TM-1196

Planning Strategy of Text Structures in Argument Text Generation System Dulcinea

by
Y. Kubo, K. Hagiwara, T. Ikeda
& A. Kotani (Mitsubishi)

July, 1992

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

Planning Strategy of Text Structures in Argument Text Generation System *Dulcinea*

KUBO Yukihiro, HAGIWARA Kaoru, IKEDA Teruo and KOTANI Akira[†]

Institute for New Generation Computer Technology

Mitsubishi Electric Corporation[†]

Abstract

In order to generate a text that asserts some opinion, we need a clear expression of the standpoint and text coherency as well as some proper argument strategy. We call these kinds of text *Argument Texts*. However, there is a big gap to be filled between the semantic contents and realized text or natural language expressions.

In this paper, we discuss some strategies for generating coherent argument text with a clear expression of standpoint from semantic contents so as to fill this gap. We provide an abstract text structure named FTS (Functional Text Structure) to represent the intermediate state in the gap that stands for not only the semantic contents but also the system's standpoint and the judgments as well as the linguistic constraints.

Using this strategy, we developed an experimental text generation system called *Dulcinea* [16] that generates Japanese text that justifies a given argument goal. The strategy is implemented as a set of parameters so that we can investigate it to find the most plausible parameters for the persuasive and coherent text. The realizing process from semantic contents is also explained.

1 Introduction

Natural language generation systems vary from the generation of a single sentence to that of coherent text. Recently, the volume of text generated has tended to increase and much research is being done on coherent text generation. In our laboratory, we developed a natural language generation tool which will realize a Japanese sentence from a syntactic structure. Then, we shifted the center of our research to text structure planning for natural text generation.

According to Hovy[8], one of the unsolved problems in the field of text generation by computer is what relations, plans and schemas are necessary to support the planning of coherent multi-paragraph texts. To investigate this problem, we choose a narrow domain of the argument as an application of multi-paragraph text generation.

To solve our problems, we investigated the following: what semantic content affects the reader, what text structure should we organize and how should we represent the text structure efficiently. As a result, we arranged various argument strategies on three levels.

1. the plans for generating the semantic content of each constituent of the argument texts,
2. the prescriptive knowledge for organizing the linguistic text structure by combining the constituents, and
3. the representation form of the local relations between adjacent sentences that hold within the argument texts.

The first level is detailed in [10]. The last two levels will be discussed in this paper. In many related works, text generation systems employ a model for the discourse structure. RST[11] and Schema[13] are typical examples of a model for generating a coherent discourse structure. Mann and Thompson formalized a set of about 25 relations sufficient to represent the relations between adjacent blocks of text by RST. McKeown's schema represents the structure of stereotypical paragraphs for describing objects, and selecting the proper schema from her four schemas enforces this coherence.

The schema that describes the typical format of argument text is suitable for our system's generation process, because it is driven by one global intention (i.e. insisting the system's standpoint effectively), and it completes multi-paragraph text without any interaction from the user. In fact, our *Dulcinea* system uses schema-like knowledge to generate the semantic contents of the text and to organize the text structure.

With schemas, however, it is difficult to represent the local relations between adjacent sentences within the blocks. So, we represent the relations between adjacent sentences with a RST-like representation form, called FTS (Functional Text Structure). In other words, the semantic contents of argument text are represented as the abstract text structure FTS according to schema-like knowledge of argument strategies and linguistic text structures. This knowledge is implemented by a set of parameters so that we can make investigations to generate coherent text.

Section 2 introduces the argument graph as semantic contents. In this paper, the argument graph along with the argument goal are regarded as a given input to realize an argument text. Section 3 shows the description of the abstract text structure that we introduce to this paper. In Section 4, we explain in detail the two kinds of argument strategies mentioned above. An example of some realized texts are shown in Section 5.

