Decision Making in Manufacturing and Services
Vol. 5 • 2011 • No. 1–2 • pp. 57–67

BPMN – A Logical Model and Property Analysis

Antoni Ligęza[*]

Abstract. Business Process Modeling Notation has become a powerful and widely accepted visual language for modeling business processes. Despite its expressive power and high usability, a weak point of BPMN is the lack of formal semantics and difficulties with assuring correctness of the overall process. In this paper an attempt is made towards investigation and development of foundations for a logical, declarative model for BPMN. Such model should enable formal analysis of desired properties referring to correct operation of Business Processes modeled with use of BPMN.

Keywords: Business Process Modeling Notation, BPMN, Business Rules, Rule-Based Systems, eXtended Tabular Trees, XTT, XTT2, System Verification, Formal Analysis, Declarative Model

Mathematics Subject Classification: 68N30

Revised: 30 May 2011

1. INTRODUCTION

Design and analysis of progressively more complex business processes requires advanced methods and tools. Two modern approaches to modeling such processes have recently gained wider popularity. These are the *Business Process Modeling Notation* (Silver 2009, Stephen and Derek 2008, Allweyer 2010), or BPMN for short, and *Business Rules* (Ross 2006, Ambler 2003, Giurca *et al.* 2009). Although aimed at a common target, both of the approaches are rather mutually complementary and offer somewhat distinctive features enabling process modelling.

BPMN constitutes a set of graphical symbols, such as links modeling workflow, various splits and joins, events and boxes symbolizing data processing. It constitutes a transparent visual tool for modeling complex processes promoted by OMG (OMG 2011). What is worth underlying is the expressive power of current BPMN. In fact it allows for modeling conditional operations, loops, event-triggered actions, splits and joins of data flow paths and communication processes (Ouyang *et al.* 2006b). Moreover, modeling can take into account several levels of abstraction which enables hierarchical approach. Another characteristic feature of BPMN is that the workflow covers

[*] AGH University of Science and Technology, Krakow, Poland

both *data processing* visualization and *workflow control* specification. The workflow diagram, however, although it provides transparent, visual picture of the process, due to lack of formal model makes attempts at more rigorous analysis problematic.

Business Rules, also promoted by OMG (OMG 2003), offer an approach to specification of knowledge in a *declarative* manner. The way the rules are applied is left over until it comes to rule execution. The rules themselves can play different roles in the system (OMG 2003). Some two most important ones cover *declarative knowledge specification* for inference of new facts, and *control knowledge specification* for efficient implementation of inference process.

The main common problem of BPMN is lack of a *formal declarative model* defining precisely the logic behind the diagram. Hence defining and analyzing correctness of BPMN diagrams (e.g. in terms of termination or determinism) is a hard task. There are only few papers undertaking the issues of analysis and verification of BPMN diagrams (Dijkman *et al.* 2007, Ouyang *et al.* 2006a, Ouyang *et al.* 2006b). However, the analysis is performed mostly at the *structural* level and does not take into account the semantics of dataflow and control knowledge.

This paper presents an attempt at defining foundations for a more formal, logical, declarative model of BPMN diagrams. The model is aimed at enabling definition and further analysis of formal properties of a class of restricted BPMN diagrams. The analysis should take into account properties constituting reasonable criteria of correctness. The focus is on development of a formal, declarative model of BPMN components and its overall structure. In fact, a combination of recent approaches to development an verification of rule-based systems (Nalepa and Ligęza 2010, Ligęza and Nalepa 2011) seems to have potential influence on BPMN analysis.

2. BPMN: A RESTRICTED STRUCTURAL MODEL

In this section a simplified structural model of BPMN diagrams is put forward. It constitutes a restricted abstraction of crucial intrinsic workflow components. As for events, only start and termination events are taken into account. Main knowledge processing units are activities (or tasks). Workflow control is modeled by three types of gateways: split and join operations. Finally, workflow sequence is modeled by directed links. No time/temporal aspects are considered. The following elements will be taken into consideration:

