PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT. © ICOT, 1992

Argument Text Generation System (Dulcinea)

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

Institute for New Generation Computer Technology
1-4-28, Mita, Minato-ku, Tokyo 108, Japan

Mitsubishi Electric Corporation[†]
5-1-1, Oofuna, Kamakura, Kanagawa 247, Japan[†]

Abstract

A generation of texts to justify some opinion requires clear expression of the system's standpoint and beliefs along with the proper strategy for structuring text. We call these kinds of texts *Argument Texts*. This paper is intended to investigate the argument strategy for the coherence of text and definite expression of standpoints. Moreover, we developed the argument text generation system, Dulcinea, using this argument strategy.

This strategy is plausible for the multi-paragraph text generation in a very narrow domain, but we believe that it is one of the answers to the question: "What relations, plans and schemas are necessary to support the planning of coherent multi-paragraph texts?".

The system generates text to justify an argument goal. The text, which reflects the standpoint and the judgment of the system, is represented by an FTS (Functional Text Structure). The FTS represents not only the semantic contents of the argument but the system's standpoint, the judgments and the linguistic constraints.

1 Introduction

Natural language generation systems produce various utterances: from single sentences in a dialog to coherent paragraphs. In recent years, the volume of text generated in text generation research has increased. Many natural language generation systems are able to generate multi-paragraph texts. The quality of these texts is also improving. The center of research is shifting from linguistic realization, which deals with linguistic forms, to structure planning, which produces semantic structures to attain the system's communicative intention. Not only the propositional content, but also the writer's intention and viewpoint are being focused on.

The multi-paragraph texts written by humans are appropriately structured to present their intentions efficiently, and to develop the topics according to their beliefs, interests and viewpoints. These properly structured coherent texts are able to express judgments and attitudes on topics based on the standpoint of the author. By computer, however, it is difficult to produce coherent texts. Therefore, it is necessary to consider coherence and appropriate structure planning for generating high quality multi-paragraph texts by computer.

Especially, the generation of texts to justify some opinion requires clear expression of the system's standpoint and beliefs along with the proper strategy for structuring text. We call these kinds of texts *Argument Texts*. This paper is intended to investigate the argument strategy for the coherence of text and definite expression of standpoints. Moreover, we developed the argument text generation system, Dulcinea, using this argument strategy.

Section 2 illustrates the features of the argument texts and gives a brief description of the belief contents, the plans to generate semantic contents for each constituent of the argument texts, and an abstract text structure form called FTS (Functional Text Structure). Section 3 describes an overview of the Dulcinea argument texts generation system. The system has four processes: generating the semantic contents of arguments, organizing linguistic text structure with argument strategy, sentence level organization for orders and connections, and realizing texts. Section 4 gives an example of argument text generation.

2 What are Argument Texts?

2.1 Features of Argument Texts Written by Humans

An argument text is a set of sentences in support of some opinion. They are generated according to some argument strategies, in order to persuade the reader to agree with the opinion. An argument strategy based on linguistic knowledge is useful to generate effective text to persuade the reader. As a manner of showing the justification of an opinion, for example, some texts give a simple but forceful sentence while others spend paragraphs in explaining the detailed grounds step by step. In addition to how justification is given, it is important to reinforce the argument with related topics. Adding examples to the grounds increases the persuasiveness of a text. Moreover, a technique in which a text mentions an expected opposing argument and refutes this argument is an effective way of persuasion.

As mentioned above, giving not only the grounds for the conclusion, but also developing topics with examples and opposing arguments persuades the reader more effectively. Such a text must reflect that the writer holds a consistent attitude from a specific standpoint to the topic. An argument text may become vague and not clearly state a view if it does not show a consistent attitude by the writer. Thus it is important to reflect a writer's consistent attitude in natural language expressions by considering the coherence of the text.

2.2 Related Work in Text Generation

In his research, Hovy developed a system[Hovy 1988] that achieves various pragmatic goals to convey more information than that contained in the literal meanings of words. This he did by setting rhetorical goals as intermediate goals between the pragmatic aspects of communication and the syntactic decision of the text generator. As a result, a single semantic content can produce a variety of texts which reflect various conversational settings in various ways.

