Text-Driven Multiple-Path Discourse Processing for Descriptive Texts

Jungyun Seo

Abstract

This paper presents a text-driven discourse analysis system, called DPAS. DPAS constructs a discourse structure by weaving together clauses in the text by finding discourse relations between a clause and the clauses in a context. The basic processing model of DPAS is based on the stack based model of discourse analysis suggested by Grosz and Sidner. We extend the model with dynamic programming method to handle various discourse ambiguities effectively and efficiently. We develop the idea of a context space to keep all information of a context. DPAS parses a text by considering all possible discourse relations between a clause and a context. Since different discourse relations may result in different states of a context, DPAS maintains multiple context spaces for an ambiguous text. Since maintaining all interpretations until the whole text is processed requires too much computing resources, DPAS uses the idea of depth-limited search to limit the search space. If there is more than one discourse relation between an input clause and a context, DPAS constructs context spaces—one context space for each discourse relation. Then, DPAS applies heuristics to choose the most desirable context space after it processes some more input clauses. Since the basic idea of DPAS is domain independent, although we used descriptive texts to demonstrate DPAS, we believe the idea of DPAS can be extended to understand other styles of texts.

I. Introduction

This paper presents an experimental system for discourse analysis that constructs a discourse structure of a descriptive text. Constructing a discourse structure for a text has practical importance for making summaries and answering questions about the text. In particular, discourse analysis of *descriptive texts* such as texts from an encyclopedia aims toward an important application: extracting knowledge from a text.

However, discourse analysis has been known to be a difficult task that requires multiple levels of knowledge and complex inference mechanisms. Besides syntactic and semantic knowledge to understand each sentence, we need domain knowledge and a sort of meta-knowledge about text structure to understand a whole text. Furthermore, since a text is a linguistic realization of an author's intention, we need knowledge to explain some tricky linguistic phenomena such as pronominalization of a topic in a discourse, gaps, and various anaphora to understand a text properly.

Manuscript received July 31, 1995; accepted October 6, 1995.

The author is with Department of Computer Science, Sogang University, Seoul, Korea.

This work was supported in part by the KOSEF under the contract 94-1400-01-01-3

Such diverse problems cause many researches in discourse analysis to concentrate on some particular aspect, such as focus tracking, plan recognition and tracking, defining coherence relations between clauses, etc. Grosz and Sidner [7], and Polanyi [13] have suggested a fairly general framework for discourse analysis. We have attempted to implement a discourse analysis system for understanding descriptive texts using the framework. We found, however, that this framework needs to be extended to handle discourse ambiguities.

In this paper, we suggest an implementation model of text-driven discourse analysis based on the stack based model with the dynamic programming technique. We can view a discourse process as a system which constructs a discourse structure of a text by weaving together clauses in the text by finding relations between a clause and the clauses in a context. Therefore, the following two problems should be addressed:

1. What kind of relationships should be found to relate a clause to others in a context? How can they be recognized?
2. How can the overall process be controlled to determine the relationships? How can a context be represented? How can the process handle situations in which there are

multiple discourse relations between a clause and a context?

In this paper, we attempt to answer the above questions. We use domain specific relations to relate clauses in a text since such relations are far more effective to generate summaries and answer the questions about texts than general, domain independent rhetorical relations. We call such relations **discourse relations**. Discourse relations can be determined by using a variation of conventional semantic network techniques.

For the second question, we develop the idea of a *context space* for DPAS to keep all information of a context. DPAS parses a text by considering all possible discourse relations between a clause and a context. Since different discourse relations may result in different states of a context, DPAS maintains multiple *context spaces* for an ambiguous text. Since maintaining all interpretations until the whole text is processed requires too much computing resources, DPAS uses the idea of *depth-limited* search to limit the search space. If there is more than one discourse relation between an input clause and a context, DPAS constructs context spaces——one context space for each discourse relation. Then, DPAS applies heuristics to choose the most desirable context space after it processes some more input clauses. Since the basic idea of DPAS is domain independent, although we used descriptive texts to demonstrate DPAS, we believe that the idea of DPAS can be extended to understand other styles of texts.

