Constructions for Modeling Product Structure

Henson Graves
Algos Associates
Fort Worth Texas, USA
henson.graves@hotmail.com

Abstract. *This paper identifies constructions needed for modeling product structure, shows which ones can be represented in OWL2 and suggests extensions for those that do not have OWL2 representations. A simplified mobile robot specification is formalized as a Knowledge Base (KB) in an extended logic. A KB is constructed from a signature of types (classes), typed properties, and typed variables and operators. Modeling product structure requires part decompositions, connections between parts, data valued properties, typed operations with variables, and constraints between property values. Data valued properties represent observable properties assumed or measured regarding a product. Operations with variables are used to define constraint properties such as fact that the total product weight is the some of weights of product components. Constructors, which take arguments from the signature and have properties as values, are used to specify families of properties such as part properties. These constructions are illustrated for the mobile robot. Operators and variables are represented as type, property, and operations within type theory.*

Keywords: Description Logic, Ontology, OWL, Product Model, SysML, Type Theory, UML.

1 Introduction

This paper uses a mobile robot design specification example to identify constructions needed for modeling product structure. Some, but not all of these constructions can be represented in OWL2 [13]. The product modeling example suggests possible OWL2 extensions for product modeling. This is a continuation of earlier work [6,7,8,9]. The mobile robot specification is represented using a language of classes (types), properties, with typed operations, and dependent type term constructors which are used to specify families of properties such as part properties. The language is that of intuitionisitc type theory [14]. A type theory is constructed from a multi-sorted signature of types (classes), properties, and operations using typing and equality axioms. While in general type theories are undecidable, the form of axioms used for product modeling is restricted to using OWL2 constructions and operation term equation reasoning. The decidability of equality has been proven for a variety of intuitionistic type theories [1] based on proving terms have a unique irreducible form. These results together with the OWL2 decidability results can possibly be combined to prove decidability for suitably restricted type theories that use the OWL2

constructions. This paper hopes to make clear with an example product specification that type theory constructions are a natural fit for product modeling and provide a formal semantics for SysML structural product models.

1.1 A design specification as a knowledge base

The informal concept of a product is distinguished by having been built or at least can be described by a design specification. Design specifications occur at different levels of abstraction. Some design specifications may not have sufficient detail to determine a parts decomposition. The example specification described here not only has a parts decomposition, but all parts decompositions are structurally the same. The structural part of the specification is its parts decomposition and the specified connections between parts. Examples of the kind of analysis to be performed on a design specification is to determine if there are design instances, do all design instances have the same structure, and does replacement of a part with another part with a different type still constitute an implementation.

A mobile robot specification is used to illustrate structural product modeling constructions. The specification is presented as a collection of SysML [4] diagrams which are graphical views of a SysML model. Each diagram represents aspects of the specification and may hide part of the specification.

