An Algebraic Method for Analysing Control Flow of BPMN Models

http://dx.doi.org/10.3991/ijes.v3i3.4862

Outman El Hichami[1], Mohamed Naoum[1], Mohammed Al Achhab[2],
Ismail Berrada[3] and Badr Eddine El Mohajir[1]
[1] The Faculty of Sciences, Tetouan, Morocco
[2] The National School of Applied Sciences, Tetouan, Morocco
[3] The Faculty of Sciences and Technology, Fez, Morocco

***Abstract*—This paper introduces an approach for formal verification of BPMN models. The incompatible constructs of the BPMN patterns can lead to wrong or incomplete semantics which resulting the behavioral errors such as deadlock and multiple termination. This research is motivated by the need to create a correct business process and in order to generate a more complete formalization of BPMN semantics than existing formalizations. We first introduce the chosen patterns which are the most used in the modelisation of the service-based business processes. Then, we illustrate a definition of the execution semantics of these patterns by using the rules of Max+ Algebra formulas, which have important benefits.**

***Index Terms*—Formal semantics, BPMN, Business process.**

I. INTRODUCTION

The Business Process Modeling Notation (BPMN) [1] is a standard notation for business process modeling. It presents an execution semantics of process instances that defines precisely how models in the BPMN notation should behave.

The BPMN models are composed of a set of activity nodes and a set of control nodes that can be connected by a flow relation. Others notations exist, for which we refer to a subset of BPMN related to control flow modelling in order to define a precise execution semantics of BPMN elements which are the most used in the modelisation of the service-based business processes.

The most challenging process modeling problem is to make it possible to create models with semantic errors. In fact, this modelisation based on process model (e.g., BPMN), and because the mix constructs in BPMN have an incompletely specified meaning, and the lack of an unambiguous denition of the BPMN notation can cause the behavioral errors.

Which business process model is correct is typically modeled with respect to several quality criteria. An important quality criterion is choosing an appropriate definition semantics of the patterns which are used in the modelisation of the business processes. Another model quality criterion is necessary to define a precise mapping between the adopted user-friendly notation and a formal language in order to support formal verification techniques. Therefore, several approaches have been proposed to the formal validation of BPMN [4],[5],[6],[7]. All these approaches are based on the mapping of BPMN to a formal presentation like Petri Nets [8], YAWL [9], PROMELA[1], and PNML [10] in order to use the formal analysis tools available for these models.

For illustrative purposes, we develop a complete execution semantics of BPMN patterns associated with control flow in terms of Max+ Algebra equations, which is a useful mathematical tool, to specify and evaluate the performance of interaction and interoperability in the processes composition. Max+ Algebra has emerged as the suitable mathematical structure to model the phenomena of synchronization, assembly, concurrency, and parallelism. It is dedicated to the analysis of systems properties whose behavior can be represented by linear equations. Consequently, our execution semantics covers more rules from the BPMN standard than any other formal semantics so far.

As already stated, the main motivation of our work is given by choice phenomena, synchronization, and concurrency in BPMN models. To manage these phenomena, especially where conflicts appear, the analytical behavior of the graphical model using Max+ Algebra system in order to arbitrate and resolve these conflicts is given. Therefore, certain errors such as deadlocks and multiple terminations in process models can be detected in the first phases of the business process modeling [2],[3] without having establishing all steps of model-checking [11].

The remainder of this paper is organized as follows. In section 2, we start by a related work of the used formalisms and their application domains. In section 3, a brief overview of BPMN standard and an abstract syntax of Max+ Algebra system is given. Analysis of execution semantics for BPMN elements related to control flow modelling are presented in section 4 and section 5. Section 6 concludes the paper and presents some perspectives.

II. RELATED WORK

Several current approaches are interested by modeling, analyzing, and evaluating the performance of business process. Furthermore, we mention that many of these researchers do not yet support all possible behavioral semantics of business process regarding the patterns related to control flow.

A variety of techniques define a formal semantics of BPMN [12],[13], which use Petri nets as the target formal model. However, Petri nets are limited in the semantics that they can represent. It is difficult to represent the inclusive and complex gateway. Such concepts can be represented in Max+ Algebra equations. Our work

[1] http://spinroot.com/spin/Man/

supports this claim by showing that the formalization of this paper is relatively complete.

Among the modeling tools used there are: Communicating Sequential Processes (CSP) [16]. In [17] a BPMN model is mapped to a set of CSP. The CSP models produced in this technique may be large and complex, and they do not preserve the structure of the BPMN model.