2 Argument Graph as Semantic Contents

In this section, the argument graph as semantic contents is explained briefly. The semantic contents that consist of four parts: main ground, refutations for the opposing arguments, examples and conclusion, are represented by the argument graph. Figure 1 is an example of the argument graphs, which insists the argument goal *"The two-way lane must be enforced"*. A two-way lane is a lane that allows buses to drive the wrong way up a one-way street.

Each node in the graph represents a state of affairs. `Ng` in the nodes indicates that the state of affairs is regarded as *no_good*, and `af` indicates that the system assumes that this is true. The system regards `f` as the fact.

Nodes (1)~(5), including assumed node (2), represent the grounds for justifying the argument goal. The thick arrows in the graph indicate general causation, and the p_cond link is used to represent the assumed node. The term in node (1), `enforce(two-way-lane;0)` is the negative state of affairs of `enforce(two-way-lane)`. The system regards node (5) as *no_good* and so does node (1), the negative state of affairs of the argument goal.

Nodes (6),(7) and (8) are examples corresponding to ground nodes (1), (3) and (4) respectively.

The `anti` link means the linked graph has contents opposite to the ground, and the `deny` link shows that the node seems to be caused, but is denied by the linked graph.

3 Abstract Text Structure FTS

We introduced the FTS as the abstract representation form of text. The FTS is able to represent information such as the writer's judgments, necessary to generate coherent text, and to reflect the writer's standpoint besides the propositional contents. In the FTS, both the local relations between the state of affairs and the global construction of the text are described together.

FTS is a text structure representation form which represents the functional relations that hold within a piece of text. FTS consists of the FTS-term, order constraints and gravitational constraints. The order constraints and the gravitational constraints are optional.

FTS-term: The data structure that represents functional dependencies that hold within a piece of text. FTS does not fix the order of the sentences.

Order constraint: The constraint of the order between two sentences. The order constraint S1<S2 means that the text in which sentence S2 comes after sentence S1 is preferable.

Gravitational constraint: The constraint of the distance between two sentences. The gravitational constraint S1-S2 means that the text in which sentence S1 is located near sentence S2 is preferable.

Table 1 is a list of attributes to describe the FTS-term. These attributes take the FTS-term recursively as its value except for `thesis` that takes a belief content as its value.

Table 1: Attribute Labels in the FTS-term

Labels	Description of attribute
thesis	A conclusion of the FTS-term
reason	A reason of the thesis
anti_t	An opposing content of the thesis
crecog	A cause of the thesis
exampl	An example of the thesis

The following is a typical FTS expression that would be realized something like *"Though he is excellent in his work, he is useless because of his poor health. For example he was absent yesterday."*.

[thesis= *He is useless.*
reason= *He has poor health,*
anti_t= *He is excellent in his work,*
exampl= *He was absent yesterday*]

FTS produces various surface text structures by deciding the order of the sentences, and whether to connect two adjacent sentences or not, as well as the type of connective. The order of the sentences and the connection of the adjacent sentences are important for making the text comprehensible. Our system is able to generate coherent text in regards to sentence order and connection by selecting the best surface text structure from the structures that FTS can generate. The selection is based on criteria we will describe later.

4 Text Generation from Semantic Contents

In order to generate an argument text that corresponds to a given argument graph, three kinds of modules are provided in this system such as FTS Generator, Text Structure Generator and Realizer.

An argument graph only stands for semantic contents which are not conscious of natural language expression at all. Therefore, in order to generate an argument text, we must get an abstract text structure firstly that is to be a natural language expression. FTS Generator transforms a given argument graph into an abstract text structure FTS.

After generating FTS, Text Structure Generator decides the order of each sentence that occurs in FTS as well as connective expressions, according to some criteria in terms of linguistic knowledge. Consequently, FTS with order and connective information is sent to Realizer.

Realizer generates sentences according to the order and connective information specified by the previous module. In each sentence, Realizer selects appropriate words that correspond to each object and fixes tense, aspect and mood. It also omits redundant words.

In the following, we explain the features of each module in more detail.

