ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
"Antonio Ruberti"
CONSIGLIO NAZIONALE DELLE RICERCHE

A. De Nicola, M. Missikoff, M. Proietti, F. Smith

A LOGIC-BASED METHOD FOR BPMN DIAGRAMS VERIFICATION

R. 09-07, 2009

Antonio De Nicola – Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30, I-00185 Roma, Italy. Email: `antonio.denicola@iasi.cnr.it`.

Michele Missikoff – Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30, I-00185 Roma, Italy. Email: `michele.missikoff@iasi.cnr.it`.

Maurizio Proietti – Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30, I-00185 Roma, Italy. Email: `maurizio.proietti@iasi.cnr.it`.

Fabrizio Smith – Dipartimento di Ingegneria Elettrica e dell'Informazione, Università degli Studi de L'Aquila, I-67040 Monteluco di Roio, L'Aquila, Italy and Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30, I-00185 Roma, Italy. Email: `fabrizio.smith@iasi.cnr.it`.

This work was partially done under IASI support

ISSN: 1128–3378

Collana dei Rapporti dell'Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", CNR

viale Manzoni 30, 00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: `iasi@iasi.cnr.it`
URL: `http://www.iasi.cnr.it`

Abstract

In this report we illustrate a method to support the verification of business processes built by using the OMG standard BPMN. Such a standard is gaining a wide acceptance in the business domain, however, the lack of a precise, formally defined semantics leads to ambiguities and problems in the interpretations of the produced diagrams. In this respect, the support provided by the existing tools is not complete nor systematic. The report proposes a systematic method, based on a logic approach, referred to as BPAL, to provide a BPMN diagram the needed formal semantics for analysis and properties verification.

Keywords: Business Process, BPMN, Horn Logic, BPAL.

1 Introduction

Business Process (BP) management is constantly gaining popularity in various industrial sectors, especially in medium to large enterprises, and in the public administration. Among the proposed BP modeling standards, the BPMN (Business Process Modeling Notation)[1], proposed by the OMG, is gaining a growing popularity among the business experts. According to Wikipedia, "The primary goal of BPMN is to provide a standard notation that is readily understandable by all business stakeholders. These business stakeholders include the business analysts who create and refine the processes, the technical developers responsible for implementing the processes, and the business managers who monitor and manage the processes."

BP modeling is a complex human activity, requiring a special competence and, typically, the use of a BP design tool. Several tools are today available on the market, many of them are open source or free of charge[2]. Many of these t

ools are able to provide, besides a graphical editor supporting the creation of a Business Process Diagram (BPD), additional services, such as some forms of verification, the simulation of the designed processes, or the (semi) automatic generation of executable code (e.g., producing BPEL[3] code). The availability of the mentioned tools has further pushed the diffusion of the BPMN use both in the academic and in the industrial realities. But, despite the growing interest and penetration in the business domain, the BPMN still presents a number of drawbacks: some of the problems that the BP designer encounters are caused by the fact that its semantics is not defined in a precise manner. The absence of a rigorously defined semantics may cause that the same BPMN diagram behaves differently from one tool to another (e.g., be considered correct or not, generate different BPEL codes). The above shortcomings are widely recognized and the OMG is making an effort to correct them in the next version BPMN 2.0, where more space is dedicated to the definition of the semantics, but still not in a rigorous way.

In this report we have primarily the objective of providing a precise, formal semantics to the current (v1.2) BPMN proposal. To this end we adopt an approach, referred to as BPAL (Business Process Abstract Language), that aims at the same time to provide a solid formal foundation and a high level of practical usability, even in a business reality. The practical usability is guaranteed by the fact that our proposal intends to be associated to the existing tools, by adopting a semantic annotation approach that can be used as an "add-on" facility to the tool of your choice. In essence, BPAL does not propose an alternative environment: it

[1] www.bpmn.org
[2] See for instance: Intalio BPMS Designer, Tibco Business Studio, ProcessMaker, XBModeler
[3] Business Process Execution Language. See: http://www.oasis-open.org/specs/#wsbpelv2.0

allows business people to continue to work as usual, with their BPMN modeling tool, and then, they can progressively use BPAL to semantically enrich their diagrams.

In essence, the proposed approach gives the possibility to: (i) represent a BPMN diagram (BPD) with BPAL sentences (BPAL BP Schema); (ii) use the BPAL representation to check the conformance of the designed BP with a significant core of the BPMN standard; (iii) use the same formalism to verify if a trace, i.e., a log produced by a given BP, respects the execution conformance with respect to the intended semantics of the drawn BP. Furthermore, this approach will also support: (iv) the organization of a repository of BPDs and BPD fragments, to be queried and retrieved for future reuse; (v) the semantic annotation of the deployed BPs in term of a domain reference ontology; (vi) the analysis and the reengineering of existing BPs where the semantic annotation and the formal representation of the process can help in the identification of, e.g., overlappings, contradictions, missings.

The above results can be achieved only if the BPAL notation is based on formal grounds, with strong mathematical rooting. For the formal foundation of BPAL we adopted a Horn Logic approach, that allows us to express the key properties of the meta-model of BPMN and, furthermore, to semantically annotate a process schema built according to the former. This approach has the concrete advantage that a BPAL BP Schema can be easily translated into a Prolog set of clauses and executed on a reasoning engine capable of automatically proving a number of desirable properties.

What briefly outlined is a vast research program, in this report we intend to illustrate some of the preliminary results. To this end we concentrate on a core of BPMN constructs and show how to verify the first two points of the BPMN conformance specifications[4]. As indicated in the conclusion, we will address other problems, connected to the integrated use of business rules and the BP reengineering.

