Model and Verify CPS using Probabilistic Automata

Asm Nurussafa[1]

Contents

Abstract: This paper provides an overview of the field of cyber physical systems and the importance of their verification and validation. It highlights the use of formal verification as a method to rigorously analyze and prove the correctness of these systems. The main focus of the paper is on the application of Markov decision processes (MDPs) in the formal verification of cyber physical systems. MDPs provide a powerful tool for modeling complex systems with uncertain and stochastic behavior, and the ability to analyze and optimize decision-making processes. The paper also explores the use of MDPs for verifying probabilistic reachability properties, which are essential for ensuring the safety and reliability of cyber physical systems.
Keywords: Cyber Physical Systems, Verification and Validation, Probabilistic Automata, Markov Decision Process.

[1] asm.nurussafa@stud.hshl.de

1 Introduction

1.1 Cyber Physical Systems (CPS)

According to Edward A. Lee [Le06], the definition of cyber-physical system is as follows: "Cyber-Physical Systems (CPS) are integrations of computation with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa.Ĭn simple terms, CPS is where the physical world and digital world comes together. These complex systems use computation and communication as the means to sense, reason, actuate and control the physical processes. This means that these systems are combinations of sensors, actuators and communication networks to monitor and control these physical components and then, provide a feedback to the computation and control components. These systems are typically used in safety-critical real-time applications such as transportation, healthcare and manufacturing.

One of the key characteristics of CPS is their ability to sense and actuate in real-time, which enables advanced control algorithms to be implemented. This real-time monitoring and control of the physical processes allows for efficient and effective operation, as well as the ability to respond quickly to changes in the environment. Verification and validation (V&V) is an important process in the development of cyber physical systems (CPSs) to ensure that the system operates as intended and meets its requirements. V&V is essential to ensure their safety and reliability.

1.2 Verification and Validation (V&V)

Formal verification in cyber physical systems (CPSs) is the process of using mathematical methods and logical reasoning to prove that a system satisfies its requirements. It is a method of verification that aims to prove the correctness of a system's design by using formal methods such as logic, model checking, theorem proving, and other mathematical techniques. [La] Formal verification can be applied to various aspects of a CPS, including its control algorithms, communication protocols, and software. By using formal methods, it's possible to prove that a system will behave as intended under all possible scenarios, which increases the confidence that the system is correct and safe. One of the key advantages of formal verification is that it can be automated, which makes it possible to perform a large number of verification checks in a relatively short amount of time. This is particularly useful in CPSs, which are complex systems that often have a large number of components and interactions. Formal verification can also be used to generate test cases, which can be used to validate the system's performance under actual or simulated conditions. It is important to note that formal verification is not a substitute for testing and simulation, but rather a complementary method that can be used to increase the confidence in the system's safety and reliability.

Validation in cyber physical systems (CPSs) is the process of evaluating the system's performance under realistic conditions to ensure that it meets its intended use and requirements, i.e. the client's actual needs and expectations are met. It is an essential step in the development of CPSs to ensure that the system operates as intended and meets its requirements, and is usually performed after the system has been designed and implemented. It typically involves testing the system in a laboratory environment or in the field, and may involve testing the system with actual users or customers. The results of validation are used to evaluate the system's performance and identify any defects or issues that need to be addressed. Hence, it is important to note that validation is actually complementary to verification of a system.

1.3 Probabilistic model checking

Probabilistic model checking is a method of formal verification that is used to analyze systems with probabilistic or stochastic behavior. It is a technique that combines formal verification with probabilistic reasoning, and it allows to reason about the behavior of a system in terms of probabilities of certain events occurring. The process of probabilistic model checking starts with a formal model of the system, which is typically a probabilistic automaton or a Markov Chain. This model describes the system's behavior in terms of states and transitions. The next step is to specify the properties that we want to verify, which are usually expressed in temporal logics such as PCTL or CSL. Once the model and the properties are specified, a verification algorithm is used to determine whether the system satisfies the properties or not. The outcome of the verification is a probability of the system satisfying the properties, which can be used to evaluate the system's reliability. [Ed99]

Probabilistic model checking is important because it allows to reason about the behavior of complex systems that exhibit probabilistic or stochastic behavior. This is particularly useful in the context of cyber-physical systems, where the behavior of the system is often influenced by random events such as sensor noise or communication failures. By using probabilistic model checking, it's possible to reason about the system's behavior in terms of probabilities, and to quantify the system's reliability in terms of the probability of certain events occurring. Probabilistic model checking is also important because it allows to perform verification in an automated and efficient way. This is particularly useful for systems with a large number of states and transitions, which would be difficult to verify manually. [SST18] . More about probabilistic model checking can be found in: [KNP07].

