Journal of Southeast University (English Edition) Vol. 22, No. 3, pp. 404 – 408 Sept. 2006 ISSN 1003—7985

On the representation of roles

Yu Yang Wang Ji Wang Ting Chen Huowang

(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

Abstract: The related work to reveal the primary characteristics of roles underlying the existing approaches is analyzed. Then, it is demonstrated that the representation of roles is rather complex and error-prone in these approaches. Especially, the constraints among roles in the same context cannot be expressed by the current web ontology language (OWL). To solve these problems, a novel model of role is presented and a corresponding ontology language is provided for representing this model. The key idea underlying the solution is that a role should be regarded as an element of a certain context and a context as a structured thing which is comprised of some internal elements. The structure of context possesses inherent modularity and local semantics, whereby the representation of roles and context is significantly simplified.

Key words: role; whole structure; ontology language

Ontology is a conceptualization of a domain and plays a crucial role in many communities such as semantic webs. Typically, ontology consists of entities, attributes, relationships and axioms. Ontology languages are, as opposed to logic languages, generally based on the understanding of the world and provide intuitive syntaxes and constructs. Without exception, all ontology languages provide classes as basic constructs and the is-a relationship by which a domain can be organized based on hierarchical structures.

Role is another candidate basic construct which has received considerable attention since role concepts are introduced in such fields of computer science as knowledge representation and conceptual modeling[1–9]. However, an integrative consensus definition of role has not been established yet. An acceptable model of role should satisfy three requirements. First, it should be natural from an ontological perspective. Secondly, it should be general from a theoretical perspective, that is, it can solve all the problems related to roles. Finally, there should be a language which can embody the former two requirements.

In this paper, we first analyze the related work to reveal the primary characteristics of role underlying many approaches. Then, we point out some problems on the representation of roles involved in the existing approaches. Especially, we point out that most constraints related to roles cannot be represented in web ontology language (OWL). A new role model by shifting emphasis on the dependence relationships between role and context is presented. In Ref. [10], a new construct called whole structure driven by the search for an integrated approach to represent context was introduced. The key idea underlying the proposed approach is that a role should be regarded as something internal to a certain context and a context as a structured entity which is comprised of some internal entities. By this, the dependence relationship between role and context is directly embodied in the structure of the context. In addition, the structure possesses inherent modularity and local semantics, whereby the representation of role and context is quite natural and simple.

1 Related Work

We provide a brief overview of role concepts in several selected formalisms in the fields of knowledge representation and conceptual modeling.

In the field of knowledge representation, the main purpose of investigating roles is to reveal their primary characteristics. The common cognition on roles is that a role is necessarily dependent on other entities. Sowa[1] distinguishes natural types "that relate to the essence of the entities" and role types "that depend on an accidental relationship to some other entity." Developing Sowa's ideas further, Guarino et al.[2–3] presented an ontological distinction between role and natural types based on two meta-properties of concepts which are called (external) dependence and rigidity. These notions are employed in a formal account of roles: a role

Received 2006-04-25.

Foundation items: The National Natural Science Foundation of China (No. 90612009, 60403050), the National Basic Research Program of China (973 Program) (No. 2005CB321802).

Biographies: Yu Yang (1976—), male, graduate; Wang Ji (corresponding author), male, doctor, professor, jiwang@nudt.edu.cn.

is a property which is anti-rigid and dependent. Another observation on roles is that the dependence of a role from another entity is involved in the definition of the role. Masolo et al. [4] generalized Fine's definition of dependence[11] into the notion of definitional dependence for describing social roles and societies.

In the field of data modeling, the main motivation for many approaches is the problem of capturing substantial, complex changes of represented entities while retaining their individual identity. Dahchour et al. [5] concentrated on dynamic change of role classes, multiple instantiation of the same class and role-specific access to objects. The latter two points cannot be realized by inheritance only.

The purpose of using roles in software engineering is two-fold. First, role is a nature concept for conceptual modeling. Roles are often used to characterize a specific responsibility of certain object in some context. In the role-centric and collaboration-based software development method, for example, OORAM[6], the concept of role modeling is introduced as the main abstraction mechanism for separation of concerns. Secondly, roles are viewed as a construct to overcome some deficiencies in programming languages: for instance, multiple and dynamic classification, objects changing their attributes and behaviors, etc. Although roles are introduced as a conceptual construct for the analysis and design phase, they are replaced by classes and objects in the implementation phase. A notable exception is the recent work[7] in which the authors explicitly introduce role and context as first class constructs in programming language.