- $\mathbb{S}$ – a non-empty set of *start events* (possibly composed of a single element),
- $\mathbb{E}$ – a non-empty set of *end events* (possibly composed of a single element),
- $\mathbb{T}$ – a set of *activities* (or *tasks*); a task $T \in \mathbb{T}$ is a finite process with single input and single output, to be executed within a finite interval of time,
- $\mathbb{G}$ – a set of *split gateways* or *splits*, where branching of the workflow takes place; three disjoint subtypes of splits are considered:
  - $\mathbb{GX}$ – a set of *exclusive splits* where one and only one of the alternative paths can be followed (a split of EX-OR type),
  - $\mathbb{GP}$ – a set of *parallel splits* where all the paths of the workflow are to be followed (a split of *AND* type or a *fork*), and

  - $\mathbb{GO}$ – a set of *inclusive splits* where one or more paths should be followed (a split of *OR* type).
- $\mathbb{M}$ – a set of *merge gateways* or *join* nodes of the diagram, where two or more paths meet; three further disjoint subtypes of merge (join) nodes are considered:
  - $\mathbb{MX}$ – a set of *exclusive merge* nodes where one and only one input path is taken into account (a merge of of EX-OR type),
  - $\mathbb{MP}$ – a set of *parallel merge* nodes where all the paths are combined together (a merge of *AND* type), and
  - $\mathbb{MO}$ – a set of *inclusive merge* nodes where one or more paths influence the subsequent item (a merge of *OR* type).
- $\mathbb{F}$ – a set of workflow links, $\mathbb{F} \subseteq \mathbb{O} \times \mathbb{O}$, where $\mathbb{O} = \mathbb{S} \cup \mathbb{E} \cup \mathbb{T} \cup \mathbb{G} \cup \mathbb{M}$ is the join set of objects. All the component sets are pairwise disjoint.

The splits and joins depend on logical conditions assigned to particular branches. It is assumed that there is defined a partial function $\texttt{Cond}\colon \mathbb{F} \rightarrow \mathbb{C}$ assigning logical formulae to links. In particular, the function is defined for links belonging to $\mathbb{G} \times \mathbb{O} \cup \mathbb{O} \times \mathbb{M}$, i.e. outgoing links of split nodes and incoming links of merge nodes. The conditions are responsible for workflow control. Some simple logical models are presented in Section 4. For intuition, a simple BPMN diagram is presented in Figure 1.

Having selected the core BPMN elements it is necessary to state restrictions on the overall diagram structure. The following is a set of typical requirements defining the so-called *well-formed diagram* (Ouyang *et al.* 2006b):

- $\forall s \in \mathbb{S}$, $\texttt{in}(s) = \emptyset$ and $|\texttt{out}(s)| = 1$ – any start node $s \in \mathbb{S}$ has no incoming links and exactly one outgoing link,
- $\forall e \in \mathbb{E}$, $|\texttt{in}(e)| = 1$ and $\texttt{out}(e) = \emptyset$ – any end event node $e \in \mathbb{E}$ has no outgoing links and exactly one incoming link,
- $\forall T \in \mathbb{T}$, $|\texttt{in}(T)| = 1$ and $|\texttt{out}(T)| = 1$ – any task node $T \in \mathbb{T}$ has exactly one input and one output link,
- $\forall g \in \mathbb{G}$, $|\texttt{in}(g)| = 1$ and $|\texttt{out}(g)| \geqslant 2$ – any split node $g \in \mathbb{G}$ has exactly one incoming link and at least two outgoing ones,
- $\forall m \in \mathbb{M}$, $|\texttt{in}(m)| \geqslant 2$ and $|\texttt{out}(m)| = 1$ – any merge node $m \in \mathbb{M}$ has at least two incoming links and exactly one outgoing link,
- $\forall f \in \mathbb{F}$, $f \in \texttt{out}(\mathbb{S} \cup \mathbb{T} \cup \mathbb{G} \cup \mathbb{M}) \times \texttt{in}(\mathbb{E} \cup \mathbb{T} \cup \mathbb{G} \cup \mathbb{M})$, i.e. every link joins some legal output of some object with a legal input of some (other) object,
- every object $o \in \mathbb{O}$ is on some path from some start event and an end event.

From now on only diagrams satisfying the above minimal requirements will be taken into account.

Note that a well-formed diagram does not assure that for any input knowledge the process can be executed leading to a (unique) solution. This further depends on the particular input data, its transformation during processing, correct work of particular objects, and correct control defined by the branching/merging conditions assigned to links.

