

# A Framework for Organizing Role Concepts in Ontology Development Tool: Hozo

Eiichi Sunagawa, Kouji Kozaki, Yoshinobu Kitamura and Riichiro Mizoguchi

The Institute of Scientific and Industrial Research, Osaka University  
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan  
{sunagawa, kozaki, kita, miz} @ei.sanken.osaka-u.ac.jp

## Abstract

Establishment of computational framework of role concepts contributes effectively to management of instance models because it provides us with a useful policy for treatment of views and contexts related to roles. In our research, we have developed an ontology building environment, which provides a framework for representation of role concepts and their characteristics. In this paper, as an extension of this framework, we present a framework for organizing role concepts according to their context dependencies. We especially focus on defining and organizing a role concept which depends on several contexts.

## 1 Introduction

Currently, Ontological Engineering attracts a lot of attention in many research areas and has been investigated from various view points; fundamental theory, development, application, and so on. However, many ontology development tools do not have enough frameworks to discriminate concepts, which is essential for “Specification of a Conceptualization<sup>1</sup>”. And, few tools provide an advanced framework for ontology description compliant with fundamental theories of ontology.

It is one of the important and essential themes for ontology development to discriminate role concepts from the others (Guarino 1992, Kozaki et al. 2002, Mizoguchi et al. 2000, Sowa 2000). By a **role concept**, we mean a concept of a role which an entity plays in a context. And, by a **basic concept**, we mean the other concept which can be defined without referring to other concepts. For example, role concepts include Lerner, Fuel and Food. Then, we strictly distinguish them from basic concepts such as Human, Gasoline and Yogurt.

However, it is difficult to conceptualize and represent roles correctly. For example, a parent is often represented by a property such as a *parent-of* property or a *parent* property in RDF(S) or OWL without fundamental discussion of their conceptualization. Furthermore, these representations are often confused with each other although they are actually differentiated from each other. The former is a relation which is conceptualized according to a parent-child relation and represented as a binary

relation like “*parent-of* (A, B)”. On the other hand, the latter is conceptualized according to a parental characteristic and represented as a unary predicate like “*parent* (A)”. Without recognition of such a difference, they are often confused with each other.

Needless to say, a parent is a role concept which is determined according to a manner of participation in a parent-child relation. This conceptualization of a Parent Role is based on clear discrimination of a parent-child relation from a parental characteristic. However, it is not easy to represent this definition only in the framework which most of the ontology description languages provide, since we often are confused by the gap between our recognition of concepts and the conceptual framework of ontology languages. One of the approaches for controlling this problem is to use a framework which helps us to differentiate concepts and to represent such differentiation.

That gap resembles to the gap between a high-level programming language and an assembler language. Every program code in any high-level language can be translated down into the assembler language. The program code in the assembler language, however, does not represent directly programmer’s intention about algorithms/data model because it regulates just lower level semantics. In contrast, a program code in a high-level language can represent it more accurately and directly. That is why a high-level language is more familiar to human programmers. Likewise, the frameworks of RDF(S) or OWL provide a framework of ontology representation, and they focus on making bases and common formats to represent definitions of concepts. So, many ontologies can be described in RDF(S) or OWL even if each of them is constructed on their own ontological primitives. However, the frameworks provided by RDF(S) or OWL are not suitable for representing higher semantics of concepts. Developers of the ontologies can represent definitions of concepts more naturally and easily in such frameworks that provide higher level semantics.

In this background, we have developed an ontology building environment, which provides a framework based on the theory of role concepts and their characteristics (Kozaki et al. 2000, 2002). However, in the framework, role concepts are dealt with in a basic-concept-centered view and their definitions are scattered around in the respective related concepts which give the context of the roles. This is why users still have some amount of

difficulty in representing relations among role concepts and grasp their whole image in an ontology. In this paper, as an extension of the previous framework, we present a framework for organizing role concepts in a hierarchy in the role-centered view. First, we investigate how to organize role concepts according to their contextual dependencies. We especially focus on defining and organizing a role concept which depends on several contexts. And we design a system to realize organization of role concepts.

## 2 Role Concepts

### 2.1 Needs of Differentiation of Role Concepts

Context dependence is one of the important characteristics of roles and explains how and why an entity changes its roles to play according to the context it depends on. For example, a Man would be regarded as a Teacher in a School and as a Husband in his Marital Relationship. While such roles can be modeled in connection with time passing, the context-dependence according to the aspect is also necessary semantics for capturing roles property.

