Structure Diagrams in Type Theory

Henson Graves

Algos Associates
2829 West Cantey Street
Fort Worth, TX 76109 United States
henson.graves@hotmail.com

Abstract. Structures consisting of parts with connections between the parts occur frequently in science and engineering. A structural description is used to capture common properties which apply to a class of structures such as water molecules. Description Logics have been used to create axiom sets which describe structures in the sense that their models consist of structures. However, description logics (DL) are restricted in their ability to capture properties common to many applications. In DL part properties are generally represented as a single property with special axioms. Many applications can be described as having a family of part properties each with a domain and range class. The graphical syntax of SysML provides a well-developed language for specifying structure diagrams with families of part properties. An abstraction of diagrams using part property families are defined and embedded within a type theory as a restricted form of axiom set. The axiom set restrictions satisfied by a structure diagram ensure that each structure satisfying the axioms has a unique part decomposition. Type theory contains Description Logic constructions which are used in embedding structure descriptions. However, type theory significantly extends the language constructions of DL. The decidability of the consistency of a Structure Diagram is established and conditions are given on Structure Diagrams which ensure that they are templates in the sense that all minimal models are isomorphic.

Keywords: Description Logic, SysML, OWL, Type Theory, Structure Diagrams

1 Introduction

Structures consisting of parts with connections between the parts occur frequently in science and engineering applications. For example, a water molecule with its decomposition into atoms and bonds between atoms is such a structure. Similarly an automobile with its decomposition into parts and connections between them is such a structure. In science a structural description is used to capture classifying properties of the structures which realize the description. Engineering also uses structural descriptions as specifications for a system to be implemented. For example a detailed design of an automobile specifies the parts and assemblies of parts with connections between them that constitute an

implementation of the product description. For a complex manufactured system such as an aircraft the delivery acceptance criteria includes a determination that all of the parts are present and connected as specified by the system description. Both biomedicine and engineering use structural descriptions to help determine abnormalities and operating faults by comparing whether a specific structure, be it a human heart or an automobile, deviates from the description or norms that apply to instances of the description.In addition rules may be used to describe how pathology observed in one part of a system affects other parts. Observations are then used to infer causes. Diagnosis and prediction involve generally involve reasoning. These applications are now beginning to be recognized as candidates for the use of automated reasoning.

Valid reasoning requires a logic in which to axiomatize structures; one wants to verify properties of an axiom set which are then true in any model. For structure axioms a model may contain multiple instances of a structure that satisfies the model. For example a model of the water molecule may contain multiple water molecules. A single water molecule is a minimal model for a water molecule axiom set. For the axioms to capture the intended applications one wants the minimal models to be finite and one is concerned to find conditions which ensure that all minimal models are copies of each other. Then, for example, deriving the weight of a water molecule from the axioms can be applied to all instances. This paper will show that Type Theory provides a suitable logic in which structural descriptions can be represented as axiom sets. For these axiom sets tractable reasoning algorithms are available.

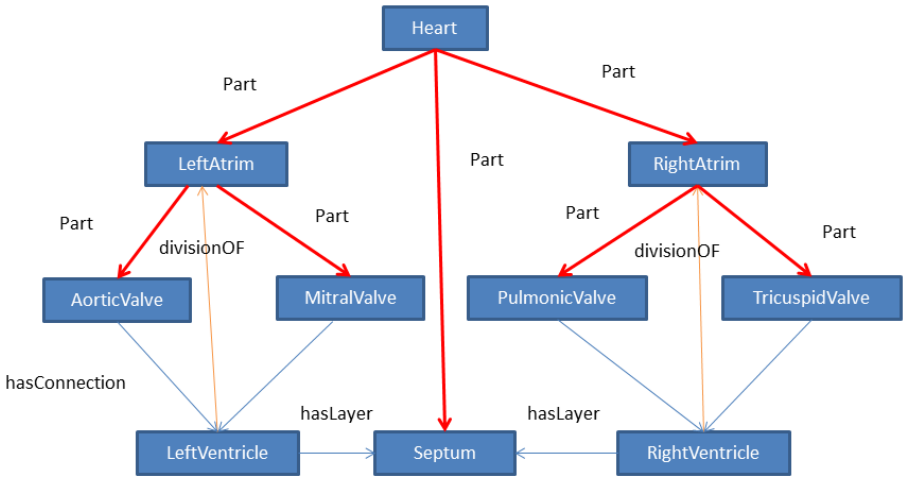

Fig. 1. Human Heart

The diagram in Figure 1 adapted from [10] is an example of an informal diagram used to abstract properties of the human heart. The vertices are labelled with components of the heart and the edges indicate relationships between these parts. For example,one might infer that $LeftVentricle$ is a subdivision of the $LeftAtrium$ of the heart. The exact meaning of the diagram is far from clear without additional information regarding diagramming conventions. This diagram is used to motivate and illustrate an embedding of Structure Diagrams into type theory [3]. In the following diagrams such as Figure 1 are embedded within type theory. While the embedding is different from that in [10] the embedding makes extensive use of DL language constructions.

