Using Reference Domain Ontologies to Define the Real-World Semantics of Domain-Specific Languages

Victorio A. de Carvalho[1,2], João Paulo A. Almeida[1], and Giancarlo Guizzardi[1]

[1]Ontology & Conceptual Modeling Research Group (NEMO)
Federal University of Espírito Santo (UFES), Vitória, ES, Brazil
[2]Research Group in Applied Informatics, Informatics Department,
Federal Institute of Espírito Santo (IFES), Colatina, ES, Brazil
victorio@ifes.edu.br; jpalmeida@ieee.org; gguizzardi@inf.ufes.br

Abstract. This paper proposes a principled approach to the definition of real-world semantics for declarative domain-specific languages. The approach is based on: (i) the explicit representation of the admissible states of the world through a reference domain ontology (which serves as semantic foundation for the domain-specific language), (ii) a representation of the valid expressions of a domain-specific language (to determine the abstract syntax of the language), and (iii) the rigorous definition of the relation between the abstract syntax and the reference domain ontology (to define the real-world semantics of the language). These three elements of the approach are axiomatized in three corresponding logic theories, enabling a systematic treatment of real-world semantics, including formal tooling to support language design and assessment.

Keywords: Ontology, Metamodel, Domain-Specific Language, Semantics.

1 Introduction

Conceptual modeling is generally considered a fundamental activity in information systems engineering [1], and comprises the use of diagrammatic languages for communication, understanding and problem solving regarding a universe of discourse. The effectiveness of a conceptual modeling language to support the aforementioned tasks is strongly related to the language's *domain appropriateness*, i.e., to the language's ability to express the relevant characteristics of the domain at hand, as discussed by a number of authors [2, 3],

A language designer must, therefore, understand the phenomena (or domain) that should be covered by the language and propose symbolic structures that will empower prospective language users to efficiently carry out certain tasks concerning the represented phenomena.

This requires the design of a language with some form of 'correspondence between its constructs and things in the external world' [4]. We call such a correspondence *real-world semantics*, following [5]. Consider for example a domain-specific language to describe genealogical relations. A real-world semantics for this language would provide meaning for the various language constructs in terms of parenthood or

ancestry relations between persons, thereby enabling its expressions (or models) to be used as a vehicle to talk about parenthood or ancestry between persons of interest.

Although essential to language design and semantic interoperability tasks, the *real-world semantics* is often defined only informally for modeling languages. As a consequence, no systematic treatment of real-world semantics is possible, and the designer must face semantic issues with little methodological support. In this paper, we address this gap by proposing a principled approach to real-world semantics definition for declarative domain-specific languages. This approach is based on: (i) the explicit representation of the admissible states of the world through a reference domain ontology (which serves as semantic foundation for the domain-specific language), (ii) a representation of the valid expressions of a domain-specific language (to determine the abstract syntax of the language), and (iii) the rigorous definition of the relation between the abstract syntax and the reference domain ontology (to define the real-world semantics of the language). These three elements are axiomatized in three corresponding logic theories, enabling a systematic treatment of real-world semantics, including formal tooling to support language design and assessment.

From the methodological perspective, the approach promotes the separation of concerns enabling designers to handle semantic issues separately from other language design concerns. By defining the semantics of a language in terms of a reference domain ontology, the language designers explicitly account for the language's ability to represent domain features truthfully [6].

Although some of us have defended in [6] that the abstract syntax of a language should ideally be isomorphic to an ontology underlying the language, this would only apply to a particular kind of (ideal) language intended to represent complete knowledge about a domain. This is not always feasible or desirable due to pragmatic and language design issues. Thus, in this paper we relax the stringent isomorphism requirement, supporting thus a more flexible relation between language metamodels and reference ontologies.

The remainder of the paper is structured as follows: section 2 discusses the notion of reference domain ontology contrasting it with the notion of language metamodels; section 3 discusses our approach to define real-world semantics for domain-specific languages; section 4 introduces our running example, defining a reference domain ontology for genealogy; section 5 defines the syntax of a domain-specific language to capture genealogy trees, specifying a semantics for this DSL based on the ontology described earlier; section 6 shows the use of formal tools to analyze the language in the light of some properties defined in our approach; section 7 presents some additional reflections on ontologies, metamodels and the real-world semantics definition; section 8 discusses related work and section 9 presents conclusions and future work.

2 Reference Domain Ontologies and Language Metamodels

An ontology can be defined as a set of entities acknowledged by a theory or system of thought [7]. In the original definition of formal ontology in philosophy, the "set of entities" in question refer to domain-independent categories such as object, quality,

relation, event, type, situation, among others. In a modern jargon, these are termed *Foundational Ontologies* [8, 9]. In contrast, in computer science (in a tradition which can be traced back to Hayes' seminal paper [10]), ontologies are specifications, i.e., particular engineering artifacts meant to represent "sets of entities" of a different nature, namely, those entities whose existence is countenanced by a *conceptualization* of a particular area of application or knowledge (e.g., law, cell biology, chemistry, software engineering). These are termed *domain ontologies* [8].