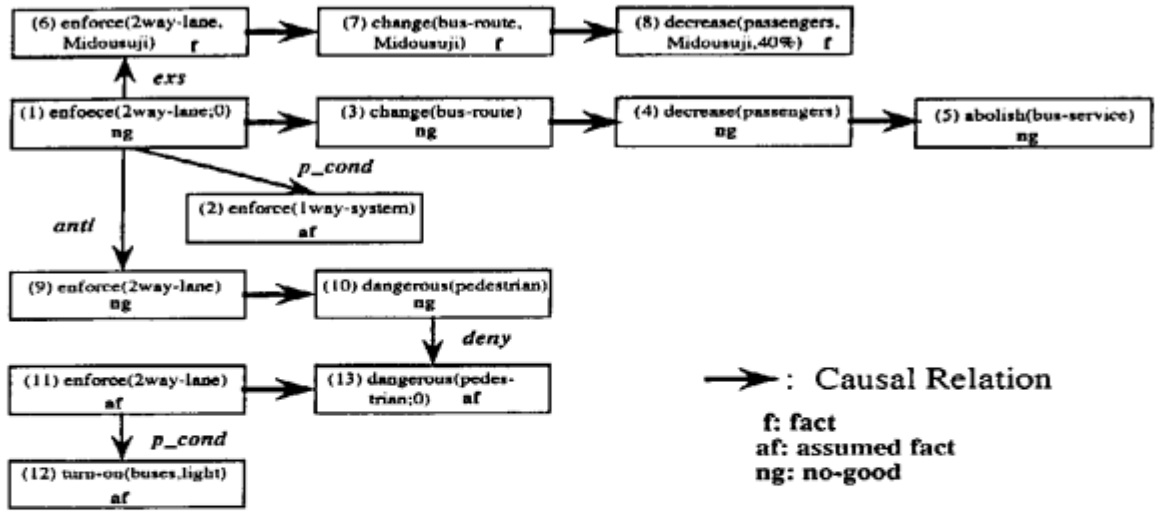


Figure 1: Argument Graph

4.1 FTS Generation

This process applies some linguistic argument strategies to an argument graph and constructs an FTS of an argument text. Since the argument graph expresses only the semantic content of the argument which is independent of the natural language expressions to be realized, there is a big gap between the semantic content and abstract text structure FTS.

In order to fill the gap, FTS Generator firstly analyzes an argument graph and recognizes the ground part and refutations for the opposing argument parts, and then decides the order in which to generate these blocks according to the strategic knowledge for the argument. This order of blocks reflects that of paragraphs in the argument text and is implemented as an order constraint of FTS. As for each block, the subgraph corresponding to the ground or refutations for the opposing argument is transformed into a suitable FTS-term expression.

This module prevents the generation of redundant expressions or unnatural arguments by ignoring some constituents of an argument graph that cannot be used for the argument or would make the text redundant or unnatural.

4.1.1 Generation of FTS-term

Here we show the transformation rules defined for each constituent of the argument graph. Note that a FTS term like [thesis= A → B] is an abbreviation for the following.

[thesis= [cause= A,
result= B]]

1. **Generation of the ground with example**

The main ground of the argument graph with no `anti_t` and `exampl` part is represented as a FTS term that has a set of thesis consists of a causal chain in an argument graph. An argument graph with length N, or which has N causal chains, is transformed into the following N theses.

[thesis= rn(n) → rn($n+1$)] $(0 \leq n \leq N-1)$

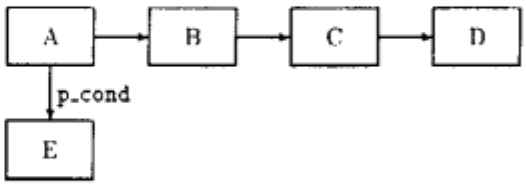
Where rn(n) stands for the content of the root node in the n-th subgraph. Therefore rn(0) means the root node of the top level graph. In the case of $N \geq 2$, all the theses are bundled into a set and placed in one thesis.

