

Experiences in Mapping the Business Intelligence Model to Description Logics, and the Case for Parametric Concepts

Alexander Borgida¹, Jennifer Horkoff², John Mylopoulos² and Riccardo Rosati³

¹ Dept. of Computer Science, Rutgers University, NJ, USA

² Dept. of Computer Science, Univ. of Toronto, Canada

³ DIS, Sapienza Università di Roma, Italy

1 Summary

BIM is a new language for capturing business models containing information relevant for strategic analysis of business operations. It has been used in several large case studies and is being pursued in industry.

The paper introduces the key notions of BIM, including goals, evidence, and influence. It also outlines their translation into DL axioms, forming an upper-level ontology. Specific BIM domain models then result from the addition of axioms to this. The result provides both a formal semantics of the BIM language, and all the familiar advantages of decidable DL reasoning, including consistency checking, defined-concept classification, and, in our case “What if” scenario analysis. We focus on the parts of the translation which are most interesting, including: i) modeling “evidence and pursuit propagation” about goals, ii) dealing with “meta-properties”, which were introduced as a result of an ontological analysis of previous BI languages, and iii) the repeated need for too many similar axioms.

For the last two, we sketch how **parametrized concepts**, together with rules, would significantly help knowledge-base maintenance. This opens up a new research area in hybrid DL+rule KBs, involving rules that generate new axioms.

2 Introduction to BIM

Business intelligence (BI) offers considerable potential for gaining insights into day-to-day business operations, as well as longer term opportunities and threats. Most businesses have a significant investment in BI; however, much of the information is data oriented – mostly low-level values difficult to understand in terms of business strategy. Instead, there is a need for analysis using terms like strategic objectives, business models, processes, markets, trends and risks.

Several BI modeling techniques exist already: the Business Motivation Model (BMM) [1], Strategy Maps (SM) [2], Balanced Scorecards (BSc) [3], Goal Modeling frameworks (GM) [4, 5]. These languages introduce many concepts informally, making it difficult to distinguish between them; (except for GM), do not support reasoning over models; and do not offer facilities for standard conceptual modeling of domain entities. The Business Intelligence Model (BIM) was introduced [8, 7] to rationalize and extend notions in previous languages.

A portion of a realistic BIM schema for the credit card industry is shown in in Fig. 1, using a (provisional) graphical notation based on the i* GM [4].

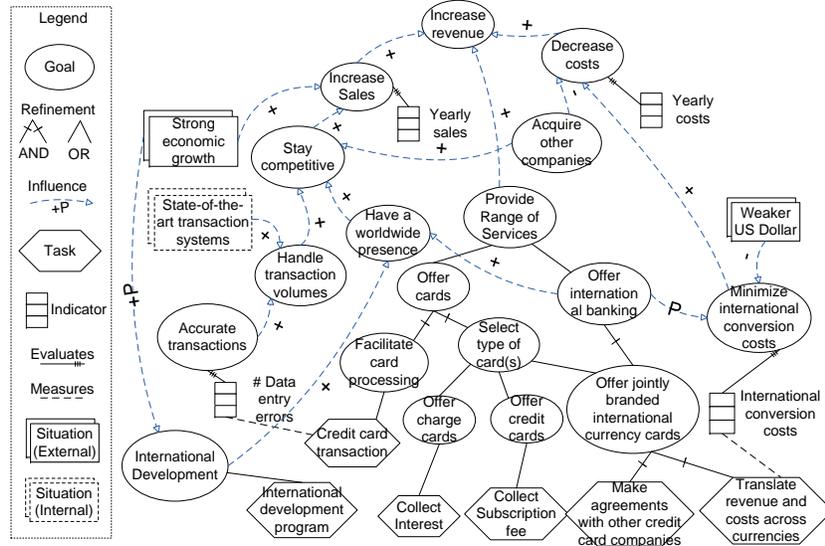


Figure 1: Fragment of BIM schema for banking industry

Generally, the model shows the decomposition of basic business services (e.g., offer cards, offer international banking) into operational tasks, their effects on strategic goals, an assessment of influencing situations, and measurement through indicators. So BIM focuses on four types of things: **Situations**, **Indicators**, **Tasks** and **Entities**, which are subclasses of **BIM_Thing**. (We abbreviate this to Thing, using OWL:Thing, if we need, to talk about the top DL concept.) Situations are partial descriptions of world state, which may affect business objectives, and in BIM are specialized into **Goals**, **Operational Situations**, and **Domain Assumptions**.