Improper modeling of roles will greatly influence the semantics of *is-a* hierarchy of concepts (Guarino 1998). We focus here on the semantics that an instance of a concept is always recognized also as an instance of its super-concept. For example, in WordNet<sup>2</sup>, Dairy Product and Food are treated as hypernyms of Yogurt. If role concepts are not discriminated from the others and these lexical hyponymies among the words are regarded as *is-a* relations among concepts with no distinction, instances of Yogurt are always recognized as instances of Dairy Product and also Food. In such a model, however, we may often have to struggle for faithful representation of events in the real world. To represent that some yogurt has been eaten, we delete the instance of Yogurt. And, it in turn means deleting instances of Dairy Product and Food, which is totally OK. However, in the case where a yogurt has rotted and become inedible, we need to manage instances more sophisticatedly. Because the instance of Yogurt has lost an identity as Food but keeps one as Dairy Product, we can delete only the instance of Food. These managements of an instance model might force us to make different semantics of *is-a* relation and to establish routines for ad-hoc management of instances. Such a strategy detracts from the value of an ontology, which ensures consistency of an instance model. Moreover, it is difficult in such a model to represent the instance of Yogurt changes its roles to play such as Load, Merchandise, Foodstuff, etc. according to changes of its contexts or aspects. It is advisable for a computer model and an ontology behind it to correspond to the real world as truly as possible.

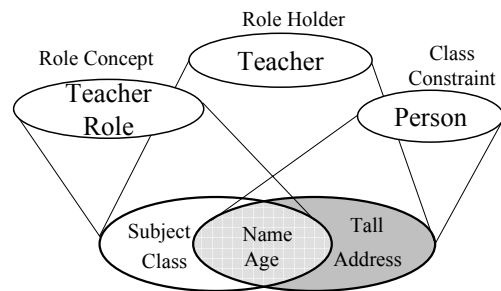


Fig.1 An example of a role concept, a role holder, a class constraint

On the other hand, based on fundamental theories of roles in an ontology (Guarino 1992, Kozaki et al. 2002), we can differentiate clearly role concepts (e.g. Food) from the others and can cope with the problems caused by adulterating role concepts and the others. For example, the hyponymy between Yogurt and Food is not regarded as an *is-a* relation. And, we acquire a consistent policy to manage instances of yogurt and food consistently. It is not easy but worth for ensuring quality of an ontology as a backbone of an instance model to differentiate role concepts from others and organize them.

### 2.2 Role Concepts in Hozo

With citing work by Charles S. Peirce, Sowa introduced the *firstness*, the *secondness* and the *thirdness* of concepts (Sowa 1995, 2000). The *firstness* can be roughly defined as a concept which can be defined without mentioning other concepts. Examples include iron, a man, a tree, etc. In a similar, the *secondness* can be defined as a concept which cannot be defined without referring to other concepts. Examples include a wife, a teacher, a child, etc. The *thirdness* links the *firstness* and the *secondness*. Examples include paternity, brotherhood, etc. Based on these theories, we call one kind of the *secondness* type a **role concept** in this paper. It represents a role which an entity plays in a context or a label changed according to the context. And, by a **class constraint**, we mean constraint on a class which an instance playing the role belongs to. On the other hand, we treat a concept which is defined without referring to a definition of other concepts as a **basic concept**. The class constraint usually refers to the basic concept which is defined elsewhere. When the instance of a basic concept plays the role, we call it a **role holder**. For example, a Man, who is recognized as a Teacher, is a role holder when he/she plays a Teacher Role. But, we do not classify him/her into a Teacher. By a role holder, we just mean that the instance is playing the role. The role holder has both properties of the role concepts and the basic concepts as its components. The relations among the definitions of these concepts are explained roughly in Fig.1. In this manner, we can recognize a role concept with identification of its context, class constraint and role holder.

<sup>2</sup> <http://wordnet.princeton.edu/>

In these considerations of role concepts, we have developed an ontology building environment, which provides a framework for representation of role concepts and their characteristics. The system is named Hozo<sup>3</sup> (Kozaki et al. 2000, 2002) and composed of Ontology Editor, Onto-Studio, Ontology Server and Ontology Manager. Users of Hozo can browse and modify ontologies with its ontology editor (in Fig.2). In Hozo, two kinds of basic concepts: a whole concept and a relational concept are defined. And, a role concept is defined within the context specified by the basic concept. The system manages some basic concepts as contexts of role concepts and provides a framework to define a role concept. Fig.2-a) shows the form of presentation for definitions of concepts on a browsing panel of Ontology Editor. A role concept is represented as a node connected with the other node representing a concept as its context. The connection is shown as a link representing a *part-of* relation (denoted by “p/o”) or a *participate-in* relation (by “p/i”) according to the classification of its context. For example, Fig.2-b) represents that a **Person**, who is referred to as the class constraint, plays a **Teacher Role**, and then becomes a role holder a **Teacher**.

