Chapter 7
Reasoning About Process Models: What Description Logic Offers to Business Process Model Analysis

Michael Fellmann

Abstract Business process models are important for the design and implementation of process-aware information systems. Up to now, process models are represented predominantly as semi-formal models. Such models rely on the natural language to describe the models content via labels associated to the model elements. Due to the ambiguities of natural language, the semantics thus is not clear and well-defined. This in turn leads to problems when analyzing process models such as misinterpretations by humans or incomplete answers to queries by machines. In order to tackle this challenge, description logic-based ontologies provide well-defined semantics and can be used to represent graph-like knowledge structures such as business process models. Yet, up to now the capabilities of modern ontology languages are not widely used to represent, query and reason about process models. Therefore, the chapter presents an amalgamation of process models with ontologies. This amalgamation is formed by process models being represented in an ontology and being annotated with further elements of that ontology. In this way, the process model elements are augmented by machine processable semantics. By means of a concrete example, it is illustrated which deductions can be inferred using standard reasoning engines. With this, "intelligent" answers to queries executed against the knowledge base containing the process knowledge are possible that advance the model-based design of process aware information systems. Finally, an existing tool is briefly presented as a proof-of-concept. It allows creating and querying the ontology-based representation. This chapter is an excerpt of the introductory part of [1] which has been extended and revised.

M. Fellmann (✉)
Information Systems Research Group, University of Rostock,
Faculty of Computer Science and Electrical Engineering,
Albert-Einstein-Str. 22, 18059 Rostock, Germany
e-mail: Michael.Fellmann@Uni-Rostock.de

G. Grambow et al. (eds.), *Advances in Intelligent Process-Aware Information Systems*, Intelligent Systems Reference Library 123,
DOI 10.1007/978-3-319-52181-7_7

7.1 Introduction

For planning, controlling and managing of business processes and in order to handle the complexity associated with them, semi-formal models have been established. Typically, semi-formal modelling languages are used for the construction of such models. These languages try to balance mathematical accuracy with intuitive comprehension. Examples are the Business Process Model and Notation (BPMN), the Event-driven Process Chain (EPC) or the UML Activity Diagram. A characteristic feature of these languages is that the labels of model elements, e.g. *Check order* as a label of a BPMN task or EPC function, are assigned by the modeller with the help of natural language. Therefore, an essential part of the semantics of a process model is always bound to natural language. Due to the ambiguities of the natural language and a lack of (formalized) domain- or background knowledge, the processing of the models semantics is a challenging task.

The chapter at hand addresses this task. The semantics-related challenges of semi-formal modelling will be examined in more depth in Sect. 7.2. The process representation based on description logic which is the prerequisite for semantically well-defined model elements and machine reasoning is described in Sect. 7.3. How the knowledge contained in a description logic representation can be accessed via a query language is presented in Sect. 7.4. The types of inferred facts are characterized in Sect. 7.5. In order to use the concepts introduced, tool support is presented in Sect. 7.6. Finally, a conclusion is given in Sect. 7.7. This chapter is a revised and translated version of the introductory paper of [1] where the approach is described in more detail.

7.2 Semantics-Related Challenges of Semi-formal Modelling

In the following, some semantics-related challenges in semi-formal modelling are described. The first two challenges relate to the lack of unambiguous and machine processable semantics of individual model elements. The third challenge described is the lack of tools to support the creation and analysis of models with machine processable semantics at the level of individual model elements.

7.2.1 Ambiguities of the Natural Language

Natural language inevitably entails room for interpretation, which is, in the context of semi-formal modelling, referred to as a *linguistic defect* in literature. In this context, it is possible to distinguish between *synonyms*, *homonyms*, *equipollence*, *vagueness* and *incorrect designations* [2]. Especially models which are collaboratively created are problematic since agreeing on common terms is difficult in practice [3, 4].

The mentioned linguistic defects reduce the benefit of models as a medium of communication and emerged as one of the biggest problems of semi-formal modelling in practice [5]. For example, a reduced benefit or additional costs can emerge from synonyms in the labels of model elements. As a result, multiple drafts or multiple implementations of supporting information systems can occur in the subsequent phases [6]. Conversely, more recent research shows that commonly accepted and comprehended terminology are a factor of success for the development and implementation of information systems [7]. Furthermore, they cut costs, improve collaboration and simplify the decision-making for managers [8]. In this respect, projects are more successful when the actors early agree on a common terminology in comparison to projects where this is not the case [9].

7.2.2 Lack of Machine Processable Semantics

An inconsistent language used in conjunction with linguistic defects, as they occur in the current status quo of semi-formal modelling approaches, do not only complicate the interpretation and processing of models by human beings. They also prevent the (exact) processing of knowledge represented by the models with machines. However, this processing is essential in order to enable advanced process modelling tool features such as construction recommendations for model completion. Also, it is indispensable for automated process analyses on the level of the semantic model content concerning their completeness and a consistent degree of abstraction. Particularly for inexperienced modellers, modelling with a consistent degree of abstraction is challenging [10, 11]. Further, a problem while searching for models or model fragments is that the detection of facts and relations implicitly contained in the models is impossible although this information is deducible by logical conclusions. One example of this is a business process which includes a function that accesses resources stockpiled in a warehouse. Hence, the process reduces the stock. This deduction cannot be derived, if these connections are not specified in a machine processable form.

7.2.3 Lack of Semantics-Based Tool Support

Despite the variety of tools for generation, analysis and administration of models which were developed in the past, most of these tools do not consider the semantic content of individual model elements. Current advancements—especially in the commercial sector—mostly improve collaboration and cooperation aspects, but not the semantic support offered by the tool. This represents a gap in the current state of science and practice in particular against the background of the already developed standardized semantics in the form of extensive ontologies such as the MIT Process Handbook or the PCF taxonomy (Process Classification Framework). These are so

far rarely used in tools for supporting model construction. Admittedly, in the area of Semantic Web Services there has been extensive research work in a scientific context to improve the transformation of (workflow) models to machine processable models [12–20]. But this research did barely lead to improved tools regarding the modelling of business-oriented process models.

Therefore, in the next Sect. 7.3 the fundamentals of an approach are introduced that enables to build advanced tools improving the machine support for construction and analysis of process models. This approach consists of a description logic-based process representation.

7.3 Description Logic-Based Process Representation

For representation purposes of formal ontologies, many languages in the area of artificial intelligence and especially in the area of the Semantic Web have been developed. The underlying description logics have been intensively researched for approximately 30 years. Semantic networks and frames can be thought of as precursors of description logic [21]. They intended a "natural" knowledge representation, while the efficiency of algorithms did not have priority. Contemporary description logics are designed with the aim to maintain an efficient computation despite a high expressiveness. Therefore, machine inference is also made possible within large knowledge bases (for the evolution of description logics, cf. [21, 22]).

