

# BUSINESS PROCESS MODELED WITH BPMN AND CTL MODEL CHECKING

F. Ouazar, M.C. Boukala, M. Ioualalen

MOVEP, Computer Science Faculty, USTHB  
BP, 32 El-Alia, Algiers, Algeria

## ABSTRACT

*Despite the richness of the BPMN language and its advantages for the specification of business processes, it remains a semi-formal language that does not allow rigorous verification of the specifications produced with it, and does not offer any methodological support to cover the verification phase. Therefore, several works have been proposed with the aim of describing the semantics of the BPMN language by a mathematical formalism. In this paper we address the issue of verifying BPMN models with an approach based on model-checking, where we focus on soundness, fairness, and safety properties. Thus by having a business process modeled by BPMN, a formal semantics for BPMN models based on Kripke structure will be provided for a formal verification of correctness. The properties are expressed with CTL (Computation Tree Logic) formulas. At the end, the model checker NuSMV is used for the verification of the formula.*

## KEYWORDS

*Business process, model-checking, formal methods, temporal logic CTL, Kripke structure*

## 1. INTRODUCTION

During the last decade a lot of proposed works focused on business process modeling. The Business Process Modeling Notations (BPMN) has emerged as standard notation to express the business processes. It has been also used as a tool for expert analysis for decision making. This success is based on its simplicity of notations and its exhaustive expressiveness. Nevertheless, the modelling of these business processes relies generally on the human expertise and lack the formal semantics. It characterizes the BPMN to cause undesirable errors which can be classified into two categories, either syntactical or semantical.

Syntactical errors can be detected by several tools such as JBPM(2017). However, semantic errors remain undetected during the design time due to the lack of BPMN semantics which are ambiguous and not concise. Subsequently, the run-time behavior of a process should be analyzed before execution to achieve the complete verification, showing whether the process model fulfils important criteria and avoid the improper functioning of the process, which can cost financially expensive damages.

Therefore, to detect BPMN model semantic errors, it will be proposed in this paper, a formal semantics for BPMN models based on Kripke structure [5], and then check the validity of major properties (detect deadlock, livelock,...) expressed in Computation Tree Logic (CTL) formulae. This ensures fairness; safety and soundness of business process model and avoid any structural errors.

The rest of this paper is structured as follows, section 2 summarizes the preliminaries used to illustrate our approach, in section 3 we explain the main idea of our work, in section 4 we give some results of tests and verification, we give and discuss similar works in section 5 at the end we conclude this work and give some perspectives.

## 2. BASIC DEFINITIONS

The following sections, briefly explain the concepts and technical terms used in the proposed approach.

### 2.1. Business process

A business process is defined by the Workflow Management Coalition (WfMC) as: "A set of one or more procedures or related activities collectively achieving a business objective, generally in the context of an organizational structure defining functional roles and relationships" [1].

### 2.2. Business Process Modeling Notation

The BPMN is a standard for process modeling that defines a graphical notation; the main concepts modeled by BPMN are tasks (activities), events, flows, gateways and sub-processes, as shown in the following figure.

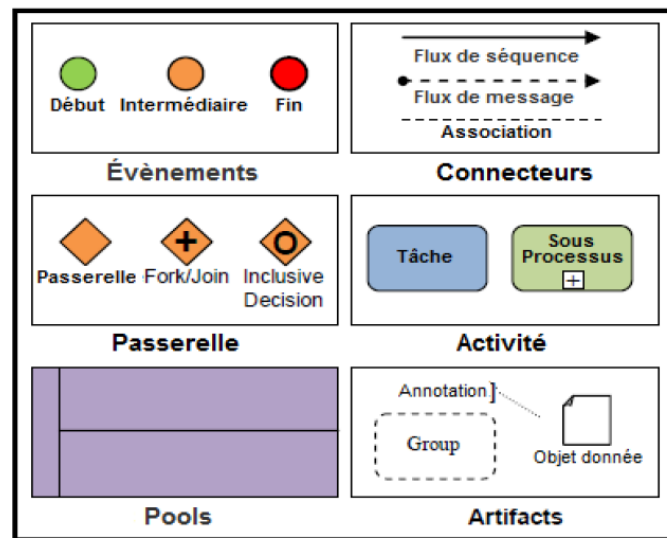


Figure 1. some BPMN elements

### 2.3. Model checking

Model checking [2] is a formal verification technique that determines whether given properties of a system are satisfied by a model. A model checker takes an abstraction of the behavior of the reactive system (a transition system) which can be a Petri net, Kripke structure, automata ... and a property expressed by some temporal logic such as CTL, LTL, TCTL... as inputs, and outputs either a claim that the property is true or a counterexample falsifying the property.