For an overview of business process modeling, we refer to [19], the authors introduce an approach, based on Business Process Execution Language (BPEL) [14], to formalize and verify BPMN. However, the types of verification problems for BPEL are different from those in BPMN. In particular, deadlocks and multiple terminations that may arise in BPMN models do not arise in BPEL systems.

When examining the BPMN process models described by [20], which rely on a mapping from a subset of BPMN to π-calculus. This tool can be used to check the soundness [21] of BPMN models. However, this mapping only covers a small subset of BPMN. For example, they do not deal with the patterns related to control flow. Whereas our approach can describe this phenomenon.

In this paper we use Max+ Algebra rules for which efficient analysis techniques are available for representing BPMN models where conflicts appear and to defining their execution semantics. Another benefit of using Max+ Algebra is their expressive power for studying and analyzing composed BPMN pattern.

III. PRELIMINARIES

In this section, we present concepts and definitions that will be used throughout the rest of the paper. We start with the BPMN that is used to modelize the business process. Then we give a brief description of Max+ Algebra, which is used to model the synchronization and parallelism phenomena of the service-based business processes in the form of linear equations.

A. Business process modeling notation (BPMN)

Before elaborating a formal semantics of BPMN, this section provides a gentle introduction to the BPMN elements related to control flow modelling that define the behavior of the processes and have an impact on the conflict situation (see Table I). Hence, three types of nodes named event, task, and gateway are considered as well as one type of edges called sequence flow. The main elements of BPMN include the following:

TABLE I.
THE BPMN ELEMENTS RELATED TO CONTROL FLOW MODELLING

Elements	Description
Events Start Event End Event	An event is represented with a circle and denotes something that happens during the cours of the process, affecting the process flow. This could be a start and end event.
Task	A task describes a type of work that has to be completed within a business process.
Sequence flow	A sequence flow is used to show the order in which particular activities will be performed in a process. It links two objects in a process diagram.
Default sequence flow	A default sequence flow is taken only if all the other outgoing sequence flows from the task or gateway are not valid.
Gateways Parallel Exclusive Inclusive Complex	Gateways are used to control how the sequence flows converge or diverge within a process. Some of the typical types of gateways are the following ones: 1. Parallel gateway: uses for synchronizing parallel flow without checking any conditions; each outgoing sequence flow receives a token [18] upon execution of this gateway. For incoming flows, the parallel gateway will wait for all incoming flows before triggering the flow through its outgoing sequence flows. 2. Exclusive gateway: is used to create alternative paths within a process flow. For a given instance of the process, only one of the paths can be taken. 3. Inclusive gateway: uses to create alternative but also parallel paths within a process flow. However, it should be designed so that at least one path is taken. 4. Complex gateway: uses to model complex synchronization behavior and to describe the precise behavior. For example, this expression could specify that tokens on three out of five incoming sequence flows are needed to activate the gateway.

B. Max+ Algebra

In Max+ Algebra, we work with the Max+ semi-ring which is the set $\mathbb{R}_{\max} = \{-\infty\} \cup \mathbb{R}$.

Scalar operations: Let a, b and $c \in \mathbb{R}_{\max}$. The operations maximum (implied by the max operator $\oplus$) and addition (plus operator $\otimes$) for these scalars are defined as:

- $a \oplus b = \max(a, b)$
- $a \otimes b = a + b$
- $\oplus$ is associative: $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
- $\oplus$ is commutative: $a \oplus b = b \oplus a$
- $\oplus$ admits a neutral element noted as ε: $a \oplus \varepsilon = a$
- $\otimes$ is associative: $(a \otimes b) \otimes c = a \otimes (b \otimes c)$
- $\otimes$ admits a neutral element noted as e: $a \otimes e = a$
- $\otimes$ is distributive over $\oplus$: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$
- ε is absorbing for $\otimes$: $\varepsilon \otimes a = a \otimes \varepsilon = \varepsilon$

In $\mathbb{R}_{\max}$: $\varepsilon = -\infty$, $e = 0$.

IV. FORMAL MODELS FOR BPMN USING MAX+ ALGEBRA

In this section, we show how to transform BPMN processes into Max+ Algebra equations. For purposes of this paper, we focus on the BPMN elements related to control flow modelling. The introduction of a token [18] facilitates the description of the behavior of conflicted system with algebraic formulas or linear representations. Furthermore, the execution semantics of BPMN elements under our consideration are the most used in the modelisation of the service-based business processes[2].