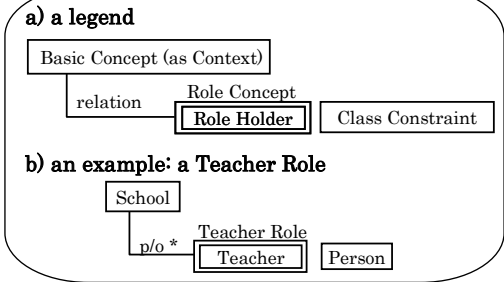
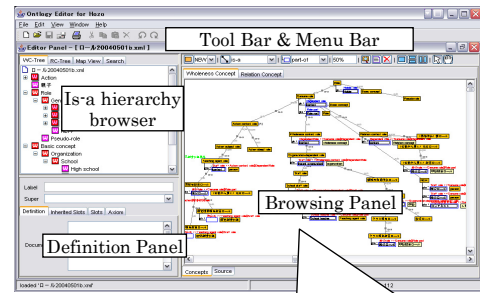


Fig.2 Ontology Editor in Hozo and its form of presentation for definitions of role concepts

### 3 Organizing role concepts

In this section, we present a framework for organizing role concepts in an ontology. By organizing role concepts, we mean mainly constructing a hierarchy of role concepts in order to grasp and represent relations among role concepts and structures of their context dependences. In the following sections, we explain some considerations as guides to organizing role concepts and what information a hierarchy of role concepts has.

Here, we discuss organizing role concepts referring to an example: hierarchies of basic concepts and role concepts constructed using Hozo. The hierarchies are composed of some concepts in school<sup>4</sup> (in Fig.3 and Fig.4).

To begin with, a **Role** is defined at the top of the hierarchy of role concepts (in Fig.4-a) as a class which has three slots: a context, a holder and a role part. The first is related by a participate-in relation and describes in what context the role concept is recognized. The second is also related by a participate-in relation and show a basic concept which can carry the role concept. The third is related by a *part-of* relation and associated with role aggregation (described in 3.2).

#### 3.1 Organizing role concepts according to classification of their contexts

Because role concepts are generally regulated in their context-dependencies, they can be organized according to categories of their contexts and formalities of the

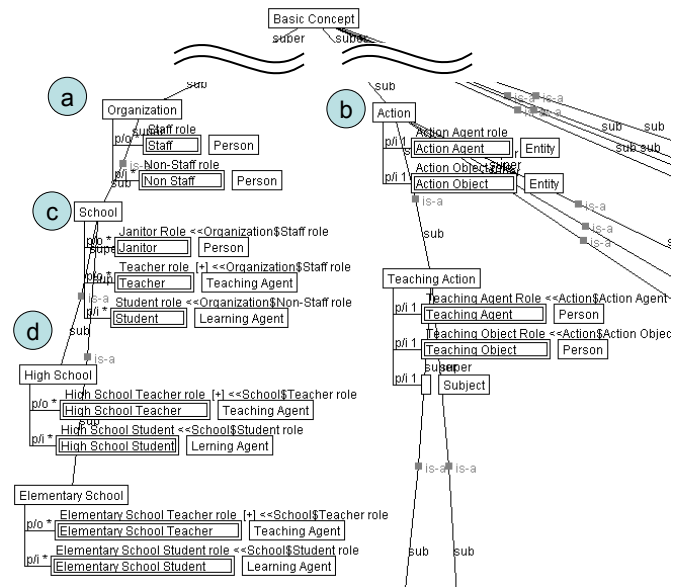


Fig.3 An example of the hierarchy of basic concepts

dependences<sup>5</sup>. For example, task knowledge for solving problem can be discriminated from domain knowledge of a target world. Then, roles recognized in the task knowledge are classified into a **Task Context Role**, which represents a set of classes of roles depending on task knowledge. And, task context roles are divided into more specific concepts according to concrete tasks. For example, a **Fault**

<sup>3</sup> <http://www.hozo.jp>

<sup>4</sup> This ontology is developed in order to discuss semantics of role concepts. So, it is incomplete and concepts are not defined in detail.

<sup>5</sup> According to domain and scope of a target world of an ontology or its purpose to use, developers of the ontology can decide a standard for recognizing contexts and their concrete categories. Here, we just mention typical ones.