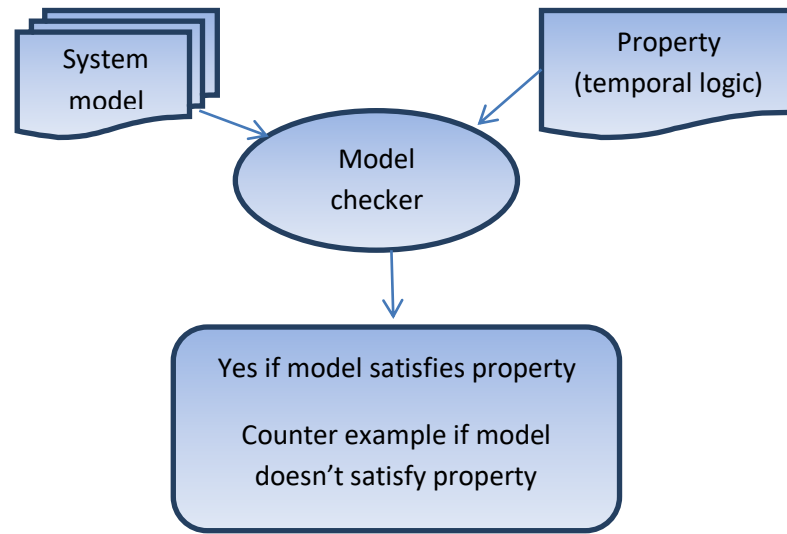


Figure 2. model-checking

## 2.4. Kripke structure

A kripke structure [3] is a variant of non-deterministic automata used in model-checking to represent the behavior of a system. It is considered as a structure, whose nodes represent the reachable states of the system and the arcs the state transitions. The labeling function labels each node with a set of atomic properties that must be valid in the corresponding state.

A Kripke structure on a set of propositional variables AP is a 4-tuple:

$M = (Q, I, R, L)$  such that :

- $Q$  is the finite set of states
- $I$  is the set of initial states such that  $I \subseteq Q$ .
- $R \subseteq Q \times Q$  is a transition relation that satisfies:  $\forall q \in Q, \exists q' \in Q$  such that  $(q, q') \in R$
- $L: Q \rightarrow 2^{AP}$  is a labeling function that defines for each state  $q \in Q$  a set  $L(q)$  of all atomic propositions holds in  $q$ .

## 2.5. Computation Tree Logic

CTL is a branching-time temporal logic, introduced by Clarke and Emerson [6]. The syntax of this language is composed of propositional logic to which are added the linear temporal operators which are essentially the potential  $F$  (possibly in the future), the  $X$  (next), the invariance  $G$  (always in the future) precedence  $U$  (until),..., it provides safe, precise and unambiguous temporal properties.

It is used to specify the usual properties of systems such as safety and liveness. Figure bellow shows CTL operators.

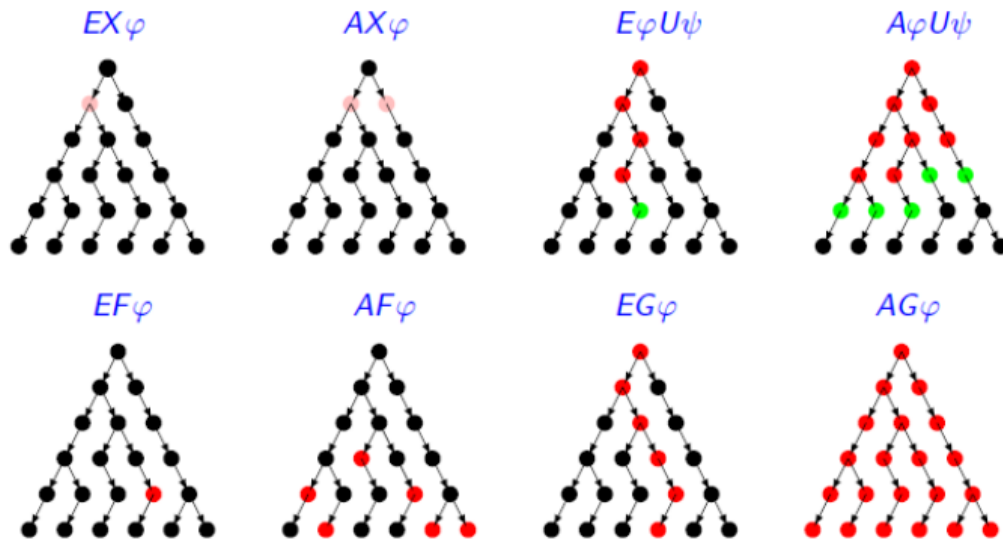


Figure 3. CTL operators

## 2.6. NuSMV model checker

NuSMV [4] is a software tool for the formal verification of finite state systems. It has been developed jointly by FBK-IRST and by Carnegie Mellon University.

