Towards a Logic of the Ontological Dodecagon

Giancarlo Guizzardi[1] and Gerd Wagner[2]

[1] Computer Science Department
Federal University of Esprito Santo, Brazil
gguizzardi@inf.ufes.br
[2] Chair of Internet Technology
Institute of Informatics
Brandenburg University of Technology, Germany
G.Wagner@tu-cottbus.de

Abstract. Tarski's semantics of predicate logic can be viewed as a correspondence theory of truth where predicate-logical statements are the truth-bearers and facts in the form of relation instances, which are based on the ontological assumption of three categories of things: individuals, relationships between individuals (aka relation instances) and relationship types (aka relations), are the truth-makers. This form of facts, and the underlying three-category ontology, although quite abstract, has turned out to be a good choice for mathematical logic as the logic of mathematical statements.

However, for defining the logic of statements in real-world domains we should better distinguish between more than just three categories of things. The experience made in the conceptual and computational modeling of real-world domains, e.g. with the help of the Unified Modeling Language (UML) or the Web Ontology Language (OWL), suggests that we need to distinguish between at least eight ontological categories: objects and object types, attributions and attributes, data values and datatypes, as well as references and reference properties. We define a predicate logic based on these eight ontological categories, but we argue, in the final part of the paper, that, for completeness, even two more categories of individuals, *qualities* and *relators*, and corresponding type categories, should be added, resulting in an ontological dodecagon.

Acknowledgements

This paper has been written to honour David Pearce on the occasion of his 60th birthday. In the academic genealogy, David is a one-step-predecessor of the second author and a two-step-predecessor of the first author (and a successor of Immanuel Kant and Gottfried Leibniz). The topic of the paper is related to David's early interest in ontological frameworks and intertheory relations as well as to his later interests in logics and knowledge representation and reasoning.

1 Introduction

Tarski's semantics of predicate logic can be viewed as a correspondence theory of truth where predicate-logical statements are the *truth-bearers* and facts, in the form of relation instances, which are based on the ontological assumption of three categories of things: individuals, relationships between individuals (aka relation instances) and relationship types (aka relations), are the *truth-makers.* This form of facts, and the underlying three-category ontology, although quite abstract, has turned out to be a good choice for mathematical logic as the logic of mathematical statements.

However, for defining the logic of statements in other domains, especially in real-world domains, we should better distinguish between more than just three categories of things. For instance, in E.J. Lowes book *The Four Category Ontology* (2006), four categories are distinguished, as depicted in Figure 1: *individuals* instantiating *types*, as well as two kinds of individuals: *objects* characterized by *tropes.* These distinctions, which have been attributed to Aristotle in the philosophical literature, are the basis of *The Logic of the Ontological Square* of (Schneider 2009). While Lowe has to defend this choice of four basic ontological categories against other philosophers who deny the existence either of universals or of tropes, there is no need to defend it in the area of foundational ontology for computational sciences where these distinctions (reading tropes as relationships) are widely accepted.

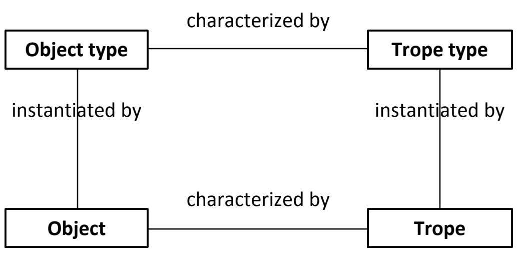

Fig. 1. The ontological square.

The experience made in the conceptual and computational modeling of real-world domains, e.g., with the help of the *Unified Modeling Language (UML)* or the *Web Ontology Language (OWL)* and its underlying description logic $\mathcal{SROIQ}$ (Horrocks et al 2006), suggests that we need to distinguish more than four ontological categories. In fact, the ontology underlying the model-theoretic semantics of OWL has eight categories, as depicted in Figure 2. Two kinds of trope types are distinguished: *reference properties*, which are binary relationship types (called 'object properties' in OWL and 'roles' in description logics) where the range is an object type, and *attributes*, which are binary relationship types (called 'data properties' in OWL) where the range is a datatype.