3. THERMOSTAT: AN ILLUSTRATIVE EXAMPLE

In order to provide intuitions, the theoretical considerations will be illustrated with a simple example process. The process goal is to establish the so-called *set-point* temperature for a thermostat system (Negnevitsky 2002). The selection of the particular value depends on the season, whether it is a working day or not, and the time of the day. A BPMN diagram of the process is specified in Figure 1.

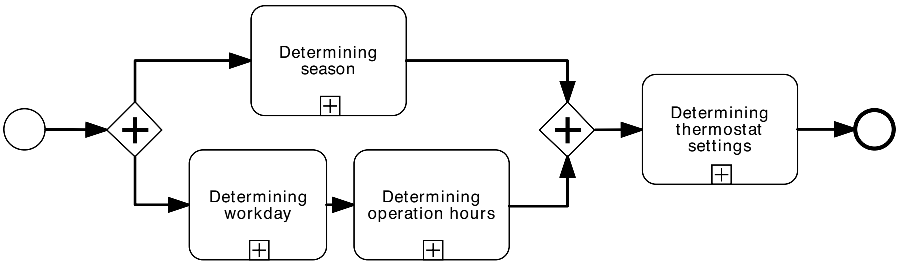

Fig. 1. *An example BPMN diagram – top-level specification of thermostat system.*

After start, the process is split into two independent paths of activities. The upper path is aimed at determining the current season[1] (aSE; it can take one of the values $\{sum, aut, win, spr\}$; the detailed specification is provided with rules 7–10 below). A more visual specification of this activity with an appropriate set of rules is shown in Figure 2.

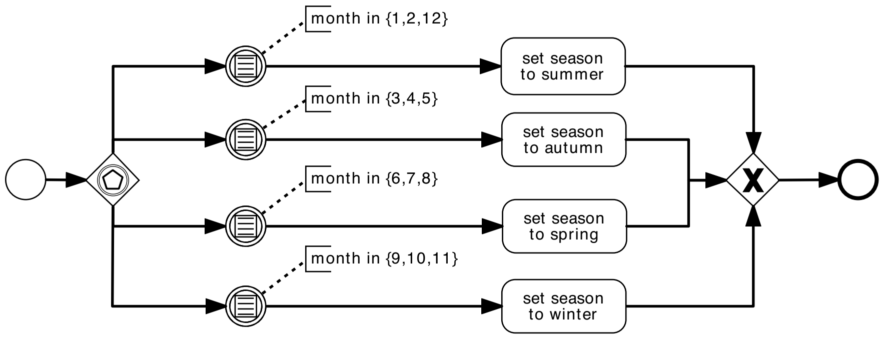

Fig. 2. *An example BPMN diagram – detailed specification a BPMN task.*

The lower path activities determine whether the day (aDD) is a workday ($aTD = wd$) or a weekend day ($aTD = wk$), both specifying the value of today (aTD; specification provided with rules 1 and 2), and then, taking into account the current time (aTM), whether the operation (aOP) is during business hours ($aOP = dbh$) or

[1] For technical reasons all attribute names used in this example start with lower-case 'a'.

not ($aOP = ndbh$); the specification is provided with rules 3–6. Finally, the results are merged together and the final activity consists in determining the thermostat settings ($aTHS$) for particular season (aSE) and time (aTM) (the specification is provided with rules 11–18).

The whole process is formally specified with the following eighteen inference rules.