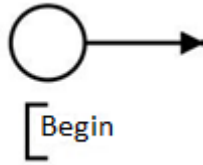

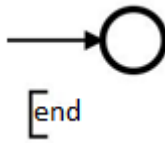
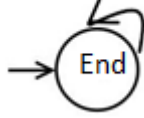

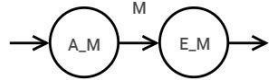
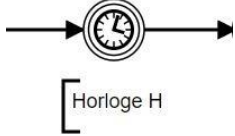
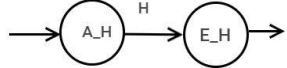
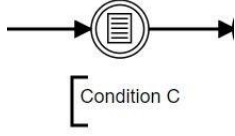
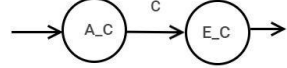
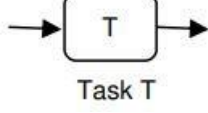
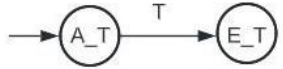
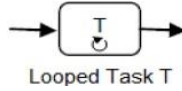
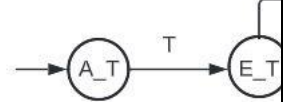
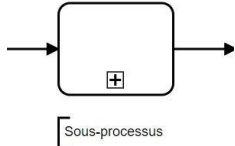
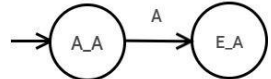
NuSMV allows checking finite state systems against specifications in the temporal logic CTL. The input language of NuSMV is designed to allow the description of finite state systems that range from completely synchronous to completely asynchronous. The NuSMV language (like the language of SMV) provides for modular hierarchical descriptions and for the definition of reusable components. The basic purpose of the NuSMV language is to describe (using expressions in propositional calculus) the transition relation of a finite Kripke structure. This provides a great deal of flexibility.

## 3. MODEL CHECKING FOR BUSINESS PROCESS

In this section we explain our approach to formally verifying the business process modeled by BPMN. In particular, the fairness and the soundness properties. The main idea is to map BPMN process model into a finite-states model (Kripke structure) to specify the system behavior and provide some CTL formulae that may be used by model checker NuSMV to verify the absence of structural errors and ensure the soundness of process model. Otherwise, it returns a counterexample. We defined the mapping rules from BPMN to KS, and we implemented a tool named BPMN2NuSMV to validate our approach.

### 3.1. Mapping rules from BPMN to KS

The table below shows the mapping rules from BPMN to Kripke structure

Definition	BPMN object	Kripke structure
<b>Start event:</b> marks the first step in a process.		
<b>End event:</b> marks last step in a process.		
<b>Message:</b> triggers the process, facilitates intermediate processes or complete the process.		
<b>Timer:</b> a single or recurring time or date triggers the process, facilitates intermediate processes, or completes the process.		
<b>Conditional:</b> a process starts or continues when a condition or business rule is met		
<b>Task:</b> a task is the most basic level of an activity, and it cannot be divided into simpler parts		
<b>Looped task:</b> is a task that repeats sequentially.		
<b>Sub-process:</b> a group of tasks that fit together well.		

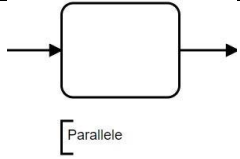
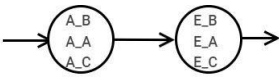
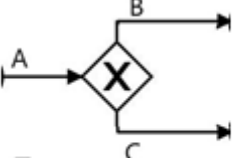
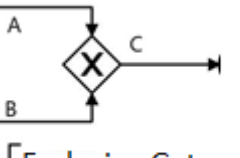
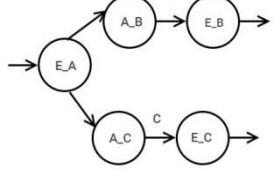
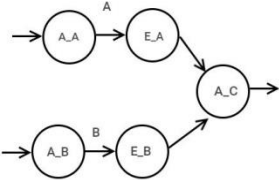
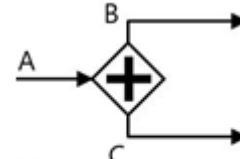
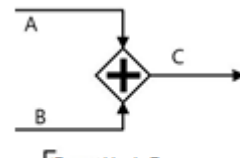
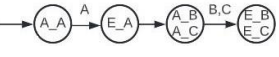
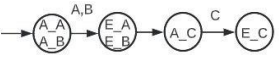
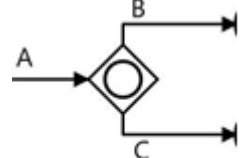
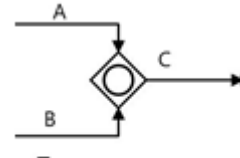
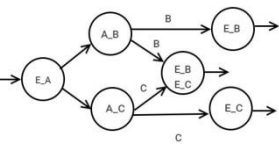
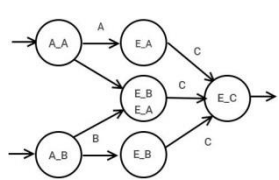
<p><b>A task with multiple instances:</b> is a task that occurs several times, these instances can occur in parallel or sequentially.</p>		
<p><b>Exclusive Gateway:</b> Evaluates the state of the business process and, depending on the case, separates the flow into one or more mutually exclusive paths</p>	<p><b>divergence</b></p>  <p><b>[Exclusive Gateway]</b></p> <p><b>convergence</b></p>  <p><b>[Exclusive Gateway]</b></p>	 
<p><b>Parallel Gateway:</b> this type of branching differs from the others in that it does not depend on conditions or events. Instead, parallel branches are used to represent two concurrent tasks in a business process</p>	<p><b>divergent</b></p>  <p><b>[Parallel Gateway]</b></p> <p><b>convergent</b></p>  <p><b>[Parallel Gateway]</b></p>	 
<p><b>Inclusive Gateway:</b> decompose or divide the process diagram into one or more processes</p>	<p><b>divergent</b></p>  <p><b>[inclusive gateway]</b></p> <p><b>convergent</b></p>  <p><b>[inclusive gateway]</b></p>	 