**Hypothesis Role** and a **Test Substance Role** are classified relatively into a **Fault Diagnosis Context Role** and an **Experiment Context Role**. In such a manner, roles in the domain knowledge are also organized characteristically according to their target worlds. When a **Function** class is defined as a concept, an **Air-conditioning Role** and a **Pressure Indicator Role** are classified into a **Function Context Role**, which is decided by the function an artifact achieves. Likewise, we consider that there are, at least, the following kinds of a role concept: an **Action Context Role** (a Weapon, a Learner), a **Relation Context Role** (a Friend, a Brother), a **Capacity Context Role** (a Club Member, a King), a **State Context Role** (a Newcomer, a High Temperature Reactor) and so on. We treat these categories as major divisions of roles at a top level of a hierarchy of role concepts.

In the example, an **Action Context Role** and an **Organization Context Role** are defined and classified into a **Role** as a top-level category of the hierarchy (Fig.4-b). The relations among these role concepts describe from the role-centered view that an **Action** and an **Organization** are categorized as contexts at the top-level of the hierarchy of basic concepts (Fig.3-a, b).

### 3.2 Aggregation of role concepts

Because some roles are conceptualized from several viewpoints and depend on several contexts, they are difficult to organize simply according to their contexts. For example, a **Teacher** is recognized not only as a **Teaching Agent** but also as a **School Staff**. In order to organize such role concepts which depend on several contexts, we need to consider how to represent and manage such multiple context-dependence. Then, we devise the idea of **Role Aggregation**: a framework for organizing role concepts, which depends on several contexts, according to their essential dependencies. Role aggregation is represented in both hierarchy of basic concepts and role concepts. And they have the same semantic information on role aggregation. Fig.5 shows hierarchies extracted from the hierarchies shown in Fig.3 and 4 in order to focus on role aggregation.

A central purpose of role aggregation is decomposition of context dependencies. As examples described above, contexts dependences are generally decomposable. And, for each the most primitive context, we can recognize a role concept which depends only on it. By a **primitive role concept**, we mean such a role concept depending on a single context.

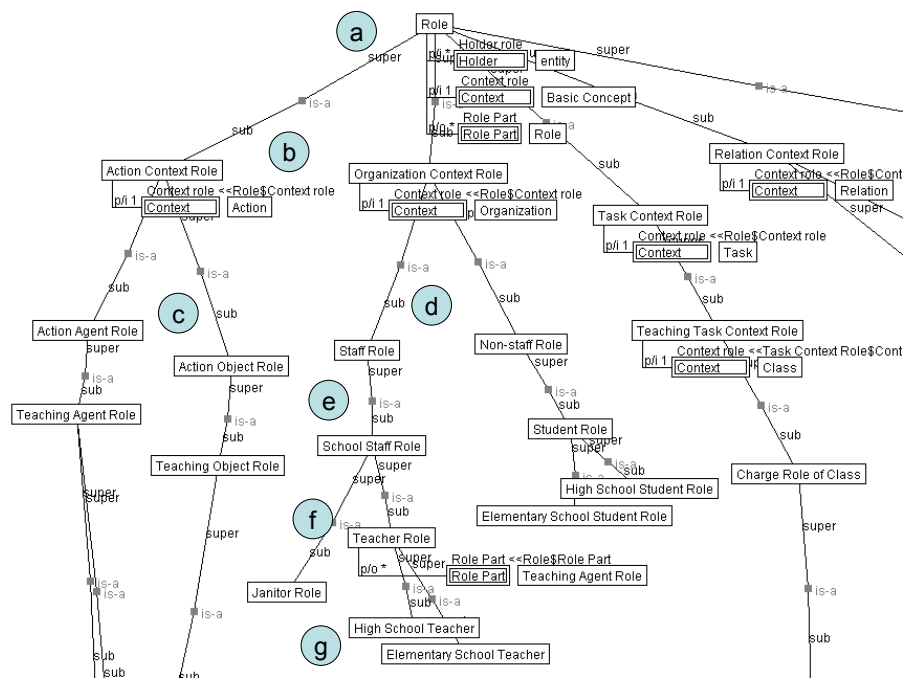


Fig.4 An example of the hierarchy of role concepts

To summarize an outline of role aggregation, we here organize an example of role concept which depends on two contexts. At the start, the most essential context is chosen among the two contexts after investigating and decomposing the context dependence of the role concept<sup>6</sup>. Assume that a **Teacher Role** depends on two contexts: an **Organization** as its essential (primary) context and a **Teaching Action** as its secondary one. And then, two primitive role concepts are identified; a **Staff Role** and a **Teaching Agent**. They depend on each of those contexts respectively.

