Querying Business Processes and Ontologies in a Logic Programming Environment

(Extended Abstract)

Michele Missikoff, Maurizio Proietti, Fabrizio Smith

IASI-CNR, Viale Manzoni 30, 00185, Rome, Italy
`{michele.missikoff,maurizio.proietti,fabrizio.smith}@iasi.cnr.it`

1 Introduction

In recent years there has been an acceleration towards new forms of cooperation between enterprises, such as virtual enterprises, networked enterprises, or business ecosystems. A networked enterprise integrates the resources and Business Processes (BPs) of the participating organizations allowing them to operate as a unique (vitual) organization. In particular, starting from a set of BPs that exist in the various participating enterprises, the objective is to build a global BP by selecting the local BPs to be included. This operation is not an easy one, since the local BPs are often built by using different tools, according to different business logics, and using different labels and terminology to denote activities and resources. To this end, the various participating enterprises need to agree on a common view of the business domain, and provide descriptions of the local BPs according to such agreed common view. Much work has been done[1] towards the enhancement of BP management systems [1] by means of well-established techniques from the area of the Semantic Web and, in particular, computational ontologies [2]. An enterprise ontology supports unambiguous definitions of the entities occurring in the domain, and eases the interoperability between software applications and the reuse/exchange of knowledge between human actors.

In this frame, we focus on the problem of querying repositories of semantically annotated BPs. The proposed solution is based on a synergic use of an ontological framework (OPAL [3]) aimed at capturing the semantics of a business scenario, and a business process modelling framework (BPAL [4]) to represent the workflow logic. Then, the semantic annotation of BPs w.r.t. ontologies allows us to query BPs in terms of the ontology vocabulary, easing the retrieval of local BP (or process fragments) to be reused in the composition of new BPs. Figure 1 depicts a birds-eye view of the querying approach, with the local BP repositories ($LBPR_X$), the common set of ontologies and vocabularies (Reference Ontology) used for the semantic annotation (Σ_x) of the BP repositories, and the query engine operating on the above structures.

[1] See, e.g., the SUPER (http://www.ip-super.org/), COIN (http://www.coin-ip.eu/) and PLUG-IT (http://plug-it.org/) initiatives.

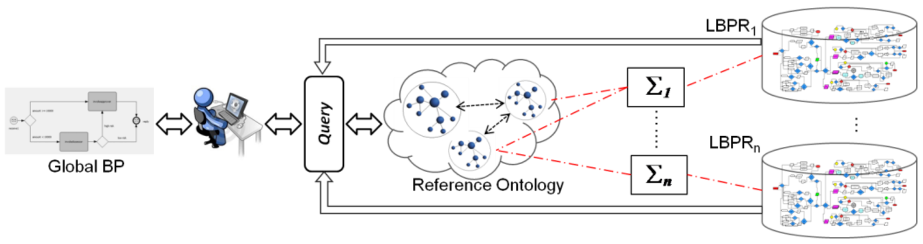


Fig. 1. Business Process Querying Approach

The proposed approach provides a uniform and formal representation framework, suited for automatic reasoning and equipped with a powerful inference mechanism supported by the solutions developed in the area of Logic Programming [5]. At the same time it has been conceived to be used in conjunction with the existing BP management tools as an ‘add-on’ to them, by supporting BPMN [6] and in particular its XPDL [7] linear form as a modeling notation and OWL [8], for the definition of the reference ontologies.