In the past 15 years, an increasingly large number of domain ontologies have been constructed in a number of domains. As discussed in depth in [6], domain ontologies can play an important role in the evaluation, (re)design and interoperability of domain-specific conceptual modeling languages informing the axiomatisation for their abstract syntax and formal semantics and guiding the design of pragmatically more efficient systems of visual concrete syntax. However, in order to play this role, these ontologies must be constructed as *reference ontologies*, i.e., they must be constructed with the sole objective of making the best possible description of a certain domain or portion of reality, capturing a shared conceptualization of that specific domain. In particular, these ontologies should be able to formally characterize the exact world states which are deemed admissible by that conceptualization[1]. Furthermore, as the author demonstrates in [6], in order to systematically achieve this desiderata, domain ontologies should be constructed with the support of a proper foundational ontology. In this sense, a foundational ontology, determines all possible world states according to fundamental ontological commitments (relating to notions of space, time, object, event, action, etc.). A domain ontology aligned with a foundational ontology, defines a subset of these states selecting only those admissible in a specific domain, as shown in Fig. 1 (for a foundational ontology and two overlapping domain ontologies).

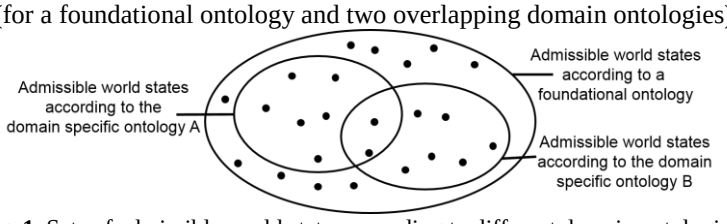

Fig. 1. Sets of admissible world states according to different domain ontologies

Language metamodels are often captured in frame-based languages (such as UML, MOF, Ecore) and enriched with additional constraints in order to define syntactic rules of the language that cannot be captured directly in the frame-based language. Examples of language used to define such syntactic constraints include OCL and first-order logic. In the remainder of this paper, we use the term "abstract syntax" referring to the syntactic rules implicit in the metamodel as well as those additional constraints.

Fig. 2 illustrates the role of a language's abstract syntax. It defines the set of syntactically valid models of a language, as a subset of all possible models that can be instantiated from metamodels expressible in a given language (such as, e.g., Ecore).

[1] A further analysis of the relation between ontologies and conceptualizations is outside the scope of this paper. We refer the reader to [6], which accounts for this relation.

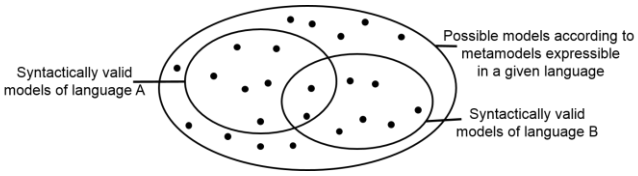

Fig. 2. Sets of language expressions according to different metamodels

Note that Fig. 1 and Fig. 2 are analogous with the important difference that, while Fig. 1 concerns states of the world deemed admissible by reference domain ontologies, Fig. 2 concerns symbolic expressions deemed syntactically valid in domain-specific languages. In our approach, we represent both reference domain ontologies and a language's abstract syntax through logic theories. Although represented in a similar fashion, the logic theories have clear and distinct roles. A reference domain ontology should aim solely at describing phenomena of reality representing a certain conceptualization and is not influenced by language design issues. The logic theory that captures a reference ontology quantifies over real-world entities. In contrast, a metamodel (and additional constraints) should define a language syntax capable of meeting information demands about phenomena of reality for some specific task. Thus, it quantifies over symbolic expressions (instead of real-world entities).

3 An Approach for Real-World Semantics Definition

In order to provide real-world semantics for a domain-specific language, we relate the valid models of the language to corresponding world states that are deemed admissible by a reference domain ontology. Since both language syntax and reference domain ontologies are axiomatized into logic theories, that task can be accomplished with a third logic theory that quantifies over symbolic expressions and world states at the same time. The resulting relation is depicted schematically in Fig. 3.

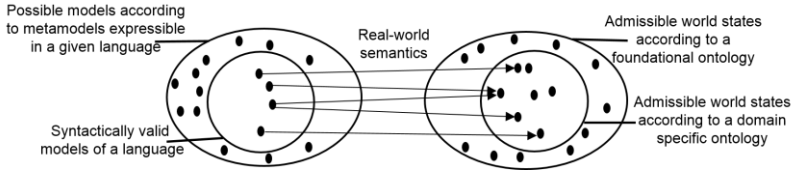

Fig. 3. Real-world semantics definition

By formalizing the correspondence between language models and world states, we can characterize an important property of a language: we say that a *language has a well-defined real-world semantics according to a reference ontology* iff each of its (syntactically-correct) models is about (at least one) admissible world according to the ontology. A language that fails to have this property is one that allows the definition of meaningless models, i.e., models with no correspondence to the phenomena they intend to represent. We consider it thus a minimum semantic requirement[2].

[2] The choice of reference domain ontology is clearly important in this process. This is discussed in section 7.

This approach also allows us to formally characterize an important class of syntactic constraints, those we call *semantically-motivated syntactic constraints.* These constraints have the purpose of reflecting real-world rules into a language abstract syntax. If a language admits models that are not about any admissible world state, it suggests that the language syntax may lack semantically-motivated syntactic constraints. In other words, if the set of semantically-motivated syntactic constraints is strong enough, the language can be said to have a well-defined real-world semantics. Conversely, if we suppose that a language has a well-defined real-world semantics, then all its semantically-motivated syntactic constraints should be entailed by the remainder of the unified theory, i.e. by the semantic mapping axioms in tandem with the ontology axioms. This means that the ontology axioms may be used to shape the definition of the language, in particular, helping in abstract syntax definition.

To illustrate the approach and its implications, we present an example to show how the three logic theories are combined to accomplish real-world semantic definition. We show the use of the Alloy formal method [11] to guarantee that the unified theory is consistent and to identify which syntactic constraints are semantically-motivated.