[thesis= set={ [thesis=...],
[thesis=...],
⋮
[thesis=...]}]

If rn(n) has a presupposition or an assumption, corresponding thesis will be

[thesis= [assume= as(n),
p_cond= pc(n),
cause= rn(n),
result= rn($n+1$)]],

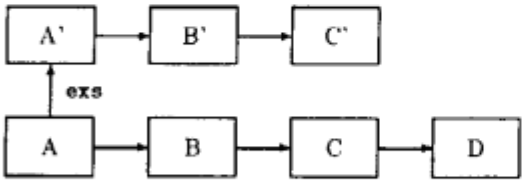
where as(n) and pc(n) represent the contents of assume and p_cond in the n-th subgraph respectively. For instance, a ground with length 3 is transformed into the following FTS-term.



[thesis= [set={ [thesis= [p_cond= E,
cause= A,
result= B]],
[thesis= B → C],
[thesis= C → D] }],
ftst_type= main]

As this is a main ground of the argument, the value of the attribute `ftst_type` of this FTS-term is *main*. This attribute is specified only in the top level FTS-terms. At this time, realizing the order of each thesis and omissions of redundant words are not specified in the FTS-term. They will be decided in the following modules.

Next, we show the FTS-term for a ground with examples. In this case, the ground has three causal chains two of which have corresponding examples. The content of the example node has more specific information than that of the corresponding node of the ground.




```
[thesis= [set={
          [thesis= [set={ [thesis= A → B],
                          [thesis= B → C] }],
           exampl= [thesis= C',
                    crecog= [thesis= B',
                             crecog= [thesis= A'] ]
                    attent= [loc,somewhere] ]],
          [thesis= C → D] }],
fts_type= main]
```

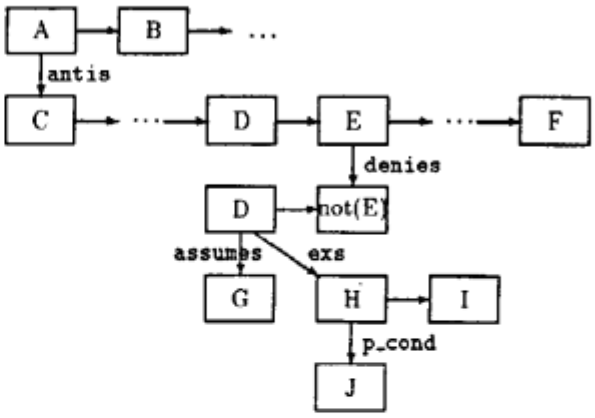
The label `exampl` is added to the thesis that corresponds to the example. FTS-term generated at the `exampl` label is defined recursively. Note that the minimum length of the example is two because these are the example of causal relations.

$$\text{exampl}(n)\text{: [thesis= ex\_rn}(n),\ \text{crecog= exampl}(n-1)]\quad (1 < n \leq N)$$

Where ex_rn(n) stands for the contents of the root node in the n-th `exs` graph. In addition, if the argument graph has attention information that are common instantiations of roles through examples such as location, then attention information is added to the FTS-term. This is used to realize the topic marker "ha" in the first sentence at the beginning of the example paragraph.

2. Generation of refutation

Refutation of the opposing argument is regarded as an independent block and may be generated as a paragraph as well as a ground. Whereas the `ftst_type` of the main ground was *main*, the `ftst_type` of the refutation of the opposing argument is *anti_deny*.



When the FTS Generator finds the above structure in a graph, it makes the following FTS term.

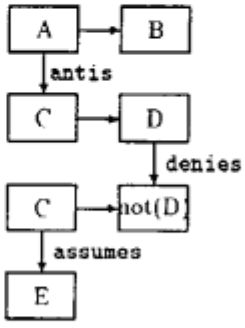
```
[thesis= [denied= t1],
 reason= [thesis= t2,
          exampl= t4],
 anti_t= [thesis= [seem= t3]],
 ftst_type= anti_deny ]
```

Where, t1,...,t4 are

```
t1: C → F
t2: [assume= G,
     though= D,
     result= not(E)]
t3: C → ... → D → E → ... → F
t4: [thesis= I,
     crecog=[set={ [thesis= J],
                   [thesis= H] }] ].
```

"not(E)" means that it has the opposite polarity to E.