[2] For more details, see Section 4

In the next section we illustrate how to generate the Max+ Algebra model of the chosen patterns.

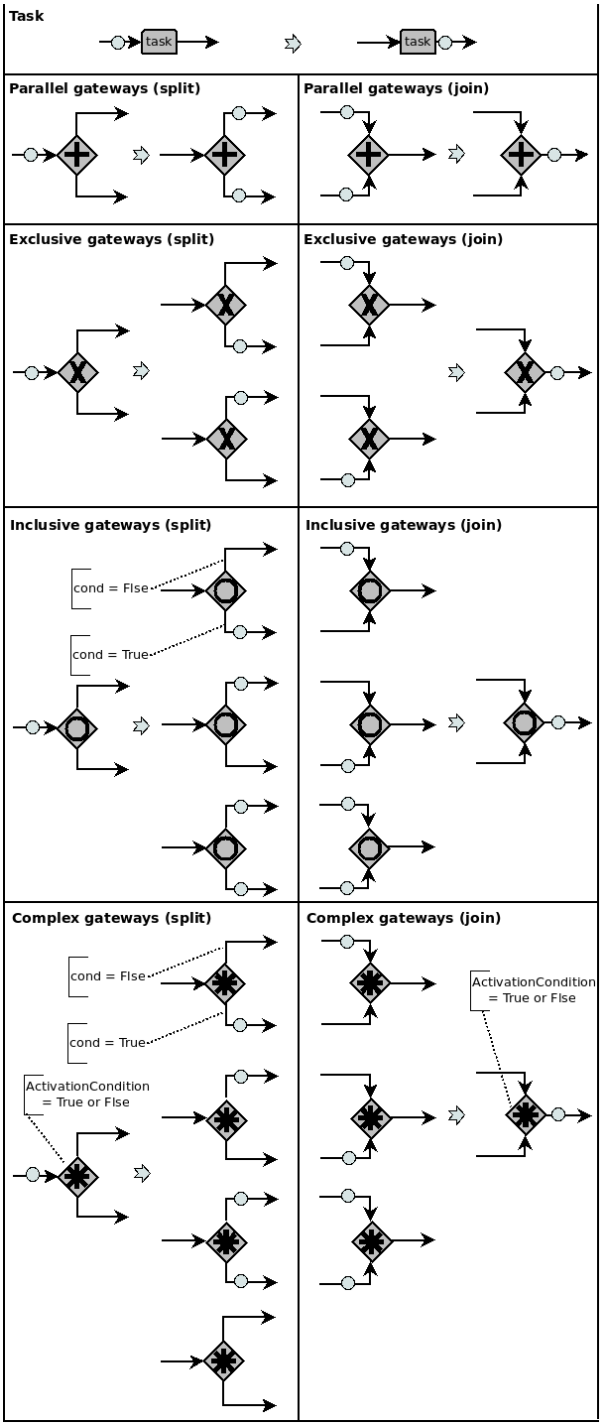

Figure 1. Execution semantics for BPMN elements

A. Cumulative application and firing condition

Before giving the Max+ Algebra model, let us define:

- The firing of a task occurs after the end of a time t_i associated to this task.
- To calculate the cumulative total at the firing of the task a_i, we define the following cumulative application that represents the date of k^{th} firing of the task a_i

$$a_i : \mathbb{N}^* \to \mathbb{R}_{max} \qquad (1)$$
$$k \to a_i(k), with : i \in \{0,1,\dots,|T|\}$$

Where |T| is the number of all tasks in the BPMN model.

Remark: When a token arrives at a task a_i, we note $a_i = 1$.

A sequence flow that has an exclusive or inclusive gateway as its source requires a condition to direct the flow. Consequently, we associate to each tasks a boolean variable that acts as a firing condition. A sequence flows is fired if this condition evaluates to *true*.

Formally, to express the firing condition related to the outgoing sequence flow connect to the task a_i, we define the following function:

$$Cond : T \to \{True, False\} \qquad (2)$$
$$a_i \to Cond(a_i)$$

B. Sequential pattern

Only task nodes are considered as sequential structure since they have exactly one incoming and one outgoing branch. This pattern is used to model dependencies between tasks so that one task cannot start before another is finished.