4 A Reference Domain Ontology in Genealogy

This section presents an ontology in the genealogy domain which will be used later to define the semantics of a domain-specific language. Our approach to present this ontology is to illustrate it with an OntoUML diagram [8], and then present the axioms that are not implied by this diagram.

OntoUML specializes the UML class diagram by differentiating various categories of classes according to taxonomy of types in the Unified Foundational Ontology (UFO) [8]. In an OntoUML diagram some ontological distinctions of UFO are represented as stereotypes. A class with a <<*kind*>> stereotype applies necessarily to its instances (e.g., instances of *Person* cannot cease to be so without ceasing to exist) and provides a uniform principle of identity for them. A class stereotyped as <<*kind*>> may be specialized in other rigid classes stereotyped as <<*subkind*>> (e.g., *Man* and *Woman*). A <<*role*>> is an anti-rigid concept that classifies instances through the relation properties the instances bear in the scope of a relational context. In this paper, we consider that this relational context can be a material relation or an event (e.g., a *Man* plays the role of *MaleProcreator* only in the scope of a *Conception* event, and does not cease to exist when it no longer plays that *role*). So, a class stereotyped with <<*role*>> classifies its instances dynamically. Finally, a <<*phase*>> is an anti-rigid concept that defines a partition of a <<*kind*>> depending on one or more of its intrinsic properties (e.g., a *Person* can be said to be in either of the two *phases: LivingPerson* or *DeceasedPerson).* Fig. 4 depicts the OntoUML diagram that illustrates the main concepts of the ontology.

The proposed domain reference ontology defines that the (biological) ancestry relationships are derived from *Conception* events. It is based on the stance that human beings are products of instantaneous *Conception* events, which occur when a human male sperm unites with a human female oocyte egg. Thus, each human being (*Person*)

is the product of a *Conception* event. For the sake of simplicity, we consider that, in the case of identical multiple siblings, several *Conceptions* occur at the same time boundary. In addition to the *Person* who plays the role of *Offspring* (the product of the *Conception)* two other *Persons* participate in a *Conception* event, namely: *(i)* a *Man,* playing the role of *MaleProcreator* and *(ii)* a *Woman* on the role of *FemaleProcreator.* We assume that the product of the *Conception* event is considered a *LivingPerson* (i.e., a fetus is a living person even before its birth). On the other hand, both *LivingPersons* and *DeceasedPersons* may participate in a *Conception* as procreators (i.e., the ontology considers the possibility of artificial insemination).

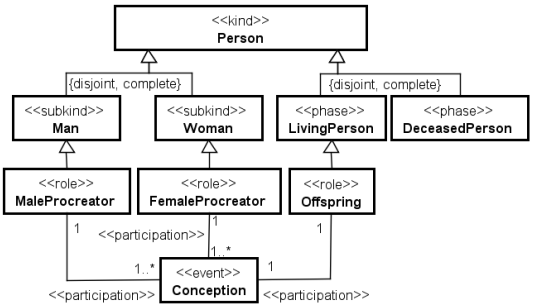

Fig. 4. A domain ontology about genealogy in OntoUML

We formalize the ontology as a theory in many-sorted first-order logic, quantifying over possible states of the world and entities that may exist (objects) or occur (events). Thus, we assume two disjoint sets of entities: a set W of worlds and a set U of entities that are typed by the classes present in the OntoUML diagram. In order to represent the dynamic of the change in world states, we use a predicate $next(w_1,w_2)$ which holds between a world w_1 and all the world states that follow it in time. We consider *next* represents an asymmetric, irreflexive, transitive and completely ordered relation (i.e. a strict total order relation) between worlds. We further assume: (i) A binary predicate for each *kind, subkind* and *phase* from the OntoUML diagram, namely, *Person(w,p), Man(w,p), Woman(w,p), LivingPerson(w,p)* and *DeceasedPerson(w,p)* (e.g. *Person*(*w,p)* holds if an entity *p* is an instance of *Person* and exists in a world *w*); (ii) A binary predicate representing the *Conception* event (e.g. *Conception(w,c)* holds if *c* is an occurrence of *Conception* that happens in the time boundary of a world *w*), and; (iii) A ternary predicate for each *Role* from the OntoUML diagram, namely, *isMaleProcreator(w,c,p), isFemaleProcreator(w,c,p)* and *isOffspring(w,c,p)* (e.g. *isOffspring* (*w,c,p)* holds if an entity *p* is an instance of *Person* and an entity *c* is an instance of *Conception*, the person *p* plays the role of *Offspring* in the context of the *Conception c* and both, *c* and *p,* exist in a world *w*).

For the sake of brevity, the axioms implied by the OntoUML diagram (classes' rigidity, lower and upper bound for cardinality constraints, specialization relations) are omitted. Those not expressed by the OntoUML diagram are presented in Table 1.

Axioms A1 and A2 determines that every *Person* must play the role of *Offspring* of one *Conception* event in which two other *Persons* play the role of *Procreators,* except for the case of the first *Persons* considered to exist (the "original" persons). We assume that these *Persons* come into existence in the same world. The origin of these

Persons is outside the scope of this ontology (as it is neutral with respect to accounts of the origin of humans such as biological evolution, theological creation).

