Supporting Ontological Analysis of Taxonomic Relationships

Christopher Welty and Nicola Guarino

1 Introduction

Ontology is a discipline of Philosophy whose name dates back hundreds of years and whose practice dates back to Aristotle. Ontology is the study of the existence and nature of things, and includes questions such as, "what is a castle?" and "what is a hole?" The answers to these questions shed light on how we perceive and interact with the world. There is a subtle distinction between Ontology and Epistemology, which is the study of knowledge and knowing.

By the early 1980s, researchers in AI and especially in Knowledge Representation had realized that work in Ontology was relevant to the necessary process of describing the world for intelligent systems to reason about and act in. This awareness and integration grew, and spread into other areas until, in the latter half of the final decade of the 20th century, the term "ontology" actually became a buzzword, as enterprise modeling, e-commerce, emerging XML meta-data standards, and knowledge management, among others, reached the top of many businesses strategic plans. In addition, an emphasis on "knowledge sharing" and interchange has placed an emphasis on ontology as an application area in and of itself.

In general the accepted industrial meaning of "ontology" makes it synonymous with "conceptual model" and is completely independent of it's philosophical antecedents. In this paper we describe research that has led to a well-founded methodology for ontological analysis based on the philosophical underpinnings of that field, and an interesting description-logic based system that can be used to support this methodology. Although the methodology is not limited to analyzing taxonomies, we focus on that aspect of it here.

Most of the methodology described here has been published previously, as will be noted in specific sections. This paper is an extended version of [15] that presents a broader view of the overall methodology and an extended discussion of a system to support it.

2 Background

The notions upon which our methodology is based are subtle, and before describing them with formal rigor we discuss the basic intuitions behind them and how they are related to some existing notions in conceptual modeling.

2.1 Taxonomies

Taxonomies are a central part of most conceptual models. Properly structured taxonomies help bring substantial order to elements of a model, are particularly useful in presenting limited views of a model for human interpretation, and play a critical role in reuse and integration tasks. Improperly structured taxonomies have the opposite effect, making models confusing and difficult to reuse or integrate.

Clearly, insights into how to properly construct a taxonomy are useful. Many previous efforts at providing these insights have focused on the semantics of the taxonomic relationship (also called *is-a*, *class inclusion*, *subsumption*, etc.) [3], on different kinds of relations (*generalization*, *specialization*, *subset hierarchy*) according to the constraints involved in multiple taxonomic relationships (*covering*, *partition*, etc.) [28], on the taxonomic relationship in the more general framework of data abstractions [8], or on structural similarities between descriptions [2,5].

Our approach differs in that we focus on the arguments (i.e. the properties or concepts) involved in the subsumption relationship, rather than on the semantics of the relationship itself. The latter is taken for granted, as we take the statement "ψ subsumes φ" for arbitrary properties ψ and φ to mean simply:

$$x\ \varphi(x) \rightarrow \psi(x) \tag{1}$$

Our focus in this chapter will be on verifying the plausibility and the well-foundedness of single statements like (1) on the basis of the *ontological nature* of the two properties φ and ψ. Where e.g. description logics can determine whether one description subsumes another, this methodology can help determine whether or not the *categories the descriptions describe* actually do.

2.2 Basic Notions

We begin by introducing the most important philosophical notions: *identity*, *unity*, *essence*, and *dependence*. The notion of identity adopted here is based on intuitions about how we, as cognitive agents, in general interact with (and in particular recognize) individual entities in the world around us. Despite its fundamental importance in Philosophy, the notion of identity has been slow in making its way into the practice of conceptual modeling for information systems, where the goals of analyzing and describing the world are ostensibly the same.

The first step in understanding the intuitions behind identity requires considering the distinctions and similarities between *identity* and *unity.* These notions are different, albeit closely related and often confused under a generic notion of identity. Strictly speaking, identity is related to the problem of distinguishing a specific instance of a certain class from other instances of that class by means of a *characteristic property*, which is unique for *it* (that *whole* instance). Unity, on the other hand, is related to the problem of distinguishing the *parts* of an instance from the rest of the world by means of a *unifying relation* that binds the parts, and only the parts together. For example, asking, "Is that my dog?" would be a problem of identity, whereas asking, "Is the collar part of my dog?" would be a problem of unity.

Both notions encounter problems when time is involved. The classical one is that of *identity through change*: in order to account for common sense, we need to admit that an individual may remain *the same* while exhibiting different properties at different times. But which properties can change, and which must not? And how can we reidentify an instance of a certain property after some time? The former issue leads to the notion of an *essential property*, on which we base the definition of *rigidity*, discussed below, while the latter is related to the distinction between *synchronic* and *diachronic* identity. An extensive analysis of these issues in the context of conceptual modeling has been made elsewhere [13].

The fourth notion, *ontological dependence*, may involve many different relations such as those existing between persons and their parents, holes in pieces of cheese and the cheese, and so on [27]. We focus here on a notion of dependence as applied to

properties. We distinguish between *extrinsic* and *intrinsic* properties, according to whether they depend or not on other objects besides their own instances. An intrinsic property is typically something inherent to an individual, not dependent on other individuals, such as having a heart or having a fingerprint. Extrinsic properties are not inherent, and they have a relational nature, like "being a friend of John". Among these, there are some that are typically assigned by external agents or agencies, such as having a specific social security number, having a specific customer i.d., even having a specific name.

It is important to note that our ontological assumptions related to these notions ultimately depend on our *conceptualization* of the world [11]. This means that, while we shall use examples to clarify the notions central to our analysis, *the examples themselves will not be the point of this paper.* When we say, e.g. that "having the same fingerprint" may be considered an identity criterion for *PERSON*, we do *not* mean to claim this is the universal identity criterion for *PERSON*s, but that *if this were* to be taken as an identity criterion in some conceptualization, what would that mean for the property, for its instances, and its relationships to other properties?

To see how these intuitive notions can be used, consider a famous philosophical problem regarding the nature of a bunch of bricks. The bricks are made and sit in a pile for a while, and are then used to build a castle. The castle, over time, crumbles back into a pile of bricks. How would we describe the lifetime of this bunch of bricks in terms of properties, entities, and relations? Do we represent castles and bunches of bricks as two different properties? If so, do we have a single entity which is always subsumed by the "bunch of bricks" property, but which "moves" in and then out of the "castle" property, or do we have two entities, one subsumed by castle and the other by bunch of bricks with a relationship between them? Our analysis helps with these choices by exposing certain assumptions underlying them. For example, many would agree that a bunch of bricks is identified by the bricks themselves - if we remove or replace a few bricks then we have a *different* bunch. A castle, on the other hand, does not seem to have this property – we can remove or add bricks to the castle and still consider that it is the *same* castle. Our framework is designed to show the logical consequences of decisions like this.

These decisions are ultimately the result of our sensory system, our culture, etc. and again the aim of this methodology is to clarify the formal tools that can both make such assumptions explicit, and reveal the logical consequences of them.

2.3 Related Notions

Identity has many analogies in conceptual modeling for databases, knowledge bases, object-oriented, and classical information systems, however none of them completely captures the notion we present here. We discuss some of these cases below.

Membership conditions. In description logics, conceptual models usually focus on the sufficient and necessary criteria for class *membership*, that is, recognizing instances of certain classes [4]. This is not identity, however, as it does not describe how instances of the same class are to be told apart. This is a common confusion that is important to keep clear: membership conditions determine when an entity is an instance of a class, i.e. they can be used to answer the question, "Is that *a* dog?" but not, "Is that *my* dog?"