II. Discourse Relations in Descriptive Texts

There are many different ways to define discourse relations for descriptive texts. Some linguistic research attempts to define a set of rhetorical coherence relations to cover all different styles of texts [9] [10] [12]. However, there has been little agreement on the kind of relationships, or even if such a global set of coherence relations can be defined.

The issue of what kind of discourse relations will be used is affected by the purpose of the resulting discourse structure. Our goal in text understanding is to construct a discourse structure within representational conventions that is sufficient for summarizing and direct question-answering applications.

We use the names of the properties of topics in a domain as the discourse relations for descriptive texts in the domain since: (1) a clause in a descriptive text usually explains a property of a topic in the text, (2) by using domain specific discourse relations——*e.g.*, "symptom" or "treatment" for disease description texts——to construct a discourse structure, a question/answering system can make intelligent responses with the resulting discourse structure without making complicated inferences. [16]

DPAS uses domain specific relations to relate clauses in a text. Domain specific relations depend on the style of texts as well as the domain of the texts. Different styles of text need different relations. For example, in narratives, relations like **setting*, **episode*, and the relations from planning such as **goal*, **pre-condition*, **post-effect*, **subaction*, etc. are effective for representing the meaning of a story. With these relations, we can easily paraphrase the story and answer typical questions about the story such as, "Why did John kill Mary?".

In a descriptive text, we want to use specific properties of topic items in the domain of the text as discourse relations in order to make meaningful summaries of the text and to generate specific answers for questions like "What is the *usage* of nickel?", "What is the *color* of Aluminum?", or "How does one *treat* the patient of infectious hepatitis?".

Domain specific discourse relations can be determined from the semantic structures of clauses with the domain specific knowledge. The basic inferencing algorithm is essentially the same as that of NEXUS in [1]; find possible connections between two concepts(words) in the knowledge base and check augmented constraints which usually consists of rules of checking arguments and/or attributes of the concepts.

With a sentence in an input text, the system makes a syntactic graph and then a semantic graph as described in [15] and [14] respectively. Then, DPAS determines all possible discourse relations using a knowledge base. [17]

III. Focus of Attention in a Discourse

DPAS relates clauses to a context by checking possible discourse relations between the concept of the predicate of the clause and the concept of a word in a context. Then the next question might be how to choose the pair of words to be checked by DPAS. If DPAS were to check all possible pairs of the predicate of a clause and words from a context, then its complexity would be unbearably high. Therefore, we adopt the idea of tracking *it focus* of a discourse to reduce the search space when relating a clause to a context. [18]

The idea of tracking focus of a discourse using a stack-like structure has been studied by by Grosz and Sidner [7] and by [13].[1] The basic idea is that there is an active part of a context and readers pay more attention to the active part when they read a clause. The active part of a context changes as readers read the text. We refer to this active part of a context as the focus of the discourse. Readers consider

1) Although Polanyi uses a right extending tree rather than a stack, the operational concept of a tree is the same as that of a stack.

only the focused part of the context when they relate a clause to the context. If he cannot relate the clause to the focused part, he shifts the focus of attention to the next most focused part of the context. By tracking the focus of attention in a discourse, DPAS can limit its search when it attempts to relate a clause to a context.

The following questions must be addressed to use the idea of the focus of a discourse in DPAS: (1) What is the unit of a focus in a text? and (2) How does one determine when the focus of a discourse is shifted? These questions are directly related to the problem of discourse segmentation. A discourse consists of **discourse segments**.

A discourse segment is a coherent piece of text. One clause may be a discourse segment. Several clauses can be combined into one segment. Although the need for discourse segmentation is suggested by several researchers [7] [13] [5], there is little consensus on how to segment a text into related discourse segments.