Representing Structure Diagrams in Description Logics Description Logics are natural candidates for representing structures and have been used in human anatomy [6] and molecular chemistry [7], as well as engineering [4]. Description Logics use classes and binary properties as primitives. In the Description Logic literature classes are called concepts and properties are called roles. A Description Logic representation of Figure 1 would represent $LeftVentricle$ as a class corresponding to the node of that name. The class can be viewed as a specification for its instances. While Description Logic constructions of classes and properties are natural candidates for structural modeling, the ability of a Description Logics to represent graph structures with parts and arbitrary connections is limited when the connections contain cycles. The diagram in Figure 1 was introduced to illustrate limitations of DL in order to motivate an extension of DL. Vertices are translated into classes and the edges are labelled with properties in [10]. However, the edges are also used to construct DL axioms to represent the semantics of the diagram. In [10]the DL axiom

$$LeftAtrium \sqsubseteq \exists Part.AorticValve. \tag{1}$$

is used represent the meaning of the arrow from $LeftAtrim$ to $AorticValve$. The axiom states that $LeftAtrium$ is a subclass of $\exists Part.AorticValve$. The second class describes things which have a part which is an aortic valve. This semantics is used for all of the arrows in the diagram. However, it is not clear that this semantics is always correct. For example, the edge from $AorticValve$ to $LeftVentricle$ labeled $Part$ appears to have a different meaning than the arrow labeled $hasConnection$. This arrow appears to means that the aortic valve of the left atrium has a connection to the left ventricle which is the same ventricle which is the same left ventricle which is a division of the left atrium. This statement can be represented by a construction which does not appear to be used in Description Logics, called an equalizer. Informally the equalizer enables specification of hearts for which the aortic valve is connected to the same left ventricle that is a division of the left atrium.

In DL, following [2] the Part concept is generally represented as a single property that satisfies axioms such as transitivity, and irreflexiveness. However, each arrow in Figure 1 can be represented as a distinct property with a specific domain and range class. In Figure 1 each node is connected by a part relation

path to $Heart$ when the $divisionOf$ properties are viewed as part properties. Other properties express connections between the nodes. The part family and the connections are functional properties in the heart example and satisfy additional constraints which simplify the task of establishing whether a structure description is consistent. Representation of a structure diagram graphically is easier to construct than the DL representation. However, the diagram still has to be embedded within a logic. Using graphical descriptions embedded into type theory with the decidability of consistency preserved provides a practical way to develop structural descriptions.

Representing Structure Descriptions in Type Theory A structure description can be embedded as a special kind of axiom set within type theory [3]. Type theory is a deduction system. The syntax is given by term and type constructions. The semantics is given by inference rules. Type theory contains both the syntax of DL as well as that of higher order logic. Type theory contains a correspondence between the types that represent DL classes and properties, and formulas within the internal logic. A correspondence between types and formulas enables the construction of internal models of axiom sets within the logic. Type theory also has the expressiveness to capture system dynamics. However, that will not be addressed here. Type theory readily accommodates families of typed properties. The restrictions on axiom sets used for a structure diagram also enable use of a graphical syntax which corresponds to that of SysML. A proof of decidability of satisfaction for a structure diagram makes use of type theory properties to eliminate quantifiers to reduce the formula representation for a structure diagram to one which uses only a single universally quantified formula with monadic predicates. In type theory a functional property can be replaced with a Skolem map term which whose value for each argument satisfies the functional property. Since there are only a finite number of connection equations each connection equation can be replaced by a finite number of unary predicates.

2 Structure Descriptions

Structure description examples such as the human heart diagram can be specified by production rules. A structure description is a set of declarations and assertions including special assertions for parts and connections. The semantics for a structure description will be given by the type theory inference rules when the syntax is embedded within type theory. To specify the syntax the symbols A, B, C, p, q, r, and a, b, c will be used respectively for classes,properties,and individuals.