Table 1. mapping rules from BPMN to SK

Once the Kripke structure is obtained by applying the above mapping rules, we then proceed to define the desired correctness formulae.

### 3.2. CTL formulae generation:

The soundness of BPMN process model to avoid structural errors can be ensured by satisfying the following temporal properties and structural errors:

- Reachability: a state can be reached, we can express it with CTL formula as  $AFp$
- Safety: can be expressed with CTL as:  $AG!p$
- Deadlock free: can be expressed as :  $AG EX \text{ true}$
- Infinite Loop: A state from which it is possible to proceed but cannot reach the desired end state, in which case the system is locked in a small subset of states and does not progress.
- Multiple terminations: Corresponds to situations where there is a parallel gateway before an exclusive gateway, in this case only one sequence is traversed when the exclusive gateway is executed.
- Blockage: Corresponds to the case where some activities cannot continue to progress, even though they have not reached the end of the process ( $AF AG! \text{ end}$ ).

### 3.3. Tool and implementation

The finite state space  $M$  and the temporal logic formulae  $\phi$  are presented as input to a model checker. The model checker verifies whether temporal formula holds for that finite state  $M$  or not. As a result, it confirms the soundness of the process models. Otherwise, it returns a counterexample in cases of structural errors. We implemented a prototype tool named BPMN2NuSMV, so we used the model checker NuSMV to check CTL formulae, which takes as input a model represented by a Kripke structure. A specific language is used to describe this model. Once the model is completely described, the properties to be checked are added to the description and the model can then be submitted to the model checker NuSMV.

We tested our approach on several cases.

### 3.4. Case study

We further illustrate the mapping from BPMN process model to SK with NuSMV model, with the help of several examples, the given cases study consist of ATM machine, coffee dispenser and payment process, we will detail ATM machine case.

### 3.5. ATM Machine

The ATM accepts and holds the person's card while contacting the bank for the account information. Next, the screen prompts the user to enter their PIN code. Depending on the user input, three alternative actions are possible:

- (1) User enters the correct PIN code and can withdraw money,
- (2) Wrong PIN is entered - a message is displayed,
- (3) Operation is interrupted - an alarm signal is issued.

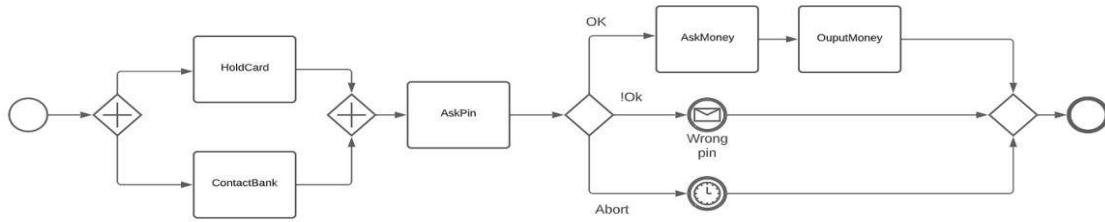


Figure 4. BPMN model of ATM machine

By using the mapping rules defined in table 1 we get the Kripke structure in the figure 4