Goals are situations that may be desired by the modeling organization, such as “Increase revenue”. Goals may or may not be actively *pursued* at any time, and have an *evidence* value. As usual in GM, goals are refined until one finds actions that help achieve them (Tasks), or domain assertions that are assumed to hold in the real world.

Additional concepts found in other BI languages can be obtained in BIM by providing values to meta-attributes (see Section 3.4). These provide an optional richer subclass structure for BIM without over-complicating the initial model design, and language learning.

BIM includes five different types of relationships between things: **influences**, **evaluates**, **refines**, and **measures**, but their domains/ranges are restricted to various subclasses of Thing.

We are only going to touch on those aspects of BIM that raise interesting issues for DL modeling. Unfortunately, this excludes one of the most interesting features of BIM: the notion of *indicators*, which measure the performance of some business activity. These use heavily numeric functions [9], and we were unable to model them in depth in OWL2.

3 The Translation of BIM to DL

3.1 Some BIM Classes and their Axioms

The DL axioms capturing the semantics⁴ of the subclass hierarchy under Thing are standard, as is the specification of disjointness between sibling classes.

BIM allows accumulation of *evidence* for or against every thing in BIM. The question “Evidence for . . . ?” is answered depending on the specific type of thing. So BIM tracks evidence for the *occurrence* of situations, the *satisfaction* of goals and domain assumptions, the *performance* of indicators, the *execution* of tasks, and the *existence* of entities. In this way, we use BIM to monitor the state of relevant business concepts. Following [6], we will accumulate various *qualitative* kinds of evidence (for and against) from *multiple sources*, and combine them using a multi-valued logic approach, so that the value of *evidence* can be zero or more of StrongEvidenceFor (SF), WeakEvidenceFor (WF), StrongEvidenceAgainst (SA), and WeakEvidenceAgainst (WA). We therefore have

$$\text{evidence} \sqsubseteq_r \text{Thing} \times \{\text{SF}, \text{WF}, \text{WA}, \text{SA}\}$$

Rather than constantly asking whether the evidence for something is strong by checking subsumption by ($\text{evidence} : \text{SF}$) (a synonym for $\exists \text{evidence}.\{\text{SF}\}$), we will find it convenient to define four concepts like

$$\text{SFThing} \equiv \text{Thing} \sqcap (\text{evidence} : \text{SF})$$

Such repetitions of axioms are frequent and annoying for BIM so we introduce schemas for them, using the notation of programming language features such as C++ templates and Java generics:

$$\text{Thing}\langle ?V \rangle \equiv \text{Thing} \sqcap (\text{evidence} : ?V) \text{ for } ?V \in \{\text{SF}, \text{WF}, \text{WA}, \text{SA}\}$$

Note that in C++ and Java, parameters may be restricted to be subtypes of other types; for example, a class `SortedList<?V>` may require `?V` to be a subtype of `Comparable` to guarantee a `lessThan` operation. We will present parameterized DL concepts using rules, whose body limits the parameter values using *P-facts* (think of these as “parameter facts”), about individuals or “punned” class identifiers. Thus after introducing P-predicate *EvidValues*, with instances SF, WF, WA and SA, we can write:

⁴ We have intentionally not chosen a particularly restrictive DL at this point, since a lower bound on complexity of BIM reasoning can only be obtained directly. Since we want to have off-the-shelf reasoners, we have limited ourselves to OWL2, though nominals, transitivity and even inverses are not strictly needed for reasoning.

$$\text{Goal}\langle ?V \rangle \equiv \text{Goal} \sqcap (\text{evidence} : ?V) :- \text{EvidValues}\langle ?V \rangle$$