As described in 2.2, we can constraint on a class which an instance plays the role. In our previous work, a class constraint refers to only basic concepts. Here, we extend our framework and enable the class constraint to refer to also role holders. In this way, a role holder, which is playing some role(s) already, can play other role(s). It also means aggregating context dependences of these roles. This role aggregation is represented in the following manner (Fig.5-a); a **Teacher Role** is defined as a specialized concept of a **Staff Role** and a **Teaching Agent** (role holder) is referred to as a class constraint of a **Teacher Role**. Then, a **Teacher Role** is defined as a role concept which depends on both contexts of a **Staff Role** and a **Teaching Agent Role**.

Next, we explain role aggregation in a hierarchy of role concepts (Fig.5-b). A role concept which has multiple context-dependencies is classified into a role concept

<sup>6</sup> The most essential context is decided by developers of an ontology. We do not discuss or conclude generally what the essential context should be. Based on the relativity of essence, we think that, essences of concepts are decided by the developers intended as far as the decision is consistent in the while ontology.

which depends on an essential context. Role aggregation is represented by using a *is-a* relation and a *part-of* relation<sup>7</sup> as the following manner; a **Teacher Role** is defined as a sub-concept of a **Staff Role** through *is-a* relation, and a **Teaching Agent Role** is defined as a part concept of a **Teacher Role** through *part-of* relation. By **Role Part**, we mean a primitive role concept defined as a part of a role concept which has multiple context dependences.

In our framework of role aggregation, an essential context is decided for each role concept. Otherwise, without such a decision, it is possible to merge context-dependences also in a framework of multiple inheritances. However, it makes relations among role concepts complicated enormously. So, we do not take multiple inheritances to aggregation of role concepts.

### 3.3 Other considerations for organizing role concepts

After classifications of role concepts according to categories of their contexts (described in 3.1), they are organized in detail. This kind of organizing role concepts is located in a middle layer of a hierarchy of role concepts between top categories of role concepts and aggregated role concepts from the bottom (described in 3.2). Here, we mention three significant points of organizing role concepts.

The first is to organize role concepts according to the aspects of entity playing the roles and manners of its participation into contexts. They are clarified in definitions of the contexts and their categories depend on the definitions. For example, we classified a **Weapon Role** and a **Lerner Role** as an **Action Context Role**. And, with investigation of them in more detail, we conclude that the former participates in an action context as an instrument and the latter participates as an agent. Then, we can define an **Action Instrument Role** and an **Action Agent Role** and classify them into an Action Context Role. In the example of the hierarchy of role concepts, a **Staff Role** and a **Non-staff Role** are classified into **Organization Context Role** depending on an **Organization** as its context (in Fig.4-d). This classification represent that a **Staff Role** and a **Non-staff Role** are defined as parts of an **Organization** in the hierarchy of basic concepts (in Fig.3-a).

The second is to organize role concepts based on an *is-a* relation between basic concepts as contexts. In general, role concepts related to an *is-a* relation depends on the same category of context. Assume that there are a sub-concept and its super-concept in a hierarchy of basic concepts. A role concept depending on the sub-concept is recognized by specialization of the context of the role concept depending on the super-concept. Then, in a hierarchy of basic concepts, a relation between these role concepts is represented by using overriding. And, in a hierarchy of role concepts, it is represented as an *is-a*

<sup>7</sup> Here, we focus on a semantics of *is-a* relation that a sub-concept inherits properties of its super-concept and *part-of* relation that a whole concept possesses properties of its part concepts.

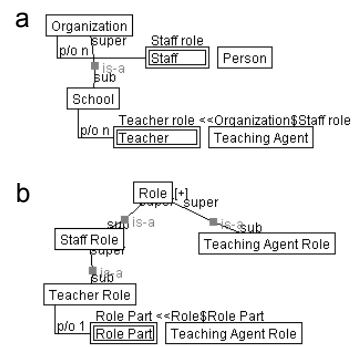


Fig.5 An example of Role Aggregation

relation. In the example of the hierarchy of basic concepts, a **High School Teacher Role** is defined as a part of a **High School** and is recognized by specialization of the context of **Teacher** from a **School** to a **High School** (in Fig.3-d). Then, according to this specialization, in the hierarchy of role concepts (in Fig.4-f,g), **<High School Teacher Role is-a Teacher Role>** is determined.

