To appear in Rose Dieng (ed.), *Proceedings of 12th Int. Conf. on Knowledge Engineering and Knowledge Management*, Lecture Notes on Computer Science, © Springer Verlag 2000

A Formal Ontology of Properties

Nicola Guarino and Christopher Welty†

LADSEB-CNR

Padova, Italy
{guarino,welty}@ladseb.pd.cnr.it
http://www.ladseb.pd.cnr.it/infor/ontology/ontology.html
† on sabbatical from Vassar College, Poughkeepsie, NY

Abstract. A common problem of ontologies is that their taxonomic structure is often poor and confusing. This is typically exemplified by the unrestrained use of subsumption to accomplish a variety of tasks. In this paper we show how a formal ontology of unary properties can help using the subsumption relation in a disciplined way. This formal ontology is based on some meta-properties built around the fundamental philosophical notions of *identity, unity, essence,* and *dependence*. These meta-properties impose some constraints on the subsumption relation that clarify many misconceptions about taxonomies, facilitating their understanding, comparison and integration.

1 Introduction

Ontologies are becoming increasingly popular in practice, but a principled methodology for building them is still lacking. Perhaps the most common problem we have seen in practice with ontologies is that, while they are expected to bring order and structure to information, their taxonomic structure is often poor and confusing. This is typically exemplified by the unrestrained use of subsumption to accomplish a variety of reasoning and representation tasks. For example, in previous work [5] several unclear uses of the *is-a* relation in existing ontologies were identified, such as:

1. a physical object is an amount of matter (Pangloss)
2. an amount of matter is a physical object (WordNet)

This striking dissimilarity poses a difficult integration problem, since the standard approach of generalizing overlapping concepts would not work, and shows that even the most experienced modelers need some guidance for using subsumption consistently.

Our answer to problems like this lies in a better understanding of the nature of the properties corresponding to taxonomic nodes. To facilitate this understanding, we first introduce some meta-properties resulting from a revisitation of the fundamental philosophical notions of *identity*, *unity*, *essence*, and *dependence*, and we show how they impose some natural constraints on taxonomic structure that facilitate ontology understanding, comparison and integration. We then explore in a systematic way how these meta-properties can be combined to form different *kinds* of properties. The result of this analysis is a meta-level ontology of properties, which helps to make explicit the meaning every property has within a certain conceptualization.

Our formal ontology of properties is part of a methodology for *ontology-driven conceptual analysis* which combines the established tradition of *formal ontology* in Philos-

ophy with the needs of information systems design [4]. An more detailed overview of the methodology can be found in [7].

2 Background

This section provides an overview of previous work, intuitive descriptions of the basic meta-properties used to form our ontology of properties, and a discussion of related notions from other information systems fields.

2.1 Previous Work

The need of distinguishing among different kinds of property is recognized only sporadically in the vast literature on knowledge representation, knowledge engineering, database conceptual modeling, and object oriented modeling.

We briefly discuss in this section a few papers belonging to the knowledge engineering and knowledge representation area. A more complete treatment of previous and related work can be found in [7].

Uschold and Gruninger [17] describe their methodology as a skeleton, acknowledge places in which "flesh" needs to be added. In our experience, one such place is in the area of organizing principles for taxonomies: there are in general a multitude of ways to represent the same knowledge, and there exists very little guidance for judging when one approach is better than another.

Some effort to accomplish this was made several years ago by the IDEF group in establishing the IDEF5 ontology capture method [1]. Of particular relevance to this paper, the IDEF5 method attempts to clarify the difference between *kinds*, *classes*, *types*, *attributes*, and *properties*. The specified differences, however, become vague and confused in a number of places. For example,

> "[Kinds] should not be identified with types or classes. [They share characteristics that are, however] distinguishing features of what are typically called *properties*. Because properties are already a part of [IDEF5], it will … be convenient to take kinds to be properties of a certain distinguished sort." (p. 16).

This definition still leaves completely open the question of *how* to distinguish kinds from other properties. The authors are clearly aware of subtle differences here, but did not precisely specify what those differences are. We have attempted to do this by first identifying the formal tools required to make such distinctions, and placing these distinctions within our methodology.

A first attempt to draw some formal distinctions among properties for knowledge engineering purposes was made in [2], and later in [3]. The present work can be seen as a radical extension and refinement of the latter paper, with a better account on the underlying philosophical notions (especially identity), a complete combinatorial analysis of the space of property kinds, and more emphasis on the impact of these distinctions on a general methodology for ontological analysis.

