Vol. 40 No. 3 Transactions of Information Processing Society of Japan Mar. 1999

Regular Paper

Computation Algorithm of Semantic Difference Measure in the SD-Form Semantics Model

MASAHIRO WAKIYAMA,† HIDEKI NODA,†† KOICHI NOZAKI††† and EIJI KAWAGUCHI††

The SD-Form Semantics Model is a new framework to analyze the meaning of natural language in a quantitative way. It is equipped with a formal language named SD-Form which describes the semantic structure of each language expression. In this model the semantic metric depends on both the syntactic structure of the language expression and the knowledge behind it. An elaboration relation is introduced to generalize IS-A, PART-OF, and IF-THEN relations into one partial order relation. This relation is the key idea to the metric definition in the model. The present paper first shows the outline of the model, then discusses the scheme of metric computation in detail. Finally, at the end the authors summarize this work and address the next steps to realize this model.

1. Introduction

There are many papers on the meaning description of natural language[3),7),18),21),34)]. Among all, frames and networks are the most standard approaches[4),20),25),26)]. The first-order predicate logic is another approach in semantics[2),6),19)]. However, nobody has succeeded in building a practical system for semantic difference evaluation based on those models in natural language processing.

Previously, the authors proposed a new semantics model using SD-Forms as a meaning description language[10),11),15),31)]. They termed it the SD-Form Semantics Model. The authors have already reported several feasibility studies on applications[14),16),23),29),32),33)]. The most important point of the model is that it is equipped with a scheme for semantic metric computation in terms of a semantic difference measure.

This scheme is associated with an elaboration relation between two concepts. When a concept (D_2) is a detailed idea of some other concept (D_1), we say that they have an elaboration relation such that D_2 is an elaborated concept from D_1. The amount of elaboration is measured by "elaboration score." The elaboration relation primarily depends on the syntactic similarity of two SD-Forms, but it also depends on the knowledge which is available in the system. The semantic difference measure between two concepts is defined by finding their common meaning, which is termed the nearest common ancestor.

We have implemented a prototype system of the model environment. It is named **SDENV-2** and is a Prolog program of about 300 KB size[13)].

The objective of the present paper is to discuss the way we can implement the computation algorithm of the semantic difference measure under given knowledge.

In Section 2 we review the SD-Form Semantics Model, focusing on an elaboration relation. We will discuss the score setting problem in Section 2.8. The detection algorithm of the nearest common ancestor is studied extensively in Section 3. It provides us with the semantic difference score. Finally in Section 4, we summarize our present work and show the problems for our future work.

2. The SD-Form Semantics Model

The SD-Form Semantics Model defines the SD-Form as a meaning description language. An SD-Form is a well-formed symbol string which we use to describe concept structures. "A concept" in English generally refers to "word meaning." While, in this paper, it refers to a piece of an idea contained in a word, sentence, greeting phrase, fact, rule, or emotional utterance.

2.1 The Background of the Model

It is very important to say that the SD-Form Semantics Model deals with quantitative treatment of meaning in human communication. Meaning in this case does not necessarily

† Kitakyushu National College of Technology
†† Kyushu Institute of Technology
††† Nagasaki University

refer to semantic data which is translated from natural language sentences. Instead, we are discussing the meaning which lies in our mind and will be finally put into wording when we actually communicate with others. We describe such meaning using the SD-Form and put it into our computer. In this regard the following statements are our fundamental view on human communication by natural language.

(1) A human creates a semantic idea when he wants to communicate with others.
(2) The idea becomes concrete before he expresses it by language.
(3) When he puts it into wording, some ambiguity arises.
(4) A human memorizes the meaning, not the precise wording, of the information he received.
(5) Most semantic processes are based on the meaning which a human perceived, not on the memorized wording.
(6) Humans can abstract two meanings into a more generalized idea if the meanings have some similarity.
(7) Humans feel a degree of semantic difference (or similarity) between two meanings. It is more than a type of "same or different" logic.

All these views are put in our SD-Form Semantics Model which we discuss in detail in the rest of this paper.

The SD-Form is like a sign language which does not show precise wording, but conveys sufficient meaning. It is regarded as an interlingua which is independent of any specific language.

2.2 The Syntax of the SD-Form

The syntax of the SD-Form is regulated by a context-free grammar named **SDG** (SD-Form derivative Grammar)[10),24)] (c.f. Appendix). Symbols of the SD-Form include the following.

(1) Concept label (X, Y, DWARF, APPLE, \$1, $\cdots$)

Concept labels are primitive symbols for SD-Forms. We often take advantage of "English-like words" as our concept labels, because it is easier to remember their usage.

(2) Modifier ("/")

When some concept has more information than a single label, we make a detailed description by using a "/" followed by a single SD-Form or a set of SD-Forms connected by "*para*" 's.

(3) Prescriptor (*nega, pass, even, only, assu, fcus*)

Prescriptors prescribe a semantic role of concepts which are negation, passive predicate, emphasis of the speaker's mind as "even" and "only", assumption, and focus of attention.

(4) Connector (*incl, excl, equa, defi, andx, orxx, eorx, caus, indu, effe, bcau, pseq, nseq, inst, addi, plus, than, ofal, exmp, siml, asas, asif, evif, csof, ptof, prof, vaof, kdof, thru, more, less, oppo, para*)

Connectors are connective operators which combine two SD-Forms. We have more than 30 different connectors. Examples of the above are inclusion, exclusion, equality, definition, logical-and, or, exclusive-or, strict causality, plausible causality, possible causality, reason (because), positive sequence, negative sequence, instead, addition, combination, comparative, superlative, exemplification, resemblance, positive degree, subjunctive, concession, consist-of, part-of, property-of, value-of, kind-of, through, more and more, less and less, opposite, and parallel.

(5) Functional item (s, v, i, o, c, b, a, r, e)

We have six types of Statement functional items.

s:	subject item
v:	predicate item
i:	indirect object item
o:	object item
c:	complement item
b:	actor item

We have three types of Emotion functional items.

a:	attention item
r:	reply item
e:	exclamation item

(6) Delimiter ("( )", "[ ]", ",")

SD-Forms are categorized into the 8 types of syntactic structures which follow. "D, D_1, D_2, $\ldots$" in the following description take one of 8 types.

Type 1) Variable label: X
X, X1, Y (something)

Type 2) Simple label: f
FOREST, SNOWHITE, HUMAN, RUN, SOME, 21, \$1

Type 3) Parameterized label: $g(f)$
AGE(40), PAGE(3), DOLLAR(25)
(age 40, page 3, 25 dollars)

Type 4) Modified SD-Form: D/D_1
APPLE/(PRETTY)*para*(RED)
(a pretty red apple)

Type 5) Prescribed SD-Form: $P_i(D)$
nega(AWAKE/MOOD/ABILITY)
(can not awake)

Type 6) Connected SD-Form: $(D_1)C_i(D_2)$
(SNOWHITE)*kdof*(HUMAN)
(Snow-White is a kind of human.)

Type 7) Statement SD-Form: $[s(D_1), v(D_2), \cdots]$ [s(QUEEN/SOME),v(BE), c(MOTHER/SNOWHITE)]
(Some Queen is the mother of Snow-White.)

Type 8) Emotion SD-Form: $[a(D)]$, $[r(D)]$ or $[e(D)]$
[a([s(1STPSN),v(SURPRISE)])]
(Oh!)

The usage of each SD-Form type is standardized by a usage manual[13]. Each SD-Form (D) has a "semantic information score" described by $\boldsymbol{SI}(D)$. The reason we associate each SD-Form with such a score is so we can evaluate the amount of semantic information of each concept by a number (c.f. Section 2.3).

2.3 Semantic Information of an SD-Form

Each SD-Form (D) carries a certain amount of semantic information. This amount depends on the syntactic structure of D. We designate it by $\boldsymbol{SI}(D)$ and call it the semantic information score of D. Although the score-assignment details are left open to each model user, we have some general ideas about it.

(1) $\boldsymbol{SI}(D)$ should be the accumulation of the scores of partial SD-Forms in D. Therefore, a simple SD-Form has a small score, while a complicated one has a large score.

(2) Each symbol of the SD-Form should be equipped with a primary score to initiate score computation.

(3) The primary score of a concept label should be the largest among other SD-Form symbols, because a concept label simulates a word, and "words" are the key information in human communication. We will give a flat score to all concept labels (except for variable labels) because they only inform "concept symbols." Further modification will be possible.

(4) The absolute value of $\boldsymbol{SI}(D)$ does not mean much. We are much more concerned with the relative score of each concept.

The following statements show the primary scores we employed in our experimental system (**SDENV-2**)[12),24),28)]. The unit of the score is termed "semit (semantic information unit)."

A. A variable label has 1 semit.
B. Each simple concept label has 10 semits.
C. The modifier "/" has 1 semit.
D. Each prescriptor has 2 semits.
E. Each connector has 1 semit.
F. Each functional item has 1 semit.
G. "[ ]" has 1 semit.
H. "( )" and "," have 0 semits.

These scores are determined in consideration of the basic ideas (1)–(4) presented above, as well as the requirements of abductive inference in a system (c.f. Section2.8). However, we are not claiming these scores as our essential idea for the SD-Form Semantics Model, rather they are all test scores in our experiment.