In multi-agent systems (MAS), roles are generally viewed as descriptions of an agent's acting and interacting. The characterization of these kinds of roles is founded on theories of action and behavior and deontic notions. In the Gaia methodology[8], a role is viewed as an "abstract description of an entity's expected function" which is defined by four attributes: responsibilities, permissions, activities, and protocols.

Based on comprehensive analysis of various approaches, Loebe[9] recently gave an account of roles by the simple role model shown in Fig. 1. These elements are recurrent terms which have frequently been used in most approaches.

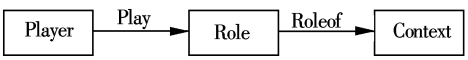

Fig. 1 A simple role model

In addition, there are two primary characteristics of roles which are assumed for most approaches: Every role instance has just one object as its player and belongs to just one context instance; further, it is existentially dependent on its player and context.

2 Problems on Representation of Roles

In this section, we use a project-team as an example to demonstrate that the representation of roles is rather complex and error-prone. A project-team is made up of one manager, one secretary and at least two programmers. In addition, a manager cannot be secretary and vice versa in any project-team.

We start by discussing the disjoint constraint between manager and secretary. With the terms of the model shown in Fig. 1, it may be wrongly expressed as formula (1). In formula (1), we implicitly assume that the classes *Manager* and *Secretary* are both subclass of *Person*. The meaning of formula (1) is that every manager cannot be a secretary and vice versa. It is not our expectation that the manager and secretary of the same project-team are not the same person. The cause is that the *roleof* relation cannot fix the role played by a player in certain context.

$$\begin{aligned}&\mathrm{PTeam}(x) \wedge \mathrm{roleof}(y, x) \wedge \mathrm{roleof}(z, x) \wedge \\ &\quad \mathrm{Manager}(y) \wedge \mathrm{Secretary}(z)) \rightarrow y \neq z \end{aligned} \tag{1}$$

There are two ways by which we can obtain the right meaning. The first one is to introduce a special binary relationship called *rolenameof*. By using *rolenameof* relations, we can correctly express the constraint as

$$\mathrm{PTeam}(x) \wedge \mathrm{Managerof}(y, x) \wedge \mathrm{Secretaryof}(z, x) \rightarrow y \neq z \tag{2}$$

The drawback of this kind of representation is apparent: we have to introduce a special *rolenameof* relationship and a corresponding constraint for every role class.

The second solution is to separate role class from object class, i. e., role instances are not objects. In this way, any *Manager* is a disjoint with any *Secretary* and vice versa. At the same time the constraint should be translated into the one that the players of *Manager* and *Secretary* of any project-team are not the same as

$$\begin{aligned}&\mathrm{PTeam}(x) \wedge \mathrm{roleof}(y, x) \wedge \mathrm{roleof}(z, x) \wedge \mathrm{Manager}(y) \\ &\quad \wedge \mathrm{Secretary}(z) \wedge \mathrm{Person}(s) \wedge \mathrm{Person}(t) \\ &\quad \wedge \mathrm{play}(s, y) \wedge \mathrm{play}(t, z) \rightarrow s \neq t)\end{aligned}$$

This kind of representation is very lengthy as shown above because we have to resort to *roleof* and *play* relations.

Another question is the representation within "a project-team has at least two programmers." Considering that a manager and a secretary can both be programmers, there are two possible interpretations of the project-team. In the first one, a project-team can be made up of two persons, and one of them is both a manager and a programmer, and the other one is both a secretary and a programmer. In the second one, a project-team has at least two programmers which are not the manager or secretary of the project-team. In this case, a project-team is made up of at least four persons. As pointed out above, the constraints among roles are very complex.

To express the constraints between the roles of the same context, we need at least three variables which respectively refer to a context and its two roles. This means that they cannot be expressed with the web ontology language OWL[12]. The logic foundation of OWL is description logics (DLs)[13] which are a well-behaved fragment of first order logic equipped with decidable reasoning. Generally speaking, most DLs are restricted within two variable fragments of first order logic. This restriction is inherited by OWL.