For more complicated cases, and more uniform notation, we might prefer to use rules extending the syntax of higher-order logics such as $\text{Hi}(\mathcal{D})$ [11]:

$$\begin{aligned} \mathbf{Define}_C(\text{Goal}\langle ?V \rangle, \mathbf{And}(\text{Goal}, \mathbf{HasValue}(\text{evidence}, ?V))) :- \\ \mathbf{InstanceOf}_C\langle ?V, \text{EvidValues} \rangle \end{aligned}$$

Returning to BIM, we need to also say that strong implies weak evidence, using axioms:

$$\text{Thing}\langle \text{SF} \rangle \sqsubseteq \text{Thing}\langle \text{WF} \rangle \quad \text{Thing}\langle \text{SA} \rangle \sqsubseteq \text{Thing}\langle \text{WA} \rangle$$

Information about the *pursuit* property can also come from multiple situations, so its value is a subset of $\{Pur, NegPur\}$.

3.2 BIM Relationships: Their Domains and Ranges

Influences. The *influences* relationship is used to represent the (transmission of) (un)favorable effects on situations. As natural, we represent BIM relationships by DL properties, so we have simply

$$\text{influences} \sqsubseteq_r \text{Situation} \times \text{Situation} \quad \text{infBy} \equiv_r \text{influences}^-$$

Borrowing from GM [6], there are a variety of influence links: a ++ (resp. +) represents strong (resp. partial) positive influence on *evidence*, and a --/- influence link represents strong/partial negative one. In Fig. 1, “Strong economic growth” has a partial positive influence on “Increase sales”. Influence links also can affect the *pursuit* of goals, using optional influence annotations P and !P, representing pursuing and its denial respectively. The different kinds of labels on influence will be encoded thru sub-properties of *influences*, with the original labels as prefixes separated by an underscore. Thus “StayCompetitive” positively influences “IncreaseSales” from Fig. 1, would be encoded, in part, by the axiom

$$\text{IncreaseSales} \sqsubseteq \exists +_infBy.\text{StayCompetitive}$$

Refines. The *refines* relationship helps decompose concepts into other, often more detailed, concepts of that type. *Refines* is also used to determine evidence for/against a thing, based on the *evidence* for/against its refinements. Unlike other relationships (or UML associations), *refines* is limited to different pairs of sub-(domain,range) pairs. Thus a goal refines other goals (not other kinds of situations), but goals can be refined into goals, domain assumptions (DA) or tasks:

$$\text{Goal} \sqsubseteq \forall \text{refines}.\text{Goal} \quad \exists \text{refines}.\text{Goal} \sqsubseteq (\text{Goal} \sqcup \text{DA} \sqcup \text{Task})$$

Every other subclass of Situation, except Task, only has axioms like

$$\text{Situation} \sqsubseteq \forall \text{refines}.\text{Situation} \quad \exists \text{refines}.\text{Situation} \sqsubseteq \text{Situation}$$

Refinements are by default interpreted disjunctively, but can also be marked as explicitly AND-ed: e.g., both “Facilitate card processing” and “Select type of card(s)” are required to satisfy “Offer cards”. Since on any particular node, we want all refinements to be AND-ed or OR-ed, we add a subclass AND_Thing of Thing, and have axioms:

$$\text{AND_Thing} \sqsubseteq \text{Thing} \quad \text{OR_Thing} \equiv \text{Thing} \sqcap \neg \text{AND_Thing}$$

These concepts will be used below to define the propagation of evidence values for AND and OR refinements.

3.3 Evidence and Pursuit

Recall that each BIM thing has an *evidence* property, with value a subset of {SF, WF, WA, SA} and *pursuit* property, with value a subset of {Pur, NegPur}. We provide the precise rules for relating both *evidence* and *pursuit* values in the presence of *refines* and *influences* relationships between nodes.