Hovy's purpose was to connect wide-range pragmatic aspects to natural language expression by various conversational settings and rhetorical goals. His work was successful in this point, but did not give a concrete structuring method to make texts coherent. According to Hovy[Hovy 1990b], one of the unsolved problems in the field of generation by computer is what relations, plans and schemas are necessary to support the planning of coherent multi-paragraph texts. We investigate this problem in the narrow domain of the argument, which we choose as one of the applications 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 described various argument strategies on three levels. First, the plans for generating the semantic content of each constituent of the argument texts. Second, the prescriptive knowledge for organizing the linguistic text structure by combining the constituents. Third, the representation form of the local relations between adjacent sentences that holds within the argument texts.

Many text generation systems employ a model for discourse structure. RST[Mann and Thompson 1987] and Schema[McKeown 1985] 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 the multi-paragraph text without any interaction with the user. In fact, Dulcinea uses the schema-like knowledge to generate the semantic contents of the text and to organize the text structure.

By schemas, however, it is difficult to represent the local relations between adjacent sentences within the blocks. We represent the relations between adjacent sentences with the RST-like representation form, called FTS (Functional Text Structure). The plans for each constituent of argument text generate the semantic contents and each linguistic structure which is represented by the FTS.

2.3 Generation Process of Dulcinea

The following is a brief review of the generation process of Dulcinea. At first, we set the standpoint of the system by giving it an argument goal. The system's standpoint is whether some states of affairs are good or not good. These are the only possible judgments of the state of affairs. Dulcinea makes the semantic contents of an argument justify the given argument goal according to its beliefs, and represents them with a data structure called the *Argument Graph*. Then, the argument strategy on linguistic text structure is applied to the argument graph to organize an abstract text structure, which is represented by the FTS. The FTS represents not only the semantic content, but also the text structure, according to the standpoint of Dulcinea, and the belief necessary to generate the persuasive argument text. The FTS produces various surface syntactic text structures. Finally, the best text structure is selected and used to form natural language expressions.

The rest of this section describes Dulcinea's belief contents, gives an argument graph for representing semantic contents, and gives the FTS representation form of the text structure.

2.4 Belief Contents and the Argument Goal

Dulcinea's standpoint, which is set by the given argument goal and system beliefs, is the basis of the argument. The beliefs consist of three types of belief contents: *Fact, Rule,*and *Judgment.* Every element of *Fact* is a belief that Dulcinea believes to be true in the real world. Every element of *Rule* is a causal relation between two states of affairs. Every element of *Judgment* is a belief content that the system regards as good or not good. Figure 1 is an example of beliefs.

We give one of the three kinds of modal expressions, defined by the judgments in the table below, for the state

- Rules
  1. enforce[obj=one-way-system, loc=L],
     enforce[obj=two-way-lane, loc=L, pol=0]
     ⇒ change[obj2=bus-route, loc=L].
  2. change[obj2=bus-route, loc=L]
     ⇒ decrease[obj2=passenger, loc=L].
  3. decrease[obj2=passenger[mod=bus]], loc=L]
     ⇒ abolish[bus, loc=L].
  4. enforce[obj=two-way-lane, loc=L]
     ⇒ dangerous[obj=pedestrian, loc=L].
  5. enforce[obj=two-way-lane, loc=L],
     turn-on[act=bus, obj=lights, loc=L]
     ⇒ dangerous[obj=pedestrian, loc=L, pol=0].
  6. enforce[obj=two-way-lane, loc=L]
     ⇒ dangerous[obj=enteringcar, loc=L].
  7. enforce[obj=two-way-lane, loc=L],
     set-up[obj=road-sign, loc=L]
     ⇒ dangerous[obj=entering-car, loc=L, pol=0].
- Facts
  1. enforce[obj=one-way-system, loc=Midosuji]
  2. enforce[obj=two-way-lane, loc=Midosuji, pol=0].
  3. change[obj2=bus-route, loc=Midosuji].
  4. change[obj2=passenger[mod=bus]], loc=Midosuji].
  5. enforce[obj=two-way-lane, loc=London].
  6. dangerous[obj=pedestrian, loc=London, pol=0].
  7. turn-on[act=bus, obj=lights, loc=London].
- Judgments
  1. ng[obj=abolish[mod=bus]].
  2. ng[obj=dangerous[obj=_]].