The rest of this report is organized as follows. In Section 2 related works are presented. A brief recap of the core BPMN addressed in this report and a running example are reported in Section 3. Section 4 presents the BPAL language and Section 5 BPAL meta-model and traces. In Section 6 some possible applications of BPAL in business process management are hinted. Finally, conclusions in Section 7 end the report.

2 Related Works

In literature much attention is given to BP modeling, since its application to the management of complex processes is an important issue in business organization. It appears increasingly evident that a good support to BP management requires reliable BP modeling methods and tools. Such reliability can be achieved only if the adopted method is based on

[4] http://www.omg.org/cgi-bin/doc?dtc/09-08-14

formal foundations. For this reason, our work is related to two main research directions, namely formal specification of workflow languages, for the verification and analysis of business processes, and semantic enrichment of process models, by using ontology-based techniques.

In [1], a formal semantics of BPMN is defined in terms of a mapping to Petri nets [2]. With respect to this work, we share the goal of overcoming the heterogeneity of BPMN constructs and the presence of ambiguity in definition of the notation, having a semantics that is only described in narrative form. The approach based on Petri Nets is inherently different from the BPAL proposal, since our operational semantics of the addressed core of BPMN is grounded in a FOL theory.

[3], [4] propose pure declarative approaches. A process is modeled by a set of *constraints* (business rules) that must be satisfied during execution: this is a partial representation that overlooks the control flow among activities, according to an imperative paradigm. [3] proposes the declarative flow language ConDec to define process models that can be represented as conjunction of Linear Temporal Logic formulae. This approach allows to verify properties by using model checking techniques. [4] proposes a verification method based on Abductive Logic Programming (ALP) and, in particular, the SCIFF framework [5], that is an ALP rule-based language and a family of proof procedures for specification and verification of event-based systems.

PSL (Process Specification Language) [6], is a language to support the exchange of process information among systems. A PSL ontology is organized into PSL-CORE and a partially ordered set of extensions. Axioms are first-order sentences written in the Knowledge Interchange Format (KIF). The PSL-CORE axiomatizes a set of intuitive semantic primitives (e.g., *activities*, *activity occurrences*, *time points*, and *objects*) allowing to describe the fundamental concepts of processes. There are two types of extensions within PSL: (1) *core theories* and (2) *definitional extensions*. Core theories introduce and axiomatize new relations and functions that are primitive. All terminology introduced in a definitional extension has conservative definitions using the terminology of the core theories. Thus, definitional extensions add no new expressive power to PSL-CORE.

[3], [4], [6] are based on rigorous mathematical foundations but they propose a paradigm shift from traditional process modeling approaches that is difficult to be understood and, consequently, to be accepted by business people. Conversely, with respect to these approaches, we propose a semantic enrichment of existing and widely used modeling techniques (in particular BPMN, but BPAL is easily applicable to UML activity diagrams, and EPC). Furthermore, our approach is not limited to formalize behavioral aspects of a business process (e.g., activity sequencing) but also to add semantics to specify the meaning of its entities (e.g., actors, input and output) in order to improve automation of business process management.

With respect to the above mentioned proposal, a further application of BPAL is the semantic annotation of BPs at an abstract level, along the line of the works presented in [7], [8], [9].

Finally, there are the ontology-based framework, such as OWL-S [10], WSDL-S [11], and WSMO [12]. They have a wider scope, aiming at modeling semantically rich web services, and have been conceived not directly connected to the business world but mainly to facilitate the automation of discovering, combining and invoking electronic services over the Web. Then, with respect to them, BPAL adopts the opposite strategy: instead of proposing a "holistic" approach that must be embraced as an exclusive choice, BPAL proposes a progressive approach: a business expert can start with the tool and notation of his/her choice and then, subsequently, proceed with BPAL semantic annotation to enrich the produced BP diagram and improve them with the help of the reasoning facilities proposed by the BPAL framework.

3 BPMN

In this section, we describe the main BPMN modeling constructs and we present a simple BPMN process to be used as a running example in the next sections of the report. Please recall that we do not intend to address the full BPMN standard (that, by the way, is evolving from its stable version 1.2 [13] to a major revision represented by version 2.0) but we are focusing on a significant subset.

3.1 Overview

BPMN provides a graphical notation for business process modeling. A BPMN specification [13] defines a Business Process Diagram (BPD), which is a kind of flowchart incorporating constructs tailored to business process modeling. Hereafter we consider a core subset of BPMN constructs, recalling a brief description of the selected core constructs

The constructs of BPMN are classified as *flow objects*, *artifacts*, *swimlanes*, and *connecting objects*.

Flow objects are *event*, *activity*, and *gateway*:

- An *event* is something that "happens" during the course of a business process. There are different types of event: *start*, *intermediate*, and *end* which can start, suspend, or end the process flow.
- An *activity* is a generic term representing some kind of work performed within a company. In BPMN, an activity can be atomic or compound.
- A *gateway* is a modeling construct used to control branching and merging of the process flow. There are:

1. *parallel gateways* to create and synchronize parallel flows;
2. *exclusive gateways* for selecting one of a set of mutually exclusive alternative flows and the subsequent merge of them;
3. *inclusive gateways* for selecting from a set of not mutually exclusive alternative flows and the subsequent merge of them.

Swimlanes are used to model business units (i.e., actors, such as organizations, functional units, or employees) able to activate or perform a process. They are *pools* and *lanes*. A pool represents a participant in a process at a high level of aggregation, while a lane is a sub-partition of a pool.