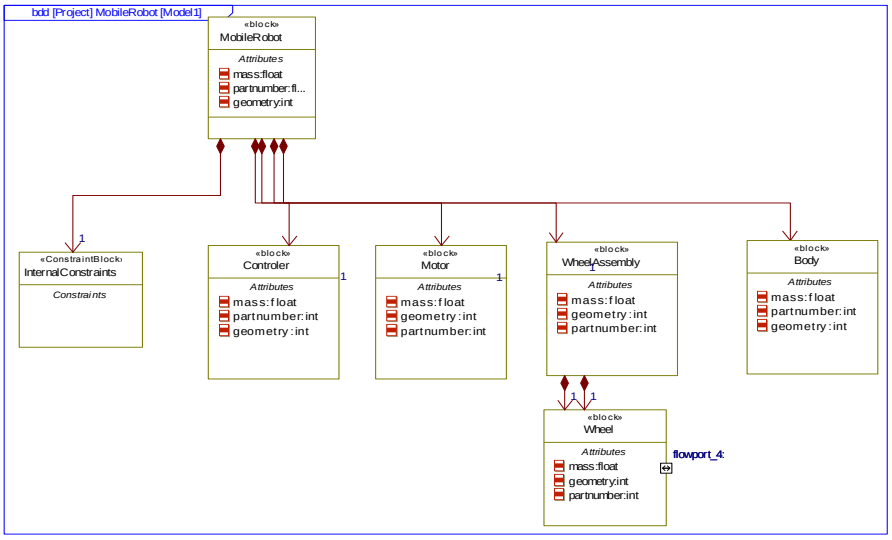

Fig. 1 This diagram uses directed arrows to connect blocks. Each arrow has a source and target with a numeric quantification. The arrows represent parts relationships of the specification. All of the blocks except InternalConstraints are physical part types. InternalConstraints represents the interrelationships between data properties of a mobile robot and its components. The blocks display data valued properties.

A design specification can be represented as a knowledge base (KB) in a language with types (classes), binary properties, and operations. The collection of named types,

properties, and operations in the model is referred to as the signature of the model. The blocks in figure 1 represent types. For example, a vehicle specification has types for MobileRobot, Body, Engine, WheelAssembly, and other components. Data properties are used for product attributes, and object properties are used for parts decompositions and connections between parts. The discussion of the robot example will use a linear syntax that slightly extends SysML syntax.

Properties, as they typically occur in product modeling have domain and range types. These are called typed properties and the notation P:(A,B) is used to express that P is a property with domain type A and range type B. Each typed property can be represented within OWL2 using axioms for domain and range. In Figure 1 each closed diamond headed arrow is a typed property. The names are not shown in this diagram, however, the arrow from MobileRobot to WheelAssembly is named itsWheelAssembly domain has domain MobileRobot and Range WheelAsembly. For a property P with P:(A,B), then the notation (P some B) and (P exactly k) are are used for restriction subtypes of A. The notation a:A is used to express that a is an instance of A, and (a,b):P to express that the pair (a,b) s an instance of P. If a pair of individuals (a,b):itsWheelAssembly then a: MobileRobot and b:WheelAssembly.

The collection of solid diamond headed properties is a family of properties which are irreflexive, anti-symmetric, acyclic, and form a connected graph. For example, the family contains no property P with P:(A,A) and if P:(A,B) then one does not have P(B,A). The types in the SysML model are assumed to be disjoint. The class MobileRobot is a top element with respect to the parts.

Figure 2 contains a more detailed description of the parts and connection specification for the mobile robot. WheelAssembly is specified to have exactly 2 wheels and that the Motor drives wheels.

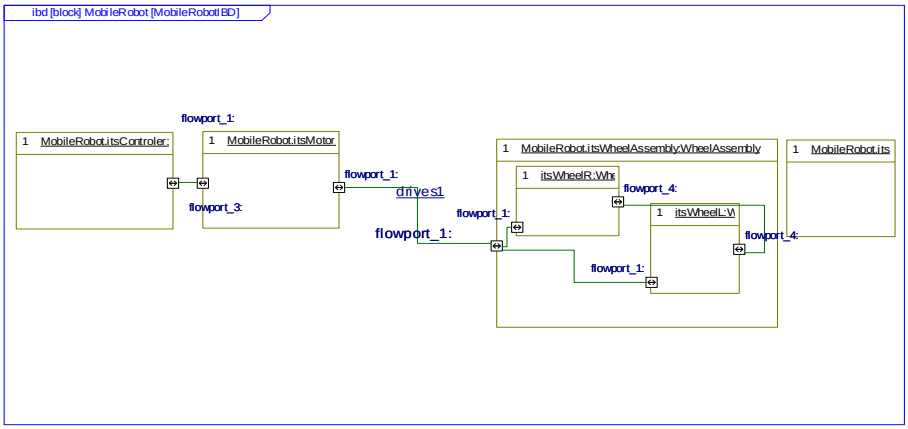

Fig. 2 This diagram uses nested containment of blocks and directed lines to define parts and connection properties. A block with the title itsWheelR:Wheel denotes the property, itsWheel with domain type WheelAssembly and range type Wheel.