Globally Unique IDs. In object-oriented systems, uniquely identifying an object

(as a collection of data) is critical, in particular when data is persistent or can be distributed [33]. In databases, *globally unique id's* have been introduced into most commercial systems to address this issue. These solutions provide a notion of identity for the descriptions, for the units of data (individuals, objects or records), but not for the entities they describe. It still leaves open the possibility that two (or more) descriptions may refer to the same *entity*, and it is this entity that our notion of identity is concerned with. In other words, globally unique IDs can be used to answer, "Is this the same description of a dog?" but not, "Is this my dog."

Primary Keys. Some object-oriented languages provide a facility for overloading or locally defining the equality predicate for a class. In standard database analysis, introducing new tables requires finding unique keys either as single fields or combinations of fields in a record. These two similar notions very closely approach our notion of identity as they do offer evidence towards determining when two descriptions refer to the same entity. There is a very subtle difference, however, which we will attempt to briefly describe here and which should become more clear with the examples at the end of the chapter.

Primary (and candidate) keys and overloaded equality operators are typically based on *extrinsic properties* (see Section 2.2) that are required by a system to be unique. In many cases, information systems designers add these extrinsic properties simply as an escape from solving (often very difficult) identity problems. Our notion of identity is based mainly on *intrinsic properties*—we are interested in analyzing the inherent nature of entities and believe this is important for understanding a domain.

This is not to say that the former type of analysis never uses intrinsic properties, nor that the latter never uses extrinsic ones – it is merely a question of emphasis. Furthermore, our analysis is often based on information which *may not be represented in the implemented system*, whereas the primary key notion can never use such information. For example, we may claim as part of our analysis that people are uniquely identified by their brain, but brains and their possession may not appear in the final system we are designing. Our notion of identity and the notion of primary keys are not incompatible, nor are they disjoint, and in practice conceptual modelers will often need both.

3 The Formal Tools of Ontological Analysis

In this section we shall present a formal analysis of the basic notions discussed above, and we shall introduce a set of *meta-properties* that represent the behaviour of a property with respect to these notions. Our goal is to show how these meta-properties impose some constraints on the way subsumption is used to model a domain, and to present a description logic system for checking these constraints.

3.1 Preliminaries

Let's assume we have a first-order language $\mathbf{L}_0$ (the modeling language) whose intended domain is the world to be modeled, and another first order language $\mathbf{L}_1$ (the meta-language) whose constant symbols are the predicates of $\mathbf{L}_0$. Our meta-properties will be represented by predicate symbols of $\mathbf{L}_1$. Primitive meta-properties will correspond to *axiom schemes* of $\mathbf{L}_0$. When a certain axiom scheme holds in $\mathbf{L}_0$ for a certain property, then the corresponding meta-property holds in $\mathbf{L}_1$. This correspondence can be seen as a system of *reflection rules* between $\mathbf{L}_0$ and $\mathbf{L}_1$, which allow us to define a

particular meta-property in our meta-language, avoiding a second-order logical definition. Meta-properties will be used as *analysis tools* to characterize the ontological nature of properties in $\mathbf{L}_0$, and will always be defined with respect to a given conceptualization. In section 5 we present an representation of $\mathbf{L}_0$.

We denote primitive meta-properties by bolded letters preceded by the sign "+", "-" or "~" corresponding to *carrying* the meta-property, *not carrying* the meta-property, and *anti* the meta-property. The latter will be used to denote special restrictions that are stronger than the simple negation, and will be described in more detail, when relevant, for each meta-property. We use the notation $\varphi^{\mathbf{M}}$ to indicate that the property φ has the meta-property **M**.

In our analysis, we adopt a first order logic with identity. This will be occasionally extended to a simple temporal logic, where all predicates are temporally indexed by means of an extra argument. If the time argument is omitted for a certain predicate φ, then the predicate is assumed to be time invariant, that is $t\varphi(x, t) \rightarrow t\varphi(x, t)$. Note that the identity relation will be assumed as time invariant: if two things are identical, they are identical forever. This means that Leibniz's rule holds with no exceptions.

Our domain of quantification will be that of *possibilia*. That is, the extension of predicates will not be limited to what exists in the actual world, but to what exists in any possible world [23]. For example, a predicate like "Unicorn" will not be empty in our world, although no instance has actual existence there. Actual existence is therefore different from existential quantification ("logical existence"), and will be represented by the temporally indexed predicate E*(x,t)*, meaning that *x* has actual existence at time *t* [18].

Finally, in order to avoid trivial cases in our meta-property definitions, we shall implicitly assume the property variables as restricted to *discriminating properties* [10], i.e. properties *P* such that $\} xP(x) \quad \} x\neg P(x)$.

3.2 Rigidity

The notion of rigidity was defined previously in [10] as follows:

Definition 1 A r*igid property* is a property that is essential to *all* its instances, i.e. a property φ such that: $x\,\varphi(x) \rightarrow \Box\varphi(x)$.

Definition 2 A *non-rigid property* is a property that is not essential to *some* of its instances, i.e. $x\,\varphi(x) \quad \neg\Box\varphi(x)$.

Definition 3 An *anti-rigid property* is a property that is not essential to *all* its instances, i.e. $x\,\varphi(x) \rightarrow \neg\Box\varphi(x)$.

For example, we normally think of *PERSON* as rigid; if *x* is an instance of *PERSON*, it must be an instance of *PERSON* in every possible world. The *STUDENT* property, on the other hand, is normally not rigid; we can easily imagine an entity moving in and out of the *STUDENT* property while being the same individual.

Anti-rigidity was added as a further restriction to non-rigidity. The former constrains all instances of a property and the latter, as the simple negation of rigidity, constrains at least one instance. Anti-rigidity attempts to capture the intuition that all instances of certain properties must possibly not be instances of that property. Consider the property *STUDENT*, for example: in its normal usage, every instance of *STUDENT* is not necessarily so.

Rigid properties are marked with the meta-property **+R**, non-rigid properties are

marked with **-R**, and anti-rigid properties with **~R**. Note that rigidity as a meta-property is not "inherited" by sub-properties of properties that carry it, e.g. if we have $PERSON^{+\mathbf{R}}$ and $\forall x STUDENT(x) \rightarrow PERSON(x)$ then we know that all instances of *STUDENT* are necessarily instances of *PERSON*, but not *necessarily* (in the modal sense) instances of *STUDENT*. In some cases, we may even wish to assert $STUDENT^{\sim\mathbf{R}}$ to indicate that an instance of *STUDENT* can cease to be a student, however it may not cease to be a person.

Rigidity is intuitively tied to existence; when a person dies, they cease to be a person, and in order for the property to be rigid *ceasing to be a person* must imply ceasing to be. Ultimately all ontological questions become questions of existence.

3.3 Identity

In the philosophical literature, an *identity condition* (IC) for an arbitrary property φ is usually defined as a suitable relation ρ satisfying the following formula:

$$\forall x \quad \varphi(x) \quad \varphi(y) \rightarrow (\rho(x, y) \leftrightarrow x = y) \tag{2}$$

For example, the property *PERSON* can be seen as carrying an IC if relations like *having-the-same-SSN* or *having-the-same-fingerprints* are assumed to satisfy (2).

As discussed in more detail elsewhere [13,17], the above formulation has some problems, in our opinion. The first problem is related to the need for distinguishing between *supplying* an IC and simply *carrying* an IC: it seems that non-rigid properties like *STUDENT* can only carry their ICs, inheriting those supplied by their subsuming rigid properties like *PERSON*. The intuition behind this is that, since the same person can be a student at different times in different schools, an IC allegedly supplied by *STUDENT* (say, having the same registration number) may be only local, within a certain studenthood experience. This leads to the notion of *local identity conditions*, which we have discussed only briefly in [17] and [16], and requires further work.

The second problem regards the nature of the ρ relation: what makes it an IC, and how can we index it with respect to time to account for the difference between *synchronic* and *diachronic* identity?

