Structural Modeling in Biomedical and Product Engineering

Henson Graves
Algos Associates
henson.graves@hotmail.com
2829 West Cantey Street, Fort Worth, Texas 76109

Abstract. Structural modeling is common for biomedical systems as well as manufactured products. Modeling effectiveness depends on the relationship between the model and the systems which realize or conform to the model. Does the model over or under specify what it is intended to describe? Examples and experience from manufactured products and biomedical fields help clarify modeling issues applicable to each field. Biomedical descriptions have employed axiom sets directly to describe and reason about molecular and biological structure. However, the logics used for biomedical modeling have difficulty representing diagrammatic structure common to manufactured systems. Embedding engineering models as axiom sets within logic enables greater precision in specifying models and in unifying modeling with reasoning. Characterizations of realizations of engineering models enable computations from a model to apply to its realizations. Suggestions are made for expanding the expressiveness of modeling languages such as SysML to include Description Logic language constructions. Conversely suggestions are made to generalize Description Logic to directly represent the diagrams that occur in both manufactured products and biomedical systems. Results are outlined which can be used to characterize classes of structural models all of whose realizations have the same structure. These results build on both the expressiveness found in engineering modeling languages such as SysML and in Description Logic, which is the formal basis for OWL, the Web ontology language. This discussion is intended for model practitioners in both fields rather than being an exposition of mathematical proofs of results.

Introduction

Structural modeling is common for biomedical systems as well as manufactured products. In biomedical engineering the models are often called descriptions. However, in both fields models are used to specify products and describe existing systems. Product engineering uses design models to specify a product to be built, but also builds descriptive models of existing products and the physical operating environment of a product. Biomedical engineering not only builds descriptive models to classify molecular and other structure, but construct design models to specify drugs and artificial components for biological systems. When a model is used as a design specification it is a prescription for something to be built. A design model often includes criteria to determine if a product example conforms or realizes its design. In biomedical engineering a description of a class of systems is often used to diagnose disease from symptoms. Products design models are also used to infer the causes of malfunction from symptoms. Examples and experience from both fields clarifies modeling issues applicable to each field and helps determine the appropriate mathematical and logic foundation framework for structural modeling.

The effectiveness of model usage depends on the relationship between the model and the systems which realize or conform to the model. Criteria which have been suggested for evaluating the effectiveness of a model for an application are: (1) correctness, no model constraint is wrong, i.e., it does not accord with the ground truth of the domain of application, (2) precision, non-intended models are excluded, and (3) accuracy, negative examples are excluded. In engineering applications the term "ground truth" corresponds to the relationship between descriptive models and the real world. In applications such as product requirements verification there are fairly precise if un-formalized methods in which the relationship between and ground truth is assessed. While the answers to these questions depend on the usage of the model, formal mathematical logic provides a framework for establishing characterizations of the realizations of a model. Mathematical methods can be used to tell if a model is so over specified that it has no realizations. This means that the model is inconsistent.

Specific meta mathematical results about classes of models can be used to determine when all of the realizations of a model are structurally identical. When all of the realizations have the same structure the model can be used to compute properties from the model which apply to all of the realizations. If all of the systems have exactly the same components each with the same weight then a weight computation from the model will be valid for all of its realizations. However, to apply these results one needs to be sure the model does not under specify the intended systems. When a model intended as a normative description of a class of systems such as the human heart or an automobile model the realizations are expected to have common structure but not necessarily be identical. Increasingly it becomes desirable to add disease and fault propagation information to models and used design models for analysis.

This paper illustrates how models can be embedded within logic to obtain additional precision of expressiveness. These results build on both the expressiveness found in engineering modeling languages such as SysML (OMG SysML 2008) and in Description Logic (Baader at al. 2010), which is the formal basis for OWL (Horrocks, et al. 2006) the Web ontology language. A mathematical result is outlined and illustrated that can be used to characterize classes of structural models all of whose realizations have the same structure. The results discussed here are limited to static descriptions, but provide the basis for generalization to models for dynamical systems.

Structural Descriptions

The diagram in Figure 1 is an example of what an engineer or scientist calls a model. The diagram is used to describe a structural pattern common to human hearts. Structures which have components and connections between the components occur commonly in manufactured products, family relationships, and biomedical systems. A structure description or model is used to specify and describe structures. The structures which conform to the description are called realizations. A structure description often starts with a diagram of nodes and edges such as Figure 1 which is an abstraction of the human heart. The diagrams may be informal or be represented within a modeling language such as SysML. In Figure 1 the nodes are labeled with components of the heart and the edges indicate relationships between these parts. As a description the nodes are types of objects and the edges are types of relationships. In the heart example the arrows are of two kinds; part arrows which describe the component structure and connection arrows which represent connections between components. In Figure 1 the part arrows are colored red and the connection arrows are colored blue. The part arrows describe the component decomposition. The blue arrows connect nodes within the context of the diagram.