Connecting objects are *sequence flows* and *associations*. They model connections between flow objects. The *sequence flow* is used to introduce a (partial) order over process activities. An *association* is used to associate *artifacts* (e.g., objects) with *flow objects*, by showing inputs and outputs of activities.

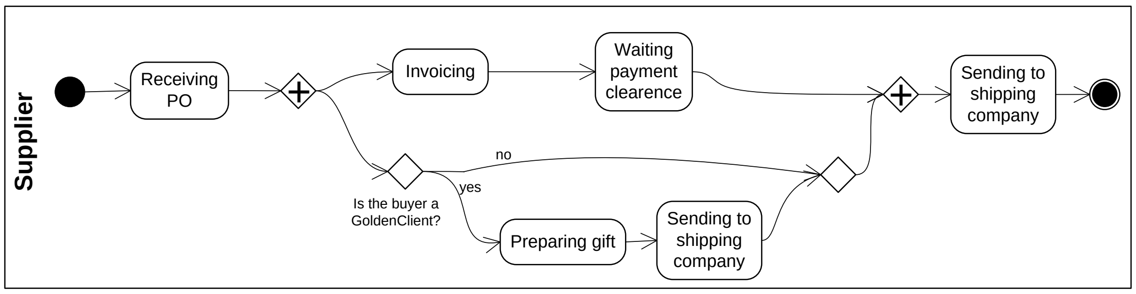
Supplier
Receiving PO
Invoicing
Waiting payment clearence
Sending to shipping company
no
yes
Is the buyer a GoldenClient?
Preparing gift
Sending to shipping company

Fig. 1. BPMN specification of a fragment of an eProcurement process

3.2 Running Example: an eProcurement Process

In concluding this section, we briefly present a fragment of an eProcurement process that will be used as a running example throughout the rest of the report.

An ACME supplier company receives a purchase order from a buyer and sends back an invoice. The buyer receives the invoice and makes the payment to the bank. In the meanwhile, the supplier sends a gift to the buyer if she/he is classified as golden client. After receiving the payment clearance from the bank, the supplier sends the goods to the buyer. The figure 1 illustrates a fragment of the eProcurement process from the supplier perspective.

4 The BPAL Approach for the Semantic Enrichment of BP Diagrams

An effective management of business processes requires a complex analyses of the business reality and the modeling of different kinds of knowledge. Besides the dynamic behavior of a business process, i.e., the execution flow represented by the activity sequencing, there are other relevant aspects, of a structural nature, such as actors associated to activities, events, managed objects, and their relationships. The proposed approach aims at providing a uniform and formal representation of both behavioral and structural knowledge related to a business process. Structural knowledge is formalized by an OPAL ontology [14] while behavioral knowledge is represented by a BPAL Process Schema (BPS), built on top of the former.

OPAL [14] is an ontology framework supporting business experts in building a structural ontology, i.e., where concepts are defined on the basis of their information structure and static relationships. In building an OPAL business ontology, the knowledge engineers typically start from a set of upper level concepts, and proceeds respecting a number of axioms that constraint the way an ontology can be implemented. The upper level concepts are:

- *OPAL_process*, representing a business activity or operation aimed at the satisfaction of a business goal, operating on a set of business objects;
- *OPAL_actor*, representing active elements of a business domain, able to activate, perform, or monitor a business process;
- *OPAL_object*, an entity on which a business process operates.

On top of OPAL there is BPAL [15] a formal language aimed at modeling behavioral aspects of a business process. Assuming that the basic knowledge has been specified by using OPAL, BPAL provides a set of modeling principles to capture the behavioral aspects of the modeled reality. BPAL constructs are derived from the BPMN standard and formalized in first order logic. Furthermore, BPAL provides a formalization of the meta-model (as a set of formation axioms and constraints) able to guarantee that a modeled business process is conformant with it (i.e., it is a well-formed BPS). Then, given a BPS, it is possible to generate a set of corresponding process traces or, when needed, to check if a given trace (i.e., BP execution) is conformant with the corresponding BPS.

For lack of space, in this report we focus on BPAL, since OPAL was already presented in [14].

4.1 The BPAL Language

The BPAL language consists of two syntactic categories: (i) a set *Const* of BPAL *constants* denoting entities to be used in the specification of a business process schema (e.g., business activities, actors, and objects) and (ii) a set *Pred* of *predicates* denoting relationships among BPAL entities. A BPAL business process schema is specified by a set of ground *facts* (i.e., atomic formulas) of the form $p(C_1, \dots, C_n)$, where $p \in Pred$ and $C_1, \dots, C_n \in Const.$

4.1.1 BPAL Entities

The BPAL constants are related to OPAL concepts by the *fulfills* relation. *fulfills(x, opal:y)* allows a bridge to be built between an OPAL ontology and a BPAL process schema. *fulfills* relates (i) a BPAL *object* with a specialization of *opal:object*, (ii) a BPAL *activity* with a specialization of *opal:process*, and (iii) a BPAL *actor* with a specialization of *opal:actor*. Basically, *fulfills* provides a semantic annotation of a process schema in terms of an ontology, by means of a semantic expression.

An example showing how an *activity ReceivingPurchaseOrder* and the corresponding OPAL concept are related is the following:

activity(ReceivingPO),
opal: ReceivingPurchaseOrder ISA opal:Process,
fulfills(ReceivingPO, opal: ReceivingPurchaseOrder)

4.1.2 BPAL Relationships

There are two kinds of predicates in BPAL: *unary* and *relational* predicates.