Declarations Declarations introduce atoms for classes,properties, and individuals and give their typing relationships. A declaration has one of the forms:

$$A : Class \,|\, a : A \,|\, < a, b >: p \,|\, p : (A, B)[k] \tag{2}$$

In a property declaration $p : (A, B[k])$ the A is the domain and the B is the range class of the property. The k in brackets is optional. Informally it says that the number of values for an argument instance is k. The type theory semantics for $p : (A, B)[k]$ is equivalent to the DL assertion $A \sqsubseteq \exists p.B$. This semantics will be supplied by type theory inference rules.

Assertions An assertion has one of the forms:

$$A \sqsubseteq B \,|\, A = B \,|\, A \perp B \tag{3}$$

$$p \sqsubseteq q \,|\, p = q \,|\, p \perp q \tag{4}$$

$$dom(p) = A \,|\, range(p) = B \tag{5}$$

The only property constructions allowed are finite compositions of atomic properties where the domains and ranges match. A finite composition of properties is called a path. A path is called irreflexive if $domain(pi) \neq range(pk)$ for any atomic properties pi and pj in the path. Two properties p and q are disjoint written as $p \perp q$ if $p \cap q = Nothing$.

Parts A part declaration is given as:

$$p : Part(A, B)[k] \tag{6}$$

The part declarations satisfy well-formedness axioms given below. A part $p : Part(A, B)$ is a property with $p : (A, B[k])$ for some k. The Part axioms are:
acyclic

$$\frac{\pi \text{ part path}}{\text{not } \pi : (A, A)} \tag{7}$$

orthongonality

$$\frac{p : Part(A, B)\, q : Part(C, B)}{p \perp q} \tag{8}$$

A part path is a finite composition of part property atoms. These axioms are syntactically checkable. The orthogonality axiom is used to ensure that in an implementation parts of the same type do not get reused.

Connections Connections are atomic functional properties declared between classes which are the domain or range of a part property. In an interpretation a connection is used to connect instances of classes that occur in a structure. The form of a connection declaration is:

$$c : Conn(A, B) \text{ where A and B are part classes} \tag{9}$$

Theorem 1. *Any part path in a structure description is finite and irreflexive*

A part path is irreflexive in the sense that if $domain(p) = range(q)$ for some p and q in the path otherwise it has a sub path π which has a loop and so $\pi = null$. A class in the signature of the structure only has a single occurrence as a domain or range of a property in a part path otherwise the path has a cycle. Hence any part path has finite length.

A *Structure Diagram* is a Structure Description with the following additional axioms. The part properties are functional and their inverses are also functional, and it has a unique root. A root is a class which is not the range of any part property. If the signature does not have a root then a root class can be added. As part properties are bounded they can always be replaced with functional properties. The axiom that the inverse property of a part is functional ensures that a part is not used more than once in any interpretation. The informal semantics of a connection is that for any instance of the root the chain of instances from the root to A followed by c is equal to the chain of instances from the root to B. The formal semantics will be the inference semantics for the type theory equalizer construction.

A Structure Diagram may have multiple part properties which have the same range, for example when an automobile has four wheels. However, a directed graph can be constructed which distinguishes classes which are the range of multiple part properties. The Internal Block Diagram (IBD)of a Structure Diagram is the directed graph whose vertices are the root together with the expressions p:A, where p a part property p with A = Range(p). The edges are the part properties. This graph is in named after the SysML diagram which is used to represent Structure Diagrams. An interpretation of this graph is a parts decomposition for the Structure Diagram.

Theorem 2. *Each node of the IBD is reachable by a unique part path.*

The IBD graph is a tree as it has no node with two arrows terminating at that node. If all of the nodes cannot be reached from Root, then there is another root which is a contradiction. If p and q are part paths $p1 \dots pn$ and $q1 \dots qn$ which terminate at the node $p : A$ then $< pn : A >=< qk : A >$ and so $pn = qk$. The argument is repeated on the remainder of the paths.

Theorem 3. *The number of part paths is finite.*

The number of part paths is bounded by the sum of the number of nodes times the lengths of paths from the root to each node.

The IBD graph satisfies most properties of a DG extension to DL [10]. The functional property of the part inverses insure that no two instances of the graph in an interpretation share any nodes. Further if an interpretation contains an instance of a class then it contains the whole graph as one can chain back to the root and then reach any other node. As will be seen type theory inference rules for a Structure Diagram provides a semantics for the heart diagram in accord with its informal semantics.

