Computational Considerations on Interpreting Even If Conditionals

Sang-Ki Han, Key-Sun Choi
(Korea Advanced Institute of Science and Technology)

I. Introduction

Situation logic proposed by Barwise and Perry [BP83; BAR84; BAR85] has been paid much attention in the natural langauge and discourse understanding. One of its major contributions is the analysis of the meaning of sentences based on the constraints between the situation types. In particular, Barwise used the situation semantics in interpreting conditionals [BAR85]. Recently, based on Barwise and Perry's work, Lee and Yoo [LY86] proposed a treatment of even-if statements in English based on the situation semantics. They classified the even-if statements into two cases (necessary and sufficient conditions) and suggest the constraint updating strategy according to the change of background conditions for each cases.

In this paper, we propose the computational model of the situation semantics and show how this model can handle the conditionals and even-if statments. For this, we use the knowledge representation system developed at KAIST called the PROKB (PROlog Knowledge Base) system [HLC86]. It is basically based on the Prolog, which is one of the Artificial Intelligence **(AI)** language and can support the semantic networks, frames, and logical representation as the representational formalism. It also provides several programming paradigms such as the logic programming, object-oriented and access-oriented programming paradigms which are useful for a number of AI applications.

We also present the representational scheme for the situation types and the *involve* relations and the mechanism to handle and maintain the *involve* relations. By combing situation logic with the conventional knowledge representation system in AI, we can get more expressiveness power. For example, the domain knowledge represented in the knowledge base can be

helpful in understanding sentences and answering to questions related to the semantics or meaning of objects themselves in the discouse.

II. Preliminaries

Over the past few years, Barwise and Perry [BP83; BAR85] have developed a significant theory of meaning based on their situation semantics. The main idea is that a situation s can contain information in virtue of some constraint that holds between type of situation. The situation type means the abstraction representing the way things stand in a situation and the constraint is the relation holding between types of situations. We denote the situation types as S, S'... and the constraints as $S \Rightarrow S'$. We read $S \Rightarrow S'$ as S involves S'.

Unlike other theories of meaning, Barwise and Perry distinguished between the meaning of a declarative sentence and the interpretation of an utterance of that sentence. They argued that the interpretation of an utterance is the fact, situation, or event it describes. In other words, it is a type of situation. On the other hand, the meaning of a sentence is considered as a constraint. Especially, they analyzed the conditionals based on this approach in [BP85]. For example, let's consider the following sentence.

(1) If Claire rubs her eyes, then she is sleepy.

When let S be the type of situation where at **l**, Claire is rubbing her eyes and S' be the type of situation where at **l**, Claire is sleepy, S and S' are written as follows:

S = [**s** | in **s**: at **l**: rubbing, Claire's eyes, Claire; 1]
S' = [**s** | in **s**: at **l**: sleepy, Claire; 1]

In this case, the constraint $S \Rightarrow S'$ means that if at some specific space-time location l, s is of type $S(l)$ (the where the parameter **l** is anchored to l), then there is a real situation s':$S'(l)$. In other words, at that very location, Claire is sleepy in s'.

In general a constraint C of the form $S \Rightarrow S'$ will have many parameters, and every parameter in S' will also be a parameter of S. Given any such constraint, and any anchor f for some or all of the parameters in S, that is, any assignment of appropriate values to the parameters, then the result of replacing the parameters by the values will give rise to an actual constraint. That is, if

$$S \Rightarrow S'$$

is actual, the so is

$$S(f) \; = = > \; S'(f).$$

Most constraints to which we are attuned do not apply to every situation, but only under certain conditions. In situation semantics, these conditions are called the background condition. Thus, the *involves* relation is represented as a three-place relation between types of situations:S involves S' given that B, which is written as:

$$S \; = = > \; S' | \; B.$$

Thus, they introduced the parametric constraint so that the interpretation of a general conditional statement is a parametric constraint $C|\mathbf{B}$, where **B** is a parameter anchored to the prevailing background, and where C is $S \; = = > \; S'$, these types being the interpretations of the antecedent and consequent, respectively.

III. Overview of The PROKB System

The basic conceptual element of PROKB is the *Concept*. It is the representation of the entity or relationship of the world being modelled. It consists of several conceptual elements such as the Role, Domain, and Links. The Role represents the attribute of the entity or the relationship between other Concepts and the Domain is a value set of the Role values. The Link represents the conceptual subsumption relation between Concepts. It is also used to represent the inheritance hierarchy in the knowledge base. Since the inheritance hierarchy in the PROKB system is a kind of lattice, we can provide the multiple inheritance.

