ICOT Technical Memorandom: TM-1195

TM-1195

Generation Mechanism of Argument Contents in Argument Text Generation System (Dulcinea)

by

K. Hagiwara, Y. Kubo, 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

Generation Mechanism of Argument Contents in Argument Text Generation System (Dulcinea)

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

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

We have investigated human written argument texts and extracted a number of features which make those texts persuasive and coherent. Based on those investigations, we have built an experiment system for argument text generation.

The system first creates an argument content on a given theme referring to its belief. In order to create a persuasive argument, the system utilizes strategy for argument. The argument content is then converted into a text structure called FTS (Functional Text Structure). The FTS represents a text structure in addition to the semantic content. In order to realize real text, the system defines the order of the sentences and the connection of the adjacent sentences in the FTS. We have found that the order of each sentences and the connection of adjacent sentences are crucial if text are to be read easily. Finally, a complete argument text is realized in the target language.

This paper is intended to investigate the structure of human written argument text, and mechanisms for generating argument text contents by computer.

1 Introduction

In general, a natural language generation system consists of two phases, namely text planning and linguistic realization. In the text planning phase, the content of a text is generated in a form independent from natural language, and the text structure is created from the content. The text structure specifies the paragraphs and their order, and the orders of sentences in the paragraphs. The text structure is passed to a linguistic realizer, which generates text in the target language.

In recent years, the focus of research in natural language generation has shifted from the generation of isolated sentences to the production of coherent and purposeful text which consists of a number of paragraphs. In other words, much attention has been paid to the text planning phase.

The purpose of our research is to find schemes for generation of purposeful and coherent text[14]. In order to achieve the goal, we have investigated the structures of real argument texts written by humans, and extracted a number of features that those texts have. Based on those investigations, we have created an experiment system for argument text generation, Dulcinea. Though the schemes for generation of purposeful text contents may be specific to texts such as argument texts, the schemes for generation of coherent text will be general enough to apply to other types of text.

In section 2, we will illustrate the structure of the argument texts written by humans. Section 3 describes a brief overview of our argument generation system Dulcinea. And, in Section 4, the generation mechanism of argument contents is described in detail. The planning strategy of the text structure will be described in our other paper which will appear in these proceedings. [9].

2 Argument Texts

An argument text is a set of statements in support of an opinion, and is used to try to convince someone that the opinion is correct. We have selected the argument texts as the subject of our generation text, because they have a clear purpose (convincing the reader) and the generated text can be evaluated objectively against the purpose.

2.1 Argument Texts Written by Humans

Argument texts written by humans have a variety of text structures. We can easily generate a number of argument texts even for the same opinion. But if we carefully look at the abstract structure of those argument texts, we can find four basic components in them.

- Introduction
- Grounds
- Refutations of the opposing arguments
- Conclusion

To illustrate these components, we will give a short argument text as follows.

Recently, bus services have been under evaluation as a city transportation systems. Under this, the number of bus services has reduced, according to the passengers decrease.

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, introducing a two-way lane seems to put pedestrians in danger. However, the pedestrians were not in danger in London when a two-way lane was introduced, because the buses turned their headlight on. In this way, if the buses turn their headlight on, the pedestrians will not be in danger, even though a two-way lane is introduced.

Therefore, the two-way lane must be enforced.

example argument text

This text argues for the enforcement of a two-way lane.

In the first paragraph the point at issue is described. We call this paragraph, the **introduction** of the argument. The purpose of the introduction is to describe the necessity of the argument, and call the attention of the reader.

The second paragraph describes the reasons for supporting enforcement of the two-way lane. We call this paragraph the **ground** of the argument. The function of the ground is to give reasons for supporting the opinion. In many cases, the ground includes **examples**. In the text above, the case in Midousuji street is first shown as an example, this is then followed by the body of the ground. The examples make the argument concrete and reinforces the ground. The ground is the kernel of the whole argument text, and is indispensable.

In the third paragraph, **refutations of the opposing arguments** are given. This component is quite specific to the argument texts. The purpose of this component is to reinforce the ground by refuting the opposing arguments to the ground. The component may also include examples. In the text above, the case in London is shown as the example. Generally speaking, the argument text may have a number of opposing arguments and refutations, though the sample text above has only one.