2 Knowledge Representation Framework

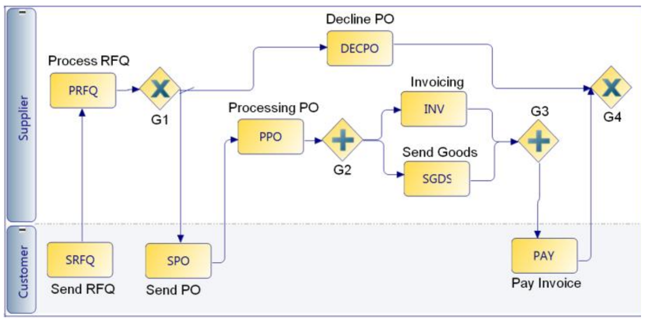
In this section we introduce the knowledge representation framework which is at the basis of the querying approach that will be proposed in Section 3. In this framework we are able to define an *Enterprise Knowledge Base* (EKB) as a collection of logical theories where: *i)* the representation of the *workflow graph* associated with each BP, together with its *behavioral semantics*, i.e., a formal description of its execution, is provided by a BPAL specification; *ii)* the representation of the *domain knowledge* regarding the business scenario is provided through an OPAL ontology.

2.1 Introducing BPAL

BPAL [4] is a logic-based language that provides a declarative modeling method capable of fully capturing the procedural knowledge in a business process. Hence it provides constructs to model activities, events, gateways and their sequencing. For branching flows, BPAL provides predicates representing *parallel* (AND), *exclusive* (XOR), and *inclusive* (OR) *branching/merging* of the control flow. A BPAL BP Schema (BPS) describes a workflow graph through a set of *facts* (ground atoms) constructed from the BPAL alphabet. In Figure 2 an exemplary BPS modeled in BPMN is depicted, together with the corresponding BPAL translation.

In order to perform several reasoning tasks over BPAL BPSs, three core theories have been defined, namely the meta-model theory ***M***, the trace theory ***TR*** and the dependency constraint theory ***D***. ***M*** formalizes a set of structural properties of a BPS, that at this level is regarded as a labeled graph, to define how the constructs provided by the BPAL language can be used to build a *well-formed* BPS. Two categories of properties should be verified by a well-formed BPS: *i) local* properties related to the elementary components of the workflow graph (for instance, every activity must have at most one ingoing and at most one outgoing sequence flow), and *ii) global* properties related to the overall structure of the process (for instance, in this paper we assume that processes are *structured*, i.e., each branch point is matched with a merge point of the same type, and such branch-merge pairs are also properly nested).

TR provides a formalization of the trace semantics of a BP schema, where a *trace* models an execution (or instance, or enactment) of a BPS as a sequence of occurrences of activities called *steps*. ***D*** is introduced for the purpose of efficiently verifying properties regarding the possible executions of a BPS. ***D*** defines properties in the form of constraints stating that the execution of an activity is dependent on the execution of another activity, e.g., two activities have to occur together (or in mutual exclusion) in the process (possibly, in a given order). Examples of such constraints are *i) precedence(a,b,p,s,e)*, i.e., in the sub-process of *p* starting with *s* and ending with *e*, if *b* is executed then *a* has been previously executed; *ii) response(a,b,p,s,e), i.e.*, in the sub-process of *p* starting with *s* and ending with *e*, if *a* is executed then *b* will be executed. In a structured BPS, like the ones considered in this paper, such constraints could be verified by an exhaustive exploration of the set of correct traces. However, this approach would be inefficient, especially when used for answering complex queries of the kind described in Section 3. Thus, we follow a different approach for defining the constraint patterns discussed in [9] by means of logic rules that infer the absence of a counterexample. The set of these rules constitutes the theory ***D***. This approach is indeed more efficient because, in order to construct a counterexample, we can avoid to actually construct all possible interleavings of the traces generated by the execution of parallel sub-processes and, in fact, we only need to perform suitable traversals of the workflow graph.



BPAL BPS

activity(SPO)
activity(PPO)
exc_branch_pt(G1)
par_branch_pt(G2)
…
seq(PPO,G2)
exc_branch(G1,SPO,DECPO)
exc_merge(DECPO,PAY,G4)
par_branch(G2,INV,SGDS)
par_merge(INV,SGDS,G3)
…

Fig. 2. BPMN eProcurement Process (left-side), partial BPAL translation (right-side)