For *refines*, we use the rules for combining evidence on AND and OR nodes inspired from [6]. Positive evidence values from the sources are propagated to the target according to its node kind: on an OR node, it is enough to have one refiner with $V=SF,WF$ to get V ; on an AND node, all refiners must have V :

$$\begin{aligned} \text{OR_Thing} \sqcap \exists \text{refinedBy.Thing}\langle?V\rangle &\sqsubseteq \text{Thing}\langle?V\rangle \quad :- ?V \in \{SF,WF\} \\ \text{AND_Thing} \sqcap \forall \text{refinedBy.Thing}\langle?V\rangle &\sqsubseteq \text{Thing}\langle?V\rangle \quad :- ?V \in \{SF,WF\} \end{aligned}$$

For negative evidence, the converse holds:

$$\begin{aligned} \text{OR_Thing} \sqcap \forall \text{refinedBy.Thing}\langle?V\rangle &\sqsubseteq \text{Thing}\langle?V\rangle \quad :- ?V \in \{SA,WA\} \\ \text{AND_Thing} \sqcap \exists \text{refinedBy.Thing}\langle?V\rangle &\sqsubseteq \text{Thing}\langle?V\rangle \quad :- ?V \in \{SA,WA\} \end{aligned}$$

Note that the presence of common DL concept constructors, such as qualified number restrictions $\geq n R.C$, immediately suggests extending BIM to support AND(n) nodes, which require the satisfaction of at least n refinements.

For *influences*, ideally we would separate the evidence and pursuit aspects, but the desired semantics sometimes requires knowing both aspects at the same time. So we introduce a taxonomy of properties, starting with leafs like $++P_InfBy$. These are then grouped in various ways as subproperties of others like $InfByP$ and $infBy++$; in turn, $infBy++$ and $infBy+$ are subproperties of $infByPositively$. This allows us to later state some axioms once, for a higher property, rather than repeat it for each subproperty.

The evidence values of the source are propagated to the target depending on the strength of the source evidence and the influence label. The 12 rules from [6] could then be written as 12 axioms, such as

$$\exists \text{infBy}++. \text{Goal}\langle SF \rangle \sqsubseteq \text{Goal}\langle SF \rangle$$

However, if we used parametrized classes and axioms, the following shows much more intuitively what happens in the 6 axioms involving positive influence:

$$\exists \text{infByPositively.} \text{Goal}\langle?V\rangle \sqsubseteq \text{Goal}\langle?V\rangle \quad :- \text{EvidValues}\langle?V\rangle$$

The meaning of negative influences requires more complexity, because the “polarity” of the evidence value must be switched. Using P-facts $\text{complement}(SF,WF)$ and $\text{complement}(SF,WF)$, with symmetric *complement*, we encode the 6 other axioms with the rule

$$\exists \text{infByNegatively.} \text{Goal}\langle?V\rangle \sqsubseteq \text{Goal}\langle?W\rangle \quad :- \text{complement}\langle?V,?W\rangle.$$

The propagation of pursuit along influence links from goals to goals is similar to that of evidence with *Pur*, *NotPur*, *P*, *!P* playing the role of $++$, $--$, *SF*, *SA* respectively. Again, we can choose 4 ordinary axioms, or 2 parameterized ones.

Reasoning with Missing Pursuit. Recall that only goals have a *pursuit* attribute. This means complex special cases.

If the source does not have a *pursuit* attribute, e.g. a situation influencing a goal, the satisfaction polarity of the source determines the pursuit of the target in P and !P influence types. This is one of the 4 axioms for this:

$$\exists \text{InfByP} . (\neg \text{Goal} \sqcap (\text{Thing}\langle \text{SF} \rangle \sqcup \text{Thing}\langle \text{WF} \rangle)) \sqsubseteq \text{PurGoal}$$