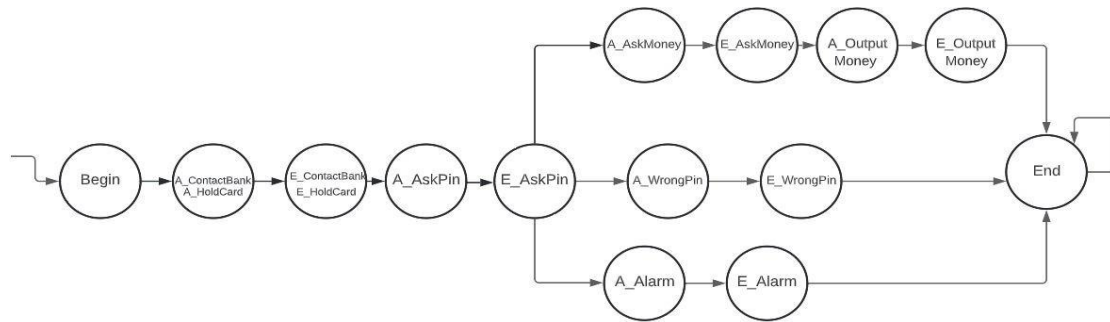


Figure 5. SK of ATM machine

### 3.6. SMV model

The first part of this code defines the Kripke structure, the state space of the SK is determined by the declaration of state variables as shown below:

```

1=MODULE main
2=VAR
3 state:{s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13};
4=ASSIGN
5 init(state) := begin;
6 next (state) := case
7 state = s0 : {s1};
8 state = s1 : {s2};
9 state = s2 : {s3};
10 state = s3 : {s4};
11 state = s4 : {s5,s9,s11};
12 state = s5 : {s6};
13 state = s6 : {s7};
14 state = s7 : {s8};
15 state = s9 : {s10};
16 state = s11 : {s12};
17 state = s8 : {s13};
18 state = s10 : {s13};
19 state = s12 : {s13};
20 state = s13 : {s13};
21 esac;

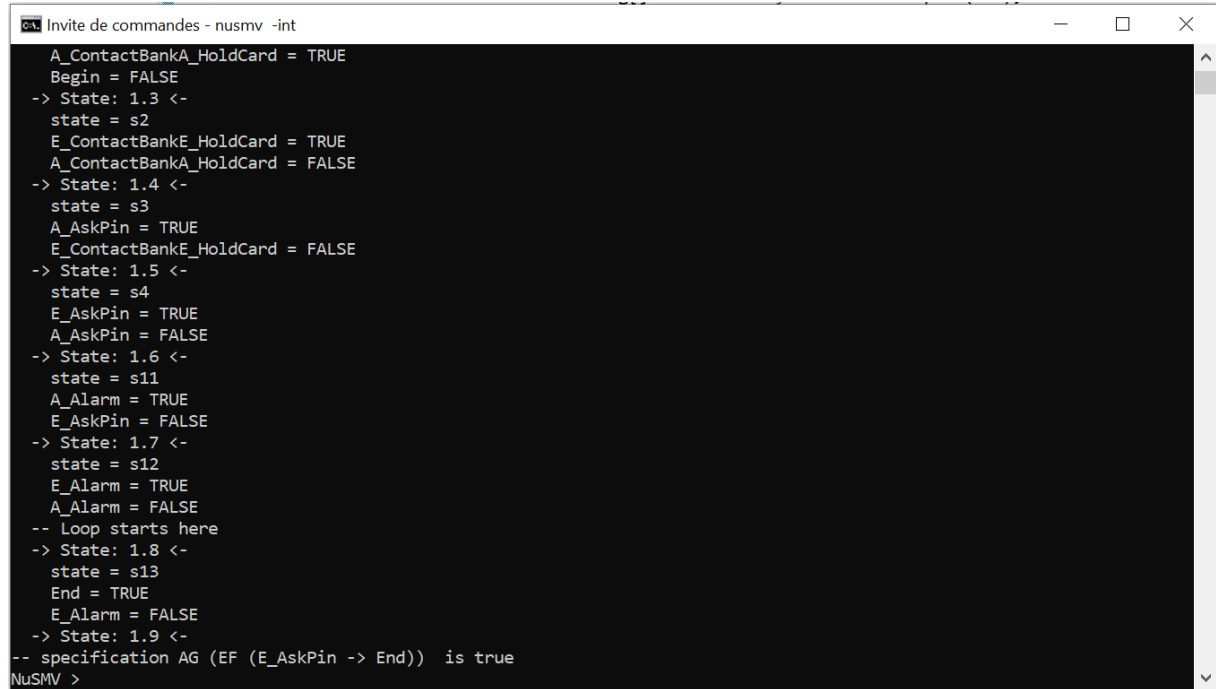
22=DEFINE
23 Begin := state = s0;
24 A_ContactBankA_HoldCard := state = s1;
25 E_ContactBankE_HoldCard := state = s2;
26 A_AskPin := state = s3;
27 E_AskPin := state = s4;
28 A_AskMoney := state = s5;
29 E_AskMoney := state = s6;
30 A_OuputMoney := state = s7;|
31 E_OuputMoney := state = s8;
32 A_WrongPin := state = s9;
33 E_WrongPin := state = s10;
34 A_Alarm := state = s11;
35 E_Alarm := state = s12;
36 End := state = s13;
37 SPEC AG (Begin -> AX A_ContactBankA_HoldCard );
38 SPEC AG (E_AskPin -> AX AF (A_AskMoney | A_OuputMoney | A_WrongPin ));
39 SPEC AG EF (E_AskPin -> End) ;
40