Figure 1: Contents of the Beliefs

of affairs to the system as the argument goals. In the table, A means some state of affairs, and $\bar{A}$ means the negative state of affairs of A. If a judgment $g(A)$ exists in the system's belief, then the system believes A to be good.

Argument goal	Assertion	Correspondent Judgement
$must(A)$	It must be A	$ng(\bar{A})$
$hb(A)$	It had better be A	$g(A)$
$may(A)$	It may be A	$\neg ng(A)$

Dulcinea converts the given argument goal to the correspondent judgment, and shows that this judgment is supported by its beliefs.

2.5 Constituents of the Argument Text

The argument texts consist of the grounds for the argument goal, the expected opposing arguments, its refutation and the examples. The plans are prepared for each constituent to generate its content. A brief description of the constituent follows.

- **Argument goal**

  The argument goal is given to the system first. It provides the system's standpoint. It is the conclusion of the text, which is mostly placed at the end of the text.

  Ex. 1 .... *Therefore, the two-way lane*[1] *must be enforced.*

- **Ground**

  The grounds are necessary to justify the argument goal. The type of argument goal and the beliefs are used to select the proper plan for generating the grounds. The plans that are very restricted based on the reasons for the states of affairs will be described in detail in Section 3. An example of the grounds is given below.

  Ex. 2 *Because the bus service stops if a two-way lane is not enforced, ...*

- **Opposing argument and its refutation**

  Showing the grounds is enough to justify the argument goal. But, add to this any expected opposing argument and its refutation increases the persuasiveness of the text. The system adds the pseudo-ground of the opposite argument goal as the opposing argument, and points out that it is incorrect by refuting it.

  Ex. 3 *Indeed enforcing the two-way lane seems to be dangerous for pedestrians, but they are safe if the buses turn their lights on.*

- **Example**

  The argument text with the example is more persuasive.

  Ex. 4 *For example, the number of passengers using the bus decreased, because the two-way lane was not enforced.*

2.6 Argument Graph

The semantic contents that consist of the above parts are represented by the argument graph. Figure 2 is an example of the argument graphs, which insist the argument goal *"The two-way lane must be enforced"*.

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. Nodes (2)~(5) with the assumed node (1) represent the grounds for justifying of the argument goal. The `cause` link in the graph means a general causation, and the `p` link is used to represent the assumed node. The term in the node (2) `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 node (2), the negative state of affairs of the argument goal which causes the *no_good* state of affairs, as node (5). Therefore, this causal relation is the ground for the argument goal. 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. The details of the ground, the opposing argument and the refutation of it are given in Section 3.

[1]A two-way lane is a lane which allows buses to drive the wrong way up a one-way street.