2 Probabilistic Automata-Markov Decision process (MDP)

Now there are two main categories of probabilistic models as shown in the table Fig.1, fully probabilistic and non-deterministic models. Among these, we are going to focus on Markov Decision Process, i.e. probabilistic automata. To discuss Markov decision processes (MDPs) further, it is essential to have an understanding of basic probability theory, references to

these would be helpful: [Bi95], [Fe68]. Additionally, to keep the paper short, it is advisable to haves some idea of the model discrete-time Markov chains (DTMCs) [St94], [Ku16].

	Fully probabilistic	Nondeterministic
Discrete time	Discrete-time Markov chains (DTMCs)	Markov decision processes (MDPs) (probabilistic automata)
Continuous time	Continuous-time Markov chains (CTMCs)	CTMDPs/IMCs
		Probabilistic timed automata (PTAs)

Fig. 1: Probabilistic models.

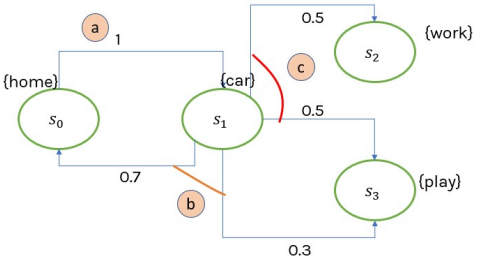

Fig. 2: Markov Decision Process example.

Markov Decision Process (MDP) is basically an extension of discrete-time markov chains (DTMCs), which revolves around the key feature of Markov property, which states that the probability of transitioning to any particular state is dependent only on the current state and the action taken, and not on the history of states that preceded it. The main difference in MDPs is that it takes non-deterministic choices into account. It is often observed that certain elements of a system may not exhibit probabilistic behavior and as such, should not be represented using probabilistic models. Some examples of non-determinism include: concurrency, unknown environments, underspecification, abstraction etc. In simple terms, Markov Decision Processes (MDPs), like discrete-time Markov chains, consist of a discrete set of states that represent various configurations of the system being modelled. The transitions between states occur in discrete time intervals, and the selection of actions is non-deterministic. Once an action is chosen, the subsequent state is determined probabilistically based on the current state and the chosen action.

According to [BK08], the formal definition of MDP is the following:

A Markov Decision Process, M, is a tuple (S, s_0, P, L) where: S is a finite set of states („state space"), $s_0 \in S$ is a initial state, $L : S \rightarrow 2^{AP}$ is a labelling function, a (partial) transition probability function, denoted as $P : S \times A \rightarrow$ Dist (S) is used in this model, where A is a set of actions and $Dist(S)$ is the set of discrete probability distributions over the set of states S. An action is considered feasible or can be executed in a given state S, if $P(s, a)$ is well-defined. The set of actions that can be performed in a state S is represented as $A(s)$.

To explain this, a simple made-up example is presented as in Fig.2. It can be observed that there are four states, so $S = \{s_0, s_1, s_2, s_3\}$. There are also four atomic propositions, $AP = \{$ home, car, work,play $\}$, for instance, $L\ (s_0) = \{$ home $\}$ and so forth. Now, to transition from *home* to *car* , there is only one action, a, so it has a probability of 1, from this we can infer, $P\ (s_0, a) = [s_1 \mapsto 1]$. Once in state s_1 (car), there are two possible actions, either it chooses action b and probabilistically, goes back to s_0 with probability 0.7 or it transitions to s_3 $(play)$ with a probability of 0.3. From this we can infer, $P\ (s_1, b) = [s_0 \mapsto 0.7, s_3 \mapsto 0.3]$. The other possible action, c is where it can transition from s_1 to s_2 with a probability of 0.5

or s_3 with a probability of 0.5 and from this we can infer, $P\,(s_1, c) = [s_2 \mapsto 0.5, s_3 \mapsto 0.5]$. The choice between actions a and b is non-deterministic. The idea behind this simple example can be carried forward to create more complex systems, such as asynchronous parallel compositions of multiple DTMCs.

A $path$ in a Markov Decision Process (MDP) is a sequence of transitions between states, along with an action a, representing a possible execution or behavior of the system that the MDP is modeling. It can be either finite or infinite. The set of all infinite paths of MDP starting from a particular state s is represented as $Path(s)$. These paths account for both non-deterministic and probabilistic choices made in the system.