***Rule 1*:**	$aDD \in \{monday, tuesday, wednesday, thursday, friday\} \longrightarrow aTD = wd.$
***Rule 2*:**	$aDD \in \{saturday, sunday\} \longrightarrow aTD = wk.$
***Rule 3*:**	$aTD = wd \wedge aTM \in (9, 17) \longrightarrow aOP = dbh.$
***Rule 4*:**	$aTD = wd \wedge aTM \in (0, 8) \longrightarrow aOP = ndbh.$
***Rule 5*:**	$aTD = wd \wedge aTM \in (18, 24) \longrightarrow aOP = ndbh.$
***Rule 6*:**	$aTD = wk \longrightarrow aOP = ndbh.$
***Rule 7*:**	$aMO \in \{january, february, december\} \longrightarrow aSE = sum.$
***Rule 8*:**	$aMO \in \{march, april, may\} \longrightarrow aSE = aut.$
***Rule 9*:**	$aMO \in \{june, july, august\} \longrightarrow aSE = win.$
***Rule 10*:**	$aMO \in \{september, october, november\} \longrightarrow aSE = spr.$
***Rule 11*:**	$aSE = spr \wedge aOP = dbh \longrightarrow aTHS = 20.$
***Rule 12*:**	$aSE = spr \wedge aOP = ndbh \longrightarrow aTHS = 15.$
***Rule 13*:**	$aSE = sum \wedge aOP = dbh \longrightarrow aTHS = 24.$
***Rule 14*:**	$aSE = sum \wedge aOP = ndbh \longrightarrow aTHS = 17.$
***Rule 15*:**	$aSE = aut \wedge aOP = dbh \longrightarrow aTHS = 20.$
***Rule 16*:**	$aSE = aut \wedge aOP = ndbh \longrightarrow aTHS = 16.$
***Rule 17*:**	$aSE = win \wedge aOP = dbh \longrightarrow aTHS = 18.$
***Rule 18*:**	$aSE = win \wedge aOP = ndbh \longrightarrow aTHS = 14.$

Even in this simple example, answers to the following important questions are not obvious:

- *correctness*: is any of the four activities specified in a correct way? Will each task end with producing desired output for any admissible input data?
- *consistency*: will it be possible to merge knowledge coming from different sources at the merge node?
- *termination/completeness*: does the specification assure that the system will always terminate producing some temperature specification for *any* admissible input data?
- *determinism*: is the output setting determined in a unique way?

Note that we do not ask about *correctness* of the result; in fact, the rules provide a kind of executable specification, so there is no reference point to claim that final output is correct or not.

4. BPMN DIAGRAM ANALYSIS: CONSISTENCY ISSUES

A BPMN diagram can model quite complex processes. Apart from *external consistency* validation (i.e. whether or not the diagram models correctly the external system in a *complete* way, does not introduce any non-existent features, and there is *isomorphism* between those two), an important issue is the *internal consistency* requirement for *correct structure* of the diagram and *correct workflow specification.* The first one refers to static specification of components and their connections. The second one consists in correct work of the structure for all admissible input data specification.

The structural correctness is defined by requirements for *well-formed BPMN diagram* (see Section 2). However, even having correct structure, the process can gone wrong due to unserved data or wrong workflow control, for example. Below, an attempt is made at specification of some minimal requirements for (i) correct work of process components (tasks), (ii) assuring data flow, (iii) correct work of splits, (iv) correct work of merge nodes, and finally – (v) termination of the overall process.

4.1. PROCESS COMPONENTS CORRECTNESS

In this section we put forward some minimal requirements defining correct work of rule-based process components performing BPMN activities. Each such component is composed of a set of inference rules, designed to work within the same context; in fact, preconditions of the rules incorporate the same attributes. In our example we have four such components: determining workday (rules 1–2), determining operation hours (rules 3–6), determining season (rules 7–10) and determining the thermostat setting (rules 11–18).

In general, the outermost logical model of a component T performing some activity/task can be defined as a triple of the form:

$$T = (\psi_T, \varphi_T, \mathcal{A}), \tag{1}$$

where ψ_T is a formula defining the restrictions on the component input, φ_T defines the restrictions for component output, and $\mathcal{A}$ is an algorithm which for a given input satisfying ψ_T produces an (desirably uniquely defined) output, satisfying φ_T. For intuition, ψ_T and φ_T define a kind of a 'logical tube' – for every input data satisfying ψ_T (located at the entry of the tube), the component will produce and output satisfying φ_T (still located within the tube at its output). The precise recipe for data processing is given by algorithm $\mathcal{A}$.

The specification of a rule-based process component given by (1) is considered *correct*, if and only if for any input data satisfying ψ_T the algorithm $\mathcal{A}$ produces an output satisfying φ_T. It is further *deterministic* (unambiguous) if the generated output is unique for any admissible input.