In the past years, there has been a huge progress concerning the expressiveness and especially the scalability of knowledge bases. In this context, in particular the results of the "Billion Triple Challenge" are relevant—a contest where different developers and providers of knowledge base storages make a contest on processing data sets consisting of one billion triples (a triple is an elementary statement consisting of a subject, predicate and object—these terms a borrowed from linguistics). This mentioned progress has enabled the extension of semi-formal process modelling, which is described in this chapter.

The Web Ontology Language (OWL) is used for representation purposes within this research, because it is widespread even beyond the AI-research community. Further, it is standardized through the W3C [23] and a huge tool support is available. Specifically description logics based on the OWL-DL profile ("DL" therefore stands for "Description Logics") have been selected, because of the high expressiveness while retaining computational efficiency. There are powerful interference machines available for OWL-DL like *Pellet, FACT++* and *Hermit.*

In the following subsection, an overview of the approach is given from a conceptual view. Following this, an example is presented with an emphasis on representing the control flow of process models.

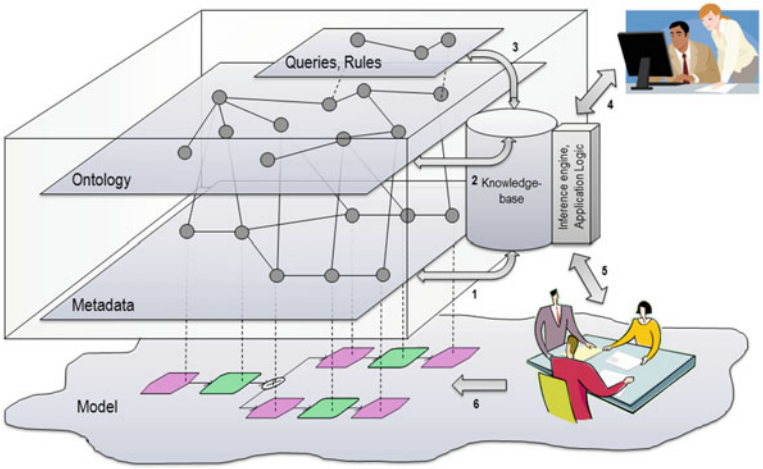

Fig. 7.1 Approach for semantic business process modeling

7.3.1 Conceptual Overview

The semantic process modelling presented in the context of this chapter is based on an ontology-based process model representation. The meta-model of the approach is described in [24]. Figure 7.1 illustrates essential elements of the approach for semantic business process modelling and their interaction.

The model-layer is connected with the layer of metadata by representing the model on the layer of metadata (dashed line between the layers). Thereby, the resulting generated metadata describing the model are stored in the knowledge base (arrow 1). They enable an interpretation of the model on the layer of ontology (arrow 2). This interpretation is possible because of the connections of elements from the metadata-layer with elements of the ontology-layer (lines between the metadata- and ontology-layer). In the context of this chapter, this connection is also named *semantic annotation*. On the layer of queries and rules, the possible inferences on the ontology-layer can be used to answer queries and check correctness conditions (arrow 3). They can not only relate to explicit represented, but also to logically deductible facts. The query and rules can both be used by analysts and model constructors (arrow 4 and 5) to retrieve information from the knowledge base and check the correctness. To do so, a user interface for example in the form of a modelling tool extension is required (cf. upper right image). The hereby possible insights can lead to a need of revising the model (arrow 6).

7.3.2 Ontology-Based Process Model Representation

In order to annotate model elements with a well-defined semantics and to query the process knowledge on a semantic level, an ontology-based process model representation is required. It will be explained briefly in this section. To begin with, in order to represent process models in an ontology, classes (also denoted as "concepts" in the context of ontologies) such as *Function*, *Event* or *Gate* have to be present in the ontology scheme. Moreover, properties (also referred to as "relations" or "connections") that connect instances (also referred to as "individuals") of those classes have to be specified such as *connects_to*. Fundamentally, the ontology schema in the form of classes and properties for representation of business process models was already presented in [25]. Thus in the remainder of this section, emphasis is put on the extension of this scheme by additional properties. In order to ease behavioural queries, the ontology schema was extended by a few OWL object properties (hereinafter simplified referred to as *properties*), which are used to represent the control flow in terms of behavioural queries. The following listing demonstrates the hierarchical structure of all properties of the extended ontology scheme by indentation.

```
graph_arc
   flow
    connects_to
      has_after_AND
      has_after_decision
        has_after_OR
        has_after_XOR
     has_after_event
     has_after_function
    flow_all
       flow_all_strict
    flow_strict
       flow_all_strict
    precedes
      precedes_all
 is_parallel_to
 is_exclusive_to
    is_exclusive_to_strict
 is_multichoice_to
    is_multichoice_to_strict
```

In the ontology language OWL, the property hierarchy is specified using the construct *rdfs:subPropertyOf* inherited from the RDF-schema. The addition of name space prefixes has been omitted in favour of a better readability. All properties are derived from the relation *graph_arc*. The transitive property *flow* specifies a directed

path between model elements. A direct connection between two model elements is specified by a property *connects_to*. The type of the element following in the control flow can be implicitly indicated with semantically more specific properties. These comprise *has_after_AND*, *has_after_OR*, *has_after_XOR*, *has_after_ event* and *has_after_function*. As the property *flow* (such as all properties in RDF or OWL) is directed, a successor relationship is specified between two elements *a* and *b*, which are connected by *flow*. Predecessor relationships are not separately represented. If these are required for the specification of graph patterns in queries, they can be specified by interchanging the elements *a* and *b* in the query. Moreover, the terms of these properties are chosen in such a way so that they closely correspond with propositional logic operators in order to enable intuitive queries. These propositional logic operators are also eponymous for control flow operators in the EPC or other languages such as BPMN.

Moreover, the property *is_parallel_to* indicates a possible parallel execution. The properties *is_exclusive_to*, *is_exclusive_to_strict* and is *is_multichoice_to* as well as *is_multichoice_to_strict* indicate an exclusiveness relation.

In general, the suffix *strict* implies that the corresponding path is located outside of loops. The suffix_*all* occurring in properties such *flow_all* and *flow_all_strict* indicates that no alternative decisions are located on the path between two nodes. Paths with the properties *precedes* and *precedes_all* are interrupted when branches caused by alternative decisions are merged. This is done because from the perspective of the corresponding join connector, it cannot be specified which elements were previously executed. The property *precedes_all* is also interrupted before loops, because elements can be activated multiple times in loops without the element before the loop to be executed.

Figure 7.2 illustrates the existence of the properties by a process example. In this example, the node $F10$ is selected. All other nodes in the model which are connected to $F10$ via chains of properties are marked with symbols representing the respective type of the properties. For example, all tokens that pass node $F10$ also pass $F9$, $E3$ and $E2$—however, it may be the case that $F9$ is executed before $F10$ or vice versa. A node that is executed strictly after $F10$ and executed for all tokens that pass $F10$ is $E2$.