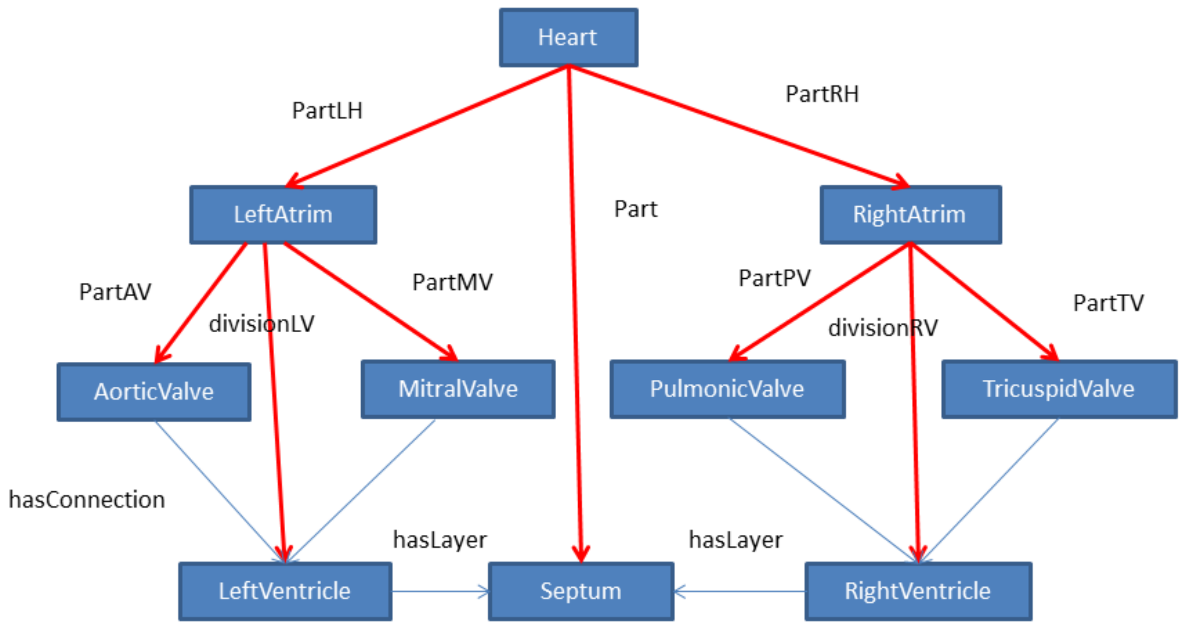

Figure 1. An Abstraction of the Human Heart

A full application description will include much more than the diagram and will require more precise language constructions and additional assumptions. However, since a structure diagram is often the starting point for constructing models the discussion will start from the diagrams and proceed to how the information can be embedded as an axiom set within logic.

Why a Logical Foundation

Why is embedding models as axiom sets within logic so important? Providing a formal logic-based semantics for language constructions is the key to semantic interoperability for modeling. Models can be developed in one context and reused within a different context without needing to be accompanied by a subject domain expert. Of course for this to work a model needs to contain information that provides sufficient conditions for its use. Currently this kind of information is not generally included in models, but this information is not hard to add in slightly more expressive modeling languages. Logic also offers richer language constructions than are currently used in modeling languages. Some of these examples are illustrated below.

A formal logical foundation is also needed to successfully integrate reasoning with modeling. Reasoning occurs in both the development and use of descriptive models, or descriptions. In product development design decisions may lead to design inconsistencies. When for example, a component is added to a design its use within the design may violate specifications for the component or violate a constraint on the design (Graves 2012). Both cases lead to an inconsistent design. For an existing product one may want to reason from pathology indications to causal connections. For example in the human heart an indicator that the left ventricle of a heart suffers from left ventricle hypertrophy is that the aortic valve suffers from aortic regurgitation. For some tasks, automated reasoning is feasible and can provide high leverage for solving difficult practical problems. The complexity of applications means that models become more complex. The result is that manual inspection and analysis of models is insufficient to determine design consistency and other design consequences.

Formal automated reasoning about structures which realize a description can be done by expressing or embedding the description as an axiom set within logic. However, for the reasoning to apply only to the applications, the axioms have to constrain the valid realizations to be the intended ones. Practically axiomatization of descriptions which sufficiently constrain the valid interpretations yet is decidable has proven difficult. Consequently reasoning often does not give the precision of results needed as the results apply to all valid models which may include non-intended models. For example when describing a molecule such as water one needs to specify that a molecule has two hydrogen and one oxygen atoms connected appropriately. In addition one needs to exclude a water molecule having other components, such as a carbon atom. However, by using a logic approach to embedding models which directly represents diagram within the logic it is possible to provide more precise understandable models and retain the decidability properties needed for automated reasoning. This approach will be outlined here with examples.

