Formalization of BPMN Gateways using the DD-LOTOS Formal Language

Toufik Messaoud Maarouk, Mohammed El Habib Souidi, Makhlouf Ledmi, and Samra Sabeg

***Abstract*—Business Process Model and Notation (BPMN), is a standardized graphical language used for the graphical modeling of business processes. A BPMN model is composed of several small graphs called elements; these elements make it possible to describe the activities, the events, and the interactions between the components of a business process. Among the essential elements of BPMN are gateways, which control the flow of data. However, the big challenge of these gateways is the existence of several interpretations of the same BPMN model containing gateways; this is due to the informal and ambiguous definition. Several works have proposed the formalization of gateways using formal languages such as process algebras, Petri nets, etc. The purpose of this article is to propose a formalization of BPMN gateways using the formal language DD-LOTOS. DD-LOTOS is defined on a semantics of true parallelism called maximality semantics and allows to support distribution and temporal constraints. We then propose the verification of certain properties using the UPPAAL model checker. Our approach has been validated through a case study representing the online purchasing process.**

***Index Terms*—BPMN, business process, Gateways, formal semantics, true concurrency, DD-LOTOS.**

I. Introduction

The BPMN notation [1] is considered the most adopted language in modeling business processes in an organization [2]. Through its basic graphic elements, BPMN offers an intuitive approach to modeling complex business processes.

BPMN provides several types of diagrams designed to describe, automate, and enhance business processes at various levels of implementation. Among them, the collaboration diagram facilitates interaction among different stakeholders in the process, while an extended version, known as the choreography diagram, is utilized to coordinate interactions.

Today, BPMN is used in many different fields, including healthcare, finance, and industry. In the healthcare sector, works [3–5] have shown how BPMN can enhance the quality of clinical care. Likewise, in finance, research [6, 7] showcases how organizations employ BPMN to deliver high-quality services at a reduced cost.

The interpretation of BPMN diagrams is defined by the meaning of each graphical element used. The BPMN 2.0 specification [1, 8] provides examples illustrating the use of graphical elements, with detailed explanations. These examples show how the elements and diagram construction rules can be applied in practice to model a process. The BPMN 2.0 specification defines the semantic concepts used in processes, associating them with graphical elements. It also aims to resolve inconsistencies and ambiguities identified in the diagrams of the previous BPMN 1.2 specification [1].

Moreover, the BPMN 2.0 specification defines the conformance property by defining the execution semantics of each element. This serves to ensure consistent use of BPMN diagrams [1]. The behavioral semantics of the diagrams are described informally through the use of the token concept, which traverses the Sequence flows elements and passes through the process elements.

Despite its widespread use in organizations, BPMN notation suffers from significant shortcomings. The primary challenge lies in the lack of rigorous and precise semantics for these diagrams. Although BPMN is defined using graphical syntax, there is ambiguity in the interpretation of these diagrams, especially when they include gateways [9, 10].

Gateways pose many semantic problems, and sometimes their behavior is ambiguous, so the interpretation of these gateways is often quite complex. Hence, the interest in proposing a rigorous semantics that can eliminate all the ambiguities associated with these gateways.

Several works [11–18] have proposed the formalization and verification of certain formal properties to ensure the consistency and smooth functioning of BPMN diagrams. Most of the works [11–15] propose a mapping with Petri nets and their extensions. Others propose a mapping with the formal languages Maude [16, 19] and the CSP process algebra [18]. Verified properties [18, 20–24] encompass various aspects such as deadlocks, coverability, accessibility, safeness, soundness, compatibility, equivalence, preservation, etc. Other works [9, 10, 25–32] focus solely on transforming BPMN gateways into formal languages.

In a previous work [33], we proposed a formal semantics for a subset of BPMN elements using the DD-LOTOS [34] formal language. The choice of DD-LOTOS is motivated by the following considerations:

1) DD-LOTOS is a language that explicitly supports the distributed aspect of applications.
2) DD-LOTOS is defined using a semantic of true parallelism known as maximality semantics [35].
3) As time is a key concept for every business process within

Manuscript received October 2, 2023; revised October 30, 2023. Date of publication November 29, 2023. Date of current version November 29, 2023. The associate editor prof. Tihana Galinac Grbac has been coordinating the review of this manuscript and approved it for publication.

Authors are with the Department of Mathematics and Computer Science, ICOSI Lab, University Abbes Laghrour Khenchela, BP 1252 EL Houria, Algeria (e-mails: {maarouk.toufik; souidi.mohammed; ledmi.makhlouf; samra.sabeg}@univ-khenchela.dz).

Digital Object Identifier (DOI): 10.24138/jcomss-2023-0138

an organization, e.g., processes have to meet certain deadlines and coordination between process tasks has to be achieved, DD-LOTOS takes the temporal aspect into account.

In this work, we focus on the formalization of BPMN gateways, in particular the inclusive gateway, the exclusive gateway, and the parallel gateway. We propose a mapping of collaboration diagrams containing gateways in terms of the DD-LOTS specification. Once the specification is generated, we can formally verify certain properties, such as deadlock, using the UPPAAL model checker. The formal verification approach is described in our work [36].

The remainder of this document is structured as follows: We review related work in Section II. We explain BPMN notation and the DD-LOTOS formal language in Section III. Section IV focuses on interpreting of BPMN gateways into DD-LOTOS behavioral expressions. The case study is presented in Section V. The paper ends with a conclusion and future work.

II. Related Works

In recent years, research on formalizing business processes has gained significant momentum. Although BPMN has become a standard in business process modeling, it suffers from a lack of formal semantics, leading to problems of inconsistency and ambiguity in models.

In this context, [32] proposed an operational semantics for BPMN collaboration diagrams in terms of labeled transition systems (LTS). The authors began by defining syntax in BNF style, comprising collaborations, processes, and gateways. The semantics were defined in terms of marked collaborations, focusing on xor-split, and-split, or-split, xor-join, and and-join gateways. The authors chose not to consider the or-join gateway for this work.

In [31], the authors defined a workflow graph to express collaboration, where the nodes of the graph represent activities, forks, and-join, splits, merges, or-join, or-split, and the two start and end events. There is a single start node and a single end node in a workflow graph. The end, activity, split, or-split, and fork nodes allow only one incoming edge. The start, activity, merge, or-join, and and-join nodes allow only one outgoing edge. The semantics is defined by the distribution of tokens on the edges. They also proposed new definitions for three key properties: soundness, completeness, and liberty, with a focus on supporting the or-join gateway. Soundness guarantees the absence of deadlock and a lack of synchronization.