However, when C=D and E=F, the content of t2 subsumes that of t1, and so generated text may be redundant. Therefore, in such a case, the content of thesis is replaced with that of `reason`, which is rather essential for the argument. Consequently, there is no `reason` part in the FTS-term. The following is an example of the case in which a main ground is also transformed into the FTS-term.




```
[set={
    [thesis= t1:A → B,
```

ftst_type= main],
[thesis= t2:[though= C,
assume= E,
result= not(D)],
anti_t= t3:[thesis= [seem= [thesis= C → D]]],
ftst_type= anti_deny] }]

All the FTS terms generated in this way can be regarded as a reason for the argument goal. Then, the top level FTS term consists of a pair of the argument goal and its reason.

[thesis= *Argument Goal*,
reason=*Reason for the Argument Goal*]

4.1.2 Constraints for sentence order

As we mentioned in Section 3, we can optionally specify some constraints on order with regard to a realized sentence. Currently, the following three parameters can be specified for each FTS.

Top level order: specifies the location of conclusion and its ground

Order in ground: specifies the order between main ground and anti_deny blocks

Order in anti_deny block: specifies the order of `thesis`, `reason`, `anti_t` and `exampl` in the anti_deny block

The parameters described above should be decided dynamically according to the argument goal and semantic contents. Although we have a static value set to them currently, we have been investigating this sort of knowledge with regard to an argument strategy. Some of the strategies may include the following.

- When the argument is rather big, the conclusion should be placed at the end of the text
- If the argument goal is type "hb", then we should put anti_deny blocks before a main ground
- In the anti_deny block, `anti_t` part should be realized first

These kinds of knowledge will depend on the applications that the system handles. Rather independent knowledge of the linguistic text structure is discussed below.

4.2 Adding Sentence Order and Connective Information

Text Structure Generator defines the connection relation of each sentence in a given FTS, and generates the surface structure of a whole text. In general, a number of surface structures can be generated from one FTS. In order to generate one plausible surface structure, the module processes the FTS in two steps.

1. Generates every possible connection relation from the given FTS.

Table 2: Criteria for Connection Relation

Depth of a memory stack	The depth of a memory stack should be shorter.
Number of bad dependency structures	The number of bad dependency structures should be smaller in a text.
Structural similarity	The structure of the surface text should be similar to the FTS.
Number of connectives with negative statements	Not more than two connectives to introduce negative statement should appear in a sentence.
Number of connecting two clauses	Two clauses should not be connected more than a certain number of times.
Gravitational constraint	Two sentences under the gravitational constraint should be placed close.
Stability of topics	Sentences should be ordered so as not to change the topic frequently.
Connecting two implications	Two implications A→B and B→C should be realized in this order
Sentence order similarity between the ground and the example	The sentence order of the example should be similar to the ground.

2. Evaluates those connection relations based on the criteria in Table 2, and chooses the best connection relation.

Using the criteria for connection relation in Table 2, the module adds an `order` attribute which represents sentence order and a `conn` attribute which represents the connection of two sentences to the FTS. The surface expression for connectives are specified by the type of the connections (see Table 3).

For example, the depth of a memory stack needed to read a text should be shorter. Because the shorter the memory stack needed the easier to understand the text. Next, we illustrate the criterion for the bad dependency structure in detail using the FTS below.

[thesis= s1,
anti_t= s2,
exampl= s3]
order constraint: s2 < s1, s2 < s3

In this case, there is a relation *negation* between s1 and s2 and a relation *example* between s1 and s3. We also suppose two order contraints s2 < s1 and s2 < s3 hold in the FTS. Then, we can generate sentences in two different orders.

