A Basic Consideration on Ontology Refine Method using Similarity among *Is-a* Hierarchies

Takeshi Masuda

Kouji Kozaki

Graduate School of Engineering Osaka University#1

The Institute of Scientific and Industrial Research (ISIR)，Osaka University#2

Abstract. Quality of ontology is important because it is connected directly with the performance of an application system using the ontology. However ontology refinement to improve its quality needs knowledge and experiments in ontology development. Therefore, ontology refinement task is too difficult especially for beginners in ontology building. In order to solve this problem this article proposes an ontology refinement support system based on similarity among is-a hierarchies and an evaluation of it. The system can support content refinements for ontologies.

1 Introduction

Nowadays, ontologies are constructed in various fields such as medical information, mechanical design, and etc. These ontologies are used as knowledge bases and knowledge models for application systems. Quality of ontologies is an important factor for the application system which uses them because it directly affects to its performance. Therefore a construction of better quality ontology is a considerable issue.

However, in order to build well organized ontologies we need knowledge and experience about ontology construction and also expertise in their target domain. For this reason, it is not easy for beginners to construct good ontologies. Because of these backgrounds, ontology construction and refinement support system are expected.

There are some approaches to support ontology building. One is an approach to give some guidelines for the whole process of ontology building [1]. Some researchers propose semi-automatic method to support ontology building process. For example, proposes a semi-automatic method to construct a large scale ontology using semi- structured information such as Wikipedia [2]. On the other hand, ontology refinement is also one of important process to improve quality of was ontologies. This paper proposes an ontology refinement method and an ontology refinement support system based on it.

The rest of this paper is organized as follows. Section 2 outlines existing approaches on supporting method for ontology refinement. In section 3, we propose ontology refinement method based on similarity among *is-a* hierarchies. Section 4 discusses evaluation of the proposed method. Finally, Section 5 concludes this paper.

2 Ontology Refinement Supporting Method

There are two kinds of supporting methods for ontology refinements to improve quality of ontologies. One is a method to detect and correct formal errors in ontologies. And the other is a method to refine contents of ontologies. The former includes many methods for consistency checking and debugging function for ontologies [3]. Poveda et al. collect common errors found in OWL ontologies, which are called ontology pitfalls, and develop a system to detect these pitfalls and correct some of them [4]. On the other hand, when it comes to the latter, there are a few methods to refine contents of ontologies. Although some methods such as voting system [5] and visualization to support understanding of ontologies are proposed, they require human intervention in order to judge right and wrong of contents of ontologies. That is, in these approaches, knowledge of their target domains and experiences on ontology building are necessary to evaluate their contents. Therefore they are not enough to support content refinement of ontologies. To overcome this problem, this paper propose an ontology contents refinement support system which could be help for users who do not have enough experiences on ontology building and domain knowledge.

3 Ontology Refinement Based on Similarity among *Is-a* Hierarchies

3.1 Similarity among *Is-a* Hierarchies

There is a guideline for building well-organized ontologies that "Each subclass of a super class is distinguished by the values of exactly one attribute of the super class. [6]" When ontologies are built under the guideline, we can find many *is-a* hierarchies whose conceptual structures are similar to other *is-a* hierarchies in the same ontologies. For example, in Fig.1, "vehicle" is classified into two lower concepts such as "ground vehicle" and "aircraft" according to their "movement space". These characteristics are represented by referring to other concepts as a class restriction for "movement space". In "vehicle" class, the class constraint of "movement space" slot is "natural space". And it is specialized in slots of its lower concepts "ground vehicle" and "aircraft" to "ground" and "air" respectively. As the result, *is-a* hierarchies of "vehicle", "movement space" slot and "natural space" have partly similar structures.

When a concept is specialized into its lower concepts, some of its slots are also specialized according to *is-a* hierarchies of concepts which are referred as their class constraints. In the same way, we can find similar structures of *is-a* hierarchies of "bicycle", "handle" slot and "handle" in Fig.1.