If the source has a *pursuit* attribute but the destination does not, e.g. a goal influencing a situation, then the influence of the source *evidence* on the destination *evidence* only occurs when the goal is pursued, in case P is on the label, or not pursued, in the case of !P. For example, if the goal is satisfied and pursued, and the label is +P, the target situation partially occurs. If the goal is satisfied and not pursued, the situation has no incoming evidence from that goal. This requires replacing each of the original 12 axioms for propagating *evidence* by pairs such as

$$\exists ++\text{P_infBy} . \text{Goal} \sqcap \text{PurGoal} \sqsubseteq \text{SFThing}$$

$$\exists ++!\text{P_infBy} . \text{SFGGoal} \sqcap \text{NotPurGoal} \sqsubseteq \text{SFThing}$$

which check the appropriate combination of edge and node labels. Again, these could be stated much more succinctly by using parametric concepts and properties:

$$\exists \text{infBy}\langle ?S, ?P \rangle . (\text{EGoal}\langle ?S \rangle \sqcap \text{PGoal}\langle ?P \rangle) \sqsubseteq \text{EThing}\langle ?S \rangle \quad :- \\ \text{EvidValue}\langle ?S \rangle, \text{PursuitValue}\langle ?P \rangle.$$

where we must now distinguish parameterizing concepts like Goal and Thing by *evidence* or by *pursuit*

$$\text{EGoal}\langle ?V \rangle \equiv \text{Goal} \sqcap (\text{evidence} : ?V)$$

$$\text{PGoal}\langle ?V \rangle \equiv \text{Goal} \sqcap (\text{pursuit} : ?V)$$

Translating BIM Models to DL Given a specific model, such as the one in Fig.1, we need to generate axioms that connect it to the generic terms of BIM axiomatized above. For this, we make every node a class, and add axioms describing its “BIM type”. For example, node “Offer International Banking”, which is a goal, would generate:

$$\text{OfferInternationalBanking} \sqsubseteq \text{Goal} \sqcap \text{AND_Thing}$$

We also add axioms declaring the disjointness of all nodes. Finally, every edge is translated into DL axioms in a manner that respects the following intuition of GM users: for every instance of a top-level goal, there is a separate set of instances connected to it, which result in an isomorphic copy of the (concept level) graph. This is assured by pairs of axioms, illustrated for the + influences edge from StayCompetitive to IncreaseSales:

$$\text{IncreaseSales} \sqsubseteq \exists +_ \text{infBy} . \text{StayCompetitive}$$

$$\text{StayCompetitive} \sqsubseteq (= 1 +_ \text{influences} . \text{IncreaseSales})$$

CWA axioms such as $\text{OfferCards} \sqsubseteq (= 2 \text{ refinedBy} . \text{Thing})$ complete the encoding of the graph.

“What if” Scenarios Frequently, business managers want to explore “What if?” scenarios, such as “How is the evidence for/against any particular model

element affected if our organization offers cards but does not offer international banking?”.

There are two approaches to such explorations. The first, more comfortable for domain experts, who view element models as propositions, is carried out at the class level. Thus, we would add:

$$\text{OfferCards} \sqsubseteq \text{EGoal}\langle\text{SF}\rangle \quad \text{OfferInternationBanking} \sqsubseteq \text{EGoal}\langle\text{SA}\rangle$$

and then check whether `BroadRangeOfServices` is classified as a subclass of `EGoal(SF)`, `EGoal(SA)` respectively. One can similarly check the classification of any other component, such as `IncreaseRevenue`, to see the effect of these assumptions on it.

One might also want to answer a different question: “Is it possible to fully satisfy `BroadRangeOfServices`?” At its simplest, this is just adding the axiom

$$\text{BroadRangeOfServices} \sqsubseteq \text{EGoal}\langle\text{SF}\rangle$$

and wait for the reasoner to detect any inconsistent concepts. Using the ability of DLs to represent incomplete information, one could of course also explore less precise scenarios (e.g., “What if we offer cards *or* international banking”).

A final variant of exploration, supported for goals in [6], is finding what (minimal) set of tasks and domain assumptions must hold if some goal is to be achieved. For this purpose one can add axioms corresponding to “predicate completion”, which end up *defining* AND and OR nodes in terms of the classes that refine them. Standard DL reasoning would then indicate what task must be executed in all circumstances if `OfferCards` is to be fully supported. However, one must use *abduction* to find a set of tasks that are sufficient to satisfy it. Unfortunately, abduction for highly expressive languages such as OWL2 has not been studied. (In [6], this is achieved using *min-SAT* algorithms for the propositional encoding.)