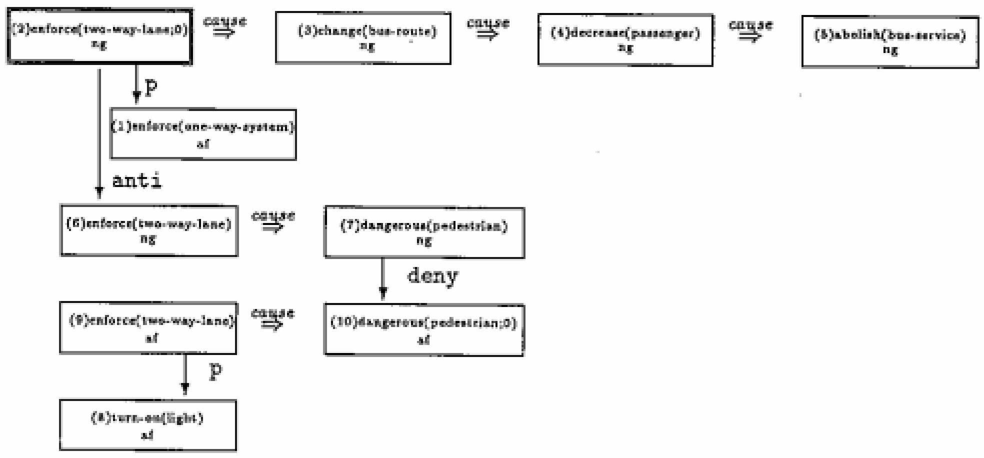


Figure 2: Argument Graph

2.7 A Structural Gap between the Argument Graph and a Linguistic Text Structure

The system realizes the semantic contents by natural language expression. However, the semantic text structure does not always correspond to the linguistic text structure. The various relations in the argument graph such as causation, temporal sequence, condition and assumption are expressed by various natural language connective expressions by considering the system's standpoint and beliefs. The direct realization of the propositional content makes for unnatural text. Therefore, the system must consider the judgment in relation to the propositional content and the role of the block that the propositional content is placed on such as the grounds and the opposing argument. One of the correspondent natural language expressions to the argument graph in Figure 2 is given below.

Ex. 5 *If the one-way system is introduced into a street*$_{(1)}$, *and a two-way lane is not enforced in the street*$_{(2)}$ *then the route of the bus service changes*$_{(3)}$, *the number of passengers decreases*$_{(4)}$, *and, unfortunately, the bus service eventually stops* $_{(5)}$.

Indeed enforcing the two-way lane$_{(6)}$ *seems to be dangerous for pedestrians*$_{(7)}$, *but they are safe*$_{(10)}$ *if the buses turn their lights on*$_{(8)}$, *even if the two-way lane is enforced*$_{(9)}$.

Nodes (2)∼(5) are linked by the cause link, but the surface expressions connect them naturally in a variety of ways. In (5), *"unfortunately"* is used to express the writer's negative attitude, this word communicates efficiently that nodes (2)∼(5) are the grounds. The expression *"Indeed ∼ seems to be ∼"* and *"even if ∼"* are used, because (6)∼(7) represent content that opposes the argument goal. These are denied by (8)∼(10). Since (6)∼(10) have a different content from (1)∼(5), two separate paragraphs are formed.

All these things make it clear that the surface natural language expression reflects the system's standpoint and beliefs besides the propositional contents. However, since there is a gap between the semantic contents and the natural language expression, the direct realization from the semantic data structure needs complicated processing because their is too much information to be referred to and a large variety of decision orders. To realize the proper expressions, as in the example above, it is necessary to not only refer to the relations between each state of affairs, but to consider how the partial structure relates to the whole text. A limited natural language expression is likely to be realized to avoid complexity, and such expressions cannot affect the reader.

In the early stages of the generation process, the organization of the linguistic text structure based on the linguistic strategy for the argument is important to realize the persuasive text, which utilizes the rich expressiveness of natural language. The linguistic strategy consists of the prescriptive descriptions of the developing topics and the local plans for each constituent to represent the local relations. In addition, we need the abstract representation form to represent the text structure generated as a result of structure planning.

2.8 FTS (Functional Text Structure)

We introduced the FTS as the abstract representation form. 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. Both the local relations between the states 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.

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

Gravitational constraint: The constraint of the distance between two sentences. The gravitational constraint S1–S2 means that the text in which sentence S1 is 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 `thesis` that takes a belief content as its value.

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 the connectives. The order of the sentences and the connection of the adjacent sentences is important to make 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.

3 Overview of the System

The argument text generation system Dulcinea consists of the four modules described below.

- **Generation of semantic contents of arguments**

  This module creates a data structure called an Argument Graph which represents the semantic content of arguments to justify a given argument goal according to the system's own beliefs.

- **Linguistic organization with argument strategy**

  This module creates an FTS from a given argument graph using linguistic knowledge. The FTS represents a whole text structure.

- **Clause level organization of orders and connections**

  Each leaf node in the FTS corresponds to a clause in natural language. This module adds order and connection information from each clause to the FTS.

- **Realization of texts**

  To realize natural language text from the FTS, appropriate words are selected. Tense, aspect and mood are fixed in this module.

These four modules are connected in sequence (Figure 3). Details on each module are described in the rest of this section.

