Ontology Design and Reuse with Conceptual Roles

Jakob Henriksson[1], Michael Pradel[1], Steffen Zschaler[1], and Jeff Z. Pan[2]

[1] Lehrstuhl Softwaretechnologie, Fakultät Informatik,
Technische Universität Dresden, Germany
{jakob.henriksson|steffen.zschaler}@tu-dresden.de,
michael@binaervarianz.de
[2] Department of Computing Science, University of Aberdeen, UK
jpan@csd.abdn.ac.uk

Abstract. Ontologies are already used in the life sciences and the Semantic Web, but are expected to be deployed in many other areas in the near future—for example, in software development. As the use of ontologies becomes commonplace, they will be constructed more frequently and also become more complex. To cope with this issue, modularization paradigms and reuse techniques must be defined for ontologies and supported by ontology languages. In this paper, we propose the use of *roles* from conceptual modeling for this purpose, and show how they can be used to define ontological reuse units and enable modularization. We present role-based ontologies as an extension of standard ontologies and define their semantics through a reduction to standard Description Logics, such that existing reasoners can be used.

1 Introduction

In conceptual modeling it has long been known that there is a fundamental distinction between different kinds of concepts: some stand on their own (e.g. *Person*), while others depend on the existence of some other concept (e.g. *Borrower*, who must be related to the borrowed item). Making this distinction explicit is favored in the role modeling community (see e.g. [17,18] and references therein), with successful applications—for example, in object-oriented programming [6]. In role modeling, concepts that can stand on their own are called *natural types*, while dependent concepts are called *role types*. Even though considered an important and fundamental conceptual differentiation, current ontology languages, such as OWL DL [11], do not support it (nor does the OWL 2 working draft [2]). The Description Logics (DLs) community—providing much of the foundations for OWL DL—is however aware of the differentiation and refers to role types as *relationship-roles* [1].[3] The DL handbook even supports the idea for the ontology development process by encouraging its readers to "distinguish independent concepts from relationship-roles" [1, p. 379].

This research has been co-funded by the European Commission within the 7th Framework Programme project MOST number 216691 (cf. http://most-project.eu).

[3] The DL community uses the term 'role' for binary properties. To avoid confusion with conceptual roles (relationship-roles) we will use the term *dl-role* to describe the former construct.

Distinguishing different kinds of concepts is not only important for a better understanding of the modeled domain, but also for ontology reuse. This second application of role types has—to the best of our knowledge—never been investigated by the ontology community. Related role types and their relationships form abstraction units that can be studied and defined on their own. Such abstraction units are traditionally called *role models*. As role models often transcend domains, they can be reused in different ontologies.

Consider, for example, the simple role model in Figure 1. It describes the role types *Producer*, *Product*, and *Consumer*, and their relationships.

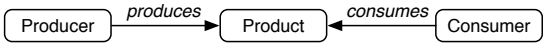

Figure 1. A simple role model describing three related role types and their relationships.

This role model could be integrated in ontologies covering topics as different as foods, wines, consumer electronics, or car dealerships. For example, a consumer electronics company modeling their infrastructure and their business processes can (re)use the generic description captured in the role model in Figure 1.

By being domain-independent units, role models have something in common with upper-level ontologies. Upper-level ontologies, however, are mainly used for integration purposes and reside "above" domain ontologies. Role models serve a different purpose, come with a different reuse approach and—as we will see—essentially reside "below" domain ontologies.

Role modeling can be seen as a design methodology, or process, allowing to break a larger description into smaller, reusable units—the role models. But to successfully deploy design processes for reuse, the underlying languages must support them. To quote Kiczales et. al. [8]: "Software design processes and programming languages exist in a mutually supporting relationship." Clearly 'programming languages' can here be substituted with 'ontology languages.' They go on to explain that the design processes allow to break a system down into smaller and smaller (reusable) units, and that "a design process and a programming language work well together when the programming language provides abstraction and composition mechanisms that cleanly support the kinds of units the design process breaks the system into." It would hence be advantageous—from a reuse point of view—if ontology languages supported the definition of role models and provided constructs for composing role models into complete ontologies.

In this paper we investigate how explicitly supporting role modeling in ontology languages can provide an important and little investigated reuse opportunity—in form of role models. We also discuss the consequences of supporting the underlying role modeling semantics. The contributions of this paper are: (1) We demonstrate the viability of role models as ontological reuse units. (2) We propose a formalization for what role modeling means for current ontology languages. (3) We explain the consequences of introducing the semantics of role modeling into ontology languages. That is, the possible reasoning effects the ontology engineer has to be aware of. (4) We describe how the role modeling semantics can be realized by reducing role modeled ontologies into

ontologies expressed in standard ontology languages. This enables the reuse of reasoning engines. Preliminary results of our work have been presented in [13].