One particular kind of property we discuss here, namely *roles*, is discussed in great detail in a recent paper [15].

2.2 The Basic Notions

At the core of our methodology are three fundamental – and yet intimately related - philosophical notions: *identity*, *unity*, and *essence* (a fourth notion, *dependence*, will be discussed later). The notion of identity we adopt 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. It fits therefore the paradigm of *descriptive metaphysics* [16], whose goal is to provide a framework in which the world *as perceived by us* can be analyzed and described. Despite its fundamental importance in Philosophy, it 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 them together (not involving anything else). 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 [6], and further development of meta-properties based on unity which are used by other parts of our methodology can be found in [8].

Finally, it is important to note that our identity judgements ultimately depend on our *conceptualization* of the world [4]. 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.* For example, the decision as to whether a cat remains the same cat after it loses its tail, or whether a statue is identical with the marble it is made of, are ultimately the result of our sensory system, our culture, etc. The aim of the present analysis is to clarify the formal tools that can both make such assumptions explicit, and reveal the logical consequences of them. 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?

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.

2.3.1 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. 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?"

2.3.2 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 [18]. 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 (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."

2.3.3 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 paper.

Understanding this subtle difference first requires understanding the difference between what we will call *intrinsic* and *extrinsic* properties. 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.

Primary (and candidate) keys and overloaded equality operators are typically based on the latter kind of extrinsic properties 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. Further-

more, 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 this information would 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.

Our analysis relies on certain fairly standard conventions and notations in logic and modal logic, which are described in more detail in [8]. It is important to note that our use of *meta-properties* does not require second-order reasoning, and this is also explained further in [8].

We shall denote primitive meta-properties by bold letters preceded by the sign "+", "-","¬" or "~" which will be described for each meta-property. We use the notation $\phi^{\mathbf{M}}$ to indicate that the property ϕ has the meta-property **M**.

3.1 Rigidity

A rigid property was defined in [3] as follows:

Definition 1 A *rigid property* is a property that is essential to *all* its instances, i.e. $\forall x\, \phi(x) \rightarrow \Box \phi(x)$.

from this it trivially follows through negation that

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

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. This notion was later refined in [4]:

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

Definition 4 A *semi-rigid property* is a property that is non-rigid but not anti-rigid.

Rigid properties are marked with the meta-property **+R**, anti-rigid with **~R**, non-rigid with **-R**, semi-rigid with **¬R**.

The notion of anti-rigidity was added to gain a further restriction. The **~R** meta-property is subsumed by **–R**, but is stronger, as the former constrains all instances of a property and the latter, as the simple negation of **+R**, 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.

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*, and we furthermore may assert $STUDENT^{\sim\mathbf{R}}$. In simpler terms, an instance of *STUDENT* can cease to be a student but may not cease to be a person.

3.2 Identity

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

$$\phi(x) \wedge \phi(y) \rightarrow (\rho(x, y) \leftrightarrow x = y) \tag{1}$$

Since identity is an equivalence relation, it follows that ρ restricted to ϕ must also be an equivalence relation. 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 (1).

As discussed in more detail elsewhere [6], the above formulation has some problems, in our opinion. The first problem is related to the need of 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. It would not supply therefore a "global" condition for identity, satisfying (1) only as a sufficient condition, not as a necessary one.

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 introduce below a notion of identity conditions that have the following characteristics: i) they can only be supplied by rigid properties; ii) they reformulate the ρ relation above in terms of a formula that explicitly takes two different times into account, allowing the distinction between synchronic (same time) and diachronic (different times) identity; iii) they can be only sufficient or only necessary.