In [30], the authors proposed an abstract state machine representing a business process as a graph. Nodes represent BPMN workflow objects (activities, events, and gateways). Arcs determine the order in which nodes are visited. During visits to nodes, a set of associated rules are executed. These rules describe the meaning of constructing the workflow for each node. Thus, the language defined is represented by the set of such rules, called rule schemes. The rules define how a node is activated, i.e., the formula for the number of tokens on incoming arcs needed for the node to be activated. This determines the number of tokens consumed on incoming edges and the number of tokens produced on outgoing edges. The authors also provided rules that include a mechanism for selecting the subset of outgoing arcs for or-split semantics. In the case of the or-join gateway, they discussed two proposals: one for acyclic graphs and another for graphs with cycles, albeit with certain restrictions. The work also checks some properties, including deadlock in acyclic workflow diagrams and deadlock in workflow diagrams with cycles, subject to specific constraints.

In [9], the authors proposed a formal semantics for the latest version of the BPMN notation, namely 2.0, and considered the properties of soundness and safety. The authors introduced two semantics: local and global. They also provided proofs of the equivalence between these two semantics. The local semantics consider the marking states of incoming edges and outgoing edges. The BPMN elements considered in this study are the two events start and end, the gateways and, xor, and or. The authors only consider safe BPMN models.

In [29], the author exposes two problems with the formalization of the or-join gateway. The first one concerns the self-reference of the or-join gateway. This self-reference is defined in the informal semantics of the gateway, which can generate a system whose exploration of the state space is exponential. The second problem concerns cyclic graphs or vicious circles. The author proposes a semantics and defines a new workflow graph class named separable graphs, which contains well-structured graphs and is a sound acyclic graph, generating a state-space based on it.

For the problem of vicious circles, the author proposes a static analysis based on graph-based approaches. The author proposes to prove the property of soundness.

In [28], the authors proposed a formal semantics for or-join, enabling the determination of when an or-join is activated. Two rules are defined: the enablement rule and the firing rule. The enablement rule decides if the object is ready to fire; the firing rule determines the result of the object's enablement. To implement the semantics defined, the authors propose an algorithm to determine the activated or-join. The naive version of the algorithm suffers from the problem of the combinatorial explosion of the state species. To overcome this problem, the authors propose a version based on or-Join's activation conditions; the algorithm's complexity is reduced to linear complexity.

In [10], the authors proposed formal semantics for a subset of the BPMN notation. The objects concerned are sequence flow, start events, end events, exclusive gateways, and inclusive gateways. This subset is called BPMN$_{inc}$. The semantics used is defined by the notion of process graphs and is implemented in two steps. The first step is to annotate each sequence flow with paths containing tokens. The second step uses the first step to determine the inclusive gateways that can be fired.

In [27], the authors proposed an algorithm to determine if an or-join element can be activated in a given state in the BPMN diagram. The originality of the proposed approach lies in the complexity of the algorithm, which is a constant time complexity.

In [26], the authors defined two constructs, namely the dynamic skip and the dynamic block. The skip dynamic construct allows one or more tasks to be skipped along a control flow path. The dynamic block construct blocks a component persistently. The authors have proposed a purely local semantics for inclusive gateways; this semantics is expressed using parallel gateways combined with the dynamic block construction. The new semantics does not support vicious circles.

In [25], the authors proposed a transformation approach and a framework for mapping BPMN elements into colored Petri nets. The mapping is defined by partitioning the BPMN model into BPMN partitions. This partitioning technique reduces the complexity of the BPMN model for possible application of formal verification techniques.

Table I provides a comparative analysis of the studies discussed in this work, using specific evaluation criteria. The first criterion under consideration is the model employed for the formalization of BPMN gateways, encompassing Labeled Transition Systems, Abstract State Machines, Process Graphs, Colored Petri Nets, and various others. Other approaches focus on rules governing nodes and the token distribution.

The second criterion for comparison pertains to the types of gateways addressed in these studies. The third criterion examines the formal properties validated, including soundness, safety, deadlock, and completeness.

III. The Basic Languages

A. BPMN language

BPMN notation offers several types of gateways[1]; the most used are the inclusive gateway, the exclusive gateway, and the parallel gateway. Table II presents in the graphical form of BPMN gateways.

The parallel gateway allows to continue the flow on a set of parallel paths or to synchronize several parallel paths. The exclusive gateway allows flows to be split into multiple paths based on an information-based condition or reconstruct a single path.

It is difficult to give a precise interpretation of the behavior of the inclusive gateway (or-join). This is due to the complex and ambiguous definition of or-join. This gateway enables the creation of both alternative and parallel paths. According to [1], all conditions are evaluated, and paths with a true condition are taken into account. A default path may be identified if no condition is evaluated as true. If this path is not specified, a runtime exception occurs.

Event-type graphic elements are used to describe something that happens, such as the start event of a diagram, the end event of a diagram, sending and receiving messages, the timer event, the conditional event, etc. Table III provides the essential BPMN events.

BPMN offers other elements used in business process modeling such as tasks, connection objects and sub-processes.

B. Distributed D-LOTOS Language

DD-LOTOS [34] is a formal language defined based on true concurrency semantics known as maximality semantics [35]. It is designed to support various aspects of real-time distributed systems. In DD-LOTOS, operators such as restriction, latency, and delay facilitate the specification of real-time systems. The concept of locality or site is integral to defining the distributed nature of this language.

1) Syntax: The DD-LOTOS syntax consists of two parts. The first part gives the rules of production of the behavioral expressions DD-LOTOS, and the second part expresses the composition of the systems. Table IV presents the syntax of DD-LOTOS.

2) Semantics:

Definition 3.1: The informal semantics of two syntactic categories is given by :

- The expression $a\{d\}$ represents the temporal restriction, the beginning of the action a is limited by the temporal interval $[0, d]$. The expression $\Delta^d E$ represents the delay operator, which means that the expression E starts after the flow of d. In $g@t[SP]$;E, t is a temporal variable and SP is a logical predicate;
- The operators of the theory of concurrency: the interiorization $hide\, L\, in\, E$, non-deterministic choice $E[]E$, parallel composition $E|[L]|E$, sequential composition $E \gg E$, and preemption $E\,[> E$;
- The expression $a!v\{d\}; E$ expresses the emission of the message v on the gate a, this emission is limited by the temporal interval $[0, d]$. The expression $a?xE$ expresses the receipt of a message on the gate a;
- A system can be empty, a composition of two systems or a behavioral expression E defined in a locality l.