Unary predicates specify the types of the entities of a business process schema. In Figure 2, we present a hierarchy showing the BPAL unary predicates and a mapping to the corresponding BPMN notation. Please note that the BPMN notation is not available whenever the corresponding BPAL predicates are defined at an abstract level (e.g., *flow_el(el)*, *event(ev)*, and *gateway(gat)*). For the intended informal semantics of the BPAL predicates we refer to the one presented for BPMN constructs in Section 3. The formal semantics of BPAL will be introduced in Section 5.

The BPAL relational predicates describe the sequencing of *flow elements* in all possible executions of the process, and the relationships between *flow elements*, *objects*, and *actors*.

- *par_branch(gat,el1,el2)*: *gat* is a *parallel branch* point (i.e., a *parallel fork*) from which the business process branches to two sub-processes started by *el1* and *el2*. The two sub-processes started by *el1* and *el2* are executed in *parallel*;
- *inc_dec(gat,el1,el2)*: *gat* is an *inclusive decision* point from which the business process branches to two sub-processes started by *el1* and *el2*. At least one of the sub-processes started by *el1* and *el2* is executed[5];
- *exc_dec(gat,el1,el2)*: *gat* is an *exclusive decision* point from which the business process branches to two sub-processes started by *el1* and *el2*. Exactly one of the sub-processes started by *el1* and *el2* is executed;

[5] Note that gateways, in their general formulation, are associated with a condition. For instance, *exc_dec* will test a condition to select the path where the process flow will continue. Furthermore, for sake of concision, in this work we consider only binary gateways since n-ary can be easily reduced to them.

- *par_mrg(el1,el2,gat)*: *gat* is a *parallel merge* point (i.e., a *join*) where the two sub-processes ended by *el1* and *el2* are synchronized, that is, both sub-processes must be completed in order to proceed;
- *inc_mrg(el1,el2,gat)*: *gat* is an *inclusive merge* point. At least one of the two sub-processes ended by *el1* and *el2* must be completed in order to proceed;
- *exc_mrg(el1,el2,gat)*: *gat* is an *exclusive merge* point. Exactly one of the two sub-processes ended by *el1* and *el2* must be completed in order to proceed;
- *seq(el1,el2)*: the flow element *el1* is followed sequentially by *el2*;
- *involved_in(actor,el)*: the actor *actor* is associated with the flow element *el*;
- *input(act,obj)*: the object *obj* is an input of the activity *act*;
- *output (act,obj)*: the object *obj* is an output of the activity *act*.

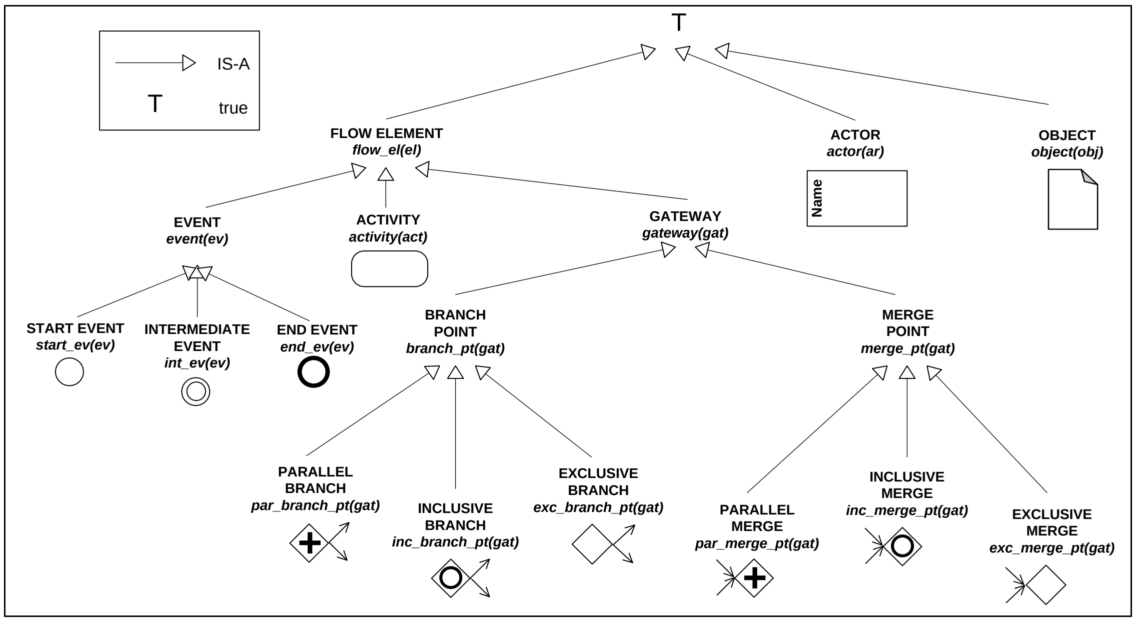
IS-A
T true
T
FLOW ELEMENT flow_el(el)
ACTOR actor(ar)
Name
OBJECT object(obj)
EVENT event(ev)
ACTIVITY activity(act)
GATEWAY gateway(gat)
START EVENT start_ev(ev)
INTERMEDIATE EVENT int_ev(ev)
END EVENT end_ev(ev)
BRANCH POINT branch_pt(gat)
MERGE POINT merge_pt(gat)
PARALLEL BRANCH par_branch_pt(gat)
INCLUSIVE BRANCH inc_branch_pt(gat)
EXCLUSIVE BRANCH exc_branch_pt(gat)
PARALLEL MERGE par_merge_pt(gat)
INCLUSIVE MERGE inc_merge_pt(gat)
EXCLUSIVE MERGE exc_merge_pt(gat)