Finally, deciding whether a property carries an IC may be difficult, since finding a ρ that is both necessary *and* sufficient for identity is often hard, especially for natural kinds and artifacts.

For these reasons, we have refined (2) as follows:

Definition 4 An *identity condition* is a *sameness formula* Σ that satisfies either (3) or (4) below, excluding trivial cases [13] and assuming the predicate E discussed in section 3.1:

$$\forall \quad E(x,t) \quad \varphi(x,t) \quad E(y,t') \quad \varphi(y,t') \quad x{=}y \rightarrow \Sigma(x,y,t,t') \tag{3}$$

$$\forall \quad E(x,t) \quad \varphi(x,t) \quad E(y,t') \quad \varphi(y,t') \quad \Sigma(x,y,t,t') \rightarrow x{=}y \tag{4}$$

An IC is necessary if it satisfies (3) and sufficient if it satisfies (4). Based on this, we define two meta-properties:

Definition 5 Any property *carries* an IC iff it is subsumed by a property supplying that IC (including the case where it supplies the IC itself).

Definition 6 A property φ *supplies* an IC iff i) it is rigid; ii) there is a necessary or sufficient IC for it; and iii) The same IC is not carried by *all* the properties subsuming φ. This means that, if φ inherits different (but compatible) ICs from multiple properties, it

still counts as supplying an IC.

Definition 7 Any property carrying an IC is called a *sortal* [30].

Any property carrying an IC is marked with the meta-property **+I** (**-I** otherwise). Any property supplying an IC is marked with the meta-property **+O** (**-O** otherwise). The letter "O" is a mnemonic for "own identity". From the above definitions, it is obvious that **+O** implies **+I** and **+R**. For example, both *PERSON* and *STUDENT* do carry identity (they are therefore **+I**), but only the former *supplies* it (**+O**).

3.4 Unity

In previous work we have extensively discussed and formalized the notion of unity, which is itself based upon the notion of part [13]. This formalization is based on the intuition that a whole is something all of whose parts are connected in such a way that each part of the whole is connected to all the other parts of that whole and nothing else. We assume here that the axiomatization of the part relation is as shown in Table 1, where $P(x,y,t)$ means that x is a (proper or improper) part of y at time t.

$PP(x,y,t) =_{def} P(x,y,t) \quad \neg x=y$	(proper part)
$O(x,y,t) =_{def} \quad z(P(z,x,t) \quad P(z,y,t))$	(overlap)
$P(x,y,t) \rightarrow E(x,t) \quad E(y,t)$	(actual existence of parts)
$P(x,y,t) \quad P(y,x,t) \rightarrow x=y$	(antisymmetry)
$P(x,y,t) \quad P(y,z,t) \rightarrow P(x,z,t)$	(transitivity)
$PP(x,y,t) \rightarrow \quad z(PP(z,y,t) \quad \neg O(z,x,t))$	(weak supplementation)

Table 1. Axiomatization of the part relation, adapted from Simons [27].

Briefly, we define:

Definition 8 An object *x is a whole under* ω iff ω is an equivalence relation such that all the parts of x are linked by ω, and nothing else is linked by ω.

Definition 9 A property φ *carries a unity condition* iff there exists a single equivalence relation ω such that each instance of φ is a whole under ω.

Depending on the ontological nature of the ω relation, which can be understood as a "generalized connection", we may distinguish three main kinds of unity for concrete entities (i.e., those having a spatio-temporal location). Briefly, these are:

- *Topological unity*: based on some kind of topological or physical connection, such as the relationship between the parts of a piece of coal or an apple.
- *Morphological unity*: based on some combination of topological unity and shape, such as a ball, or a morphological relation *between wholes* such as for a constellation.
- *Functional unity*: based on a combination of other kinds of unity with some notion of purpose as with artifacts such as hammers, or a functional relation between wholes as with artifacts such as a bikini.

As the examples show, nothing prevents a whole from having parts that are themselves wholes (with a different UC). This can be the foundation of a theory of *pluralities*, which is however out of this paper's scope.

As with rigidity, in some situations it may be important to distinguish properties that

do not carry a *common* UC for all their instances, from properties all of whose instances are not wholes. As we shall see, an example of the former kind may be *LEGAL AGENT*, all of whose instances are wholes, although with different UCs (some legal agents may be people, some companies). *AMOUNT OF MATTER* is usually an example of the latter kind, since none of its instances can be wholes. Therefore we define:

Definition 10 A property has *anti-unity* if every instance of the property is not a whole.

Any property carrying a UC is marked with the meta-property **+U** (**-U** otherwise). Any property that has anti-unity is marked with the meta-property **~U**, and of course **~U** implies **-U**.

3.5 Dependence

The final meta-property we employ as a formal ontological tool is based on the notion of dependence. As mentioned in Section 2.2, we focus here on ontological dependence as applied to properties. The formalization below is based on Simons' definition of *notional dependence* [27]. We are aware that this is only an approximation of the more general notion of extrinsic (or relational) property, and that further work is needed (see for instance [20]).

Definition 11 A property φ is *externally dependent* on a property ψ if, for all its instances x, necessarily some instance of ψ must exist, which is not a part nor a constituent of x:

$$x \Box (\varphi(x) \rightarrow \exists y \psi(y) \quad \neg P(y, x) \quad \neg C(y, x)) \tag{5}$$

The part relation P was discussed in Section 3.4. The relation C(x,y) is used to denote *constitution*. Constitution differs subtly from part, in that it refers to the substance of which an entity is made. A castle is made of bricks, a statue from (perhaps) marble. Constitution usually relates concrete entities to mereologically essential wholes (i.e. collections or masses). Constitution is an important notion to grasp, because it is commonly confused with subsumption. We discuss constitution with more rigor in [13], and give further examples of it in Section 6.

Clearly if we do not discount parts and constituents in (5), nearly all properties denoting classes of concrete entities would be dependent, since all non-atomic concrete entities have parts and are constituted of some material. In addition to excluding parts and constituents, a more rigorous definition must exclude qualities (such as colors), things which necessarily exist (such as the universe), and cases where ψ is subsumed by φ (since this would make φ dependent on itself). Intuitively, we say that, for example, *PARENT* is externally dependent on *CHILD* (one can not be a parent without having a child), but *PERSON* is not externally dependent on heart nor on body (because any person has a heart as a part and is constituted of a body).

An externally dependent property is marked with the meta-property **+D** (**-D** otherwise).

3.6 Constraints and Assumptions

Our meta-properties impose several constraints on taxonomic relationships, and to these we add several methodological points that help to reveal modeling problems in taxonomies.

A first observation descending immediately from our definitions regards some *sub-*

sumption constraints. If φ and ψ are two properties then the following constraints hold:

$$\varphi^{\sim \mathbf{R}} \text{ can't subsume } \psi^{+\mathbf{R}} \quad (6)$$

$$\varphi^{+\mathbf{I}} \text{ can't subsume } \psi^{-\mathbf{I}} \quad (7)$$

$$\varphi^{+\mathbf{U}} \text{ can't subsume } \psi^{-\mathbf{U}} \quad (8)$$

$$\varphi^{\sim \mathbf{U}} \text{ can't subsume } \psi^{+\mathbf{U}} \quad (9)$$

$$\varphi^{+\mathbf{D}} \text{ can't subsume } \psi^{-\mathbf{D}} \quad (10)$$

Properties with incompatible ICs/UCs are disjoint. (11)

Constraints (6-10) follow directly from our meta-property definitions (see [14] for more discussion and examples), and (11) should be obvious from the above discussion of identity and unity, but it is largely overlooked in many practical cases [17,14]. See Section 6 for an example that shows the practical use of these constraints.