Definition 5 A rigid property ϕ *carries the necessary IC* $\Gamma(x,y,t,t')$ if Γ contains x,y,t,t' as the only free variables, and:

$$\neg\forall xytt'(\Gamma(x,y,t,t') \leftrightarrow x{=}y) \tag{2}$$

$$\mathrm{E}(x,t) \wedge \phi(x,t) \wedge \mathrm{E}(y,t') \wedge\ \phi(y,t') \wedge x{=}y \rightarrow \Gamma(x,y,t,t') \tag{3}$$

$$\neg\forall xy(\mathrm{E}(x,t) \wedge \phi(x,t) \wedge \mathrm{E}(y,t) \wedge \ \phi(y,t') \rightarrow \Gamma(x,y,t,t')) \quad (4)$$

Definition 6 A rigid property ϕ *carries the sufficient IC* $\Gamma(x,y,t,t')$ if Γ contains x,y,t,t' as the only free variables, satisfies (2), and:

$$\mathrm{E}(x,t) \wedge \phi(x,t) \wedge \mathrm{E}(y,t') \wedge \phi(y,t') \wedge \Gamma(x,y,t,t') \rightarrow x{=}y \quad (5)$$

$$\exists xytt'\ \Gamma(x,y,t,t'). \quad (6)$$

In the formulas above, E is a predicate for *actual existence* at time t (see [8] for further clarification of our usage, which is based on [9]), (2) guarantees that Γ is bound to identity under a certain sortal, and not to arbitrary identity, (4) is needed to guarantee that the last conjunct in (3) is relevant and not tautological, and (6) ensures that Γ is not trivially false.

ICs are "inherited" along a hierarchy of properties, in the sense that, if $\phi(x) \rightarrow \varphi(x)$ and, for example, Γ is a necessary IC for φ, then (3) above will hold for ϕ replacing φ.

Definition 7 A non-rigid property *carries* an IC Γ iff it is subsumed by a rigid property carrying Γ.

Any property carrying an IC is marked with the meta-property **+I** (**-I** otherwise).

Definition 8 A property ϕ *supplies* an IC Γ iff i) it is rigid; ii) it carries Γ; and iii) Γ 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.

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**).

Definition 9 Any property carrying an IC (**+I**) is called a *sortal* [16].

Notice that to recognize that a property is a sortal we are not forced to know *which* IC it carries: as we shall see, distinguishing between sortals and non-sortals is often enough to start bringing order to taxonomies.

3.3 Dependence

The final meta-property we employ as a formal ontological tool is based on the notion of dependence. This is a very general notion, whose various forms and variations are discussed in detail in [14]. We shall introduce here a specific kind of dependence, based on Simons' *notional dependence:*

Definition 10 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:

$$\forall x\, \Box\, (\phi(x) \rightarrow \exists y\, \psi(y) \wedge \neg P(y, x) \wedge \neg C(y, x)) \quad (7)$$

The part and constituent relations are discussed further in [8]. An externally dependent property is marked with the meta-property **+D** (**-D** otherwise).

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).

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).

4 Constraints and Assumptions

Let us now discuss the constraints that follow from our definitions, which are largely overlooked in many practical cases [5]. In the following, we take ϕ and ψ to be arbitrary properties.

$$\phi^{\sim \mathbf{R}} \text{ can't subsume } \psi^{+\mathbf{R}} \tag{8}$$

This constraint follows immediately from Definitions 1-3. As we shall see, this means that if $PERSON^{+\mathbf{R}}$ and $AGENT^{\sim \mathbf{R}}$, the latter cannot subsume the former.

$$\phi^{+\mathbf{I}} \text{ can't subsume } \psi^{-\mathbf{I}} \tag{9}$$

$$\text{Properties with incompatible ICs are disjoint.} \tag{10}$$

(9) follows immediately from our definitions, while (10) deserves some comment. An important point is the difference between *different* and *incompatible* ICs, related to the fact that they can be inherited and specialized along taxonomies. Consider the domain of abstract geometrical figures, for example, where the property *POLYGON* subsumes *TRIANGLE*. A necessary and sufficient IC for polygons is, "Having the same edges and the same angles". On the other hand, an *additional* necessary and sufficient IC for triangles is, "Having two edges and their internal angle in common" (note that this condition is only-necessary for polygons). So the two properties have *different* ICs (although they have one IC in common), but their extensions are not disjoint. On the other hand, consider *AMOUNT OF MATTER* and *PERSON*. If we admit mereological extensionality for the former but not for the latter (since persons can replace their parts), they have *incompatible* ICs, so they must be disjoint (in this case, we can't say that a person is an amount of matter).

$$\phi^{+\mathbf{D}} \text{ can't subsume } \psi^{-\mathbf{D}} \tag{11}$$

This constraint trivially follows from our definitions.

Finally, we make the following assumptions regarding identity, adapted from [12]:

- *Sortal Individuation.* Every domain element must instantiate some property carrying an IC (**+I**). In this way we satisfy Quine's dictum "No entity without identity" [13].