Structurally, a product is an assemblage of parts where each part has an identification number which identifies the specific instance of the part of the vehicle. A product implementation is a composite of individual parts (parts decomposition) and interrelationships between the parts. The common way to express the fact that any

v:MobileRobot has an engine, body, wheel assembly, etc. is to use a computer science "dot" notation

v.itsEngine

for the engine instance dependent on an instance of MobileRobot. The parts decomposition for an instance v:MobileRobot is obtained by constructing a dependent enumeration type:

Parts(v) = {v, v.itsEngine, v.itsWheelAssembly, … }

An individual instance of MobileRobot has a parts decomposition which determines the type of specific parts and the number of parts. The parts decomposition is a tree and the cardinality of the enumeration set is the number of parts in an implementation. Since each part has a typing, the number of distinct types used by the decomposition can be determined as can the number of occurrences of a given type.

On figure 2 the ports w1.port_1 and w2.port_2 are parts of Wheel. Properties are used to represent connections between parts. The motor port is connected to the Wheel ports. For example, the two individuals, v1.itsMotor.port1 and v2.itsWheelAssembly.port1 may be connected by a drives property which implies

(itsMotor.port_1,itsWheelAssembly.port_1):drives

A connection structure is a family of properties defined for some subset of the type signature. A connection structure is a family of properties which all have the typing :(A,A) for a port type A. For example, drives(Motor.port_1, Wheel.port_1). A subsystem can be defined by what components are reachable from a given node such as a battery through a specified connection property.

The Robot KB in addition to its structural properties uses operations and variables to express relationships between property values, such as the relationship between the total weight of the robot and the weight of its parts. The types in the mobile robot specification have attributes such as mass which is an operator variable. Both weight and 3D geometry are represented as typed variables. Variables and operations are typed and are always declared as being a property of an individual. An individual may have multiple variables. An operator symbol f has a type signature f(A1,…,An):B where A1,…,An, B are types. An operator symbol a():B is written a:B.

Each mobile robot part has a mass attribute and they are all distinct. Constructors with names can be declared which have type arguments and values. For example, the expression

mass = attribute_1(Motor): Var(Int)

declares that

var attribute_1(Motor): Int

Var(Int) is the type of integer variables, and attribute_1 is a constructor defined for the type Motor. For m:Motor we use the notation

mass(m) = m.mass

and the notation

var m.mass :Int

The Interaction block in Figure 1 represents interactions between value properties of the robot and its components. The interactions between attributes on the product have been separated out into a separate block which can be potentially reused. In Figure 3 constraints on the relationship between weight and geometries is expressed with a parametric diagram. This construction allows the relationships to be defined independently of the product properties. Interaction constraints are represented using variables and operations where the variables can be bound to attributes.

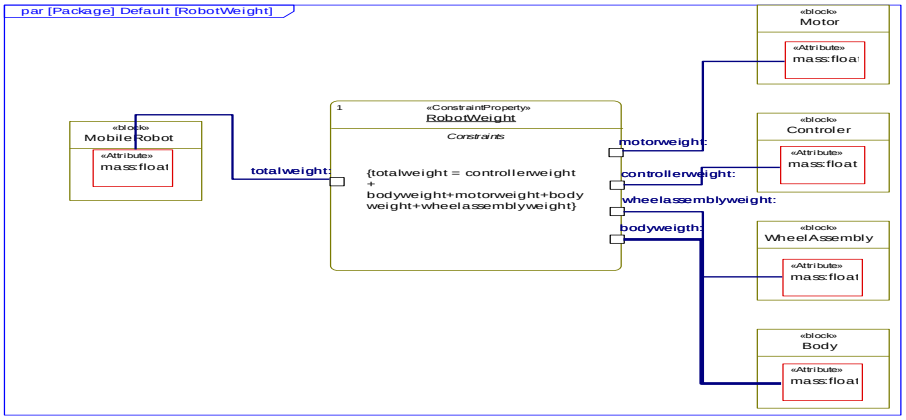

Fig. 3 The parametric diagram binds the attributes of the robot and its parts to constraint parameters in RobotWeight. The constraint property contains an equality relationship between the weight attributes.