2.2 Semantic Annotation through a Business Reference Ontology

For the design of a *Business Reference Ontology* (**BRO**) to be used in the alignment of the terminology and conceptualizations used in different BP schemas, we consider as the reference framework the OPAL methodology [3]. **OPAL** organizes concepts through a number of meta-concepts aimed at supporting the domain expert in the conceptualization process, identifying active entities (*actors*), passive entities (*objects*), and transformations (*processes*). The latter are represented only in terms of their information structure and static relationships, without modeling behavioral issues, i.e., sequencing of activities, for which BPAL is delegated to. OPAL concepts may be defined in terms of concepts described in an ontology (or set of ontologies) describing a specific domain (or set of domains). Then the BRO is composed by an OPAL model linked to a set of domain ontologies, that can be already existing resources or artifacts developed on purpose.

The *Semantic Annotation* **Σ** defines a correspondence between elements of a BPS and concepts of a BRO, in order to describe the meaning of the former through a suitable conceptualization of the domain of interest provided by the latter in terms of related *actors, objects,* and *processes*. **Σ** is specified by the relation *σ*, which is defined by a set of assertions of the form *σ(El,C),* where *El* is an element of a BPS and *C* is an OPAL concept.

Technically, the language adopted for the definition of a BRO is a fragment of OWL, falling within the OWL-RL profile. OWL-RL, is an OWL subset designed for practical implementations using rule-based techniques. In the *EKB,* ontologies are encoded using the triple notation by means of the predicate *t(s,p,o)*, representing a generalized RDF triple (with subject *s*, predicate *p*, and object *o*). For the semantics of an OWL-RL ontology we refer to the axiomatization (OWL 2 RL/RDF rules) described in [8].

Figure 3 reports an example of semantic annotation related to the eProcurement process of Figure 2, where a basic definition in terms of *inputs*, *outputs* and related *actors* is provided for *IssuingPO* (we assume the usual prefixes *rdfs* and *owl* for the RDFS/OWL vocabulary, plus *opal* for the introduced vocabulary and *bro* for the specific example).

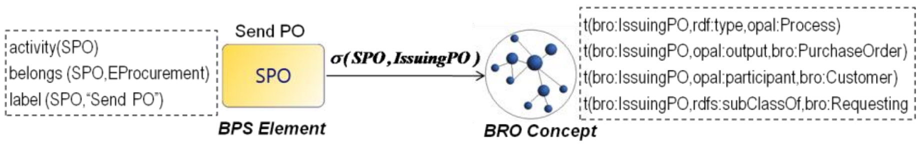

activity(SPO)
belongs (SPO,EProcurement)
label (SPO,"Send PO")
Send PO
SPO
BPS Element
σ(SPO, IssuingPO)
BRO Concept
t(bro:IssuingPO,rdf:type,opal:Process)
t(bro:IssuingPO,opal:output,bro:PurchaseOrder)
t(bro:IssuingPO,opal:participant,bro:Customer)
t(bro:IssuingPO,rdfs:subClassOf,bro:Requesting

Fig. 3. Semantic Enrichment of Process Schemas

3 Querying an Enterprise Knowledge Base

An *EKB* is formalized by a First Order Logic theory, defined by putting together the theories introduced in the previous section:

$$\boldsymbol{EKB} = \boldsymbol{BRO} \cup \boldsymbol{OWL\_RL} \cup \boldsymbol{\Sigma} \cup \boldsymbol{M} \cup \boldsymbol{B} \cup \boldsymbol{TR} \cup \boldsymbol{D}$$

where: *i)* ***BRO*** ∪ ***OWL_RL*** ∪ ***Σ*** represents the *domain knowledge*, i.e., ***BRO*** is an OPAL Business Reference Ontology, encoded as a set of triples of the form *t(s,p,o)*; ***OWL_RL*** is the OWL 2 RL/RDF rule set, included into the *EKB* to support reasoning over the *BRO*; and ***Σ*** is a semantic annotation, including a set of assertions of the form *σ(El,C); ii)* ***M*** ∪ ***B*** represents the *structural knowledge* about the business processes, i.e., ***M*** is the meta-model theory and ***B*** is a *repository* consisting of a set of BP schemas defined in BPAL; *iii)* ***TR*** ∪ ***D*** is a formalization of the *behavioral semantics* of the BP schemas, i.e., ***TR*** is the trace theory and ***D*** is the theory defining the dependency constraints.