In this paper, we call "an *is-a* hierarchy of slots whose class constraints are specialized" *"slot hierarchy"*, and "an *is-a* hierarchy of concepts which are referred as class constraints of these slots" *"referred concept hierarchy"*. "*Slot hierarchy*" and its "*referred concepts hierarchy*" have a similar structure. In addition, because a *"slot hierarchy"* is built according to a basic concept hierarchy[1] in which these slots are defined, the basic concept hierarchy also has similar structure with the slot hierarchy. Therefore, we can find partly similar structure among *"basic concept hierarchy"*, *"slot hierarchy"* and *"referred concept hierarchy"* each other (see Fig.1).

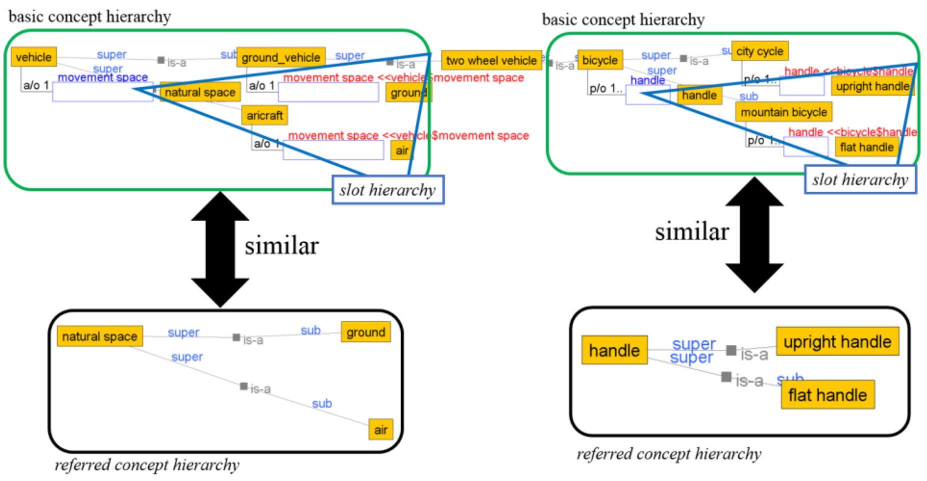

Fig. 1. Similarity among *Is-a* Hierarchies

3.2 Ontology Refinement Using Similarity among *Is-a* Hierarchies

In this paper, we consider ontology refinement system using similarity among three kinds of *is-a* hierarchies introduced in the previous section. There are two directions to use the similarity among them for the ontology refinement.

(1) The first direction is to propose refinement of a basic concept hierarchy according to structure of its referred concept hierarchy. The refinement system compares "referred concept hierarchy" to "slot hierarchy", then it proposes some modification of the "slot hierarchy" to have similar structure with the referred concept hierarchy. As the result, the user also refines "basic concepts hierarchy" according to the proposal.

(2) The other direction is to propose refinement of a referred concept hierarchy according to structure of a basic concept hierarchy which refers to it. The refinement system compare "slot hierarchy" to "basic concept hierarchy", then it make proposal to modify

[1] It corresponds to a normal class hierarchy. We call it basic concept hierarchy to distinguish it with slot hierarchy and referred concept hierarchy.

so that they have a same structure. The proposal involves addition of some concepts to "referred concepts hierarchy". According to the proposal, the user decides what concept should be added and modifies the ontologies.

In this paper, we call the former approaches "Forward refinement" and the later ones "Backward refinement".

3.3 Forward Refinement

Outline of Forward Refinement

For forward refinement, at first the system compare a referred concept hierarchy to the slot hierarchy which refers to it. For comparing them, the system focuses on concepts which are not refereed by any concepts in the slot hierarchy. When the referred concept hierarchy contains such un-referred concepts, it means that these two hierarchies are not similar structure. Therefore, such un-referred concepts are considered as *support target concepts* for forward refinement.

Here, we consider an example for proposals for modifications of an ontology based on forward refinement. In *is-a* hierarchy shown in Fig.2, "gear change" is a support target concept for forward refinement because it is not referred by any other concepts. Then the system consider its upper and lower concepts, "driving operation" and "shift to a lower gear". These two concept are referred by "driving action" and "slowdown" as class constraints of the same slot "part of action". Based on this observation, the refinement system can propose *to add "part of action" slot on "speed control"(fig.2 ①), which is a middle concept between "driving action" and "slowdown", and refer "gear change" (the support target concept) as its class constraint(fig.2 ②).* By following this proposal, "basic concepts hierarchy", "slot hierarchy" and "referred concepts hierarchy" become a similar structure.