Realizations

The diagram in Figure 2 represents a simplified product description which describes components and the connections between the components for a simplified product model. The model contains two fuel tanks which are assumed to be distinct even though they have the same structure. An implementation of the design is a collection of objects and relationships between the objects which realize the design. The right side of Figure 2 describes two possible interpretations of the model. Both diagrams are valid interpretations in that they both have objects corresponding to the nodes of the vehicle diagram. However, they connect the fuel tanks in different ways. In interpretation 1 one tank is connected to the pump and the second tank is connected only to the first tank. In interpretation 2 each tank is directly connected to the pump. This ambiguity may represent only the preliminary design before the decisions have been made. It may be intentional that both realizations are permitted, or the ambiguity may be unintentional.

The current engineering and science usage of the term model is different from the logician's usage of the term. In logic a model means a valid interpretation of an axiom set. The engineer's usage of the term model is close to the use of an axiom set. The term realization has been used here to describe a structure which conforms to a model. A logician’s model in general may contain multiple realizations of the axiom set. A realization of a model/axiom set is a minimal model in that it is the smallest possible model. If the axiom set is inconsistent then there are no models.

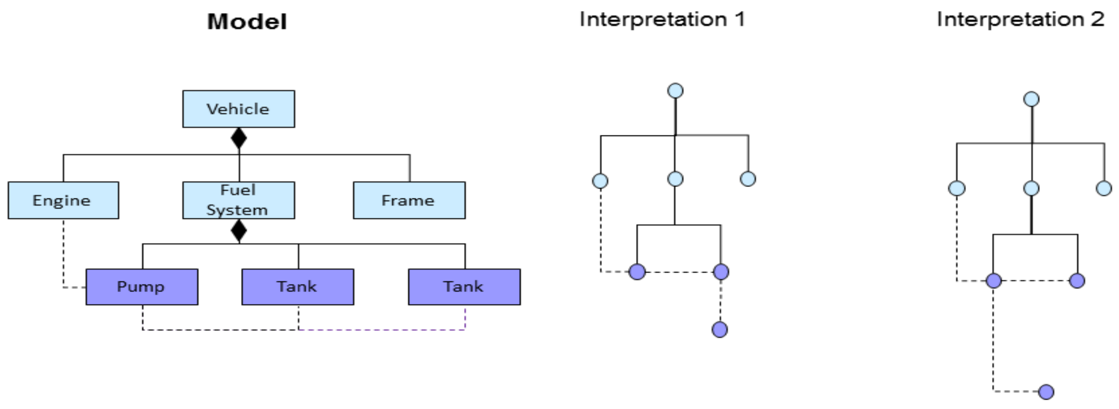

Figure 2. Relationship between Model and Interpretation

Axioms as long as they are not contradictory are used to classify their interpretations. In a sound logical system the derived results will be correct in all valid interpretations of the model. The embedding results for models in logics which are known to be sound provide the foundation for the use of automated reasoning to solve application problems. Many partial results are known (Berardi et al 2005) but much work remains to be done. By embedding a model as an axiom set many application problems are equivalent to the consistency of the axiom set (Graves 2012). Automated reasoning can be used to check consistency and other services such as checking material implication. The results can be translated back to answer engineering and science questions.

What Logic is needed for modeling

To illustrate how models can constrain realizations, add precision to specifications and descriptions, and be used for analysis of disease and fault propagation we start with a SysML (OMG SysML 2008) model of the water molecule. The modeling language SysML is as applicable to modeling biomedical systems as it is modeling manufactured products. It does lack some of the expressiveness of Description Logic, and does not have a mathematical semantics as part of its formal specification. Modeling language constructions overlap with logical language constructions, but are generally not as complete. SysML uses types called blocks and properties. Blocks correspond to classes in many modeling languages. In general a SysML property has a domain and range type. On the other hand SysML does have a well-developed graphical notation which greatly facilitates human understanding and can be directly embedded in logic. After we build the model we see what is needed to constrain its realizations to have the structure we intend for a water molecule realization to have.

Water

Figure 3 is a SysML model for water molecule. A valid interpretation may have one or more water molecule realizations. A water molecule is described as having three parts, an oxygen atom and two hydrogen atoms. The oxygen atom is connected to each of the hydrogen atoms with a bonding relation. The figure below contains two diagrams which are views of the same SysML model of a water model. This model is not a representation of a specific water model, but describes the pattern used to classify water molecules. The upper diagram called a Block Definition Diagram (BDD) shows the part structure of the water molecule. In Figure 3 the rectangles *Water, Hydrogen*, and *Oxygen* are SysML blocks. Blocks correspond to classes in other modeling languages. The arrows *hasHydrogenAtom1, hasHydrogenAtom2, and hasOxygenAtom* in the top diagram are called properties in SysML and relations in some other modeling landaus. The properties define the component structure of a water molecule. Each has a multiplicity of 1 which means that any water molecule instance has exactly 1 of each of the three parts.