Axioms A6 and A7 state that the *Man* and the *Woman* who play the roles of procreators in a *Conception* event are considered, respectively, the *father* and the *mother* of the person who plays the role of *Offspring* in such event. Thus, we can infer that every "non-original" *Person* has exactly one father and exactly one mother. Axiom A9 defines the ancestry relationship based on the parenthood relation (defined in A8). Thus, the ontology defined here precisely defines the ancestry relationships explaining the concepts of *parent* and *ancestor* in terms of the concept of *Conception* event.

Table 1. Axioms not implied by the OntoUML diagram

A1	A *Person* who plays the role of *Offspring* in a world does not exist in previous worlds.
	∀w:W, ∀p,c:U (isOffspring(w, c, p) → ¬∃w':W(next(w', w) ∧ Person(w', p)))
A2	If a *Person* exists in a world w and it does not exist in any world previous to w then this *Person* plays the role of *Offspring* in w or he/she is an "original" person(i.e., there are no persons in worlds previous to w).
	∀w,w':W, ∀p:U ((next(w', w) ∧ Person(w,p) ∧ ¬Person(w', p)) → (∃c:U (isOffspring(w, c, p)) ∨ ¬∃p':U (Person(w', p'))))
A3	Once a *Person* exists in a world w, it will exist in all worlds subsequent to w.
	∀w,w' :W, ∀p:U(((Person(w, p)∧ next(w, w')) → Person(w', p))
A4	A *Person* cannot simultaneously play the roles of *Procreator* and *Offspring.*
	∀w:W, ∀c,p:U (isOffspring(w, c, p) → ¬∃c':U (isFemaleProcreator(w, c', p) ∨ isMaleProcreator(w, c', p)))
A5	A *LivingPerson* eventually becomes a *DeceasedPerson*.
	∀w:W, ∀p:U (LivingPerson(w, p) → ∃w':W(next(w, w') ∧ DeceasedPerson(w', p)))
A6	If a person is a *DeceasedPerson* in a world it remains a *DeceasedPerson* in all subsequent worlds.
	∀w, w':W, ∀p:U ((DeceasedPerson(w, p) ∧ next(w, w'))→ DeceasedPerson(w', p))
A7	The father of a person *y* is the person *x* who played the role of *MaleProcreator* in the *Conception* in which *y* played the role of *OffSpring.*
	∀w:W, ∀x,y:U (FatherOf(w, x, y) ↔ ∃w':W, ∃c:U(((w'=w) ∨ next(w', w)) ∧ isOffspring(w', c, y) ∧ isMaleProcreator(w', c, x)))
A8	The mother of a person *y* is the person *x* who played the role of *FemaleProcreator* in the *Conception* in which *y* played the role of *OffSpring.*
	∀w:W, ∀x,y:U (MotherOf(w, x, y) ↔ ∃w':W,∃c:U(((w'=w)∨ next(w', w)) ∧ isOffspring (w', c, y) ∧ isFemaleProcreator(w', c, x)))
A9	A person *x* is a parent of a person *y* iff *x* is *y's* father or mother
	∀w:W, ∀x,y:U (ParentOf(w, x, y) ↔(FatherOf(w, x, y) ∨ MotherOf(w, x, y)))
A10	A person *x* is ancestor of a person *y* iff *x* is parent of *y* or *x is* ancestor of *y's* parent.
	∀w:W, ∀x,y:U (AncestorOf(w, x, y) ↔ ParentOf(w, x, y) ∨(∃z:U(ParentOf(w, z, y) ∧ AncestorOf(w, x, z))

5 A DSL to Represent Genealogy Trees – Syntax and Semantics

In this section, we define the abstract syntax of a DSL for describing genealogy trees, and later we define the semantics for this language in terms of the genealogy ontology presented in the previous section.

5.1 Abstract Syntax Definition

Considering that a genealogy tree aims to map the ancestry of a given person, a DSL for representing genealogy trees must provide constructs to represent persons and to represent parenthood. Fig. 5 depicts the metamodel of such a DSL represented in Ecore. From now on we will refer to this language as *DSL1*. The names of metamodel elements are prefixed with "M" to avoid confusion with the ontology concepts.

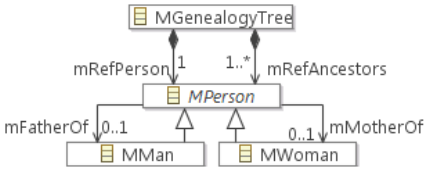

Fig. 5. A metamodel for representing genealogical trees (*DSL1*)

In the depicted metamodel, the *MGenealogyTree* class represents the genealogy tree itself. An *MGenealogyTree* is composed by instances of *MPerson*. The *MPerson* construct represents persons in a genealogy tree. The *MPersons* that compose a *MGenealogyTree* are divided into two groups according to their role in the tree: (i) one *MPerson*, identified by the *mRefPerson* association, represents the person whose ancestry is modeled by the tree, called the reference person of the tree; (ii) the other *MPersons* that composes the tree are referenced by the *mRefAncestors* association and represent the ancestors of the reference person. An *MPerson* may be associated with one *MMan* through an *mFatherOf* relation. In turn, an *mMotherOf* relation may be used to associate an *MPerson* with an *MWoman*. The *mFatherOf* and the *mMotherOf* constructs represent the parenthood relationships between the represented persons.