Hierarchical structuring of the properties for control flow representation enables queries with different accuracy. This is also requested from Beeri et al. [26] for an adequate query language. For instance, with the property *flow* it can be discovered whether a connection between two elements exist. Further, with the property *has_after_XOR* only such model elements in the ontology-based representation can be found which directly follow a logical alternative decision. The featured properties in queries can be combined arbitrarily. In this way, some parts of queries may be more precisely while others are coarser enabling a flexible querying of process knowledge.

The presented ontology-based representation of process models can be generated using the algorithm described in [1] for so called "structured models" only. That is, the models must contain balanced splitting and merging connectors whereby each opening connector is closed with a connector of the same type so that blocks of nested

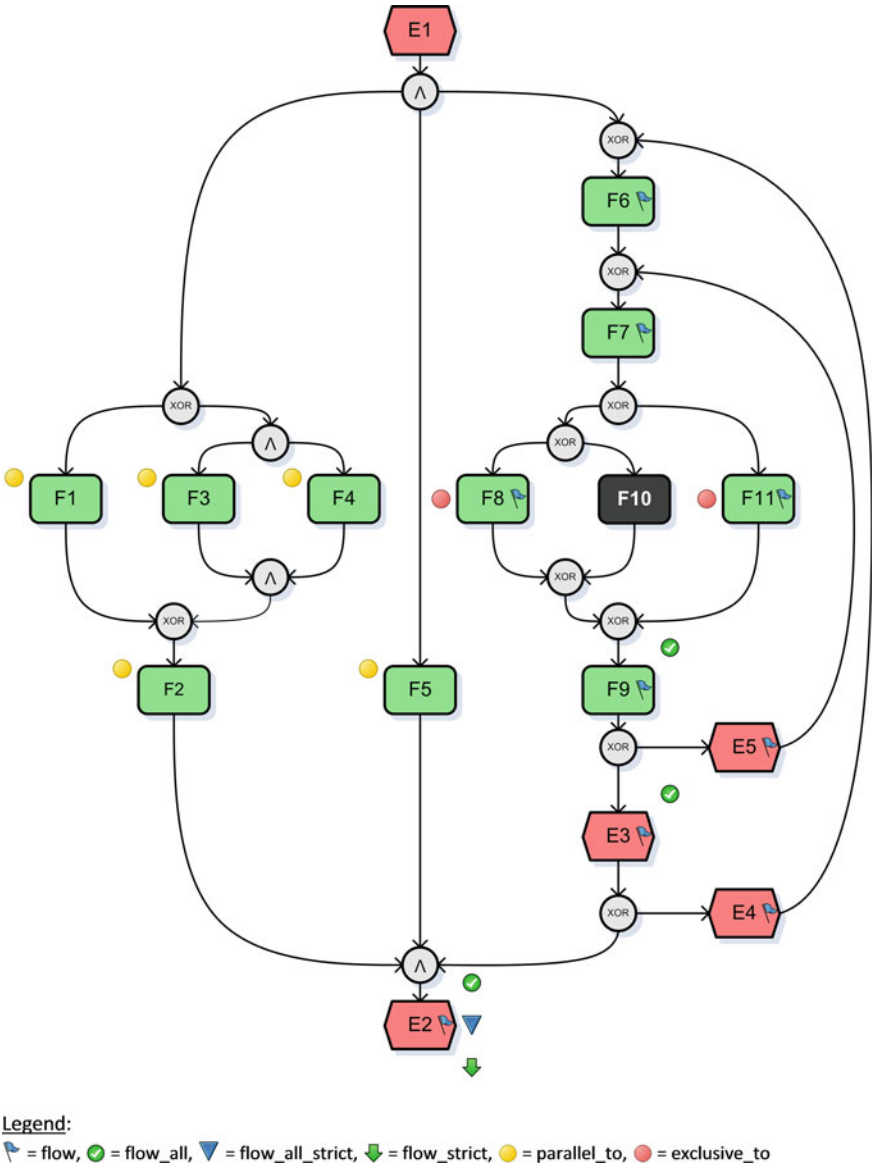

Fig. 7.2 Sample model with ontology properties for control flow representation

connectors occur. However, the models are allowed to contain do-while-loops (such as in Fig. 7.2 between $F6$ and $E3$). If the models are unstructured, a subset of the properties can be generated. To do so, the behavioural profiles described and implemented by [27] can be used. Using these profiles, three relations can be computed

that are strict order, exclusiveness and interleaving order. The first two can be used to detect the `flow_strict` and `is_exclusive_to_strict` properties.

7.4 Querying Process Knowledge

In this Section it is explained how the knowledge captured in the ontology-based process model representation (cf. Sect. 7.3) can be queried using the standard semantic web query language SPARQL. This language is a standardized query language by the W3C to query graph structures. It achieved a de facto status especially in the area of Semantic Web and the Linked Open Data (LOD)-movement [28]. This success may be caused by the simple basic structure of the queries. SPARQL is meanwhile supported by major commercial databases of the business environment. These also offer a scalable storage of large quantities of OWL ontologies or RDF files as well as manifold inference capabilities. The relevance of this aspect is emphasized by real world enterprises, which use hundreds of models [29], which achieve a substantial size (the so-called "process wallpaper") [30].

A SPARQL query refers to RDF triples, which represent a directed graph. Basically, the query consists of a *pattern matching*, a *modifier of solutions* and a *return* part [28]. The function of *pattern matching* is to compare a graph pattern, which is composed out of triple patterns, against an RDF-graph. In the context of the ontology-based process representation, graph patterns can be intuitively understood as structure or "a way through the graph". The triple patterns, which determine the graph pattern, represent the navigation step. At the position of the subject and object of a pattern, classes of ontology or the instances of the ontology can be specified. At the position of the predicate, the ontology properties can be specified. Moreover, at the position of the object, simple data values such as literals (e.g. character strings) can be specified. A simple query to return all instances in the ontology, which are connected through a property `predicate`, can be formulated as follows.

```
SELECT ?var1 ?var2 WHERE { ?var1 predicate ?var2 }
```

Within this chapter, this general scheme will be applied to query process models. Due to the chosen representation of the process models, in which the elements of the model are represented as instances and the control flow as properties in the ontology, an intuitive application of SPARQL is possible for searching in process graphs. The represented elements of the model in the ontology occur in these queries at the position of the subject or the object in a triple pattern. By combining several patterns, complex queries can be created. Queries may also contain placeholders, so called blank- or *anonymous nodes*. A simple example for this would be to return all activities being followed by a XOR decision.

```
SELECT ?decision WHERE { ?decision :has_after_XOR [] }
```

The graph model of this query defines one single triple pattern; its subject is the variable `?decision`, which is also used for the return, its predicate has the property `:has_after_XOR` and its object is an anonymous node [] specified as a dummy for arbitrary nodes. Due to the integration of behavioral properties like