3 Algos type theory

Algos is a deduction system. The syntax is given by map and type term constructions with logical predicates for term equality, containment, and typing. The semantics is given by inference rules which have the form of a list of antecedents followed by a conclusion. The antecedents and conclusions are primitive predicates such as $a : A$ and $X = Y$ in which the arguments are term variables. The variables are substituted for in application of a rule. Map terms are composable. A composition construction is given as one of the map term constructions. For example the rule for the composition of maps is given as:

$$\frac{p : A \to B, q : B \to C}{q(p) : A \to C} \tag{10}$$

Both classes and properties are represented as types; classes as subtypes of an atomic type, $Thing$ and properties as subtypes of the protuct type $(Thing, Thing)$. Properties have both a domain and a range type. A property $p : (A, B)$ is a subtype of the product (A, B) which is a subtype of $(Thing, Thing)$. The notation $a : A$ is used for an instance a of a type A, and the notation $< a, b >: p$ is used for a pair which is an instance of a property p.

Type theory contains a truth value type, Ω. Map terms whose value type is Ω are called formulae. The formulae represent a higher order logic. The DL class constructions of union, intersection, as well as existential and universal quantification are definable as types using the power and abstraction type constructions. Similarly DL property constructions are definable.

Classes The axiom sets used for representing Description Logics have a type atom $Thing$. Subtypes of $Thing$ are called classes. Type theory provides a correspondence between subtypes of $Thing$ and formulas of the form $p : Thing \to \Omega$ and a characterization of subtypes of $Thing$ as abstraction types. An abstraction type is one defined by a formula $p : X \to \Omega$ using the rule:

$$\frac{p : X \to \Omega}{\{x : p(x) = true\}} \tag{11}$$

Any class A, i.e., subtype of $Thing$ has a characteristic map $char_A : Thing \to \Omega$ with

$$A = \{x : Char_A(x) = true\} \tag{12}$$

Further if A and B are classes then $A \sqsubseteq B$ is equivalent to $A \cap B = A$. The union and intersection of classes exist and their characteristic functions satisfy $char_{A \cap B} = char_A \cap char_B$.

Properties A property is a subtype of the product type (A, B) where A and B are subtypes of $Thing$. Properties are in one-one correspondence with characteristic maps whose domain is $(Thing, Thing)$. Property composition is defined

and has the well-formedness rule:

$$\frac{p : (A, B), q : (B, C)}{q.p : (A, C)} \tag{13}$$

Note that property composition is given in reverse form from map composition.

DL constructions The DL existential and universal quantification types are defined for abstraction types. For types X and Y an abstraction type B of X and an abstraction subtype R of the product $(x : X, y : Y)$ with respective characteristic maps b and r, the universally quantified type and the domain and range types are defined as

$$\forall R.B = \{x : \forall y.r(x, y) \Rightarrow b(y) = true\} \tag{14}$$

$$Dom(R) = \{x : \forall y.r(x) = true\} \tag{15}$$

$$Range(R) = \{y : \forall x.r(x, y) = true\} \tag{16}$$

and existentially quantified type is defined as

$$\exists R.B = \{x : \exists y.\epsilon(y, r^*(x)) \wedge b(y) = true\}. \tag{17}$$

In (17) ϵ is an atomic map of type theory with $\epsilon : (Y, Power(Y)) \to \Omega$ and $r^* : X \to Power(Y)$ is the type theory construction for the non-deterministic map corresponding to the property r. In Algos a functional property $p : (A, B)$ may be replaced by a Skolem map which is give by:

$$\frac{p : (A, B), Func(p)}{p^* : A \to B}, \tag{18}$$

with the rule

$$\frac{< a, b >: p}{< a, p^*(a) >: p)} \tag{19}$$

General declarations such as $p : Part(A, B)[k]$ can be defined within Algos. A functional property declaration $p : (A, B), Func(p)$ is equivalent to

$$A \sqsubseteq \exists p[1].B \tag{20}$$

In proofs functional properties will be replaced by their Skolem function.

Equalizers Algos has an equalizer type construction. Given a type A and two functional properties p and q with the same domain and range, the equalizer defines a subtype

$$A\{p, q\} = \{x : x.p^* = x.q^*\}. \tag{21}$$

The term $x.p$ is the relational composition. The equalizer is the subclass of A whose values under the maps p^* and q^* coincide. The rule is

$$\frac{a : A\{p, q\}}{a.p^* = a.q^*} \tag{22}$$

The converse rule also holds.

Semantics of Structure Diagrams Syntactically a structure diagram is a restricted Algos axiom set; the semantics is given by the Algos inference rules. The model theory of a structure diagram is the same as for any Algos axiom set. A model consists of a domain in which the individuals, classes, and properties are interpreted in the standard way and in which all of the axioms are satisfied.