We argue that the eight-category ontology of OWL is still not rich enough for being able to account for all fundamental categories of individuals. We add two more categories of individuals (*qualities* and *relators*) and corresponding type categories, resulting in an ontological dodecagon as depicted in Figure 3. The goal is to develop a predicate logical formalism ***L12***, the logic of the ontological dodecagon, based on this twelve-category ontology.

Before we can start to develop ***L12***, we first establish its foundation, the logic of the ontological octagon ***L8***, as a *dialect*, more precisely: as a *semantic extension*, of Common Logic (CL), which is a non-standard classical first-order predicate logic that has been proposed as an ISO standard [1] for information exchange and transmission. Languages for traditional first-order logic, as introduced by Russell, Whitehead, Peano, Frege, Peirce and Tarski, exclude predicate quantifiers and the use of the same name in both predicate and argument position in an atomic sentence, which is permitted in Common Logic dialects.

This syntactic freedom is unusual, and may create the impression that CL is a higher-order logic in disguise. As has been pointed out by Hayes [4], however, "a superficially 'higher-order' syntax can be given a completely first-order semantics", and CL "indeed has a purely first-order semantics, and satisfies all the usual semantic criteria for a first-order language, such as compactness and the Skolem-Lowenheim property".

CL provides a superior framework for formal ontology since it allows to directly reflect ontological assumptions in a Tarski-style formal semantics. For instance, it allows to interpret predicates as names for certain things in the universe of discourse, representing various kinds of universals, in the same way as names for individuals have been interpreted in standard accounts of Tarski model-theoretic semantics. This allows expressing statements about universals and quantifying over them in the formal ontology, which is desirable whenever the ontological assumption is that universals exist along with individuals.

Traditional first-order predicate logics can be treated as 'segregated' dialects of CL which require a distinction to be made between lexical categories of names in order to check the admissibility of an expression in that dialect where names in one or more categories do not denote things in the universe of discourse.

2 The Logic of the Ontological Octagon

In this section, we define the logic ***L8*** of the ontological octagon depicted in Figure 2 as a dialect of Common Logic. Its semantics is defined in terms of a Tarski-style satisfaction relation between structures called ***L8****-interpretations* and ***L8***-text, which are sets of ***L8***-sentences.

We also show how to use ***L8*** for formalizing the object classification theory proposed in [3].

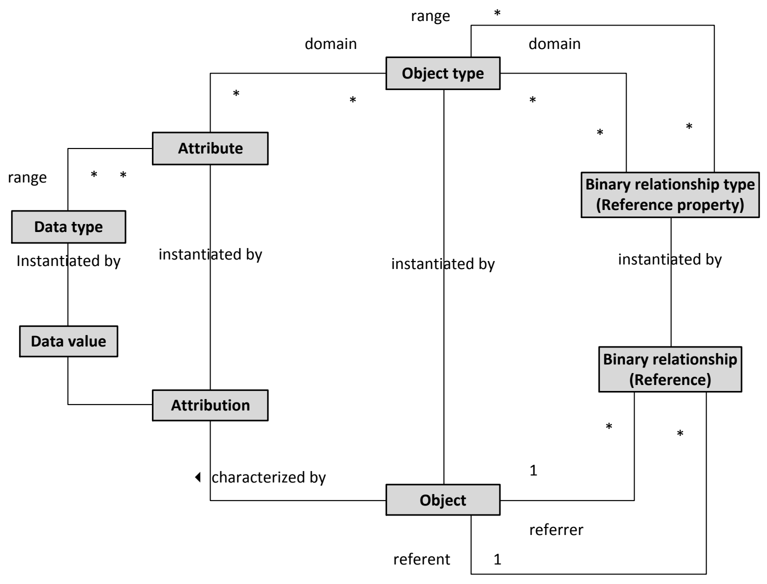

Fig. 2. The eight category ontology of OWL.

2.1 Datatypes

A datatype provides the possible values of an attribute. Attributes represent intrinsic properties that characterize object types. ***L8*** adopts the datatype concept of [5].