The third is also based on an *is-a* relation between basic concepts, but it is shown only in a hierarchy of role concepts. For organizing role concepts appropriately, it is indispensable to define role concepts which cannot be described in a hierarchy of basic concepts. Such role concepts are defined for constraint of contexts as intermediate concepts among role concepts described in a hierarchy of basic concepts. They are used mainly for constraint of contexts and not instantiated directly. We call them **Abstract Role Concepts** like an abstract class in an object oriented programming. In the example of the hierarchy of basic concepts, a **Teacher Role** and a **Janitor Role** as parts of a **School** is defined by specializing a **Staff Role** as a part of **Organization** (in Fig.3-c). So, in the hierarchy of role concepts, **<Teaching Staff Role is-a Staff Role>** and **<Janitor Role is-a Staff Role>** are held (in Fig.4-d,f). And then, according to **<School is-a Organization>**, **<School Staff Role is-a Staff Role>** is described (in Fig.4-d,e). In this case, according to their context dependences, a **School Staff Role** is classified into a **Staff Role** and defined as a super class of a **Teaching Staff Role** and a **Janitor Role** in the hierarchy of role concepts (in Fig.4-e).

### 3.4 Information of a hierarchy of role concepts

In this paper, we suggest not only organizing role concepts in a hierarchy of basic concepts but also constructing a hierarchy of role concepts and organizing them also in it.

As described in 2.2, Hozo provides a framework to treat role concepts in a hierarchy of basic concepts. It enables developers of an ontology to represent context-dependences and definitions of role concepts and also some relations among them (e.g. specialization). Furthermore, by extension of this framework, role aggregation can be represented in a hierarchy of basic concepts. Therefore, if the developers organize role

concepts only from these viewpoints, it is not always necessary to construct a hierarchy of role concepts because both of the hierarchies have the same semantic information on role concepts in an ontology. In this case, the hierarchy of role concepts can be constructed automatically according to definitions of role concepts in the hierarchy of basic concepts and used as a viewer of the definitions of role concepts from the other aspect.

However, if the developers organize role concepts in more detail, they need to construct the hierarchy of role concepts for representation of their organizing. For example, the hierarchy of role concepts shows the developers viewpoints of classification of role concepts in the ontology. An abstract role concept compiles role concepts appropriately and contributes to grasping a total image of them. This information represents conceptualization of roles on development of the ontology in detail and helps the developers to understand them. So, the hierarchy of role concepts enhances descriptive quality of role concepts and contributes to making agreement among the developers. With these intentions, we suggest construction of the hierarchy of role concepts as a center of organizing them.

#### 4 Instances of Role Concepts

In this section, we discuss what characteristics of instances of role concepts should be represented in their instance model. Although the instance model is not the main topic in this research, it is indispensable for application of ontologies developed with Hozo and clarification of our strategy for treatment of roles to consider the characteristics of the instances of role concepts. We only describe our current understanding.

While we have investigated basic issues of role concepts in our previous work (Kozaki et al. 2002), it does not include consideration of role concepts which depend on multiple contexts. So, in this paper, we generalize the framework of role concepts. In the following, **R** denotes a role concept,  $C_1...C_n$  its depending contexts,  $R_1...R_n$  primitive role concepts aggregated for definition of the role concept and **P** a concept referred to as the class constraint by the role concept. An instance of **P** can play the role conceptualized as **R**. We explain the framework using an actual example of a **Teacher Role** described in section 3.

(A) States of an instance of a role concept

An instance of **R** has the following two states. (1) Only the role conceptualized as **R** is instantiated (realized). (2) An instance of **P** plays the **R**.

For example, an instance of a **Teacher Role** has two states. One is a teacher role just defined as a part of an instance of **School**. As a vacant position, it is undetermined about who will play it. The other is a role which some person is playing when he/she is recognized as a **Teacher** (role holder).

(B) Dependence of instances of role concepts on their context

An instance of **R** exists if (and only if) all instances of  $C_1...C_n$  are instantiated. When, at least, one of them is deleted, so does the instance of **R**.

For example, a **Teacher Role** is instantiated and a **Teacher** is recognized, on the assumption that a **School** and a **Teaching Action** are instantiated. When the school is closed down or when a teaching class is finished, an instance of a **Teacher Role** is deleted.

(C) Dependence of instances of role concepts on their players

An instance of **R** is dealt with as a defective instance by itself. When instances of  $R_1...R_n$  as constituents of **R** are played by the same instance of **P**, a role holder of **R** is recognized with being composed by an instance of **R**.

For example, when someone is employed as a staff by a school and he/she teaches, all values or ranges of properties of **Teacher** (role holder) are fixed. Then, a **Teacher Role** can be instantiated and he/she is recognized as a teacher.