The alternative approach to studying scenarios, more natural to those familiar with DL modeling, would be to create an A-Box with individuals describing the particular goals, etc. being considered. For example, it would contain A-Box assertions such as `BroadRangeOfServices(brs1)`, `OfferCards(oc1)` and `refines(oc1,brs1)`. One can then provide *evidence* for a scenario, such as `SFGoal(oc1)`, and check for consequences on individuals.

The advantage of such an approach is that it does not require changing the schema (enabling better run-time monitoring), as well as allowing the co-existence of models for multiple businesses, with potentially overlapping individuals. The disadvantage is that for a single business, one essentially duplicates the concept level axioms in the A-Box.

3.4 BIM Meta-properties

Rather than simply make BIM the union of all sorts of unrelated concepts found in other business analysis languages (e.g., Vision, Mission, Strategy (BMM), Softgoal, Hardgoal (GM), Initiative (BSc)), an ontological analysis was performed on their underlying meaning. The result is a set of six meta-properties: *duration* (long-term/short-term), *likelihood of fulfillment* (high/low), *nature of*

definition (formal/informal), *scope* (broad/narrow), *number of instances* (many/few), and *perspective* (financial/ customer/ internal/ learning and growth).

New, more specialized, BIM subconcepts can now be obtained using values for these metaproperties. For example, the BMM concept of a **Vision** is a “goal with a long duration, broad scope, low chance of fulfillment, informal definition, and few instances”. Examples of Visions from our credit card organization could include “Stay competitive” or “Have a worldwide presence”.

Not all meta-properties must take on specific values in order to express a more specialized BIM concept. Thus, **Vision** does not deal with *perspectives*. And, **Softgoals/Hardgoals** from GM can just be goals with an informal/formal definition, leaving the values of other metaproperties open.

The values of metaproperties at the class level do not constrain class instances, but only say something about the nature of instances, that they are likely or generally conform to the expressed metaproperties.

The representation of metaproperties in DLs is known to be problematic, especially in our case, where we want the metaproperties to behave so that restricting their possible values results in subclasses. However, this is exactly the behavior one would get if the metaproperties were treated as ordinary (functional) properties. So we could just add property axioms like

$$\text{number_of_instances} \sqsubseteq_r \text{Thing} \times \{ \text{few, many} \}$$

and then define classes

$$\text{Vision} \equiv \text{Goal} \sqcap (\text{number_of_instances} : \text{few}) \sqcap \dots$$

$$\sqcap (\text{nature_of_definition} : \text{informal})$$

$$\text{SoftGoal} \equiv \text{Goal} \sqcap (\text{nature_of_definition} : \text{informal})$$

DL reasoning would then automatically classify Vision as a subclass of SoftGoal. The main difficulty with the above approach is that this conceptual model no longer makes sense intuitively, since it associates with individual goal g , belonging to Vision, (which I might be pursuing tomorrow), the property *number_of_instances*, with value *few*. It would be much more desirable to be able to have real meta-properties of classes, but then state that, for a group of such metaproperties, value restrictions result in subclasses at the class (rather than metaclass) level.

Ideally, one would be able to state rules dealing with meta-properties, like *duration*, such as the following (*namedC/1* is a predicate that is true of atomic concept names used in DL axioms) :

$$?C \sqsubseteq ?D \quad :- \text{namedC}(?C), \text{namedC}(?D), \text{not } \text{diff_duration}(?C, ?D).$$

$$\text{diff_duration}(?C, ?D) \quad :- \text{duration}(?C, ?X), \text{duration}(?D, ?Y), ?X \neq ?Y.$$

Since *duration* is functional, this says that C is a subclass of D as long as D has not been specified to have a different meta-property value than C (i.e., D’s duration is unrestricted, or restricted to the same value as C’s). The idea is that such rules are interpreted as in Logic Programming, with negation as failure. (The precise semantics of such rules is given in Section 4.1.)