⟨**Example 2-1**⟩
$\boldsymbol{SI}$(WOMAN/VAIN) = 21 (Vain woman)
$\boldsymbol{SI}$([s(SNOWHITE), v(EAT/PAST), o(APPLE/POISON)]) = 56
(Snow-White ate a poisoned apple.)

2.4 Hierarchy of System Knowledge

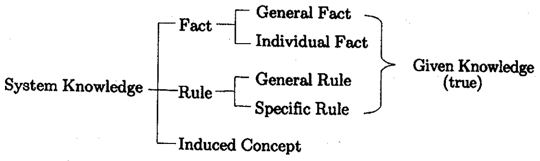


Fig. 1 Hierarchy of system knowledge.

In the SD-Form Semantics Model, system knowledge is categorized as facts, rules, and induced concepts. Facts are classified as general facts and individual facts, although there are no definite categorization standards. Similarly, rules are either general or specific (c.f. **Fig. 1**). As far as the concept reliability is concerned, all the given knowledge is true in the system. An induced concept is a concept generated by unifying a general rule with a concept which is true. When we speak of a knowledge piece, it means a piece of knowledge installed in the system by an SD-Form.

General Facts are all true regardless of the individual topic, story, object world, etc. Individual Facts are concepts referring to a specific concept such as a proper noun. For example, the following SD-Forms illustrate a general fact and an individual fact, respectively.

[s(HUMAN), v(EAT), o(X)]
(Human eats something.)

[s(SNOW-WHITE), v(LIKE), o(BIGMAC)]
(Snow-White likes Big Macs.)

Rules are concepts registered in a system using an "if-then" form. General Rules are given rules in the system. A general rule consists of abstract labels such as variables and common-noun labels. A Specific Rule is either a given rule or an instantiated rule by unifying a general rule with a concept which is true. As for an Induced Concept, let $(assu(D_{2x}))caus(D_{1x})$ and D be a general rule and a true concept, respectively. We unify D_{2x} and/or D_{1x} with D and get an instantiated rule such that $(assu(D_2))caus(D_1)$. In this case, D_1, D_2 and $(assu(D_2))caus(D_1)$ are defined as Induced Concepts.

Classification of knowledge by the words "specific" and "general" is another knowledge categorization for the "knowledge based elaborations" which appear in Sections 2.6.2–2.6.3.

2.5 Elaboration in the SD-Form Semantics Model

An elaboration relation in the SD-Form Semantics Model is a generalized idea of the traditional "IS-A", "PART-OF" or "IF-THEN" relations. This is because all the "IS-A", "PART-OF" and "IF-THEN" relations describe the inheritance of a true property, and the elaboration relation in the SD-Form Semantics Model also refers to the inheritance of a true concept property.

Elaboration from one D_1 to another D_2 means that D_2 is a more specific or detailed expression of D_1. It has two categories, one is a syntactic elaboration, and the other is a knowledge based elaboration. The degree of elaboration is quantitatively measured by an elaboration score. The elaboration score is taken as a measure of uncertainty when we abductively infer D_2 out of true D_1. Formally speaking, the elaboration relation is a partial order relation on a meaning set which is described by the SD-Form.

In natural language, however, the elaboration score is regarded as the amount of new semantic information which D_2 conveys when D_1 is already known. For instance, let

$D_1 =$ "Mary went to Japan"
$([s(\text{MARY}), v(\text{GO}/(\text{PAST})para\ (\text{PLACE}/\text{JAPAN}))])$

be known to us, and

$D_2 =$ "Mary went to Kyoto where Tom visited three years ago"
$([s(\text{MARY}), v(\text{GO}/(\text{PAST})para\ (\text{PLACE}/\text{KYOTO}(\$1)/[s(\text{TOM}), v(\text{VISIT}/(\text{PAST})para\ (\text{TIME}/\text{BEFORE}/\text{YEAR}(3))para\ (\text{PLACE}/\$1))])))])$

is given to us afterward. Then the amount of new information conveyed by D_2 should be the following.

The amount of new information
= (the semantic information of D_2)
− (the semantic information of D_1)

On the other hand, we often face a situation such as follows. Even if some statement is admitted as true, it often becomes uncertain, or unreliable, if we elaborate it by specializing the statement without any concrete given information. For instance, let "Women like desserts" be admitted as true. This expression, however, is ambiguous because "Women" is not necessarily "Any woman." It will be better to take it as "Most women." Furthermore, "desserts" consists of "cake", "fruit", "ice-cream", etc. Therefore, "Snow-White likes pie", which is an elaborated expression without any newly given information, is not very certain even if Snow-White is a woman and a pie is a dessert. This example shows that a meaning in natural language is ambiguous, yet some natural language statements can be regarded as true. In this situation the elaboration score which is equal to the amount of new information is introduced to measure the degree of uncertainty in specializing (abducting) a statement.

Another aspect of the elaboration relation is that it provides a deductive inference rule. When D_2 is true, and it is an elaborated expression from D_1, we can deduct D_1 as true. For example, let "The Prince saved Snow-White" be a fact. Then we can safely say "Some man saved some woman." Moreover, we can say "Snow-White is happy" because "If X saved Y, then Y is happy" is a general rule in an ordinary sense, and "X saved Y" is an elaborated expression of "Y is happy." Thus, we can concatenate the elaboration relation, from detailed to abstract, to make a multi-step deduction. The SD-Form elaboration relations give us the basis for such deductive inference.

When an elaboration relation between D_1 and D_2 holds true in a system, it gives us a concrete basis for computing the semantic difference measure, which is the central topic in this paper (c.f. Section3).

2.6 Formal Definition of $ELAB(D_1, D_2)$

2.6.1 Syntactic Elaboration Relation

When D_1 and D_2 are related in one of the following cases, D_2 is called "syntactically elaborated" from D_1 by the score n. We designate it by,

$\boldsymbol{ELAB_{synt}}(D_1, D_2) = n$, or

$\boldsymbol{ELAB_{synt}}(D_1, D_2, n)$.

1) D_2 is generated from D_1 by the SDG rule (c.f. Appendix). In this case the elaboration score is
$\boldsymbol{ELAB_{synt}}(D_1, D_2) = \boldsymbol{SI}(D_2) - \boldsymbol{SI}(D_1)$.
2) D_1 is a label and D_2 is of the form D_1/D. The elaboration score for this case is,
$\boldsymbol{ELAB_{synt}}(D_1, D_2) = \boldsymbol{SI}(D) + 1$.
3) The case where all the following conditions are satisfied.
   3-1) D_2 is not generated from D_1 by the SDG rule.
   3-2) D_1 and D_2 are the same SD-Form type (c.f. Section 2.2).
   3-3) D_1 is a part of D_2 (See (5) in **Example 2-2**).
   3-4) Each syntactically corresponding pair of SD-Forms (D_1' and D_2'), which are parts of D_1 and D_2, respectively, satisfy $\boldsymbol{ELAB_{synt}}(D_1', D_2')$ (c.f. Fig. 3).

   The elaboration score in this case is,
   $\boldsymbol{ELAB_{synt}}(D_1, D_2) = \boldsymbol{SI}(D_2) - \boldsymbol{SI}(D_1)$.

A syntactically elaborated SD-Form describes the original concept more specifically. As we see in Case 3), syntactic elaboration is recursively defined. A syntactic elaboration relation $\boldsymbol{ELAB_{synt}}(D_1, D_2, n)$ is called "computable" if the score n is computable by the algorithms defined above.

⟨Example 2-2⟩
The score value of $\boldsymbol{ELAB_{synt}}(D_1, D_2)$ depends on the structure of D_1 and D_2. We are using respective scores illustrated in Section 2.3.

(1) $\boldsymbol{ELAB_{synt}}(\text{APPLE}, \text{APPLE}) = 0$ (Case 1)

(2) $\boldsymbol{ELAB_{synt}}(\text{X}, \text{APPLE}) = \boldsymbol{SI}(\text{APPLE}) - \boldsymbol{SI}(\text{X}) = 10 - 1 = 9$ (Case 1)

(3) $\boldsymbol{ELAB_{synt}}(\text{APPLE}, \text{APPLE/(RED)}\textit{para}\text{(BIG)}) = \boldsymbol{SI}(\text{(RED)}\textit{para}\text{(BIG)}) + 1 = 21 + 1 = 22$ (Case 2)

(4) $\boldsymbol{ELAB_{synt}}(\text{APPLE/(RED)}\textit{para}\text{(X)}, \text{APPLE/(RED)}\textit{para}\text{(BIG)} \textit{para}\text{(FRESH)}) = 20$ (Case 2)

(5) $\boldsymbol{ELAB_{synt}}([s(\text{HUMAN}), v(\text{EAT})], [s(\text{HUMAN/ANY}), v(\text{EAT}), o(\text{FOOD})]) = 22$ (Case 3)

(6) $\boldsymbol{ELAB_{synt}}(a([s(\text{1STPSN}), v(\text{MEET}), o(\text{FRIEND})]), a([s(\text{1STPSN}), v(\text{MEET/TIME/MORNING}), o(\text{FRIEND/NEW})])) = 33$ (Case 3)

("MEET" and "FRIEND" syntactically correspond to "MEET/TIME/MORNING" and "FRIEND/NEW", respectively.)