```



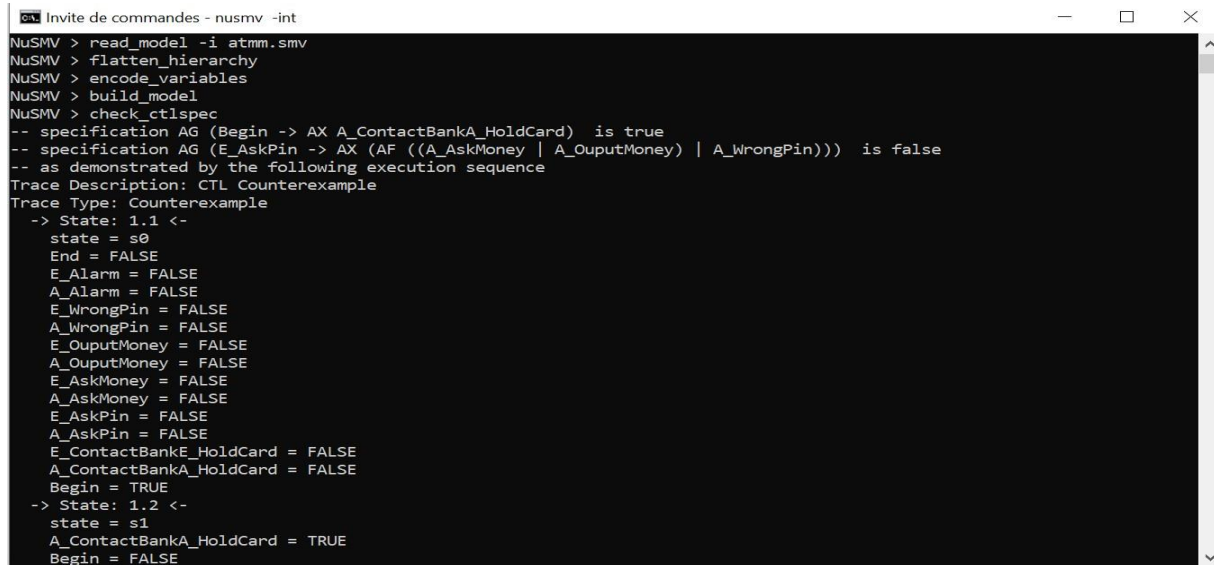
### 3.7. CTL properties check

We checked some CTL properties, related to ATM case, such as AG (Begin  $\rightarrow$  AX A\_ContactBank\_HoldCard), AG(A\_AskPin $\rightarrow$ AX AF (A\_AskMoney | A\_OutputMoney | A\_WrongPin)), and AG EF(E\_AskPin $\rightarrow$ End), then we get the following results:



```

Invite de commandes - nusmv -int
A_ContactBankA_HoldCard = TRUE
Begin = FALSE
-> State: 1.3 <-
state = s2
E_ContactBankE_HoldCard = TRUE
A_ContactBankA_HoldCard = FALSE
-> State: 1.4 <-
state = s3
A_AskPin = TRUE
E_ContactBankE_HoldCard = FALSE
-> State: 1.5 <-
state = s4
E_AskPin = TRUE
A_AskPin = FALSE
-> State: 1.6 <-
state = s11
A_Alarm = TRUE
E_AskPin = FALSE
-> State: 1.7 <-
state = s12
E_Alarm = TRUE
A_Alarm = FALSE
-- Loop starts here
-> State: 1.8 <-
state = s13
End = TRUE
E_Alarm = FALSE
-> State: 1.9 <-
-- specification AG (EF (E_AskPin -> End)) is true
NUSMV >
  