- *Sortal Expandability*. If two entities (instances of different properties) are the same, they must be instances of a single property carrying a condition for their identity. In other words, every entity must instantiate at least one rigid property.

5 Property Kinds

We now explore the various combinations of meta-properties discussed in the previous section in order to characterize some basic kinds of properties that usually appear in taxonomies.

5.1 A systematic analysis

Analyzing properties based exclusively on the meta-properties discussed in the previous section gives us 24 potential categories (**I, O, D** are boolean, **R** partitions into three cases, **+R**, **~R**, **¬R**). Since **+O**→**+I** and **+O**→**+R** we reduce the number to 14, shown in Table 1, that collapse into the 8 relevant classes of properties discussed below. Each class is labelled with what we consider as the prototypical kind of property belonging to that class. In some cases (for the non-rigid properties), these labels may not be precise, in the sense that further investigation is needed to understand the nature of non-prototypical properties belonging to a certain class.

Table 1: Formal ontological property classifications.

+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	
-O	**-I**	**~R**	**-D**	Attribution	
		¬R	**+D**		
			-D		
+O	**-I**			incoherent	
	+I	**~R**			
		-R			

The taxonomic structure of these classifications is shown in Figure 1. At the top level (the left), we distinguish between *sortal* and *non-sortal* properties, based on the presence or absence of ICs (the meta-property **+I**). *Roles* group together anti-rigid, dependent properties (**~R+D**), and split into *formal roles* (**-I**) and *material roles* (**+I**). Sortals

are divided into *rigid* (**+R**) and *non-rigid* (**-R**), and non-rigid sortals have a further specialization for *anti-rigid* (**~R**). This taxonomy refines and extends the work presented in [2] and [3].

The next sections describe the meta-properties of each of the classes above, as well as the intuitive definition, where properties of that kind should appear in a taxonomy (see Figure 2), and examples of the property kind.

5.1.1 Categories

Categories are properties that are rigid but do not carry identity. Since they can not be subsumed by sortals, categories are normally the highest level properties in an ontology. They carve the domain into useful segments, and they are often primitive, in the sense that no necessary and sufficient membership conditions can be defined for them.

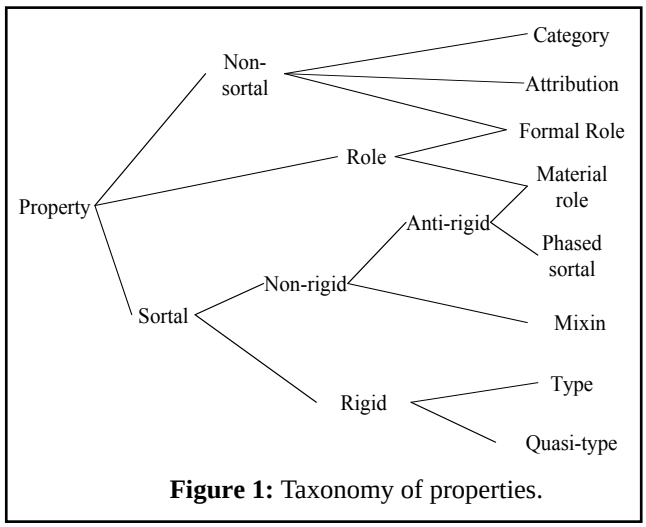

Figure 1: Taxonomy of properties.

According to our constraints, categories can be subsumed by other categories and attributions, and they can subsume any other kind of property. In our experience, categories tend naturally to form a tree. We recommend that at least the topmost categories be disjoint. The archetypal category may be *ENTITY*, other examples may be *CONCRETE ENTITY* and *ABSTRACT ENTITY.*

5.1.2 Types

Types are rigid properties that supply their own identity. They are the most important properties in an ontology, being the only ones that *supply* identity (**+O**), and as a consequence of the Sortal Individuation assumption, *every domain element must instantiate at least one type*. Intuitively, types should be used to represent the main properties in an ontology – not the highest level nor the lowest, simply the properties that will be used the most when describing the domain.

Types can only be subsumed by categories, other types, quasi-types, and attributions. They can subsume any kind of sortal property, and can not subsume any non-sortal properties. We recommend that types by subsumed by at least one category.