The parametric diagram serves as an axiom that the total weight of the mobile robot is constrained to be the sum of the weight of its parts. Provided we know that any design instance has the same parts structure the total weight axiom would be used to show that any instance of the design satisfies a weight constraint by constructing a parts decomposition and summing the weights.

An implementation of the robot specification is a collection of type and property instances which satisfy the graph structure defined in the specification. The most rigid notion of product equivalence is that two products are equivalent when they have the same product decomposition and each of the corresponding parts have the same typing. However, often one wants to replace a part with an alternative. The alternative may be a later revision of the original part. Replacement is generally acceptable so long as specific parts properties are preserved and total product properties such as total weight, or power consumption are preserved. Impact analysis is an analysis of changes of properties of a design when components are replaced with other components.

Since the Mobile Robot KB does not use any subclass axioms between the basic types and all of the parts properties have an exact numeric restriction the informal interpretation is that the parts decomposition is unique; any two individuals of root type, i.e., v1 and v1 in MobileRobot have the same parts decomposition and connection graph structure. Analysis of the graph structure can answer questions such as how many parts are required to implement the design. For an arbitrary property the

collection of individuals reachable from a given individual can be determined. Note that a mobile robot may have additional parts which are not part of the specification, e.g., "after market" parts. Note also that there may be multiple parts concepts. For example a "reference-part" is a part which is a part used by a product, but whose complete specification is not part of the product specification. For example, a robot may use batteries, but they are not part of the product specification.

This example presents an incomplete example structural design specified specification for a mobile robot and describes the informal semantics. The next section will describe how this specification can be represented as an axiom set within a formal logical system.

3 Formalization of Types, properties, and operations

The structural product modeling constructions used in the mobile robot example include classes, properties, and typed operations, as well as enumeration types and data types are represented in intuitionistic type theory [2]; the semantics of type theory accords well with the informal semantics described above. An intuitionistic type theory is generated from axioms expressed in the language of a multi-sorted signature of types, properties, and operations. For product modeling a type constant, Thing, is used and the property symbols are restricted to have binary arity. The typed properties are closed under composition, inverse, restriction, and union, provided the typing conditions are met. The types are closed under intersection, union, finite enumeration types, existential restriction, and have constants for top and bottom types. Intersection allows the subtype relation to be defined for types and properties. An intuitionistic type theory contains a full higher order formula language with logical connectives is a consequence of having a truth value type. The formulas are represented as operations with truth value type. The presence of a type for truth values allows a language of logic to be represented as terms within the language.

The types generated from the signature by the OWL2 type constructions are closed under the formation of product types. Product types are not used in the OWL2 type constructions. Operator symbols in the signature are declared with types, e.g., var x:C and f(x1:C1,…,xn:Cn):D. Operator terms are constructed from operations and variables using composition, and substitution and are typed. The notation f(g1,…gn) is used for composition and t[t1/x1] for the substitution of a term t1 for a variable x1 in an operator term t. Type, Property, and Operator equality are used. The language of OWL2 is contained within type theory with the addition of the type constant Thing. OWL individuals are zero-ary operations, classes are subtypes of Thing, and properties are subtypes of a product type (Thing,Thing).

Type theory uses inference rules to construct new judgments from judgments (axioms) assumed to be true about the domain of discourse. The judgments are in the form of term equations and type statements. The rules are presented in a numerator/denominator form the antecedents above a line and conclusions below the line, as is done for natural deduction calculi [15]. Equality inference rules obey usual term substitution rules. Some of the inference rules may be derived from other rules. Type and property constructions with their rules allow the introduction of subtype and

subproperty relations in addition to equality. An example of an inference rule for typed properties is

P:(A,B), (a,b):P
---
a:A, b:B

The type inference rule for the inverse is

P:(A,B)

P* : (B,A)

An Abstract Block Diagram (ABD) is a term language generated from a signature of individual, class, property symbols, using type constructors from OWL2, and operator constructions. Given a type signature and class, operator, and property constructions a Type Theory Knowledge Base (KB) is a collection axioms of the form of Term = Term for the valid type, property, and operation term constructions from a signature. As type and property constructions are closed under intersection, subtype and subproperty relations can be defined and will be used.