A Concept can be considered as an aggregation of properties and relations of the entity. With this feature, it can support the frame as in other knowledge representation systems [RG77; BW76]. However, the basic representational formalism provided in the PROKB system is rather the structured semantic networks [BS85; SFS80]. This means that the Concept can be considered as a node in the semantic networks and it consists of a number of other nodes which denote their conceptual subcomponents. There are two types of Concepts, one of which is the Generic Concept and the other is the Individual Concept. The Generic Concept is the classification of the Individual Concept, which is the abstract representation of the real entity in the world. Figure 2.1 shows an example of the Concept definition and its diagram is illustrated in Figure 2.2.

All conceptual elements in PROKB are represented internally as a set of Horn clause since the PROKB system is based on the logic program. Thus, it has very clear semantics and its internal representation has both the declarative semantics and procedural semantics as the logic program

does. In other words, the internal representation can be regarded as the declaration of semantics of each components and, simultaneously, it can be used as a procedure to check the semantic constraints or to obtain the necessary knowledge.

```
defConcept student isa [person]
        with
         [s_no : id_no,
         advisor : professor,
         dept : department,
         lab (default) : laboratory,
         required_credits : integer,
         max_enrollment : integer]
        actions
         [assign_advisor : assign_advisor,
         assign_lab : lab_assign].
```

Figure 2.1 The Definition of 'student' Concept

Basically, the PROKB system can be divided into two major components one of which is the *terminological definition component* and the other is the *assertional component* as other hybrid systems [BFL83; RIC85; VIL85]. The terminological component is for the definition of the conceptual elements and the assertional component is to assert a fact and query for the knowledge base constructed by the terminological component.

We can classify knowledge represented in PROKB into two categories. One is the *definitional knowledge* and the other is the *incidental knowledge*. As our system is based on the logic clauses. For the incidental knowledge, we use the *default logic* developed by R. Reiter [REI80]. Therefore, we can

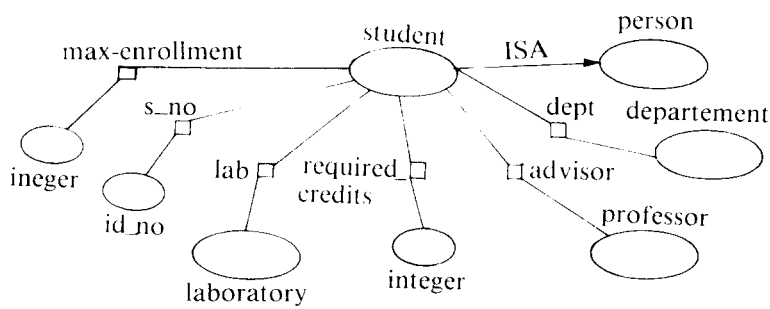

Figure 2.2 The Diagram for A Concept in PROKB

represent and process both kinds of knowledge based on the logic program. By combining the incidental or default knowledge with the definitional knowledge, we can support the prototype of Rosch's prototy theory [CM84].

The PROKB system is a component of the knowledge programming system called *Sphinx* [HLC86]. For a number of AI applications, it also provide several programming paradigms such as the logic programming, object-oriented, and access-oriented programming. The logic programming is used in the knowledge query for the knowledge base and in the knowledge definition mode. The object-oriented programming paradigm is used to represent the behavioral knowledge and transaction, and to carry out the actions by sending messages. Access-oriented programming is similar to the demon processing and triggered procedure in other knowledge representation systems. The procedures represented by the access-oriented paradigm are invoked when a certain situation happens or a certain condition is satisfied.

V. The Representation of Situations

To represent the situation based on the situation logic, we can use our representational scheme without much difficulties. The situation type corresponds to the Generic Concept in PROKB. For example, the following situation type can be represented as in Fig. 3.1

S = [**s** | in **s**: at **l**: rubbing, Claire's eyes, Claire;1]

```
defConcept eye_rubbing
        with
        [sit : situation,
          loc : location,
          exp : [rubbing (claire's eyes, claire)],
          polarity : 1].
```

Figure 3.1 The Representation of A Situation Type

In Fig. 3.1, 'eye_rubbing' denotes the identifier of the situation type and there is no situation type which has the same identifier. When it is an unbound variable, the system provides the generated identifier. As shown above, the 'sit' Role must be a situation and the 'loc' Role must be a location in PROKB. When one of these parameters is anchored, it is represented as the partially instantiated Generic Concept. For example, Sl1 situation

type can be defined as a specialized Concept as follows:

define_concept (eye_rubbingl, eye_rubbing, loc,l1).