We believe that the segmentation of a text depends on the style of the text. For example, in narratives or dialogues, clauses which develop the same plan can be combined into one segment. If a clause cannot be related to the plan in which some of its preceding clauses are related, the clause signals a discourse jump---maybe starting a new plan or returning to one of the old plans which is interrupted. [7] [11]

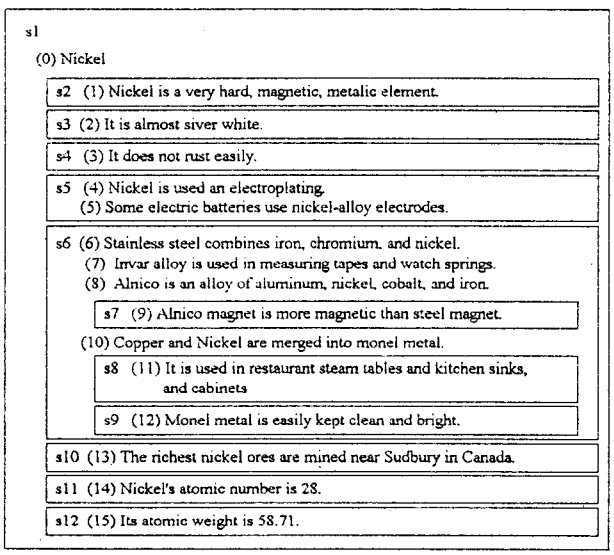

Fig. 1. A segmentation of a nickel text.

In descriptive texts, if clauses explain the same property of the same topic, we can group them together into a discourse segment. For example, Figure 1 shows a segmentation of a nickel text from [20]. The segmentation is based on topic words and their properties. Segment **s2** to **s6** and **s10** to **s12** explain properties of nickel, while segment **s7** explains a property of alnico magnets, and **s8** and **s9** describe properties of monel metal. Clauses (4) and (5) are combined into segment **s5** because both clauses explain **usage* of nickel. Since clauses (6), (7), (8), and (10) explain **alloy* of nickel, they can be combined into one segment.

Notice that a discourse structure, in our view, is a set of discourse segments and the relations among the segments. Therefore, the problem of how to construct a discourse structure from a text can be reduced to the problem of how to segment the text.

When DPAS reads a clause, DPAS must decide if the clause continues the preceding segment, starts a new segment, or continues an older segment. In descriptive text, if the clause explains the same property of the same topic of the preceding segment, the clause is regarded as a continuation of the preceding segment. If the clause explains a property of a new topic, then it is regarded as starting a new segment.

In most texts, a clause usually can be related to the preceding segment, either as continuing the segment or as starting to explain a new topic which was just introduced in the preceding segment. However, not all clauses can be related to the preceding segment. Sometimes a clause explains old topics in one of the prior discourse segments, either as continuing the prior segment or as starting to explain a topic which was introduced in the prior segment. In this case, the focus of attention must be shifted to the prior discourse segment.

For example, in Figure 1, clause (9) starts a new segment because it explains a property of alnico in clause (8). However, since clause (10) cannot be included in segment **s7**, DPAS must be able to shift its focus of attention into the prior segment, **s6**, and relate the clause (10) as a continuation of the segment **s6**.

To do this, DPAS also uses a focus stack(FS) [7], to keep the prior segments in the order of recency. Each space in an FS is a discourse segment. DPAS performs a combination of three operations—*mix*, *push*, and *pop*—as described in the following when it relates an input clause to an FS.

- Whenever DPAS reads a clause, it checks the segment in the top of an FS which is the current focus of attention. If the clause can be a continuation of the discourse segment, *i.e.*, the clause explains the same property of the same topic as the clauses in the segment, DPAS *mixes*, *i.e.*, adds, the clause into the discourse segment in the top. A mix operation does not change the top segment of an FS, but the mixed clause becomes a part of the top segment.
- If the clause explains a property of a newly introduced topic in the discourse segment in the top of an FS, the clause starts a new discourse segment and DPAS *pushes* the clause into the FS. As a result of the push operation,

the new segment becomes a new focus of attention for the following input clauses.