For any KB, the theory is the closure of the axioms under the inference rules. The reason for calling such a type theory an ABD is that such a type theory can be used as a logical formalism for SysML. A SysML Block Diagram model translates directly into an ABD KB. By restricting the form of axioms used to generate a theory to be OWL constructions with typed operator declarations where the types are generated from the OWL2 signature and type constructions preserves OWL2 computational properties.

In the ABD constructed from the Mobile Robot signature MobileRobot is defined as a recursively using the existential restriction type construction.

MobleRobot = (itsEngine exactly 1) and (itsWheelAssembly exactly 1) and (itsWheelL exactly 1) and ….

The semantics is given by term constructor inference rules described below.

3.1 Term constructors

An ABD theory following SysML has a term constructor which takes as an argument a property P:(A,B) and returns a term of type B which is dependent on a term of type A. If P:(A,B) and a: A, then a.P is a term with a.P:B for which P(a, a.P). If a, b are distinct individuals of type A then the qualified terms a.P and b.P are distinct.

The Dot constructor for (P exactly k) introduces for an individual a with a:A individuals a.P_1,…,a.P_k. For any property P:(A,B), the Dot constructor uses the notation a.P. The Dot constructor uses the inference rule

P : (A,B), A subtype (P exactly 1), a:A
---

a.P:B, (a,a.P): P

For example, the semantics of itsBody:(MobileRobot, Body) with the cardinality restriction that a MobileRobot only has one body is given by an application of the inference rule

v:MobileRobot, MobileRobot subtype (itsBody exactly 1)
---
v.itsBody:Body, (v, v.itsBody):itsBody

The dot constructor is extended to properties whose range type is an operator type. For example,

v:MobileRobot, MobileRobot subtype (itsOperator some (x1:C1,…,xn:Cn):D)
---
v.itsOperator: (x1:C1,…,xn:Cn):D, (v, v.itsOperator):itsOperator

and

v:MobileRobot, MobileRobot subtype (itsVar some A)
---
v.itsVar:A, (v, v.itsVar):itsVar

3.2 Dependent types

The product modeling constructions that used enable the introduction of terms which are dependent on other terms. The mobile robot specification makes use of constructors to specify part properties, connection properties, as well as properties used to associate ports, and operations and variables with components. For example, these constructions enable associating operations and parts with instances of a type. A parts decomposition

part{v}

is an enumeration type dependent on a variable v of type MobileRobot.

3.3 Model Theory

During the development of a specification such as the mobile robot the attributes (variables) such as the mass variables may be unbound. The semantics for a design specification with free (unbound) variables is an interpretation defined on the domain typing which satisfies all of the constraint relationships. An interpretation of the Robot KB is a mapping which maps types to domains and an operator with a type declaration f(A,…,An):B to a map from the product of domains (A1,…An) to B. The map is extended to operator terms formed by substitution of terms for variables. The interpretation is a model of the KB, provided the constraints are satisfied in the interpretation. For the Mobile Robot KB a valid implementation includes a parts

decomposition whose attributes satisfy the interaction constraints. A design solution for the robot specification is a parts decomposition with all of the variables of each part bound to values and the constraint relationships satisfied. While finding a solution for a design specification which satisfies arbitrary constraints may not always be possible, a solution for the mobile robot is obtained by specifying a parts decomposition, substituting values for the variables which satisfy the constraints.