In this way, the system makes proposals for modification of an ontology so that these three kinds of hierarchies have similar structures by focusing on un-referred concepts and their upper and lower concepts. There are patterns how the system propose modification of an ontology based on forward refinement. In order to detect which patterns are applicable by the system, we have to make them explicit as conditions which computer are able to distinguish. Using these condition, the system can find out proposals for ontology refinement automatically.

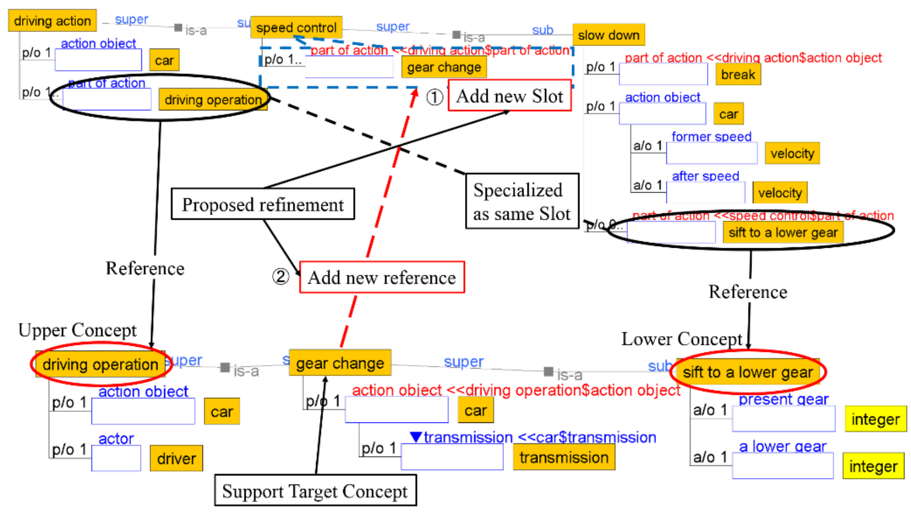

Fig. 2. Refinement Suggestion Example

Patterns for Forward Refinement

Patterns for forward refinement are classified into the following three types by the structure of focusing referred concept hierarchy.

(i) Focusing on the support target concept and both of its upper and lower concepts as the referred concept hierarchy.
(ii) Focusing on the support target concept and its upper concept as the referred concept hierarchy[2].
(iii) Focusing on the support target concept and one of its lower concept as the referred concept hierarchy[3].

On the other hand, which kinds of modifications are proposed for ontology refinement depends on the basic concept hierarchy to which the proposed modification are apply. Therefore patterns for forward refinement are classified into two more types as followings,

A) There is a concept which can be added a new slot referring to the support target concept as its class constraint in the basic concept hierarchy.
B) There is no concept which can be added a new slot referring to the support target concept as its class constraint in the basic concept hierarchy.

[2] We does not care whether the support target concept has lower concept or not. We assume that the ontology does not have multiple inheritances.

[3] As the support target concept which has multiple lower concepts, this system propose each suggestions for each lower concepts. And we distinguish these proposals as different one. When we discuss the result in section 4, we also count these proposals as different suggestions.

In the case of type B), the system can propose two kinds of modification methods; (1) to add new slot on an existing concept and refer target concept as its class constraint and (2) to add new concept which has a slot referring to the support target concept as its class constraint. The users can choose the method (2) when they consider the method (1) is not appropriate, that is, the existing concept could not have the proposed slot. In the case of type B, the system can propose only the method (2) because there is no concept which can be added the new slot.
Based on the above classifications, forward refinement is classified into six patterns by combinations of the above (i) - (iii) and A) - B).

3.4 Backward Refinement

Outlines of Backward Refinement

For backward refinement, the system firstly compares a “slot hierarchy” to its “basic concept hierarchy”. Then, the system proposes to add a new concept to its “referred concept hierarchy” so that these three hierarchies have similar structure. For example, a “basic concepts hierarchy” shown in Fig.3 consists of “car race”, “formula car race”, “F1 race”, “F3 race” and “Indy 500 race”. At the same time, their “machine type” slots are specialized according to its “referred concept hierarchy” shown in Fig.4 and compose its “slot hierarchy”. However in this example, these two hierarchies are not similar. In this case, the system can propose *to add a new middle concept which is referred to as a class constraint concept of the “machine type” slot at “formula car race” concept in the referred concept hierarchy* in Fig. 4 so that it has the similar structure with its slot hierarchy.