Following this, if we wanted to consider the probability of some behavior of a MDP, we need *'strategies'*. To do this, we need to resolve the non-deterministic choices, i.e. by defining the paths of the MDP and this would, in actuality, result in a DTMC. From this, we can define a probability measure over the paths. A *strategy* can also be referred to as a *ßcheduler", "policy",* or *ädversary"*. In formal terms, a strategy, denoted as σ, of a MDP is a function that maps every finite path to an action that is available for the final state of the path, s_n, so that it resolves any non-determinism based on the execution history.

We can use the same example in Fig. 2, to explain how strategies are used in MDPs. For instance, we define a strategy, σ_1 that picks the action c the first time it is in state s_1. From this, we can infer, $\sigma_1\,(s_0 s_1)$ = c. Next, in another instance, let us define another strategy, σ_2 that chooses action b the first time and then, chooses action c, from this we can derive, $\sigma_2\,(s_0 S_1 S_0 s_1)$ = c. And from this, we can derive the respective paths.

3 Probabilistic Reachability

There are many ways to verify a system both in quantitative and qualitative manner. For the sake of this paper, we are going to look at the most fundamental quantitave property, i.e. probabilistic reachability. This property concerns the probability of reaching a target set $'T'$, $P^\sigma(s, T)$ would be the probability of reaching target $'T'$ under the strategy σ from state s, just as in the cases of DTMCs. Probabilistic automata or MDPs provide the best-/worst - case analysis based on lower/upper bounds on probabilities over all strategies. So, $-P^{\min}(s, T) = inf_\sigma P^\sigma(s, T)$ is how we would denote the minimum probability of reaching T over all strategies. $P^{\max}(s, T) = \sup_\sigma P^\sigma(s, T)$ is how we would denote the maximum probability of reaching T over all strategies.

Let us again look at the example in Fig. 2 to understand this. For instance, we choose our target, T to be s_3 $(play)$ and we consider that we choose a strategy σ_i that selects action b for the first $i-1$ times in state s_1 and then action c. So, for the first instance, the probability would simply go to s_3 $(play)$, so from this, we can derive: $P^{\sigma_1}\,(s_0, T) = 0.5$. For the next instance, since it chooses b first and then c, it would transition first to s_0, then to s_1 and lastly, to s_3. From this, we can derive: $P^{\sigma_2}\,(s_0, T) = 0.3 + 0.7 \cdot 0.5 = 0.65$. For the

next instance, it chooses the action b twice and then, chooses c. From this we can derive: $P^3\ (s_0, T) = 0.3 + 0.7 \cdot 0.3 + 0.7 \cdot 0.7.0.5 = 0.755$. We keep doing this, and we will observe that maximum probability reaches 1. As a result, we can concur that $P^{\min}\ (s_0, T) = 0.5$ and $P^{\min}\ (s_0, T) = 1$.

There are other strategies, for instance memoryless or finite-memory strategies that can be applied.

4 Verifying and computing reachability properties

There are several approaches to computing reachability properties. Value iteration method gives us an approximate with iterative solution method and corresponds to fixed point computation. These are preferably in practice to be done in a software, e.g. PRISM. Some other methods, for instance, linear optimisation techniques that provides exact solution using well-known mathematical methods and policy iteration method that iterates over different strategies are used to compute such property.

Let us now look at an example to verify such properties in a system. We will take into consideration the famous dining philosopher's problem. The problem states that only a certain number of philosophers spend their time thinking or eating. A philosopher needs both forks in order to eat and a philosopher may only pick up one fork at a time and when he is done eating, both forks are placed back and returns to thinking.

Based on a randomised solution from Lehmann and Rabin [LR81], we can derive a MDP model as shown in Fig. 3. The model shows the different states for one philosopher at a time and the respective probabilities for each transition.

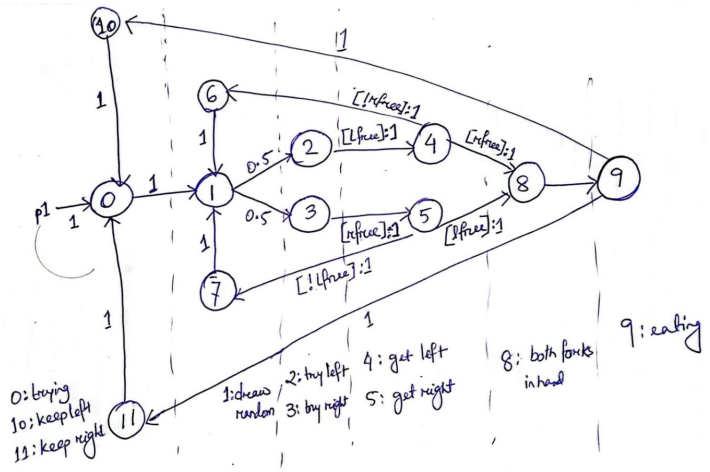

Fig. 3: Hand-drawn model to dining problem.