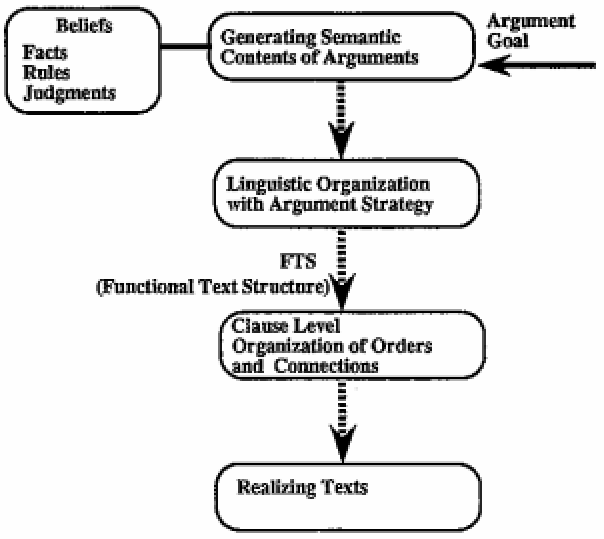

Beliefs
Facts
Rules
Judgments
Generating Semantic Contents of Arguments
Argument Goal
Linguistic Organization with Argument Strategy
FTS
(Functional Text Structure)
Clause Level Organization of Orders and Connections
Realizing Texts

Figure 3: Dulcinea's Architecture

3.1 Generating Semantic Contents of Arguments

This module creates an argument graph which represents the semantic content of the argument from the given goal. The module refers to a knowledge base which contains the system's own beliefs while creating the argument graph.

As described in the previous section, an argument consists of three different parts: grounds for the argument, refutations of the opposing arguments, and examples of the arguments or refutations. Every argument has at least one ground argument, whereas refutations of the opposing argument and examples are optional to the argument. We will describe the procedure for creating each part and combining those parts into one argument.

1. **Generation of grounds**

   The procedure for creating ground differs according to the type of goal. The procedures are as follows.

   (a) **goal type 1** (*must*, *hb*)

   The module searches a reason for believing a judgment corresponding to a given goal. If there is a rule in beliefs which predicts a result state B from a state A, and state B is believed to be good ($g(B)$), then state A is also believed to be good ($g(A)$).

$$\frac{A_1, A_2, ..., A_n \Rightarrow B \quad A_2 \sim A_n \quad g(B)}{g(A_1)} \qquad \frac{A_1, A_2, ..., A_n \Rightarrow B \quad A_2 \sim A_n \quad ng(B)}{ng(A_1)}$$

   In the course of applying these schemas, states $A_2 \sim A_n$ are proved by applying rules backward. If those states cannot be proved by the schemas below, the module assumes those states hold in its belief.

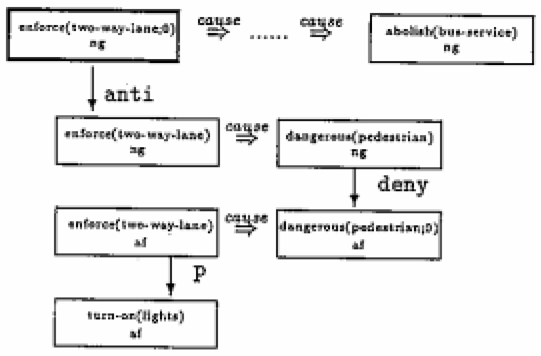


Figure 5: Generation of Refutation of Opposing Argument

$A_1, A_2, ..., A_n \Rightarrow B$
$A_1 \sim A_n$
B

The result of application of rules is represented in an argument graph. Figure 4 shows an example.

(b) **goal type 2** (*may*)

A goal of the form $may(A)$ corresponds to a judgment $\neg ng(A)$. We cannot obtain this type of judgment using the schema above. we define the semantics of $\neg ng(A)$ as follows, "There seems to be grounds for a judgment $\neg ng(A)$. But, in fact, there is a refutation to the argument"

This idea is also used in the creation of refutations of an opposing argument.

2. **Generation of opposing arguments and their refutation**

An argument A_1 whose goal is contrary to the goal of argument A_2 is called an opposing argument of A_2. Its goal and its opposing argument's goal are listed below.