:flow_all_strict in the ontological representations, behavioral queries are possible as well. For example all activities, which in any case will be executed once after the process is started, can be retrieved by the following query.

```
SELECT ?mandatory WHERE  {
   ?x a :StartEvent ; :flow_all_strict ?mandatory
}
```

For queries with complex paths between two nodes, *property path expressions* enable a significant gain in expressiveness and likewise a simplified notation. For example to return such model element pairs, which are on a path connected with at least one subsequent XOR- or OR-split and after this with at least two sequenced AND-splits, the following query can be used.

```
SELECT ?node1 ?node2 WHERE {
   ?node1 ( :has_after_XOR | :has_after_OR )+
              / :has_after_AND{2,}/:connects_to ?node2
}
```

Path expressions allow the declaration of cardinalities, in short form for arbitrary *, for at least one + and optional ? or in detail by a min-max-notation {min,max}. The concatenation of several properties for the navigation through the graph can be written with a slash / and alternatives are represented in parentheses (opt1 | opt |...| optN). However, with SPARQL it is not possible to return the specific pattern, including the nodes, when not only the nodes at the end of a path are relevant. This can easily be upgraded with extensions like GLEEN [31]. Other extensions of SPARQL allow more comprehensive restrictions for searching on a graph. For example, these restrictions can refer to the presence or absence of specific nodes on a path. With RPL (RDF Path Language) [32] an approach exists for the navigation in RDF-graphs, similar to XPath expressions in the context of XML-standards. SPARQ2L supports the subgraph extractions from RDF-data and with CSPAROL (Constrained SPARQL) [33] an extension for describing path-restrictions exists, which can be explained in cooperation with the extension PSPARQL (Pattern SPARQL) [34]. Sometimes also new query languages like RDFPath are developed, which enable an expressive path-declaration on large RDF graphs [35]. More research that could be applied to extend the approach presented here exists in the area of graph databases [36].

To reflect the semantic annotation in the query, the graph model has to be extended with corresponding triple patterns, specifying the annotation. For example to return all model elements that are annotated with an ontology instance :notifiy_customer via the annotation property :equivalent_to, the following query can be used.

```
SELECT ?notify WHERE {
    ?notify :equivalent_to :notify_customer
}
```

A more abstract form of a query occurs when the ontology-instance used to annotate a model element is unknown. Therefore an ontology-class will be specified instead of an ontology-instance. In addition, the annotation can be formulized more unspecific, by using the super-property :has_annotation instead of the sub-properties :equivalent_to and :narrower_than. In this way, annotations that are semantically equivalent as well as more general will be found. For example to return all model elements that are annotated with an ontology-instance of the class :StrategicActivity via the property :has_annotation, the following query can be used.

```
SELECT ?strategic WHERE {
   ?strategic :has_annotation [ a :StrategicActivity ]
}
```

By querying the ontology-instance type with a, which is a short form for rdf:type, the entire spectrum of conclusions for classifying instances, that modern description logics like OWL 2 for the classifications of instances provide, can be used. This machine inferred facts will be included in the result of the query together with other inferences for example based on transitive properties.

In the next section, the deductions that are added to the ontology-based process model representation by standard reasoners are described. The deductions result in additional facts being inferred by machine reasoning procedures. These facts enrich the results that can be retrieved by queries such as described in this chapter. In the next chapter, the facts are characterized in further detail.

7.5 Use of Machine Reasoning

In this section, at first an overview on the subject of automatic reasoning is provided. After this, a characterization of important types of inferences is provided. Inferences are distinguished first according to whether they relate to type information or relation information. Second, according to whether they involve the ontology instances representing the process model or the domain. These distinctions lead to a matrix of four inference types (cf. Fig. 7.3 bottom). Examples to explain these inference types are subsequently illustrated by (a) pointing to a coherent annotated sample process graphically illustrated in Fig. 7.3, (b) by describing the inference type and its practical value for querying using natural language and (c) by giving an example in description logics syntax along with explanations.