2.6.2 Specific-Knowledge Based Elaboration Relations

We naturally feel that MAN is a more elaborated concept than HUMAN. Also [s(JOE), v(BE),c(TAXI-DRIVER)] is a more elaborated expression than [s(JOE),v(DRIVE),o(CAR)]. Yet, there are no syntactic elaboration relations between them. It only comes from some specific knowledge which we have.

Now, we introduce the "specific-knowledge based" elaboration relation. We designate it by,

$\boldsymbol{ELAB_{know}}(D_i, D_j) = n$, or

$\boldsymbol{ELAB_{know}}(D_i, D_j, n)$

Let a system be equipped with specific knowledge shown below (on the left side of each arrow.) Then the corresponding $\boldsymbol{ELAB_{know}}(D_i, D_j)$ relations on the right side hold true.

Specific knowledge must be explicitly given to the system by users. Therefore, different systems have different specific-knowledge based elaboration scores depending on the knowledge difference.

1) $(D_i)\textit{equa}(D_j) \rightarrow \boldsymbol{ELAB_{know}}(D_i, D_j) = 0$
2) $(D_i)\textit{defi}(D_j) \rightarrow \boldsymbol{ELAB_{know}}(D_i, D_j) = 0$
3) $(D_i)\textit{csof}([D_{j1}, \cdots, D_{jk}, \cdots, D_{jn}]) \rightarrow \boldsymbol{ELAB_{know}}(D_i, D_{jk}) = 2$
4) $(\textit{assu}(D_j))\textit{caus}(D_i) \rightarrow \boldsymbol{ELAB_{know}}(D_i, D_j) = 2$
5) $(D_i)\textit{incl}(D_j) \rightarrow \boldsymbol{ELAB_{know}}(D_i, D_j) = 3$
6) $(D_j)\textit{ptof}(D_i) \rightarrow \boldsymbol{ELAB_{know}}(D_i, D_j) = 3$
7) $(D_j)\textit{kdof}(D_i) \rightarrow \boldsymbol{ELAB_{know}}(D_i, D_j) = 3$

We always set the scores to zero for Cases 1) and 2). This is because "*equa*" and "*defi*" are used for equivalent concept pair of D_i and D_j. While, the scores for Cases 3)–7) are experimental values in our system (c.f. Section 2.8.) The score difference between "2" and "3" does not mean much. However, we regard the "*csof*" and "*caus*" relations as describing slightly closer relations than "*incl*", "*ptof*" and "*kdof*." Closer relations between two concepts convey smaller new information to the system than distant relations.

⟨**Example 2-3**⟩

$\boldsymbol{ELAB_{know}}(D_1, D_2)$ depends on the specific knowledge given to the system. If the left side of each arrow in the following example is the given knowledge to the system, then the right side relations hold true.

- (TOKYO)*equa*(CAPITAL/JAPAN) $\rightarrow$ $\boldsymbol{ELAB_{know}}$(TOKYO, CAPITAL/JAPAN) $= 0$
- (HUMAN)*csof*([MAN, WOMAN]) $\rightarrow$ $\boldsymbol{ELAB_{know}}$(HUMAN, MAN) $= 2$
- (*assu*([*s*(JOE), *v*(BE), *c*(TAXI-DRIVER)])) *caus*([*s*(JOE), *v*(DRIVE), *o*(CAR)]) $\rightarrow$ (If Joe is a taxi-driver, then Joe drives a car.) $\boldsymbol{ELAB_{know}}$([*s*(JOE), *v*(DRIVE), *o*(CAR)], [*s*(JOE), *v*(BE), *c*(TAXI-DRIVER)]) $= 2$
- (KID)*incl*(BOY) $\rightarrow$ $\boldsymbol{ELAB_{know}}$(KID, BOY) $= 3$
- (BOY)*kdof*(MAN) $\rightarrow$ $\boldsymbol{ELAB_{know}}$(MAN, BOY) $= 3$

2.6.3 General-Knowledge Based Elaboration Relations

The SD-Form Semantics Model can deal with "quantifiers" in predicate logic in terms of a "general-knowledge based elaboration." It is as follows.

General-knowledge based elaboration relations are independent of given knowledge. They are introduced as a general nature of the SD-Form Semantics Model. Let

$\boldsymbol{ELAB_{know}}(D_i, D_j) = 2$ and

$\boldsymbol{ELAB_{know}}(D_i, D_k) = 3$

be elaboration relations already established by the specific knowledge in the system. Then we have the following general-knowledge based elaboration relations. The respective score setting was made experimentally in relation to the specific-knowledge based score setting (c.f. Section 2.6.2).

$\boldsymbol{ELAB_{know}}(D_i/X, D_i/\text{SOME}) = 1$

$\boldsymbol{ELAB_{know}}(D_i/\text{MOST}, D_i/\text{ANY}) = 1$

$\boldsymbol{ELAB_{know}}(D_i/\text{SOME}, D_j) = 1$

$\boldsymbol{ELAB_{know}}(D_i/D, D_i/\text{ANY}) = 2$

$\boldsymbol{ELAB_{know}}(D_i/\text{SOME}, D_k) = 1$

$\boldsymbol{ELAB_{know}}(D_j/D, D_i/\text{ANY}) = 2$

$\boldsymbol{ELAB_{know}}(D_i/\text{SOME}, D_i/\text{MOST}) = 3$

$\boldsymbol{ELAB_{know}}(D_k/D, D_i/\text{ANY}) = 2$

$\boldsymbol{ELAB_{know}}(D_i/\text{SOME}, D_i/\text{ANY}) = 4$

$\boldsymbol{ELAB_{know}}(D_j, D_i/\text{ANY}) = 3$

$\boldsymbol{ELAB_{know}}(D_i/\text{SOME}, D_i/D) = 2$

$\boldsymbol{ELAB_{know}}(D_k, D_i/\text{ANY}) = 3$

To be more general, in the cases where the specific knowledge data takes the form

$\boldsymbol{ELAB_{know}}(D_i^*/D_0, D_j) = 2$ or

$\boldsymbol{ELAB_{know}}(D_i^*/D_0, D_k) = 3$,

all the expressions D_i/SOME, D_i/MOST, D_i/D and D_i/ANY in the above formulas should be replaced by, D_i^*/(SOME)*para*(D_0), D_i^*/(MOST)*para*(D_0), D_i^*/(D)*para*(D_0) and D_i^*/(ANY)*para*(D_0), respectively.

⟨**Example 2-4**⟩

Let the System knowledge be

(HUMAN)*csof*([MAN, WOMAN]),
(HUMAN/SMALL)*incl*(DWARF),
(BOY)*kdof*(MAN).

In this case,

$\boldsymbol{ELAB_{know}}$(HUMAN/SOME, WOMAN) $= 1$

$\boldsymbol{ELAB_{know}}$(HUMAN/(SOME)*para*(SMALL), DWARF) $= 1$

$\boldsymbol{ELAB_{know}}$(BOY, MAN/ANY) $= 3$

$\boldsymbol{ELAB_{know}}$(MAN/SOME, MAN/ANY) $= 4$

As we have seen in Sections 2.6.2 and 2.6.3, a knowledge based elaboration score designates a semantic distance between two concepts after all the given knowledge is counted as the system knowledge. Both "specific" and "general" knowledge based elaboration relations are simply referred to as "knowledge based" elaboration relation.

2.6.4 Definition of *ELAB* Relation

In the SD-Form Semantics Model, an elaboration relation between D_1 and D_2 is denoted by

$\boldsymbol{ELAB}(D_1, D_2) = n$, or

$\boldsymbol{ELAB}(D_1, D_2, n)$

where, n is the elaboration score given by

$$n = \min\{\boldsymbol{ELAB_{synt}}(D_1, D_2), \boldsymbol{ELAB_{know}}(D_1, D_2)\}. \quad (1)$$

(i.e., n is the minimum value of $\boldsymbol{ELAB_{synt}}(D_1, D_2)$ and $\boldsymbol{ELAB_{know}}(D_1, D_2)$).

$\boldsymbol{ELAB_{synt}}$ and $\boldsymbol{ELAB_{know}}$ are syntactic and knowledge based elaborations, respectively.

The unit of both scores is "semit." If no $\boldsymbol{ELAB_{synt}}(D_1, D_2)$ (or $\boldsymbol{ELAB_{know}}(D_1, D_2)$) relation exists, we consider that $\boldsymbol{ELAB_{synt}}(D_1, D_2)$ (or $\boldsymbol{ELAB_{know}}(D_1, D_2)$) has an infinite score. $\boldsymbol{ELAB}$ becomes infinite when both $\boldsymbol{ELAB_{synt}}$ and $\boldsymbol{ELAB_{know}}$ are infinite.

In **SDENV-2**, we have set $\boldsymbol{ELAB_{know}}$ scores as

$\boldsymbol{ELAB_{know}}(D_1, D_2) \leq \boldsymbol{ELAB_{synt}}(D_1, D_2)$ for any given D_1 and D_2. This is because we regard each knowledge piece as conveying more semantic information to the system than any syntactic relation. Therefore, the amount of semantic information new to the system, that is the elaboration score, is bigger in the syntactic case than in the knowledge based case (c.f. Section 2.8). The details of the $\boldsymbol{ELAB_{synt}}(D_1, D_2)$ and $\boldsymbol{ELAB_{know}}(D_1, D_2)$ definitions have been spelled out in other literature[12],[24].