Argument goal	Opposing goal
$must(A)$ $(= ng(A))$	$ng(A), g(A)$
$hb(A)$ $(= g(A))$	$ng(A), g(A)$

The module creates the pseudo-ground for the goal opposing the original goal. Find, then, creates the refutation to the opposing argument. Figure 5 shows an example of the refutation of the opposing argument.

3. **Generation of examples**

We define a pair of facts which are unifiable to a rule as an "example" of the rule. By attaching examples to the rules in an argument, we can reinforce the argument (See Figure 6).

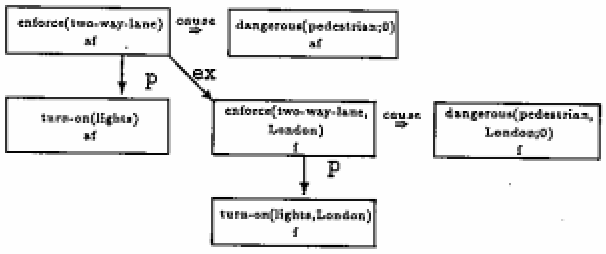


Figure 6: Generation of the Example

3.2 Linguistic Organization with Argument Strategy

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, the structure of the graph is independent of the natural language expressions to be generated. Therefore, in order to generate the argument text, the argument graph should be translated into the FTS, which can be transformed into suitable natural language expressions.

First of all, basic constituents in the argument graph such as the ground, the example and the refutation of the opposing argument are recognized. Then the order of these constituents is decided according to the prescriptive knowledge. The order is described using the order constraints of the FTS.

For instance, the opposing arguments are placed before the argument goal. The examples are placed after the ground. The ground is realized earlier than its opposing arguments and refutations of it.

At the same time, each constituent is transformed into the FTS-term according to the transformation rules. Those constituents which cannot be used for the argument or would make the text unnatural are ignored.

The following shows the transformation rules defined for each constituent of the argument graph.

1. **Generation of the ground**

A causal relation in the argument graph is transformed into a new term which has a pair of labels cause and result. If the causal relation has precondition link p, then the content of the precondition with a label p_cond is added to the term. For example, the argument graph in Figure 4 is transformed into the following FTS-term. The FTS-term which represents the ground for the given argument goal has an attribute fts_type and its value main.

```
[thesis=[set={
         [thesis=[p_cond=enforce(two-way-lane),
                  cause= enforce(two-way-lane;0),
                  result=change(bus-route)]],
         [thesis=[cause= change(bus-route),
                  result=decrease(passenger)]],
         [thesis=[cause= decrease(passenger),
                  result=abolish(bus-service)]]}],
 fts_type= main]
```

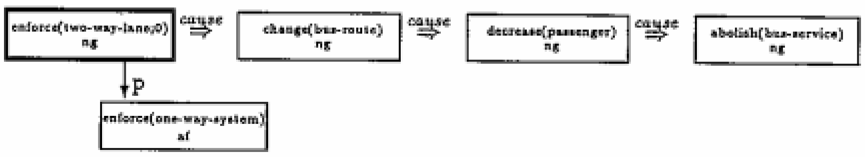


Figure 4: Generation of the Ground

2. **Generation of the opposing argument and the refutation of it**

   The FTS-term which represents the opposing argument is generated with a label anti_t in the FTS-term which represents the ground. The contents of the opposing argument in the argument graph are transformed into the FTS-term in the same way as the transformation of the ground part.

   The FTS-term which expresses refutation of the opposing argument has an attribute fts_type with the value anti_deny. The following shows the FTS-term corresponding to the argument graph in Figure 5.

```
[set={
 [thesis=t1:...,
  fts_type= main],
 [thesis=t2:[though=enforce(two-way-lane),
            assume=turn-on(lights),
            result=dangerous(pedestrian;0)],
  anti_t=t3:[thesis=
             [seem=
              [thesis=
               [cause=enforce(two-way-lane),
                result=dangerous(pedestrian)]]]],
  fts_type=anti_deny]
     }]
order constraints: t1<t2, t3<t1
```

3. **Generation of the example**

   The FTS-term which stands for the example of the ground is generated with the label exampl in the FTS-term expressing the ground. The FTS-term that corresponds to the argument graph in Figure 6 is as follows.

```
[thesis=
  [though=enforce(two-way-lane),
   assume=turn-on(lights),
   result=dangerous(pedestrian;0)],
 exampl=
  [thesis=dangerous(pedestrian,London;0),
   crecog=[set={
           [thesis=enforce(two-way-lane,London)],
           [thesis=turn-on(lights,London)]}]]]
```

3.3 Clause Level Organization of Orders and Connections

This module 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.