Finally, we make the following assumptions regarding identity (adapted from Lowe [24]):

- *Sortal Individuation.* Every domain element must instantiate some property carrying an IC (**+I**). In this way we satisfy Quine's dicto "No entity without identity" [26].
- *Sortal Expandability.* If two entities (instances of different properties) are the same, they must be instances of a property carrying a condition for their identity.

4 Methodology

We are developing a methodology for conceptual analysis whose specific goal is to *make modeling assumptions clear.* One of the most important ways the methodology is used is in analyzing taxonomies to form *well-founded taxonomies*.

The methodology is made up of a number of formal analysis tools that can be grouped into four distinct layers, such that the notions and techniques within each layer are based on the notions and techniques in the layers below. In Figure 1, the methodology is depicted as four layers that support a process in which a person's or group's conceptualization evolves into a concrete conceptual model. In this section, we very briefly outline the purpose of each layer and the tools in it, followed by a brief discussion of a system to support the methodology.

4.1 First Layer: Foundations

In the lowest, foundational, layer of the methodology are the meta-properties described in Section 3. As discussed there, the meta-properties correspond to axiom schemes in the modeling language and properties in the meta-language. Properties in the modeling language correspond to constant symbols in the meta-language. This proves important for our support system, which implements only the meta-language.

4.2 Second Layer: Useful Property Kinds

The second layer in the methodology contains an ontology of useful property kinds. This is an extension of the formal ontology of *basic* property kinds presented in [14], which includes further specializations of *sortal* properties (Def. 7), each one corresponding to an identity or unity condition commonly found in practice. This ontology can be seen as a library of reference cases useful to characterize the meta-properties of a given property, and to check for constraint violations.

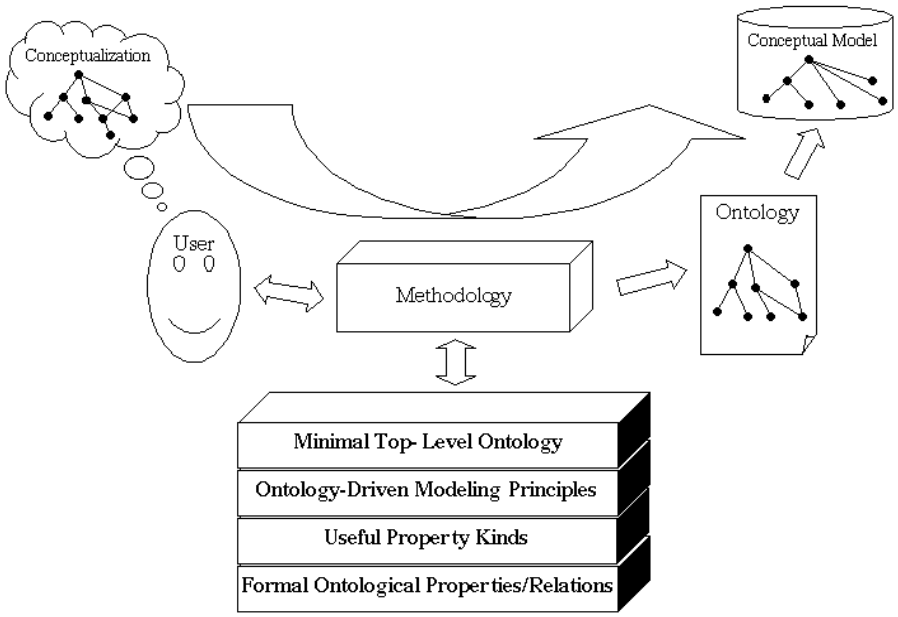

Figure 1: Overview of the methodology.

The formal ontology of properties discussed in [14] distinguishes eight different kinds of properties based on the valid and most useful combinations of the meta-properties discussed in Section 3. These are shown in Table 2 and in Figure 2. These property kinds enrich a modeler's ability to specify the meaning of properties in an ontology, since the definition of each property kind includes an intuitive and domain-

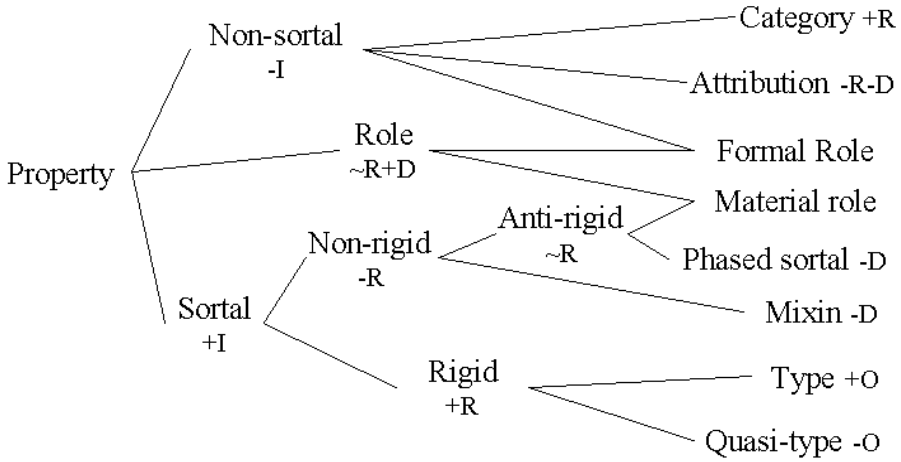

Figure 2: Taxonomy of properties.

independent description of how that kind of property should be used in an ontology.

+O	**+I**	**+R**	**+D**	**Type**	Sortal
			-D		
-O	**+I**	**+R**	**+D**	**Quasi-type**	
			-D		
-O	**+I**	**~R**	**+D**	**Material role**	
-O	**+I**	**~R**	**-D**	**Phased sortal**	
-O	**+I**	**¬R**	**+D**	**Mixin**	
			-D		
-O	**-I**	**+R**	**+D**	**Category**	Non-sortal
			-D		
-O	**-I**	**~R**	**+D**	**Formal Role**	
			-D	**Attribution**	
		¬R	**+D**		
			-D		
+O	**-I**			**incoherent**	
	+I	**~R**			
		-R			

Table 2: All possible combinations of the meta-properties.

In addition to these eight property kinds, we have identified several other specializations of sortal property that are useful in practice, and often particularly helpful in determining the basic meta-properties when their determination is not immediately clear. For each, we provide the meta-property symbol used to denote it:

CO. Countable Properties. This is an important specialization of sortals. In many cases, besides carrying identity (**+I**), countable properties also carry unity (**+U**). All subsumed properties must also be countable. Note that we appeal to a strict definition of countability provided in [13], which may not be immediately intuitive in the case of collections, such as a group of people. One can count possible groups of people in a combinatorial sense, but by our current definition of countability, a group of people is not countable because it does not have unity.

ME. Properties carrying a mereologically extensional IC. Certain properties, as discussed in Section 3.5, concerning masses or plural entities have as a necessary identity condition that the parts of their instances must be the same (instances cannot change their parts). For example *LUMP-OF-CLAY* or *GROUP-OF-PEOPLE*, if the people change, it is a different group. These properties cannot subsume properties with **-ME**.

UT. Properties carrying topological unity. See Section 3.4. Properties with **+UT** have unity (**+U**), and can not subsume properties with **-UT**.

UM. Properties carrying morphological unity. See Section 3.4. Properties with **+UM** have unity (**+U**), and can not subsume properties with **-UM**.

UF. Properties carrying functional unity. See Section 3.4. Properties with **+UF** have unity (**+U**), and can not subsume properties with **-UF**.