For example, consider the component determining operation hours. Its input restriction formula ψ_T is the disjunction of precondition formulae $\psi_3 \vee \psi_4 \vee \psi_5 \vee \psi_6$, where ψ_i is a precondition formula for rule i. We have $\psi_T = ((aTD = wd) \wedge (aTM \in [0,8] \vee aTM \in [9,17] \vee aTM \in [18,24])) \vee (aTD = wk)$. The output restriction formula is given by $\varphi_T = (aOP = dbh) \vee (aOP = ndbh)$. The algorithm is specified directly by the rules; rules are in fact a kind of *executable specification*.

In order to be sure that the produced output is unique, the following *mutual exclusion* condition should hold:

$$\not\models \psi_i \wedge \psi_j \tag{2}$$

for any $i \neq j$, $i, j \in \{1, 2, \ldots, k\}$. A simple analysis shows, that the four rules have mutually exclusive preconditions, and the joint precondition formula ψ_T covers any admissible combination of input parameters; in fact, the subset of rules is locally *complete* and *deterministic* (Ligęza 2006).

4.2. CORRECT DATA FLOW

In our example we consider only rule-based components. Let ϕ define the context of operation, i.e. a formula defining some restrictions over the current state of the knowledge-base that must be satisfied before the rules of a component are explored. For example, ϕ may be given by $\varphi_{T'}$ of a component T' directly preceding the current one. Further, let there be k rules in the current component, and let ψ_i denote the joint precondition formula (a conjunction of atoms) of rule i, $i = 1, 2, \ldots, k$. In order to be sure that at least one of the rules will be fired, the following condition must hold:

$$\phi \models \psi_T, \tag{3}$$

where $\psi_T = \psi_1 \vee \psi_2 \vee \ldots \vee \psi_k$ is the disjunction of all precondition formulae of the component rules. The above restriction will be called the *funnel principle*. For intuition, if the current knowledge specification satisfies restriction defined by ϕ, then at least one of the formula preconditions must be satisfied as well.

For example, consider the connection between the component determining workday and the following it component determining operation hours. After leaving the former one, we have that $aTD = wd \vee aTD = wk$. Assuming that the time can always be read as an input value, we have $\phi = (aTD = wd \vee aTD = wk) \wedge aTM \in [0, 24]$. On the other hand, the disjunction of precondition formulae $\psi_3 \vee \psi_4 \vee \psi_5 \vee \psi_6$ is given by $\psi_T = (aTD = wd) \wedge (aTM \in [0, 8] \vee aTM \in [9, 17] \vee aTM \in [18, 24])) \vee (aTD = wk)$. Obviously, the funnel condition given by (3) holds.

4.3. CORRECT SPLITS

An exclusive split $GX(q_1, q_2, \ldots q_k) \in \mathbb{GX}$ with k outgoing links is modelled by a fork structure assigned excluding alternative of the form:

$$q_1 \veebar q_2 \veebar \ldots \veebar q_k,$$

where $q_i \wedge q_j$ is always false for $i \neq j$. An exclusive split can be considered correct if and only if at least one of the alternative conditions is satisfied. We have the following logical requirement:

$$\models q_1 \vee q_2 \vee \ldots \vee q_k, \tag{4}$$

i.e. the disjunction is in fact a tautology. In practice, to assure (4), a predefined exclusive set of conditions is completed with a default q_0 condition defined as $q_0 = \neg q_1 \wedge \neg q_2 \wedge \ldots \wedge \neg q_k$; obviously, the formula $q_0 \vee q_1 \vee q_2 \vee \ldots \vee q_k$ is a tautology.

Note that in case when an input restriction formula ϕ is specified, the above requirement given by (4) can be relaxed to

$$\phi \models q_1 \vee q_2 \vee \ldots \vee q_k. \tag{5}$$

An inclusive split $GO(q_1, q_2, \ldots q_k \in \mathbb{GO}$ is modelled as disjunction of the form:

$$q_1 \vee q_2 \vee \ldots \vee q_k,$$

An inclusive split to be considered correct must also satisfy formula (4), or at least (5). As before, this can be achieved through completing it with the q_0 default formula.

A parallel split $GP(q_1, q_2, \ldots q_k) \in \mathbb{GP}$ is referring to a fork-like structure, where all the outgoing links should be followed in any case. For simplicity, a parallel split can be considered as an inclusive one, where all the conditions assigned to outgoing links are set to *true*.