When D_2 is elaborated from D_1, D_1 is called an "ancestor of D_2"and D_2 a "descendant of D_1." The most important property in $\boldsymbol{ELAB_{synt}}$ is that a sequence of syntactic elaborations is always reduced to a single $\boldsymbol{ELAB_{synt}}$, while $\boldsymbol{ELAB_{know}}$'s cannot be reduced[10]. The fact that each multi-step $\boldsymbol{ELAB_{synt}}$ is reducible to a singe step one can be proved by checking all the three cases which are itemized in Section 2.6.1.

When D_1 and D_2 satisfy a $\boldsymbol{ELAB_{know}}(D_1, D_2)$ relation, D_1 is called a knowledge head, and D_2 a knowledge tail.

2.7 Multi-step Recursive Elaboration

A set of elaboration relations can be concatenated into a multi-step elaboration. In the case of an m-step elaboration, we designate it by $\boldsymbol{ELAB}_m(D_1, D_2)$. Because an elaboration is either syntactic or knowledge based, and a concatenation of syntactic elaborations is always reduced to a single step, the number of combinations of syntactic (denoted by (S)) and knowledge based (denoted by (K)) relations in an m-step elaboration is less than 2^m. We can prove that this number is a Fibonacci number. For $m = 1, 2, 3$, all the m-step elaboration types are the following. We call each combination pattern a "path type."

1 step: D_1-(K)-D_2,
D_1-(S)-D_2

2 step: D_1-(K)-(K)-D_2,
D_1-(K)-(S)-D_2,
D_1-(S)-(K)-D_2

3 step: D_1-(K)-(K)-(K)-D_2,
D_1-(K)-(K)-(S)-D_2,
D_1-(K)-(S)-(K)-D_2,
D_1-(S)-(K)-(K)-D_2,
D_1-(S)-(K)-(S)-D_2

SDENV-2 is equipped with the procedures to compute the scores for all the path types up to 3 steps. It is important to know that each syntactic part (i.e., the (S) part) of the m-step elaborations also includes recursive m-step elaborations in part. A given concept pair (D_1, D_2) can have many m-step elaboration paths with different scores corresponding to different knowledge combinations for the same m. In a practical system we have to limit the maximum number of m to some M when we compute $\boldsymbol{ELAB}_m(D_1, D_2)$, otherwise the computation never ends. Therefore, the elaboration score defined by Eq. (1) must be revised as

$$\begin{aligned} &\boldsymbol{ELAB}(D_1, D_2) = n, \text{ or} \\ &\boldsymbol{ELAB}(D_1, D_2, n) \\ &\quad n = \min\{\boldsymbol{ELAB}_m(D_1, D_2)\} \qquad (2) \\ &\quad (m = 1, 2, \cdots, M \\ &\qquad \text{for some } M \text{ and for all paths}). \end{aligned}$$

The following example illustrates a 3-step elaboration relation $(D_1$-(S)-(K)-(S)-$D_2)$. Two (S)'s here contain either 1-step internal syntactic or knowledge based elaborations.

⟨Example 2-5⟩

Let D_1 and D_2 be

D_1 = [*s*(FRIEND/(SOME)*para*(KING)), *v*(BE), *c*(MOTHER)],
(Some of the King's friends are mothers.)

D_2 = [*s*(QUEEN/CERTAIN), *v*(GIVEBIRTH/PAST/AGO/LONG), *o*(SNOWHITE)],
(A Queen gave birth to Snow-White long ago.)

and the knowledge available in this case be,

Individual Fact:
[*s*(QUEEN/CERTAIN),*v*(GIVEBIRTH/PAST),*o*(SNOWHITE)]
(QUEEN/CERTAIN)*kdof*(FRIEND/KING)

General Rule:
(*assu*([*s*(X),*v*(GIVEBIRTH/PAST),*o*(Y)]))
caus([*s*(X),*v*(BE),*c*(MOTHER/Y)]).
(If X gave birth to Y, then X is Y's mother.)

In this case we can compute
$\boldsymbol{ELAB}_3(D_1, D_2) = 36$ semits.

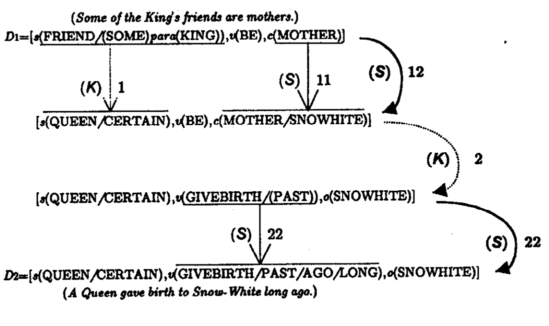

(Some of the King's friends are mothers.)
D1=[s(FRIEND/(SOME)para(KING)),v(BE),c(MOTHER)]
(K) 1
(S) 11
(S) 12
[s(QUEEN/CERTAIN),v(BE),c(MOTHER/SNOWHITE)]
(K) 2
[s(QUEEN/CERTAIN),v(GIVEBIRTH/(PAST)),o(SNOWHITE)]
(S) 22
(S) 22
D2=[s(QUEEN/CERTAIN),v(GIVEBIRTH/PAST/AGO/LONG),o(SNOWHITE)]
(A Queen gave birth to Snow-White long ago.)
ELAB(D1,D2)= 12+ 2 + 22 =36 semits

Fig. 2 An example of a multi-step elaboration relation.

Figure 2 illustrates this computation. In this computation, we use a specific rule:

$$(assu([s(\text{QUEEN/CERTAIN}), v(\text{GIVEBIRTH/PAST}), o(\text{SNOWHITE})]))$$
$$caus([s(\text{QUEEN/CERTAIN}),v(\text{BE}), c(\text{MOTHER/SNOWHITE})])$$

which is induced by instantiating the General Rule with D_2.

2.8 How We Should Assign the Elaboration Scores

In this section we discuss how we should assign the elaboration scores. The point of the discussion is to determine how to balance the syntactic and knowledge based elaboration scores. As we mentioned in Section 2.5, an elaboration relation gives us an abductive inference rule from D_1 to D_2 with an uncertainty degree $\boldsymbol{ELAB}(D_1, D_2) = n$, which is either $\boldsymbol{ELAB_{know}}$ or $\boldsymbol{ELAB_{synt}}$. So, we will discuss the score-balancing problem from this point of view.

$\boldsymbol{ELAB_{know}}$ is primarily based on specific knowledge available in the system. We can concatenate $\boldsymbol{ELAB_{know}}$'s if such knowledge is chained in a sequence. However, the more we concatenate, the more uncertain the abduction becomes. This is similar to the human inference process. For a human, a 2-step abductive inference will be meaningful, but a 5-step abduction may mean nothing.

Meanwhile, an abductive inference by an $\boldsymbol{ELAB_{synt}}$ relation has no concrete basis to stand on. The inference in this case is very uncertain. Sometimes it is meaningless.

⟨Example 2-6⟩

Let

"An apple is a kind of fruit."

be factual knowledge in terms of natural language, and

D_1 = "Snow-White likes fruit."

be given as a true concept. For this D_1,

D_{20} = "Snow-White likes apples."

is a knowledge based elaboration.

Furthermore, we can make a virtually unlimited number of syntactically elaborated statements such as the following.

D_{21} = "Snow-White likes fresh fruit",
D_{22} = "Snow-White likes red juicy fruit",
D_{23} = "Snow-White likes round fruit which is expensive."

In these examples, D_{20} sounds realistic, and some of D_{21}, D_{22}, and D_{23} could be possible as far as our general knowledge is concerned. However, all these syntactically elaborated statements have no reason to be true. They are all uncertain.

We can conclude from this discussion that we could balance the score of a maximum-step concatenated $\boldsymbol{ELAB_{know}}$ with the score of a minimum $\boldsymbol{ELAB_{synt}}$.

This idea was realized in the score balancing problem in **SDENV-2** in such a way that most 3(or more)-step $\boldsymbol{ELAB_{know}}$'s are comparable to a single step $\boldsymbol{ELAB_{synt}}$. This made the score setting of **SDENV-2** just as was shown in Sections 2.3 and 2.6.

3. Computation of Semantic Difference Measure

One of the very fundamental problems in modeling human communication by natural language is an introduction of a semantic metric. It is an indispensable framework for natural language understanding by machine. Traditional researchers, however, were rather reluctant to start struggling with such new attempts. Among them, Ref. 17) studied conceptual distance by example learning. While, Ref. 20) discussed the existence of the triangular property in word meaning. People in database technology appreciate this triangular property because of the simplicity of distance evaluation[1]. Case-based reasoning methods always discuss the similarity of examples (concepts)[5],[8],[9]. However, to the best of our knowledge, no one has worked out an intelligent system with a metric of sentence-by-sentence natural language expressions in a very generalized form, even if they knew it was very important. The authors studied the semantic metric application to story understanding[28]~[30], conversational text retrieval[24] and natural language generation problems[23].

The highlight of the SD-Form Semantics Model lies in a concrete definition of the semantic metric (or "semantic difference measure") on a concept set described by an SD-Form. We give, in this section, the principle of the semantic difference computation by showing an illustrative example.

The elaboration score between D_1 and D_2, as defined in Section 2, is interpreted as a measure of semantic difference between them. However, if they are not related directly, we cannot measure the difference by this score. In this section we introduce a general scheme to evaluate the semantic difference between any two concepts.