4.3 Third Layer: Ontology-Based Modeling Principles

The third layer in the methodology contains the notions of *backbone property* and *stratification.*

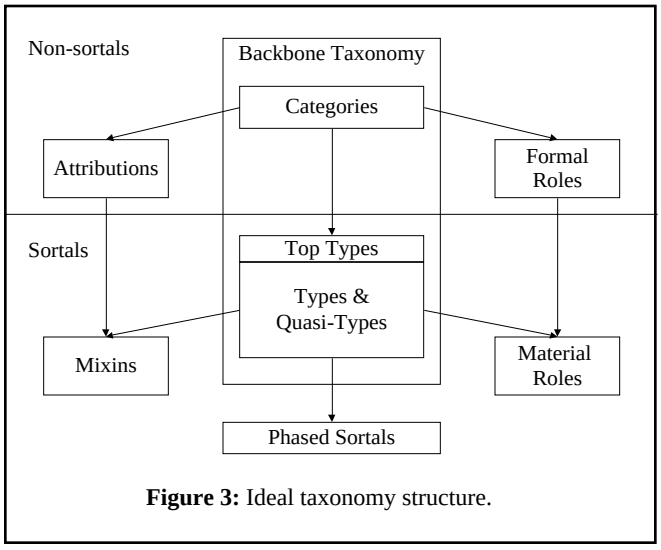

Figure 3: Ideal taxonomy structure.

The backbone taxonomy. One of the principal roles of taxonomies is to impart structure on an ontology, to facilitate human understanding, and to enable integration. We have found that a natural result of our analysis is the identification of special properties in a taxonomy that best fill this role. We call these properties *backbone properties*, which constitute the *backbone taxonomy* [14].

The backbone taxonomy consists only of rigid properties, which are divided into three kinds (as discussed above): *categories, types*, and *quasi-types*. Categories can not be subsumed by any other kinds of properties, and therefore represent the highest level (most general) properties in a taxonomy. They are usually taken as primitive properties because defining them is too difficult (e.g. entity or thing).

Types are critical in our analysis because according to the assumptions presented in Section 3.6, *every instance instantiates at least one of these properties.* Therefore considering only the elements of the backbone gives someone a survey of the entire universe of possible instances.

These notions give rise to an idealized view of how ontologies should be structured taxonomically, shown in Figure 3. While strict adherence to this idealized structure may not always be possible, we believe that following it to the degree possible will grow to be an important design principle for conceptual modeling, with payoffs in understandability and ease of integration.

Stratification. A very important result of our analysis is the recognition of multiple entities, based on different identity or unity criteria, where usually only one entity is conceived. The classical example is the statue and the clay it is made of, which count as different objects in our analysis. As discussed further in [12] as well as [16], this view results in a *stratified ontology*, where entities belong to different levels, depending on their identity and unity assumptions: we may distinguish for instance the physical level, the functional level, the intentional level, the social level. Entities at the higher levels are *constituted* (and co-located with) entities at the lower levels. The advantage of this view is a better semantic account of the taxonomic relation, a better account of the hid-

den ontological assumptions, and in general better ontologies. The costs are a moderate proliferation (by a constant factor corresponding to the number of levels) of the number of entities in the domain, and the necessity to take into account different relations besides *is-a*, such as *dependence*, *spatio-temporal colocalization*, and *constitution*.

4.4 Fourth Layer: Top Level Ontology

The highest layer of our methodology is a top-level ontology designed using the notions and techniques of the layers below. This layer of the methodology is not yet complete, however first steps towards this have been discussed in [12], and more recently in the context of the IEEE Standard Upper Ontology effort [7].

4.5 Question/answer system

Finally, we are capturing the notions and techniques from these four layers in a knowledge-based system that guides conceptual modelers through the analysis process. This approach is similar to that of [29], and is described more fully in Section 5. The system implements only the meta-language, providing some consistency checking of the constraints outlined in Section 3.6.

5 Knowledge Based Support

The methodology based on these techniques requires that the assignment of meta-properties to properties in an ontology be performed by hand. This analysis in all cases requires that the modeler be very clear about what each property *means*. We have developed a support system that can help modelers with this analysis in two important ways. First of all, it is an artifact capable of capturing and checking the consistency of the useful property kinds (see Section 4.2). For example, it is sometimes difficult to determine whether a property carries identity, however countability is something that is usually more clear. The system captures information such as “all countable properties have identity and unity,” and can infer one from the other. As we add new meta-properties to the methodology, they are captured in the system and tested.

Second, it can verify the consistency of the taxonomy based on the constraints described in Section 3.6. A modeler enters information about properties to be used in a conceptual modal, and the proposed taxonomic structure. Meta-properties are assigned, and the consistency of the taxonomy is then checked automatically.

In this section we describe aspects of the system and in the next section walk through a simple example using the system.

5.1 Overview

The system implements all the constraints and all the inferences described in previous sections for the meta-language. It has two basic modes of operation: a Q/A mode and a batch mode. It is our intention to make both modes of the system available on line.

In Q/A mode, the system is designed to assist a modeler in choosing the appropriate meta-properties for their properties by asking a series of questions. More general questions are asked first, such as “Does the property carry identity?” and the modeler may respond yes, no, or unsure. If unsure about a meta-property, more specific questions can be asked such as, “Are instances of the property countable?” It is always possible to leave answers unknown, of course in those cases the system can not verify the correctness of those properties.

As the properties are being entered, the information presented so far is checked for consistency, and any inferences are made. For example, if a modeler answers “yes” to the question that assigns the **~R** meta-property, the system will infer that the property is also **-R**. If a modeler assigns **+R** to a property and also asserts that it is subsumed by a property with **~R**, the system will immediately raise an appropriate error.

The system is implemented in CLASSIC, a description logic system developed at AT&T Bell Labs in 1990 [4]. CLASSIC was chosen mainly because of its familiarity to the implementors, however there are some good rationale for this choice. All the reasoning required of the meta-language is provided, almost no code other than I/O was needed. CLASSIC is the only description logic with a full explanation system implemented and included, making it possible to generate explanations for constraint violations (as opposed to simply saying there was a violation).

The total system without the meta-property definitions is under 14K of LISP code. The system works by first loading in the definitions, allowing us to add and modify them. Then the modeler may invoke the Q/A system or simply load a batch file consisting of properties and their meta-property tags (+R, +I, etc.).An example of how the

```
(define-meta-prop rigid-property nil
  :disjoint rigid
  :tag "+R"
  :classify-message "The property ~a is rigid"
  :question "Is this property rigid?")

(define-meta-prop non-rigid-property nil
  :tag "-R"
  :classify-message "The property ~a is non-rigid"
  :question "Is this property non-rigid?"
  :disjoint rigid)
```

```
(define-meta-prop anti-rigid-property
  non-rigid-property
  :tag "~R"
  :disjoint anti-rigid
  :classify-message "The property ~a is anti-rigid"
  :question "Is this property anti-rigid?")

(define-property 'red-apple :ask? nil
  :tags "+I-O+U-D~R"
  :rvs '((subsumed-by apple red)))
```

Figure 4: Example meta-property definitions.

meta-properties are defined is given in Figure 4, as well as an example property specification for batch mode.

5.2 Reasoning

The reasoning required of the system is fairly rudimentary for a description logic. In addition to the obvious implications of the property kind taxonomy shown in Figure 2 (i.e. a Type is a Sortal and a Rigid Sortal), the system is designed so that no redundant information need be specified, making the job of the modeler a bit easier.