Types in general should represent the major properties in an ontology, in our methodology we recommend starting by enumerating the types in a system, since again they should account for every entity. Examples may be *PERSON*, *CAT*, and *WATER*.

Top-types are types that are directly subsumed by categories, and are therefore the highest level (most general) properties that supply identity. We recommend that top-types be subsumed *only* by categories, and furthermore that all top-types be disjoint to the degree possible. Examples may be *LIVING-BEING* or *AMOUNT-OF-MATTER*.

5.1.3 Quasi-Types

Quasi-types are sortals that do not *supply* identity, but nevertheless carry it and are rigid. They often serve a highly organizational purpose by grouping entities based on useful combinations of properties that do no affect identity. Often they tend to introduce new necessary and sufficient *membership* conditions (see the Related Notions section).

Quasi-types can be subsumed by categories, types, attributions, and mixins, and they must be subsumed by at least one type (in order to inherit identity). Quasi-types can subsume any sortal property, and can not subsume non-sortal properties. Like types, we do not recommend subsuming quasi-types by mixins, and recommend minimizing the subsumption of quasi-types by attributions. Examples may be *INVERTEBRATE-ANIMAL*, or *HERBIVORE*.

5.1.4 Backbone Properties

Collectively, the rigid properties in an ontology (categories, types, and quasi-types) form what we call the *backbone taxonomy*. This is of high organizational importance in any ontology, since it identifies the properties that can not change. Backbone properties are also considerable value in understanding a ontology, as they form a subset of all the properties in the ontology and carry a relevant structural information. The backbone carves the domain into useful segments through the categories, identifies every kind of entity in the domain through the types, and contains the most useful groupings of entities through the quasi-types.

5.1.5 Formal Roles

In general, roles are properties expressing the *part played* by one entity in an event, often exemplifying a particular relationship between two or more entities. All roles are anti-rigid and dependent (compare this with [2]). In addition, *formal* roles do not carry identity, and intuitively represent the most generic roles that may form the top level of role hierarchies. For example, the property of being the *PATIENT* of an action is a formal role, since there are no common identity criteria for recipients in general (they may be objects, people, etc.).

Formal roles can be subsumed only by other formal roles, attributions, or categories, and can subsume any non-rigid dependent property, therefore dependent attributions, dependent mixins, and material roles. We recommend that formal roles be used only to organize role taxonomies, i.e. that they not subsume mixins or attributions. Examples include *PATIENT* and *INSTRUMENT*.

To capture our intuitions about formal roles (and roles in general) we may need more than the simple combinations of meta-properties presented here. There might be properties like *BEING LOVED BY JOHN* that seem to belong to the same class as formal roles, without sharing their intuitions. A precise characterization of roles is probably still an open issue (see [2], [15]).

5.1.6 Material Roles

Material roles are anti-rigid and dependent, but inherit identity conditions from some type. Material roles represent roles that are constrained to particular kinds of entities. Intuitively, when a property is recognized to be a role, there should be some event that the role corresponds to.

Material roles can be subsumed by anything, and must be subsumed by at least one type (to inherit identity). They can subsume other material roles, and dependent mixins. We recommend that material roles only subsume other material roles, and that they be subsumed only by roles and backbone properties.

The prototypical material role is *STUDENT*, which would be subsumed by *PERSON* and corresponds to the event *enroll*, other examples may be *MARRIED*, and *FOOD*.

5.1.7 Phased Sortals

Phased sortals [19] are an interesting kind of property that come from combining a requirement for carrying identity with anti-rigidity and independence. Although they do not supply a *global* IC, they supply a *local* IC, corresponding to a certain temporal phase of their instances. Intuitively, they account for entities which naturally, yet fundamentally, change some of their identity criteria over time and in discrete phases. For example, an individual may at one time be a *CATERPILLAR* and at another time be a *BUTTERFLY*. Some local ICs change across these phases, but it is still the same entity and this fact should be reflected in some global ICs.

Phased sortals can be subsumed by anything independent, and can subsume anything non-rigid. According to the Sortal Expandability principle, we have that phased sortals must be subsumed by a type, because it must be possible to determine that they are the same entity at these two times. We recommend that phased sortals be subsumed by backbone properties and that they subsume other phased sortals and material roles. Furthermore we strongly recommend that all the phases of a phased sortal be subsumed by a type or quasi-type that subsumes only them.