Note that, if ϕ is the restriction formula valid for data at the input of the split, then any of the output restriction formula is defined as $\phi \wedge q_i$ for any of the outgoing link i, $i = 1, 2, \ldots, k$.

4.4. CORRECT JOINS

Consider a workflow merge node, where k knowledge inputs satisfying restrictions $\phi_1, \phi_2, \ldots, \phi_k$ respectively meet together, while the selection of particular input is conditioned by formulae $p_1, p_2, \ldots, p_k$, respectively.

An exclusive merge $MX(p_1, p_2, \ldots, p_k) \in \mathbb{MX}$ of k inputs is considered correct if and only if the conditions are pairwise disjoint, i.e.

$$\not\models p_i \wedge p_j \tag{6}$$

for any $i \neq j$, $i, j \in \{1, 2, \ldots, k\}$. Moreover, to assure that the merge works, at least one of the conditions should hold:

$$\models p_1 \vee p_2 \vee \ldots \vee p_k, \tag{7}$$

i.e. the disjunction is in fact a tautology. If the input restrictions $\phi_1, \phi_2, \ldots, \phi_k$ are known, condition (7) might possibly be replaced by $\models (p_1 \wedge \phi_1) \vee (p_2 \wedge \phi_2) \vee \ldots \vee (p_k \wedge \phi_k)$.

Note that in case a join input restriction formula ϕ is specified, the above requirement can be relaxed to

$$\phi \models p_1 \vee p_2 \vee \ldots \vee p_k, \tag{8}$$

and if the input restrictions $\phi_1, \phi_2, \ldots, \phi_k$ are known, it should be replaced by $\phi \models (p_1 \wedge \phi_1) \vee (p_2 \wedge \phi_2) \vee \ldots \vee (p_k \wedge \phi_k)$.

An inclusive merge $MO(p_1, p_2, \ldots, p_k) \in \mathbb{MO}$ of k inputs is considered correct if one is assured that the merge works – condition (7) or (8) hold.

A parallel merge $MP \in \mathbb{MP}$ of k inputs is considered correct by default. However, if the input restrictions $\phi_1, \phi_2, \ldots, \phi_k$ are known, a consistency requirement for the combined out takes the form that ϕ must be consistent (satisfiable), where:

$$\phi = \phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_k \tag{9}$$

An analogous requirement can be put forward for the active links of an inclusive merge.

Note that even correct merge leading to a satisfiable formula assure only passing the merge node; the funnel principle must further be satisfied with respect to the following-in-line object. To illustrate that consider the input of the component determining thermostat setting. This is the case of parallel merge of two inputs. The joint formula defining the restrictions on combined output of the components for determining season and determining operation hours is of the form:

$$\phi = (aSE = sum \vee aSE = aut \vee aSE = win \vee aSE = spr) \wedge (aOP = dbh \vee aOP = ndbh).$$

A simple check of all possible combinations of season and operation hours shows that all the eight possibilities are covered by preconditions of rules 11-18; hence the funnel condition (3) holds.

4.5. GLOBAL TERMINATION CONDITIONS

Finally, having a well-formed BPMN diagram with correct components, one can ask if the work terminates for a given input data. This question can be reformulated as if for a given termination node $e \in \mathbb{E}$, the node will be eventually reached. This can be assured by the requirement for *node reachability condition* defined recursively as follows:

- any start node $s \in \mathbb{S}$ is trivially reachable; it is simultaneously initiated by assigning it current input data satisfying formula ϕ;
- consider task T specified by (1); its output node is reachable if and only if its input node is reachable with restriction formula ϕ, such that the funnel condition given by (3) is satisfied;
- the case of split nodes:
  - an output node of an exclusive split node corresponding to a branch defined by switching condition q_i is reachable if and only if its input node is reachable with restriction formula ϕ, and $\phi \models q_i$; note that for a given ϕ *only one* output node can be reachable, and for correct exclusive split nodes exactly one can be reached for a particular current data;
  - an output node of an inclusive split node corresponding to a branch defined by switching condition q_i is reachable if and only if its input node is reachable with restriction formula ϕ, and $\phi \models q_i$; note that for a given ϕ *more than one* output node can be reachable, and for correct exclusive split nodes at least one will be reached for the current data;
  - any output node of a parallel split node is reachable, if and only if its input node is reachable.
- the case of merge nodes:
  - the output node of an exclusive merge node is reachable if and only if at least one of its input nodes is reachable, and the conjunction of input formula ϕ_i and the link condition p_i is satisfied,
  - the output node of an inclusive merge node is reachable if and only if at least one of its input nodes is reachable, and the conjunction of input formula ϕ_i and the link condition p_i is satisfied; in general case, when several input