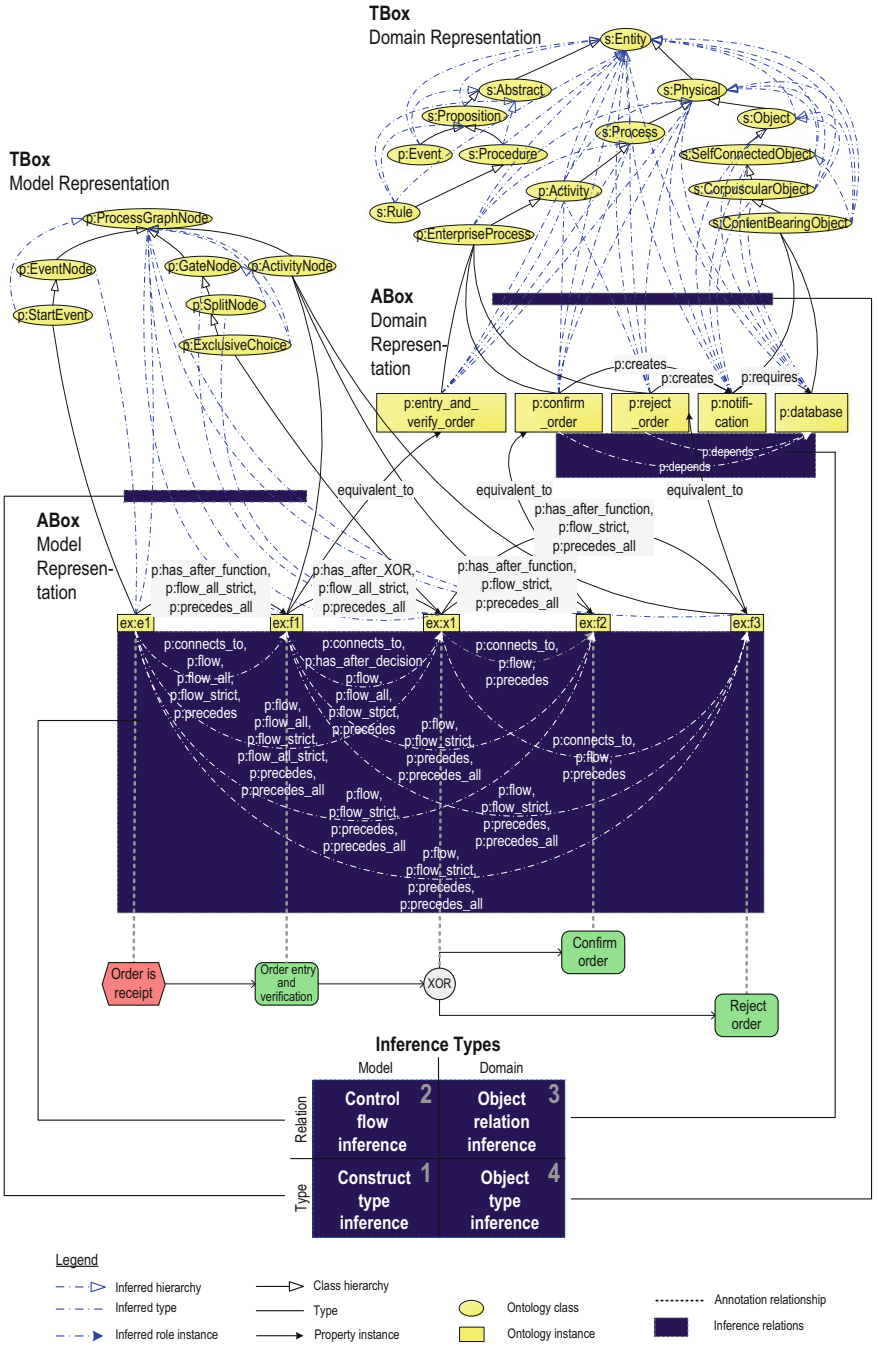

Fig. 7.3 Inference types of the ontology-based process representation

7.5.1 Overview

In this section, types of machine inferences will be characterized. The types are intended and described in terms of their relevance for the business-level interpretation of process models, not as general categories of inferences possible with description logics. Figure 7.3 illustrates the types that are connected to the respective inferences depicted by a line with a dot-and-dash-pattern integrated in the sample ontology-based process representation. In more detail, the classification of an inference to an inference type follows out of the crossing of such lines through a dark filled rectangle, which is connected to a corresponding inference type. The namespace `p:` used in Fig. 7.3 stands for the extensions of the SUMO-ontology [37], `s:` for the SUMO-ontology itself and `ex:` (example) for any example-namespace. The inferences presented in Fig. 7.3 represent additions both to the TBox (Terminological Box, also called *ontology scheme*) and the ABox (Assertional Box, also called *ontology data*) of the ontology. In this contribution, the term *ontology* includes both TBox and ABox. This use is common in the context of OWL ontologies.

7.5.2 Characterization of the Four Types of Inferences

According to their content, the conclusions can be classified into four inference types being *construct type inference*, *control flow inference*, *object relation inference* and *object type inference*. They will to be characterized in the following.

7.5.2.1 Construct Type Inference

Concerned here are conclusions that relate to the *type* of ontology instances which represent the model elements. In queries, the *construct type inference* enables an abstraction of the ontology classes which are used to represent the constructs of a modelling language. For example, the fact that the ontology instance `ex:x1` (the XOR-connector) is inferred to be of type `p:GateNode` is a construct type inference since the type of the model element is inferred. In Fig. 7.3, this inference is depicted by the dotted line between the ontology instance belonging to the ABox model representation and the corresponding ontology class belonging to the TBox model representation. In addition to the graphical illustration, this inference is explained in the following using the DL-syntax common in description logics. The descriptions are divided into two parts: An inferred fact and its explanation.

Inferred fact:
$p{:}\,GateNode(ex{:}\,x1)$

Explanation:
$p{:}\,ExclusiveChoice(ex{:}\,x1)$
$p{:}\,ExclusiveChoice \sqsubseteq p:SplitNode$
$p{:}\,SplitNode \sqsubseteq p{:}\,GateNode$

The inferred fact expresses that the individual `ex:x1` is a member of the class `p:GateNode`. This can be explained by the membership of `ex:x1` in the class `p:ExclusiveChoice` being a subclass of `p:SplitNode` which in turn is a subclass of `p:GateNode`.

An example for the practical use of the construct type inference in queries is the search for events, without the need to specify whether it is a start, intermediate or end event. Without this type of inference, the desired type of event has to be specified exactly or all sorts have to be enumerated in the query. Hence using construct type inference, a variable degree of abstraction from the constructs of the modelling language originally used for constructing the model can be achieved.

7.5.2.2 Control Flow Inference

These are conclusions relating to the properties in the ontology representing the control flow of the process model. In queries, the *control flow inference* provides an abstraction of control flow structures of a process model. For example, the fact that `ex:e1` (Order is receipt) is inferred to be connected via a `p:flow`-property with `ex:f2` (Confirm order) is a control flow inference since two elements that are not directly connected are inferred to be connected via properties. In Fig. 7.3, this inference is depicted by the arrow with a dotted line and a label "p:flow" between the two ontology instances in the ABox model representation. In DL-Syntax, the example is described as shown below.

Infered fact:
$p{:}\,flow(ex{:}\,e1, ex{:}\,f2)$

Explanation:
$p{:}\,flow\_strict(ex{:}\,e1, ex{:}\,f1)$
$p{:}\,flow\_strict(ex{:}\,f1, ex{:}\,x1)$
$p{:}\,flow\_strict(ex{:}\,x1, ex{:}\,f2)$
$p{:}\,flow\_strict^{+} \equiv p{:}\,flow\_strict$
$p{:}\,flow\_strict(ex{:}\,e1, ex{:}\,f2)$
$p{:}\,flow\_strict \sqsubseteq p{:}\,flow$

The inferred fact expresses that the individual ex:e1 has a property p:flow with a value ex:f2, in other words, that ex:e1 and ex:f2 are connected by the property p:flow. This can be explained by the three connections ex:e1 with ex:f1, ex:f1 with ex:x1 and ex:x1 with ex:f2 via the property p:flow_strict. Since the property is transitive (which is indicated using the symbol "+" in the DL-syntax), it follows that ex:e1 is also connected to ex:f2. Since the property p:flow_strict is a sub-property of p:flow, it follows that ex:e1 is connected to ex:f2 via p:flow.

The practical use of the control flow inference lies in the abstraction of concrete structures in queries, because the inferred properties in the ABox allow to query "direct connections" between the represented model elements in the ontology. Moreover, using the property hierarchy in the TBox of the ontology, a variable degree of abstraction of the control flow can be achieved.

7.5.2.3 Object Relation Inference

Conclusions of this type refer to the relations between ontology instances used for annotating model elements and the domain representing instances in the ontology. The *object relation inference* allows to request more context and information about annotated model elements that are not explicitly covered in the ontology, but can be inferred by the inference engine. In particular by transitive properties or property chains diverse conclusions may result which lead to an advanced semantic interpretation of an annotated model element and thus to new insights. For example, the fact that the instance p:confirm_order is dependent on the p:database is an object relation inference since a relation between an annotated model element and another object from the domain is inferred. The inferred fact is also depicted in Fig. 7.3 by the arrow with a dotted line and a label "p:depends" between the two instances in the ABox domain representation. In DL-Syntax, the example is described in more detail containing the relevant background knowledge as shown below.