3.1 The Nearest Common Ancestor and the Semantic Difference Measure

Let D, D_1, D_2 be three concepts in SD-Form which satisfy:

$$\boldsymbol{ELAB}(D, D_1) = n_1 \text{ and}$$
$$\boldsymbol{ELAB}(D, D_2) = n_2$$

We call such a D "a common ancestor" of D_1 and D_2. A simple example of such D is the variable "X" because it is an ancestor of any SD-Form. D can be either $D = D_1$ or $D = D_2$, or both.

One of the common ancestors (D_0) which is the nearest to D_1 and D_2 is called the nearest common ancestor ("***NCOA***" in short) of D_1 and D_2. We describe this relation as

$$\boldsymbol{NCOA}(D_1, D_0, D_2, n_1, n_0, n_2) \qquad (3)$$

where,

$$\begin{aligned} n_0 &= n_1 + n_2 \\ &= \boldsymbol{ELAB}(D_0, D_1) + \boldsymbol{ELAB}(D_0, D_2) \\ &= \min_D\{\boldsymbol{ELAB}(D, D_1) \\ &\qquad + \boldsymbol{ELAB}(D, D_2)\} \end{aligned}$$

When we do not care about the score, we describe (3) as

$$\boldsymbol{NCOA}(D_1, D_0, D_2).$$

The "semantic difference measure" between two concepts in the SD-Form Semantics Model is defined by;

$$\boldsymbol{DIFF}(D_1, D_2) = n_0. \qquad (4)$$

This is the most important proposition in the model. n_0 is simply referred to as the semantic difference score. The following example illustrates this score's computation. The example is not very realistic, but will exemplify the computation scheme.

⟨Example 3-1⟩

Let D_1 and D_2 be two "given concepts."

$D_1 = [s(\text{PIERRE-II}),$
$\quad v(\text{DRINK/REGULARLY}),$
$\quad o(\text{WINE/RED})]$
(PIERRE-II always drinks red wine.)

$D_2 = [s(\text{TARO}),$
$\quad v(\text{BUY/PAST/YESTERDAY}),$
$\quad o(\text{DUD})]$
(Taro bought a DUD yesterday.)

The "Facts" in the system (Individual and General) are as follows.

$F_1 = (\text{PERSON})\mathit{incl}([\text{PRINCE, JAPANESE}])$
(A prince and a Japanese are persons.)

$F_2 = (\text{ALCOHOL})\mathit{incl}([\text{WINE, BEER}])$
(Wine and beer are alcohol.)

$F_3 = (\text{JAPANESE})\mathit{incl}(\text{TARO})$
(TARO is a Japanese.)

$F_4 = (\text{PIERRE-II})\mathit{equa}(\text{PRINCE/FRENCH})$
(PIERRE-II is the French prince.)

$F_5 = (\text{BEER/AMERICAN})\mathit{incl}(\text{DUD})$
(DUD is an American beer.)

$F_6 = D_1 = [s(\text{PIERRE-II}),$
$\quad v(\text{DRINK/REGULARLY}),$
$\quad o(\text{WINE/RED})]$

$F_7 = D_2 = [s(\text{TARO}),$
$\quad v(\text{BUY/PAST/YESTERDAY}),$
$\quad o(\text{DUD})]$

General "Rules" in the system are

$R_1 = (\mathit{assu}([s(\text{X}), v(\text{DRINK/}$
$\quad \text{REGULARLY}), o(\text{Y})]))$
$\quad \mathit{caus}([s(\text{X}), v(\text{LIKE}), o(\text{Y})]),$
(If X always drinks Y, then X likes Y.)

$R_2 = (\mathit{assu}([s(\text{X}), v(\text{BUY}), o(\text{Y})]))$
$\quad \mathit{caus}([s(\text{X}), v(\text{LIKE}), o(\text{Y})]).$
(If X buys Y, then X likes Y.)

Under this circumstance the two following concepts (induced concepts) work as system knowledge (specific rules) after instantiating general rules with facts.

$R'_1 = (\mathit{assu}([s(\text{PIERRE-II}),$
$\quad v(\text{DRINK/REGULARLY}),$
$\quad o(\text{WINE/RED})]))$
$\quad \mathit{caus}([s(\text{PIERRE-II}),$
$\quad v(\text{LIKE}), o(\text{WINE/RED})])$

$R'_2 = (\mathit{assu}([s(\text{TARO}), v(\text{BUY}), o(\text{DUD})]))$
$\quad \mathit{caus}([s(\text{TARO}), v(\text{LIKE}), o(\text{DUD})])$

By using this system knowledge, the ***NCOA*** of D_1 and D_2 is detected by the algorithm described in Sections 3.2–3.6. All the elaboration

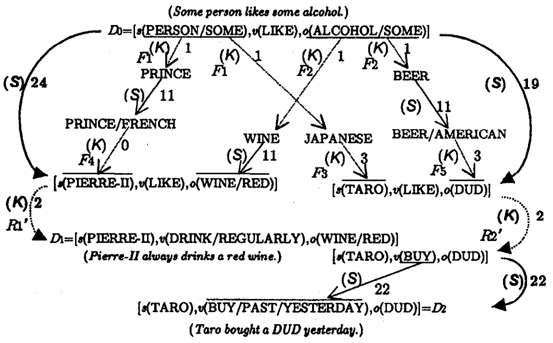

(Some person likes some alcohol.)
D0=[s(PERSON/SOME),v(LIKE),o(ALCOHOL/SOME)]
(K) 1 F1
(K) 1 F1
(K) 1 F2
(K) 1 F2
PRINCE
BEER
(S) 24
(S) 19
(S) 11
(S) 11
PRINCE/FRENCH
WINE
JAPANESE
BEER/AMERICAN
(K) 0 F4
(S) 11
(K) 3 F3
(K) 3 F5
[s(PIERRE-II),v(LIKE),o(WINE/RED)]
[s(TARO),v(LIKE),o(DUD)]
(K) 2 R1'
(K) 2 R2'
D1=[s(PIERRE-II),v(DRINK/REGULARLY),o(WINE/RED)]
(Pierre-II always drinks a red wine.)
[s(TARO),v(BUY),o(DUD)]
(S) 22
(S) 22
[s(TARO),v(BUY/PAST/YESTERDAY),o(DUD)]=D2
(Taro bought a DUD yesterday.)

Fig. 3 An example of the $\boldsymbol{DIFF}(D_1, D_2)$ computation.

relations are illustrated in **Fig. 3**. The numbers are elaboration scores. Knowledge symbols $(F_1, \cdots, F_7, R_1, R_2, R'_1, R'_2)$ are also attached to each (K). The overall elaboration from D_0 to D_1 takes 2 steps, while the one from D_0 to D_2 takes 3 steps. Therefore, the semantic difference measure in this case is computed as

$$\begin{aligned}&\boldsymbol{DIFF}(D_1, D_2)\\ &= \boldsymbol{ELAB}_2(D_0, D_1) + \boldsymbol{ELAB}_3(D_0, D_2)\\ &= (((1+11+0)+(1+11))+2)\\ &\quad + (((1+3)+(1+11+3))+2+22))\\ &= 26+43 = 69\,\text{semits}.\end{aligned}$$

If all these knowledge pieces are registered, our experimental system (**SDENV-2**) can execute the instantiation operations automatically, and finally compute the $\boldsymbol{DIFF}(D_1, D_2)$ score correctly.

As we see here, given concepts (D_1 and D_2) are always regarded as facts in score computation.

The idea to define $\boldsymbol{DIFF}(D_1, D_2)$ as the semantic difference measure between D_1 and D_2 has some similarity to the idea of "The Inferential Distance Ordering" proposed by D.S. Touretzky[27]. However, his idea is categorized in our model as a limited case when both D_1 and D_2 are labels. Our model covers more complex cases in natural language semantics.

It is our central concern whether or not $\boldsymbol{DIFF}(D_1, D_2)$ is computable in a practical system. The topic in the following four sections is to prove that it is computable by using "M-step elaboration relations" for any given M that is finite.

3.2 Type-wise *NCOA* Detection Algorithm

There are many types in the relation between D_0 and (D_1, D_2). We call them $\boldsymbol{NCOA}$-types. They can be described by the chain of (S) and (K) symbols. The simple types are as follows.

Type-a: D_1-(K)-D_0-(S)-D_2
$\boldsymbol{ELAB}_{\boldsymbol{know}}(D_0, D_1)$,
$\boldsymbol{ELAB}_{\boldsymbol{synt}}(D_0, D_2)$

Type-b: D_1-(S)-D_0-(S)-D_2
$\boldsymbol{ELAB}_{\boldsymbol{synt}}(D_0, D_1)$,
$\boldsymbol{ELAB}_{\boldsymbol{synt}}(D_0, D_2)$

Type-c: D_1-(K)-D_0-(K)-D_2
$\boldsymbol{ELAB}_{\boldsymbol{know}}(D_0, D_1)$,
$\boldsymbol{ELAB}_{\boldsymbol{know}}(D_0, D_2)$

As we see in the following, we can detect the nearest common ancestor (D_0) for these types. We denote such algorithm by $\boldsymbol{G}(1, D_1, D_0, D_2, n_0)$ which we call a "1-step algorithm."