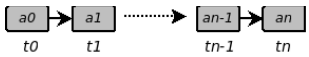

Figure 2. Sequence model

The analytical behavior of the model presented in Fig. 2 is given as follows:

$$\forall k \in \mathbb{N}^*, a_0(k) \neq \varepsilon,$$
$$\begin{cases} a_1(k) = t_1 \otimes a_0(k) \\ a_2(k) = t_2 \otimes a_1(k) \\ \dots = \dots \\ a_{n-1}(k) = t_{n-1} \otimes a_{n-2}(k) \\ a_n(k) = t_n \otimes a_{n-1}(k) \end{cases} \qquad (3)$$

The system (3) can be written in the following equation form:

$$\forall k \in \mathbb{N}^*, a_0(k) \neq \varepsilon,$$
$$a_{n(k)} = \left(\bigotimes_{i=1}^{n} t_i \right) \otimes a_0(k) \qquad (4)$$

C. Parallel gateway pattern

This pattern describes the synchronization phenomenon. It is used to synchronize multiple concurrent branches and to spawn new concurrent threads on parallel branches without checking any conditions. So that the gateway can not be fired if all previous branches are activated (see Fig. 3).

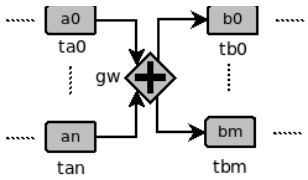

Figure 3. Parallel gateway pattern

The analytical behavior of this graphical model is given in system (5). All the tasks $b_0, \dots, b_m$ will be executed when a token arrives on its incoming sequence flow over all upstream tasks $a_0, \dots, a_n$:

$$\forall k \in \mathbb{N}^*, \forall a_i \in \bullet gw, \forall b_j \in gw \bullet, a_i(k) \neq \varepsilon,$$
$$\begin{cases} b_0(k) = t_0 \otimes (a_n(k) \oplus a_{n-1}(k) \dots a_0(k)) \\ b_1(k) = t_1 \otimes (a_n(k) \oplus a_{n-1}(k) \dots a_0(k)) \\ \dots = \dots \\ b_{m-1}(k) = t_{m-1} \otimes (a_n(k) \oplus a_{n-1}(k) \dots a_0(k)) \\ b_m(k) = t_m \otimes (a_n(k) \oplus a_{n-1}(k) \dots a_0(k)) \end{cases} \quad (5)$$

Where $gw \bullet$ is the set of all downstream tasks of gw and $\bullet gw$ is the set of all upstream tasks of gw.

The system (5) can be expressed as:

$$\forall k \in \mathbb{N}^*, \forall a_i \in \bullet gw, \forall b_j \in gw \bullet, a_i(k) \neq \varepsilon,$$
$$\begin{cases} b_0(k) = t_0 \otimes \left(\bigoplus_{i=0}^{n} a_{i(k)} \right) \\ b_0(k) = t_0 \otimes \left(\bigoplus_{i=0}^{n} a_{i(k)} \right) \\ \dots = \dots \\ b_{m-1}(k) = t_{m-1} \otimes \left(\bigoplus_{i=0}^{n} a_{i(k)} \right) \\ b_m(k) = t_m \otimes \left(\bigoplus_{i=0}^{n} a_{i(k)} \right) \end{cases} \quad (6)$$

D. Exclusive gateway pattern

The Fig. 4 describes an exclusive gateway model. In this pattern, *m* tasks are in conflict situation. When a token arrives at any incoming sequence flows, it can activate the gateway, but we don't know which downstream task will be fired. So, the task that will be fired can be chosen by the evaluation of the conditions in order. The first condition that evaluates to true determines the sequence flow the token is sent to. If and only if none of the conditions evaluates to true, the token is passed on the default sequence flow.

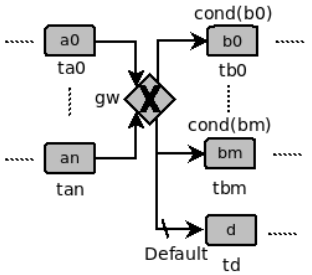

Figure 4. Exclusive gateway pattern

Furthermore, it is not obvious to formally express the firing of the downstream tasks. With the aim to describe this functioning by Max+ Algebra equations and in order to facilitate the mathematical analysis, we associate to each task the following function:

$$f : \mathbb{R}_{max} \to \{e, \varepsilon\}, \quad x \to f(x) \quad (7)$$

When only a task a_i is fired for the k^{th} firing (i.e.,$f(a_i(k)) = e$), all other tasks a_x (with $a_x \neq a_i$) are not fired (i.e., $f(a_x(k)) = \varepsilon$).