We define the language's abstract syntax with a theory in many-sorted first-order logic, quantifying over well-formed instances of the metamodel and over all possible instances of the language constructs. Thus, we assume two disjoint sets of symbolic entities: a set M of well-formed models (instances of the top-level container *MGenealogyTree*), and a set P of instances of *MMan* and *MWoman*. We further assume: (i) Binary predicates to represent the *mRefPerson* and the *mRefAncestors* containment relations, namely *mRefPerson(m,p)* and *mRefAncestor(m,p)* (e.g. *mRefPerson(m,p)* holds if an entity *p* is the reference person of a model *m*); (ii) Binary predicates to represent the constructs *MMan* and *MWoman*, namely *MMan(m,p)* and *MWoman(m,p)* (e.g. *MMan(m,p)* holds if an entity *p* is an instance of *MMan* and is part of a model *m*), and; (iii) Ternary predicates to represent *mFatherOf* and *MMotherOf* relations, namely *MFatherOf(m,a,b)*and *MMotherOf(m,a,b)* (e.g. *MFatherOf(m,a,b)* holds if, in a model m∈M, an instance *a* of *MMan* is associated to an *MPerson b* through a *mFatherOf* relation). For the sake of brevity, we omit here syntactic constraints that are implied by the metamodel (cardinality constraints, specialization relations).

The syntactic constraints not implied by the metamodel are shown in Table 2. The constraint C1 is a helper defining the *MAncestorOf* relation deriving it from *MFatherOf* and *MMotherOf* relations. The C2 constraint guarantees that a genealogy tree is all connected. C3 ensures that there are no ancestry cycles in a tree. Finally, C4 defines that the father and the mother of the *MRefPerson* must be represented.

Table 2. Syntactic constraints for *DSL1* (not implied by the metamodel)

C1	∀m:M, ∀a,b:P (MAncestorOf(m, a, b) ↔ MFatherOf(m, a, b) ∨ MMotherOf(m, a, b) ∨ ∃c:P((MFatherOf(m, c, b) ∨ MMotherOf(m, c, b)) ∧ MAncestorOf(m, a, c))
C2	∀m:M, ∀a,b:P ((MRefPerson(m, a) ∧ MRefAncestor(m, b)) → MAncestorOf(m, b, a))
C3	∀m:M, ∀a,b,c:P (MAncestorOf(m, a, b) → ¬ MAncestorOf(m, b, a))
C4	∀m:M, ∀a:P (MRefPerson(m, a) →∃b,c:P (MFatherOf(m, b, a) ∧ MMotherOf(m, c, a))

5.2 Real-World Semantics Definition

The syntax we have defined in the previous section is silent with respect to the various semantic issues for genealogy trees. For example, it does not define whether the language represents biological or legal parenthood (or both), if an (unborn) fetus may be represented in a model as a person, if a deceased person may be represented. These issues are the object of the language's real-world semantics, which we will define here as a logic theory binding the reference ontology and the abstract syntax.

Considering that a *DSL1* model aims to represent information about a set of admissible world states according to the genealogy ontology, the predicate *isAbout(m,w)* relates a model *m* in *DSL1* to an admissible world according to the ontology. When a model *m* represents information about a world *w*, the model elements may denote elements that exist in the world states. Thus, to represent the relation between an instance of a language construct and an instance of a real-world entity we define the predicate *refersTo(e,u)* that holds if an instance of a syntactic element $e \in$ P refers to an instance of real-world entity $u \in$ U. Table 3 shows the axioms that formalize the definition of the predicate *isAbout*, thereby characterizing the real-world semantic definition formally.

Table 3. Defining Real-World Semantics for *DSL1*

S1	∀m:M, ∀*w*:*W* (isAbout(m, *w*) → ∀a:P (MMan(m, a) → ∃!*x*:*U*(*Man(w, x)* ∧ refersTo(a, *x*))))
S2	∀m:M, ∀*w*:*W* (isAbout(m, *w*) → ∀a:P (MWoman(m, a) → ∃!*x*:*U*(*Woman(w, x)* ∧ refersTo(a, *x*))))
S3	∀m:M, ∀*w*:*W* (isAbout(m, *w*) → ∀a,b:P(MFatherOf (m,a,b) →∃!*x*,*y*:*U* (*FatherOf(x,y)* ∧ refersTo(a,*x*) ∧ refersTo(b,*y*))))
S4	∀m:M, ∀*w*:*W* (isAbout(m,*w*) → ∀a,b:P(MMotherOf (m,a,b) →∃!*x*,*y*:*U* (*MotherOf(x,y)* ∧ refersTo(a,*x*) ∧ refersTo(b,*y*))))
S5	∀m:M, ∀*w*:*W*((∀a:P (MMan(m,a) → ∃!*x*:*U*(*Man(w,x)* ∧ refersTo(a,*x*))) ∧ ∀a:P (MWoman(m,a) → ∃!*x*:*U*(*Woman(w,x)* ∧ refersTo(a,*x*)))∧ ∀a,b:P(MFatherOf (m,a,b) →∃!*x*,*y*:*U* (*FatherOf(x,y)* ∧ refersTo(a,*x*) ∧ refersTo(b,*y*))) ∧ ∀a,b:P(MMotherOf (m,a,b) →∃!*x*,*y*:*U* (*MotherOf(x,y)* ∧ refersTo(a,*x*) ∧ refersTo(b,*y*)))) → isAbout(m,*w*))

S1 relates the *MMan* construct to the ontology concept of *Man*, defining that if we have a model *m* which aims to describe a portion of a world *w* and there is in such model an instance *a* of the *MMan* construct then *a* must refer to a *Man x* which exists in *w*. S2 is similar to S1 dealing with the relation between the *MWoman* construct and

the concept of *Woman*. S3 states that if, in a model *m*, there is a *MFatherOf* relation between an instance *a* of *MMan* and an instance *b* of *MPerson*, then there must be, in the world *w*, a *FatherOf* relation between the *Man x* referred by *a* and the *Person y* referred by *b*. S4 places a similar statement on the relation between *MMotherOf* and *MotherOf*. Thus, S1-S4 define necessary conditions to predicate that a model *m* in *DSL1* is about a world *w*.