Fig. 2. Hierarchy of the BPAL unary predicates

Below we present the BPAL process schema of the eProcurement application (Section 3.2).

start_ev(Start)	*seq(Start,ReceivingPO)*
activity(ReceivingPO)	*seq(ReceivingPO,Gat1)*
activity(Invoicing)	*seq(Invoicing,WaitingPaymentClearence)*
activity(WaitingPaymentClearence)	*seq(PreparingGift, SendingToShippingCompany1)*
activity(SendingToShippingCompany1)	*seq(Gat2,SendingToShippingCompany2)*
activity(PreparingGift)	*seq(SendingToShippingCompany2,End)*
activity(SendingToShippingCompany2)	*par_branch(Gat1,Invoicing,Gat3)*

par_branch_pt(Gat1)
par_merge_pt(Gat2)
exc_dec_pt(Gat3)
exc_merge_pt(Gat4)
end_ev(End)
	par_merge(WaitingPaymentClearence,Gat4,Gat2)
	exc_dec(Gat3,Gat4,PreparingGift)
	exc_merge(Gat3,SendingToShippingCompany1,Gat4)

5 BPAL Metamodel and Traces

There are inherent difficulties in using a language that, like BPMN, does not have a commonly agreed-upon formal semantics, nor a precisely defined execution environment. To overcome these limitations, we proposed to associate to a BPMN diagram a formal representation, based on BPAL, capable of providing the needed formal semantics. BPAL allows us to automatically prove two fundamental properties: the well-formedness of a BPAL process schema w.r.t. the BPAL meta-model and the correctness of a process trace w.r.t. a well-formed BPAL process schema. Seen the correspondence[6] of BPAL and BPMN, the proposed results are immediately transferred to a BPMN diagram. For our formalization we use standard notions of first order logic and logic programming [16].

5.1 BPAL Metamodel

In this section we describe the core of the metamodel of BPAL by means of a first order logic theory ***M***, which specifies when a business process schema is well-formed, i.e., it is correct from a syntactical point of view. The theory ***M*** consists of two sets of formulas: (1) a set ***K*** of first order formulas, called *schema constraints*, and (2) a set ***F*** of Horn clauses, called *formation axioms.* All formulas in ***M*** are universally quantified in front and, for reasons of simplicity, we will omit to write those quantifiers explicitly.

5.1.1 Schema Constraints

The set ***K*** of schema constrains consists of three disjoint subsets: (i) the *domain constraints*, (ii) the *type constraints*, and (iii) the *uniqueness constraints*.
Domain constraints are formulas expressing the relationships among BPAL unary predicates. For instance, some domain constraints assert that activities, events, and gateways belong to pairwise disjoint sets, e.g.:

$$activity(x) \rightarrow \neg\, event(x) \wedge \neg\, gateway(x)$$

[6] We are currently work on an automatic export of a BPMN diagram in BPAL form

Type constraints are formulas specifying the types of the relational predicates. For example, for the predicate *par_branch*, we have:

$$par\_branch(x,l,r) \rightarrow par\_branch\_pt(x) \wedge flow\_el(l) \wedge flow\_el(r)$$

Uniqueness Constraints are formulas expressing that the precedence relations between flow elements are specified in an unambiguous way. In particular, the following *sequential uniqueness axioms* state that by the *seq* predicate we can specify at most one successor and at most one predecessor of any flow element:

$$seq(x,y) \wedge seq(x,z) \rightarrow y=z \qquad seq(x,z) \wedge seq(y,z) \rightarrow x=y$$

Thus, in order to specify that the process flow branches, or merges, we cannot use the *seq* predicate, but we need to use one of the predicates *par_branch*, *inc_dec*, *exc_dec*, or *par_mrg*, *inc_mrg*, or *exc_mrg*, respectively. Similarly, in ***K*** there are: (i) *branching uniqueness constraints* asserting that every (parallel, inclusive, exclusive) branching point has exactly one pair of successors and (ii) *merging uniqueness constraints* asserting that every merge point has exactly one pair of predecessors.

5.1.2 Formation Axioms

The set ***F*** of Horn clauses defines a predicate *wf_process*(*s,e*) which holds if the business process started by the event *s* and ended by the event *e* is *well-formed*. Formation axioms can also be viewed as rules to construct well-formed process schemas.

F1. A business process schema is *well-formed* if (i) it is started by a start event *s* which has a successor *x*, (ii) it is ended by an end event *e* which has a predecessor *y*, and (iii) the sub-process from *x* to *y* is well-formed.

$$(start\_ev(s) \wedge seq(s,x) \wedge wf\_sub\_process(x,y) \wedge seq(y,e) \wedge end\_ev(e)) \rightarrow wf\_process(s,e)$$

The predicate *wf_sub_process*(*x,y*), specifying that the sub-process started by *x* and ended by *y* is well-formed, is defined by the following clauses.

F2. Any business activity *x* is a well-formed sub-process started and ended by *x* itself:

$$activity(x) \rightarrow wf\_sub\_process(x,x)$$

F3. Any intermediate event *x* is a well-formed sub-process started and ended by *x* itself:

$$int\_ev(x) \rightarrow wf\_sub\_process(x,x)$$

F4. A sub-process started by *x* and ended by *z* is well-formed if it can be decomposed into a sequence (*x*, *y*) and, recursively, a well-formed sub-process started by *y* and ended by *z*.

$$seq(x,y) \wedge wf\_sub\_process(y,z) \rightarrow wf\_sub\_process(x,z)$$

F5. A sub-process started by a parallel decision gateway x and ended by a flow element z is well-formed if it can be decomposed into (i) a branch from x to some flow elements l and r, (ii) two well-formed sub-processes started by l and r and ended by some flow elements m and r, respectively, (iii) a merge of m and r into a parallel merge gateway y, and (iv) a well-formed sub-process from y to z.