Definition 3.2: The actions in global system are:

- Set of communication actions between localities are messages emission or reception through a communication channel $Act_{com} ::= a!m \mid a?x \mid \tau$ (output actions, input actions and the silent action);
- $\mathcal{G}$ a set of observable actions,
- $\delta \notin \mathcal{G}$ is the successful termination action,
- Set $Act = \mathcal{G} \cup \{i, \delta\}$.

Definition 3.3: The set $\mathcal{L}$ ranged over by l, denotes set of localities. ϑ an infinite set of channels defined by users ranged over by a,b,... channels are used for communication messages between localities.

3) Structured Operational Semantics: The operational semantics of behaviors are given by the operational semantics of D-LOTOS. This semantics is extended to DD-LOTOS by giving the semantic rules for communicated systems as follows:

Process $a!v\{d\}; E$ In the configuration ${}_{M}[a!v\{d\}; E]$, the sending of the message v starts only if all the actions in the set M terminate. In the rule 1 below, the condition $Wait(M) = false$, which means that there are actions that have not completed their executions. Rules 2 and 3 express a passage of time. Rule 4 expresses that the sending action must respect the temporal restriction operator, otherwise it is transformed to $Stop$.

1) $$\frac{\neg Wait(M)}{{}_{M}[a!v\{d\};E] \xrightarrow{M^{a!v}x} {}_{\{x:a!v:t\}}[E]} \qquad x = get(\mathcal{M})$$

2) $$\frac{Wait(M^{d'}) or (\neg Wait(M^{d'}) and\, \forall \varepsilon > 0.\ Wait(M^{d'-\varepsilon}))\ d' > 0}{{}_{M}[a!v\{d\};E] \xrightarrow{d'} {}_{M^{d'}}[a!v\{d\};E]}$$

TABLE I
COMPARISON OF WORKS ON BPMN FORMALIZATION.

Paper	Model used	Considered BPMN gateways	Verified formal properties
[32]	Labeled Transition Systems	Xor-split, And-split, Or-split, Xor-join, and And-join	/
[31]	Workflow Graph	And-join, And-split, Or-join, Or-split	Soundness, completeness, liberty
[30]	Abstract State Machine	Or-join	deadlock
[9]	Process Graph	Xor-split, And-split, Or-split, Xor-join, And-join, and Or-join	Soundness, safety
[29]	Separable Graphs	Or-join, cyclic graphs	/
[28]	Enablement and Firing Rules	Or-join	/
[10]	Process Graphs	Xor, Or	/
[27]	Workflow Graph	Or-join	/
[26]	Multipolar workflow graphs	Or	/
[25]	Colored Petri Nets	Xor, And, Or, and Complex gateway	/

TABLE II
BPMN GATEWAY

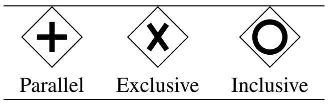


TABLE III
BPMN EVENTS

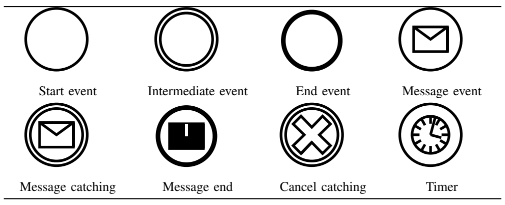


TABLE IV
SYNTAX OF DD-LOTOS

$E ::=$	**Behaviors**
	$stop \mid exit\{d\} \mid \Delta^d E \mid X[L] \mid$
	$g@t[SP]; E \mid i@t\{d\}; E \mid hide\, L\, in\, E \mid$
	$E[]E \mid E \mid [L] \mid E \mid E \gg E \mid E[> E \mid$
	$a!v\{d\}; E \mid$
	$a?xE$
$S ::=$	**Systems**
	$\phi \mid S \mid S \mid\ l(E)$

3) $$\frac{\neg Wait(M)}{{}_M[a!v\{d'+d\};E] \xrightarrow{d} {}_M[a!v\{d'\};E]}$$

4) $$\frac{\neg Wait(M)\ \ and\ \ d'>d}{{}_M[a!v\{d\};E] \xrightarrow{d'} {}_M[stop]}$$

Process: $a?xE$ As in the previous configuration ${}_M[a?xE]$, the reception begins once all actions in the set M complete their execution.

$$\frac{\neg Wait(M)}{{}_M[a?xE] \overset{M^{a?x}\, y}{\longrightarrow} {}_{\{y:a?x:0\}}[E]}$$

Remote Communication: The sending and receiving of messages via the same communication gate consume the silent action, is expressed by the following rule:

$$\frac{-}{{}_M[l(a!v\{d\};E1)]|_{M'}[k(a?xE2)] \xrightarrow{\tau} {}_M[l(E1)]|_{M'}[k(E2\{v/x\})]}$$

Time Evolution on System:

$$\frac{E \xrightarrow{d} E'}{l(E) \xrightarrow{d} l(E')}$$

$$\frac{S_1 \xrightarrow{d} S_1' \quad S_2 \xrightarrow{d} S_2'}{S_1 \mid S_2 \xrightarrow{d} S_1' \mid S_2'}$$

IV. INTERPRETATION OF BPMN GATEWAY IN DD-LOTOS

In this section, we provide an interpretation of different types of BPMN gateways using the DD-LOTOS formal language. Specifically, we focus on three types of gateways: the exclusive gateway (XOR), the parallel gateway (AND), and the inclusive gateway (OR). These gateways hold a pivotal role in controlling data flow within business processes and require precise interpretation to ensure consistency.

Gateways are used when it is necessary to control sequence flows; they serve as trigger mechanism that allows or prohibits the passage of the gateway [1].

In our previous work [33], we proposed an approach to transforming BPMP elements, including the different categories of activities and tasks, sub-processes, different events, message flows, and the two gateways: exclusive and parallel. However, the inclusive gateway has not been addressed in [33].