The behavior of the modeled exclusive gateway pattern is represented by the system (8):

$$\forall k \in \mathbb{N}^*,$$
$$\begin{cases} -\forall a_i \in \bullet gw, \forall b_j \in gw \bullet, \exists! a_i \in \bullet gw, \exists! b_j \in gw \bullet; \\ a_i(k) \neq \varepsilon, f\left(b_j(k)\right) = e \\ \Rightarrow b_{j,Cond(b_j)}(k) = \left(\left(t_{b_j} \otimes a_i(k) \right) \otimes f\left(b_j(k)\right) \right), \\ b_{j,\neg Cond(b_j)}(k) = \varepsilon \\ -\forall a_i \in \bullet gw, \forall b_j \in gw \bullet, \exists! a_i \in \bullet gw; \\ a_i(k) \neq \varepsilon, \neg Cond(b_j), f(d(k)) = e \\ \Rightarrow d(k) = \left(\left(t_d \otimes a_i(k) \right) \otimes f(d(k)) \right), b_j(k) = \varepsilon \end{cases} \quad (8)$$

E. Inclusive gateway pattern

As shown in Fig. 5, the inclusive gateway is activated if at least one incoming sequence flow has at least one token. In order to determine the outgoing sequence flows that receive a token, all conditions on the outgoing sequence flows are evaluated. The evaluation does not have to respect a certain order. If none of the conditions evaluates to true, the token is passed on the default sequence flow.

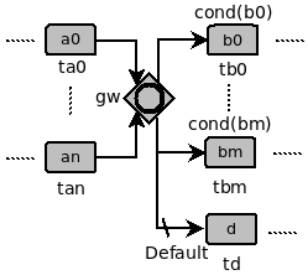

Figure 5. Inclusive gateway pattern

Using a standard formalization, this pattern may be expressed under the following form:

$$\forall k \in \mathbb{N}^*,$$
$$\begin{cases} -\forall a_i \in \bullet gw, \forall b_j \in gw \bullet, \exists! a_i \in \bullet gw, \exists! b_j \in gw \bullet; \\ a_i(k) \neq \varepsilon, f\left(b_j(k)\right) = e \\ \Rightarrow b_{j,Cond(b_j)}(k) = \left(t_{b_j} \otimes \bigoplus_{i=0, a_i = 1}^{n} a_{i(k)} \right) \otimes f\left(b_j(k)\right), \\ b_{j,\neg Cond(b_j)}(k) = \varepsilon \\ -\forall a_i \in \bullet gw, \forall b_j \in gw \bullet, \exists! a_i \in \bullet gw; \\ a_i(k) \neq \varepsilon, \neg Cond(b_j), f(d(k)) = e \\ \Rightarrow d(k) = \left(t_d \otimes \bigoplus_{i=0, a_i = 1}^{n} a_{i(k)} \right) \otimes f(d(k)), b_j(k) = \varepsilon \end{cases} \quad (9)$$

F. Complex gateway pattern

A complex gateway (see Fig. 6) can be used to describe the precise synchronization behavior. it has an attribute *ActivationCondition* that refers to the activation of incoming flows. For example, an *ActivationCondition* could be $(a_0 + a_1 + \ldots + a_n \geq Cte = 3)$ stating that it needs $(Cte = 3)$ out of the n incoming flow to have a token in order to proceed.

The complex gateway is in one of the two states: (represented by the attribute *WaitingForStart* = True) and waiting for reset (*WaitingForStart* = False). If it is waiting for start, then it waits for the *ActivationCondition* to become True. The *ActivationCondition* is not evaluated before there is at least one token on some incoming sequence flow.

When the *ActivationCondition* becomes True, the complex gateway uses the synchronization semantics of the split inclusive gateway. The gateway changes its state to waiting for reset (*WaitingForStart* = False).

When waiting for reset, the gateway waits for a token on each of those incoming sequence flows from which it has not yet received a token in the first phase. If tokens arrive later, those tokens cause a reset of the gateway.

When the gateway resets, it consumes a token from each incoming sequence flow that has a token and from which it had not yet consumed a token in the first phase. Then it uses the synchronization semantics of the split inclusive gateway. Finally, the gateway changes its state back to the state waiting for start.