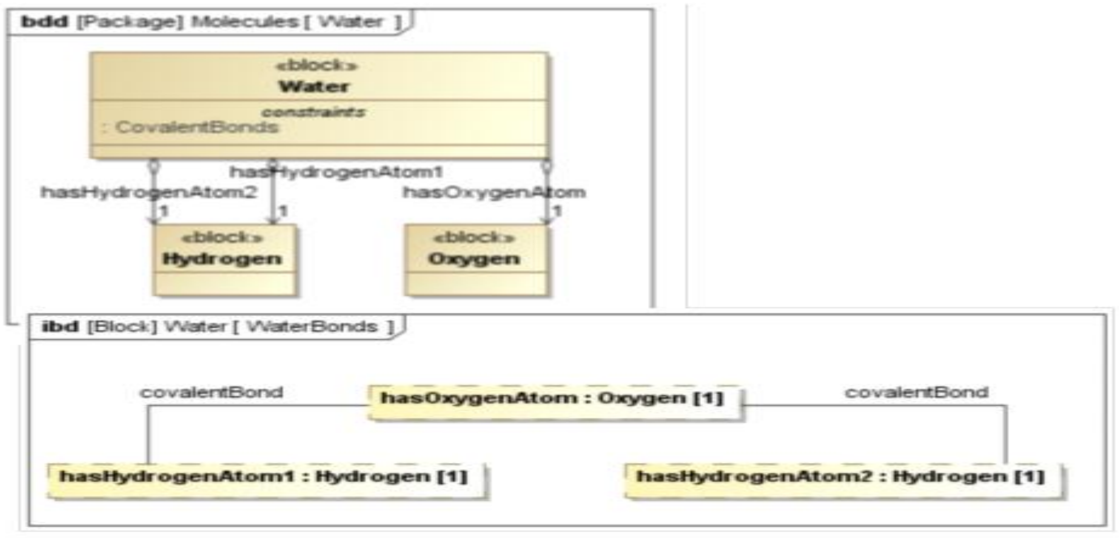
bdd [Package] Molecules [Water]
«block»
Water
constraints
: CovalentBonds
hasHydrogenAtom1
hasHydrogenAtom2
hasOxygenAtom
1
1
1
«block»
Hydrogen
«block»
Oxygen
ibd [Block] Water [WaterBonds]
covalentBond
hasOxygenAtom : Oxygen [1]
covalentBond
hasHydrogenAtom1 : Hydrogen [1]
hasHydrogenAtom2 : Hydrogen [1]

Figure 3. SysML model for the water molecule

The lower diagram in Figure 1 is called an Internal Block Diagram (IBD). The lower Internal Block Diagram (IBD) shows the internal connections between the three part properties. In the IBD the rectangles represent part properties rather than blocks. The multiplicity of 1 indicates that a property is functional. The rectangles such as *hasOxygenAtom:Oxygen[1]* denote a part property, its range, and its multiplicity. The arrow labeled *covalentbond* connecting *hasOxygenAtom* to *hasHydrogenAtom1* means that for a water molecule its oxygen atom is connected by a covalent bond to a hydrogen atom.

Informally the part properties with their multiplicity 1 say the each water molecule has three components. From the diagram the *covalentbond* relation is assumed to be defined for at least the oxygen atoms that are part of a water molecule. The connection arrows in the IBD express an equation between a composition the bonding relation with a part relation. Using a small circle for composition the equation can be expressed as:

$$hasOxygenAtom \circ covalentbond = hasHydrogenAtom1.$$

A similar equation is used to ensure that the oxygen atom in the water molecule is bonded with the other hydrogen atom of the molecule.

Informally, the intended realization of a water molecule is a tree with a root node *w1* corresponding to a water molecule and three other nodes, an oxygen atom *o1*, and two hydrogen atoms *h1, h2*. We expect that the covalent bonds from oxygen atom are connected to distinct hydrogen atoms. However, without additional axioms one could have h1 = h2. Both of the diagrams are valid models of water unless additional assumptions are made which constrains the model's realizations.

