS represent systems under timing constraints. Although current modeling languages do not support timing, further extension is feasible. In addition, PA is implemented to capture the stochastic behaviors of languages such as PML-MAS.

In the verification layer, different algorithms are implemented according to the specific semantic model. For instance, we use Büchi Automata approach to verify LTL properties in concurrent systems and apply Rabin Automata approach to do the LTL verification in stochastic systems.

Given this layered design, it is convenient for users to implement model checkers for their own languages. What they need is to implement the parser from their languages' syntax to corresponding semantic models; afterwards, they can reuse the existing algorithms to perform the verification. An existing language even can be extended to support semantic models which it does not before. Users can also add their own verification algorithms in verification layer. We remark that MCMAS applies symbolic model checking while our framework adopts explicit representation of states. In average, our current implementation processes 10K states per second (or million states in one hour), indicating that the efficiency of our framework is competitive. More details can also be found at [1].

4. CONTRIBUTIONS AND FUTURE WORK

The contributions of our work are summarized as follows: 1) We propose a new modeling language called PML-MAS to describe the stochastic behavior in MASs; 2) we design a model checking framework aiming to support different modeling languages, which is highly extensible. For the future work, we will extend the framework to integrate more modules such as ISPL, MCK and Brahms. In addition, other attractive properties of MASs will be supported, such as alternating-time temporal logic and epistemic properties.

5. ACKNOWLEDGEMENT

This work is supported by "Formal Verification on Cloud" project under Grant No: M4081155.020.

6. REFERENCES

[1] `http://youtu.be/9YhQ41Da7Mk`.

[2] P. Gammie and R. van der Meyden. Mck: Model checking the logic of knowledge. In *CAV*, pages 479–483, 2004.

[3] J. Hunter, F. Raimondi, N. Rungta, and R. Stocker. A synergistic and extensible framework for multi-agent system verification. In *AAMAS*, pages 869–876, 2013.

[4] A. Lomuscio, H. Qu, and F. Raimondi. Mcmas: A model checker for the verification of multi-agent systems. In *CAV*, pages 682–688, 2009.

[5] J. Sun, S. Z. Song, and Y. Liu. Model Checking Hierarchical Probabilistic Systems. In *ICFEM*, pages 388–403, 2010.