nodes, say $i, i+1, \ldots, j$ have satisfied selection conditions, the formula $(p_i \wedge \phi_i) \vee (p_{i+1} \wedge \phi_{i+1}) \vee \ldots \vee (p_j \wedge \phi_j)$ must be satisfied,
- the output node of a parallel merge node is reachable if and only if all of it input nodes are reachable, and the conjunctive input formula is satisfiable.

Finally, a BPMN diagram specifies a workflow that terminates, if all at least one of its terminal nodes is reachable for a given input data and the selected start node.

5. CONCLUDING REMARKS

An attempt at providing a logical, declarative model for well-defined BPMN diagram is presented. The model is aimed at defining formal semantics of diagram components and the workflow operation. The main focus is on specification of correct components and correct dataflow. Global termination conditions are specified in a recursive way.

Note that the logical analysis can be performed *off-line*, on the base of logical requirements ϕ, ψ and φ of data. However, if such specifications are data-dependant (e.g. in case of loops or more complex non-monotonic data processing) the analysis may be possible only in *on-line* form, separately for any admissible input data.

ACKNOWLEDGMENTS

Research supported from the Polish Ministry of Science and Education under the BIMLOQ Project N N516 422338, 2010–2012.

REFERENCES

Allweyer, T., 2010. *BPMN 2.0. Introduction to the Standard for Business Process Modeling.* BoD, Norderstedt.

Ambler, S.W., 2003. *Business Rules.* `http://www.agilemodeling.com/artifacts/businessRule.htm`, 2003.

Dijkman, R. M., Dumas, M. and Ouyang, C., 2007. *Formal semantics and automated analysis of BPMN process models, preprint 7115.* Technical report, Queensland University of Technology, Brisbane, Australia.

Giurca, A., Gasevic, D. and Taveter, K., (Eds.), 2009. *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches.* Information Science Reference, Hershey, New York.

Ligęza, A., 2006. *Logical Foundations for Rule-Based Systems.* Springer-Verlag, Berlin, Heidelberg.

Ligęza, A., Nalepa G.J., 2011. A study of methodological issues in design and development of rule-based systems: proposal of a new approach. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **1**(2), 117–137, `http://dx.doi.org/10.1002/widm.11`.

Nalepa, G.J., Ligęza, A., 2010. HeKatE methodology, hybrid engineering of intelligent systems. *International Journal of Applied Mathematics and Computer Science*, **20**(1), 35–53, March 2010.

Negnevitsky, M., 2002. *Artificial Intelligence. A Guide to Intelligent Systems.* Addison-Wesley, Harlow, England, London, New York.

OMG, 2003. Production Rule Representation RFP. Technical report, Object Management Group, 2003.

OMG, 2011. Business Process Model and Notation (BPMN): Version 2.0 specification. Technical Report formal/2011-01-03, Object Management Group, January 2011.

Ouyang, C., Dumas, M., ter Hofstede, A.H. and van der Aalst, W.M., 2006a. From bpmn process models to bpel web services. *IEEE International Conference on Web Services (ICWS'06).*

Ouyang, C., van der Aalst, M.P., Dumas, M. and ter Hofstede, A.H., 2006b. *Translating BPMN to BPEL.* Technical report, Faculty of Information Technology, Queensland University of Technology.

Ross, R.G., 2006. The RuleSpeak Business Rule Notation. *Business Rules Journal*, **7**(4), April 2006.

Silver, B., 2009. *BPMN Method and Style.* Cody-Cassidy Press.

Stephen, W.A., Derek, M., 2008. *BPMN Modeling and Reference Guide: Understanding and Using BPMN.* Future Strategies Inc., Lighthouse Point, Florida, USA.