The remainder of the paper is structured as follows: We begin in the next section by giving some background on role modeling and DLs. In Section 3 we discuss an example of how role models can provide reuse opportunities, before providing a formal reduction of our concepts to an underlying DL in Section 4.

2 Background

This section gives background knowledge the rest of the paper is based on. First, we introduce role modeling as it is known from conceptual and software modeling. Then, we discuss Description Logics, the formal underpinning of many current ontology languages.

2.1 Role Modeling

In modeling, types (or concepts) abstract over sets of individuals. Role modeling at its core argues for the existence of two inherently distinguishable abstractions: *natural types* and *role types*, a terminology first introduced by Sowa [16]. Intuitively, natural types describe the part of individuals that are essential to their identity while role types describe accidental or temporal relations to other individuals.

A common example is the natural type *Person* and its associated role type *Actor*. In this case, a person is said to *play the role* of an actor. Along with being an actor comes, for example, giving performances led by stage managers and attending rehearsals led by directors. Hence, in the role of being an actor one stands in certain relations to individuals of other role types such as *Director*.

Guarino defines natural types and role types using the notions of *founded types* and *semantically rigid types* [3]. A type is founded if all of its individuals have to be related to an individual of another type, where the relation is not part-of. For example, one could say that an actor necessarily has to be related to a director in order to be an actor. A type is semantically rigid, if it contributes to the identity of its individuals. For instance, the name and date of birth of a person are part of its identity. Hence, an individual cannot drop the type *Person*, while ceasing to be an *Actor* is possible. Using these two notions, we can define natural types and role types:

- A *natural type* is non-founded and semantically rigid.
- A *role type* is founded and semantically non-rigid.

Although the notion of roles seems intuitively clear, different definitions exist in the literature. Steimann summarizes the fifteen most common characteristics the research community associates with roles in object-oriented and conceptual modeling [17]. As an ontology describes a static view of the world, those referring to dynamic aspects observable in software systems are not relevant in this paper. Among the remaining, we consider the following to be the most fundamental (each of Steimann's role features is referred to as 'Sx'):

S1 "*A role comes with its own properties and behaviour.* This basic property suggests that roles are types."

S2 "*Roles depend on relationships.* As suggested by the work of Sowa and Guarino, a role is meaningful only in the context of a relationship."

S3 "*An object may play different roles simultaneously.* This is one of the most broadly accepted properties of the role concept."

S14 "*An object and its roles share identity.* In the object-oriented world this entails that an object and its roles are the same."

Roles alone are beneficial to separate inherent and accidental characteristics of individuals. Encapsulating several related roles into a *role model* is another important aspect focused on in this paper. Results from the object-oriented software community [5,14] show that role models are an interesting unit of abstraction for mainly two reasons: First, role models focus on one specific concern of a domain, and hence, help in separating concerns. Second, role models are often reusable across domain boundaries because they can describe relations between individuals on a more general level than natural types can. We will focus on the second point, reuse of role models, in this work and show how role models serve as an ontological reuse unit.

An often discussed question is the relation of natural types and role types. Intuitively, natural types are related to role types via the "can play role" relationship, but the question is what semantics to associate. We will use the $N \rhd R$ notation for the "can play role" relation in this paper, where N is a natural type and R a role type. Sowa originally proposed to use the subsumption relationship ("IS-A") to explain $\rhd$, such that:

$$N \rhd R \;\equiv\; R \sqsubseteq N$$

where $\sqsubseteq$ represents the IS-A relationship. This interpretation is quite intuitive and works remarkably well, but not in all situations as we discuss in Section 4.3.

An alternative way of representing roles are separate individuals that are attached to individuals of natural types in some way. However, that contradicts role feature S14, since a role-playing individual would be split into at least two separate individuals. For a detailed discussion, the reader is referred to [17].

2.2 Description Logics and OWL

Description Logics (DLs) are a family of knowledge representation formalisms, where most members are sub-languages of first-order logics. DLs are used to capture the important *concepts* and relations (*roles* in DL parlance) between individuals of the modeled domain. We will refer to (binary) relations in DL as *dl-roles* to distinguish them from the conceptual role types. Concepts and dl-roles can be described by complex *concept* (resp. *dl-role*) *descriptions* using the construction operators available in the particular DL.

The most widely used DL is the one underlying the ontology language OWL DL. To simplify the presentation, we do not cover datatypes in this paper. An OWL DL *interpretation* is a tuple $I = (\Delta^I, \cdot^I)$ where the individual domain Δ^I is a nonempty set of individuals, and $\cdot^I$ is an individual interpretation function that maps (i) each

individual name o to an element $o^I \in \Delta^I$, (ii) each concept name A to a subset $A^I \subseteq \Delta^I$, and (iii) each dl-role name RN to a binary relation $RN^I \subseteq \Delta^I \times \Delta^I$.