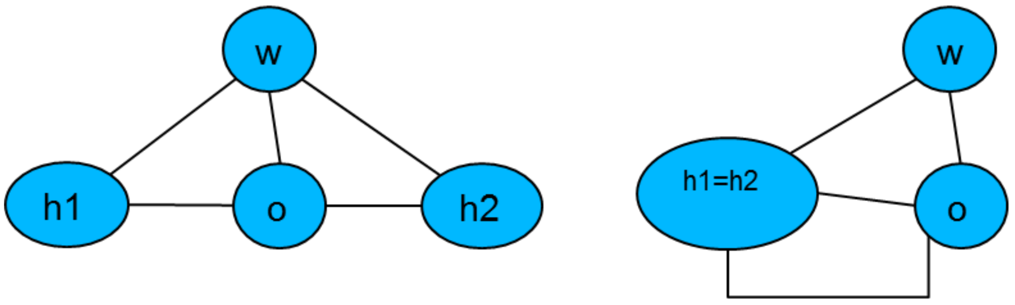

Figure 4. Two Realizations of the Water Molecule Model

Some of the constraints are so intuitively obvious that one simply doesn't think to mention them. Some of the assumptions needed are that no oxygen atoms are hydrogen atoms and conversely. Another assumption is that within a molecule the two hydrogen atoms are distinct, i.e., the intersection of hydrogen and oxygen atoms is disjoint.

Axiomatic Semantics

The assumptions on the water molecule needed to constrain the realizations can be made precise by adding some logic to the model. For example, for a class *A* and a class *B*, to say that *A* is a subclass of *B* means that the elements of *A* are elements of *B*. In a mathematical notation this is written as:

$$A \sqsubseteq B \equiv x \in A \Rightarrow x \in B$$

where the symbol "≡" means that the left and right sides are equivalent. Within a logic that uses a membership relation the disjointness of the two atoms can be expressed.For the disjointness of the atoms can be written as:

$$Hydrogen \sqcap Oxygen = \emptyset.$$

The notation *Hydrogen* $\perp$ *Oxygen* is also used. Using membership this means that

$x \in \mathit{Hydrogen} \sqcap \mathit{Oxygen} \equiv \mathit{false}.$

The membership notation can also be used for relations. In Figure 3 the relation *hasHydrogenAtom1* has domain *Water* and range *Hydrogen*. This means that if the ordered pair *<x,y>* $\in$ *hasOxygenAtom* then x $\in$ *Water* and *y* $\in$ *Oxygen*. The notation

hasOxygenAtom:(Water, Oxygen)[1]

is an abbreviation that the domain of *hasOxygenAtom* is *Water* and range is *Oxygen*. The integer in the square brackets is called the multiplicity of the relation. If the multiplicity is k it means that for the first member of a tuple in a relation there are exactly k individuals in the range for each first coordinate. When k = 1 the property is functional. That means for a relation P with domain A and range B then for <x,y1> $\in P$ and <x,y2> $\in P$ imply that $y1 = y2$. The logic that is being implicitly used has the property that a functional relation *P:(A,B)[1]* can always be replaced by a functional operator f_P. The operator f_P has the property that if <*a*,y> $\in P$ then <a, $f_P(a)$> $\in P$. For a function f we use the notation

$f{:}A \rightarrow B$

for Domain(f) = A and Range(f) = B. The operator f_P is functional in that if $a \in A$ then $f_P(a) \in B$. A function f can be applied to any individual a in its domain to yield a unique value f(a) in the range of f. Thus *hasHydrogenAtom1(w)* is a hydrogen atom. Two functions p and q with the same domain are orthogonal is for any a in their common domain p(a) $\neq$ q(a) and we write p $\perp$ q. hasHydrogenAtom1 and *hasHydrogenAtom2* satisfy an orthogonally property that reflects the fact that components in Hydrogen do not get used twice.

When a functional relation is replaced by an operator f_P the composition is written in reverse order from relational composition. The relation *covalentbond* is assumed to be functional and is replaced by a function. The expression *covalentbond(hasHydrogenAtom1)* is the composition of the two functions. The fact that the oxygen part is connected to the hydrogen part of the water molecule

covalentbond1(hasOxygenAtom) = hasHydrogenAtom1
covalentbond2(hasOxygenAtom) = hasHydrogenAtom2

While *covalentbond* is assumed to be functional its domain is not assumed to be all of *Oxygen*, but only those oxygen atoms that are components of a water molecule. That is the domain of *covalentbond* is assumed to be *Image(hasOxygenAtom)* rather than all of *Oxygen*. *Image(hasOxygenAtom)* are the individuals which are components of a water molecule. Thus *Image(hasOxygenAtom)* is a subtype of *Oxygen*.

From these axioms one can prove that any realization of the water molecule has exactly four nodes and five edges connected exactly like the diagram of the model. The argument is that any realization will have a node *w* of type *Water*. The iterations *hasOxygenAtom(w), hasHydrogenAtom1(w),* and *hasHydrogenAtom2(w)* provide four nodes. They are distinct as hydrogen and Oxygen are distinct and *hasHydrogenAtom1(w)* $\neq$ *hasHydrogenAtom2(w).*

These axioms do not preclude the components of a water molecule from being connected with some other molecule or atom. The axiom that water does not have any other part properties can be expressed with

$$\neg\, \boldsymbol{f}{:}Water \rightarrow \boldsymbol{X}$$

where $\boldsymbol{f}$ and W are variables respectively for functions and classes. Other axioms can be given to ensure that no atoms of water are connected to anything else.