- If the clause cannot be related to the discourse segment in the top of an FS, DPAS *pops* the top segment from the FS and relates the clause to the new top of the popped FS. As a result of the pop operation, the popped segment is *closed* and becomes unreachable.[2] The information of popped segment is kept as a partial discourse structure which represents the discourse of the segment. Therefore, the information of the status of a context can be represented in a focus stack and the partial discourse structures of popped segments.

In this way, DPAS can segment a text using the three operations of a FS; pushes a clause to start a new segment, mixes a clause into a focus segment to enlarge the segment, and pops a segment to close it.

IV. DPAS: a multiple-path discourse parser

When DPAS relates a clause to the FS, if more than one discourse relation is found, the simple FS manipulation algorithm is not enough to construct a discourse structure. DPAS uses the idea of an *all-path parser* to handle multiple discourse relations for a clause. Given a context and an input clause, DPAS constructs a new context by adding the input clause to the old context by finding a discourse relation. If there is more than one discourse relation for the input clause, DPAS constructs new contexts using the discourse relations——one new context for each discourse relation. Since different contexts have different discourse relations, each context maintains its own FS. Some contexts can be developed further so that all remaining clauses in an input text can be related. Some contexts may fail to lead to complete discourse analyses of the whole input text and will be removed naturally when DPAS finds no more clauses can be related to those contexts.

Only contexts which lead to final contexts that include all clauses of an input text will be considered as complete discourse analyses of the text. Since there may be more than one possible way to construct final contexts, DPAS uses heuristics to choose one discourse analysis for the input text.

Since bringing all alternative discourse interpretations to the end of an input text results in huge search spaces, DPAS uses the idea of *depth-limited search* to prune contexts. In most cases, ambiguous relations for a clause to a context can be resolved after processing more input clauses. Therefore, DPAS uses a *depth-limited search* so that it can choose the best discourse relations by applying heuristics after processing additional input clauses.

Whenever there is more than one relation for a clause, DPAS generates all search spaces using the relations and continues processing the next input clause. After DPAS processes a certain number, **depth limit**, of input clauses, it chooses one relation for the clause. In other words, if the depth limit is M, then when there are multiple search spaces after processing the Nth clause, DPAS will choose one among those search spaces for the *Nth* clause after processing the $(N+M-1)th$ clause.

If the depth limit were the same as the number of the clauses in an input text, DPAS would construct all possible discourse structures of the text. If the depth limit were zero, DPAS would immediately choose one discourse relation whenever there were multiple discourse relations. According to our experiments, DPAS constructs an acceptable discourse structure when the depth limit is three. In the following subsections, we discuss what kind of information of a context must be found and used by DPAS, and then we describe heuristics that DPAS uses to prune out less plausible interpretations.

1. Information in a context for DPAS

Whenever DPAS relates a clause to a context, it constructs new contexts. We refer to such contexts as **context spaces**. From now on, we distinguish the terms *context* and *context space*: *context* represents the real context and a *context space* represents a context constructed by DPAS. A context space is a data structure for representing a context. Context spaces are the main search spaces for DPAS. Each context space must contain all information about the context so that DPAS can use the information when it relates the next input clause to the context. The primary information of a context are an FS and a partial discourse structure constructed up to the last clause of the context. Since different context spaces may have different FS's for the same text, the information of each context space must be local to the context space so that the information cannot be used in other context spaces. Let us use the following segment of the nickel text to illustrate what kind of information must be local to each context space.

(1-1)

(0). Nickel.
(1). Nickel is a very hard metallic element.
(2). It is almost silver white.
(3). It does not rust easily.