Valid OWL DL concept descriptions are defined by the DL syntax:

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \{o\} \mid \exists R.C \mid \forall R.C \mid \geqslant mR \mid \leqslant mR$$

The interpretation function $\cdot^I$ is extended to interpret $\top^I = \Delta^I$ and $\bot^I = \emptyset$. The concept $\top$ ($\bot$) is called *Thing* (*Nothing*) in OWL. The interpretation function can further be extended to give semantics to the remaining concept and dl-role descriptions (see [12] for details).

An OWL DL ontology consists of a set of *axioms*, including concept axioms, dl-role axioms and individual axioms.[4] A DL knowledge base consists of a TBox, an RBox and an ABox. A *TBox* is a finite set of concept inclusion axioms of the form $C \sqsubseteq D$, where C, D are concept descriptions. An interpretation I satisfies $C \sqsubseteq D$ if $C^I \subseteq D^I$. An *RBox* is a finite set of dl-role axioms, such as dl-role inclusion axioms ($R \sqsubseteq S$). The kinds of dl-role axioms that can appear in an RBox depend on the expressiveness of the ontology language. An interpretation I satisfies $R \sqsubseteq S$ if $R^I \subseteq S^I$. An *ABox* is a finite set of individual axioms of the form $a : C$, called *concept assertions*, or $\langle a, b \rangle : R$, called *dl-role assertions*. An interpretation I satisfies $a : C$ if $a^I \in C^I$, and it satisfies $\langle a, b \rangle : R$ if $\langle a^I, b^I \rangle \in R^I$.

Let C, D be concept descriptions, C is *satisfiable* wrt. a TBox T iff there exist an interpretation I of T such that $C^I \neq \emptyset$; C subsumes D wrt. T iff for every interpretation I of T we have $C^I \subseteq D^I$. A knowledge base Σ is *consistent* (*inconsistent*) iff there exists (does not exist) an interpretation I that satisfies all axioms in Σ.

Many people also use the Manchester OWL syntax [7], which is more user friendly for non-logicians.

3 A Role-Based Design Process for Ontological Reuse

The purpose of this section is to give an intuitive understanding of why role models are reusable entities and how they can be beneficial to ontology engineers. We use a running example to discuss consequences of a role modeling approach to ontology design. Furthermore, we discuss what the benefits of ontological role modeling are from a conceptual modeling point of view, and what the methodology of defining role-based ontologies is. Then, in Section 4, we discuss the semantics of such ontologies.

The examples are written in Manchester Syntax [7], which is not further described here but should be no problem to understand. We extend the syntax for the purpose of defining and composing role models; the keywords of the extended constructs are underlined (and in a different color).

Listing 1.1 shows an ontology that models a faculty, introducing main concepts such as *Professor*, *FacultyMember*, and *PhDStudent*. The faculty is managed by a board which is described in the role model in Listing 1.2. A board consists of board members that elect a chairman.[5] The chairman can appoint one of the members as secretary. The

[4] Individual axioms are called *facts* in OWL.

[5] A 'chairman' is here a person designated to preside over a meeting.

```
Ontology: http://ex.org/Faculty
  ImportRoles: http://ex.org/Board
  Class: FacultyMember
    CanPlay: BoardMember'
  Class: Professor
    SubclassOf: FacultyMember
    CanPlay: ChairMan'
  Class: PhDStudent
    SubclassOf: FacultyMember
  Individual: smith
    Types: Professor, Chairman'
  Individual: mike
    Types: PhDStudent, BoardMember'
```

Listing 1.1. Role-based ontology.

```
RoleModel: http://ex.org/Board
  Role: BoardMember'
  Role: Chairman'
    SubclassOf: BoardMember' and
      electedBy' some BoardMember'
  Role: Secretary'
    SubclassOf: BoardMember'
  ObjectProperty: electedBy'
    Domain: Chairman'
    Range: BoardMember'
  ObjectProperty: appointedBy'
    Domain: Secretary'
    Range: Chairman'
```

Listing 1.2. Role model.

ontology in Listing 1.1 imports the board role model and can so use the concepts it defines. Concepts and properties defined in the role model are marked with ′ to distinguish them from the concepts introduced in the base ontology.

Readers might ask why the board is described in a role model. The reason is that boards have a recognizable structure with a typical set of relationships that hold between entities in that context, regardless of the particular underlying domain. It makes therefore sense to detach the description of the board from the faculty ontology.

The ontology in Listing 1.1 is made up of standard DL constructs, save the *ImportRoles* and *CanPlay* constructs. The meaning of the *ImportRoles* construct is the obvious, making the role model available to the ontology. The *CanPlay* constructs are crucial since they define the relations between the base ontology and the role model. We refer to such connecting statements as *bridge axioms*. The role model in Listing 1.2 makes use of two additional constructs, *RoleModel* and *Role* that have to obvious meaning. The URL of a *RoleModel* can be used to import it using the *ImportRoles* construct.