Backward refinement has a significant different from forward refinement. It is that the system always proposes to add new concept on the “referred concept hierarchy” for backward refinement.

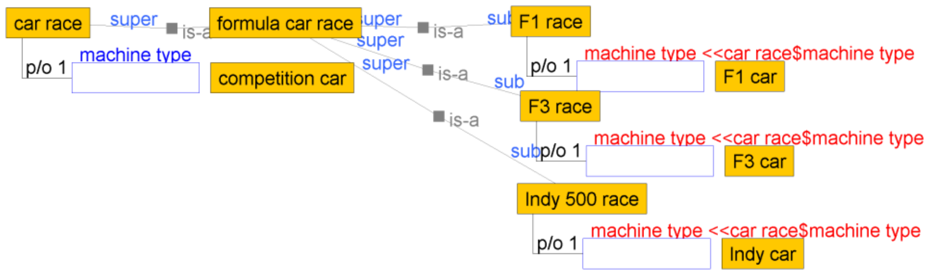

Fig. 3. Base Concepts Hierarchy and Slots Hierarchy about “car race”

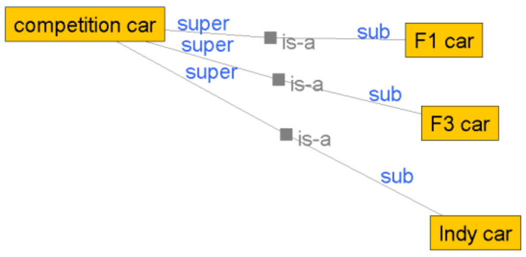

Fig. 4. Referred Concept Hierarchy

Patterns for Backward Refinement

Patterns for backward refinement are classified into the following three types by the structure of focusing basic concept hierarchy.

(i) Focusing on a pair of upper and lower slots which compose a *"slot hierarchy"* and the *"basic concepts hierarchy"* which consists of concepts which have these slots.
(ii) Focusing on a slot, the *"basic concept"* which has the slot and the *"basic concepts hierarchy"* which consists of the concept and its upper concept.
(iii) Focusing on a slot, the *"basic concept"* which has the slot and the *"basic concepts hierarchy"* which consists of the concept and its lower concept.

On the other hand, which kinds of modifications are proposed for ontology refinement In addition, the structure of its referred concept hierarchy affects to which kinds of modifications are proposed for ontology refinement. Therefore patterns for backward refinement are classified into two more types by considering its referred concepts hierarchy as follows;

A) Focusing referred concept hierarchy and basic concept hierarchy have similar structure and there is a concept which can be referred as class constraint for its slot hierarchy in the referred concept hierarchy.
B) There is no concept in the focusing *"referred concepts hierarchy"* which can be referred as class constraint for its slot hierarchy.

In the case of type A), the system can propose two kinds of modification methods; (1) to add new slot on an existing concept in the basic hierarchy and refer to a concept in the referred hierarchy as its class constraint and (2) to add new concept in the referred hierarchy and add new slot which refers to the concept as its class constraint on an existing concept in the basic hierarchy. The users can choose the method (2) when they consider the method (1) is not appropriate, that is, there is no concept which is appropriate as a class constraint of the added slot. In the case of type B), the system can propose only the method (2) because there is no concept which can be referred as class constraint for the added slot.

Based on the above classifications, backward refinement is classified into six patterns by combinations of the above (i) - (iii) and A) - B).

3.5 The Prototype of the Ontology Refinement System.

Based on ontology refinement methods discussed the above, we designed and developed a prototype of ontology contents refinement support system. This system consists of the following three modules;

- *The candidates estimating module* estimates and proposes possible modifications for ontology refinement based on pattern matching discussed in section 3.3 and 3.4.
- *The candidates display and select module* shows the results proposed by the candidates estimating module. The users can select proposed modification they want to apply.
- *The modification apply module* applies the selected modification on the target ontology.