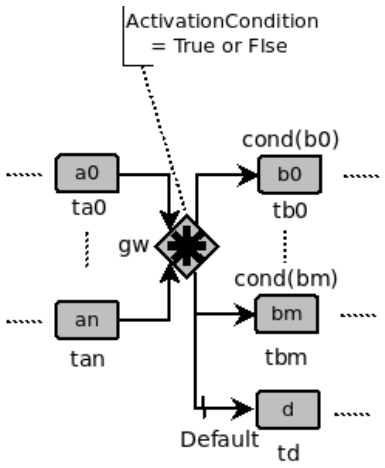

Figure 6. Complex gateway pattern

The behavior of the incoming sequence flows related to the complex gateway pattern is represented by the system (10).

$$\forall k \in \mathbb{N}^*,$$

$$\begin{cases} -WaitingForStart, \forall a_i \in \bullet gw, \forall b_j \in gw \bullet, \\ \exists! a_i \in \bullet gw, \exists! b_j \in gw \bullet;\ ActivationCondition, \\ a_i(k) \neq \varepsilon, f\left(b_j(k)\right) = e, Cond\left(b_j\right) \\ \Rightarrow b_{j,Cond(b_j)}(k) = \left(t_{b_j} \otimes \bigoplus_{i=0, a_i = 1}^{n} a_{i(k)}\right) \otimes f\left(b_j(k)\right), \\ \quad b_{j,\neg Cond(b_j)}(k) = \varepsilon, \neg WaitingForStart \\ -\neg WaitingForStart, \forall a_i \in \bullet gw, \forall b_j \in gw \bullet, \\ \exists! a_i \in \bullet gw, \exists! b_j \in gw \bullet, \exists Cte \in \mathbb{N}^*; \\ \quad ActivationCondition, a_i(k) \neq \varepsilon, f\left(b_j(k)\right) = e, \\ \quad Cond\left(b_j\right), \sum_{i=0}^{n} a_i \geq Cte \\ \quad \Rightarrow\ b_{j,Cond(b_j)}(k) - t_{b_j}, WaitingForStart \\ -WaitingForStart, \forall a_i \in \bullet gw, \forall b_j \in gw \bullet, \\ \exists! a_i \in \bullet gw; ActivationCondition, a_i(k) \neq \varepsilon, \\ f\left(d(k)\right) = e, \neg Cond\left(b_j\right) \\ \Rightarrow d(k) = \left(t_d \otimes \bigoplus_{i=0, a_i = 1}^{n} a_{i(k)}\right) \otimes f\left(d(k)\right), \\ \quad b_j(k) = \varepsilon \end{cases} \tag{10}$$

V. SEMANTIC ANALYSIS OF MAX+ ALGEBRA MODELS

To verify and to create the correct BPMN model in the early stage of the conception, the transformation into Max+ Algebra equations of the chosen patterns which are the most used in the modelisation of the service-based business processes is given.

Moreover, the interaction between the chosen patterns into one composed BPMN model can lead to create an incompatible or an ambiguous semantics in this model which resulting the behavioral errors such as deadlock, multiple termination and undesirable cyclic behavior.

In this section, we discuss the use of our proposed model for detecting the deadlock and multiple termination. In fact, the correct BPMN model contains only the compatible composed patterns.

As already mentioned, the execution semantics of BPMN patterns under our consideration is illustrated in Fig. 1. If a token is on one sequence flow, then the destination node for this sequence flow is ready to be triggered.

A. Deadlock Patterns

Deadlock patterns have already been identified by Onada et al. in [15]. Two concepts were behind these patterns. The first is reachability. Reachability between two nodes A and B (*A (resp. B) represents a task or a task flow*) in a process graph simply means that there is at least one path from A to B. The second is absolute transferability. This is a much stronger concept because it states that a token (work item) can always be transferred

from node A to all input points of node B. What makes absolute transferability reduce reachability between two nodes is the existence of routing control nodes in between.

In BPMN, deadlock occurs when a parallel gateway receives inputs which contain exclusive split. From the definition of exclusive connector and the definition of parallel gateway, the parallel gateway requires every incoming sequences to be processed. However, the exclusive split chooses only a single outgoing sequence to be processed. The example of deadlock is shown in Fig. 7.