1. `s2<s1<s3`
2. `s2<s3<s1`

Figure 2 represents the dependency structure of each text. Since dependency structure 2 does not have direct dependency between `s2` and `s3`, the reader cannot find the semantic dependency of `s3` when `s3` is reached while reading this text. They can find the semantic dependency only after reading through `s1`. This means that the text in order 2 is much more difficult to understand than that in 1.

Table 3: Type of Connections and Their Expressions

deduction	s	shitagatte,dakara,yotte,yueni,...
	c	~kara,~node,...
causation	s	sonokekka,sonotame,...
	c	~tame,~(ren'youkei),...
reason	s	nazenara~karadearu,toiunoha~karadearu,...
	c	×
development	s	suruto, ...
	c	~to,~(ren'youkei),...
negation1	s	shikashi,daga,...
	c	~ga,...
negation2	s	~ga,...
	c	×
juxtaposition	s	mata,...
	c	~shi,...
example	s	tatoeba,jissai,...
	c	×
generalization	s	konoyouni,...
	c	×
presentation	s	(just placed continuously)
	c	~ga,
implication	s	×
	c	naraba,reba,to,...
addition	s	×
	c	te,(ren'youkei),...
concession	s	×
	c	temo,tatoe~temo,...

s: separate
c: connect

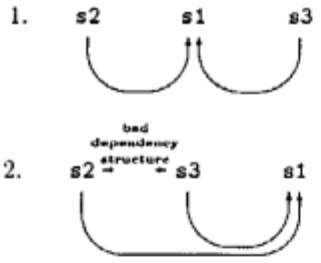


Figure 2: Dependency Structure

We call the dependency structure in 2 the bad dependency structure.

In evaluation of the bad dependency structure, order 1 is preferred to 2, and the attributes' values become the following.

[thesis= s1,
anti_t= s2,
exampl= s3,
order= [s2,s1,s3],
conn= [(negation1,s),(example,s)]]

Figure 3 shows the FTS with order and conn information generated from the argument graph shown in Figure 1.

4.3 Realizing Argument Text

This process realizes the content of the FTS added order and conn attributes in terms of natural language expressions. An FTS with order and conn is a tree structure which represents the syntactic structure of the whole text. The leaves of the tree structure are realized as clauses. Syntactic structural relations which hold at a higher level than clauses, such as the relation between clauses and the relation between sentences, have already been generated by adding order and conn to the FTS.

For each term in the tree structure, lexicons correspond to each object in the term. Here, suffixes expressing the functions for each object, tense and aspect expressions of predicates and the system's judgment expressions are all decided. Then, connectives which represent the relations between terms are determined by Table 3.

Among the causal relations between terms, those that have been described as a rule in the beliefs are represented as an "implication", which is a strongly dependent connective relation. The following rule in the belief

change(bus-route)⇒decrease(passenger)

will be realized *"If the bus route is changed, the passengers decrease."* The relation between **thesis** and **reason** described in the FTS-term is represented as a "deduction", which is a weakly dependent connective relation. For instance, the FTS-term:

[thesis= must(enforce(two-way-lane)),
reason= abolish(bus-service)]

will be expressed *"The bus service will be abolished. Therefore, a two way lane should be enforced."*

Redundant words are also omitted by this module.

5 Example of Generated Argument Text

Dulcinea can generate various texts relative to its parameters. In this section, we show two kinds of argument texts.

The following is the argument text which justifies the argument goal *"A two-way lane must be introduced"* using plausible parameters.