The development of context spaces as DPAS processes text (1-1) is illustrated in Figure 2. There are five context

2) In dialogues, somtimes a closed segment must be re-activated to become a focus. In that case, cue phrases must be used to signify re-focusing the old closed segment. Since descriptive texts are, however, usually well organized, it is hard to find such cue phrases.

spaces numbered 1 to 5 in the figure. The first one at the top of the figure represents the context space of clause 0 and 1. There is a binding list, "nickel(1) = nickel(0)", below the FS of context space 1. We call it the *referent binding list*. This binding means the token nickel in clause (0) is the referent of the token nickel in clause (1).[3] Whenever DPAS determines the referent of an anaphor, it adds the referent-anaphor pair into a referent binding list. A referent binding list is determined and constructed as DPAS relates clause into the context. Since different ways of relating clauses may result in different referent bindings for anaphor, a referent binding list must also be local for each context space.

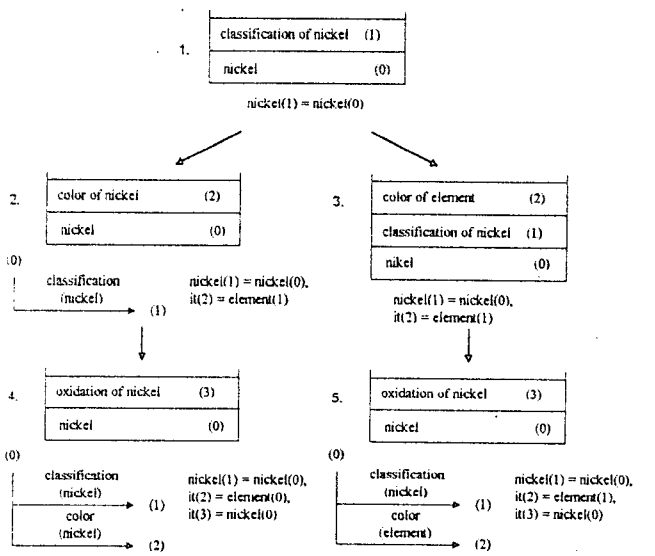

Fig. 2. Context spaces for text (1-1).

Given context space 1, DPAS relates clause (2) to the context space. Since the predicate of clause (2) is the *be-verb*, DPAS finds that clause 2 explains the **color* of the referent of it in clause (2). There are two possible referents of it—nickel and element—in clause (1) which is in the top of the FS in context space 1. Since nickel(1) in clause 1 refers to nickel(0) in clause (0), the referents of it in clause 2 are nickel(0) and element(1). Therefore, there are two relations: [**color*, nickel(0), 2] and [**color*, element(1), 2].

Since clause (2) can be related to nickel(0) with the first relation, DPAS creates an FS for context space 2 by popping the FS in the context space 1 and pushing the information of clause (2) with the relation. As a result of the pop operation, the information of the popped segment is recorded into the discourse structure which is in the left-bottom side of context space 2. In the course of finding discourse relations, DPAS determines the referent of it(2) in clause (2) as nickel(0) in clause (0) and adds the referent-anaphor pair to the referent binding list of context space 2. The referent binding list is shown at the right bottom side of context space 2.

With the second relation, clause (2) can be related to element(1) in clause (1). Therefore, DPAS pushes clause (2) onto the FS in context space 1 to create an FS for context space 3. Since there was no pop operation, no discourse structure is constructed. However, since the referent of it(2) is determined as element(1) in clause (1), DPAS adds the pair to the referent binding list and records the FS and the binding list as context space 3.

Now, DPAS has two context spaces for the first three clauses in (1-1). They have different focus history stacks, different discourse structures and different referent binding lists.

DPAS takes the next input, clause (3), and relates it to the two context spaces. First, DPAS relates the input clause to context space 2. Since the predicate of clause (3) is rust, DPAS attempts to relate the concept of rust and the concepts of the words in the top of the FS in the context space using knowledge base. As a result, DPAS finds the relation between rust(3) and it(2). Since it(2) refers to nickel(0), DPAS determines that the referent of it(3) is nickel(0), and clause (3) is related to nickel(0). Since clause (3) is related to nickel(0) in clause (0), DPAS pops the FS in the context space and pushes clause 3 to create a new FS of context space 4. As a result of the pop operation, the discourse relation for clause (2) is added to the discourse structure shown in the left-bottom side of context space 4. Now, a new context space 4 is made and recorded for the next input clause.