Phased sortal properties should never appear alone, each phased sortal must have at least one other phase into which it changes, but note that this does not make it dependent –the properties for each phase will not be instantiated by different entities.

The prototypical examples of phased sortals are *CATERPILLAR* and *BUTTERFLY*. True phased sortals seem rare outside of biology, and often properties classified as phased sortals are single properties into which multiple meanings have been collapsed (see for instance the example of *COUNTRY* discussed in (7), which might be–mistakenly–classified as **~R** thinking that a region may become a country and cease to be a country, while a further analysis reveals that geographical regions and countries are disjoint–although related–entities).

Other properties that belong to the same class as phased sortals might be those resulting from a conjunction of attributions and types, like for instance *RED APPLE:* despite *RED* is usually conceived as semi-rigid (since it may be essential for some instances but not for all), it seems plausible to assume that its restriction to *APPLE* becomes anti-rigid (because every red apple might become brown, for instance). In this case, *RED APPLE* is **~R +I -D**, being therefore a phased sortal.

5.1.8 Attributions

Attributions are the most relevant example of non-sortal properties that are either semi-rigid, or anti-rigid and independent. They intuitively represent values of *attributes* (or *qualities*) like *color, shape,* etc.

The possibility exists that attributions should be always anti-rigid, but we have left it open for now, pending further analysis of cases where types have attributions as essential properties. One might say that e.g. instances of the type *HAMMER* necessarily have the property *HARD*, whereas other types such as *SPONGE* have the property conditionally (a dry sponge is hard, a wet sponge is soft).

Attributions can subsume anything, and can be subsumed by any non-sortal properties. We recommend that attributions subsume only mixins, discussed below, or other attributions, and that they be subsumed only by categories.

Examples include *RED, TRIANGULAR,* and *MALE* (assuming that it is possible to change sex).

5.1.9 Mixins

We generically call *mixins* all the properties that carry identity and are semi-rigid. These properties intuitively represent various combinations (disjunctions or conjunctions) of rigid and non-rigid properties.

Mixins can be subsumed by anything, and can subsume any sortal property. They must be subsumed by at least one sortal. We recommend that mixins not subsume rigid properties. In a sense, mixins should “hang off” the backbone.

Mixins are a difficult kind of property because they are so weakly constrained by our meta-properties. For example, the property *CAT-OR-WEAPON*, subsuming the type *CAT* and the role *WEAPON*, is semi-rigid; some of its instances (instances of *CAT*) are necessarily so, others (instances of *WEAPON* and not *CAT*) are not. We strongly discourage the use of these artificial properties, and in general recommend minimizing the use of mixins. While they may seem useful in large ontologies for organization, we have found unrestrained proliferation of this kind of property to generate confusion more than order.

6 Methodology

Our methodology at the moment focuses mainly on precisely describing properties and clarifying their taxonomic structure. One result of this analysis is what we believe to be “cleaner” taxonomies. In this section we briefly discuss the part played by the formal ontology of properties in the methodology, and then present a short example that uses our meta-property analysis to “clean” a taxonomy.

6.1 The role of property kinds

In general, the ideal structure of a clean taxonomy based on our property kinds is shown in Figure 2. The intuition presented here is that non-rigid properties should "hang off" the backbone taxonomy, and should not subsume anything in the backbone. This makes it possible to easily view the rigid properties without the non-rigid ones, offering a simplified view that still describes every entity in the domain. This structure is not always possible to achieve strictly, but approaching it is desirable.

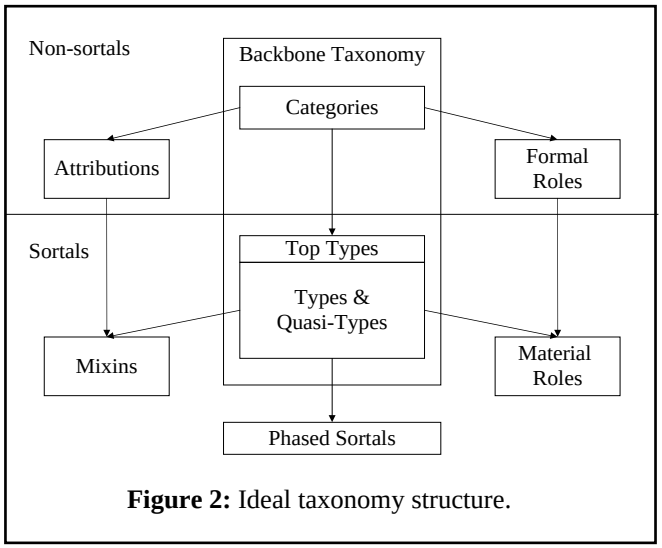

Figure 2: Ideal taxonomy structure.