A datatype definition is a quadruple $\langle D, LS, VS, L2V\rangle$ where

1. D is the name of the datatype
2. LS is a non-empty set of Unicode character strings called the *lexical space* of D;
3. VS is a non-empty set called the *value space* of D;
4. $L2V$ is a mapping from LS to VS called the lexical-to-value mapping of D.

The elements of the lexical space are called *data literals*, and the elements of the value space are called *data values*. Data literals are (predefined) names for data values. They have a fixed interpretation provided by the lexical-to-value mapping.

The *extension* of a datatype is its value space.

2.2 Syntax

We adopt the syntax of the Common Logic Interchange Format (CLIF), which is defined using Extended Backus-Naur Form (EBNF), as specified by ISO/IEC 14977:1996. Literal chararacters are ‘quoted, sequences of items are separated by commas, | indicates a separation between alternatives, { } indicates a sequence of zero or more expressions in the enclosed category, - indicates an exception, [ ] indicates an optional item, and parentheses () are used as grouping characters. Productions are terminated with a semicolon.

Terms and Term Sequences A ***term*** is either a name or a complex term consisting of an operator, which is itself a term, together with a list of arguments in the form of a term sequence.

```
term = name | ( open, operator, termseq, close );

operator = term ;
```

Parentheses are self-delimiting lexical tokens used for grouping lexical tokens.

```
open = '(' ;

close = ')' ;
```

A ***term sequence*** is a finite sequence of terms.

```
termseq = { term } ;
```

A name is any lexical token denoting some thing from the universe of discourse. Predefined names with a fixed meaning are distinguished from interpretable names, which are given a meaning by an interpretation.

```
name = predefinedname | interpretablename ;
```

Predefined Names Quoted strings of the form 'aaa', which are called *plain data literals*, are predefined names. A *typed data literal* is a term (D 'aaa') consisting of a datatype name D as an operator and a quoted string 'aaa'.

In addition to a set of datatype names and the associated sets of data literals, the set of predefined names of ***L8*** contains the following

1. names for ontological categories: Object, ObjectType, DataValue, DataType, Attribute, and ReferenceProperty;
2. names for special properties: superclass, superproperty, domain, and range.

The meaning of these names is defined in Table 2 in Section 2.3.

Sentences A sentence is either atomic, Boolean (i.e., a negation, conjunction, disjunction, implication or biconditional), or quantified.

```
sentence = atomicsentence | booleansentence | quantifiedsentence ;
```

Atomic Sentences An atomic sentence is either an equation, a classification statement or property statement.

```
atomicsentence = equation | classificationstatement
                          | propertystatement ;

equation = open, '=', term, term, close ;

classificationstatement = open, predicate, term, close ;

propertystatement = open, predicate, term, term, close ;

predicate = term ;
```

Classification statements are expressed with the help of a unary predicate, while property statements are expressed with the help of a binary predicate. Semantically, see Section 2.3, we distinguish between two kinds of properties: intrinsic properties are called *attributes*, extrinsic relational properties are called *reference properties*.

Boolean Sentences Boolean sentences include negations of a sentence, conjunctions and disjunctions of any number of sentences, and implications and biconditionals of two sentences. The sentences (and) and (or), which denote a conjunction and a disjunction with zero arguments, can be used as the truth-values **true** and **false** respectively.

```
booleansentence = ( open, ('and' | 'or'), { sentence }, close ) |
                  ( open, ('if' | 'iff'), sentence, sentence, close ) |
                  ( open, 'not', sentence, close ;
```

Quantified Sentences A quantifier may bind any number of variables, which may be restricted to a category given by a term.

```
quantifiedsentence = open, ('forall' | 'exists'), boundlist, sentence, close ;

boundlist = open, { interpretablename |
            ( open, interpretablename, term, close )}, close ;
```

2.3 Semantics

The vocabulary of an ***L8***-text is the set of interpretable names occurring in the text.