A relevant property of the *EKB* is that it has a straightforward translation to a logic program [5], which can be effectively used for reasoning within a Prolog environment. This translation allows us to deal within a uniform framework with several kinds of reasoning tasks and combinations thereof. Every component of the *EKB* defines a set of predicates that can be used for querying the knowledge base. The reference ontology ***BRO*** and the semantic annotation **Σ** allow us to express queries in terms of the ontology vocabulary. The predicates defined by the meta-model theory ***M*** and by the BP schemas in ***B*** allow us to query the schema level of a BP, verifying properties regarding the flow elements occurring in it (*activities, events, gateways*) and their relationships (*sequence flows*). Finally ***TR*** and ***D***, allow us to express queries about the behavior of a BP schema at execution time, i.e., verify properties regarding the execution semantics of a BP schema.

In order to provide the user with a simple and expressive query language that does not require to understand the technicalities of the logic engine, we propose *QuBPAL*, a query language based on the SELECT-FROM-WHERE paradigm (see [10] for more details) that can be translated to logic programs and evaluated by using the XSB engine (*http://xsb.sourceforge.net*). More specifically, QuBPAL queries which do not involve predicates defined in ***TR***, i.e., queries that do not explicitly manipulate traces, are translated to logic programs belonging to the fragment of Datalog with stratified negation. For this class of programs the tabling mechanism of XSB guarantees an efficient, sound and complete top-down evaluation. As an example, below we report a *QuBPAL* query and its corresponding Datalog translation. We prefix variables names by a question mark (e.g., *?x*) and we use the notation *?x::Conc* to indicate the semantic typing of a variable, i.e., as a shortcut for *σ(x,y)* ∧ *t(y,rdfs:subClassOf,Conc)*, in order to easily navigate the ontology taxonomy.

```
SELECT <?p,?s,?e>
WHERE activity(?s::bro:Requesting) AND belongs(?b::bro:FinancialTransaction,?p,?s,?e) AND
precedence(?a::bro:Invoicing,?b,?p,?s,?e)
```

```
q(P,S,E):- t(C_1,rdfs:subClassOf,bro:Requesting),t(C_2,rdfs:subClassOf,bro:FinancialTransaction),
t(C_3,rdfs:subClassOf,bro:Invoicing),σ(S,C_1),σ(B,C_2),σ(A,C_3),belongs(S,P),belongs(E,P),
belongs(A,P,S,E),belongs(B,P,S,E), wf_subproc(P,S,E),precedence(A,B,P,S,E).
```

This query returns every well-formed process fragment (i.e., structured block) that starts with a *requesting* activity and that contains a *financial transaction* preceded (in every possible run) by an *invoicing*. The *SELECT* statement defines the output of the query evaluation, which in this case is a process fragment identified by the triple *<?p,?s,?e>*, where *?p* is a BP identifier, *?s* is the starting element, and *?e* is the ending element. The query may include a *FROM* statement (absent in the above example), indicating the process(es) from which data is to be retrieved (possibly the whole repository). In the *WHERE* statement it can be specified an expression which restricts the data returned by the query, built from the set of predicates defined in the *EKB*, the = predicate and the onnectives AND, OR, NOT with the standard logic semantics. If we consider the process fragment of Section 2.1, the answer to the above query contains the sub-process starting with SPO and ending with PAY.

This query shows the interplay of the different components of the *EKB*: the notions of well-formed process fragment (*wf_subproc*) and containment (*belongs*) are formalized in the BPAL meta-model theory, *precedence* is a dependency constraint regarding the behavioral semantics of the BPS, σ and *t* are defined in terms of the semantic description of the domain specified in the BRO.