Inferred fact:
$p{:}depends(p{:}confirm\_order, p{:}database)$

Explanation:
$p{:}creates(p{:}confirm\_order, p{:}notification)$
$p{:}requires(p{:}notification, p{:}database)$
$p{:}createsorequires \rightarrow depends$

The inferred fact expresses that p:confirm_order has a property p:depends with a value p:database, in other words, that p:confirm_order is connected to p:database by the property p:depends. This can be explained by two properties and a property chain. The first property p:creates connects

p:confirm_order with p:notification, the second property p:requires connects p:notification with p:database. Since there is a property chain specified (last line in the explanation-part of the DL-syntax above) that says that the composition of the properties p:creates and p:requires constitutes a p:depends-relation, it can be inferred that p:confirm_order and p:database are connected by the property p:depends.

The practical use of object relation inferences in queries lies for example in discovering dependencies that are imposed on the execution of activities or the detection of objectives (transitively) supported by an activity.

7.5.2.4 Object Type Inference

Conclusions of this type relate to the type of instances in the ontology which represent domain knowledge and that are used to annotate model elements. In queries, the *object type inference* enables the abstraction of these concrete ontology instances used for annotation. For example, the fact that p:entry_and_verify_order is of type s:Process is an object type inference since the type of the instance p:entry_and_verify_order which is used to annotate the activity ex:f1 is inferred. In Fig. 7.3, this inferred fact is also depicted by the dotted line between the ontology instance in the ABox domain representation and the respective ontology class in the TBox domain representation. In DL-Syntax, the example is described as shown below.

Inferred fact:

$s{:}\,Process(p{:}\,entry\_and\_verify\_order)$

Explanation:

$p{:}\,EnterpriseProcess(p{:}\,entry\_and\_verify\_order)$

$p{:}\,EnterpriseProcess \sqsubseteq p{:}\,Activity$

$p{:}\,Activity \sqsubseteq s{:}\,Process$

The inferred fact expresses that the individual p:entry_and_verify_order is a member of the class s:Process. This can be explained by the membership of this individual in the class p:EnerpriseProcess being a subclass of the more general class of activities p:Activity which in turn is a subclass of s:Process.

A practical example for the application of this type of conclusion in queries would be the search for all nodes in the process graph that are annotated with an ontology instance that creates a document type which is an instance of the class ContentbearingObject. As a result of the query, the exact document type such as fax, e-mail, or letter, which can be defined as subclasses, are abstracted by means of the object type inference.

Finally, the inferences of the object relation inference and the object type inference are heavily dependent on the descriptions and richness of the ontology. The scope and content of the ontology should be determined according to the questions to be answered, which are also called *competence questions* in the area of knowledge-based systems [38]. Examples of competence questions would be "What organizational unit is responsible for an activity?", "Which employee performs which task?", "What best practices apply for a job?", "Which cost causes the execution of an activity?", "What guidelines need to be followed?", "Which national category corresponds to a process?", just to mention a few examples. The questions to be answered influence the design of the ontology.

7.6 Tool Support

In the following, we describe the tool that has been developed for querying the ontology-bases process model representation. The following description and screenshots are based on [39] where the tool is described in more detail in [1]. The tool is called *SemQuu*—Semantic Query Utility, since much of the functionality is also relevant when exploring ontologies in general. SemQuu is implemented using the Jena library (jena.sourceforge.net), the Pellet inference engine (pellet.owldl.com), the Tomcat web server and JSP, XSLT, JavaScript, CSS and HTML for the user interface. Process models can be imported from arbitrary tools in arbitrary formats, since they can be transformed on the server into the ontology-based representation which is accomplished by a plugin-converter for each format. As an example, we have implemented an extension of Visio which can export EPC process models being annotated with ontology concepts (see Fig. 7.4 for an illustration of the following procedure). After having been exported from Visio to SemQuu, the model is transformed to an OWL-DL ontology and added to the repository (cf. Fig. 7.4 bottom right).

An overview of SemQuu is provided by Fig. 7.5 (B). In order to query the repository of ontology-based process representations with SPARQL, the user can use a simple form-based query builder (A) for successively constructing a graph pattern. This is done by inserting multiple triple patterns with the help of drop-down list boxes (A) which are aware of `rdfs:domain` and `rdfs:range` information so that no semantically wrong query can be constructed. Moreover, drop-down list boxes are dynamically updated upon selection of a value in one of the boxes. Alternatively to the drop-down list boxes, the user can leverage the full expressivity of SPARQL by using the text area input field. Moreover, when the user modifies the query she or he is supported by an "intelligent" auto-completion feature (C) which is fully aware of the ontology schema and instance data and only proposes meaningful suggestions. When queries are executed in batch mode, the result of the queries can be displayed as an information, warning or error with respective graphical symbols appearing for each type (D). The result for each query is initially collapsed but can be unfolded if the user clicks on the "+" sign symbols. In order to measure the effectiveness of

Fig. 7.4 SemQuu converter and repository

SPARQL queries for the task of semantic correctness checking, we conducted an experiment with 21 participants described in [40].

7.7 Discussion and Outlook

Using a description logics-based ontological process representation enables a well-defined semantics for model elements that at the same time is also machine processable. The querying against ontology-based process representations enables a comprehensive analysis of process models. Structure-related as well as behavioural queries can be answered through the control flow representation. Relationships between processes, as they are possible in BPMN, EPC or other languages e.g. via process interfaces or sub-processes are currently not taken into account yet, but in the future, they can be regarded by introducing additional properties such as `starts_subprocess` and `returns_to_process` in the ontology-based process modeling.