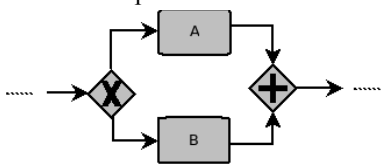

Figure 7. BPMN contains deadlock

Formally, the analytical behavior of the BPMN model presented in Fig. 7 is given as follows:

$$\begin{cases} A(k) = \varepsilon & or: A(k) \neq \varepsilon, with\ \forall k \geq 1 \\ B(k) \neq \varepsilon & B(k) = \varepsilon \end{cases} \quad (11)$$

On the contrary, to deal with this deadlock error, it is necessary that the earlier Max+ Algebra equations will be expressed as:

$$\begin{cases} A(k) \neq \varepsilon,\ with\ \forall k \geq 1 \\ B(k) \neq \varepsilon \end{cases} \quad (12)$$

Therefore, it is clearly that there is a deadlock in the merge parallel gateway.

Remark: Note that if the *ActivationCondition* in the case of the complex gateway never becomes true in the first phase (*WaitingForStart* = True), tokens are blocked indefinitely at the gateway, which causes a deadlock of the entire process model.

B. Multiple Termination Patterns

The multiple termination is the situation that there exists a parallel split before an exclusive gateway as shown in Fig. 8. Only one sequence is traversed when the exclusive gateway is executed. This leads to the violation of soundness criterion [21]. Some of the tasks are not terminated in one of predefined terminate process.

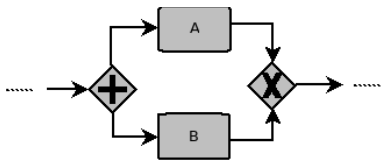

Figure 8. BPMN contains multiple termination

The analytical behavior of the scenario presented in Fig.8 is:

$$\begin{cases} A(k) \neq \varepsilon,\ with\ \forall k \geq 1 \\ B(k) \neq \varepsilon \end{cases} \quad (13)$$

And in order to activate the merge parallel gateway, it is necessary that the cumulative applications are expressed by this system:

$$\begin{cases} A(k) = \varepsilon & or: A(k) = \varepsilon, with\ \forall k \geq 1 \\ B(k) \neq \varepsilon & B(k) \neq \varepsilon \end{cases} \quad (14)$$

VI. CONCLUSION

The BPMN standard is a graphical notation that describes the logic of steps in a business process, but it is ambiguous and inconsistent when it comes to defining their semantics. The lack of formal semantics of BPMN patterns related to control flow motivates the works in this area for checking the correctness of BPMN models from a semantic perspective.

This paper deals with the development of a theory and a generic method to model and analyze business process with conflicts in Max+ Algebra. This method allows to arbitrate these conflicts by given the corresponding linear equations of the chosen BPMN patterns which are the most used in the modelisation of the service-based business processes.

In future work, we plan to adapt the proposed approach with our previous works [4], [7] so that to develop a plug-in which can integrate the formal verification techniques of business processes in the design phase.

REFERENCES

[1] OMG. Business Process Modeling Notation (BPMN) Version 2.0. OMG Final Adopted Specification. Object Management Group, 2011.

[2] W. van der Aalst. , "Workflow verification Finding control-flow errors using petri-net-based techniques," *Business Process Management*, pp. 19-128, 2000. http://dx.doi.org/10.1007/3-540-45594-9_11

[3] N. Russell, A. ter Hofstede, D. Edmond, and W. van der Aalst: Workflow data patterns, "Identification, representation and tool support," *Conceptual Modeling-ER 2005*, pp. 353-368, 2005. http://dx.doi.org/10.1007/11568322_23

[4] O. El Hichami, M. Al Achhab, I. Berrada and B. El Mohajir, "Graphical specification and automatic verification of business process," *the International Conference on Networked systems, (NETYS 2014), LNCS 8593, Springer,* pp. 341-346, 2014. http://dx.doi.org/10.1007/978-3-319-09581-3_27

[5] W. van der Aalst and B.F. van Dongen , "Discovering Petri Nets From Event Logs," pp. 372-422. *Springer-Verlag, Berlin*, 2013. http://dx.doi.org/10.1007/978-3-642-38143-0_10

[6] Dirk Fahland, Cadric Favre, Jana Koehler, Niels Lohmann, Hagen Volzer, Karsten Wolf, "Analysis on demand: Instantaneous soundness checking of industrial business process models," *Data Knowl. Eng.* 70(5): pp.448-466, 2011. http://dx.doi.org/10.1016/j.datak.2011.01.004

[7] O. El Hichami, M. Al Achhab, I. Berrada, and B. El Mohajir: , "Visual specification language and automatic checking of business process," *8th International Workshop on Verification and Evaluation of Computer and Communication Systems (VECoS 2014)*, CEUR Workshop Proceedings, Vol.1256, pp. 93-101, Bejaia, Algeria, September 29-30, 2014.