When DPAS relates clause 3 to context space (3), it fails to find any discourse relation because the referent of it(2) is element(1), not nickel(1), and there is no proper relation between the concept of rust and element. Therefore, DPAS pops the FS in the context space and tries to relate the clause to the new top, clause (1). The information of the popped segment is recorded into the discourse structure of the context space. DPAS determines that clause (3) can be related to nickel(1). Since, however, the referent of nickel(1) is nickel(0),[4] DPAS relates clause (3) to nickel(0) in clause (0). Therefore, DPAS pops the FS again and pushes clause (3) to create a new FS for a new context space. The pop operation adds the discourse relation of clause (1) into the discourse structure, and it(3) is added to the referent binding list with its referent nickel(0). In this way, context space 5 is constructed by relating clause (3) to context space 3.

DPAS uses three different kinds of knowledge in discourse analysis: (1) world knowledge in long-term memory, (2)

3) The number in parentheses attached to each word represents the number of the clause in which the word appears.

4) Since nickel is a generic term in metal domain, we simply treat it as an anaphor. How to handle generic terms is another difficult problem in its own right.

semantic structure of a sentence in the working memory, and (3) information in each context space. The first two types of knowledge are global, while the third one is local information.

When DPAS chooses one context space among competing context spaces, one of the most important heuristics used by DPAS is comparing the number of pop, push and mix operations performed to create each context space. Therefore, the list of pop, push and mix operations must also be kept in each context space as local information.

We expect there will be more local information required for each context space for DPAS to properly analyze texts as we expand the domain of the target texts.

2. Heuristics for selecting one context space

A certain text may have more than one generally acceptable discourse structure. Sometimes it is hard to say which discourse structure is better than another. We do not attempt to develop heuristics to resolve such ambiguous discourse interpretations which are controversial even for human readers. In many cases, however, for some discourse structures of a text constructed by DPAS, we can easily ascertain that one is better than others. Human readers share certain preferences when they make a discourse structure for a text. The heuristic rules in DPAS are based on the kind of preferences that most of human readers seem to be using when they analyze a text. This heuristic is can also be used in writing a descriptive text. Sometimes one text is easier to comprehend than other texts, even when they have the same contents and expressions.

One of the most basic preferences in grouping clauses into a discourse segment is making the discourse segment as big as possible. If we can relate several clauses with one discourse relation, it is better than using many different discourse relations to relate the same clauses. This preference is similar to the rule of Occam's razor. When we relate clauses in a text into a discourse structure we prefer to use a minimum number of discourse relations.

In DPAS, since clauses are mixed into one discourse segment if they explain the same property of the same topic in descriptive texts, one heuristic is to choose the context space which leads to more mix operations than the other competing context spaces. According to our experimentation with several different texts, the best discourse structure always has more mix operations.

Another heuristic is that if there are multiple topics in a context to which a clause can be related, we prefer the more recent topic to relate the clause. This preference causes DPAS to choose a context space which leads to fewer pop operations than the other competing context spaces.

After DPAS has applied the above preference rules, *i.e.*, more mix and fewer pop operations, if there is still more than one context space, DPAS tries to apply the following heuristics one by one until only one context space is left.

1. Choose a more specific property over an abstract one: When there are multiple relations for the same topic, if one relation is more specific than the others, DPAS chooses the most specific relation. For example, in text (1-1), clause 3 can be related to nickel as explaining **oxidation* of nickel. However, in the knowledge base, there are two possible discourse relations between them, **oxidation* and **chemical-activeness*. Since **oxidation* is a more specific concept, DPAS chooses **oxidation* over **chemical-activeness*.
2. When an input can be related to two topics, and the topics are in the same clause which was related to the context as a **classification* relation, DPAS chooses a main topic which is the subject of the clause. For example, in text (1-1), clause 2 can be related to both nickel and element in clause 1 since both of them can be the referent of it in clause 2. In this case, DPAS chooses nickel since clause 1 was related to the context as explaining **classification* of nickel and nickel is the subject of clause 1.
3. Since, in DPAS, finding the discourse relations for a clause with a self-relating predicate depends on finding the referent of an anaphor in the clause, if there are more than one possible referents for the anaphor, DPAS usually comes up with more than one discourse relation for the clause. This happens especially when the anaphor is a pronoun. In this case, DPAS must choose which referent is the most plausible one for the pronoun to assign one discourse relation.