The model for water has now been implicitly embedded as an axiom set in a logic. The classes and functions that occur in the description of the water molecule form a directed graph provided we replace the range classes with the images of the functions. For the water molecule the nodes of the graph are *Water, Image(hasHydrogenAtom1), Image(hasHydrogenAtom1)* and *Image(hasOxygenAtom).* The edges are

$$\begin{aligned}
&hasHydrogenAtom1{:}Water \rightarrow\ , Image(hasHydrogenAtom1),\\
&hasHydrogenAtom2{:}Water \rightarrow Image(hasHydrogenAtom2),\\
&hasOxygenAtom{:}Water \rightarrow Image(hasOxygenAtom),\\
&covalentbond{:}\ Image(hasOxygenAtom) \rightarrow Hydrogen.
\end{aligned}$$

This graph has the node *Water* as a root, i.e., no edges have *Water* as their target. There are no compositions of the part functions, only compositions with the covalent bond functions. A generalization of these properties, which hold for many structural descriptions, imply that the consistency of the model is decidable.

Description Logic Constructions

OWL 2 is the most common modeling language with a formal logical foundation. The foundation for OWL 2 (Horrocks, et al. 2006) is SROIQ which is a Description Logic (DL) (Baader at al. 2010). Description Logic (Baader at al. 2010) axiom sets have been used for modeling human anatomy and molecular chemistry (Dumontier 2007) as well as engineering (Graves and Bijan. 2011). The type operations $\sqcap$, $\sqcup$, ¬, defined for a universal type, *Thing*, are obviously useful to make models more precise; the DL language constructions would be useful extensions to SysML. The DL quantification types ∀P.A and P.∃A correspond to modeling language constructions. The modeling language construction

$$P{:}(A,B)[k]$$

corresponds to the Description Logic assertion:

$$A \sqsubseteq \exists\, [k]P.B.$$

Class operations can be extended to relations which is also useful. For example, an automobile design model might include power connections between the engine and the generator as well as the axel. Each of the individual relations is a function. However, if we want to describe exactly what is powered by the engine one could define a powers relation using the DL constructions as

$$powers = powersAxel \sqcup powersGenerator.$$

Thus *powers* is a relation which describes exactly which components are powered by an engine.

The ability of Description Logic to represent graph structures with parts and arbitrary connections and constrain them is limited. DL does not have functions, only functional relations. Further the relational equations such as those used in the water molecule model are prohibited as in general they lead to undecidability of axiom set consistency.

Structure Descriptions

The logic introduced enabled us to constrain the realizations of water to what was intended. The properties of the water function graph can be generalized to apply to other applications. A structure description is an axiom set which contains a function graph as part of its primitive terms (signature). The directed graph whose nodes are the domains and ranges of the function terms and whose edges are the functions is called the function graph. The function graph has a root *S* and that root is the only root. For *S* to be a root for any part map p in the signature one has Range(p) ≠ *S.* In this case the axiom is written as *Root(S).* To say that S is the only root means that

$$Root(S') \text{ implies } S = S'.$$

Two nodes are orthogonal if their intersection is empty. Thus for any two nodes A and B

$$A \perp B \text{ iff } A \sqcap B = \emptyset.$$

Two maps p and q with the same domain are orthogonal is for any a in their common domain p(a) ≠ q(a) and we write p ⊥ q. Another assumption for a Structure Diagram is that no part map has the same domain and same range. More generally we assume that no part path loops back on itself. That is for any composable sequence of maps no node occurs more than once as the domain or range of a part in the sequence. The composition of such a sequence is said to be antisymmetric which we write as Antisymmetric(p) for a part path p. In a Description Diagram all part paths are antisymmetric. This assumption reflects the informal idea that a component cannot be a part of itself. The connections and connection paths can have loops. The assumption made is that each path connection r is of the form *r(p) = q.*

Human Heart

The diagram in Figure 1 is an abstraction of the human heart. It was introduced to illustrate limitations of DL in its ability to constrain realizations and in order to motivate an extension of DL called the Description Graph extension of Description Logic. The limitations make it difficult to reason from axioms to conclude causes of disease from symptoms. The axioms used here for the axiom set imply that only the valid realizations have exactly the same structure as the diagram in Figure 1. The disease propagation knowledge can be safely added without affecting the decidability of questions posed using the axioms. The axioms used here for the heart description start with the directed graph of Figure 1. Its nodes are:

Heart, LeftAtrim, RightArtim, AorticValve, MirtalValve, LeftVentricle, Septum, TricuspidValve, RightVentricle