3.1 Methodology

Role modeling provides a methodology for developing ontologies, a methodology that we claim encourages good design and supports reuse. The following are the intuitive development steps that we propose for constructing a role-based ontology:

1. *Define base concepts.* Define a base ontology that contains the main concepts of the modeled domain. These concepts correspond to natural types of the domain. That is, each concept in the base ontology should be semantically rigid and non-founded. In our example, an ontology modeler would start by defining basic concepts of a faculty, such as *Professor* and *PhDStudent*. Notice that, in a different universe of discourse, *Professor* may itself be a role type (for example, for an underlying natural type *Person*).
2. *Identify role models.* Identify accidental or temporal relationships that individuals, abstracted by the base concepts, may participate in. Then, identify the contexts that

those relationships appear in and what concepts (role types) are involved. Such contexts should be described in separate role models to be integrated into the base ontology.
   (a) If a role model for the desired relationships already exists, it can be reused.
   (b) If no fitting role model exists, then define it. This involves capturing the important role types in the context and defining the relationships between them. There will exist relationships, since role types are by definition semantically non-rigid and founded. For instance, the board role model contains the role types *Chairman'* and *BoardMember'*, which are related by the *electedBy'* property.

   To ensure reusability of role models they have to be self-contained. In particular, each property defined in a role model must have its domain and range restricted to a role type from the same role model. This guarantees that each individual that participates in such a property actually belongs to a role type of the role model.
3. *Define bridge axioms.* Describe how the identified role models should be integrated into the base ontology by defining appropriate bridge axioms. A bridge axiom can bind a role type to a natural type, assert that an individual belongs to a certain role type, or assert two individuals to be connected via a property that is part of a role model. For example, we connect *Professor* and *Chairman'* through a *CanPlay* axiom; the individual *smith* is asserted to be a *Chairman'*.

3.2 Benefits of Separating Role Models and Base Ontologies

We argue that it is beneficial to separate role models from base ontologies during the ontology design process. In particular, following the above methodology brings the following advantages:

- Modularization during development:
  - Base ontology development can focus on the main domain concepts and their hierarchical relations.
  - Role models can be defined, and refined, without necessarily focusing on the domain concepts, because role models typically transcend domains.
  - A role model focuses on a single context and important relationships holding between entities in this context.
- Reuse of role models:
  - As role models concentrate on a single concern, reuse is more likely than with complete ontologies that intermingle different concerns.
  - Role models constitute ontological modules. A base ontology can use role type names and property names of a role model, but not redefine them. Hence, a role model provides an interface, via the names of its role types and properties.

3.3 Reusing Role Models

The role-based ontology in Listing 1.3 demonstrates the reusability of the board role model from Listing 1.2. The role model is being deployed in a different setting, this time in an ontology modeling a company, instead of a university. Because the *concern* which is captured in the role model appears in both domains, it can be reused.

```
Ontology: http://ex.org/Company
  ImportRoles: http://ex.org/Board
  Class: President
    CanPlay: ChairMan'
  Class: VicePresident
    CanPlay: Secretary'
  Class: CompanyAdvisor
    CanPlay: BoardMember'
  Individual: donald
    Types: President, Chairman'
  Individual: jane
    Types: VicePresident, BoardMember'
```

Listing 1.3. Role-based ontology reusing the role model from Listing 1.2.

4 Formalization and Semantics of Ontological Role Modeling

In Section 4.1 we formalize ontological roles and role models. Then, in Sections 4.2 to 4.3, we propose two possible semantics for the role modeling constructs used in the preceding section. The two semantics cover different aspects of role modeling and are realized by mapping role-based ontologies to different DL constructs.

4.1 Formalization of Role-Based Ontologies

We formalize role-based ontologies in three parts, based on the methodology described in Section 3. A base ontology only contains natural types. Role models define role types and their relationships. We use bridge axioms to combine a base ontology with role models into a role-based ontology.

Definition 1 (Base ontology). *A* base ontology *is a finite set of axioms in some DL, capturing concepts that are assumed to correspond to natural types.*

A base ontology is assumed to capture concepts that provide semantic rigidity for individuals of the modeled domain. Naturally, any properties that inherently relate such concepts are also introduced, as are concrete individuals. We do not commit to a particular DL, since this definition is general enough to cover many DLs.

Definition 2 (Ontological role model). *An* ontological role model *is a TBox where each concept name is considered a role type. Each concept name must be "related" to another concept name, either via a dl-role, or by at least one axiom (e.g. a subsumption axiom). All dl-roles must be domain and range restricted to a type from the role model.*

The restriction of "related" concept names prevents role models from being divided into subparts with pairwise disjoint signatures.[6] If such a division is possible, the role model should be split into separate role models. Intuitively, this restriction ensures that a role model only describes one concern.

[6] As pointed out by a reviewer, what "related" means is not formalized. We recognize this formalization as important future work, but here stay with the intuitive notion, as described.