The system uses the open world assumption regarding the meta-properties of the properties the modeler enters. A property is not assumed to have any meta-properties until they are asserted by the modeler. To accomplish this within the description logic framework, as well as provide for the possibility of “unknown” answers in a binary truth valued logic, *each possible* meta-property assignment is represented as a concept. For example, there is a concept corresponding to the Rigid meta-property, as well as concepts for non-rigid and anti-rigid, i.e. three concepts as shown in Figure 4. Opposite meta-properties are handled by making their corresponding concepts disjoint, thus when you assert a property is e.g. rigid, the system knows that it can not be non-rigid. Anti-rigid is asserted to be subsumed by non-rigid, thus when a property is assigned **~R** it is known to be **-R** as well. To leave a meta-property as unknown, the modeler must basically answer “no” to two questions, e.g. “Is the property Rigid?” and “Is the property non-rigid?”

It is important to keep in mind that since the properties of the modeler’s ontology are actually constant symbols in the meta-language, the modelers properties are repre-

sented as individuals in the system, and are instantiated when the modeler enters their names. They are allowed to have two relations: the generalization/specialization-of relation and the name of a characteristic relation. For each relation, the modeler is asked if there are values for it.

Since the modeler's properties are individuals, the description logic does not provide any special reasoning services for subsumption between them. Classic provides automatic relation inverses, so asserting that one property is a specialization of another causes the generalization inverse to be asserted as well. In addition, the generalization/specialization relations between the individuals are defined to be transitive, as expected. This requires some special machinery since Classic does not support transitivity, but can easily be accomplished in the standard way with a primitive (non-transitive) version of the relation and the transitive version, as with the parent/ancestor relations in Prolog.

All the constraints (other than the obvious ones provided by disjointness) are expressed as necessary conditions on the concepts representing the meta-properties. For example, constraint (6) in Section 3.6 is represented as a necessary condition on anti-rigid properties: *generalization-of* **.** *NON-RIGID-PROPERTY* (i.e. an anti-rigid property can only be the generalization of non-rigid properties).

All inference and constraint checking is done as soon as the information is made available by the modeler. The explanation system provided by Classic is therefore particularly important in batch mode, as the modeler can not benefit from the context of having just answered a question to know why an inconsistency was generated.

5.3 Evaluation

While the system was not designed to be particularly usable, we have evaluated it along two dimensions: effectiveness of the questions and scalability.

5.3.1 Scalability

We have tested the system on randomly generated hierarchies up to 20,000 nodes in batch mode. The reasoning the system performs is trivial and experimental results indicate two dimensions of complexity: number of properties and number of parents. In each dimension, complexity was observed to be linear, with the system performing all reasoning as fast as the batch files would load. Combining the two dimensions (i.e. large datasets with much multiple inheritance) resulted in polynomial increases.

We are not concerned with the latter result as the artificial data was difficult to generate, and based on our experiences does not seem to correspond to real systems. In fact, we believe an important result of our methodology is a drastic reduction in multiple inheritance links between properties [14].

5.3.2 Effectiveness of Questions

We have found that among people who have done a lot of conceptual modeling, many aspects of our methodology make sense. The main difficulty in applying it, however, is understanding when and what identity and unity conditions apply to properties in a domain.

We have attempted to gather together a few examples of common identity and unity criteria, and are in the process of collecting and analyzing them. The system is designed to incorporate this additional information as further sub-concepts of the existing meta-property definitions.

When a modeler is unsure about a particular meta-property, and therefore answers "no" to the questions for the two disjoint concepts that represent it, the system moves further down the hierarchy of concepts. If a modeler is sure about a meta-property and therefore answers "yes" to a question, the Q/A system does not ask any more questions about it.

We have made some informal attempts to test the effectiveness of this approach in leading modelers to the correct decisions about the meta-property assignments, however we have nothing concrete to report. The methodology is only useful for conceptual modelers with a certain amount of training, and access to these people for this kind of testing is not widely available. Testing on readily available student subjects has been entirely inconclusive, as their experience with conceptual modeling in general is suspect.

6 Example

In this section we provide a brief example of the way our analysis can be used. A complete version of this example is available [16].

We begin with a set of properties arranged in a taxonomy, as shown in Figure 5. In Table 3 we provide some basic explanations of the intended meaning of these properties. The taxonomy we have chosen makes intuitive sense *prima facie*, and in most cases the taxonomic pairs were taken from existing ontologies such as Wordnet [25], Pangloss [21], and CYC [22]. See [12] for more similar examples of intuitive taxonomic orderings in existing ontologies that fail our analysis.

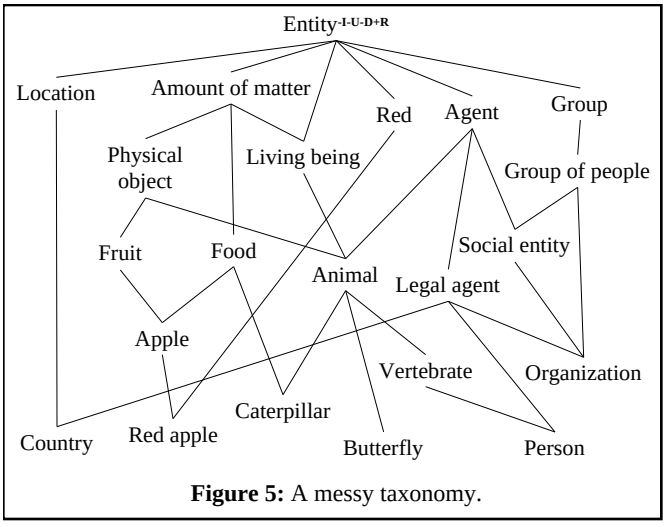

Figure 5: A messy taxonomy.

Property	Meta-properties	Kind	Notes
Entity	**-I-U-D+R**	Category	Everything is an entity.
Location	**+O~U-D+R**	Type	A generalized region of space. Locations supply +ME (mereological extensional IC), no UC, there is no way to isolate a location.
Amount-of-matter	**+O~U-D+R**	Type	Mass sortal: unstructured or scattered stuff, as lumps of clay or some bricks. +ME, no UC.

Table 3: Basic descriptions of the example properties.

Property	Meta-properties	Kind	Notes
Red	**-I-U-D-R**	Formal Attribution	Really Red-Thing. Generally a bad property to represent.
Agent	**-I-U+D~R**	Formal Role	An entity playing a part in some event. Agents have no *universal* IC/UC, i.e. one condition that holds for all its instances.
Group	**+O~U-D+R**	Type	An unstructured collection of wholes. +ME.
Physical Object	**+O+U-D+R**	Type	Isolated material objects. IC: same-spatial-location, topological UC.
Living Being	**+O+U-D+R**	Type	IC: same-DNA (necessary), biological UC.
Group of People	**+I-O~U-D+R**	Quasi-type	A group whose elements are people. Like a group, IC is ME, when the people change it is a different group.
Social Entity	**-I+U-D+R**	Category	A group of people brought together for some social reason. UC: social connection.
Fruit	**+O+U-D+R**	Type	A whole piece of fruit. IC: same-plant and same-shape (necessary), topological UC.
Food	**+I-O~U+D~R**	Material Role	An amount of edible stuff. Dependent on something that eats it, nothing is food necessarily.
Animal	**+O+U-D+R**	Type	IC: same-brain, biological UC.
Legal Agent	**+I-U+D~R**	Material Role	A legally recognized entity. Local IC, no universal unity, dependent on the legal body that recognizes it.
Apple	**+O+U-D+R**	Type	IC: same shape, color, skin pattern (necessary). Topological UC.
Country	**+I+U-D~R**	Phased Sortal	A place, recognized by convention. IC: government, regions. Countries are countable, so some UC. A place can stop being country and still exist (e.g. Prussia).
Red Apple	**+I-O-D~R**	Mixin	Inherits IC and UC from Apple. No apple is necessarily red.
Caterpillar	**+I+U-D~R**	Phased sortal	IC: legs, spots, cocoon. Biological UC, but the same entity can be something else, so anti-rigid.
Butterfly	**+I+U-D~R**	Phased sortal	IC: wing pattern, Biological UC, but the same entity can be something else, so anti-rigid.
Vertebrate	**+I-O+U-D+R**	Quasi-type	Really vertebrate animal. Biological classification, adds membership conditions but no IC/UC.
Organization	**+O+U-D+R**	Type	A group of people together for some reason, with roles that define some structure. IC: same-mission (necessary), and functional UC.
Person	**+O+U-D+R**	Type	IC could be same-fingerprint (sufficient), biological UC.