(D) Extinction of a role holder

A role holder of **R** is recognized as the summation of both instances of **R** and **P**. Here, they are denoted **Ri** and **Pi**. Then, there are four cases in which the role holder is disappear: (1) **Pi** has been disappeared. (2) **Ri** has been disappeared. (3) **Pi** has stopped playing **Ri**. (4) At least, one of role holders of  $R_1...R_n$  is disappeared.

For example, there are three cases in which a person is not recognized as a Teacher. They are (1) when he/she has died, (2) when the post he/she filled has disappeared because of closing down his/her school, personnel reduction and so on, (3) when he/she has retired his/her job as a teacher and (4) when his/her teaching class has been finished.

#### 5 Implementation of the framework

As an extension of the ontology editor in Hozo, we provide a pane for constructing a hierarchy of role concepts and function to support organizing role concepts.

We add a pane for building and editing a hierarchy of role concepts to the ontology editor in a line of panels for basic concepts provided previously (Fig.6). The pane provides almost the same functions as those of the panes for basic concepts. And, we improve the ontology editor to support organizing role concepts in the strategies described in section 3. Firstly, we extend the framework to define concepts for representation of role aggregation. Secondly, we add a function for keeping consistency between role concepts defined in the hierarchies of basic concepts and those defined in the one of the role concepts. This function is based on the fact that some parts of the role concepts defined in both of the hierarchies share the common semantics. For example, if a developer aggregates role concepts in a hierarchy of role concepts, this aggregation is represented automatically also in a hierarchy of basic concepts. And, we provide some wizards for organizing

role concepts. They support operation to deal with role concepts and guide ontology developers.

## 6 Related Work

Guarino and his colleagues aim to establish a formal framework for dealing with roles (Guarino 1992, 1998, Masolo et al. 2004). Their research is concerned with formalities and axioms of an ontology. In contrast, we do not formalize role concepts because our goal is to develop a computer environment for building ontologies. Our notions of role concepts share a lot with their theory of roles; that is, context-dependence, specialization of roles, and so on. According to their theory, our framework can be reinforced in terms of axioms. They describe specialization and requirements as kind of sub-class relations between role concepts. The former corresponds to *is-a* and the latter to role aggregation in our framework. However, they do not describe clearly that *is-a* relations between role concepts are established only if the two concepts share the same category of context-dependency. While we have discussed how to define a role concept which has complicated context-dependences, they only point out a requirement relation. Our notions differ from their work on other two points; that is dynamics of a role and clear discrimination of a role from its player (role holder). Firstly, we focus on context-dependence of a role concept and its categories. So, time dependence of a role concept is treated implicitly in our framework because an entity changes its roles to play according to its aspect without time passing. As opposed to this, their framework deals with time-dependency explicitly. Secondly, we distinguish role concepts and role holders (Kozaki et al. 2002, Mizoguchi et al. 2000). On the basis of this distinction, we propose a tool for properties and relations on roles, such as an aggregation of role concepts.

Fan also recognizes the importance of constructing a hierarchy of role concepts based on differentiation of them from the others and shows an example in that a Thing is classified into an Entity and a Role in (Fan et al. 2001). And, he gives an Agent and an Instrument as sub-concepts of a Role. However, he does not clarify a point of view for organizing them. To our knowledge, they are regarded as being organized according to their manner they participate in their contexts.

Breuker develops ontologies for legal domains based on epistemology and discusses characteristics of roles in (Breuker and Hoekstra 2004). He also mentions adulteration between a role itself and playing role and others between a role and its player. We share his notion in discriminations of these concepts and differentiate a role concept, a class constraint and a role holder from one another (Kozaki et al. 2002, Mizoguchi et al. 2000). He describes two kinds of roles; as a concept and as a relation. However, he does not organize them in more detail. And, in contrast of that he defines roles according to behavioral requirements and so on, we allow developers of an ontology to define role concepts just as the developers

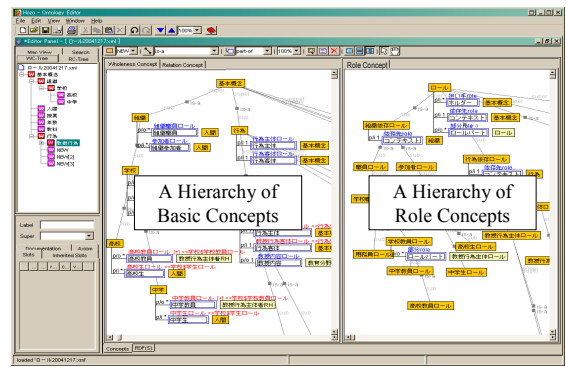


Fig.6 Panes for building and editing hierarchies of basic concepts and role concepts in Ontology Editor