The concept of an interpretation of a vocabulary is defined with respect to a given set of datatype definitions Δ. An ***L8***(Δ)-interpretation I of a vocabulary V is a structure $\langle U_I, O_I, OT_I, DT_I, A_I, RP_I, int_I, ext_I \rangle$ where

1. the set U_I, called universe of discourse, includes the pairwise disjoint sets O_I, OT_I, DT_I, A_I, RP_I, VS_1, ..., VS_n, where the VS_i are value spaces of the datatype definitions from Δ;
2. $O_I \subset U_I$ is a set of objects;
3. $OT_I \subset U_I$ is a set of object types;
4. $DT_I \subset U_I$ is a set of datatypes;
5. $A_I \subset U_I$ is a set of attributes;
6. $RP_I \subset U_I$ is a set of reference properties;
7. int_I maps datatype names from Δ and interpretable names from V to things from U_I
8. ext_I maps things that are types to their extensions:
   (a) an object type from OT_I is mapped to a subset of O_I,
   (b) a datatype from DT_I is mapped to some value space VS_i,
   (c) an attribute from A_I is mapped to a subset of $O_I \times VS_i$ for some value space VS_i,
   (d) a reference property from RP_I is mapped to a subset of $O_I \times O_I$;

We define the set of all properties $Prop_I$ to be the union of A_I and RP_I.

For any subset W of V, an interpretation J of V is a W-variant of I if J is just like I except that int_I and int_J might differ on what they assign to the members of W.

The interpretation of any expression of ***L8***(Δ) is then determined by the entries in Table 1.

The predefined names of ***L8*** are interpreted as in Table 2.

2.4 A Theory of Object Classification

We now formalize the object classification theory proposed in [3] in the form of an ***L8***-Text consisting of the axioms C1 – C11. In this theory, the following names for special categories of object types are defined: Sortal, RigidSortal, Kind, Subkind, AntiRigidSortal, Role, and PhaseType.

(C1) A sortal is an object type [that is endowed with an object identity condition for its instances].

```
(supertype Sortal ObjectType)
```

The object types Person, Car, Dog, Child and Student are examples of sortals.

(C2) A non-sortal object type cannot have direct instances.

```
(forall ((C ObjectType) x)
  (if
    (and (not (Sortal C)) (C x))
    (exists ((C' Sortal))
      (and (supertype C' C) (C' x))
    )
  )
)
```

Expression E	**The interpretation** $I(E)$
A plain data literal (quoted string) 'aaa'	aaa
A typed data literal (D_i 'aaa') where $\langle D_i, LS_i, VS_i, L2V_i \rangle \in \mathit{\Delta}$ and 'aaa' $\in LS_i$	$L2V_i('aaa')$
A datatype name or an interpretable name	$int_I(E)$
An equation ($= T_1\ T_2$)	true if $I(T_1) = I(T_2)$, otherwise false
A classification statement ($P\ T$) with P denoting an object type or a datatype	true if $I(T) \in ext_I(I(P))$, otherwise false
A property statement ($P\ T_1\ T_2$)	true if $\langle I(T_1), I(T_2) \rangle \in ext_I(I(P))$, otherwise false
A negation (not S)	true if $I(S)$ = false, otherwise false
A conjunction (and S_1 ... S_k)	true if $I(S_i)$ = true for all i, otherwise false
A disjunction (or S_1 ... S_k)	false if $I(S_i)$ = false for all i, otherwise true
An implication (if $S_1\ S_2$)	false if $I(S_1)$ = true and $I(S_2) = false$, otherwise true
A biconditional (iff $S_1\ S_2$)	true if $I(S_1) = I(S_2)$, otherwise false
A quantified sentence (forall (N_1 ... N_n) S) where $N = \{N_1, ..., N_n\}$ is the set of bindings for S	true if for every N-variant J of I, $J(S)$ = true, otherwise false
A quantified sentence (exists (N_1 ... N_n) S) where $N = \{N_1, ..., N_n\}$ is the set of bindings for S	true if for some N-variant J of I, $J(S)$ = true, otherwise false

Table 1. Interpretation of expressions.

Examples of non-sortal object types are RedThing and InsurableItem, as these object types do not provide any identity conditions for their instances, so we could not tell, for instance, if two red objects perceived at different times are the same or not.