Table 3: Basic descriptions of the example properties.

The modeler, after making these initial decisions about the meta-properties and taxonomy, starts the system. For brevity, we assume the modeler enters the first four properties in batch mode, and the system responds as shown in Figure 6.

The final error indicates to the modeler that something is wrong with having *AMOUNT-OF-MATTER* subsume *LIVING-BEING*; the more general concept is mereologically extensional and has no unity, and a living being has biological unity. This is one of the most common modeling mistakes our methodology can reveal: living beings are not amounts of matter, they are *constituted* of matter. Constitution is not subsumption. The correction is to make *LIVING-BEING* subsumed directly by *ENTITY*.

```
---The property ENTITY does not carry unity
---Property ENTITY is a category.
---The property ENTITY does not carry identity and
is a non-sortal
---The property ENTITY is rigid
---The property ENTITY is independent
---The property ENTITY does not carry its own iden-
tity

Initial Classification of ENTITY: +CA +R -D -U -I -
O

---The property LOCATION does not carry unity
---Property LOCATION is a type.
---The property LOCATION is rigid
---The property LOCATION carries identity and is a
sortal
---The property LOCATION carries its own identity
---The property LOCATION is independent

Initial Classification of LOCATION: +I +R -D -U +O
+TP

---The property AMOUNT-OF-MATTER carries anti-
unity.
---The property AMOUNT-OF-MATTER does not carry
unity
---Property AMOUNT-OF-MATTER is a type.
---The property AMOUNT-OF-MATTER is rigid
---The property AMOUNT-OF-MATTER carries identity
and is a sortal
---The property AMOUNT-OF-MATTER carries its own
identity
---The property AMOUNT-OF-MATTER is independent

Initial Classification of AMOUNT-OF-MATTER: +I +O
~U -D +R -U +TP

---The property LIVING-BEING carries unity.
---Property LIVING-BEING is a type.
---The property LIVING-BEING is rigid
---The property LIVING-BEING carries identity and
is a sortal
---The property LIVING-BEING carries its own iden-
tity
---The property LIVING-BEING is independent
*CLASSIC ERROR* while processing:
  Trying to combine disjoint primitives:
    @tc{UC-PROP} and @tc{NON-UC-PROP}.
*EXPLANATION*: ~U (AMOUNT-OF-MATTER) cannot sub-
sume +U (LIVING-BEING).
```

Figure 6: Sample output for first four properties in batch mode.

The modeler makes this correction and proceeds. For the next property, *RED*, we show an interaction with the Q/A system in Figure 7. In this interaction, we can see that the modeler is unsure about whether or not the property carries an IC, answering no to both questions. The system then asks a question regarding countability, which is an indicator for identity and unity. The modeler indicates that not all instances of red can be counted (consider the number of red patches in a red carpet), and the system concludes that the property does not carry unity or identity.

This gives a flavor for how the systems works. We now skip to the next problem property in the ontology. When the modeler enters the information for *PHYSICAL-OBJECT*, the system raises an error:

```
Trying to combine disjoint primitives: UC-PROP and NON-UC-PROP.
*EXPLANATION*: ~U (AMOUNT-OF-MATTER) cannot subsume +U (PHYSICAL-OBJECT).
```

This is yet another example of constitution being confused with subsumption. Physical objects are not themselves amounts of matter, they are constituted of matter. The

```
? (define-property 'red)
Is this property subsumed by any others? (y or n)  y
What is it (list for multiple values): entity
Are instances of this property identified by a characteristic relation? (y or n)  n
Is this property anti-rigid? (y or n)  n
Does the property carry its own identity? (y or n)  n
Is this property rigid? (y or n)  n
Is this property non-rigid? (y or n)  y
Are instances of this property dependent on instances of another property? (y or n)  n
Are instances of this property independent? (y or n)  y
Does the property carry an identity criterion (answer no if unknown)? (y or n)  n
Are instances of this property unindentifiable from each other? (y or n)  n
Are all instances of this property countable? (y or n) n
Are there instances of this property that are not countable? (y or n) y
---The property RED does not carry unity
---Property RED is a non-sortal
---Property RED is an attribution

Initial Classification of RED: -I -U -D -R -O +AT
```

Figure 7: Sample output for *RED* in Q/A mode.

solution is to make *PHYSICAL-OBJECT* subsumed directly by *ENTITY*.

The next problem occurs with the property animal, which was declared to be subsumed by agent:

```
Trying to combine disjoint primitives: INDEPENDENT-PROP and DEPENDENT-PROP.
*EXPLANATION*: +D (AGENT) cannot subsume -D (ANIMAL).
```

This is a different kind of problem in which subsumption is being used to represent a type restriction. The modeler intends to mean, not that all animals are agents, but that animals *can be* agents. This is a very common misuse of subsumption, often used by object-oriented programmers. The correct way to represent this kind of relationship is with a covering, i.e. $x\ AGENT(x) \rightarrow SOCIAL\text{-}ENTITY(x) \quad ANIMAL(x)$. Clearly this is a different notion than subsumption. The solution is to remove the subsumption link between *ANIMAL* and *AGENT*, and represent this information elsewhere.

The system then finds two problems with the property *COUNTRY*. It is subsumed by $LOCATION^{\sim U}$, which creates a conflict since we have $COUNTRY^{+U}$, and it is subsumed by $LEGAL\text{-}AGENT^{+D}$, which creates a conflict since we have $COUNTRY^{-D}$. Closer inspection reveals another common misuse of subsumption: collapsing multiple meanings into a single concept. In this case, the modeler was thinking of a country as both a geographic region and a political entity. The methodology shows that, since these two meanings of country have different meta-properties, they can not be represented as one property. The solution is to break the concept into two, *COUNTRY* and *GEOGRAPHICAL-REGION*.

The modeler proceeds, fixing problems like these until finished. The final corrected taxonomy is shown in Figure 8. More detailed descriptions of all the errors in the initial taxonomy and the solutions can be found in [16].

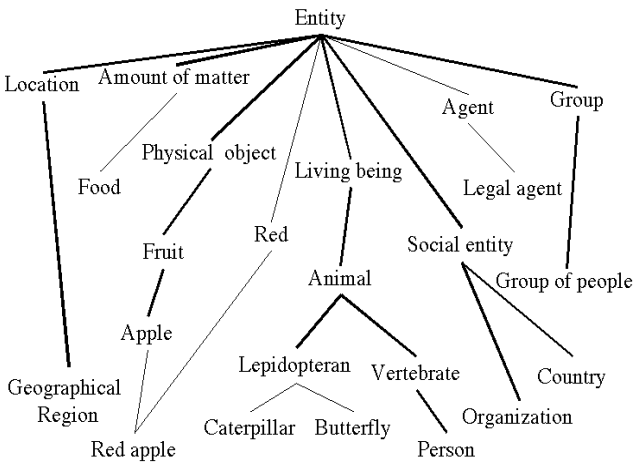

Figure 8: The final taxonomy with highlighted backbone.

In addition to checking constraints and performing simple inference, the system also supports the methodology with several simple procedures for displaying useful slices of the ontology, such as the backbone taxonomy, the role taxonomy, phased sortals, etc.