To implement this MDP, we will take the help of a software named $PRISM$, which is a probabilistic model checker. Since it would be cumbersome to explain the solution algorithm here, the solution to the algorithm for this model can be found in the repository: `https://github.com/Asm-Nurussafa/Probabilistic-Automata`. The solution ensures that atleast one of gets to eat (no deadlock). For this, the software uses the verification command in the properties list as ,
$filter(forall,"hungry" \implies P >= 1[F"eat"])//deadlockfreeness.$
This would produce a prompt that shows that this verifies the deadlock property, as shown in diagram Fig. 4

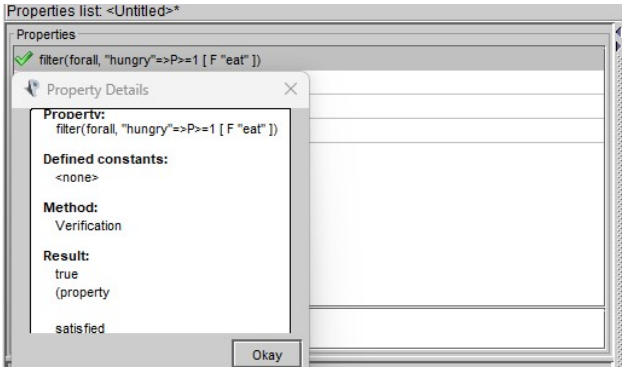

Fig. 4: Deadlock property satisfied.

The software can also be used to verify the minimum or maximum expected time to reach a certain state for each or all the philophers.

5 Conclusion

In conclusion, cyber-physical systems are becoming increasingly important in various fields in thriving of the Industry 4.0. The verification and validation of these systems is crucial to ensure their safe and correct operation. We have discussed in this paper how Probabilistic Automata such as Markov Decision Processes can be used to model such systems and then systematically, derive various properties and verify them. In this paper, we discussed one of the fundamental properties of such system, probabilistic reachability using the philosophers dining problem, that at the end, verified of having no deadlock. The notion of such examples can be taken forward to model and verify more complex cyber-physical systems.

6 Declaration of Originality

I, ***Asm Nurussafa***, herewith declare that I have composed the present paper and work by myself and without the use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form have not been submitted to any examination body and have not been published. This paper was not yet, even in part, used in another examination or as a course performance. I agree that my work may be checked by a plagiarism checker.

12.01.2023, Hamm **Asm Nurussafa**

Bibliography

[Bi95] Billingsley, P: Probability and Measure. 3rd Wiley. New York, 1995.

[BK08] Baier, Christel; Katoen, Joost-Pieter: Principles of model checking. MIT press, 2008.

[Ed99] Edmund, M; Clarke, Jr; Grumberg, Orna; Peled, Doron A: Model checking. MIT Press, p. 314, 1999.

[Fe68] Feller, William: An introduction to probability theory and its applications, volume IJ Wiley & Sons. New York,, 1968.

[KNP07] Kwiatkowska, Marta; Norman, Gethin; Parker, David: Stochastic model checking. In: International School on Formal Methods for the Design of Computer, Communication and Software Systems. Springer, pp. 220–270, 2007.

[Ku16] Kulkarni, Vidyadhar G: Modeling and analysis of stochastic systems. Chapman and Hall/CRC, 2016.

[La] Langley Formal Methods Program • What is formal methods (https://shemesh.larc.nasa.gov/fm/fm-what.html).

[Le06] Lee, Edward A: Cyber-physical systems-are computing foundations adequate. In: Position paper for NSF workshop on cyber-physical systems: research motivation, techniques and roadmap. volume 2. Austin, TX, pp. 1–9, 2006.

[LR81] Lehmann, Daniel; Rabin, Michael O: On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In: Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 133–138, 1981.

[SST18] Seshia, Sanjit A; Sharygina, Natasha; Tripakis, Stavros: Modeling for verification. In: Handbook of Model Checking, pp. 75–105. Springer, 2018.

[St94] Stewart, William J: Introduction to the numerical solution of Markov chains. Princeton University Press, 1994.