> In *Midousuji* street, a one-way system was introduced while a two-way lane for buses was not introduced. This forced a change in the routes of bus services resulting in a decrease in passengers of 40%. In this way, if a two-way lane is not introduced into a one-way system, the routes of bus services changes and the number of passengers will decrease. Moreover, if the number of passengers decreases, the bus service may be forced to stop.
>
> On the other hand, introducing a two-way lane seems to put pedestrians in danger, however introduction of a two-way lane does not put pedestrians in danger if the buses turn their headlights on. For example, in London, a two-way lane is introduced with the headlights of the buses turned on. So, the pedestrians were not in danger.
>
> Therefore, a two-way lane must be introduced.

```
[thesis= 1:must[cont= enforce[obj=two-way-lane]],
 reason= 2:[set={
    1:[thesis= [set={
          1:[thesis= 1:[set={
                1:[thesis= [p_cond= 1:(enforce[obj=one-way-system],af),
                           cause= 2:(enforce[obj=two-way-lane,pol=0],af),
                           result= 3:(change[obj2=bus-route],af),
                           order= [1,2,3], conn= [(condition,c),(implication,c)]]],
                2:[thesis= [cause= 1:(change[obj2=bus-route],af),
                           result= 2:(decrease[obj2=passenger[mod=bus]],af),
                           order= [1,2], conn= [(implication,c)]]]},
             order= [1,2], conn= [(development,c)]],
          exampl= 2:[thesis= 1:(decrease[obj2=passenger[mod=bus],amo=40%,tem=prec,loc=Midosuji],f,ng),
                     crecog= 2:[thesis= 1:(change[obj2=bus-route,loc=Midosuji],f),
                                crecog= 2:[set= {
                                   1:[thesis= (enforce[obj=one-way-system,loc=Midosuji],f) ],
                                   2:[thesis= (enforce[obj=two-way-lane,loc=Midosuji,pol=0],f)]},
                                   order= [1,2], conn= [(juxtaposition,c)]],
                     order= [2,1], conn= [(causation,s)]],
          attent= [{loc,Midosuji}],
          order= [2,1], conn= [(causation,s)]]]},
    order=[2,1], conn=[(generalization,s)]],
       2:[thesis= [cause= 1:(decrease[obj2=passenger[mod=bus]],af),
                   result= 2:(abolish[obj=bus],af,ng),
                   order= [1,2], conn=[(implication,c)]],
          order= [1,2], conn= [(juxtaposition,s)]],
    ftst_type= main],
  2:[thesis= 1:[though= 1:(enforced[obj=two-way-lane],af),
                assume= 2:(turn-on[obj=lights[mod=bus]],af),
                result= 3:(dangerous[obj2=pedestrian,pol=0],af),
                order= [2,1,3], conn= [(implication,c),(concession,c)]],
     ftst_type= anti_deny,
     attent= [{obj2,pedestrian}],
     anti_t= 2:[thesis= [seem= [cause= (enforce[obj=two-way-lane],af),
                                result= (dangerous[obj2=pedestrian],af,ng),
                                order= [1,2], conn= [(implication,c)]]
                                order= [1], conn= []]],
     order= [2,1], conn= [(negation1,s)]] },
  order= [1,2], conn= [(change,s)]],
order= [2,1], conn= [(deduction,s)]]
```

Figure 3: FTS with order and conn attributes

On the other hand, rather implausible parameters like the following can make the realized text poor. In this case, three kinds of criteria: bad dependency structure, structural similarity and connecting two implications are changed for the worse.

> In *Midousuji* street, a two-way lane for buses was not introduced. Introduction of the one-way system forced a change in the routes of bus services resulting in a decrease in passengers of 40%. In this way, if the routes of bus services change then the number of passengers decrease. If a two-way lane is not introduced into a one-way system, the routes of bus services change. Moreover, if the number of passengers decrease, the bus service may be forced to stop.
>
> On the other hand, introducing a two-way lane seems to put pedestrians in danger. However the two-way lane was introduced in London. There the buses turned their headlights on, and the pedestrians were not put in danger. In this way, if the buses turn their headlights on, the pedestrians will not be in danger even though a two-way lane is introduced.
>
> Therefore, a two-way lane must be introduced.

In the first part of text, where the example of *Midousuji* street was given, the structural similarity is broken. In the next part, two causal relations are placed in reverse order. A bad dependency structure is generated according to the bad criteria in the second paragraph. In this way, bad criteria make the realized text very hard to understand. In other words, it is very important for us to investigate the proper criteria so as to realize a coherent text.