[8] T. Murata and J.Y. Koh, "Petri nets: Properties, Analysis and Applications," an invited surve y paper, Proceedings of the *IEEE*, Vol.77, No.4 pp.541-580, 1989. http://dx.doi.org/10.1109/5.24143

[9] J.-H. Ye, S.-X. Sun, L. Wen, and W. Song, "Transformation of BPMN to YAWL," In *CSSE* (2), pp. 354-359. *IEEE Computer Society*, 2008. http://dx.doi.org/10.1109/csse.2008.980

[10] L. Hillah, F. Kordon, L. Petrucci, and N. Trves, "PNML Framework: an extendable reference implementation of the Petri Net Markup Language," Petri Nets, *LNCS 6128*, pp. 318--327, 2010. http://dx.doi.org/10.1007/978-3-642-13675-7_20

[11] G. Holzman , "The Spin Model Checker: Primer and Reference Manual," *Addison-Wesley*, 2004.

[12] R. M. Dijkman, M. Dumas, and C Ouyang, "Formal semantics and analysis of BPMN process models using Petri nets," *Technical Report 7115*, Queensland University of Technology, Brisbane, 2007.

[13] R. M. Dijkman, M. Dumas, and C. Ouyang, "Semantics and analysis of business process models in BPMN," Inf. *Softw.*

Technol, vol. 50, no. 12, pp. 1281-1294, 2008. http://dx.doi.org/10.1016/j.infsof.2008.02.006

[14] A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, C. K. Liu, S. Thatte, P. Yendluri, and A. Yiu, "Web Services Business Process Execution Language Version 2.0," Committee Draft. *WS-BPEL TC OASIS*, 2005.

[15] S. Onoda, Y. Ikkai, T. Kobayashi, and N. Komoda, "Definition of deadlock patterns for business processes workflow models," In *HICSS99*: Proceedings of the Thirtysecond Annual Hawaii International Conference on System Sciences-Volume 5, pages 50-65, Washington, DC, USA, IEEE Computer Society, 1999. http://dx.doi.org/10.1109/hicss.1999.772966

[16] C. A. R. Hoare, "Communicating sequential processes. Commun," ACM, vol. 21, pp. 666-677, 1978.

[17] P. Y. H. Wong and J. Gibbons, "A Process Semantics for BPMN,"Lecture Notes in Computer Science. *Springer Berlin Heidelberg*, vol. 5256, ch. 22, pp. 355â€"374, 2008.

[18] Christiansen, DR, Carbone, M. and Hildebrandt, T. , "Formal semantics and implementation of BPMN 2.0 inclusive gateways", in Proc. of the 7th international conference on Web services and formal methods, ser. Web Services and Formal Methods 2010, Berlin, Heidelberg: *Springer-Verlag*, 146-160, 2011. http://dx.doi.org/10.1007/978-3-642-19589-1_10

[19] C. Ouyang, H.M.W. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas, and A.H.M. ter Hofstede, "Formal semantics and analysis of control flow in WS-BPEL". Science of Computer Programming, 67(2-3):162-198, *Elsevier*, 2007.

[20] F. Puhlmann and Matthias Weske, "Investigations on Soundness Regarding Lazy Activities". In Proceedings of the International Conference on Business Process Management (*BPM'2006*), volume 4102 of Lecture Notes in Computer Science, pages 145-160. Springer, 2006.

[21] Fahland, D., Favre, C., Koehler, J., Lohmann, N., Volzer, H.,Wolf, K., "Analysis on demand: Instantaneous soundness checking of industrial business process models". *Data Knowl. Eng*. 70(5), 448-466, 2011. http://dx.doi.org/10.1016/j.datak.2011.01.004

AUTHORS

Outman El Hichami is with the Faculty of Sciences, Tetouan, Morocco.

Mohamed Naoum is with the Faculty of Sciences, Tetouan, Morocco.

Mohammed Al Achhab is with the National School of Applied Sciences, Tetouan, Morocco.

Ismail Berrada is with the Faculty of Sciences and Technology, Fez, Morocco.

Badr Eddine El Mohajir is with the Faculty of Sciences, Tetouan, Morocco.

Submitted 14 July 2015. Published as resubmitted by the authors 10 October 2015.