4 Implementation

A prototype of the proposed framework has been implemented as a Java application, interfaced with the XSB logic programming engine through the Interprolog library (*http://www.declarativa.com/interprolog*). The population of an *EKB* is based on two modules: *i) XPDL2BPAL* to import a process repository ***B*** from XPDL files *ii) OWL2LP,* based on the Jena2 toolkit (*http://jena.sourceforge.net/*), to import the reference ontology ***BRO*** and the semantic annotation ***Σ*** from OWL documents by a translation into a set of ground facts in the triple notation. The *EKB* is then completed by the Prolog programs encoding the meta-model theory ***M***, the trace theory ***TR***, the dependency constraints ***D*** and the OWL 2 RL/RDF rule set ***OWL_RL.*** Having populated the *EKB,* the reasoning tasks are performed by querying the knowledge base through *QuBPAL* queries that are translated into Datalog by the module *QuBPAL2LP* and evaluated by the XSB engine. Finally, the computed results can be exported through the *XpdlWriter* module as a new XPDL file, for its visualization in a BPMS and its further reuse.

We conducted in [10] a preliminary evaluation of the system performance on a desktop machine (Intel Core2 E4500 CPU (2x2.20 GH), 2GB of RAM), to show the feasibility of the approach. In particular, the rule-based implementation of the OWL reasoner and the effective goal-oriented evaluation mechanism of the Prolog engine shown good response time and significant scalability. The results are summarized in Table 1. Timings are expressed in seconds and represent the average value over 10 runs. We generated artificial XPDL files, describing three BP repositories, **T1-T3** of different size and structure. In the first part of Table 1 we report, for each repository, the number of BPs, the total size, i.e. the total number of flow elements, the total number of gateways and the size of the smallest and biggest BP. As Business Reference Ontology we created an eProcurement ontology (about 400 named concepts described by about 2500 triples), by including part of the OWL translation of the SUMO ontology (*http://www.ontologyportal.org/translations/ SUMO.owl*). In particular, we used the *Process* hierarchy introduced in SUMO as root for the activity taxonomy (about 250 concepts) adopted for the random annotation of the generated BPs. First, we tested the set up phase (middle part of Table 1), by importing into the platform each repository from an XPDL file, the ontology and the semantic annotation from OWL. Then, we performed three queries **Q1-Q3** against each repository. Q1 is analogous to the one shown in Section 3. Q2 *retrieves every opal:Object that is related to a concept used for the annotation of an activity lying on* *a path from an activity annotated with B to an activity annotated with C.* Q3 *retrieves every sub-process that is executed as an alternative to one where an activity annotated with C is eventually executed*. We report for each run (bottom part of Table 1) the number of results obtained and the total time spent for the evaluation, including the QuBPAL query translation (*QuBPAL2LP*), the communication overhead between Java and XSB and the export of the results as a new XPDL file (*XpdlWriter*).

Table 1. Evaluation Results

Test Data Sets					
	Nr. of BPS	***Tot. Size***	***Nr. of Gateways***	***Min BPS Size***	***Max BPS Size***
T1	50	11757	4114	172	308
T2	100	18888	6442	157	237
T3	200	25229	8556	104	164

Set Up Phase Evaluation						
	BP Repository Import		**BRO Import**		**Σ Import**	
	XPDL2BPAL	*XSB Compile*	*OWL2LP*	*XSB Compile*	*OWL2LP*	*XSB Compile*
T1	3.6	7.4	1	0.7	1.8	1.2
T2	7.8	11.2	1	0.7	2.5	1.7
T3	15.3	18	1	0.7	3.3	2.5

Run Time Phase Evaluation						
	Q1		**Q2**		**Q3**	
	Nr. of Res.	*Time*	*Nr. of Res.*	*Time*	*Nr. of Res.*	*Time*
T1	11	2.5	133	4.8	47	10.2
T2	15	5.3	125	11.3	66	14.7
T3	9	8	109	17.2	44	16.9

5 Related Work and Conclusions

In this paper we presented a framework conceived to complement existing BPMS by providing advanced querying services. The proposed solution is based on a synergic use of ontologies to capture the semantics of a business scenario, and a business process modelling framework, to represent the underlying application logic. Both frameworks are seamlessly connected thanks to their grounding in logic programming and therefore it is possible to apply effective reasoning methods to query the knowledge base encompassing the two.