Since we want to do this for an entire set of meta-properties, we could try to use the idea of HiLog [13] to allow variables ranging over named properties, stating rules like

different_on_some_metaProp(?C,?D) : –
namedC(?C), namedC(?D), namedI(?V), namedI(?W),
?MP ∈ {duration, scope, ...}, ?MP(?C, ?V), ?MP(?D, ?W),
?V ≠ ?W.

or simply use ternary P-assertions of the form *hasValue(?C, ?MP, ?V)* in the formula above.

4 More on Parametric Concepts and Rules

The Galen ontology of medical concepts [14] provides further evidence for the utility of parametric concepts/axioms. Consider the pervasive use of so-called **Selectors**. Here is one example of its use in the OWL translation of Galen (shortened by eliminating ‘Object’ vs ‘Data’, and URIs):

```
Declaration(Class(#LeftEye))
EquivalentClasses(#LeftEye
  IntersectionOf(SomeValuesFrom(#hasLeftRightSelector #leftSelection)
    #Eye))
Declaration(Class(#RightEye))
EquivalentClasses(#RightEye
  IntersectionOf(SomeValuesFrom(#hasLeftRightSelector #rightSelection)
    #Eye))
```

The following parametric declaration

```
Declaration(Class(#Eye<?LR>), ?LR in {#leftSelection, #rightSelection})
EquivalentClasses(#Eye<?LR>
  IntersectionOf(SomeValuesFrom(#hasLeftRightSelector ?LR) #Eye))
```

is meant to capture the two axioms, using an enumeration of possible concept values for the variables. Instead of the name *#LeftEye*, we would then use *#Eye(#leftSelection)*, or more likely abbreviate the values, and say *#Eye(#left)*. Both *#Eye(#left)* and *#Eye(?V)* could then be used in axioms. If this was the only example, the gain would not be much. But there are far more complex definitions involving *#hasLeftRightSelector*. And the above kind of repetition occurs for everything we have two in our body due to vertical symmetry. Also, there are other selectors, including *#hasPositionalSelector*, *#hasMedialLateralSelector*, *#hasAnteriorPosteriorSelector*, some with more than 2 values, while some concepts, such as *#LeftInferiorPulmonaryVein*, combine multiple selectors.

In fact a grep of the Galen OWL files revealed over 26,000 lines containing **Selector** (and naming in Galen is very systematic). So our approach would not only eliminate roughly half these axioms, but, more importantly would make *maintenance of the ontology much easier and likely to be correct*, by lessening the chance of errors when the definitions are modified later, since it is highly likely that both definitions need to be changed the same way.

Note also that in BIM, the set of qualitative values such as *StrongEvidenceFor* (SF), etc. might be extended to include more alternatives, such as *VeryStrongEvidenceFor* (VSF). For our above axiomatization of evidence propagation, this could be handled by adding only three more P-assertions: *EvidValues*(VSF), *EvidValues*(VSA), and *complement*(VSF, VSA) – clearly showing that we had captured significant patterns in our rules.

4.1 Formal and Computational Aspects

The following is a sketch of the simplest formalization of the hybrid DL+rule language we used to give the semantics of BIM. The set of primitive identifiers is split into *atomic* ones and *parametric* ones. Then axioms are formed as usual, except that parametric identifiers require atomic primitive constants or variables as arguments, and variables may occur alone as identifiers in axioms. However, any non-ground DL axiom must appear as the head of a rule, whose body binds positively all the variables in the axiom. Rules have the form

$$\gamma(\mathbf{X}) : - r_1(\mathbf{Y}_1), \dots, r_k(Y_k), \mathbf{not} s_1(\mathbf{Z}_1), \dots, \mathbf{not} s_m(Z_m)$$

where r_i and s_j are P-predicates (i.e., not in the signature of the DL), and all variables in \mathbf{X} and \mathbf{Z}_j appear among the variables in \mathbf{Y}_i for some i . The P-atoms in the body are constructed using variables or constants, some of which are *atomic* identifiers from the DL. γ is either another P-atom, or a DL axiom with free parameters \mathbf{X} . When $k = m = 0$, if γ is a P-atom then we have a P-assertion, such as *EvidValues(SF)*; otherwise, it is an ordinary (ground) DL-axiom.