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 $A \rightarrow B$ and $B \rightarrow C$ 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.

1. Generates every possible connection relation from the given FTS.
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 (Table 3).

We illustrate the criterion for the bad dependency structure using the FTS below.

```
[thesis= t1
 anti_t= t2
 exampl= t3]
t2 < t1,  t2 < t3
```

In this case, we can generate sentences in two different orders.

1. t2<t1<t3
2. t2<t3<t1

Figure 7 represents the dependency structure of each text. Since dependency structure 2 does not have direct dependency between t2 and t3, the reader cannot find the semantic dependency of t3 when t3 is reached while reading this text. They can find the semantic dependency only after reading through t1. This means that the text in order 2 is much more difficult to understand

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

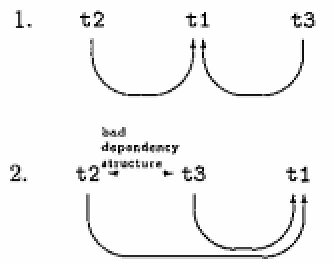


Figure 7: Dependency Structure

than that in 1. We call the dependency structure in 2 the bad dependency structure.

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

```
[thesis= t1
 anti_t= t2
 exampl= t3
 order= [2,3,1],
 conn= [(negation1,s),(example,s)]]
```

3.4 Realizing Texts

This process realizes the content of the FTS added order and conn attributes in terms of natural language expressions. The FTS with order and conn is the 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."*

4 Example

In this section, we show an example of Dulcinea's argument. The beliefs used for the argument are shown in Figure 1.

When the argument goal

```
must(cont=enforce(obj=two-way-lane)),
```

which means *"The two-way lane must be enforced."*, is given to Dulcinea, it generates the argument graph shown in Figure 9 according to the beliefs.

Then this argument graph is transformed into the FTS, to which information on sentence order and connectives are added afterwards. The FTS with these two kinds of information is shown in Figure 10.

From this FTS structure the argument text shown in Figure 8 is realized.

5 Conclusion

We have described the argument text generation system Dulcinea, which generates text to justify the argument goal. The text, which reflects the standpoint and the judgment of the system, is represented by the FTS. The FTS represents not only the semantic contents of the argument but the system's standpoint, the judgments and the linguistic constraints.

In addition to the generation frame-work, we have investigated the argument strategy to generate coherent and persuasive argument texts. This strategy is plausible for the multi-paragraph text generation in a very narrow domain, but we believe that it is one of the answers to the question: "What relations, plans and schemas are necessary to support the planning of coherent multi-paragraph texts?".

御堂筋で一方通行を実施して，逆行レーンを実施しなかった．その結果，バスルートが変化した．そのため，バスの乗客が40%減少してしまった．このように，一方通行を実施するときに，逆行レーンを実施しなければ，バスルートが変化し，バスの乗客が減少する．また，バスが廃止されてしまう．

一方，逆行レーンを実施すれば，歩行者が危険なようにみえる．しかし，バスのライトを点灯すれば，逆行レーンを実施しても，歩行者は危険でない．したがって，逆行レーンを実施しなければならない．

Generated Text

When the one-way system was introduced in *Midosuji* street, a two-way lane was not enforced. As a result, the route of the bus service changed. Therefore, the number of passengers decreased by 40%. In this way, when a one-way system is introduced to a street, if a two-way lane is not enforced, then the route of the bus service changes, and this makes the number of passengers decrease. Finally, the bus service is abolished.

On the other hand, enforcing the two-way lane seems to be dangerous for pedestrians. But they are safe if the buses turn their lights on. Therefore, the two-way lane must be enforced.