```



```

Invite de commandes - nusmv -int
NuSMV > read_model -i atmm.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > check_ctlspec
-- specification AG (Begin -> AX A.ContactBankA.HoldCard) is true
-- specification AG (E_AskPin -> AX (AF ((A_AskMoney | A_OutputMoney) | A_WrongPin))) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  state = s0
  End = FALSE
  E_Alarm = FALSE
  A_Alarm = FALSE
  E_WrongPin = FALSE
  A_WrongPin = FALSE
  E_OutputMoney = FALSE
  A_OutputMoney = FALSE
  E_AskMoney = FALSE
  A_AskMoney = FALSE
  E_AskPin = FALSE
  A_AskPin = FALSE
  E_ContactBankE_HoldCard = FALSE
  A_ContactBankA_HoldCard = FALSE
  Begin = TRUE
-> State: 1.2 <-
  state = s1
  A_ContactBankA_HoldCard = TRUE
  Begin = FALSE

```

#### 4. DISCUSSION AND RELATED WORK

The lack of formal semantic of BPMN model, has been addressed in several studies, based on formal models as Petri nets, Kripke structures...

Petri nets are widely used to describe and verify business processes. Indeed, several research works have exploited the strengths of these nets in order to verify the errors that a business process may contain such as the absence of blocking, liveness, etc. The idea consists of transforming business process models expressed in BPMN or BPEL into a Petri net using “translators” and then analyzing the latter using tools called “analyzers”.

Indeed, there are approaches and tools that make it possible to rewrite the specification of business processes in terms of Petri nets. Among these tools we can cite BPEL2PNML [7] or BPEL2oWFN [8] which automatically transforms a business process model expressed by the BPEL language into a Petri net expressed in PNML "Petri Net Markup Language". The BPEL2PNML tool generates a format that is accepted by the majority of RdP parsers.

Takemura in [9], shows how it is possible to verify the transaction and compensation mechanisms of BPMN models using Petri nets and therefore, how to apply the analysis of the reachability of these networks to these mechanisms of business process.

Works like those of Dijkman et al.[10], or even [11], propose to provide a formal semantics to BPMN models by transforming them into Petri nets and this while identifying a certain number of components of this BPMN notation which cannot have a match in Petri nets like multiple instances, exception handling for concurrent sub-process cases, etc. also, the authors in [12] propose to use colored Petri nets to provide formal semantics to BPMN models instead of classic Petri nets for more expressiveness.

In the work of [13], the authors propose an approach based on colored Petri nets to assess the feasibility of a BPMN model. Indeed, the proposed approach allows to analyze two properties on a BPMN model namely, blocking and infinite loops. For this, the authors propose a methodology consisting in manually translating the BPMN model into a modified representation of the BPEL4WS language, then into the XML representation of colored Petri nets (CPNXML) which

can be verified using the CPNTools tool. However, this work does not take into consideration the key elements of BPMN such as sub-processes, exceptions and cancellations.

In [14], [15] the authors propose an approach allowing the verification and the validation of a web services orchestration specified with BPEL to ensure its consistency and to avoid unexpected changes. This approach makes it possible to check a set of generic properties, such as liveness and safety, and specific ones using the SPIN model-checker with its PROMELA input language.

In [16], [17] the authors propose the use of model-checking to verify a set of rules in business processes modeled by EPC diagrams (Event-driven Process Chain).

In [18] the proposed approach consists in using the model-checking technique to check the consistency properties of the business processes modelled in BPMN, the BPMN model is translated into Kripke Structure (SK), then the SK is translated into PROMELA language, the property to be checked is expressed in LTL then use the SPIN model checker to check the correctness of the property by generating a counter-example in the opposite case. This work doesn't take in consideration some elements of BPMN as inclusive gateway.

Other works exist based on automata and process algebra, but Petri nets remain the most used models to describe and verify business processes.

In this work, we presented our approach, which is the translation of BPMN model to Kripke structure model by taking in consideration some missing elements in the other works, express the property to check with CTL language, then we used NuSMV model checker to check the correctness of the property and give a counter example in case the property not correct.

## 5. CONCLUSION

In this paper, we present our idea for business process model checking based on the mapping of BPMN model to Kripke structure to express the behaviour of the process models, the generated states of the Kripke structure are used to check the temporal properties, which are expressed with CTL logic, then we use NuSMV model checker to ensure the correctness of the specification (property).

We defined the mapping rules from BPMN to SK and we implemented BPMN2NuSMV prototype tool, we studied several cases such as ATM machine. The objective of our work is to give assistance for the process modellers to easily detect errors and correct them.

As future work, we intend to continue with business process verification, focus on fairness property and use modular Petri nets as formal model for the mapping of BPMN models.