Phased sortals are themselves cause for special consideration. Some attempt at describing them was made in [14] based on the work of Wiggins [34], however further analysis and clarification is needed. This remains an open issue. Real phased sortals seem to appear rarely in our experience, and therefore isolating them and checking that they are correct is a useful practice.

7 Conclusion

We have discussed several notions of Formal Ontology used for ontological analysis in Philosophy: identity, unity, essence, and dependence. We have formalized these notions in a way that makes them useful for conceptual modeling, and introduced a methodology for ontological analysis founded on these formalizations.

Our methodology is supported by a system that helps the conceptual modeler study the deep ontological issues surrounding the representation of properties in a conceptual model, and we have shown how this methodology can be used to analyze individual taxonomic links and make the taxonomy more understandable. In particular, we have also shown how to identify the backbone taxonomy, which represents the most important properties in an ontology that subsume every instance.

Unlike previous efforts to clarify taxonomies, our methodology differs in that:

- It focuses on the nature of the properties involved in subsumption relationships, not on the nature of the subsumption relation itself (which we take for granted).
- It is founded on formal notions drawn from Ontology (a discipline centuries older than database design), and augmented with practical conceptual design experience, as opposed to being founded solely on the former or latter.
- It focuses on the validation of single subsumption relationships based on the *intended meaning* of their arguments in terms of the meta-properties defined here, as opposed to focusing on structural similarities between property descriptions.

Finally, it is important to note again that in the examples we have given, we are providing a way to make the *meaning* of properties in a certain conceptualization clear. We do not, for example, mean to claim that "Person is-a Legal-Agent" is wrong. We are trying to point out that *in a particular conceptualization* where *LEGAL-AGENT* has certain meta-properties (such as being anti-rigid) and *PERSON* certain others (such as being rigid), it is inconsistent to have person subsumed by legal-agent.

References

1. Artale, A., Franconi, E., Guarino, N., and Pazzi, L. 1996. Part-Whole Relations in Object-Centered Systems: an Overview. *Data and Knowledge Engineering*, **20**(3): 347-383.
2. Bergamaschi, S. and Sartori, C. 1992. On Taxonomic Reasoning in Conceptual Design. *ACM Transactions on Database Systems*, **17**(3): 285-422.
3. Brachman, R. 1983. What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer*, **16**(10): 30-36.
4. Brachman, R. J., McGuinness, D. L., Patel-Schneider, P. F., Resnick, L., and Borgida, A. 1990. Living with CLASSIC: When and How to Use a KL-ONE-like Language. In J. Sowa (ed.) *Principles of Semantic Networks.* Morgan Kaufmann: 401-456.
5. Calvanese, D., Lenzerini, M., and Nardi, D. 1998. Description Logics for Conceptual Data Modeling. In J. Chomicki and G. Saake (eds.), *Logics for Databases and Information Systems*. Kluwer: 229-264.
6. Elmasri, R., Weeldreyer, J., and Hevner, A. 1985. The category concept: An extension to the Entity-Relationship model. *Data and Knowledge Engineering*, **1**(1): 75-116.
7. Gangemi, A., Guarino, N., Masolo, C., and Oltramari, A. 2001. Understanding top-level ontological distinctions. LADSEB/CNR Technical Report 04/2001.

8. Goldstein, R. C. and Storey, V. C. 1999. Data abstractions: Why and how? *Data and Knowledge Engineering*, **29**: 293-311.
9. Gruber, T. R. 1993. Model Formulation as a Problem-Solving Task: Computer-Assisted Engineering Modeling. *International Journal of Intelligent Systems*, **8**: 105-127.
10. Guarino, N., Carrara, M., and Giaretta, P. 1994. An Ontology of Meta-Level Categories. In D. J., E. Sandewall and P. Torasso (eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*. Morgan Kaufmann, San Mateo, CA: 270-280.
11. Guarino, N. 1998a. Formal Ontology in Information Systems. In N. Guarino (ed.) *Formal Ontology in Information Systems*. Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998. IOS Press, Amsterdam: 3-15.
12. Guarino, N. 1999b. The Role of Identity Conditions in Ontology Design. In *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*. Stockholm, Sweden, IJCAI, Inc.
13. Guarino, N. and Welty, C. 2000. Identity, Unity, and Individuality: Towards a Formal Toolkit for Ontological Analysis. In *Proceedings of ECAI-2000: The European Conference on Artificial Intelligence.* Berlin, Germany, IOS Press.
14. Guarino, N. and Welty, C. 2000. A Formal Ontology of Properties. In Rose Dieng (ed.), *Proceedings of 12th Int. Conf. on Knowledge Engineering and Knowledge Management*, Springer Verlag LNCS.
15. Guarino, N. and Welty, C. 2000. Ontological Analysis of Taxonomic Relations. In A. Länder and V. Storey (eds.), *Proceedings of ER-2000: The International Conference on Conceptual Modeling*. Springer Verlag LNCS Vol. 1920.
16. Guarino, N. and Welty, C. 2000. *Ontology-Driven Conceptual Analysis.* AAAI-2000 Tutorial Presentation. Austin, Texas. Notes available at: http://www.cs.vassar.edu/faculty/welty/aaai-2000/
17. Guarino, N. and Welty, C. 2001. Identity and Subsumption. In, Green, R., ed., *Semantic Relations.* Kluwer.
18. Hirst, G. 1991. Existence Assumptions in Knowledge Representation. *Artificial Intelligence*, **49**: 199-242.
19. Huitt, R. and Wilde, N. 1992. Maintenance Support for Object-Oriented Programs. *IEEE Trans. on Software Engineering*, **18**(12).
20. Humberstone, I. L. 1996. Intrinsic/Extrinsic. *Synthese*, **108**: 205-267.
21. Knight, K. and Luk, S. 1994. Building a Large Knowledge Base for Machine Translation. In *Proceedings of American Association of Artificial Intelligence Conference* (AAAI-94). Seattle, WA: 773-778.
22. Lenat, D. and Guha, R. V. 1990. *Building Large Knowledge-Based Systems.* Addison-Wesley, Reading, MA.
23. Lewis, D. 1983. New Work for a Theory of Universals. *Australasian Journal of Philosophy*, **61**(4).
24. Lowe, E. J. 1989. *Kinds of Being. A Study of Individuation, Identity and the Logic of Sortal Terms.* Basil Blackwell, Oxford.
25. Miller, G. A. 1995. WORDNET: A Lexical Database for English. *Communications of ACM*, **2**(11):39-41.
26. Quine, W. V. O. 1969. *Ontological Relativity and Other Essays.* Columbia University Press, New York, London.
27. Simons, P. 1987. *Parts: a Study in Ontology.* Clarendon Press, Oxford.
28. Storey, V. C. 1993. Understanding Semantic Relationships. *Very Large Databases Journal*, **2**: 455-488.

29. Storey, V., Dey, D., Ullrich, H., and Sundaresan, S. 1998. An ontology-based expert system for database design. *Data and Knowledge Engineering,* **28**: 31-46.
30. Strawson, P. F. 1959. *Individuals. An Essay in Descriptive Metaphysics.* Routledge, London and New York.
31. Teorey, T. J., Yang, D., and Fry, J. P. 1986. A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. *ACM Computing Surveys,* **18**(2): 197-222.
32. Welty, C. *A description-logic based system for ontology-driven conceptual analysis.* System demo available at http://untangle.cs.vassar.edu/odca/.
33. Wieringa, R., De Jonge, W., and Spruit, P. 1994. Roles and dynamic subclasses: a modal logic approach. In *Proceedings of European Conference on Object-Oriented Programming*. Bologna.
34. Wiggins, D. 1980. *Sameness and Substance*. Blackwell, Oxford.