Currently, this prototype system supports for only forward refinement. We are developing supporting function for backward refinement. The system is implemented using Java with some APIS called HozoCore and Hozo OAT (Ontology Application Toolkit) which are libraries deal with ontologies developed by Hozo.

4 Evaluation

4.1 Evaluation Methods

Target ontologies

To evaluate the proposed refinement system, we applied the prototype system to refine several ontologies. Its targets are 9 ontologies built by beginners and 3 ontologies which are developed by ontology experts and open to the public. 4 of them are ontologies which were revised by beginners after meeting ontology experts about given advices for them. They are shown as ontologies whose name have suffix "2" (e.g. race2) in Table.1.

Evaluation criteria

In this paper, we used the following evaluation criteria.

(1) Applicable scope

In the case of forward refinement, the system proposes some modifications focusing on un-referred concepts as support target concept. Therefore, in order to evaluate applicable scope of the method, we calculated the following rate in each target ontology.

(a) The rate of *support target concepts* to *the all concepts* in the target ontology
(b) The rate of *concepts which the system could propose any modifications for refinement* to *support target concepts* in the target ontology

(2) Validity of proposed modifications for refinement

The author evaluated whether proposed modifications are appropriate for ontology refinements. This evaluation was conducted for one of target ontologies built by beginners ("race1" in Table.1).

4.2 Result and Consideration

Applicable Scope

Table.1 shows the evaluation result for applicable scope of forward refinement. The rates of support target concepts to the all concepts in each target ontology are 49% for ontologies built by beginners 61% for ontologies built by experts on average (Tabel.1-(a)). This result is reasonable because of two reasons. First, all concepts do not have to be similar completely. Second, ontologies built by experts have some concept as concepts at middle layer of the ontology. The rates of concepts which the system could propose any modifications for refinement to them are 49% for the beginner's ontologies and 34% for the experts' ontologies on average (Table.1-(b)). We suppose this result shows that the proposed method could cover enough number of concepts as its target for refinement support.

	Ontology made by beginner										Ontology made by expert			
	race1	race2	drums	drums2	traffic	traffic2	rail	rails2	hero_1	average	soccer_ver.12	vehicle	YAMATO101215	average
Group 1((i)A)	1	2	1	1	6	7	2	5	0		6	1	10	
Group1/ All refinement proposal	1%	1%	1%	1%	4%	4%	3%	4%	0%	2%	3%	2%	2%	2%
Group 2((i)B)	2	9	2	2	0	10	3	5	0		0	3	24	
Group1/ All refinement proposal	3%	6%	2%	2%	0%	6%	4%	4%	0%	3%	0%	5%	6%	4%
Group 3((ii)A)	22	34	40	30	20	23	14	19	68		39	6	120	
Group3/ All refinement proposal	29%	24%	43%	36%	14%	13%	20%	14%	49%	27%	23%	9%	29%	20%
Group 4((ii)B)	32	44	23	23	81	80	27	62	67		68	32	148	
Group4/ All refinement proposal	42%	32%	24%	27%	55%	46%	38%	47%	48%	40%	39%	50%	36%	42%
Group 5((iii)A)	3	1	0	0	1	1	0	0	1		0	0	4	
Group5/ All refinement proposal	4%	1%	0%	0%	1%	1%	0%	0%	1%	1%	0%	0%	1%	0%
Group 6((iii)B)	17	49	28	28	39	53	25	42	4		60	22	104	
Group6/ All refinement proposal	22%	35%	30%	33%	27%	30%	35%	32%	3%	27%	35%	34%	25%	31%
Total														
Number of the whole concepts	133	212	135	135	193	251	112	175	50		261	122	561	
Number of all non-reference concepts	79	88	58	58	87	69	77	97	29		148	75	363	
Number of non-reference concepts be found refinement proposal	34	52	34	34	40	41	29	36	11		55	28	101	
Number of all refinement proposal	77	139	94	84	147	174	71	133	140		173	64	410	
(a) Support target concepts/all concepts	59%	42%	43%	43%	45%	27%	69%	55%	58%	49%	57%	61%	65%	61%
(b) Refinement proposal findable rate	43%	59%	59%	59%	46%	59%	38%	37%	38%	49%	37%	37%	28%	34%

(a)The rates of support target concepts to the all concepts in each target ontology.
(b)The rate of each patterns for refinement to concepts which the system could propose any modifications for refinement.