Several researchers claim that the referent of a pronoun-anaphor can be determined by keeping track of the entities with which a discourse is most centrally concerned. The entities are called *topics* in [8], *focus*[5] in [19] and [14], and *backward-looking center* in [6]. They give a preference ordering among the entities and use the ordering to determine the referent of a pronoun-anaphor. However, as they have admitted, their orderings can be overridden by semantic feature constraints and discourse structural preferences. We found that it is desirable for DPAS to use the preference ordering as a heuristic to choose the most plausible referent of a pronoun-anaphor after DPAS checks semantic feature constraints and discourse structural preferences, *i.e.*, the heuristics described above. The ordering of potential topic entities in a clause is based on the syntactic structure of the clause and is provided by the syntactic process. This ordering is based on [4].

4. In most descriptive texts, once the texts start to explain a

5) This focus is the focus within a sentence. We can distinguish it by calling it a *local focus* and the focus of attention in a discourse as a *global focus*.

property of a topic, they are likely to finish explaining the property before starting to explain another property of the topic. Therefore, in DPAS, once a discourse segment which explains a property is popped out of an FS, the same property of the same topic is not likely to be explained in later clauses. In DPAS, when there is a discourse relation which has already been used for a discourse segment, and if the segment using the relation has been popped, DPAS removes the relation from the competing discourse relations.

5. Similarly, once a topic starts to be explained, it is likely to be the topic for the following clauses. When there are multiple discourse relations, DPAS chooses the relation with the continuing topic---a topic which the preceding discourse segment explains.

These heuristics are not complete. There may be more heuristics needed in some texts. We expect that different styles of texts need different heuristics.

DPAS applies the above heuristics one by one in order to prune less preferable discourse relations until one relation is left.

V. Concluding Remarks

DPAS has been developed on a SUN-SPARC system using Quintus-Prolog and tested with five sample texts in the disease and metal domains from [20].

DPAS uses domain specific relations as discourse relations in descriptive genre. There is, however, no formal categorization of the discourse relations used in DPAS. In this framework, choosing discourse relations is fairly subjective. Any domain specific relation which is useful to organize a text for summarizing and answering the question about the text can be included as a discourse relation.

The most important contribution of this work is providing an implementation theory of a multiple-path discourse parser. The context space representation makes it possible for DPAS to explore multiple interpretations of a context. Since there are multiple possible discourse interpretations for a text, constructing a discourse structure can be reasonably done only by comparing the competing interpretations, and not by making absolute criteria to decide whether an interpretation is the best interpretation for the text. In this sense, we believe, developing a multiple-path discourse parser is very important for developing a text understanding system.

Although DPAS has been developed for descriptive texts, we believe that the basic idea of DPAS can be applied to different types of text such as argumentative texts, narrative texts, and dialogues. To do this, we need to answer the following questions for each style of text.

- What kind of discourse relations will be used to relate clauses (or other discourse segments).
- How are those relations to be determined? What kind of knowledge representation and inferences can be used to determine such relations?
- When should pop, push, and mixing a clause (or other discourse segment unit) be performed to maintain a focus history stack?
- What kind of discourse heuristics can be used to prune out less plausible context spaces?

One of the interesting future works is using statistical methods for disambiguation [2, 3]. It requires, however, lots of pre-tagged texts to get reliable conditional probabilities.

References

[1] R. Alterman, "A Dictionary Based on Concept Coherence," *Artificial Intelligence*, Vol. 25, No. 2, pp. 153-186, 1985.