, which means that the 'eye_rubbing1' Concept is a subConcept of the 'eye rubbing' Concept whose 'loc' Role is l1. The subConcept inherits any other properties, which is not declared specifically, from its superConcepts.

One of the must distinguished features of the situation logic is the consideration of the meaning by the constraint. The constraint is a three place relation as we explained before. It is represented in our system as follows:

involve(S,S',B).

, where S and S' are types of situations and B is the prevailing background condition. We can interpret conditionals by using this involve relation. If the conditional of the form [if Φ, then Ψ] and S is the interpretation of Φ and S' is the interpretation of Ψ, then the interpretation is a parametric proposition, a proposition relative to the background conditions B. If the constraint is represented as above, we can deduce the actual situation $s':S'$ if S is realized and B holds. For example, let's consider (1) in Section II. In this case, S and S' are the same as shown in Section II, and assume that B is as follows:

$$B = [\mathbf{s} \mid \text{in } \mathbf{s}\text{: at } \mathbf{l}\text{: pollen X;0}].$$

The procedure used in our system to handle conditionals is: if a situation type represented as a Concept is instantiated (realized), the situation type appeared in the involve relation is also instantiated (realized) if the prevailing background conditions are hold. If one of the parameters of the situation type is anchored, it is represented as a partially instantiated Concept and the same procedure is applied. Thus, if $\mathbf{l}$ in S is anchored to a specific space-time location l, then $S'(l)$ is also defined if $B(l)$ holds. Figure 3.2 shows the procedure written in Prolog.

```
define_situation (X, SitType, RoleRestriction):-
        define_concept (X, SitType, RoleRestriction),
        existed (prokb, involve(SitType, Sitype2  BackCond),
        hold (BackCond),
        define_concept (Y, SitType2, RoleRestriction).
```

Figure 3.2 Situation Specialization Procedure

Therefore, we can determine and answer if Claire is sleepy or not, after

we know she is rubbing her eyes by the above procedure. That is, if the prevailing background conditions for that senetence hold, the consequent, Claire is sleepy, become exist in PROKB for the current situation. Therefore, we can obtain information contained in conditionals from the representation of the constraints.

IV. Understanding of Even If Statement

In the case of even if statements, Lee and Yoo [LY86] suggest a treatment based on the situation semantics. They viewed that an even-if statement denies either a sufficient or a necessary condition which is presumed to be part of some conditional knowledge. This can be summarized as the following rules:

[1] In case a conditional of the form [if A, then C] is interpreted as stating that A ==> C | $B1$, that is, on the ground that $B1$, A is a sufficient condition for C, its denial is coherently expressed by an even-if statement (e.g. even if A1, not C) describing the constraint $A1$ =\=> C | B, where $A1 \subseteq$ A.

[2] In case a conditional of the form [if A, then C] is interpreted as stating that C ==>A | $B2$, that is, on the ground that $B2$, A is a necessary condition for C, its denial is choerently expressed by an even-if statement (e.g. even if not A, C) describing the constraint C =\=> $\sim$ A|B, where $A \subseteq \sim$ A.

In order to represent and handle with these problems in PROKB, we must develop the representational scheme for =\=> relation and the negation of the situation type. To represent the not-involve relation, it is only required to assert the following fact in PROKB:

$$\sim\text{involve } (S, S', B).$$

The relation $\subseteq$ between types of situations can be also easily represented as 'isa' Links. For example, in the case of [1], the constraint given by the even-if statement can be represented as follows:

$$\sim\text{involve } (A1,\ C,\ B,).$$

This will not conflict with involve (A, C, $B1$) because A is one of the super-Concept of $A1$ and the background condition B is different from $B1$. Therefore, if the current situation is compatible with B and $A1$ is realized, then we can not deduce the actual situation of C. To do this, we must up-

date the 'define_situation' procedure by adding the followings:

```
define_situation(X, SitType, RoleRestriction) :-
        define concept(X, SitType, RoleRestriction),
        existed(prokb, ~involve (SitType, SitType2, BackCond),
        hold(BackCond).
```

The situation type described by the negation of statement, say not A, means that the polarity of the expression stated by A is reversed. Let's consider the following dialogue:

Sue: If Paul wears a coat, he won't be cold.
Lee: No, even if he wears a coat, he will be cold.