$(par\_branch(x,l,r) \land wf\_sub\_process(l,m) \land wf\_sub\_process(r,s) \land par\_mrg(m,s,y) \land wf\_sub\_process(y,z)) \rightarrow wf\_sub\_process(x,z)$

The clauses defining the predicate $wf\_sub\_process(x,y)$ in the cases where x is an inclusive or an exclusive decision gateway are similar and are omitted.

Both the formation axioms in $\boldsymbol{F}$ and the business process schema $\boldsymbol{B}$ are Horn clauses and, thus, the theory $\boldsymbol{F}\cup\boldsymbol{B}$ has a *least Herbrand model* [16], denoted by $lhm(\boldsymbol{F}\cup\boldsymbol{B})$. We say that $\boldsymbol{B}$ is well-formed if:

(i) every schema constraint C in $\boldsymbol{K}$ is true in $lhm(\boldsymbol{F}\cup\boldsymbol{B})$, and

(ii) for every start event S and end event E, $wf\_process(S,E)$ is true in $lhm(\boldsymbol{F}\cup\boldsymbol{B})$.

5.2 Business Process Traces

An execution of a business process is a sequence of instances of activities called *steps*; the latter may also represent instances of events. Steps are denoted by constants taken from a *Step* set disjoint from *Const*. Thus, a possible execution of a business process is a sequence $[s_1, s_2,\ldots, s_n]$, where $s_1, s_2,\ldots, s_n \in Step$, called a *trace*. The *instance* relation between steps and activities (or events) is specified by a binary predicate *instance*(*step,activity*). For example, *instance*(*RPO1*, *ReceivingPO)* states that the step *RPO1* is an activity instance of *ReceivingPO*.

A trace is *correct* w.r.t. a well-formed business process schema $\boldsymbol{B}$ if it is conformant to $\boldsymbol{B}$ according to the intended semantics of the BPAL relational predicates (as informally described in Section 4.1.2). Below we present a formal definition of the notion of a correct trace. Let us first give some examples by referring to the eProcurement application presented in Section 3.2. In Figure 3, we present again the BPMN specification of the eProcurement process where, for sake of conciseness, the name of each activity or event has been substituted by an upper case letter (e.g., “Receiving Purchase Order” has been substituted by “A”).

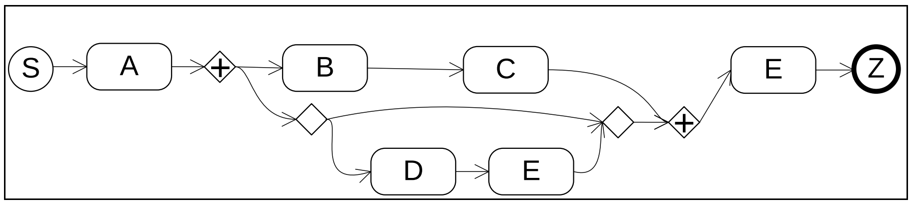

Fig. 3. Example of a business process schema.

Below we list two correct traces of the business process schema corresponding to the above BPMN specification:

- [*s*,*a*,*b*,*d*,*e*1,*c*,*e*2,*z*]
- [*s*,*a*,*b*,*c*,*e*2,*z*]

where a lower case letter represents a step which is an instance of the activity (or event) denoted by the corresponding upper case letter. Note that (i) the sub-trace [*b*,*d*,*e*1,*c*] of the first trace is an interleaving of the two sub-traces [*b*,*c*] and [*d*,*e*] which are instances of two parallel branches, and (ii) the second trace has been constructed by choosing the upper branch going out from the exclusive decision gateway.

We now introduce a predicate *trace*(*t*), which holds if *t* is a correct trace, with respect to a BPS, of the form $[s_1, s_2, \ldots, s_n]$, where s_1 is an instance of a start event and s_n is an instance of an end event. The predicate *trace*(*t*) is defined by a set ***T*** of Horn clauses, called *trace axioms*, which can also be viewed as rules for constructing correct traces. Each trace axiom corresponds to a formation axiom. For lack of space, we list below only the trace axioms corresponding to the formation axioms F1-F5 presented in Section 5.1.2. The trace axioms are defined by induction on the length of the trace *t*.

TA1. A sequence [*s1,x1,…,y1,e1*] of steps is a *correct trace* if: (i) *s1* is an instance of a start event, (ii) *e*1 is an instance of an end event, and (iii) [*x1,…,y1*] is a correct *sub-trace* from *x1* to *y1*:

$$(start\_ev(s) \wedge instance(s1,s) \wedge seq(s,x) \wedge instance(x1,x) \wedge sub\_trace(x1,t1,y1) \wedge instance(y1,y) \wedge seq(y,e) \wedge end\_ev(e) \wedge instance(e1,e) \wedge append([s1|t1],[e1],t)) \rightarrow trace(t)$$

where: (i) *sub_trace*(*x1*,*t1*,*y1*) holds iff *t1* is a correct trace where *x1* is the first step and *y1* is the last step, (ii) [*s1*|*t1*] is a sequence whose head is *s1* and tail is *t1*, and (iii) *append*(*t1*,*t2*,*t3*) holds iff *t3* is the concatenation of the sequences *t1* and *t2*.

The predicate *sub_trace(t)* is defined by induction on the length of the trace *t* as shown below.