6 Conclusion

We have described the planning strategy of text structures from semantic contents in argument text generation System *Dulcinea* that produces a set of sentences in support of some opinion in various domains. The generation process is also illustrated along with examples.

In order to fill the gap between semantic contents and natural language expressions, we designed the abstract text structure FTS which represents not only the semantic contents but also the system's standpoint, the judgments and the linguistic constraints.

Furthermore, so as to generate coherent and persuasive argument text, we have investigated some strategies on linguistic text structure that was implemented as a set of parameters in the text structure generator. By changing these parameters, we can generate various kinds of text the quality of which we can evaluate. This strategy is plausible for multi-paragraph text generation in a very narrow domain. However, we believe we could offer one solution to

the question of realizing coherent multi-paragraph texts. For future, we are considering the following.

1. Evaluation of parameters used in the Text Structure Generator
2. Analysis and formalization of criteria used in the FTS Generator and their implementation with evaluation
3. Investigation of criteria for selecting the plausible argument graph

In order to implement the last two, some user model or context sensitive model would be indispensable and would make the realized text more coherent and natural.

7 Acknowledgments

We would like to thank Yuichi Tanaka for helpful advice. We would also like to thank all the members of the Sixth Research Laboratory in ICOT and the members of the NLU working group for their intensive discussions and suggestions.

References

[1] *the Proceedings of the International Joint Conference on Artificial Intelligence*. 1985.

[2] *the Proceedings of 6th AAAI Conference*, 1987.

[3] Douglas E. Appelt. Planning natural-language referring expressions. In David D. McDonald and Leonard Bolc, editors, *Natural Language Generation Systems*. Springer-Verlag, 1988.

[4] Laurence Danlos. Conceptual and linguistic decisions in generation. In *the Proceedings of the International Conference on Computational Linguistics*, 1984.

[5] E. H. Hovy. Pragmatics and natural language generation. *Artificial Intelligence*, 43:153–197, 1990.

[6] Eduard H. Hovy. Integrating text planning and production in generation. In *the Proceedings of the International Joint Conference on Artificial Intelligence* [1].

[7] Eduard H. Hovy. Interpretation in generation. In *the Proceedings of 6th AAAI Conference* [2].

[8] Eduard H. Hovy. Unresolved issues in paragraph planning. In *Current Research in Natural Language Generation*. Academic Press, 1990.

[9] Aravind K. Joshi. Word-order variation in natural language generation. In *the Proceedings of 6th AAAI Conference* [2].

[10] Hagiwara Kaoru, Kubo Yukihiro, Ikeda Teruo, and Kotani Akira. Generation mechanism of planning strategy of text structures in argument text generation system Dulcinea. In *IEICE. NLC*, 1992.

[11] W. C. Mann and S. A. Thompson. Rhetorical structure theory: Description and construction of text structures. In *Natural Language Generation: New Results in Artificial Intelligence, Psychology, and Linguistics*. Dordrecht: Martinus Nijhoff Publishers, 1987.

[12] David D. McDonald and James D. Pustejovsky. Description-directed natural language generation. In *the Proceedings of the International Joint Conference on Artificial Intelligence* [1].

[13] K. R. McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, 1985.

[14] K. R. McKeown and W. R. Swartout. Language generation and explanation. In Michael Zock and Gérard Sabah, editors, *Advances in Natural Language Generation*, volume 1. Ablex Publishing Corporation, 1988.

[15] Marie W. Meteer. The 'generation gap' the problem of expressibility in text planning. Technical report, BBN Systems and Technologies Corporation, 1990.

[16] Ikeda Teruo, Kotani Akira, Hagiwara Kaoru, and Kubo Yukihiro. Argument text generation system (Dulcinea). In *International Conference on Fifth Generation Computer Systems 1992*, volume 1, pages 385–394. ICOT, 1992.