The semantics of the resulting hybrid system obeys the desirable property of “modularity of reasoning” [12], by (i) using the rules first to obtain a complete set of variable-free DL axioms, and then (ii) using pure DL reasoning on the result. The semantics of rules are the standard stable-model semantics of logic programming with default negation (see [12] for a summary), with the Herbrand universe of a set of rules consisting of constants appearing in P-assertions or atomic primitive constants occurring in ground axioms. The semantics of DL are also standard, except that names of the form $C\langle d_1, \dots \rangle$ receive interpretation as atomic concepts, when there are no variables in the arguments.

In our case the rules are restricted to be non-recursive, so there are no problems with infinite Herbrand universes, decidability and complexity in part (i): one can use bottom-up evaluation as for stratified Datalog⁻; so the complexity will likely reduce to that of part (ii), since we are using a fairly expressive DL. (The precise details of this formalization can be found at <http://www.cs.rutgers.edu/~borgida/BIM/d112.appendix.pdf>.)

As usual, the semantics should not be taken as guide to preprocess the KB and eliminate rules. First, if the benefits of parametric concepts for KB maintenance are to be realized, then the rule format should be maintained for editing, etc. Second, lightweight type-checking techniques used in Programming Languages can be used to detect certain errors in axioms without theorem proving. Third, even for absorption in tableau implementations one can perform unification to see if the parameterized axiom should be applied. Only experimental evaluation can tell whether this would result in speed-ups in an ontology like Galen, where thousands of axioms might be eliminated.

5 Summary

The presentation of BIM semantics as translation into DL provided several potentially interesting observations for the DL community.

Foremost, it led us to consider **parametric** *concepts, axioms* and *rules*. We have only scratched the surface of this area, and there remain lots of formal questions on how far one can push this in terms generalizing the syntax, semantics, complexity, and implementations.

In addition, we provided a novel way to express so-called “goal reasoning” using DL constructors. This translation makes possible the posting of goal models on the Semantic Web, and made evident the possibility of a useful “at least k subgoals need to be satisfied” variant of AND decomposition. However, in order to compete with [6], this requires more research in DL on abduction in languages more expressive than \mathcal{ALC} : the minimal language needed in our translation requires the ability to state that attributes are functional.

Acknowledgments We are very grateful to Daniel Amyot, Daniele Barone, Lei Jiang, and Eric Yu for co-developing and applying BIM.

References

1. Business motivation model, version 1.0. Object Management Group (2004).
2. Kaplan, R. S., Norton, D. P.: Strategy maps: converting intangible assets into tangible outcomes. Harvard Business School Press (2004)
3. Kaplan, R. S., Norton, D. P.: Balanced scorecard: translating strategy into action. Harvard Business School Press (1996).
4. Yu, E.: Towards modeling and reasoning support for early-phase requirements engineering. In: Proc. IEEE Int. Symp. on Req. Eng. (1997).
5. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming 20(1-2) (1993).
6. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Formal reasoning techniques for goal models. Journal of Data Semantics (2004).
7. Barone, D., Yu, E., Won, J., Jiang, L., Mylopoulos, J.: Enterprise modeling for business intelligence. In: Proc. PoEM'10 (2010).
8. Jiang, L., Barone, D., Amyot, D., Mylopoulos, J.: Strategic models for business intelligence: reasoning about opportunities and threats. In Proc. ER'11 (2011)
9. Barone, D., Jiang, L., Amyot, D., Mylopoulos, J.: Composite indicators for business intelligence. In Proc. ER'11 (2011).
10. Berardi, D., Calvanese, D., de Giacomo, G.: Reasoning on UML class diagrams. Artificial Intelligence (2005) .
11. de Giacomo, G., Lenzerini, M., Rosati, R.: Higher-order description logics for domain metamodeling. In: Proc. AAAI-11 (2011)
12. Rosati, R.: Integrating ontologies and rules: semantic and computational issues. In: Reasoning Web'06 (2006).
13. Chen, W., Kiffer, M., Warren, D. S.: HILOG: a foundation for higher-order logic programming. J. Logic Programming 15(3) (1993).
14. OpenGALEN. <http://www.opengalen.org/>