A first body of related works is based on the use of techniques developed in the context of the semantic web that have been extended to business process management. Relevant work in this field has been done within the SUPER project (http://www.ip-super.org/), where several foundational ontologies to model functional, organizational, informational and behavioral perspectives have been developed. In [11] a querying framework based on such ontologies is presented. Other approaches based on meta-model ontologies have been discussed, e.g., [12,13]. Unlike the aforementioned works, where the behavioral aspects are hidden or abstracted away, properties defined in terms of the execution semantics can be used in a QuBPAL query. Hence, the *EKB* provides a homogeneous framework where one can evaluate complex queries that combine properties related to the ontological dscription, the workflow structure, and the behavioral semantics of the modeled BPs.

Other approaches for BP querying are grounded in graph matching, through visual languages [14,15] grounded in graph grammars. Such approaches allow the user to query the graph representation of a process workflow in an intuitive way, but they need to be combined with external tools to reason about properties of the behavioral semantics (e.g., [14] implements translations to finite state models to be verified by using model checking techniques). Our framework not only provides a method based on Datalog for querying the structure of the workflow graph, but due to the logic-based representation it also integrates additional reasoning services. In particular, a very relevant advantage provided by our approach is the possibility of formulating queries involving the knowledge represented in domain models formally encoded by means of ontologies, hence: *i)* decoupling queries from specific processes, *ii)* overcoming semantic heterogeneities deriving, e.g., from different terminologies, *iii)* posing queries at different generalization levels, taking advantage of the semantic relations defined in the ontology, such as *subsumption*.

Future works are intended to increase the expressivity of the approach, by supporting a larger number of workflow patterns [1], and to perform the optimization of the query evaluation process, that can be strongly improved by exploiting query rewriting techniques.

6 References

1. ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., Russell, N.: Modern Business Process Automation: YAWL and its Support Environment. Springer, 2010.
2. Hepp, M., et al: Semantic business process management: A vision towards using semantic web services for business process management. Proc. ICEBE 2005.
3. De Nicola A., Missikoff M., Navigli R.: A software engineering approach to ontology building. *Information Systems*, 34(2):258--275(2009).
4. De Nicola, A., Missikoff, M., Proietti, M., Smith, F.: An Open Platform for Business Process Modeling and Verification. Proc. DEXA 2010. LNCS 6261, pp. 66--90, Springer, 2010.
5. Lloyd, J.W.: Foundations of Logic Programming. Springer-Verlag, Berlin, 1987. 2nd Ed.
6. OMG: Business Process Model and Notation, http://www.omg.org/spec/BPMN/2.0.
7. XPDL 2.1 Complete Specification, http://www.wfmc.org/xpdl.html.
8. OWL 2: Profiles, http://www.w3.org/TR/owl2-profiles.
9. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. Proc. ICSE'99, pp. 411-420, 1999.
10. Missikoff, M., Proietti, M., Smith, F.: Querying semantically annotated business processes, IASI-CNR, R. 10-22, 2010.
11. Markovic, I. Advanced Querying and Reasoning on Business Process Models. Proc. BIS 2008. LNBIP 7, pp.189--200, Springer, 2008.
12. Di Francescomarino, C., Tonella, P.: Crosscutting Concern Documentation by Visual Query of Business Processes. Business Process Management Workshops 2008.
13. Haller, A. Gaaloul, W., Marmolowski, M.: Towards an XPDL Compliant Process Ontology. SERVICES I 2008, pp.83-86, 2008.
14. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. Proc. BPM 2008. LNCS 5240, pp. 326--341. Springer, 2008.
15. Beeri, C., Eyal, A., Kamenkovich, S., and Milo, T.: Querying business processes with BP-QL. *Information Systems*. 33, 6 (Sep. 2008), 477-507.