We defined a formal semantics for a subset of BPMN using the DD-LOTOS formal language. Additionally, we developed a transformation tool named BPMN2DDLOTOS [33], which allows to create BPMN collaboration diagrams and automatically generates the corresponding DD-LOTOS specifications.

In this paper, our main aim is to examine the formalization of three types of gateway: Exclusive, Inclusive and Parallel. First, we will discuss the interpretation of these three gateways using the DD-LOTOS formal language. For each gateway, we'll illustrate the interpretation with an example. We will then develop this approach through a case study representing the online purchasing process.

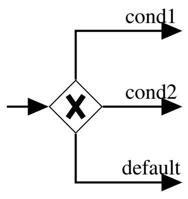
A. Exclusive Gateway XOR

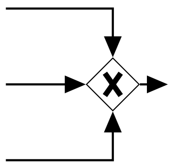
For merge behavior, incoming branches behave according to pass-through semantics. In branching behavior, one branch from the set of outgoing branches is activated [1].

The table V shows the interpretation of the exclusive gateway in terms of the DD-LOTOS specification. The table is divided into two lines, the first for the split version specification, which splits the input flow into several flows. The second line is for the join version, which merges several input flows into a single flow.

TABLE V
EXCLUSIVE GATEWAY

Exclusive Gateway	DD-LOTOS interpretation

	`process XorSplit[]::=`
	`gateway ? x: Token;`
	`([cond1=true] -> gateway!x;task1;`
	`[]`
	`[cond2=true] -> gateway!x;task2;`
	`[]`
	`gateway!x;task3;//Default branch`
	`[]`
	`stop) //exception`
	`endproc.`

	`process XorJoin[]::=`
	`Choice g in [$g_1,g_2,g_3]$ []`
	`g ? x:Token;`
	`gateway ! x;exit;`
	`endproc.`

The behavioral expression of XorSplit in Table V expresses that an outgoing branch is chosen when its condition expression is evaluates as true. If no conditions are true, the default outgoing branch is selected, as defined in [1]. In this case, the BPMN process generates an exception, which is translated into the DD-LOTOS Stop process to handle exceptions.

The behavioral expression of XorJoin in the table V expresses that only one incoming branch is required, so the XorJoin gateway is enabled.

Example: Let's consider a process in which student requests are received and processed. Requests can be of two types: "information request" or "service request". An XOR gateway is used to decide on processing according to the type of request.

Figure IV-A shows the BPMN collaboration diagram for the student request processing using the XorSplit gateway. The diagram consists of start and end events, an XorSplit gateway, two tasks, and an message intermediate catching event.

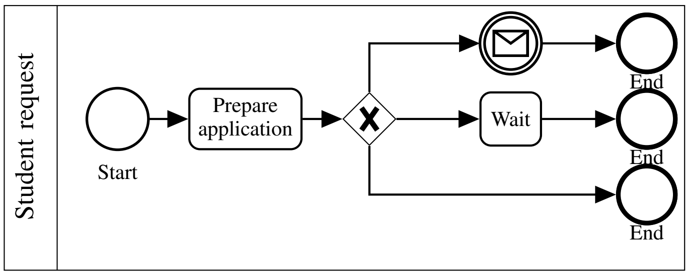


Fig. 1. Xor split gateway

The process begins with the start event, followed by the request preparation task. Next, an exclusive gateway is introduced, allowing the selection of one or more outgoing branches. If the student requests information, the first branch is activated, resulting in the execution of the intermediate message capture event. However, if the student requests a service, the waiting task is triggered by the second branch. Finally, the third branch, representing the default branch, leads to the execution of the end event.
The DD-LOTOS specification generated by the tool is given as follows:

```
Specification ExclusiveGateway[]::=
Behavior L(Student_request)

where
process Student_request[gateway]::=

[true]->(
  Prepare_application;
  gateway ? M:Token; //Reception of request
  [M=”request_information”]-> gateway ! M;exit

  []

  [M=”request_service”]-> wait;exit)

  []

  exit)

EndProc

EndSpec
```

B. Parallel Gateway AND

In the merge behavior, all incoming branches must terminate before the parallel gateway can continue the flow on the outgoing branch.

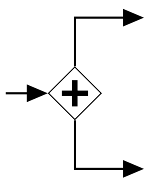
In split behavior, flow continues on all outgoing branches simultaneously.

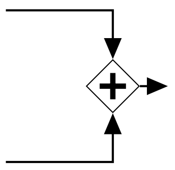
The table VI shows the interpretation of the parallel gateway in terms of the DD-LOTOS specification. The table is divided into two lines, the first for the split version specification, which splits the input flow into several flows. The second line is for the join version, which merges several input flows into a single flow.

The AndSplit gateway in the table VI indicates that the incoming flow is represented by the reception of a message via the parallel gateway, followed by the simultaneous sending of the message on all outgoing branches.

TABLE VI
PARALLEL GATEWAY

Parallel Gateway	DD-LOTOS interpretation

	`process AndSplit[]::=`
	`  gateway ? x: Token;`
	`    ( g1 ! x ||| g2 ! x )`
	`endproc`

	`process AndJoin[]::=`
	` ( g1 ? x: Token;`
	`|||`
	`   g2 ? x: Token;`
	`  ) >> gateway ! x; exit`
	`endproc`

On the other hand, for the AndJoin gateway, all incoming branches must receive the message before the flow can continue by sending the received message to the outgoing branch. This is expressed by the $>>$ operator in DD-LOTOS. In DD-LOTOS, the semantics of the process $E >> P$ stipulate that P can only begin execution once E has completed its execution.

Example: In this example, we offer a student enrollment service for a specific course using a parallel gateway.

The course is offered to students both in class and online. Both categories are mandatory for course validation.

Figure IV-B shows the BPMN collaboration diagram for the student enrollment service using the two gateways AndSplit and AndJoin. The diagram consists of start and end events, an AndSplit gateway, an AndJoin gateway, the enrollment task, and two tasks: Inclass and OnLine.

The process begins with the start event, followed by the student enrollment task. Next, a parallel gateway is introduced to split the flow (AndSplit). This gateway activates all outgoing branches, resulting in two branches in this case: one for online course sessions and one for in-class course sessions. Both branches must be validated by the student for the second AndJoin gateway to be activated. However, if the student fails to validate one or both branches, the AndJoin gateway will never be activated.