Here as in the water molecule example the classes are constrained to be those that consist of components of hearts. The graph has 15 arrows. For example

$partAV: LeftAtrim \rightarrow AorticValve$

The aortic valve is connected to the same left ventricle that is a division of the left atrium. This can be express as:

$hasConnection(PartAV) = DivisionLH.$

and the disjointness and orthogonality axioms are assumed. The result is that the complete heart axiom set is a Structure Description. Pure Structure Diagrams always have a model and so are consistent. However, Structure Descriptions may have propagation axioms which could make them inconsistent. If a Structure Diagram is consistent then it too will have a model. By assuming the disjointness of the nodes and the orthogonality of the part arrows the axioms for a Structure Description are apparent. The connection equations rule out pathology from a realization. Any realization will have 10 nodes and 15 edges connected as in Figure 1. The axioms rule out one of the connection arrows connecting a component in one structure to a component in another structure. If a heart Structure Description is included within some larger Structure Diagram then component may be connected by other arrows to components of the larger description.

In order to use the heart diagram to make inferences about heart disease from knowledge of disease about parts of the heart, additional axioms are added; for example, to make the inference that when the aortic valve suffers from aortic regurgitation when the left ventricle suffers from left ventricular hypertrophy. The disease knowledge adds the classes *hasAR* and *hasLVH* to the signature and the axiom

$Range(hasConnection|hasAR) = LeftVentricle \sqcap hasLVH$

The semantics of this axiom is that if $a \in AorticValve \sqcap hasAR$ then $hasConnection(a) \in LeftVentricle \sqcap hasLVH.$ This axiom enables the representation propagation information through connections. The heart example provides a good illustration of using propagation axioms to infer disease from symptoms.

As noted this axiom is not sufficient for the axioms used in (Motik 2008) as the axioms have a valid interpretation in which there are two left ventricles. As we have seen in the case of the fuel tank model Structure Descriptions may contain multiple identical components. However, such an interpretation is ruled out for the Structure Description axiom set for the heart. A model of the heart axiom set does not contain any hearts with two left ventricles. The components of a specific heart t are reachable by a part arrow path *p(t)* and there is only one path whose range type is *LeftVentricle*. Thus, no heart realization of this axiom set contains a heart with two left ventricles.

Logic Foundations

The three descriptions we have encountered so far, the vehicle description, the water molecule, and the human heart all include a graphical diagram which serves as part of the model. The diagrams are function graphs that have special properties which enable constraining the possible realizations and in establishing that consistency of the axiom set decidable. Additional assumptions represent constraints on their realizations. For the water molecule description we have informally given a proof that all of the realizations of water have the same structure. The properties of these models can be expressed within a logic in which these models are axiom sets. The axiom sets all include a collection of axioms called structural axioms which provide the semantics for the individuals, classes, relations, and function graphs, as well as axioms that express the properties of the diagrams.

The language used here has individuals and term constructions for classes and relations, and arrows. The term constructions for classes and relations are those found in Description Logic for concepts and roles. In addition the language has maps (generalized functions) which have composition. Structure predicates include the predicates of DL such as $=$ and $\sqsubseteq$. Additional predicate symbols may be introduced for applications. Formulae are constructed from terms using predicates using the logical operations of conjunction and implication and variables. The terms of the formulae may have variables. The language constructions have axioms provide an inference semantics for the language constructions. An axiom set is a set of formulae which contain the formulae which provide semantics for the language constructions.

This logic can be compared with other logic formalisms used to represent descriptions of biomedical systems and manufactured products. It is a generalization of Description Logic in that it contains the DL class and relation constructions as term constructions. When DL axioms are represented in the logic a DL axiom such as $A \sqsubseteq B$ where A and B are classes is represented by a ground atom. That is DL axioms are atomic formulae such as subclass and subrelation assertions between two ground class or relation terms in some subset of the Predicate symbols. As the axiom sets used here admit class and relation variables these formulae are much more general than DL axiom sets. Of course restrictions are required on the Structure Axiom Sets for decidability.

In addition to generalizing DL language constructions the logic directly represents directed graphs. The nodes of a directed graph are represented as classes and the edges as maps. By representing classes and maps as terms graph theoretic properties which involve quantifying over nodes and edges such as the property of a node being a root are first order statements in the logic. This capability enables defining axioms for a directed graph embedded in the signature of an axiom set which ensure that the graph has a finite part decomposition structure. These axioms are called Structural Descriptions. The use cases satisfy these axioms. The properties of a Structural Description are syntactically checkable from the axiom set. Further consistency of Description axioms is decidable and if they are consistent then they have a minimal model.