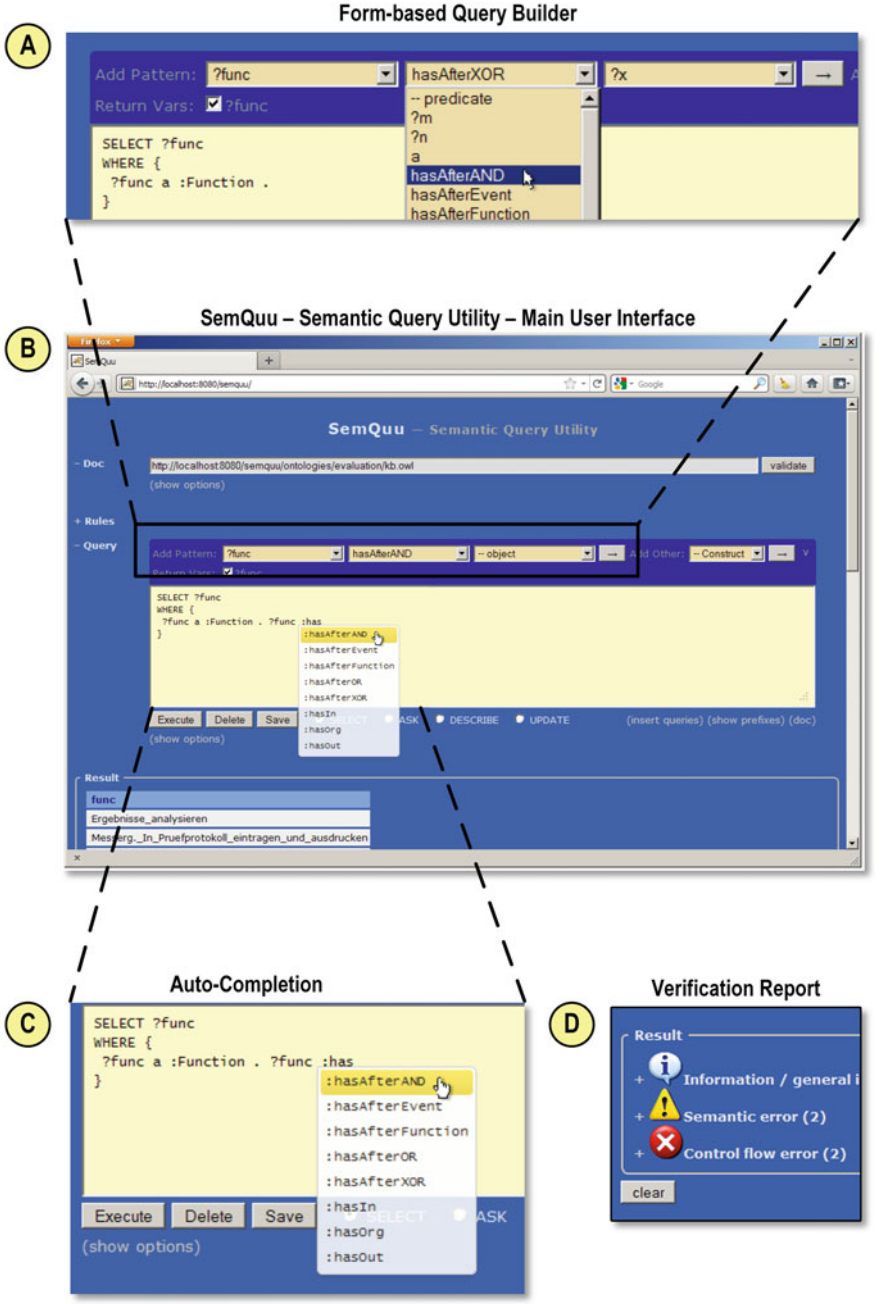

Fig. 7.5 Overview of SemQuu

In contrast to the existing approaches, the approach presented here allows the integration of the full spectrum of possible deductions with descriptions logics like OWL-DL in the results of a query, which enables the extensive use of "mechanically" created conclusions. This spectrum has been described by four classes of inference types. While the *construct type inference* and the *control flow inference* refer to inferences based on the structure of the represented process model, the *object relation* inference and the *object type inference* refer to the domain knowledge formalized in the ontology. With the presented approach of ontology-based process representation, it is possible to seamlessly combine these two knowledge areas and to reason about the process representation using description logics inference engines. Thus, a richer analysis and interpretation of the organizational process knowledge can be achieved. Future research in this field can examine a refinement of inference types or a quantification and measurement of its occurrence and usefulness in large process model repositories. Another direction of research is the automated annotation of process models (or at least, creating suggestions for annotation in an automated way). Finally, evolving the ontology collaboratively and at the same time keeping the annotation intact is subject to future research.

References

1. Fellmann, M.: Semantic Process Engineering – Konzeption und Realisierung eines Werkzeugs zur semantischen Prozessmodellierung, PhD thesis. Osnabrück University, Osnabrück (2013)
2. Ortner, E.: Methodenneutraler Fachentwurf: Zu den Grundlagen einer anwendungsorientierten Informatik. Teubner-Reihe Wirtschaftsinformatik. Teubner, Stuttgart (1997)
3. Scheer, A.W., Klueckmann, J.: BPM 3.0. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) Proceedings of the Business Process Management 7th International Conference (BPM 2009), 8–10 Sept, Ulm, Germany. LNCS, vol. 5701, pp. 15–27. Springer, Berlin (2009)
4. Hadar, I., Soffer, P.: Variations in conceptual modeling: classification and ontological analysis. J. Assoc. Inf. Syst. **7**(8), 568–592 (2006)
5. Sarshar, K., Weber, M., Loos, P.: Einsatz der Informationsmodellierung bei der Einführung betrieblicher Standardsoftware: Eine empirische Untersuchung bei Energieversorgerunternehmen. Wirtschaftsinformatik **48**(2), 120–127 (2006)
6. Rosemann, M.: Komplexitätsmanagement in Prozessmodellen: Methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung. Schriften zur EDV-orientierten Betriebswirtschaft, Gabler, Wiesbaden (1996)
7. Corvera, M., Rosenkranz, C.: Natural language alignment as a process: applying functional pragmatics in information systems development. In: Proceedings of the 18th European Conference on Information Systems (ECIS 2010), 6–9 June, Pretoria, South Africa. Paper 5 (2010)
8. Schafermeyer, M., Grgecic, D., Rosenkranz, C.: Factors influencing business process standardization: a multiple case study. In: Proceedings of 43rd Hawaii International Conference on System Sciences (HICSS 2010), 5–8 Jan, Poipu, Kauai, Hawaii, pp. 1–10. IEEE (2010)
9. Rosenkranz, C., Räkers, M., Behrmann, W., Holten, R.: Supporting financial data warehouse development: a communication theory-based approach. In: Proceedings of the Thirty First International Conference on Information Systems (ICIS 2010), 12–15 Dec, Saint Louis, Missouri, USA. Paper 12 (2011)
10. Nielen, A., Költer, D., Mütze-Niewöhner, S., Karla, J., Schlick, C.: An empirical analysis of human performance and error in process model development. In: Proceedings of the 30th