TA2. Any instance *x1* of a business activity *x* is a correct sub-trace started and ended by *x1* itself.

$instance(x1,x) \wedge activity(x) \rightarrow sub\_trace(x1,[x1],x1)$

TA3. Any instance *x1* of an intermediate event *x* is a correct sub-trace started and ended by *x1* itself.

$instance(x1,x) \wedge int\_ev(x) \rightarrow sub\_trace(x1,[x1],x1)$

TA4. If *x1* is an instance of a flow element *x*, the successor of *x* is a flow element *y* which has an instance *y1*, and there is a correct sub-trace *t* from *y1* to *z1*, then [*x1*|*t*] is a correct sub-trace from *x1* to *z1*:

$instance(x1,x) \wedge seq(x,y) \wedge instance(y1,y) \wedge sub\_trace(y1,t,z1) \rightarrow sub\_trace(x1,[x1|t],z1)$

TA5. In the case where *x1* is an instance of a parallel branch point, the correctness of a sub-trace *t* from *x1* to *z1* is defined by the following clause:

$(instance(x1,x) \wedge instance(l1,l) \wedge instance(r1,r) \wedge par\_branch(x1,l1,r1) \wedge instance(m1,m) \wedge$
$sub\_trace(l1,t1,m1) \wedge instance(s1,s) \wedge sub\_trace(r1,t2,s1) \wedge interleaving(t1,t2,t3) \wedge$
$instance(y1,y) \wedge par\_mrg(m1,s1,y1) \wedge instance(z1,z) \wedge sub\_trace(y1,[y1|t4],z1) \wedge$
$append(t3,t4,t)) \rightarrow sub\_trace(x1,t,z1)$

where the predicate *interleaving*(*t1*,*t2*,*t3*) is defined by the following four clauses:

$interleaving([\ ],y,y)$
$interleaving(x,[\ ],x)$
$interleaving(x,y,z) \rightarrow interleaving([s|x],y,[s|z])$
$interleaving(x,y,z) \rightarrow interleaving(x,[s|y],[s|z])$

We omit, for lack of space, the clauses defining the predicate *sub_trace*(*x1*,*t*,*z1*) in the cases where *x1* is an instance of an inclusive or exclusive branch point.

For any business process schema ***B***, ***T***∪***B*** is a Horn theory and has a least Herbrand model *lhm*(***T***∪***B***). We say that a trace *t* is correct w.r.t. ***B*** if *trace*(*t*) is true in *lhm*(***T***∪***B***).

Note that ***T***∪***B*** can also be viewed as a Prolog program which can be used for several purposes, such as: (i) checking whether a given trace *t* is correct or not (by evaluating the query *trace*(*t*), where *t* is a ground sequence), (ii) checking whether there exists at least one correct trace (by evaluating the query *trace*(*t*), where *t* is a free variable), and (iii) generating *all* correct traces (by using the *findall* predicate).

6 Business Process Management with BPAL

Business Process Management a collection of methods and techniques to assist business practitioners and employees in the management of business processes along their entire life cycle (i.e., design and implementation, execution, and diagnosis) [17].

Our approach allows to access all the information required by an organization to perform effective decision making, relying on expressive, logic-based representation techniques. Since the description of the dynamic behavior of the process is enriched by an ontology-based specification of the involved entities, the presented framework allows queries as: "*get all the activities that involve the actor x and produce the object y as output*";*"get all business processes having activities that depend on system x"*; *"search generalization y of the object x and get all the activities that produce the object y as output"*.

Since knowledge to answer such queries is implicit, this kind of reasoning cannot be supported by current process modeling framework without using workarounds, like narrative descriptions of rules and conditions, spreadsheets and external tables, and additional tools that allow users to create hyperlinks to documents, meta-tags, and attribute fields.

Another issue related to the effective management of business process knowledge is related to the verification of properties that dynamic business process must satisfy. Going back to the example of Figure 1, a property to verify could be "is there any possible execution in which goods are sent to the shipping company before the clearance from the bank is received?".

Business Process Reengineering (BPR) is defined as "*the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service, and speed"* [18]. Two key phases of BPR are redesign and evaluation of process models. Currently, despite some interesting proposals (e.g., see [19]), redesign is mainly a manual process whereas evaluation is partially automated with adoption of simulation engines (e.g., Tibco Business Studio[7]).

For this reason, a promising application of the BPAL approach is to provide a semantics-based support to redesign process models by reasoning over structural knowledge represented in the OPAL ontology. In Section 4.1.1, we already stated that semantic annotation of a BPAL process model in the terms of an ontology is guaranteed by the *fulfills* relationship.

The idea presented in this section is to support business experts performing BPR by automating existing BPR best practices, as those described in [20]. [20] identifies 29 best practices and classifies them according to the business process aspect that benefits from

[7] Tibco Business Studio: http://developer.tibco.com/business_studio (Accessed on the 24th November 2009)

their application: customers, business process operation, business process behavior, organization, information, technology, and external environment. Since a complete discussion on this application is out of the scope of this report, we just give some hints that will be further elaborated in future work.

In particular, we consider an *organization best practice*, named "*flexible assignment*". This best practice is devoted to assign resources in such a way that guarantees maximal flexibility for the near future. For example, according to [20], if a task can be executed by either of two available resources, it should be assigned to the most specialized resource. In this way, the possibilities to have the free, more general resource executing another task are maximal. By using the BPAL approach, a *bpal:Actor* is annotated by an *opal:Actor*. An automatic support to a business consultant, performing the flexible assignment, can be provided by proposing to him alternative actors. In fact, they can be discovered in the OPAL ontology, for instance, by browsing the specialization hierarchy of *opal:Actor*s, to find the most specialized concept (as required by the flexible assignment best practice). Another approach is to calculate the semantic similarity[8] between the given *opal:Actor* and the other *opal:Actor*s. Then, semantic similarity results are ranked and proposed to business consultants that can select the most appropriate actor.