3 Role Model and Its Representation

To overcome the problem pointed out in the previous section, we present a new role model as shown in Fig. 2. The basic elements of our model are similar to the one of Loebe except that the *roleof* relationship is eliminated. However, they are different from each other in structure.

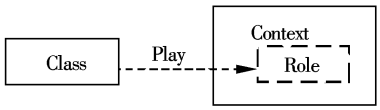

Fig. 2 The new role model

At the concept level, our key choice is to put *Role* in *Context*, thus *Role* is implicitly separated form *Player*. The immediate inference from this choice is that roles are exclusive universals; that is, all object classes are not role classes and vice versa. At the instance level, we assume the two primary characteristics of roles which are assumed for most approaches: every role instance has just one object as its player and belongs to just one context instance; further, it is existentially dependent on its player and context.

An acceptable model of role should not only be natural from an ontological perspective but also have a language which can represent roles and context in a simple and natural way. In the previous work[10], we introduced a new construct called whole structure driven by the search for an integrated approach to represent context. The whole structure can facilitate the organization of concepts and axioms.

Definition 1 Whole structure A whole structure is a description of some type which is a relationship, or collaboration, or a composite class. A whole structure consists of: {Features, Parts, Roles, Relations, Axioms: 〈HasParts, Dependences, Configurations, Qualifications〉, Others} where

① Features are a set of attributes and behaviors. They are optional and are often needed when the type of whole structure is a composite class.

② Parts are a set of concept names whose type is an object class.

③ Roles are a set of concept names whose type is a role class.

④ Relations are a set of relation names whose type is a relation type.

⑤ HasParts is a set of restrictions which describe how many instances of the concepts of *Parts* and *Roles* may exist in the whole.

⑥ Dependences are a set of restrictions which describe the dependence relations between the properties of the whole and the properties of the parts.

⑦ Configuration is a set of constraints which characterize the integrity of the whole.

⑧ Qualifications are a set of restrictions which specify the concepts outside the whole from which the concepts of *Parts* are specialized or by which the concepts of roles are played.

Definition 1 serves as a framework for the definition of concrete context type and it is incomplete and extensible. What a concrete whole structure consists of is determined by its type. We regulate that all concepts and constraints defined in a whole structure are local to this structure and make sense only within it. By this, the whole structure possesses inherent modularity and local semantics. In addition, all axioms except the ones in Qualifications can only use the features, concepts and relations defined in the same whole structure.

For clarity, we use brackets to denote whole structure: every concept which occurs in a pair of brackets is defined in the direct outer concept which is followed by a colon mark. By using the whole structure, the example of project-team can be defined as follows:

ProjectTeam: {
Roles: Manager, Secretary, Prgrammer;
Haspart: $\exists_{=1}x\text{Manager}(x)$, $\exists_{=1}x\text{Secretary}(x)$,
$\exists_{\geqslant 2}x\text{Prgrammer}(x) \wedge \neg\ \text{Secretary}(x) \wedge \neg\ \text{Manager}(x)$
Con: $\text{Manager}(x) \rightarrow \neg\ \text{Secretary}(x)$
Qua: $\text{Secretary} \rightarrow_Q \text{Person}$, $\text{Manager} \rightarrow_Q \text{Man}$; }

The main benefit we gain from this kind representation results from the inherent modularity of the whole structure. Project-team serves as an organization element, and the three role classes *Manager*, *Secretary*, *Programmer*, and the integrity constraints among them are defined in the project-team. In addition, the local semantics of the whole structure is crucial for the representation, that is, the constraints among these participants make sense only in the same project-team. For instance, the constraint $\text{Manager}(x) \rightarrow \neg\ \text{Secretary}(x)$ means that the manager and secretary of the same project-team are not the same. By the local semantics, the *roleof* and *rolenameof* relations are eliminated and thus the constraints are significantly simplified.

In traditional languages, role classes are regarded as subclasses of object classes. The representation of roles in them is rather complex because the interdependence must be explicitly expressed. The main drawback is that more axioms are needed and these axioms are flattened, i. e. , all axioms are assembled at the same level. This makes it difficult to manage and organize ontologies. In addition, this kind of representation is short on intuition and simplicity from an ontological point of view.

Because the local semantics significantly simplify the expression of constraints, many constraints can be expressed by using the formulae with just two variables. By extending DLs with the whole structure, we can gain a more natural, simple and powerful ontology language without changing syntax restrictions. This is another great benefit we gain from the whole structure.