The DD-LOTOS specification generated by the tool is given as follows:

```
Specification ParalleGateway[]::=
Behavior L(Student_enrollment)

where
process Student_enrollment[gateway]::=

[true]->(
 Enrollment;
(
 gateway ? M:Token; //
 Reception of enrollment
 (Inclass;exit ||| Onclass;exit )
 )
  >>gateway!M; exit
  )
EndProc

EndSpec
```

C. Inclusive Gateway OR

The inclusive gateway behavior (OrSplit), which splits the flow, can be used to create alternative paths as well as parallel paths within a process flow. All condition expressions are evaluated, and all outgoing branches with an evaluation of the condition expression as true will be taken. Therefore, when designing the gateway, it is essential to take into account that at least one outgoing branch will be selected.

An inclusive merge gateway (OrJoin) is used to merge multiple incoming, alternative, and parallel paths [1].

The semantics of the OrJoin gateway is quite complex. In addition to token based activation of incoming branches, it may also depend on the distribution and evolution of tokens on branches other than the incoming branches [9].

In the context of this work, we will restrict ourselves to cases where the activation of the ”OR Join” gateway depends solely on the tokens in the incoming branches. This means focusing on local semantics, where the merging of parallel paths is determined by the tokens coming directly from the incoming branches without taking into account the distribution of tokens on other branches.

So the semantics of the ”OR Join” gateway in the context of the above-mentioned restriction depend solely on the fact that at least one token is present in one of the incoming branches. As soon as at least one of the branches contains a token, the gateway is activated, and the flow can continue.

In other words, the OR Join gateway is activated as soon as at least one token is present in one of the incoming branches, or if several branches have tokens, all of them will be taken into account to activate the gateway.

The table VII shows the interpretation of the inclusive gateway in terms of the DD-LOTOS specification. The table is divided into two lines, the first for the split version specification, which splits the input flow into several flows. The second line is for the join version, which merges several input flows into a single flow.

The DD-LOTOS specification of OrSplit in table VII expresses that all outgoing branches whose conditions are evaluated by the true value are executed.

The DD-LOTOS specification for the OrJoin gateway in the table VII expresses that the OrJoin gateway is activated when tokens occur on all branches or on a single or subset of branches.

Example: Let’s look at the process for managing and evaluating projects submitted by students. Projects are distributed among several evaluators, each of whom is responsible for evaluating one project. This is a parallel process, i.e., several evaluators can work at the same time.

The evaluators begin assessing the projects assigned to them. This stage is also carried out in parallel.

As soon as at least one evaluator has finished evaluating a project, the process can continue to the next stage, even if other evaluators are still evaluating other projects. This step is ensured by an OR Join gateway.

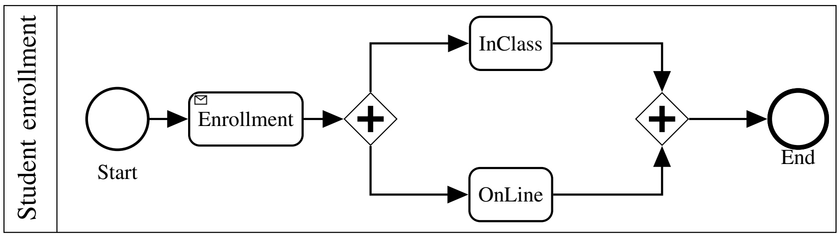


Fig. 2. And split and And join gateway

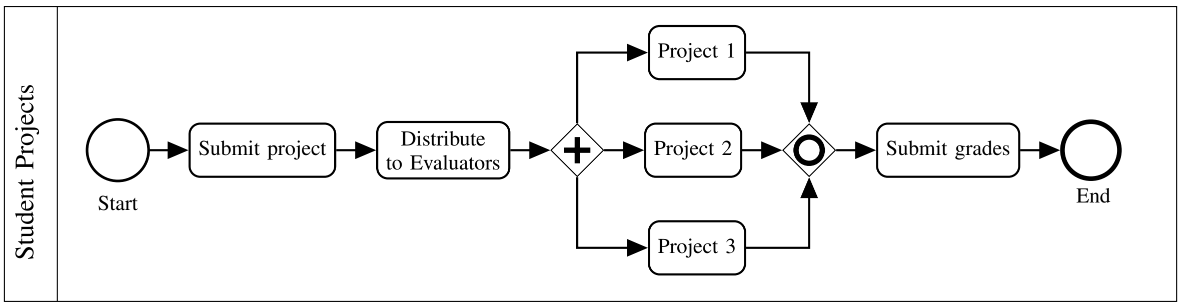
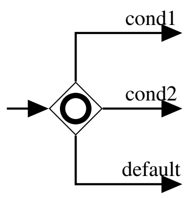


Fig. 3. Or join gateway

TABLE VII
INCLUSIVE GATEWAY

Inclusive Gateway	DD-LOTOS interpretation

	`process OrSplit[]::=`
	`gateway ? x: Token;`
	`(([cond1=true] -> gateway!x;task1;`
	`      $\|\|\|$`
	`  [cond2=true] -> gateway!x;task2;)`
	`  []`
	`  gateway!x;task3;)//Default branch`
	`EndProc.`

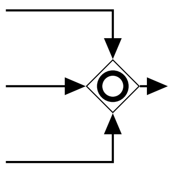
	`process OrJoin[]::=`
	
	`gateway?x:Token;`
	`(`
	`[x=g1 and x=g2 and x=g3]->`
	`//Gateway activation`
	`//by all branches`
	
	`  ( g1 ? x;`
	`     |||`
	`    g2 ? x;`
	`     |||`
	`    g3 ? x;`
	`   )>>gateway!x;exit;`
	`[]`
	
	`[x=g1 or x=g2 or x=g3]->`
	`//gateway activation by a single`
	`//branch or subset of branches`
	
	`   ( g1 ? x;`
	`      []`
	`     g2 ? x;`
	`      []`
	`     g3 ? x;`
	`)>>gateway!x;exit;`
	`)`
	`endproc`

Figure IV-C shows the BPMN collaboration diagram used to evaluate projects submitted by students, using the two gateways AndSplit and OrJoin.