(C3) A subtype of a sortal is again a sortal.

```
(forall ((C Sortal) (C' ObjectType))
  (if (supertype C' C) (Sortal C'))
)
```

(C4) A rigid sortal is a sortal [that necessarily classifies all its instances (as long as they exist)].

```
(supertype RigidSortal Sortal)
```

In other words, if x instantiates a rigid sortal O in some possible world, then x must instantiate O in all possible worlds, in which x exists.

Sentence	**Interpretation**
(Object T)	true if $I(T) \in O_I$, otherwise false
(ObjectType T)	true if $I(T) \in OT_I$, otherwise false
(DataValue T)	true if $I(T) \in \bigcup VS_i$, otherwise false
(DataType T)	true if $I(T) \in DT_I$, otherwise false
(Attribute T)	true if $I(T) \in A_I$, otherwise false
(ReferenzProperty T)	true if $I(T) \in RP_I$, otherwise false
(supertype P_1 P_2)	true if $I(P_1), I(P_2) \in OT_I$ or $I(P_1), I(P_2) \in DT_I$, and $ext_I(I(P_1)) \subseteq ext_I(I(P_2))$, otherwise false
(superproperty P_1 P_2)	true if $I(P_1), I(P_2) \in A_I$ or $I(P_1), I(P_2) \in RP_I$, and $ext_I(I(P_1)) \subseteq ext_I(I(P_2))$, otherwise false
(domain P_1 P_2)	true if $I(P_1) \in Prop_I$ and $I(P_2) \in OT_I$, and $ext_I(I(P_1))[1] \subseteq ext_I(I(P_2))$, otherwise false
(range P_1 P_2)	true if $I(P_1) \in A_I$ and $I(P_2) \in DT_I$, or $I(P_1) \in RP_I$ and $I(P_2) \in OT_I$, and $ext_I(I(P_1))[2] \subseteq ext_I(I(P_2))$, otherwise false

Table 2. Interpretation of predefined names.

(C5) A kind is a top node in a rigid sortal hierarchy.

```
(forall ((C Kind) (C' RigidSortal)) (if
  (supertype C C')
  (C' = C))
)
```

(C6) Every object must instantiate exactly one kind.

```
(forall ((o Object)) (exists ((C Kind)) (C o)))
```

(C7) A subkind is a rigid subtype of a kind.

```
(forall ((C RigidSortal) (C' Kind)) (if
  (supertype C C')
  (Subkind C))
)
```

(C8) An anti-rigid sortal is a sortal [that doesn't necessarily classify any of its instances].

```
(supertype AntiRigidSortal Sortal)
```

In other words, if x instantiates an anti-rigid sortal O in some possible world, then there is another possible world, in which x exists, but does not instantiate O.

(C9) A phase type is an anti-rigid sortal [that is specialized from a kind or subkind by means of a condition depending only on attributes of the kind or subkind].

```
(supertype PhaseType AntiRigidSortal)
```

(C10) A role is an anti-rigid sortal.

```
(supertype Role AntiRigidSortal)
```

(C11) A role is the image of a reference property].

```
(forall ((R Role))
  (exists ((rp ReferenzProperty))
    (forall ((o,o' Object))
      (if (rp o o') (R o'))
    )
  )
)
```

3 The Ontological Dodecagon

We motivate the ontological dodecagon depicted in Figure 3, as an extension of the ontological octagon, by arguing that there are two more kinds of individuals:

1. ***Qualities***, which are the foundation of attributions.
2. ***Relators***, which are the foundation of relationships, including references.

3.1 Attributions and Qualities

Attributions represent qualities.