4 Conclusion and Future Work

In this paper, we analyze the related work on the notion of role and point out that the representation of roles is rather complex and error-prone in these approaches. Especially, most constraints among roles in the same context cannot be expressed by OWL. The main reason lies in the dependence between role and player as well as the dependence between role and context. To overcome these problems we present a novel model of role and provide an ontology language for representing the model. The main idea underlying our approach is that a role should be regarded as something internal to a certain context and a context should be regarded as a structured entity which is comprised of some internal entities. By this, the dependence among roles in the same context is directly embodied by the structure of context. The structure possesses inherent modularity, whereby all roles related to a certain context and the constraints among them are organized into the same context. In addition, the structure possesses local semantics, whereby the *roleof* and *rolenameof* relations are eliminated and thus the constraints are significantly simplified.

The future work is to give a formal syntax and semantic of the language introduced. Especially, we will give a language extending DLs with the whole structure and role, and investigate the decidability problem of the language. Another future work is to refine the model into top-level ontology.

References

[1] Sowa J. *Conceptual structures*: *information processing in mind and machine* [M]. New York: Addison Wesley Publishing Company, 1984.

[2] Guarino N. Concepts, attributes and arbitrary relations [J]. *Data & Knowledge Engineering*, 1992, **8**(3): 249 − 261.

[3] Welty C, Guarino N. Supporting ontological analysis of taxonomic relationships [J]. *Data & Knowledge Engineering*, 2001, **39**(1): 51 − 74.

[4] Masolo C, Vieu L, Bottazzi E, et al. Social roles and their descriptions [A]. In: *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning* [C]. Menlo Park, California: AAAI Press, 2004. 267 − 277.

[5] Dahchour M, Pirotte A, Zimányi E. A role model and its metaclass implementation [J]. *Information Systems*, 2004, **29**(3): 235 − 270.

[6] Reenskaug T, Wold P, Lehne O A. *Working with objects*: *the OOram software engineering method* [M]. Englewood Cliffs: Prentice-Hall, 1995.

[7] Tamai T, Ubayashi N, Ichiyama R. An adaptive object model with dynamic role binding [A]. In: *Proceedings of the* 26*th International Conference on Software Engineering* (*ICSE*) [C]. St Louis, Missouri: ACM Press, 2005. 166 − 175.

[8] Zambonelli F, Jennings N R, Wooldridge M. Developing multiagent systems: the Gaia methodology [J]. *ACM Transactions on Software Engineering and Methodology*, 2003, **12**(3): 317 − 370.

[9] Loebe F. Abstract vs. social roles: a refined top-level ontological analysis [A]. In: *Proceedings of the* 2005 *AAAI Fall Symposium "Roles*, *an Interdisciplinary Perspective*: *Ontologies*, *Languages*, *and Multiagent Systems"* [C].

Menlo Park, California: AAAI Press, 2005. 93 - 100.

[10] Yu Y, Wang J, Wang T, et al. Representing wholes by structure [A]. In: *Proceedings of the* 14*th International Conference on Conceptual Structures*, *LNAI*[C]. Berlin: Springer, 2006, **4068**: 441 - 454.

[11] Fine K. Ontological dependence [A]. In: *Proceedings of the Aristotelian Society* [C]. Arstotelian Society, 1995. 269 - 290.

[12] Dean M, Schreiber G. OWL web ontology language reference [EB/OL]. W3C Recommendation, (2004-02-10) [2006-04-01]. http://www.w3.org/TR/owl-ref/2004.

[13] Baader F, Calvanese D, McGuinness D, et al. *The description logic handbook* [M]. Cambridge University Press, 2003.

关于角色的表示研究

于 洋 王 戟 王 挺 陈火旺

(国防科学技术大学计算机学院,长沙 410073)

摘要:通过分析相关工作来揭示角色的基本特征,指出使用现有方法表示角色所遇到的困难和问题,指出角色间的约束超出了 web 本体描述语言 OWL 的表达能力. 为了克服这些问题,提出了一个新的角色模型并提供了一种新的本体描述语言. 该模型的核心思想是将角色看作是上下文的组成元素,并将上下文构造看作是由内部元素组成的结构化实体. 由于上下文构造具有内在的模块化和局部语义性质,因此角色和上下文的表示非常自然和简单.

关键词:角色;上下文;本体语言

中图分类号:TP182