International Conference on Conceptual Modeling (ER 2011), Brussels, Belgium, pp. 514–523 (2011)
11. Wilmont, I., Brinkkemper, S., Weerd, I., Hoppenbrouwers, S.: Exploring intuitive modelling behaviour. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling: Proceedings of the 11th International Workshop, BPMDS 2010 and 15th International Conference, EMMSAD 2010 held at CAiSE 2010, 7–8 June, Hammamet, Tunisia. LNBIP vol. 50, pp. 301–313. Springer, Berlin (2010)
12. Cabral, L., Domingue, J., Motta, E., Payne, T.R., Hakimpour, F.: Approaches to semantic web services: an overview and comparisons. In: Bussler, C., Davies, J., Fensel, D., Studer, R. (eds.) The Semantic Web: Research and Applications: Proceedings of the First European Semantic Web Symposium, ESWS 2004 Heraklion, Crete, Greece, 10–12 May. LNCS, vol. 3053, pp. 225–239. Springer, Berlin (2004)
13. Cardoso, J., Sheth, A.P.: Introduction to semantic web services and web process composition. In: Cardoso, J., Sheth, A.P. (eds.) Semantic Web Services and Web Process Composition: First International Workshop, SWSWPC 2004, San Diego, CA, USA, 6 July, pp. 1–13 (2005)
14. 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. Appl. Ontol. **1**(1), 77–106 (2005)
15. The OWL Services Coalition (eds.) OWL-S: Semantic Markup for Web Services
16. Farell, J., Lausen, H. (eds.): Semantic Annotations for WSDL: W3C Recommendation, 28 Aug 2007. W3C (2007)
17. Wetzstein, B., Ma, Z., Filipowska, A., Kaczmarek, M., Bhiri, S., Losada, S., Lopez-Cob, J.-M., Cicurel, L.: Semantic business process management: a lifecycle based requirements analysis. In: Hepp, M., Hinkelmann, K., Karagiannis, D., Klein, R., Stojanovic, N. (eds.) Proceedings of Workshop on Semantic Business Process and Product Lifecycle (SBPM 2007) in conjunction with the 4th European Semantic Web Conference (ESWC 2007), 3–7 June, Innsbruck, Austria. CEUR Workshop Proceedings, vol. 251, pp. 1–11. RWTH, Aachen (2007)
18. Weber, I.M.: Verification of annotated process models. In: Weber, M. (ed.) Semantic Methods for Execution-level Business Process Modeling, Part 2. LNBIP, vol. 40, pp. 97–148. Springer, Berlin (2009)
19. Weber, I., Hoffmann, J., Mendling, J.: Beyond soundness: on the verification of semantic business process models. Distrib. Parallel Databases **2010**(27), 271–343 (2010)
20. Drumm, C., Filipowska, A., Hoffmann, J., Kaczmarek, M., Kaczmarek, T., Kowalkiewicz, M., Markovic, I., Scicluna, J., Vanhatalo, J., Völzer, H., Weber, I., Wieloch, K., Zyskowski, D.: Dynamic Composition Reasoning Framework and Prototype. Project IST 026850 SUPER, Deliverable 3.2. SAP (2007)
21. Baader, F.: What's new in description logics. Informatik-Spektrum **34**(5), 1–9 (2011)
22. McGuinness, D.L.: Description logics emerge from ivory towers. In: Goble, C.A., McGuiness, D.L., Möller, R., Patel-Schneider, P.F. (eds.) Proceedings of the International Workshop on Description Logics, 1–3 Aug, Stanford University, California, USA, pp. 64–68 (2001)
23. OWL Working Group: OWL 2: W3C Recommendation, 11 Dec 2012 (2012)
24. Thomas, O., Fellmann, M.: Semantische Prozessmodellierung – Konzeption und informationstechnische Unterstützung einer ontologiebasierten Repräsentation von Geschäftsprozessen. Wirtschaftsinformatik **51**(6), 506–518 (2009)
25. Thomas, O., Fellmann, M.: Semantic process modeling—design and implementation of an ontology-based representation of business processes. Bus. Inf. Syst. Eng. **1**(6), 438–451 (2009)
26. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes. In: Dayal, U., Whang, K.Y., Lomet, D.B., Alonso, G., Lohman, G.M., Kersten, M.L., Cha, S.K., Kim, Y.K. (eds.) Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006), 12–15 Sept, Seoul, Korea, pp. 354–366. VLDB Endowment, ACM (2006)
27. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Causal behavioural profiles—efficient computation, applications, and evaluation. Fundamenta Informaticae (FI) **113**(3–4), 399–435 (2011)

28. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. (TODS) **34**(3), 16 (2009)
29. Awad, A.: BPMN-Q: A language to query business processes. In: Reichert, M., Strecker, S., Turowski, K. (eds.) Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA2007), 8–9 Oct, St. Goar, Germany, pp. 115–128 (2007)
30. Polyvyanyy, A., Smirnov, S., Weske, M.: Business process model abstraction. In: vom Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 1: Introduction, Methods, and Information Systems: Part II Methods. International Handbooks on Information Systems, pp. 149–166. Springer, Berlin (2010)
31. Detwiler, L.T., Suciu, D., Brinkley, J.F.: Regular paths in SparQL: querying the NCI thesaurus. In: Proceedings of the American Medical Informatics Association Symposium on Biomedical and Health Informatics (AIMA 2008), 8–12 Nov, Washington, DC, pp. 161–165 (2008)
32. Zauner, H., Linse, B., Furche, T., Bry, F.: A RPL through RDF: expressive navigation in RDF graphs. In: Hitzler, P., Lukasiewicz, T. (eds.) Proceedings of the 4th International Conference on Web Reasoning and Rule Systems (RR 2010), 22–24 Sept, Bressanone/Brixen, Italy. LNCS, vol. 6333, pp. 251–257. Springer, Berlin (2010)
33. Alkhateeb, F., Baget, J.F., Euzenat, J.: Constrained regular expressions in SPARQL: Proceedings of the International Conference on Semantic Web and Web Services (SWWS), 14–17 July, Las Vegas, NV US, pp. 91–99 (2008)
34. Alkhateeb, F., Baget, J.F., Euzenat, J.: Extending SPARQL with regular expression patterns (for querying RDF). Web Semant. Sci. Serv. Agents World Wide Web **7**(2), 57–73 (2009)
35. Przyjaciel-Zablocki, M., Schätzle, A., Hornung, T., Lausen, G.: RDFPath: path query processing on large RDF graphs with MapReduce. In: Proceedings of the 1st Workshop on High-Performance Computing for the Semantic Web (HPCSW2011) co-located with the 8th Extended Semantic Web Conference, ESWC2011, 29 May, Heraklion, Greece (2011)
36. Angles, R., Gutierrez, C.: Querying RDF data from a graph database perspective. In: Gómez-Pérez, A., Euzenat, J. (eds.) Proceedings of the Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29–June 1. LNCS, vol. 3532, pp. 346–360. Springer, Berlin (2005)
37. Niles, I., Pease, A.: Towards a standard upper ontology. In: Welty, C., Smith, B. (eds.) Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), 17–19 Oct, Ogunquit, Maine, pp. 2–9 (2001)
38. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web. Springer, London (2004)
39. Fellmann, M., Thomas, O.: Process model verification with SemQuu. In: Nüttgens, M., Thomas, O., Weber, B. (eds.) Enterprise Modelling and Information Systems Architectures (EMISA 2011), 22–23 Sept, Hamburg, Germany. GI LNI P-190, pp. 231–236. Köllen, Bonn (2011)
40. Fellmann, M., Thomas, O., Busch, B.: A query-driven approach for checking the semantic correctness of ontology-based process representations. In: Abramowicz, W. (ed.) Proceedings of the 14th International Conference on Business Information Systems (BIS 2011), 15–17 June, Poznan, Poland. LNBIP, vol. 87, pp. 62–73. Springer, Berlin (2011)