The process begins with the start event, followed by the task of students handing in projects. The ”Distribution to Evaluators” task and the AndSplit parallel gateway then distribute the projects to the evaluators. The AndSplit gateway enables all outgoing branches to be activated, thus ensuring parallel evaluation. After this, each assessor evaluates the project assigned to him, and an inclusive gateway is introduced, enabling one or more outgoing branches to be selected. The final step is for the evaluators to submit their scores.
The DD-LOTOS specification generated by the tool is given as follows:

```
Specification InclusiveGateway[]::=
Behavior L(Student_request)

where
process Student_request[gateway]::=

[true]->(
   Submit_project;
   Distribute_evaluators;
   gateway ? M:Token;
   ( Project1; gateway ! M;exit;
       |||
     Project2; gateway ! M;exit;
       |||
     Project3; gateway ! M;exit;
    )>>Submit_grades;exit;

EndProc
EndSpec
```

V. CASE STUDY

In this section, we illustrate our approach through a case study widely treated in the literature, namely the online purchasing process.

The approach comprises two stages: the first focuses on the design of the collaboration diagram, highlighting the different actors and the communications between them. The second step focuses on generating the corresponding DD-LOTOS specification. This specification can then be verified using model-checking tools such as UPPAAL.

A. Description

The online purchasing process can be described as follows: A customer begins by browsing the website of an online store, where he or she explores the various product catalogues. Once they find the product or products they want, they add them to their shopping cart.

Next, the customer selects the delivery address where they wish to receive their order. The customer then enters the payment details and confirms the order.

The company begins processing the order, prepares the products and organizes delivery. Customers can track the status of their order using the tracking number provided. Once the order has been delivered, the customer receives it at the specified delivery address.

Figure V-A shows the BPMN collaboration diagram used for the online purchasing process.

The process begins with the START event, followed by an exclusive gateway. This gateway can be activated in two ways: either after the START event, or if the customer decides to cancel the purchase of a product. Once activated, the customer can select products and add them to the shopping cart. A message is then sent to the store process to either cancel the operation and repeat the purchase, or confirm and prepare the order. The parallel gateway enables both tasks to be carried out simultaneously: verification of card information and verification of the card itself. The customer then selects the delivery method, either express or standard, and the process is completed.

B. DD-LOTOS Specification

The DD-LOTOS specification generated by the tool is given as follows:

```
Specification  Online_purchasing[canal]::=

Behavior L(Customer)|[canal]|L(Store)

where

process  Customer[canal]::=

[true]->(
   choice g in[g1,g2][];
   g?x: Token;
   gateway ! x;
   Select_product;
   Add_product;
   canal ! m;
   exite;
EndProc

process  Store[canal]::=

canal?x:Token;
(
   [x="cancel"]->  gateway!x;Select_product;
                  Add_product;canal! m;exit
  []

  [x="ok"]-> (
             Confirm_ordre;Prepare_order;
             (
             Card_inf;exit
                 |||
             Card_check;exit
             )
             >>(gateway!x;
                (
      [x="express"]->Express_delivery;exit
          []
      [x="standard"]->Standard_delivery;exit
                 )
                 )
                 )
  []

  exit)
EndProc

EndSpec
```

C. Model Checking

In this section, we propose the formal verification of certain smooth functioning properties in our case study. This approach involves two main steps. First, we generate a semantic model that corresponds to the DD-LOTOS specification, which is named C-DATA [34, 37]. The second step involves utilizing the UPPAAL model checker, which takes both the C-DATA model and the property to be checked as input. Subsequently, UPPAAL provides either a result of a satisfied or unsatisfied property.

Within this context, we present the properties that have been verified.

A. **Deadlock property** : It is a safety property expressed through the following CTL formula.

```
A[]not deadlock
```

This property is satisfied in our case study, ensuring the absence of deadlock in our model.

B. If the customer successfully confirms and pays for the order, the online purchasing system guarantees delivery. It's a liveness property that ensures that all confirmed orders will be delivered.

```
A[](E<>(customer.pays ->
        A<>(online.delivery)))
```

This property is satisfied in our case study.

VI. Conclusion and Future Work

BPMN notation is widely used in business process modeling. However, BPMN diagrams lack formal semantics, which makes them difficult to analyze and complicates the detection of certain errors or inconsistencies in specifications. Therefore, it is essential to formalize BPMN diagrams and integrate