The number of $\boldsymbol{NCOA}$-types in **SDENV-2** is equal to 100. It is the combination of 10 different $\boldsymbol{ELAB}(D_0, D_1)$ path types and 10 different $\boldsymbol{ELAB}(D_0, D_2)$ path types (c.f. Section 2.7).

Type-b above is very special in the sense that D_0 must be synthesized from D_1 and D_2, while D_0 in the other cases (Type-a and Type-c) can be detected from among all knowledge-heads in the system.

An example of D_0 synthesizing (c.f. Section 3.4) is illustrated in Fig. 3. In this case,

$$\begin{aligned}D_0 = [&s(\text{PERSON/SOME}), v(\text{LIKE}),\\ &o(\text{ALCOHOL/SOME})]\end{aligned}$$

is synthesized from the two concepts (D_{01} and D_{02}) which follow.

$$\begin{aligned}D_{01} = [&s(\text{PIERRE-II}), v(\text{LIKE}),\\ &o(\text{WINE/RED})]\\ D_{02} = [&s(\text{TARO}), v(\text{LIKE}), o(\text{DUD})]\end{aligned}$$

3.3 The Detection Algorithm of D_0 for D_1-(K)-D_0-(S)-D_2 Type *NCOA*

The D_1-(K)-D_0-(S)-D_2 type D_0 is detectable by the following.

(1) Find all knowledge-heads D_x's from all knowledge pieces in the system for which $\boldsymbol{ELAB}_{\boldsymbol{know}}(D_x, D_1, n_{1x})$ and $\boldsymbol{ELAB}_{\boldsymbol{synt}}(D_x, D_2, n_{2x})$ are satisfied for some n_{1x} and n_{2x}.

(2) Find a D_x for which $n_0 = n_{1x}+n_{2x}$ is the minimum. Such D_x is the desired D_0.
This algorithm is applicable to the D_1-(S)-D_0-(K)-D_2 type $\boldsymbol{NCOA}$ by symmetry.

3.4 D_0 Synthesis Algorithm for D_1-(S)-D_0-(S)-D_2 Type *NCOA*

For the D_1-(S)-D_0-(S)-D_2 type D_0, we have to synthesize such D_0 from the given (D_1, D_2) by referring to case-wise $\boldsymbol{ELAB}_{\boldsymbol{synt}}$

definitions. We actually made such an $\boldsymbol{NCOA}(D_1, D_0, D_2)$ program in **SDENV-2** for all combinations of every SD-Form type for D_1 and D_2. The details are beyond the scope of this paper. Some examples are in the following, where n_1, n_2 and n_0 are given by

$$\boldsymbol{ELAB_{synt}}(D_0, D_1) = n_1,$$
$$\boldsymbol{ELAB_{synt}}(D_0, D_2) = n_2,$$
$$n_0 = n_1 + n_2.$$

A. The case where D_1 and D_2 are SD-Forms of different types.
In this case there are no common ancestors other than a variable X.
$$D_0 = \mathrm{X},$$
$$n_0 = \boldsymbol{ELAB_{synt}}(\mathrm{X}, D_1) + \boldsymbol{ELAB_{synt}}(\mathrm{X}, D_2) = \boldsymbol{SI}(D_1) + \boldsymbol{SI}(D_2) - 2$$
(This is because we set $\boldsymbol{SI}(\mathrm{X}) = 1$ in **SDENV-2**.)

B. The case where D_1 and D_2 are simple labels which are not identical. This is the same as Case A.
$$D_0 = \mathrm{X}, n_0 = 18.$$
(This is because we set the $\boldsymbol{SI}$ score for a label to be 10.)

C. The case where $D_1 = D/D_{1x}, D_2 = D/D_{2x}$
We can recursively synthesize D_0 as $D_0 = D/D_{00}$, where
$$\boldsymbol{NCOA}(D_{1x}, D_{00}, D_{2x}, n_1, n_0, n_2).$$

D. The case where D_1 and D_2 are both statement SD-Forms such as
$$D_1 = [s(D_{11}), v(D_{12}), c(D_{13})],$$
$$D_2 = [s(D_{21}), v(D_{22}), i(D_{23}), o(D_{24})].$$
In this case D_0 is generated as
$$D_0 = [s(D_{01}), v(D_{02})]$$
$$\boldsymbol{ELAB_{synt}}(D_0, D_1, n_1),$$
$$\boldsymbol{ELAB_{synt}}(D_0, D_2, n_2),$$
where, D_{01} and D_{02} are recursively detected as
$$\boldsymbol{NCOA}(D_{11}, D_{01}, D_{21}) \text{ and}$$
$$\boldsymbol{NCOA}(D_{12}, D_{02}, D_{22}).$$

3.5 The Detection Algorithm for D_0 in a D_1-(K)-D_0-(K)-D_2 Type $NCOA$

The D_1-(K)-D_0-(K)-D_2 type D_0 is detectable by the following.

(1) Find all knowledge-head D_x's from all knowledge pieces in the system for which $\boldsymbol{ELAB_{know}}(D_x, D_1, n_{1x})$ and $\boldsymbol{ELAB_{know}}(D_x, D_2, n_{2x})$ are satisfied for some n_{1x} and n_{2x}.

(2) Find a D_x for which $n_0 = n_{1x} + n_{2x}$ is the minimum. Such D_x is the desired D_0.

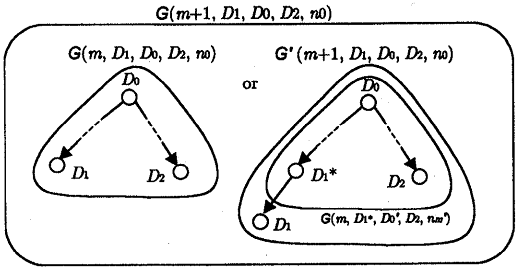


Fig. 4 $G(m+1, D_1, D_0, D_2, n_o)$.

As we have seen above, $\boldsymbol{NCOA}$ for Type-b in Section 3.2 is recursively synthetic, while other types (Type-a and Type-c) are all knowledge dependent.

3.6 General m-step Algorithm to Detect D_0

In Sections 3.3, 3.4 and 3.5, we have shown that for three simple $\boldsymbol{NCOA}$ types,

1. D_1-(K)-D_0-(S)-D_2,
2. D_1-(S)-D_0-(S)-D_2,
3. D_1-(K)-D_0-(K)-D_2,

the nearest common ancestor D_0 of (D_1, D_2) is detectable. We integrate those algorithms into one expression as $\boldsymbol{G}(1, D_1, D_0, D_2, n_0)$ which we call the 1-step algorithm. We generally designate an algorithm which detects D_0 for a given (D_1, D_2) by $\boldsymbol{G}(m, D_1, D_0, D_2, n_0)$, where m shows the maximum number of elaboration steps from D_0 to either D_1 or D_2. We call this algorithm the "m-step algorithm."

In this section we show that we can build an $(m+1)$-step algorithm by using the m-step algorithm.

$\boldsymbol{G}(m+1, D_1, D_0, D_2, n_0)$ consists of two parts (**Fig. 4**.) The first part is $\boldsymbol{G}(m, D_1, D_0, D_2, n_0)$ itself, the second part is an extended algorithm of $\boldsymbol{G}(m, D_1, D_0, D_2, n_0)$ which we denote
$$\boldsymbol{G}'(m+1, D_1, D_0, D_2, n_0).$$

In the extending process from $\boldsymbol{G}(m, D_1^*, D_0, D_2, n_0)$ to $\boldsymbol{G}'(m+1, D_1, D_0, D_2, n_0)$, it is enough to consider the following three cases, where (D_1, D_2) is the given concept pair. This is because the D_1-(S)-D_1^*-(S)-combination never appears in a multi-step elaboration (c.f. Section 2.6.4).

Case-1: D_1-(S)-D_1^*-(K)- $\cdots$ -D_0-$\cdots$-D_2
Case-2: D_1-(K)-D_1^*-(S)- $\cdots$ -D_0-$\cdots$-D_2
Case-3: D_1-(K)-D_1^*-(K)- $\cdots$ -D_0-$\cdots$-D_2

In every case we assume an elaboration from D_0 to D_1^* actually takes m-steps.

```
Algorithm: $G'(m+1, D_1, D_0, D_2, n_0)$
[For Case-1]
(1) Pick up a knowledge-tail $D_1^*$ in the system, and check if $\boldsymbol{ELAB_{synt}}(D_1^*, D_1, n_1')$ is computable.
(2) If it is computable Execute (3), else Execute (4).
(3) Execute $\boldsymbol{G}(m, D_1^*, D_0', D_2, n_m')$. Set $n_0' = n_m' + n_1'$. Store $D_0'$ (together with $n_0'$) as a candidate for $D_0$.
(4) Execute (1) for another $D_1^*$. If no such $D_1^*$ is left, Execute (5).
[For Case-2 and 3]
(5) Pick up a $D_1^*$ satisfying $\boldsymbol{ELAB_{know}}(D_1^*, D_1, n_1')$ and Execute (6).
    If no such $D_1^*$ exists, Execute (8).
(6) Execute $\boldsymbol{G}(m, D_1^*, D_0', D_2, n_m')$. Set $n_0' = n_m' + n_1'$. Store $D_0'$ (together with $n_0'$) as a candidate for $D_0$.
(7) Execute (5) for another $D_1^*$.
(8) If no candidates are stored, set $D_0 = \mathrm{X}$, else Execute (9).
[Minimization]
(9) Find out a candidate $D_0'$ for which $n_0'$ is the smallest. Set $D_0 = D_0'$ and halt.
```

```
Algorithm: $\boldsymbol{G}(m+1, D_1, D_0, D_2, n_0)$
(1) Execute $\boldsymbol{G}(m, D_1, D, D_2, n_m)$.
(2) Execute $\boldsymbol{G'}(m+1, D_1, D', D_2, n_{m+1})$.
(3) Set $n_0 = \min\{n_m, n_{m+1}\}$, and set $D_0$ to $D$ or $D'$, whichever gives $n_0$.
```

Thus, we can build an M-step algorithm $\boldsymbol{G}(M, D_1, D_0, D_2, n_0)$ by starting from $\boldsymbol{G}(1, D_1, D_0, D_2, n_0)$ and extending it to $\boldsymbol{G}(2, D_1, D_0, D_2, n_0)$, $\boldsymbol{G}(3, D_1, D_0, D_2, n_0)$, $\cdots$, and to an M-step algorithm for any M which is finite. By using $\boldsymbol{G}(M, D_1, D_0, D_2, n_0)$ we can detect the nearest common ancestor D_0 for a given D_1 and D_2.