## REFERENCES

- [1] BPMN : <http://www.bpmn.org/>
- [2] E.M. Clarke Jr., O. Grumberg, and D.A. Peled. , "Model Checking," The MIT Press, Cambridge, Massachusetts and London, UK, 1999.
- [3] Kripke, S. (1963). Semantical Considerations on Modal Logic. *Acta Philosophica Fennica* (16), 83-94.
- [4] <https://nusmv.fbk.eu/NuSMV/>
- [5] Browne, M. C., Clarke, E. M., & Grumberg, O. (1988). Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science - International Joint Conference on Theory and Practice of Software Development Volume 59 Issue 1-2*, 115-131.
- [6] E. M. Clarke, E. A. Emerson, and A. P. Sistla, 1986. Automatic verification of finite state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, pages p. 244\_263.

- [7] Hinz, S., Schmidt, K., & Stahl, C. (2005). Transforming BPEL to Petri Nets. The International Conference on Business Process Management (BPM'05) (pp. 220-235). Springer-Verlag.
- [8] Ouyang, C., Verbeek, E., van der Aalst, W. M., Breutel, S., Dumas, M., & ter Hofstede, A. H. (2005). WofBPEL: A Tool for Automated Analysis of BPEL Processes. The Third International Conference On Service-Oriented Computing - (ICSOC'05) (pp. 484-489). Amsterdam, Netherlands: Springer Berlin Heidelberg.
- [9] Takemura, T. (2008). Formal Semantics and Verification of BPMN Transaction and Compensation. The IEEE Asia-Pacific Services Computing Conference (APSCC '08), (pp. 284-290). Los Alamitos.
- [10] Dijkman, R. M., Dumas, M., & Ouyang, C. (2007). Formal Semantics and Analysis of BPMN Process Models using Petri Nets.
- [11] De Backer, M., & Snoeck, M. (2008). Business Process Verification: a Petri Net Approach. Belgium: Technical report, Catholic University of Leuven.
- [12] Ramadan, M., Elmongui, H. G., & Hassan, R. (2011). BPMN Formalisation using Coloured Petri Nets. The 2nd GSTF Annual International Conference on Software Engineering & Applications (SEA'11). Singapore, Singapore.
- [13] Ou-Yang C. and Y. D. Lin. BPMN-based business process model feasibility analysis : a Petri net approach. International Journal of Production Research, 46(14) :3763–3781, 2008.
- [14] Fisteus, J. A., Fernández, L. S., & Kloos, C. D. (2004). Formal Verification of BPEL4WS Business Collaborations. The 5th International Conference on E-Commerce and Web Technologies (EC-Web'04) (pp. 76-85). Zaragoza, Spain: Springer Berlin Heidelberg.
- [15] Nakajima, S. (2006). Model-Checking Behavioral Specification of BPEL Applications. Electronic Notes in Theoretical Computer Science (ENTCS) Vol.151 Issue 2, 89-105.
- [16] Feja, S., Speck, A., & Pulvermüller, E. (2009). Business Process Verification. GI-Jahrestagung, (pp. 4037-4051). Lübeck, Germany.
- [17] Speck, A., Feja, S., Witt, S., & Pulvermüller, E. (2011). Formalizing Business Process Specifications. Computer Science and Information Systems 2011 Vol. 8, Issue 2, 427-446.
- [18] O. Kherbouche, A. Ahmed, H. Besson, Using model checking to control the structural errors in BPMN models, 7th IEEE International Conference on Research Challenges in Information Science (RCIS), May 2013, Paris, France. 10.1109/RCIS.2013.6577723. hal-03108809

## AUTHORS

**Fatiha Ouazar Lounnaci** researcher teacher at the faculty of Computer Science at the university of science and technology Houari Boumediene for more than 10 years, she is preparing her doctoral thesis, her research domain is the formal verification of systems, she is member of Modelisation and performances evaluation of systems (MOVEP) Laboratory, she worked before in telecommunication company for more than seven years as team leader in VAS department where her main tasks were managing the team and developing VAS services.



**Mohand Cherif Boukala** lecturer at the faculty of Computer Science at the University of Science and Technology of Algiers (USTHB). He works on formal verification of systems, he is interested particularly in analysis time reduction using the distributed verification and modular verification. Furthermore, he also leans on checking compatibility components in finite and infinite systems, He is member Modelisation and performances evaluation of systems (MOVEP) Laboratory and the head of SVS team.



**Malika Ioualalen-Boukala** received her PhD thesis in computer Science at USTHB university, Algiers in 2000. She is full Professor from 2005, and she is currently the head of Computer Science faculty at USTHB, and the head of Modelisation and performances evaluation of systems (MOVEP) Laboratory. Her main research interests are in formal methods, Verification and Performance Evaluation of complex systems. She organized the 1st school in Modelisation and Verification of Complex Systems at USTHB, in March 2016. She serves as PC member of many conferences, as chair of International Symposium of Programming and Systems (ISPS'2011), and program committee chair of ISPS'2003. She has been the initiator of the workshop VECOSin2007, in Algeria.