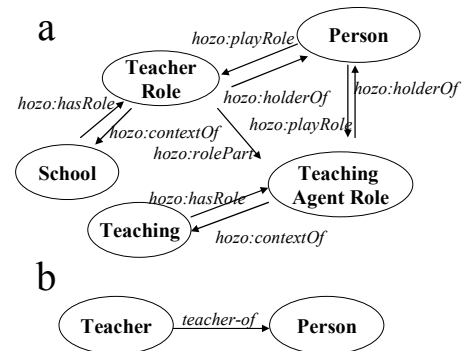


Fig.7 Role representation in OWL

intended because it is outside the scope of our research to discuss how to conceptualize roles.

Next, we focus on distinction between our framework and one OWL provides.

In order to represent characteristics of role concepts discussed in this paper, we define a *hozo:holderOf* property and a *hozo:contextOf* property. Their domains are the same and role concepts. The range of the former is its player and the latter the context which the role concept depends on. And, we define a *hozo:playRole* as property for a class which plays a role. Furthermore, to represent role aggregation, we define *hozo:rolePart* property as a property while range is its role parts. A role concept defined with role aggregation and its role part must refer to the same class as the ranges of *hozo:holderOf* properties. For example, Fig.7-a shows the definition of a **Teacher Role** (described in 3.2 and in Fig.5) in OWL.

Here, we emphasize that role concepts are dealt with not as an *owl:ObjectProperty* but as an *owl:Class* in our framework. They are often defined as a *owl:ObjectProperty* like Fig.7-b. However, in such a definition, characteristics of role concepts discussed in this paper are not represented well. In Fig.7-b, we cannot recognize differences of a **Teaching Role** and a **Teacher** (role holder). And, we cannot represent an instance of a role concept in a state that its player is not determined like a vacant position.

## 7 Conclusion

In this paper, we have developed a framework for organizing role concepts in a hierarchy according to their context-dependences. Then, we investigated instances of role concepts. The definitions of role concepts can be translated into statements in OWL. In conclusion, our framework in Hozo provides a layer in which developers can construct ontologies with high quality description of role concepts and a mechanism for setting it in the current linguistic expression. As future work, we plan to implement the framework in Hozo and investigate a theory of organizing role concepts (e.g. semantics of *is-a* relation between role concepts).

## References

- Breuker, J. and Hoekstra, R. 2004. Epistemology and ontology in core ontologies: FOLaw and LRI-Core, two core ontologies for law. In Proceedings of the EKAW04 Workshop on Core Ontologies in Ontology Engineering. 15-27. Northamptonshire, UK.
- Fan, J., Barker, K., Porter, B., and Clark, P. 2001. Representing Roles and Purpose. In Proceedings of the International Conference on Knowledge Capture (K-Cap2001). 38-43. Victoria, B.C., Canada.: ACM Press
- Guarino, N. 1992. Concepts, attributes and arbitrary relations. *Data and Knowledge Engineering* (8). 249-261.
- Guarino, N. 1998. Some Ontological Principles for Designing Upper Level Lexical Resources. In Proceedings of the First International Conference on Language Resources and Evaluation. 527-534. Granada, Spain.
- Kozaki, K., Kitamura, Y., Ikeda, M. and Mizoguchi, R. 2000. Development of an Environment for Building Ontologies which is based on a Fundamental Consideration of "Relationship" and "Role". In Proceedings of the 2000 Pacific Knowledge Acquisition Workshop (PKAW2000). 205-221. Sydney, Australia.:
- Kozaki, K., Kitamura, Y., Ikeda, M. and Mizoguchi, R. 2002. Hozo: An Environment for Building/Using Ontologies Based on a Fundamental Consideration of Role" and "Relationship". In Proceedings of the 13th International Conference Knowledge Engineering and Knowledge Management (EKAW2002). 213-218. Sigüenza, Spain.: Springer-Verlag.
- Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gengami, A. and Guarino, N. 2004. Social Roles and their Descriptions. In Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR2004). 267-277. Whistler, Canada.: AAAI Press.
- Mizoguchi, R., Kozaki, K., Sano, T., and Kitamura, Y. 2000. Construction and Deployment of a Plant Ontology. In Proceedings of 12th International Conference on Knowledge Engineering and Knowledge Management. 113-128. Juan-les-Pins, France.: Springer-Verlag.
- Sowa, J. F. 1995. Top-level ontological categories. *International Journal of Human-Computer Studies* 43(5-6): 669-685.
- Sowa, J. F. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations.*: Brooks/Cole Publishing Co.