Once the $\boldsymbol{G}(M, D_1, D_0, D_2, n_0)$ algorithm is put into a program and run, it recursively calls the lower step programs automatically. It always comes down to one of the three simple 1-step programs (c.f. Section 3.3–3.5).

The algorithm shown here is more theoretical than practical. Its purpose is only to prove that we can compute $\boldsymbol{NCOA}$ for the given D_1 and D_2.

As we have seen so far, $\boldsymbol{G}(M, D_1, D_0, D_2, n_0)$ is a recursive algorithm, and it is knowledge dependent. Therefore, executing $\boldsymbol{G}(M, D_1, D_0, D_2, n_0)$ is rather time-consuming. However, even if the system knowledge changes, the operation of $\boldsymbol{G}(M, D_1, D_0, D_2, n_0)$ can adapt to it without any change in the algorithm.

4. Concluding Remarks

The elaboration relation is the most fundamental idea in the SD-Form Semantics Model. It is either a syntactic or knowledge based one. The degree of elaboration from one concept (D_1) to another (D_2), i.e., the elaboration score, provides the semantic discrepancy between them. It is a generalized idea of the traditional "IS-A", "PART-OF" and "IF-THEN" relations.

Based on the idea of elaboration, we gave the computation algorithm for the semantic difference measure. It is denoted by $\boldsymbol{DIFF}(D_1, D_2)$, where D_1 and D_2 are given concepts in terms of the SD-Form. This works as the metric in the SD-Form Semantics Model and provides us with a framework for quantitative analysis of meaning in natural language understanding.

The $\boldsymbol{DIFF}(D_1, D_2)$ score depends on knowledge, and every time the knowledge increases, the semantic difference measure must be revised. However, it is possible for a system to keep up with knowledge revision without any revision in the system program. This relieves us from worrying about all the knowledge maintenance problems. We can give knowledge data to the system piece-by-piece almost independently and randomly.

In this paper we have focused our discussion on the semantic difference score evaluation scheme between two concepts. Some reader may wonder how useful this model is, and suspect its practical application. The soundness of the model might be another point of discussion.

However, we believe such questions can only be answered through experimental research. Presently, we are working on the following projects in conjunction with SD-Form Semantics Model.

(1) Question answering system[28]
(2) Story understanding system[14],[29]
(3) Self organization process of concept hierarchy[11],[22]
(4) English and Japanese sentence generation[23],[32]
(5) Retrieval of multimedia data[33]

All these projects are more experimental than theoretical. We will soon become more explicit about soundness and usefulness of the SD-Form Semantics Model through these research projects.

The merits of the SD-Form Semantics Model are the following.

(1) It gives a well-formed syntactic structure for meaning description.
(2) The syntactic difference of two SD-Forms expresses the semantic difference.
(3) It is an interlingua reflecting the basic structure of natural language such as English and Japanese.
(4) It is easy to learn SD-Form usage in the meaning description.

The algorithm of the semantic difference computation will open many ways to the meaning processing of natural language, such as in recognition, understanding and learning.

Some of the problems we should work on in the near future are the following.

(1) Testing the model in a more realistic world, even if it is small.
(2) An abstracting algorithm of story data written by SD-Forms.
(3) To make our **SDENV-2** a faster system.

Acknowledgments This work was financially supported in part by the SCAT Foundation. The authors express their deep gratitude to Prof. Rick Eason for his critical reading of the manuscript.

References

1) Barros, J., French, J., Martin, W. Kelly, P. and Cannon, M.: Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval, *Proc. SPIE-Int. Soc. Opt. Eng.* (USA), Vol.2670, pp.392–403 (1996).
2) Bossu, G. and Siegel, P.: Saturation, Nonmonotonic Reasoning and the Closed-World Assumption, *Artif. Intell.*, 25, pp.13–63 (1985).
3) Blachman, R.J., Levesque, H.J. and Reiter, R.: Special Volume on Knowledge Representation, *Artif. Intell.* (1991).
4) Blachman, R.J. and Schmolze, J.G.: An overview of the KL-ONE knowledge representation system, *Cognitive Science*, Vol.9, pp.171–216 (1985).
5) Cardie, C.: A case-based approach to knowledge acquisition for domain-specific sentence analysis, *Proc. Eleventh Natinal Conference on Artificial Intelligence*, pp.793–803 (1993).
6) Clocksin, W.F. and Mellish, C.S.: *Programming in Prolog*, Third Revised and Extended Edition, Springer-Verlag, Berlin, Heidelberg, New York (1987).
7) Cohen, P.R. and Perrault, C.R.: Elements of a plan-based theory of speech acts, *Cognitive Science*, Vol.3, pp.177–212 (1979).
8) Cranias, L., Papageorgiou, H. and Piperidis, S.: Clustering: A technique for search space reduction in example-based machine translation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.1, pp.1–6 (1994).
9) Jantke, K.P. and Lange, S.: Case-based representation and learning of pattern languages, *Theor. Comput. Sci.*, Vol.137, pp.25–51 (1995).
10) Kawaguchi, E., Wakiyama, M. and Nozaki, K.: A Semantic Structure Description Model of General Concepts in a Natural Language World, *Proc. PRICAI*, pp.298–303 (1990).
11) Kawaguchi, E., Kamata, S. and Wakiyama, M.: Elaboration Relation and the Nearest Common Ancestor of a Concept Pair in the SD-Form Semantics Model, *Proc. 2nd PRICAI*, pp.426–432 (1992).
12) Kawaguchi, E., Lee, M. and Nozaki, K.: Meaning description by SD-Forms and a prototype of a conversational-text retrieval system, *Proc. TAI93*, Boston, pp.306–310 (1993).
13) Kawaguchi, E., Wakiyama, M., Nozaki, K. and Shao, G.: How to describe the Meaning of Natural-Language Sentences and System Knowledge by Using SD-Forms, *Proc. NLPRS'93*, pp.380–386 (1993).
14) Kawaguchi, E., Wakiyama, M., Lee, M. and Shao, G.: A Framework of Story Understanding in the SD-Form Semantics Model, *Information Modeling and Knowledge Bases V*, Kangassalo, H., et al., (Eds.), IOS (1994).
15) Kawaguchi, E., Kamata, S. and Wakiyama, M.: The Semantic Metric Computation Scheme in the SD-Form Semantics Model, *Proc. 3rd PRICAI*, pp.623–629 (1994).
16) Kawaguchi, E. and Wakiyama, M.: Some Topics on the SD-Form Semantics Model, *Proc. NLPRS'95*, Poster Session China (1995).
17) Kodratoff, Y. and Tecuci, G.: Learning based on conceptual distance, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.10, pp.897–909 (1988).
18) Lebowitz, M.: Generalization from natural language text, *Cognitive Science*, Vol.7, pp.1–40 (1983).
19) McAllester, D.A. and Givan, R.: Natural language syntax and first-order inference, *Artif. Intell.*, Vol.56, pp.1–20 (1992).
20) Rada, R., Mili, H., Bicknell, E. and Blettner, M.: Development and application of a metric on semantic nets. *IEEE Trans. System, Man, and Cybernetics*, Vol.19, pp.17–30 (1989).
21) Schank, R.C. and Leake, D.B.: Creativity and learning in a case-based explainer, *Artif. Intell.*, Vol.40, pp.353–387 (1989).
22) Shao, G. and Kawaguchi, E.: A Self Organization Process of Concept Hierarchy in the SD-Form Semantics Model, *Proc. 5th International*

Symposium on Artificial Intelligence, pp.393–400 (1992).

23) Shao, G., Wakiyama, M. and Kawaguchi, E.: A Prototype of English Sentence Generation System Based on SD-form Semantics Model, *Proc. 1st PACLING*, pp.261–268 (1993).

24) Shao, G., Nozaki, K., Kamata, S. and Kawaguchi, E.: SD-Forms as interlingua and a prototype of a conversational-text retrieving system, *J. Japanese Society for Artificial Intelligence*, Vol.9, No.5, pp.684–693 (1994).