While the logic that we have introduced represents basic structural concepts it is not sufficiently expressive to represent what are called composite structures in engineering modeling. A composite structure such as an automobile has components which may have state, operations, and state machines which communicate asynchronously. Representing composite structures requires extending what has been informally presented here to what can be called a topos logic (Lambek and Scott 1986). The further language constructions needed are a more complete type theory, an abstraction (comprehension) type construction. A version of this kind of logic has been used as a foundation for aerospace modeling (Graves and Bijan 2012). A full topos-based logic framework is sufficiently rich that the dynamic properties of biomedical structures can be described.

Description Logics have been used to embed diagrams used in human anatomy and molecular chemistry, as well as engineering as axiom sets. While Description Logic constructions for classes and relations are natural candidates for structural modeling, the ability of a Description Logic to represent graph structures with parts and arbitrary connections is limited when the connections contain cycles. DL is restricted in the ability to constrain the models of axiom sets to represent graph theoretic structures such as the design of an automobile. For example to constrain the models of an automobile description axioms are needed to constrain an engine in one automobile from being connected to the wheels of another automobile. Limitations of DL led to an extension of DL called the Description Graph extension (Motik et al 2007). The diagram in Figure 1 was introduced to motivate Description Graph extension of DL. More recently Description Graph Logic Programs (DGLP) (Magka et al 2012) has been proposed as a framework for axiom systems to represent structural descriptions. The logic addresses these same problems and is a generalization of both the DL and DGLP approaches. Both DL and DGLP have difficulty describing graphs and ensuring that the models contain the correct realizations of a graph. Graph theoretic properties involve quantifying over nodes and edges. DL without the DG extension can represent graph nodes and edges as classes and binary relations, but DL axioms do not contain variables over classes and relations which limit what can be expressed. In DGLP nodes and edges are represented as unary and binary predicates. The statement of a graph property such a node is a root becomes higher order in DGLP.

Conclusion

The examples provide illustration that the modeling issues in biomedicine and product modeling have a lot of commonality and that a common logical foundation can serve both areas. An informal indication of what axioms for some of the class and relation constructions look like. Of course the present discussion is extremely incomplete. It is only intended to raise possibilities for an expansion in expressability, why it would be useful, and how axioms for the language constructions can work. Modeling language expressiveness needs and can be extended in many directions. The examples used here represent only one direction. As modeling language expressiveness can be expanded only as there is consensus on the informal meaning of the extensions. The constructions illustrated here have been the subject of mathematical foundation research for over a hundred years and there is now a lot of clarity as to the semantics of these constructions and even how they can be incorporated into computational systems.

References

Baader, F., Calvanese, D., McGuinness, D. L., and Nardi, D. 2010. *The Description Logic Handbook*. Cambridge University Press.

Berardi, D., Calvanese, D., and De Giacomoa, G. 2005. “Reasoning on UML class diagrams.”

Dumontier, M., Describing chemical functional groups in OWL-DL for the classification of chemical compounds. OWLED, 2007. Artificial Intelligence Volume 168, Issues 1-2.

Estefan, J.A., 2008. ``Survey of Model-Based Systems Engineering (MBSE) Methodologies," Rev. B, INCOSE Technical Publication, International Council on Systems Engineering.

Graves, H., Horrocks, I. 2008. “Application of OWL 1.1 to Systems Engineering”, OWL Experiences and Directions April Workshop.

———, 2010. Ontological Foundations for SysML, Proceedings of 3rd International Conference on Model-Based Systems Engineering.

———2008. Representing Product Designs Using a Description Graph Extension to OWL 2. OWL Experiences and Directions October Workshop.

———, Bijan, Y. 2011. Modeling Structure in Description Logic, DL2011.

———, Bijan, Y. “Using Formal Methods with SysML in Aerospace Design and Engineering” to be published in: *Annals of Mathematics and Artificial Intelligence*.

Hastings, J., Dumontier, M., Hull, D., Horridge, M,. Representing chemicals using OWL, description graphs and rules, 2010.

Horrocks, I., Kutz, O., Sattler, U. 2006. The Even More Irresistible SROIQ, in: Proc. KR 2006, Lake District, UK.

Lambek, J., Scott, P. J., Introduction to higher-order categorical logic, Cambridge University Press, 1986.

Magka, D., Motik, B., Horrocks, I., Modelling Structured Domains Using Description Graphs and Logic Programming, DL2012.

Motik, B., Cuenca Grau B., Sattler, U., Structured objects in OWL: Representation and reasoning. Proceeding of the 17th international conference on World Wide Web, 2008.

OMG Systems Modeling Language (OMG SysML™), V1.1, November 2008.

OWL 2 Web Ontology Language, W3C Working Draft 11, June 2009.

Biography

Dr. Henson Graves is a Lockheed Martin Senior Technical Fellow Emeritus and a San Jose State University Emeritus Professor in Mathematics and Computer Science. He has a PhD in mathematics from McMaster University. Dr. Graves is the principle of Algos Associates, a technology consulting firm.