Translated from Japanese

Figure 8: Argument Text

To generate more coherent and natural texts using the rich expressiveness of natural language, some user model and more complicate conversational settings will be necessary in the future.

Acknowledgments

Thanks are due to TANAKA Yuichi for reading the draft and making a number of helpful suggestions. We also wish to thank all the members of the Sixth Research Laboratory and the members of working group NLU for valuable advice.

References

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

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

[Hovy 1985] Eduard H. Hovy. Integrating text planning and production in generation. In *the Proceedings of the International Joint Conference on Artificial Intelligence*.

[Hovy 1987] Eduard H. Hovy. Interpretation in generation. In *the Proceedings of 6th AAAI Conference*.

[Hovy 1988] Eduard H. Hovy. *Generating Natural Language under Pragmatic Constraints*. Lawrence Erlbaum Associates, Publishers, 1988.

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

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

[Joshi 1987] Aravind K. Joshi. Word-order variation in natural language generation. In *the Proceedings of 6th AAAI Conference*.

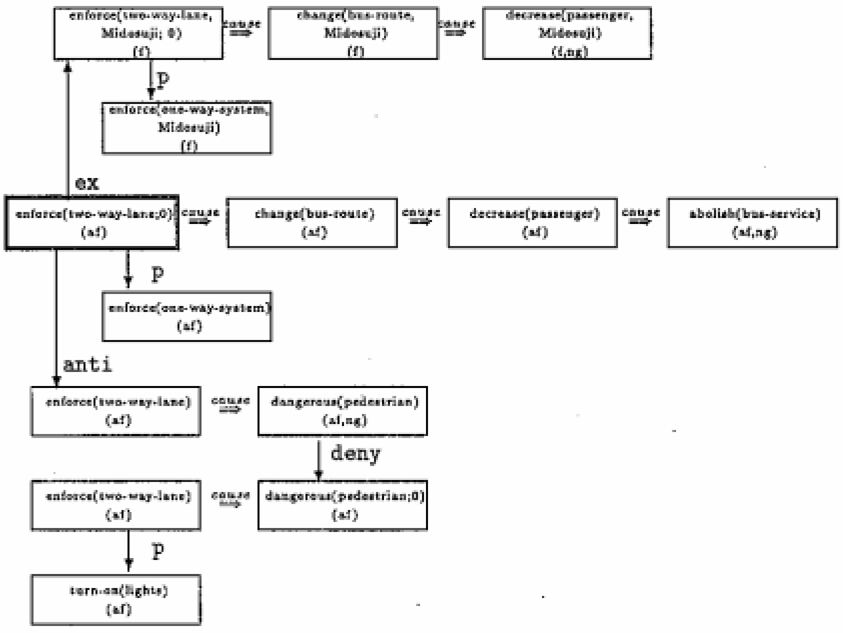
[Mann and Thompson 1987] W. C. Mann and S. A. Thompson. Rhetorical structure theory: Description and construction of text structures. In *Natural Language Generation: New Results in Artificial Intelligence, Psychology, and Linguistics*. Dordrecht: Martinus Nijhoff Publishers, 1987.

[McDonald and Pustejovsky 1985] David D. McDonald and James D. Pustejovsky. Description-directed natural language generation. In *the Proceedings of the International Joint Conference on Artificial Intelligence*.

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

[McKeown and Swartout 1988] 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.

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



enforce(two-way-lane, Midosuji; 0) (f)
cause
change(bus-route, Midosuji) (f)
cause
decrease(passenger, Midosuji) (f,ng)
p
enforce(one-way-system, Midosuji) (f)
ex
enforce(two-way-lane;0) (af)
cause
change(bus-route) (af)
cause
decrease(passenger) (af)
cause
abolish(bus-service) (af,ng)
p
enforce(one-way-system) (af)
anti
enforce(two-way-lane) (af)
cause
dangerous(pedestrian) (af,ng)
deny
enforce(two-way-lane) (af)
cause
dangerous(pedestrian;0) (af)
p
turn-on(lights) (af)

Figure 9: The Argument Graph

```
[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%,ten=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 10: FTS with order and conn attributes