[2] E. Charniak "*Statistical Language Learning,*" MIT Press, 1993.

[3] J. Cho, J. Seo, and G. Kim, "*Syntactic Analysis in English using Automatically Acquired Context-Sensitive Grammar Rules from Corpus,*" Korea Information Science Society Journal, Vol.21 No.9, Sep, 1994. (In Korean)

[4] D. Dahl, "*Focusing and Reference Resolution in PUNDIT,*" *Proceedings of AAAI-86*, pp. 1083-1088, 1986.

[5] K. Dahlgren, "*Naive Semantics for Natural Language Understanding,*" Kluwer Academic Publishers, 1988.

[6] B. Grosz, A. K. Joshi, and S. Weinstein, "*Providing a Unified account of Definite Noun Phrases in Discourse,*" *Proceedings of 21st Annual Meeting of the Association for Computational Linguistics*, pp. 44-50, 1983.

[7] B. Grosz, and C. Sidner, "*Attention, Intention, and the Structure of Discourse,*" *Computational Linguistics*, Vol.12, No. 3, pp. 175-204, 1986.

[8] J. K. Gundel, "*Role of Topic and Comment in Linguistic Theory,*" Ph.D. thesis, University of Texas, Austin, 1974.

[9] M. Halliday, and R. Hasan., "*Cohesion in English,*" Longman, New York, 1976.

[10] J. R. Hobbs, "*On the Coherence and Structure of Discourse,*" Tech. Report No. CSLI-85-37, Center for the Study of Language and Information, Stanford University, October, 1985.

[11] D. LitMan, and J. Allen, "*A Plan Recognition Model for Subdialogues in Conversations,*" in *Cognitive Science*, Vol.11 pp. 163-200, 1987.

[12] W. C. Mann, and S. Thompson, "*Relational*

Propositions in discourse," *Discourse Processes*, Vol.9 pp. 57-90, 1986.

[13] L. Polanyi, *A formal model of the structure of discourse. Journal of Pragmatics*, Vol. 12, pp. 601-638, 1988.

[14] H. C. Rim, J. Seo, and R. Simmons, "*Transforming Syntactic Graphs into Semantic Graphs,*" In *Proceedings of 28th Annual Meeting of the Association for Computational Linguistics*, 1990.

[15] J. Seo, and R. F Simmons, "*Syntactic Graphs: A Representation for the Union of All Ambiguous Parse Trees,*" *Computational Linguistics*, Vol. 15, No. 1, pp. 19-32, 1989.

[16] J. Seo, "*Text Driven Construction of Discourse Structures for Understanding Descriptive Texts,*" Ph.D Dissertation, Department of Computer Sciences, The University of Texas, Austin, TX, 1990.

[17] J. Seo, "*Knowledge Representation and Inferencing Discourse Relations in Descriptive Texts,*" Korea Information Science Society SIGAI Letters, Vol.6 No.1, March, 1991.

[18] J. Seo, "*DPAS: A Multiple Path Discourse Parser,*" In *Proceedings of the 3rd Pacific Rim conference on Artificial Intelligence*, pp. 679-685, Aug. 1994.

[19] C. L. Sidner, "*Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse,*", MIT-AI TR-537, MIT, 1979.

[20] Staff of National College of Education (editor), "*Young People's Science Encyclopedia,*" Children's Press, Chicago, 1970.

Jungyun Seo is an associate professor in the Department of Computer Science, Sogang University in Seoul, Korea. Previously, he was an assistant professor in the Computer Science Department at the Korea Advanced Institute of Science and Technology in Taejon, Korea. He had worked at the UniSQL, Inc., Austin, Texas as a member of technical staff. His research interests include Natural Language Processing, Computer-Human Interface, especially, Multi-Modal Dialogue interface. Seo obtained a BA in mathematics from Sogang University, Seoul, Korea, and an MS and PhD in computer science from the University of Texas, Austin.