7 Conclusions and future work

In this report we presented a systematic approach, based on the BPAL representation language, to provide a semantic enrichment to BPMN diagrams. The proposed approach is intended to complement the existing BPMN diagramming tools, so that business people can enrich their BPD with a formal, logic-based notation and take advantage of the reasoning possibilities that the approach offers.

This report mainly intends to lay the foundations of the BPAL method for a systematic semantic enrichment of BPDs. The proposed solution establishes a logic-based framework that we intend to expand in several directions. The first direction concerns the Business Rules (BR)s. In real world applications the operations of an enterprise is regulated by a set of BPs that are often integrated by specific business rules. We intend to develop an extended framework where BPs and BRs are integrated and jointly analyzed to check if, for instance, there are processes that violate a newly issued rule. A second direction will be the tight integration of the structural knowledge, represented by OPAL in our work, and the behavioral knowledge, represented by BPAL. OPAL has been formalized, according to the

[8] Given two concepts c_i and c_j, their similarity, *consim*(c_i,c_j), is defined as the maximum information content shared by the concepts divided by the information contents of the two concepts *consim*(c_i,c_j) function [21].

tradition, i.e., in Description Logic; its integration with BPAL requires its reformulation in Horn logic.

On an engineering ground, we intend to investigate the problem of Business Process Reengineering, and explore the possibility of manipulating a set of business processes to produce a new, optimized (e.g., in terms of process length or aggregating sub-processes that are shared by different BPs) set of reengineered BPs. Another practical problem is the automatic generation of a BPAL specification starting from the export format of a BPMN diagramming tool. This service will make the logic-based support of BPAL fully transparent to BP analysts.

Acknowledgements

This work is partially supported by the Tocai Project (http://www.dis.uniroma1.it/ ~tocai/), funded by the FIRB Programme of the Italian Ministry of University and Research (MIUR).

References

1. Dijkman, R.M., Dumas, M., Ouyang, C.: Formal semantics and automated analysis of BPMN process models. In: Preprint 7115. Queensland University of Technology, Brisbane, Australia (2007).
2. W. Reisig and G. Rozenberg, editors. Lectures on Petri Nets I: Basic Models,volume 1491 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1998.
3. Pesic, M., van der Aalst, W.M.P: A Declarative Approach for Flexible Business Processes Management. In BPM 2006 Workshops, LNCS 4103. pp. 169-180, 2006.
4. Montali, M., Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verification from Declarative Specifications Using Logic Programming. In ICLP 2008, LNCS 5366, pp. 440–454, 2008.
5. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: the SCIFF framework. ACM Transactions on Computational Logics , 9(4):1-43, 2008.
6. Conrad, B., Gruninger, M.: Psl: A semantic domain for flow models. Software and Systems Modeling, 4(2):209–231, May 2005.
7. Dimitrov, M., Simov, A., Stein, S., Konstantinov, M.: A BPMO based Semantic Business Process Modelling Environment. In Proc. of the Workshop on Semantic Business Process and Product Lifecycle Management at the ESWC, volume 251 of CEUR-WS, 2007.
8. Weber, I., Hoffmann, J., Mendling, J.: Semantic business process validation. In: Proceedings of the SBPM Workshop, 2008.

9. Francescomarino, C.D., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P.: Reasoning on Semantically Annotated Processes. In: Proceedings of the 6th International Conference on Service Oriented Computing (ICSOC'08), Sydney, Australia (2008).
10. W3C: OWL-S, Semantic markup for web services. 22 november 2004, http://www.w3.org/Submission/OWL-S/.
11. W3C: Web Service Semantics. 7 /11/2005, http://www.w3.org/Submission/WSDL-S.
12. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web Service Modeling Ontology. Applied Ontology, 1(1): 77-106, IOS Press, 2005.
13. OMG: Business Process Model and Notation. Version 1.2, January 2009, http://www.omg.org/spec/BPMN/1.2
14. D'Antonio, F., Missikoff, M., Taglino, F.: Formalizing the OPAL eBusiness ontology design patterns with OWL. In the 3rd I-ESA Conference, 2007.
15. De Nicola, A., Missikoff, M., Tininini, L.: Process Composition in Logistics: An Ontological Approach. Interop-ESA 2008 Conference. Berlin, Germany, March, 2008.
16. Lloyd, J.W.: Foundations of Logic Programming. Springer-Verlag, Berlin, 1987.
17. Dumas M., van der Aalst W., ter Hofstede A.H.M.: Process-Aware Information Systems. WILEY-INTERSCIENCE, 2005.
18. Hammer M, Champy J.: Reengineering the corporation: a manifesto for business revolution. Harper Business Editions; New York (1993).
19. Castano S., De Antonellis V., Melchiori M.: A Methodology and Tool Environment for Process Analysis and Reengineering. Data Knowl. Eng. 31(3): 253-278 (1999).
20. Reijers, H. A., Limam Mansar, S: Best practices in business process redesign: An overview and qualitative evaluation of successful redesign heuristics. Omega, 33(4), 283–306, 2005.
21. Formica A., Missikoff M., Pourabbas E., Taglino F.. Weighted Ontology for Semantic Search. R. Meersman and Z. Tari (Eds.): OTM 2008, Part II, LNCS 5332, pp. 1289–1303, 2008.