S5 states that if all *MMan* constructs that exist in *m* refers to instances of *Man* that exist in *w*, all *MWoman* constructs that exist in *m* refers to instances of *Woman* that exist in *w*, all *mFatherOf* relations that exist in the *m* refers to instances of *FatherOf* relations that exist in *w*, and all *mMotherOf* relations that exist in the *m* refers to instances of *MotherOfRelation* that exist in *w*, then the model *m isAbout* the world *w*. Thus, S5 defines the sufficient conditions to predicate that a model *m* in *DSL1* is about a world *w*.

6 Language Analysis

We have used Alloy to analyze some properties of *DSL1*. Alloy is a structural modeling language based on first-order logic [11]. A software framework named Alloy Analyzer supports model simulation and consistency checking assuming the *small scope hypothesis* in order to ensure tractability [11].

To enable the analysis, we have represented the three logic theories in three Alloy modules. The module corresponding to the genealogy ontology was derived automatically using the transformation from OntoUML to Alloy [12]. The module corresponding the *DSL1* metamodel (presented in Fig. 5) was derived following the systematic patterns defined in [13]. In this way, we have ensured the correspondence between the Alloy specifications and the models presented in Figures 4 and 5. The module representing the real-world semantics definition refers to the two previous modules, allowing us to verify that the combined specification is consistent.

In order to automatically verify which of the syntactic constraints are semantically-motivated, we have added the assumption that for every model *m* there is at least one world *w* such that *isAbout(m,w)* holds, i.e. we have assumed that the language has a *well-defined real-world semantics*. Then, we have verified each syntactic constraint as an assertion. When no counter-example is found, the Alloy Analyzer guarantees that, within a bounded scope, the assertion (in this case the tested syntactic constraint) is entailed by the remainder of the specification (in this case the semantic axioms in tandem with the ontology axioms). Thus, we can conclude the tested syntactic constraint is *semantically-motivated*.

Using this method, we have found that the syntactic constraint C3 (Table 2) is *semantically-motivated*. In fact, C3 reflects, in *DSL1* abstract syntax, the rule that ancestry cycles are not allowed by the ontology. Note that, assuming that for every model *m* there is at least one world *w* such that *isAbout(m,w)* holds, C3 is entailed by the ontological axioms A1, A2, A4, A7, A8, A9 and A10 in tandem with the semantic axioms S3 and S4. Thus, considering the abstract syntax in isolation, if C3 were missing there would be syntactically valid expressions of *DSL1* with no semantics.

Applying the same method, we have concluded that the syntactic constraints C2 and C4 are not *semantically-motivated.* Indeed, they are concerned with representation characteristics of genealogy trees and are not necessitated by the ontology (C2 guarantees that a genealogy tree is all connected and C4 defines the father and the mother of the *mRefPerson* must be represented). Consequently, the syntactic constructs *mRefPerson* and *mRefAncestors,* used in those constraints, are not mapped into ontological concepts by the *DSL1* real-world semantics definition. Another consequence is that these two constraints can be omitted and/or altered without affecting the *DSL1* real-world semantics as long the resulting theory is still consistent. This again can be checked automatically.

The minimum and maximum cardinality constraints of the *mFatherOf* and the *mMotherOf* relations were also analyzed (both are defined with 0..1 multiplicity on the metamodel). The analysis reveals that the maximum cardinality constraint is *semantically-motivated*. This is because the maximum cardinality constraint reflects the ontological rule that every *Person* has at most one father and one mother. The analysis also reveals that the minimum cardinality constraint is not semantically-motivated. This reflects a choice of the designer of *DSL1* to allow models in which a person is represented omitting her father and/or mother. The ontology, in turn, states that all *Persons* must have exactly one father and one mother (with the exception of the "original persons" which do not have parents).

This divergence between the ontological rules and syntactic constraints is a consequence of the fact that the language allows incomplete models with respect to the world states. In the genealogy case, it is reasonable to imagine that no one has information about all his ancestors. On the other hand, the ontology ought to describe a conceptualization of the reality and shall not be influenced by pragmatic issues, as the need to foresee a lack of information. It is worth noticing that this difference on constraints does not lead to inconsistencies on the *DSL1* semantics definition using the referred domain ontology. According to the semantic axiom S1, every *MMan* in a model must refer to a *Man* that exists in the worlds described by that model. So, if a model in *DSL1* presents an instance *m* of *MMan* with no *mFatherOf* association we can infer that the *Man* referred to by *m* has a *Father* which is not represented in the model or he is an "original" *Person*. No syntactic or semantic inconsistences arise.

7 Discussion

Our running example illustrates how the different purposes of reference domain ontology and language metamodel affect their definition. On the one hand, the domain ontology accounts for the parenthood and ancestry relationships in terms of participations in *Conception* events. On the other hand, since the language only aims at *representing* the parenthood relationship (instead of aiming at *grounding* the concept of parenthood), the *DSL1* metamodel represents the concepts of *MotherOf* and *FatherOf* as primitive syntactic constructs, not including constructs to represent *Conception* events. In this example we can observe that, the real-world semantics definition explicitly settles that *MotherOf* and *FatherOf* represent the biological notions of

parenthood (as grounded on *Conception* events). This is of course a particular worldview or conceptualization (or to be more precise, a particular ontological commitment) which is reflected by the ontology.