Definition 3 (Role-based ontology). *A* role-based ontology *$\mathcal{O}$ is a triple $\mathcal{O} = (\mathcal{N}, \mathcal{R}, \mathcal{B})$ where $\mathcal{N}$ is a* base ontology*, $\mathcal{R}$ is a finite set of* role models *and $\mathcal{B}$ a finite set of* bridge axioms*. The base ontology and the role models have pairwise disjoint signatures. The bridge axioms in $\mathcal{B}$ are of the form:*

1. *$N \rhd R$ (*terminological bridge axiom*), or*
2. *$R(a)$ or $S(a,b)$ (*assertional bridge axiom*)*

where N is an arbitrary concept description, and a, b are individuals, in $\mathcal{N}$. R is a concept name (role type), and S a dl-role, in one of the role models in $\mathcal{R}$.

The $\rhd$ symbol reads "can play" and specifies that instances of a natural type can play a role of a certain role type.

To be able to reuse existing tools, most importantly, reasoners, we define the semantics of role-based ontologies via reduction to the underlying DL. Thus, the reduction algorithm unambiguously defines the semantics of role-based ontologies by referring to the already understood model-theoretic semantics of DLs.

4.2 Conjunctive Role Modeling Semantics

The goal is to define a semantics for role-based ontologies that cover desirable properties of role modeling. As a minimal requirement, the semantics should cover the Steimann criteria S1, S2, S3, and S14 described in Section 2. More importantly, we need to account for the differentiation between natural and role types according to the distinction made by Guarino [3]. That is, natural types are semantically rigid, while role types are not, and do not provide identity for its instances. We will address this by acknowledging that individuals cannot only be instances of role types. This suggests that role types that are not explicitly related to some natural type should—by definition—be unsatisfiable (empty in all models of the ontology). Conveniently, unbound role types can easily be detected with standard ontology reasoners. Relations between natural and role types should explicitly be modeled by ontology engineers using terminological bridge axioms (that is, using the *CanPlay* construct).

The semantics of a role-based ontology $\mathcal{O} = (\mathcal{N}, \mathcal{R}, \mathcal{B})$ is here given by a transformation to an ontology $\mathcal{O}'$ in the DL of $\mathcal{N}$ according to the following transformation:

$$\begin{aligned}\mathcal{O}' = {} & \mathcal{N} \cup \mathcal{R} \\ & \cup \{R \sqsubseteq N | N \rhd R \in \mathcal{B}\} \\ & \cup \mathcal{B} \setminus \{N \rhd R | N \rhd R \in \mathcal{B}\} \\ & \cup \{R \sqsubseteq \bot | R \in \mathcal{R} \wedge \neg\exists N : N \rhd R \in \mathcal{B}\}\end{aligned}$$

As can be seen, the translation scheme consists of four steps:

1. *Integration.* The base ontology and the role models (with pairwise disjoint signatures) are combined.
2. *Terminological bridge axioms.* Here we use the semantics proposed by Sowa for realizing the $\rhd$ relationship [15]. That is, if instances of a natural type N can play a role of a role type R, we specify R to be subsumed by N.

```
Ontology: http://ex.org/Faculty
  Class: FacultyMember
  Class: Professor
    SubclassOf: FacultyMember
  Class: PhDStudent
    SubclassOf: FacultyMember
  Class: BoardMember'
    SubclassOf: FacultyMember
  Class: Chairman'
    SubclassOf: BoardMember' and
      electedBy' some BoardMember'
      and Professor
  Class: Secretary'
    SubclassOf: BoardMember' and
                Nothing
  ObjectProperty: electedBy'
    Domain: Chairman'
    Range: BoardMember'
  ObjectProperty: appointedBy'
    Domain: Secretary'
    Range: Chairman'
  Individual: smith
    Types: Professor, Chairman'
  Individual: mike
    Types: PhDStudent, BoardMember'
```

Listing 1.4. Translation of a role-based ontology into the underlying ontology language.

3. *Assertional bridge axioms.* All other bridge axioms are incorporated into $\mathcal{O}'$. That is, all the assertional bridge axioms.
4. *Unbound role types*. Role types that are not related to any natural type through $\rhd$ subsume $\bot$ (*Nothing* in OWL), that is, they are unsatisfiable.

We will illustrate the transformation using the example from Listings 1.1 and 1.2. Applying the transformation yields the ontology in Listing 1.4. This ontology only uses standard ontology constructs. It must be highlighted that the ontology in Listing 1.4 only captures the *meaning* of the ontology units from Listings 1.1 and 1.2. The ontology engineer is never expected to continue working on the "compiled" ontology. The abstractions gained through explicit role modeling are lost in the compilation step, making the resulting ontology more difficult to maintain. At the same time, because the compilation result is expressed in a standard DL, existing ontology reasoners can handle it directly.