Table 1. Evaluation result for applicable scope of forward refinement.

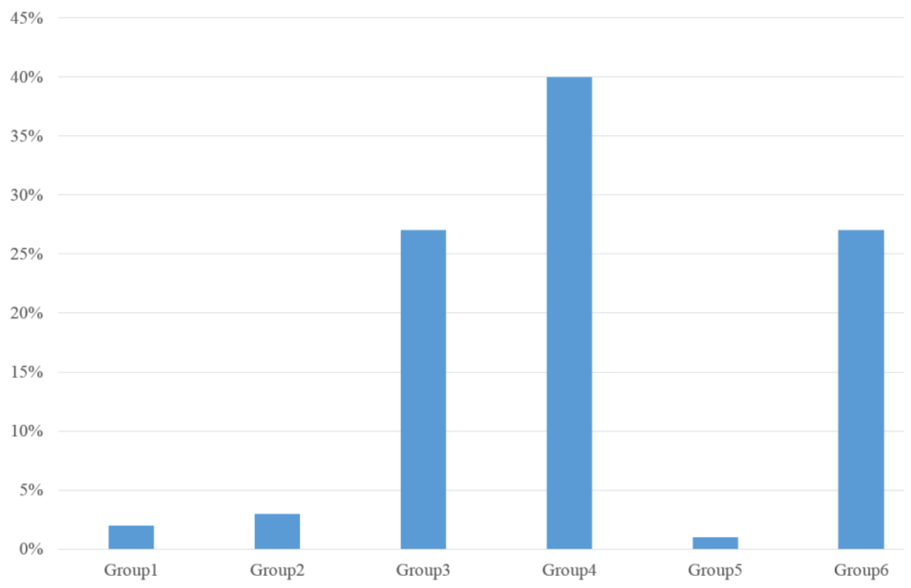

Fig. 5. Distribution of Proposed Refinement Candidates Patterns

Fig.5 said that the number of pattern (i)-A) and (i)-B) are fewer than other patterns. It is because they have stricter conditions to be applied, which consider both of upper and lower concepts of support target concepts, than others. The pattern (i) is also expected higher validity than others because of its condition. On the other hand, the number of pattern (iii)-A) is also fewer than others. It is because the pattern is applied when the lower concept of the support target concept refer a concept which has no upper concept while there is only few such concept.

Validity of Proposal Modifications for Refinement

The author evaluated whether proposed modifications are appropriate for "race2" which was built by beginner. As a result, 20% of the suggestions by this system is evaluated as appropriate modifications (see Table.2).

	Group(i)		Group(ii)		Group(iii)		
	A	B	A	B	A	B	Total
Number of suggested refinement	2	9	34	44	1	49	139
Number of the reasonable proposal	2	5	8	2	0	11	28
Reasonable proposal rate	100%	56%	24%	5%	0%	22%	20%

Table 2. Validity of proposed Modifications for Refinement of race2 ontology

Here, we consider an example of proposed modification which is evaluated as inappropriate. When the system regards “artifact” as a support target concept because it is not referred and its upper concept “entity” is referred by “target” slot at “action” concept, the system makes a proposal that *to add a new lower concept of “action” and add “target” slot whose class constraint is “artifact” on the new concept* [fig.6]. This suggestion means to add a new concept like “artifact target action”, but such concept is unnatural to define. Such unnatural examples were appeared at around top level concept in the ontology. Note here that these examples are just unnatural while they are not ontological errors.