Furthermore, considering that it is assumed here unnecessary to know if a person is alive or not in the context of a genealogy tree specified in *DSL1* and aiming to reduce the language complexity (another example of pragmatic issue that may influence DSL metamodels) the concepts of *LivingPerson* and *DeceasedPerson* have no counterpart in the language metamodel. Thus, the approach of using a domain ontology to define the language semantics allowed us to clearly specify the worldview underlying the language without creating a counterpart construct for each real-world concept.

We should finally note that, when selecting a domain reference ontology to define the semantics of a language one must carefully analyze the suitability of assuming the underlying conceptualization as provider of the language real-world semantics. According to our ontology, a person exists as a result of the *Conception* event in which it plays the role of *Offspring*. So, even a fetus is considered a person. Since we have used this ontology as a basis to define *DSL1's* real-world semantics, this has the consequence that fetuses may appear in a genealogy tree. If this is considered undesirable by language designers, a domain ontology that would provide a distinction between a fetus and a person who is already born would be required. (This could possibly be a refinement of the ontology used in this paper).

8 Related Work

Our investigation was initially motivated by the apparent confusion involving the relation between ontologies and language metamodels in the literature. Some authors (e.g. [14]) posit that the origin of this confusion may be the fact that both are often depicted with the same or similar languages, such as variants of frame-based languages (of which UML in an example). There are many works in the literature (e.g. [6], [15], [16]) that point to similarities and differences between these two concepts and explore their relations. It is worth to note that various authors propose different criteria to distinguish ontology from metamodel, which indicates that this is still an open issue in the literature.

For example, Bezivin et al. [17] and Atkinson [18] suggest that the distinction between ontologies and model engineering artefacts (such as metamodels) is primarily a matter of technical space. As we have shown here, ontologies and metamodels play clear distinct roles in language engineering. Thus, while they can be harmonized into the same overall framework eliminating accidental differences from technical spaces (as we have done here with axiomatic logic theories) they serve different yet complementary purposes.

The integration of modeling languages is the focus of [16]. The authors argue that most existing integration approaches are metamodel-based, and that they face some difficulties because (domain) "concepts can be hidden in a metamodel". Then the authors propose a semi-automatic process to refactor metamodels into ontologies to reveal those hidden concepts. The defined patterns are based on the fact that "in a

metamodel not necessarily all modeling concepts are represented as first-class citizens". While the presented patterns are shown to be useful, we argue that the semantic issues are more complex than what can be addressed by metamodel refactoring, which does not address explicitly choices concerning the semantics of language elements.

In [15] the authors discuss the use of ontologies, models and metamodels in model-driven engineering (MDE). They present a proposal for the role of ontologies in meta-pyramid of MDA. This proposal argues that role of ontologies is to describe the existing world and the domain of the system while the role of system models is to specify and control the system under study. They allude to a relation between real-world objects and software objects at M0 ("is described by"). However, they position ontologies at M1 and do not elaborate on a possible relation between reference ontologies and language metamodels (at M2). We believe this is an important gap in their ontology-aware meta-pyramid for MDE, which we address in this paper.

The work of Ciocoiu and Nau [19] also consider the problem of providing semantics for declarative languages based on ontologies. Similarly to our approach, they show that information in language expressions may be incomplete or partial, and that a mapping to a domain ontology will reveal assumptions that are implicit in language expressions. Differently from our approach, they focus on language translation, and not on implications for the language engineering effort.

Finally, we believe that the work described in this paper has implications beyond language design and could also be applied to the area of database design. In particular, we believe that the so-called "semantic integrity constraints" in database systems literature is analogous to what we have called semantically-motivated syntactic constraints. According to [20], the purpose of semantic integrity constraint in database systems is "to avoid database states for which no correspondence can exist in the real world". While great importance has been given to the management of "semantic integrity constraints" in database systems, little attention has been devoted to the design and validation of such constraints. It is said intuitively, in the literature, that they reflect the universe of discourse but no guidelines are given on how to discover the necessary constraints or how to control their quality. Thus, the data modeler (not unlike the language designer) must face semantic issues (e.g., the definition of "semantic integrity constraints") with little methodological support. We believe our work has implications to the design of database systems, in that it could account for "semantic integrity constraints" as semantically-motivated syntactic constraints. Given that we could quantify over temporal aspects and histories (instead of world states only) the approach can be extended to cover both the so-called static and dynamic constraints.

9 Conclusions and Future Work

In this paper we have discussed a principled approach to define the real-world semantics for declarative domain-specific languages in terms of a reference domain ontology. We illustrated our approach with a running example, describing a domain ontology and using it to define the real-world semantics of a DSL. It allowed us to show clear examples of how reference ontologies and language metamodels differ.

We have argued for a strict separation of concerns distinguishing the role of reference ontologies and language metamodels, separating syntactic and semantic concerns and then linking them explicitly and precisely. A clear separation of concerns allows us to apply suitable modeling disciplines to each of the tasks at hand: understanding and capturing a domain conceptualization (a concern of ontology engineering) and dealing with abstract syntax design (a concern of language engineering).

Our approach is grounded on the fact that the relation between a language and reality is always mediated by a certain conceptualization [21]. If nothing is said about the conceptualization underlying a language, each language user may interpret a model based on his/her own concepts about reality. Thus, the formal definition of a language real-world semantics is essential to empower language's users to efficiently communicate about reality.