For instance, the role type *Secretary'* is not bound to any natural type, and hence, defined as a subtype of *Nothing* ($\bot$). To understand the consequences of this translation, let us consider two scenarios:

1. Assume an additional assertion, stating that *mike* is an instance of *Secretary'*. As *Secretary'* is unbound, and thus, unsatisfiable, the resulting ontology would be inconsistent.
2. Assume another role type relies on instances of *Secretary'*. For instance, *Chairman'* could be a subclass of the concept *(assistedBy' some Secretary')*. As there can be no instances of *Secretary'*, *Chairman'* would also become unsatisfiable.

Obviously, our translation scheme for unbound role types helps in finding situations where role types are misused as natural types. The logical solution to repair our ontology for the two scenarios would be an additional bridge axiom *FacultyMember* $\rhd$ *Secretary'*. In this case, *Secretary'* is no longer unsatisfiable and, as a consequence, the above illustrated inconsistencies do not occur.

Finally, let us come back to the four properties of Steimann from Section 2.1. As we define roles as concepts that can be described with all constructs of the underlying

DL, we fulfill S1. The second property, S2, is also supported, since we require that the context of each role type is captured in its surrounding role model. The second kind of bridge axiom in Definition 3 can be used arbitrarily often, hence, allowing one individual to be an instance of multiple role types (S3). The last property (S14) is supported as we do not represent roles as additional individuals but combine them with natural types using subsumption.

We refer to the above-described semantics as *conjunctive role modeling semantics* because a role type R played by different natural types $N_1, \ldots, N_n$ is interpreted as a subtype of the conjunction of the natural types, that is, $R \sqsubseteq N_1 \sqcap \ldots \sqcap N_n$. This follows immediately from defining $\rhd$ using standard subsumption. While simple, this semantics does not come without problems from a role modeling perspective. We investigate this further in the next section.

4.3 Disjunctive Role Modeling Semantics

Another important role modeling feature from Steimann's overview paper [17] is the following:

S7 "*Objects of unrelated types can play the same role.* Although a fundamental observation [...] it is not acknowledged by all authors."

Although "not acknowledged by all authors", it is a rather intuitive and useful modeling notion. Imagine, for example, that we replace the class *Professor*, from Listing 1.1, with the two classes *FullProfessor* and *AssistantProfessor*. Imagine, furthermore, that they are declared to be disjoint (natural, since you cannot be both). Suppose we want to express that both kinds of professors can be chairmen in a board. Not only is this a natural thing to express, but doing so would also enable us to discuss the properties of a chairman (defined by *Chairman'*), regardless of which kind of professor it is. To do this, we would issue the following bridge axioms:

$$FullProfessor \rhd Chairman'$$
$$AssistantProfessor \rhd Chairman'$$

While the above axioms are intuitive to understand and write, the conjunctive role modeling semantics, based on Sowa's original interpretation of $\rhd$, does not work as we perhaps would like. The reason is that the above gets interpreted as $Chairman' \sqsubseteq FullProfessor \sqcap AssistantProfessor$, which renders $Chairman'$ unsatisfiable. This is the case since the intersection of $FullProfessor$ and $AssistantProfessor$ is necessarily empty, since they are disjoint. In general, the conjunctive role modeling semantics can result in unexpected results when types that are not related via subsumption (e.g $FullProfessor$ and $AssistantProfessor$ in the above example) are related to the same role type via $\rhd$.

To be able to address S7, we provide an alternative semantics by letting role types be subsumed by the *union* of all natural types their are bound to. For the above example, this results in $Chairman' \sqsubseteq FullProfessor \sqcup AssistantProfessor$, which in this case does not make $Chairman'$ unsatisfiable. We call this the *disjunctive role modeling semantics* and its formal realization is as follows:

```
...
Class: FullProfessor
  SubclassOf: FacultyMember
Class: AssistantProfessor
  SubclassOf: FacultyMember
...
Class: Chairman'
  SubclassOf: BoardMember' and
     electedBy' some BoardMember' and
     (FullProfessor or AssistantProfessor)
...
```

Listing 1.5. Translation of a role-based ontology into the underlying ontology language using disjunctive role modeling semantics.

$$\begin{aligned}\mathcal{O}' = \mathcal{N} \cup \mathcal{R} \\ \cup \{R \sqsubseteq N_1 \sqcup \ldots \sqcup N_n | \{N_1, \ldots, N_n\} = \{N | N \rhd R \in \mathcal{B}\}\} \\ \cup \mathcal{B} \setminus \{N \rhd R | N \rhd R \in \mathcal{B}\} \\ \cup \{R \sqsubseteq \bot | R \in \mathcal{R} \wedge \neg \exists N : N \rhd R \in \mathcal{B}\}\end{aligned}$$

The above equations only differ from the corresponding equations for conjunctive semantic in the translation of terminological bridge axioms. Therefore, disjunctive semantics only differs from the conjunctive in cases where several natural types are bound to the same role type. In the case from the previous section with a single concept *Professor* bound to role type *Chairman'*, both semantics are equivalent. In contrast, the two semantics give different results if both *FullProfessor* and *AssistantProfessor* are bound to the role type *Chairman'*. While we obtain an inconsistency with the conjunctive semantics, disjunctive semantics allow for a consistent interpretation. The result is shown in Listing 1.5 (parts that are equal to Listing 1.4 are left out).