For instance, the eye color of a person is a quality, which is something that inheres in that object, and in no other object. We can use a predicate standing for a color attribute for making an attribution statement about the eye color of a person. E.g. we could say that "The eye color of Kate is light blue". The underlying truthmaker of this statement is the corresponding attribution, which associates a corresponding color data value with the object referenced by the name 'Kate'. This attribution represents the quality. The same quality of an object may be represented by different attributions, involving different attributes associated with different datatypes. And the same data value may represent

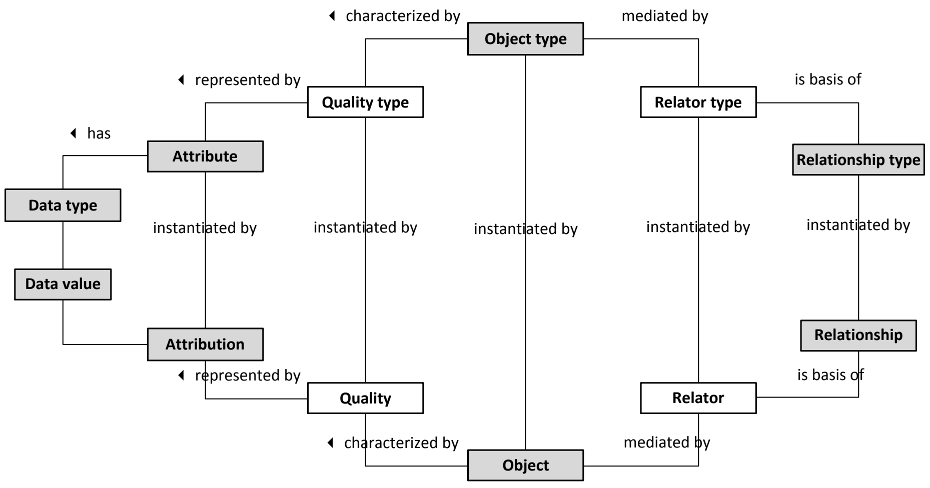

Fig. 3. The twelve category ontology of ***L12***.

different qualities in different attributions involving different objects. So, while we may say that two persons have the same eye color, the underlying eye color qualities are not the same.

It is important to understand that in our natural languages, we do normally not have predefined names ('constants') for qualities, but only for the data values that represent them (recall that these names are called *data literals*). We may consider the data value in an attribution as a 'conceptual approximation' of the quality it represents.

3.2 References, Relationships and Relators

Relationships represent relators. In particular, references represent binary relators. There may be different references, or relationships, representing the same relator. E.g., the two references 'Peter's father is Tom' and 'Tom's son is Peter' represent the same relator.

Our account of relators as the foundation of relationships supports the critical view of Kit Fine [2] who argues that the standard account of 'relations', according to which they apply to instances of their relata types in a specific order is flawed. This traditional account implies that the binary relationship of Tom being the father of Peter and its inverse, of Peter being the son of Tom, constitute two different facts (or states of affairs). However, the relationships of Tom being the father of Peter and Peter being the son of Tom correspond to two descriptions of the same state of affairs. Therefore, either we have to revise the concept of a 'relation' as a set of tuples, or we revise the Tarskian idea that the tuples of a 'relation' directly correspond to facts.

4 Conclusions

We have shown how a logic ***L8*** of the ontological octagon, which is the ontological foundation of the Web ontology language OWL, can be construed as a semantic extension of Common Logic. We have argued that, for ontological completeness, the ontological octagon has to be extended by adding qualities and quality types as well as relators and relator types, resulting in a twelve category ontology as depicted in the ontological dodecagon. In future work, we plan to define a logic ***L12*** of the ontological dodecagon as a semantic extension of ***L8***.

References

1. Common Logic (CL): a framework for a family of logic-based languages. ISO Standard ISO/IEC 24707:2007(E), `http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175_ISO_IEC_24707_2007%28E%29.zip`, October 2007.
2. Kit Fine. Neutral relations. *The Philosophical Review*, 109:1–33, 2000.
3. Giancarlo Guizzardi. *Ontological foundations for structural conceptual models.* CTIT Ph.D.-thesis series No. 05-74, University of Twente, Enschede, The Netherlands, 2005. ISBN 90-75176-81-3, `http://doc.utwente.nl/50826/1/thesis_Guizzardi.pdf`.
4. Pat Hayes. Translating Semantic Web languages into Common Logic. online draft manuscript (accessed 20120815), `http://www.ihmc.us/users/phayes/CL/SW2SCL.html`, July 2005.
5. Patrick Hayes. RDF Semantics. W3C Recommendation 10 February 2004, 2004. `http://www.w3.org/TR/rdf-mt/`.