In addition to providing this idealized structure, our formal ontology of properties also adds to 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-independent description of what part that kind of property should play in an ontology.

7 Conclusions

We have presented here the basic steps of a methodology for ontology design founded on a formal ontology of properties, which is itself built on a core set of meta properties. These meta-properties are formalizations of the basic notions of identity, rigidity, and dependence. We have seen how a rigorous analysis based on these notions offers two main advantages to the knowledge engineer:

- It results in a cleaner taxonomy, due to the semantic constraints imposed on the *is-a* relation;
- The backbone taxonomy is identified.
- It forces the analyst to make ontological commitments explicit, clarifying the intended meaning of the concepts used and producing therefore a more reusable ontology.

8 Acknowledgments

We are indebted to Bill Andersen, Massimiliano Carrara, Pierdaniele Giaretta, Dario Maguolo, Claudio Masolo, Chris Partridge, and Mike Uschold for their useful comments on earlier versions of this paper.

References

1. Benjamin, P. C., Menzel, C. P., Mayer, R. J., Fillion, F., Futrell, M. T., deWitte, P. S., and Lingineni, M. 1994. IDEF5 Method Report. Knowledge Based Systems, Inc., September 21, 1994.
2. Guarino, N. 1992. Concepts, Attributes and Arbitrary Relations: Some Linguistic and Ontological Criteria for Structuring Knowledge Bases. Data & Knowledge Engineering, 8(2): 249-261.
3. Guarino, N., Carrara, M., and Giaretta, P. 1994. An Ontology of Meta-Level Categories. *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*. Morgan Kaufmann.
4. Guarino, N. 1998. Some Ontological Principles for Designing Upper Level Lexical Resources. *Proceedings of LREC-98.*
5. Guarino, N. 1999. The Role of Identity Conditions in Ontology Design. In *Proceedings of IJCAI-99 workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*. Stockholm, Sweden, IJCAI, Inc.: 2-1 2-7.
6. Guarino, N., and Welty, C. 2000a. Identity, Unity, and Individuality: Towards a Formal Toolkit for Ontological Analysis. In *Proceedings of ECAI-2000.* IOS Press, Amsterdam. Available from http://www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html.
7. Guarino, N., and Welty, C. 2000b. Ontological Analysis of Taxonomic Relationships. In *Proceedings of ER-2000: The Conference on Conceptual Modeling.* Available from http://www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html.
8. Guarino, N., and Welty, C. 2000c. Towards a methodology for ontology-based model engineering. In *Proceedings of the ECOOP-2000 Workshop on Model Engineering*. Available from http://www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html.
9. Hirst, G. 1991. Existence Assumptions in Knowledge Representation. *Artificial Intelligence*, 49: 199-242.
10. Huitt, R., and Wilde, N. 1992. Maintenance Support for Object-Oriented Programs. *IEEE Transactions on Software Engineering*. **18**(12).
11. Lewis, D. 1983. New Work for a Theory of Universals. *Australasian Journal of Philosophy*, **61**(4).
12. Lowe, E. J. 1989. *Kinds of Being. A Study of Individuation, Identity and the Logic of Sortal Terms.* Basil Blackwell, Oxford.
13. Quine, W. V. O. 1969. *Ontological Relativity and Other Essays*. Columbia University Press, New York, London.
14. Simons, P. 1987. *Parts: a Study in Ontology.* Clarendon Press, Oxford.
15. Steimann, F. 2000. On the Representation of Roles in OBject-Oriented and Conceptual Modelling. *Data and Knowledge Engineering* (to appear).
16. Strawson, P. F. 1959. *Individuals. An Essay in Descriptive Metaphysics.* Routledge, London and New York.
17. Uschold, M. and Gruninger, M. 1996. Ontologies: Principles, Methods and Applications. *The Knowledge Engineering Review*, 11(2): 93-136.

18. 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.
19. Wiggins, D. 1980. *Sameness and Substance.* Blackwell, Oxford.