The disjunctive role modeling semantics satisfies S7, as well as the previously discussed role modeling requirements. Being able to connect unrelated, possibly disjoint, natural types to the same role type can be valuable from a modeling perspective. However, fulfilling S7 turns out to have a drawback: In contrast to standard DLs, the disjunctive semantics is non-monotonic. A logic is monotonic if adding a new axiom never falsifies assertions that were true before adding the axiom.

Lemma 1. *Ontological role modeling under disjunctive semantics is non-monotonic.*

The reason for this is that adding assertional bridge axioms can redefine previous knowledge. Consider the following role-based ontology $O = (\mathcal{N}, \mathcal{R}, \mathcal{B})$, with:

$$\begin{aligned}\mathcal{N} = \{&FullProfessor \sqcap AssistantProfessor \sqsubseteq \bot, \\ &FullProfessor(smith), AssistantProfessor(jones)\} \\ \mathcal{R} = \{&Chairman' = electedBy' \textbf{ some } BoardMember'\} \\ \mathcal{B} = \{&FullProfessor \rhd Chairman'\}\end{aligned}$$

Based on the disjunctive role modeling semantics we have $O \models \neg Chairman'(jones)$. But adding the bridge axiom $AssistantProfessor \rhd Chairman'$ to $\mathcal{B}$, this is not the case anymore, that is, $O \not\models \neg Chairman'(jones)$. As we have to retract knowledge when adding a bridge axiom, role modeling under the disjunctive semantics is non-monotonic.

5 Related Work

Reuse is an important part in the ontology development process. A structured reuse opportunity is presented by *upper-level ontologies* (e.g. [10]). Upper-level ontologies typically describe generic concepts related to notions such as time, space, matter, and have a high reuse value since they span many different domains. Role models also span different domains, but their reuse usage is different from upper-level ontologies. In general, upper-level ontologies reside "above" the base ontology and reuse is accomplished by declaring a base concept B as a subclass of the upper-level concept U, that is, $B \sqsubseteq U$ in DL syntax. Role models on the other hand reside "below" the base ontology and the reuse relationship is inverted from the upper-level one. That is, reuse is in general accomplished using axioms such as $R \sqsubseteq B$, where R is a role type from a role model. As such, role models are *complementary* to upper-level ontologies. The main incentive for upper-level ontologies is to achieve base ontology integration and reuse is a condition for its success. For role models, reuse *is* the incentive.

The OntoClean methodology proposed by Guarino [4] shares a number of ideas with our approach. The paper describes common misuses of the subsumption concept, for instance, to represent part/whole relations, instantiations, or meta-level relationships, and proposes to identify them using meta-properties for ontological classes. The two basic meta-properties of a class are *essence* and *rigidity*. Properties of a class belong to its essence, if they *must* hold for an instance (in contrast, for example, to properties of a role type, which *can* hold). Rigidity means that the properties of the class must hold for all instances. Our approach relates to this work as essential and rigid classes correspond to natural types, while non-essential and non-rigid classes are role types. Based on the meta-properties, Guarino proposes to impose constraints on subsumption relationships, for instance, forbidding a rigid concept to be subsumed by a non-rigid concept. This constraint is enforced in our approach by translating the terminological bridging axiom into $R \sqsubseteq C$, where a non-rigid concept R is always subsumed by a rigid concept C. Furthermore, OntoClean claims that each ontology has a backbone taxonomy with its rigid classes and their subsumption relationships. Such a backbone taxonomy roughly corresponds to our base ontology that exclusively consists of natural types.

The work in [19] proposes, similarly to us, to discriminate *role concepts* from *basic concepts* to overcome the gap between the recognition of different types of concepts and what is provided in standard ontology languages. The approach builds upon three notions: *role concepts*, equivalent to our role types, *potential players*, which roughly correspond to classes bound to a role type via a *CanPlay* relationship, and *role-holder*, that is, instances actually playing a role. The authors argue for two distinct type hierarchies and emphasize the relation of a role to its context. Furthermore, the paper describes compound roles that are built from primitive roles, realizing ideas that are

similar to roles playing roles as in [17]. In contrast to our approach, the authors implement roles in their own ontology framework, including an ontology language and custom-built tools.[7] Instead, we propose to embed roles into existing languages using syntactic extensions that can be translated into the underlying ontology language.

6 Conclusions and Outlook

Ontologies are increasingly often applied in real-life scenarios, where they are used for modeling large knowledge domains. Perhaps the most prominent example to name here is the Gene Ontology [20] with its almost 25.000 terms (as of January 2008). To successfully develop and maintain such large ontologies, powerful modularization and reuse techniques are required.