Interpretations and models can be defined not only within set theory, but within any topos [3]. The semantics of the constructions are assumed to be the OWL2 semantics for the constructions which occur in OWL2; semantics for other constructions will be defined in terms of KB models. There are soundness and completeness results for the first order intuitionistic natural deduction theory of the type theory with respect to topos models [3]. The inference rules can be translated into formula in the multi-sorted first order theory generated by the signature. The first order version of the MobileRobot mobile robot theory contains as a consequence the existential statement:

For any v:MobleRobot then (there exists y.(v,y):hasWheelAssembly) and (there exists z. (y.z):LeftWheel) and …)

A formula such as the constraint formula

(for any v:Vehicle)(totalweight(v) = sum(weight(parts(v))

which is used conclude that the mobile robot design instances satisfy the total weight constraint can also be represented within the internal logic of the type theory where the operations are truth valued and logical equality is replaced by an equality operation whose value type is the truth value type. By using inference rules an ABD can express axioms which are not OWL2 axioms and consequences can be derived that are not OWL2 consequences. For example, two parts decompositions of MobileRobot can be made disjoint by adding an axiom that ensures that if an individual is in two decompositions then the instances of the root are equal.

4 Conclusions

The constructions used in the mobile robot specification are common in product modeling and need to have a formal logical semantics. These constructions make use of OWL2 class and property constructions; however, they make use of additional constructions to introduce context dependent terms, and to introduce operations and variabls. The constructions correspond closely to those found in SysML. As SysML is widely used in industry to develop and analyze complex product designs [3], it seems natural to consider these constructions either as additions to OWL2, or to view OWL2 as embedded in a type theory. The axioms of the mobile robot KB are implicit in the SysML model. Conversely the graphical syntax of SysML provides a good way to specify property associations for OWL2 KBs.

The reasoning needed for product modeling includes the ability to prove a property for a generic product instance and conclude that the property holds for any instance. Replacing a product part with another part in the simplest case requires validating

only that data valued property constraints are not violated. However, using a different part may require changes in connections which require verification that the modified product structure is still a subtype of MobileRobot. Special purpose graph theoretic algorithms can be used for the KB signature analysis. Impact analysis which assesses the impact property changes to a design when components are replaced with other components. A question such as does replacing a part with another part which may have another typing can be accommodated by introducing additional variables and operations and constraints for the robot which express total constraints such as maximum weight.

The full mobile robot specification has additional information not covered here. The specification has a state chart behavior component and the ports are typed as continuous. The full robot specification may be compiled and instances created, which when placed in an operating environment instance can be operated by an operator with a remote controller. These constructions needed to be added to any formalism for product modeling.

References

[1] Abel, A., Coquand, T., Dybjer, P., *Normalization by evaluation for Martin-Lof type theory with typed equality judgements*, Logic in Computer Science, 2007.

[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] Bell, J., *The Development of Categorical Logic*, Handbook of Philosophical Logic, 2005

[4] Friedenthal, S., Moore, A., and Steiner. F., OMG Systems Modeling Language (OMG SysML™) Tutorial, INCOSE Intl. Symp, 2006.

[5] Graves, H., Guest, S., Vermette, J., and Bijan, Y., *Air Vehicle Model-Based Design and Simulation Pilot*, Spring 2009 Simulation Interoperability Workshop.

[6] Graves, H., Horrocks, I., *Application of OWL 1.1 to Systems Engineering*, OWL Experiences and Directions April Workshop, 2008.

[7] Graves, H., *Representing Product Designs Using a Description Graph Extension to OWL 2*. OWL Experiences and Directions October Workshop, 2008.

[8] Graves, H., Leal, D., *Using OWL 2 For Product Modeling*, Proceedings of 11th NASA-ESA Workshop on Product Data Exchange, 2009.

[9] Graves, H. *Integrating SysML and OWL*. OWL Experiences and Directions October Workshop, 2009.

[10] Martin-Lof, P., constructive mathematics and computer programming. *Logic, Methodology and Philosophy of Science*, 1982.

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

[12] OMG Formal Ontology Definition Metamodel

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

[14] Pitts, A., *Notes on Categorical Logic*, Cambridge University, 1989.

[15] Troelstra, A., Schwichtenberg, H. *Basic proof theory*, books.google.com, 2000