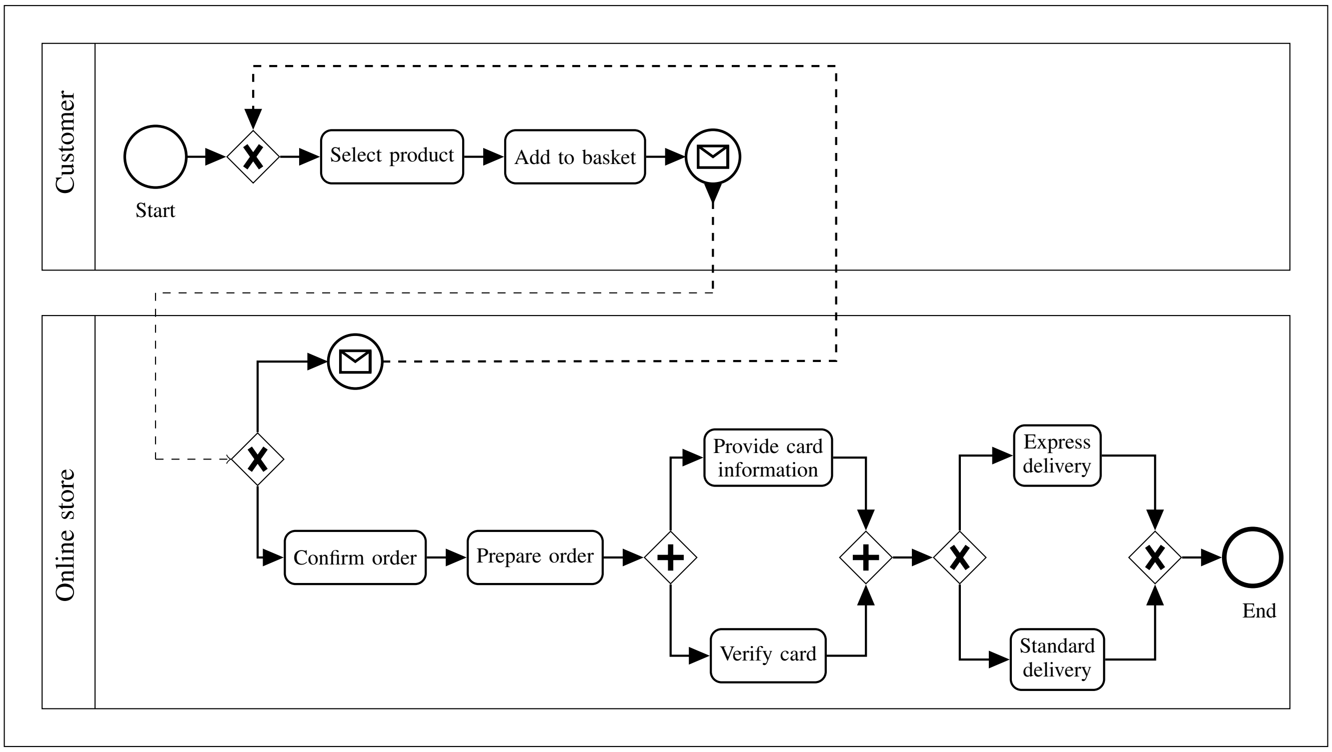


Fig. 4. Online purchasing business process

formal methods with BPMN notation to ensure the reliability of designed business processes.

The authors of [11–18] have proposed approaches to remedy the lack of formal semantics in BPMN diagrams. In general, their approach involves of mapping BPMN diagrams to formal languages, such as Maude, CSP and Petri nets.

Gateways are fundamental elements of BPMN notation and play a crucial role in the design of BPMN diagrams. For this reason, the works [9, 10, 25–32] have focused on exploring different types of gateways, including exclusive, inclusive, and parallel gateways.

In this paper, we have proposed a formalization of exclusive, parallel, and inclusive gateways using the DD-LOTOS formal language. The choice of the DD-LOTOS language is motivated by the formal semantics of the DD-LOTOS language, defined on a model of true concurrency semantics. Business process examples were studied to illustrate the use of each gateway, and a case study was presented to highlight all the formalization stages.

In our future work, we plan to focus on formalizing collaboration diagrams containing nested gateways with cycle.

Acknowledgment

The authors would like to thank the ICOSI laboratory for its financial support and encouragement.

References

[1] "Business Process Modeling Notation (BPMN)," Object Management Group. [Online]. Available: www.omg.org/spec/BPMN/2.0/

[2] F. Alexandra, "Business process modeling notation - an overview," *Annals. Computer Science Series*, vol. 4, no. 1, pp. 41–49, 2006.

[3] I. Ajmi, H. Zgaya, L. Gammoudi, S. Hammadi, A. Martinot, R. Beuscart, and J.-M. Renard, "Mapping patient path in the pediatric emergency department: A workflow model driven approach," *Journal of Biomedical Informatics*, vol. 54, pp. 315–328, 2015.

[4] N. Iglesias, J. M. Juarez, and M. Campos, "Business process model and notation and openehr task planning for clinical pathway standards in infections: Critical analysis," *J Med Internet Res*, vol. 24, no. 9, p. e29927, Sep 2022. [Online]. Available: https://doi.org/10.2196/29927

[5] E. R. Aguilar, F. García, F. Ruiz, M. G. Piattini, L. Calahorra, M. García, and R. Martin, "Process modeling of the health sector using bpmn: A case study," in *International Conference on Health Informatics*, 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:6610857

[6] B. Kissa, E. Gounopoulos, M. Kamariotou, and F. Kitsios, "Business process management analysis with cost information in public organizations: A case study at an academic library," *Modelling*, vol. 4, no. 2, pp. 251–263, 2023. [Online]. Available: https://www.mdpi.com/2673-3951/4/2/14

[7] A. Van Looy, "A quantitative and qualitative study of the link between business process management and digital innovation," *Information and Management*, vol. 58, no. 2, p. 103413, 2021.

[8] Object Management Group, *BPMN 2.0 by example*, Object Management Group, Inc., 140 Kendrick Street, Needham, MA 02494, U.S.A, 2010. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/examples/PDF

[9] F. Corradini, C. Muzi, B. Re, L. Rossi, and F. Tiezzi, "BPMN 2.0 OR-Join Semantics: Global and local characterisation," *Journal Information Systems*, vol. 105, p. 101934, 2022.

[10] D. Christiansen, M. Carbone, and T. Hildebrandt, "Formal Semantics and Implementation of BPMN 2.0 Inclusive Gateways," vol. 6551, p. 10, 2011.

[11] R. Dijkman, M. Dumas, and C. Ouyang, "Semantics and analysis of business process models in BPMN," *Information Software Technology*, vol. 50, no. 12, pp. 1281–1294, 2008.

[12] A. Lyazidi and S. Mouline, "A model-driven engineering approach to formally verify BPMN models using petri nets," *Int. J. Business Process Integration and Management*, vol. 8, no. 4, pp. 273–284, 2017.

[13] T. Takemura, "Formal semantics and verification of BPMN transaction and compensation," in *Proceedings of the 3rd IEEE Asia-Pacific Services Computing Conference, APSCC 2008*, Yilan, Taiwan, 2008, pp. 284–290.

[14] I. Raedts, M. Petkovic, Y. S. Usenko, J. M. Van der Werf, J. F. Groote, and L. J. Somers, "Transformation of BPMN models for behaviour

analysis," in *Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS 2007)*, Funchal, Madeira, Portugal, 2007, pp. 126–137.

[15] A. Kheldoun, K. Barkaoui, and M. Ioualalen, "Formal verification of complex business processes based on high-level petri nets," *Information Sciences*, vol. 385-386, pp. 39–54, 2017.

[16] N. El-Saber and A. Bornat, "BPMN formalization and verification using maude," in *Proceedings of the 2014 Workshop on Behaviour Modelling-Foundations and Applications*. ACM, 2014, pp. 1–12.

[17] M. Güdemann, P. Poizat, G. Salaün, and A. Dumont, "Verchor: A framework for verifying choreographies," in *Springer Berlin Heidelberg*, 2013, pp. 226–230.

[18] P. Y. H. Wong and J. Gibbons, "A relative timed semantics for BPMN," *Electronic Notes in Theoretical Computer Science*, vol. 229, no. 2, pp. 59–75, 2009.

[19] F. Corradini, F. Fornari, A. Polini, B. Re, and F. Tiezzi, "A formal approach to modeling and verification of business process collaborations," *Science of Computer Programming*, vol. 166, pp. 35–70, 2018.

[20] P. Poizat and G. Salaün, "Checking the realizability of BPMN 2.0 choreographies," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC'12*. ACM, 2012, pp. 1927–1934.