In this dialogue, Lee simply states that wearing a coat is not sufficient for his getting warm. Let S donote the situation type is which Paul wears a coat and S' denote the type of situation in which he will be cold. Then, Sue's statement means:

$$S = = > \sim S' \mid B1$$

where $\sim S'$ means the situation type of [**s** | in **s**: at **l**: be_cold, paul;0]. Lee's statement means that the constraint does not hold in the current situation since the the background condition $B1$ on which Sue's statement has been based differs from the present background condition B which Lee assumes. This is represented as follows:

$$S = \backslash = > \sim S' \mid B$$

Then, after Lees saying, we will be able to say that Paul will be cold even if we know that he wears a coat in the current situation. This can be illustrated by the following session in our system using the formal language.

```
?- define_concept(X,wearing_coat,[loc:today, exp:[wear(paul,coat)],
   polarity:1]).
?- define_concept(X,weather,[loc:today,exp:[so_cold(weather)],
   polarity:1]).
?- find_situation(X,[loc:today, exp:[be_cold(paul)], polarity:1]).
```

When the final query is satisfied and the variable X is unified to a certain situation type, we can say that Paul is cold today.

The case of [2] can be dealt with by the similar method. That is, if the current conditional means the necessary condition, the constraint will be represented as explained in [2]. If the even-if statement is stated, the not-involve relation shown in [2] is asserted in PROKB.

VI. Conclusions

Until now, we consider the representational scheme for situations based on the situation semantics. We also examined how this representation can handle with the conditionals and even-if statements. The knowledge representation system developed for AI systems - the PROKB system - is proved to be useful for representing the situations and constraints. In addition, since the PROKB system can contain the domain related knowledge, our system is very expressive in understanding the natural language sentences.

To be a complete natural language understanding system, we must develop a parser which can parse the natural language sentences into the forms used in situation semantics. Furthermore, a maintenance module for the 'involve' relations must be developed to handle with the world changing dynamically. Finally, an appropriate logic language to define and query the situations should be also developed as a further research.

References

[BAR84] Barwise, J., "The Situation in Logic--I," CSLI-84-2, CSLI, Stanford University, March 1984.

[BAR85] Barwise, J., "The Situation in Logic-II: Conditionals and Conditional Information," CSLI-85-21, CSLI, Stanford University, January 1985.

[BFL83] Brachman, R., J., Fikes, R. E., and Levesque, H.J., "Krypton: A Functional Approach to Knowledge Representation," IEEE Computer, Vol. 16, No. 10, pp. 67-73, 1983.

[BP83] Barwise, J., and Perry, J., *Situations and Attitudes*, Cambridge: The MIT Press, 1983.

[BS85] Brachman, R. J., and Schmolze, J. G., "An Overview of the KL-ONE Knowledge Representation System," Cognitive Science **9**, pp. 171-216, 1985.

[BW76] Bobrow, D.G., and Winograd, T., "An Overview of KRL: A Knowledge Representation Language," AIM-293, Stanford

AILab., November 1976.

[CM84] Cohen, B., and Murphy, G. L., "Models of Concepts," Cognitive Science **8**, pp. 27-58, 1984.

[HLC86] Han, S., Lee, Y. J., and Cho, J. W., "A Multiparadigm Knowledge Representation System," CS-TR-86-16, Dept. of CS, KAIST, 1986.

[LY86] Lee, K., and Yoo, S., "On 'Even If'-Statements," SICOL '86, Seoul, 1986.

[REI80] Reiter, R., "A Logic for Default Reasoning," Artificial Intelligence **13**, pp. 81-132, April 1980.

[RG77] Roberts, B., and Goldstein, I. P., "The FRL Manual," AI Memo 409, MIT AILAB, September 1977.

[RIC85] Rich, C., "The Layered Architecture of a System for Reasoning about Programs," in the Proc. of IJCAI-85, Los Angeles, Calif., August 1985.

[SFS80] Smith, R. G., Friedland, P., and Stefik, M., "Unit Package User's Guide," HPP-80-28, HPP, Stanford University, December 1980.

[VIL85] Vilian, M., "The Restricted Language Architecture of a Hybrid Representation System," in the Proc. of IJCAI-85, Los Angeles, Calif., August 1985.