Besides its importance to avoid misunderstandings in a communication process, the explicit real-world semantics representation may bring some additional benefits to language's designers and users: (i) the real-world rules formalized by the ontology inform language design (e.g. guiding the definition of constructs and syntactic constraints); (ii) the explicit real-world semantic definition allows systematic evaluation of truthfulness of a language with respect to a specific domain conceptualization; (iii) reasoning may be used to infer some information not represented (explicitly) in the models using the semantic mapping and the ontological axioms.

Moreover, by formally characterizing the syntactic constraints, ontology axioms and semantic definition, our approach forms a basis for DSL design automation. Leveraging the existing transformations from OntoUML to Alloy, we have shown that it is possible to automatically verify which of the syntactic constraints are semantically-motivated. Without these constraints the language would produce models with no meaning. We believe the formal approach discussed here can be used to support more advanced evaluation of DSL semantics, systematizing the application of the ontological analysis approach discussed in [6]. We also intend to use the presented approach to enable the semantic interoperation of languages in multi-viewpoint modeling. Another challenge is to define a step-by-step ontology-driven DSL design approach. These are topics for further investigation.

Finally, the availability of a reference domain ontology greatly simplifies the task of the language designer. However, we should emphasize that our approach does not presuppose the development of a specific reference ontology with the sole purpose of supporting the design of a particular language. Ideally, the effort invested in the design of a reference ontology should be compensated by its reuse in a number of applications, e.g., the design of several languages, (semantic) language interoperability, database construction and integration, etc. A number of challenges concerning methodological implications of our approach remain open for further investigation, including guidelines for choosing a reference domain ontology suitable to define the real-world semantics of a DSL and guidelines for developing reusable ontologies.

Acknowledgments. This research is funded by the Brazilian Research Funding Agencies CAPES and CNPq (grants number 310634/2011-3, 311578/2011-0 and 485368/2013-7).

References

1. Olivé, A.: Conceptual Modeling of Information Systems. Springer(2007)
2. Weber, R.: Ontological Foundations of Information Systems. Coopers & Lybrand (1997)
3. Grice, H.P.: Logic and conversation. In: Cole, P., Morgan, J.(Eds.), Syntax and Semantics, Vol 3, pp. 43-58. Academic Press, New York (1975)
4. Davis, R., Shrobe, H., Szolovits, P.: What is a knowledge representation?. AI Magazine, Spring (1993)
5. Sheth, A.P., Kashyap, V.: So far (schematically) yet so near (semantically). In: Proc. of the IFIP WG 2.6 Database Semantics Conference on Interoperable Database Systems (1992)
6. Guizzardi, G.: On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. In: Vasilecas, O., Eder, J.,Caplinskas, A. (Eds.), Databases and Information Systems IV, pp. 18–39,. IOS Press, Amsterdam(2007)
7. Lowe, E.J.: Ontology. In: Honderich, T. (Ed.), The Oxford companion to philosophy, p. 670. Oxford University Press (2005)
8. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. University of Twente, Enschede, The Netherlands (2005)
9. Guarino, N.: Formal ontology and information systems. In: Proceedings of International Conference on Formal Ontology in Information Systems(FOIS), pp. 3-15, IOS Press (1998)
10. Hayes, P.: The Naïve Physics Manifesto. In: Ritchie, D. (Ed.), Expert Systems in Microeletronics age, pp 242-270. Edinburgh University Press (1978)
11. Jackson, D.: Software Abstractions: Logic, Language and Analysis. The MIT Press, (2006)
12. Benevides, A. B., Guizzardi, G., Braga, B. F. B., Almeida, J. P. A.: Validating Modal Aspects of OntoUML Conceptual Models Using Automatically Generated Visual World Structures. In: Journal of Universal Computer Science, 16, p. 2904-2933 (2011)
13. Anastasakis, K., Bordbar, B., Georg, G, Ray, I.: On Challenges of Model Transformation from UML to Alloy. Software & Systems Modeling, vol. 9, pp. 69-86. Springer (2010)
14. Ruiz, F., Hilera, J.R.: Using ontologies in software engineering and technology. In: Calero,C., Ruiz F.,Piattini, M. (Eds.), Ontologies for Software Engineering and Software Technology, 49-102. Springer-Verlag, Berlin (2006)
15. Aßmann, U., Zschaler, S., Wagner, G.: Ontologies, meta-models, and the model-driven paradigm. In: Calero,C., Ruiz F.,Piattini, M. (Eds.), Ontologies for Software Engineering and Software Technology, 249-273. Springer-Verlag, Berlin (2006)
16. Kappel, G., et al.: Lifting metamodels to ontologies – a step to the semantic integration of modeling languages. In: Proc. ACM/IEEE 9th Int'l Conf. on Model Driven Engineering Languages and Systems (MODELS'06), 528-542. Springer-Verlag, Berlin (2006)
17. Bézivin J., et. al.: An M3-neutral infrastructure for bridging model engineering and ontology engineering. In: Proc. of the First International Conference on Interoperability of Enterprise Software and Applications, p. 159-171. Springer (2005)
18. Atkinson C.: On the unification of MDA and web-based knowledge representation technologies. In: 1st International Workshop on the Model-Driven Semantic Web (2004)
19. Ciocoiu, M., Nau, D.: Ontology-Based Semantics. In: Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning, (2000)
20. Türker, C.: Semantic Integrity Constraints in Federated Database Schemata. Dissertations in Database and Information Systems, Vol. 63. Infix-Verlag, Sankt Augustin (1999)
21. Baldinger, K: Semantic Theory: Towards a Modern Semantics. Palgrave Macmillan (1980)