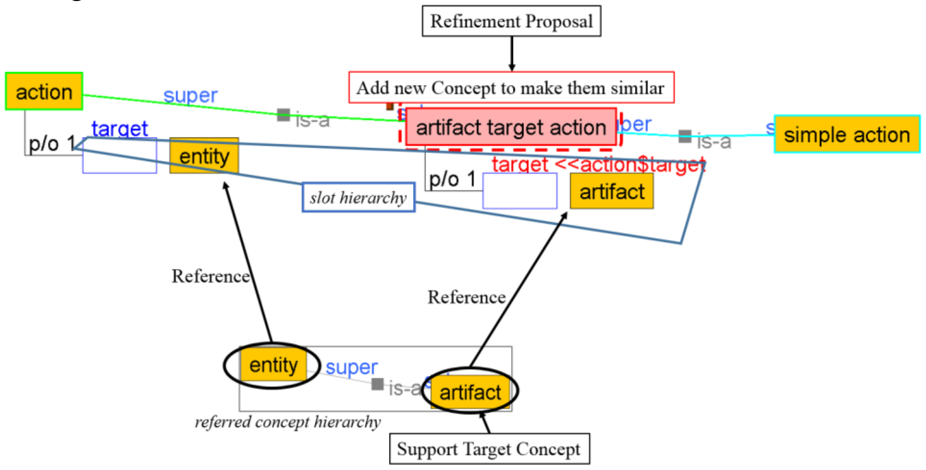

Fig. 6. Inappropriate Example of Refinement Candidate

On the basis of this observation, we removed top level concepts from the support target concepts and evaluated validity of proposed modifications again. As the result, the validity rate promoted 20% to 39% (see Table.3). This suggest us it is effective to remove top level concept from supporting target concepts for improving validity of the propose method.

	Group(i)		Group(ii)		Group(iii)		
	A	B	A	B	A	B	Total
Number of suggested refinement	2	5	18	25	0	21	71
Number of the reasonable proposal	2	5	8	2	0	11	28
Reasonable proposal rate	100%	100%	44%	8%	0%	52%	39%

Table 3. Validity of proposed modifications for refinement of race2 ontology after top level concepts are removed from supporting target concepts.

4.3 Discussion

In the proposed ontology refinement method, the users decide whether they apply suggested modifications or not. So, we believe that 39% of validity is enough high to use it. In addition, inappropriate suggestions are just unnatural while they are ontological correct. Therefore, we think they are not so inconvenient for ontology refinement. However, it is difficult for beginners to decide whether they apply suggested modifications. Therefore, it is an important future work to consider how to compare priority of refinement proposal and how to support finding the most appropriate modification for refinement.

Another important future work is to evaluate the proposed method for more ontologies. At the present, the author evaluated validity of forward refinement for only one ontology made by beginner. We plan to evaluate for more other ontologies and for backward refinement in the future.

5 Conclusion

This article proposed an ontology refinement support system based on similarity among *is-a* hierarchies and evaluated the proposed method partly. The system can support content refinements for ontologies. The feature of the proposed refinement system is to refine ontology semi-automatically. In the future, we will evaluate and consider the backward refinement, and develop the refinement system using forward and opposite refinement and improve the precision of this system's proposal.

References

1. Natalya F. Noy and Deborah L. McGuinness. ``Ontology Development 101: A Guide to Creating Your First Ontology''. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.
2. Takeshi Morita, Yuka Sekimoto, Susumu Tamagawa, Takahira Yamaguchi. : Building Up a Class Hierarchy with Properties from Japanese Wikipedia. WI-IAT '12 Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01, Pages 514-521
3. Oscar Corcho [OEG], Catherine Roussey [LIRIS], Luis Manuel Vilches blázquez [OEG], Ivan Pérez [IMDEA]. : Pattern-based OWL ontology debugging guidelines Dans Workshop on Ontology Patterns (WOP 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009)., Eva Blomqvist, Kurt Sandkuhl, Francois Scharffe, Vojtech Svatek. ed. Washington D.C., USA. pp. 68-82. CEUR Workshop proceedings Vol 5. ISSN 1613-0073.
4. María Poveda-Villalón, Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez. : Validating Ontologies with OOPS! Knowledge Engineering and Knowledge Management Lecture Notes in Computer Science Volume 7603, 2012, pp 267-281
5. Benjamin M Mark D. Wilkinson. : Ontology engineering using volunteer labor. WWW '07 Proceedings of the 16th international conference on World Wide Web, Pages 1243-1244
6. Tutorial on ontological engineering - Part 2: Ontology development, tools and languages New Generation Computing, OhmSha&Springer, Vol.22, No.1, pp.61-96, 2004