In this paper we have discussed the notion of roles as explicit modeling constructs, that has been discussed in the domain of conceptual modeling for some time now, but never really utilized in ontology languages. We have shown how making roles explicit concepts—instead of encoding them implicitly in dl-roles—enables us to encapsulate role models and reuse them in different ontologies, even across domain boundaries. A role model in this context is an ontology consisting of role concepts and relationships between them. We have shown how to construct role-based ontologies from a base ontology, a role model and a set of bridging axioms relating natural concepts from the base ontology to role concepts from the role model via a *CanPlay* relationship. The semantics of this new relationship has been defined by translating role-based ontologies to equivalent ontologies in a standard DL. In this paper, we have discussed two such semantics: *conjunctive semantics* and *disjunctive semantics.* The former is simpler, but does not allow individuals of disjoint natural concepts to play the same role, which may be counter intuitive to ontology designers. The latter semantics allows such situations, but at the cost of a non-monotonic logic. It is at this stage not possible to recommend one of the semantics as canonical—they serve different purposes. We first need to study how role models are used and applied in practice to better understand which semantics is more intuitive to ontology engineers.

In conclusion, we suggest to integrate explicit role concepts with ontology languages, such as OWL, as they offer a unique reuse and modularization opportunity that goes beyond currently available mechanisms for ontology reuse. Role models form *components*; their role types define a component interface enabling to check correct usage of the component—for example, every used role type must be assigned to some base concept. A notion of components that allows checking for correct usage is lacking in today's ontology languages. This gap can be closed by role modeling. Role modeling goes beyond upper-level ontology reuse, as it allows more specific ontologies to be reused and also allows reusing multiple role models within one ontology. Role models are, therefore, an important tool in every ontology designer's tool box.

Of course, more work is needed. We would like to perform a larger case study of applying role models to refactoring, modularizing, and partially reusing a large ontology. Role modeling should be supported by ontology modeling tools that can transform the

[7] `http://www.hozo.jp`

role-based ontologies before applying reasoners. Furthermore, it will be interesting to study other applications of role models in ontologies—for example, where ontologies are used for describing situations in an action calculus [9].

References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. B. Cuenca-Grau and B. Motik. OWL 2 Web Ontology Language: Model-theoretic semantics. W3C Working Draft, 2008. `http://www.w3.org/TR/owl2-semantics/`.
3. N. Guarino. Concepts, attributes and arbitrary relations - Some linguistic and ontological criteria for structuring knowledge bases. *Dat. Knowl. Eng.*, 8(3):249–261, 1992.
4. N. Guarino and C. A. Welty. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, 2002.
5. S. Herrmann. Object Teams: Improving modularity for crosscutting collaborations. In *Net Object Days*, 2002.
6. S. Herrmann. A precise model for contextual roles: The programming language ObjectTeams/Java. *Applied Ontology*, 2(2):181–207, 2007.
7. M. Horridge and P. F. Patel-Schneider. Manchester syntax for OWL 1.1. In *International Workshop OWL: Experiences and Directions (OWLED '08)*, 2008.
8. G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In *European Conference on Object-Oriented Programming (ECOOP '97)*, pages 220–242. Springer, 1997.
9. H. Liu, C. Lutz, M. Milicic, and F. Wolter. Reasoning about actions using description logics with general TBoxes. In *European Conference on Logics in Artificial Intelligence (JELIA '06)*, pages 266–279. Springer, 2006.
10. I. Niles and A. Pease. Towards a standard upper ontology. In *International conference on Formal Ontology in Information Systems (FOIS '01)*, pages 2–9. ACM, 2001.
11. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language semantics and abstract syntax. W3C Recommendation, 2004. `http://www.w3.org/TR/owl-semantics/`.
12. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Technical report, W3C, Feb. 2004. W3C Recommendation.
13. M. Pradel, J. Henriksson, and U. Aßmann. A good role model for ontologies: Collaborations. In *International Workshop on Semantic-Based Software Development*, 2007.
14. T. Reenskaug, P. Wold, and O. Lehne. *Working with Objects, The OOram Software Engineering Method*. Manning Publications Co, 1996.
15. J. Sowa. *Using a lexicon of canonical graphs in a semantic interpreter*, pages 113–137. Cambridge University Press, 1988.
16. J. F. Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley Longman Publishing Co., Inc., 1984.
17. F. Steimann. On the representation of roles in object-oriented and conceptual modelling. *Data Knowledge Engineering*, 35(1):83–106, 2000.
18. F. Steimann. The role data model revisited. *Roles, an interdisciplinary perspective, AAAI Fall Symposium*, 2005.
19. E. Sunagawa, K. Kozaki, Y. Kitamura, and R. Mizoguchi. Role organization model in Hozo. In *International Conference on Managing Knowledge in a World of Networks (EKAW)*, pages 67–81. Springer, 2006.
20. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.