Theorem 4. *All of the minimal models of a Structure Diagram are isomorphic.*

An instance of the root may be added to the type theory generated by the Structure Diagram axiom set. For any instance of the root class a finite set is obtained by iterating the application of the part properties in these paths. The orthogonality axiom for parts with the same range ensure that each instance path is unique. All of the axioms are satisfied by the resulting part decomposition. Any minimal model has this graph structure.

Decidability of Satisfaction for Structure Diagram Consistency Verification of decidability for satisfaction of a Structure Diagram results from the fact that the internal formulas corresponding to a Structure Diagram are equivalent to monadic Ackermann forumlae which are known to be decidable [1]. For simplicity the same notation will be used for a class and its characteristic map. In general a typed property declaration $p : (A, B)$ corresponds to the formula

$$\forall x \forall y . p(x, y) \Rightarrow A(x) \wedge B(y). \tag{23}$$

However, the properties in a Structure Diagram are all functional and in type theory these properties can be replaced by a Skolem map. A property assertion of the form $p : (A, B)[1]$ is equivalent to

$$\forall x . A(x) \Rightarrow B(p^*(x)). \tag{24}$$

where p^* is the Skolemization map corresponding to the property p. The application of p^* to x is written as $x.p^*$. All part property declarations are assumed to be functional. A part path declaration corresponds to a formula of the form

$$\forall x A(x) \Rightarrow A1(x.p1) \wedge \ldots \wedge An(x.p1 \ldots . pn). \tag{25}$$

A connection assertion has the form $A \sqsubseteq A\{\pi 1, \pi 2.c\}$ where the πi are part properties, as a connection is defined only for classes that occur as domain of ranges of part properties. For each of the finite number of part paths a monadic predicate $eq_{\pi 1, \pi 2, c}(x)$ is defined enabling translation:

$$\forall x . A(x) \Rightarrow eq_{\pi 1, \pi 2, c}(x) \tag{26}$$

Each disjointness assertion $A \perp B$ is equivalent to

$$\forall x . A(x) \Rightarrow not\, eq_{p,q}(x) \tag{27}$$

and each orthogonality assertion $p \perp q$ is equivalent to

$$\forall x A(a) \Rightarrow p^*(x) \neq q^*(x). \tag{28}$$

These formulas and the structure diagram assertions as represented within Algos are equivalent in that the internal formulas are provably equal to *true* if and only if the external type relations are provable within the first order theory.

4 Conclusion

A Structure Diagram abstracts the properties of diagrams used to describe the properties of a structure such as the human heart. The embedding of a Structure Diagram in type theory extends and generalizes the embedding of class diagrams into DL. These diagrams are entirely within type theory and have both an inference and a model theoretic semantics. Type theory is sufficiently rich that other descriptive properties of biomedical structures can be described. The technique of exploiting the correspondence between types and internal formulas can be extended to cover additional constructions which are useful for a DL. The arguments given here will work in other type theories but are more direct in Algos [4]. Algos is similar to the type theory in [3] based on equality. The major difference between Algos and that type theory is that Algos is closer to the original axioms for a topos. The differences between these alternatives are grounded in the quest to develop a language to be used in a mechanized computational context as opposed to just being talked about.

References

1. Ackermann, W., Solvable Cases of the Decision Problem. Studies in Logic and the Foundations of Mathematics. North-Holland, 1954.
2. Artale, A., Franconi, E., Guarino, N., Pazzi, L., Part-Whole Relations in Object-Centered Systems: An Overview, Data and Knowledge Engineering 20, North-Holland, Elsevier, 1996.
3. Lambek, J., Scott, P. J., Introduction to higher-order categorical logic, Cambridge University Press, 1986.
4. Graves, H., Bijan, Y., Using formal methods with SysML in aerospace design and engineering, Annals of Mathematics and Artificial Intelligence, Springer 2011.
5. OWL 2 Web Ontology Language, W3C Working Draft 11 June 2009.
6. Baader, F., D Calvanese, McGuinness, D., Nardi, D., The description logic handbook, Cambridge University Press, 2007.
7. Dumontier, M., Describing chemical functional groups in OWL-DL for the classification of chemical compounds. OWLED, 2007.
8. Hastings, J., Dumontier, M., Hull, D., Horridge, M,. Representing chemicals using OWL, description graphs and rules, 2010.
9. Horrocks, I. Kutz, O., and Sattler, U., The Even More Irresistible SROIQ, in Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57-67, American Association of Artificial Intelligence Press, 2006.
10. 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.
11. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., The description logic handbook, Cambridge University Press, 2007.
12. OMG Systems Modeling Language (OMG SysML), V1.2, 2010.