[21] P. Poizat, G. Salaün, and A. Krishna, "Checking business process evolution," in *Springer International Publishing*, 2017, pp. 36–53.

[22] P. Y. H. Wong and J. Gibbons, "Verifying business process compatibility," in *The Eighth International Conference on Quality Software*, 2008, pp. 126–131.

[23] F. Corradini, A. Morichetta, C. Muzi, B. Re, L. Rossi, and F. Tiezzi, "Well-structuredness, safeness and soundness: A formal classification of BPMN collaborations," *Journal of Logical and Algebraic Methods in Programming*, vol. 119, p. 100630, 2021.

[24] F. Corradini, A. Morichetta, A. Polini, B. Re, and F. Tiezzi, "Correctness checking for BPMN collaborations with sub-processes," *Journal of Systems and Software*, vol. 166, p. 110594, 2020.

[25] C. Dechsupa, W. Vatanawood, and A. Thongtak, "Transformation of the BPMN Design Model into a Colored Petri Net Using the Partitioning Approach," *IEEE Access*, vol. 6, pp. 38 421–38 436, 2018.

[26] D. Fahland and H. Völzer, "Dynamic Skipping and Blocking and Dead Path Elimination for Cyclic Workflows," in *Business Process Management. BPM 2016*, ser. Lecture Notes in Computer Science, M. La Rosa, P. Loos, and O. Pastor, Eds., vol. 9850. Springer, Cham, 2016.

[27] B. Gfeller, H. Völzer, and G. Wilmsmann, "Faster Or-Join Enactment for BPMN 2.0," in *Business Process Model and Notation. BPMN 2011*, ser. Lecture Notes in Business Information Processing, R. Dijkman, J. Hofstetter, and J. Koehler, Eds., vol. 95. Springer, Berlin, Heidelberg, 2011, p. 3.

[28] M. Dumas, A. Grosskopf, T. Hettel, and M. Wynn, "Semantics of Standard Process Models with OR-Joins," in *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 4803. Springer, Berlin, Heidelberg, 2007, p. 5.

[29] H. Völzer, "A New Semantics for the Inclusive Converging Gateway in Safe Processes," vol. 6336, p. 21, 2010.

[30] E. Börger, O. Sörensen, and B. Thalheim, "On Defining the Behavior of OR-joins in Business Process Models," *Journal of Universal Computer Science*, vol. 15, pp. 3–32, 2009.

[31] T. M. Prinz and W. Amme, "A Complete and the Most Liberal Semantics for Converging OR Gateways in Sound Processes," *Complex Systems Informatics and Modeling Quarterly*, pp. 32–49, 2015. [Online]. Available: http://dx.doi.org/10.7250/csimq.2015-4.03

[32] F. Corradini, A. Polini, B. Re, and F. Tiezzi, "An Operational Semantics of BPMN Collaboration," in *Formal Aspects of Component Software*, ser. Lecture Notes in Computer Science, C. Braga and P. C. Ölveczky, Eds., vol. 9539. Springer, Cham, 2016, pp. 161–180.

[33] E. Maarouk, T. M. Merah, S. Ghaoui, and N. Rahabi, "Formal Semantics and Transformation of BPMN Models," *International Journal of Business Process Integration and Management*, vol. 9, no. 3, pp. 158–169, 2019.

[34] T. Maarouk, D. Saidouni, and M. Khergag, "DD-LOTOS : A distributed real time language," in *Proceedings 2nd Annual International Conference on Advances in Distributed and Parallel Computing (ADPC 2011) Special Track: Real Time and Embedded Systems (RTES 2011)*, 2011, pp. 45–50.

[35] D. Saidouni, "Sémantique de Maximalité: Application au Raffinement d'Actions en LOTOS," Ph.D. dissertation, LAAS-CNRS, 7 av. du Colonel Roche, 31077 Toulouse Cedex France, 1996.

[36] T. M. Maarouk, M. E. H. Souidi, and N. Hoggas, "Formalization and model checking of bpmn collaboration diagrams with dd-lotos," *COMPUTING AND INFORMATICS*, vol. 40, no. 5, p. 1080–1107, Dec. 2021. [Online]. Available: https://www.cai.sk/ojs/index.php/cai/article/view/202151080

[37] M. T. Messaoud, S. D. Eddine, M. Rafik, and H. Hichem, "Interpretation of dd-lotos specification by c-data*," in *New Trends in Databases and Information Systems*, T. Morzy, P. Valduriez, and L. Bellatreche, Eds. Cham: Springer International Publishing, 2015, pp. 414–423.

Toufik Messaoud Maarouk received his engineering degree in Computer Science from Annaba University, Algeria, in 1998, and obtained his Ph.D. in Computer Science from Constantine University, Algeria, in 2012. Currently, he holds the position of associate professor in the Department of Mathematics and Computer Science at the Faculty of Sciences and Technology, Abbes Laghrour University of Khenchela, Algeria. He also holds the position of Director at the Knowledge Engineering and Computer Security Lab, University of Khenchela. His main research areas include formal methods, concurrency theory, formal semantics, and distributed computing.

Mohammed El Habib Souidi received his BS degree in computer science from University of Khenchela (Algeria) in 2011. He also received his Master degree in Computer Science from the same university in 2013, and his Ph.D in Computer Science from Harbin Institute of Technology (China) 2017. He is working as a Lecturer in Department of Mathematics and Computer Science in University of khenchela (Algeria). Moreover, he is affiliated as a researcher in ICOSI Lab (University of Khenchela). His research interests include: Multi-agent task coordination, Reinforcement learning, Game theory, and Path planning.

Makhlouf Ledmi received his BS degree from the University of Annaba, in 1997, MS degree from the University of Khenchela, in 2010 and PhD from the University of Batna 2, Algeria, in 2020, all in Computer Science. He served as the Head of Mathematics and Computer Science Department at the University of Khenchela, between 2015 and 2017. He is currently an Associate Professor at the Mathematics and Computer Science Department at Abbes Laghrour University of Khenchela and member of Knowledge Management Group ICOSI Lab. His research interests include data mining, soft computing, machine learning and bioinformatics.

Sabeg Samra obtained her degree in Computer Science from the Batna University, Algeria in 2006. She received her MSc in Computer Science from the Khenchela University, Algeria in 2019. She is a Ph.D. student at the Khenchela University, Algeria.