25) Shapiro, S.C. and Rapaport, W.J.: SNePS Considered as a fully Intentional propositional semantic network, Cercone, N. and McCalla, G. (Eds.), *The Knowledge Frontier: Essays in the Representation of Knowledge*, pp.262–315, Springer, New York, (1987).

26) Sowa, J.F.: *Conceptual Structure: Information Processing in Mind and Machine*, Addison-Wesley, Reading, MA (1984).

27) Touretzky, D.S.: *The Mathematics of Inheritance Systems*, Pitman Publishing (1986).

28) Wakiyama, M., Nozaki, K. and Kawaguchi, E.: Semantic processing of story data by the SD-Form model approach, *Proc. NLPRS'91*, pp.288–295 (1991).

29) Wakiyama, M., Shao, G., Nozaki, K. and Kawaguchi, E.: Anaphora Processing of Story Data by the SD-Form Semantics Model Approach, *Proc. PRICAI*, pp.1169–1175 (1992).

30) Wakiyama, M., Shao, G., Nozaki, K. and Kawaguchi, E.: Anaphora Resolution by Machine in the SD-Form Semantics Model, *Proc. 2nd NLPRS'93*, pp.256–263 (1993).

31) Wakiyama, M., Shao, G., Nozaki, K. and Kawaguchi, E.: The Toward Generalization of the Semantic Metric in the SD-Form Semantics Model, *Proc. 4th PRICAI'96 Poster*, pp.61–68 (1996).

32) Wakiyama, M., Yoshihara, S. and Kawaguchi, E.: A Prototype of Japanese Sentence Generation System from SD-Formed Meaning Data, *Proc. PACLING'97*, pp333–344 (1997).

33) Wakiyama, M. and Kawaguchi, E.: Retrieval of Multimedia Data for Natural Language in the Network, *Proc. 4th NLPRS'97 Poster*, pp.605–608 (1997).

34) Wilensky, R.: *PAM, Inside Computer Understanding*, pp.136–179 (1981).

Appendix

The syntax of the SD-Form is defined by the following context-free grammar SDG.

$\mathrm{SDG} = \{\Phi, \mathrm{P}, \Sigma_n, \Sigma_t\}$

Parameters in SDG are as follows.

Φ: Start symbol

P: Set of generation rules

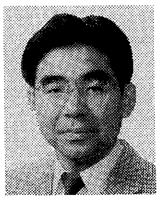
$\Sigma_n = \{\Phi, \mathrm{X}_1, \cdots, \mathrm{X}_n\}$: Non-terminals

$\Sigma_t = \{f_1, f_2, \cdots, f_i, f_j, \cdots, g_1, g_2, \cdots, g_i, \cdots, /,$ *nega*, *pass*, *even*, *only*, *assu*, *fcus*, *incl*, *excl*, *equa*, *defi*, *andx*, *orxx*, *eorx*, *caus*, *indu*, *effe*, *bcau*, *pseq*, *nseq*, *inst*, *addi*, *plus*, *than*, *ofal*, *exmp*, *siml*, *asas*, *asif*, *evif*, *csof*, *ptof*, *prof*, *vaof*, *kdof*, *thru*, *more*, *less*, *oppo*, *para*, *s*, *v*, *i*, *o*, *c*, *b*, *a*, *r*, *e*, [,], (,), "," }: Terminals

P is defined as follows.

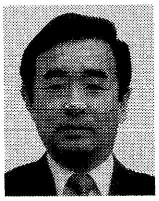
0.	$\Phi \to \mathrm{X}$	(1)
10.	$\mathrm{X} \to \mathrm{X}_1(\mathrm{X}_2, \cdots, \mathrm{X}_n)$	(0)
20.	$\mathrm{X} \to f_i$	(9)
30.	$\mathrm{X} \to g_i(f_j)$	(19)
41.	$\mathrm{X} \to g_i/\mathrm{X}$	(11)
42.	$\mathrm{X} \to g_i(f_j)/\mathrm{X}$	(11)
43.	$\mathrm{X} \to (f_1)plus(f_2) \cdots plus(f_n)/\mathrm{X}$	$(11n)$
44.	$\mathrm{X} \to g_i/(\mathrm{X}_1)para(\mathrm{X}_2) \cdots para(\mathrm{X}_n)$	$(2n+9)$
45.	$\mathrm{X} \to g_i(f_j)/(\mathrm{X}_1)para(\mathrm{X}_2) \cdots para(\mathrm{X}_n)$	$(2n+19)$
46.	$\mathrm{X} \to (f_1)plus(f_2) \cdots plus(f_m)/(\mathrm{X}_1)para(\mathrm{X}_2) \cdots para(\mathrm{X}_n)$	$(11m+2n-2)$
50.	$\mathrm{X} \to P_i(\mathrm{X})$	(2)
60.	$\mathrm{X} \to (\mathrm{X})C_i(\mathrm{X})$	(2)
61.	$\mathrm{X} \to (\mathrm{X}_1)plus(\mathrm{X}_2) \cdots plus(\mathrm{X}_n)$	$(2n-2)$
62.	$\mathrm{X} \to (\mathrm{X})C_i([\mathrm{X}_1, \cdots, \mathrm{X}_n])$	$(n+1)$
70.	$\mathrm{X} \to [s(\mathrm{X}), v(\mathrm{X})]$	(4)
71.	$\mathrm{X} \to [s(\mathrm{X}), v(\mathrm{X}), o(\mathrm{X})]$	(6)
72.	$\mathrm{X} \to [s(\mathrm{X}), v(pass(\mathrm{X})), b(\mathrm{X})]$	(8)
73.	$\mathrm{X} \to [s(\mathrm{X}), v(\mathrm{X}), c(\mathrm{X})]$	(6)
74.	$\mathrm{X} \to [s(\mathrm{X}), v(\mathrm{X}), i(\mathrm{X}), o(\mathrm{X})]$	(8)
75.	$\mathrm{X} \to [s(\mathrm{X}), v(pass(\mathrm{X})), i(\mathrm{X}), b(\mathrm{X})]$	(10)
76.	$\mathrm{X} \to [s(\mathrm{X}), v(pass(\mathrm{X})), o(\mathrm{X}), b(\mathrm{X})]$	(10)
77.	$\mathrm{X} \to [s(\mathrm{X}), v(\mathrm{X}), o(\mathrm{X}), c(\mathrm{X})]$	(8)
78.	$\mathrm{X} \to [s(\mathrm{X}), v(pass(\mathrm{X})), c(\mathrm{X}), b(\mathrm{X})]$	(10)
79.	$\mathrm{X} \to [s(\mathrm{X}), v(pass(\mathrm{X})), b(\mathrm{X}), c(\mathrm{X})]$	(10)
80.	$\mathrm{X} \to [a(\mathrm{X})]$	(2)
81.	$\mathrm{X} \to [r(\mathrm{X})]$	(2)
82.	$\mathrm{X} \to [e(\mathrm{X})]$	(2)

(Received November 7, 1997)
(Accepted December 7, 1998)

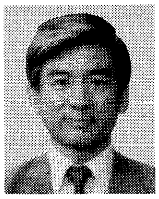


Masahiro Wakiyama was born in Fukuoka, Japan in 1954. He received the B.E. and M.E. degrees in 1979,1990, respectively, from Kyushu Institute of Technology, Fukuoka, Japan. He is currently a Ph.D. candidate in Computer Engineering Department of Kyushu Institute of Technology. Since 1991 he has been with Kitakyushu National College of Technology, where he is a associate professor in the Department of Control & Information Systems Engineering. His research interests are mainly related to artificial intelligence. In particular, he has been working in the field of natural language processing. He is a member of IPSJ and JSAI of Japan.

Hideki Noda received B.E., M.E. degrees in electronics engineering from Kyushu University, Japan, in 1973 and 1975 respectively, and Dr.Eng. degree in electrical engineering from Kyushu Institute of Technology, Japan, in 1993. He worked in Daini-Seikosha Ltd. from 1975 to 1978, in National Research Institute of Police Science, Japan National Police Agency from 1978 to 1989 and then in Communications Research Laboratory, Japan Ministry of Posts and Telecommunications from 1989 to 1995. From 1984 to 1985, he was a visiting scientist in National Research Council of Canada. In 1995, he moved to Kyushu Institute of Technology where he is now an associate professor in the Department of Electrical, Electronic & Computer Engineering. His research interests include speaker and speech recognition, image processing and neural networks. He is a member of the IEICE and the Acoustical Society of Japan.



Koichi Nozaki has graduated from the Faculty of Electrical Engineering of Kyushu University in 1975. He then worked at the Computer Center of Nagasaki University as a research associate. He is currently a lecturer of the Information Science Center, Nagasaki University, Nagasaki, Japan. He has contributed in his university students education for computer science and network system, management of the large computer and network system. His research interests are on Image Processing and Knowledge Information Processing. He is a member of IPSJ and ACM.



Eiji Kawaguchi received a Doctor of Engineering Degree in 1972 from Department of Electronics Engineering at Kyushu University, Japan. He started his academic career at universities from 1969 in engineering. Currently he is a professor of Electrical, Electronics and Computer Engineering Department at Kyushu Institute of Technology. His research interests include speech recognition, pattern understanding, image processing, knowledge engineering, information hiding, as well as natural language processing and semantics modeling.