The final paragraph concludes the whole text by mentioning the opinion being argued. We call this paragraph the **conclusion** of the argument.

Besides these four components, definitions of concepts and terms will appear in the text.

In real human written argument texts, these components may be jumbled, and may not appear in each separate paragraphs. However, the main issues in the argument texts belong to one of those four components.

2.2 Model of Argument Texts

We have extracted three components from argument texts written by humans for our argument text model.

- Grounds
- Refutations of the opposing arguments
- Conclusion

Our system generates argument text with these three components and includes examples. **Introduction of argument** has been omitted in our model, because it is beyond our approach to the argument content generation scheme.

3 Overview of the System

The argument text generation system Dulcinea consists of four modules. We will describe these modules briefly in this section. The generation mechanism of argument contents will be described in detail in the next section. Other modules will be described in our other paper which is to appear in these proceedings [9].

- **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 complete 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 1).

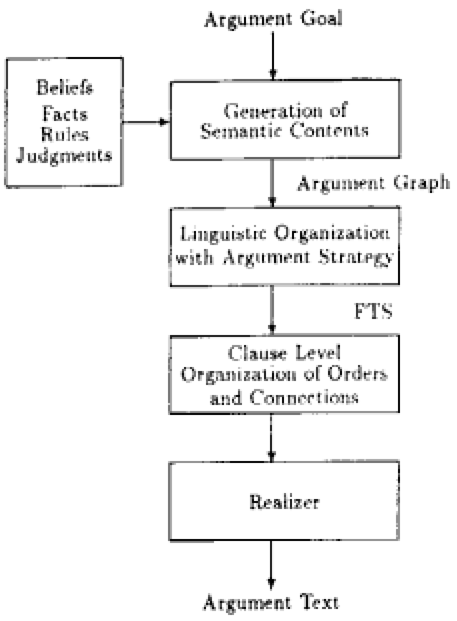

Argument Goal
Beliefs
Facts
Rules
Judgments
Generation of Semantic Contents
Argument Graph
Linguistic Organization with Argument Strategy
FTS
Clause Level Organization of Orders and Connections
Realizer
Argument Text

Figure 1: Dulcinea's Architecture

4 Generation Mechanism of Argument Contents

The content of the argument which is represented as the **argument graph** is generated from a given argument goal referring to the systems belief. As described in the previous section the argument content consists of three components, the ground, refutations to opposing arguments, and the conclusion. These components may also include examples.

We will describe the contents of belief, the argument goal, the argument graph, and the generation mechanisms of argument contents in this order.

4.1 Belief

Dulcinea's belief consists of a set of belief contents which fall into one of three types of beliefs, *Fact*, *Rule* and *Judgment*.

Fact is a belief content that Dulcinea believes to be true in the real world. Every element of *Rule* is a causal relation between two states of affairs that Dulcinea believes to hold in the real world. *Judgment* is a belief content that the system regards as good or not good.

Figure 2 is an example of belief.

The first rule in the example above represents a causal-relation that,

> "if a one-way-system is enforced at some location, and a two-way-lane is not enforced at the location, then the bus route will change at that place"

The first fact in the example above represents the fact that,

- 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 2: Contents of the Beliefs

> "A one-way-system has been enforced in Midousuji street"

Finally, the first judgment in the example above represents a judgment that,

> "abolition of bus service is not good"

To generate argument contents in support of some opinion, the module must make use of the judgment. Without the judgment the module cannot support any opinion because it cannot judge the goodness or badness of any event.

4.2 Argument Goal

We give one of the three kinds of modal expressions as the argument goal to the system. Each modal expression has a correspondent expression of judgments (Table 1).

In the table, A stands for some state of affairs, and $\bar{A}$ stands for the negative state of affairs of A.

The module converts the given argument goal to the correspondent judgment, and shows that the judgment is supported by its belief. For instance, to generate grounds for argument goal $must(A)$, the module tries to find the reason for $ng(\bar{A})$.

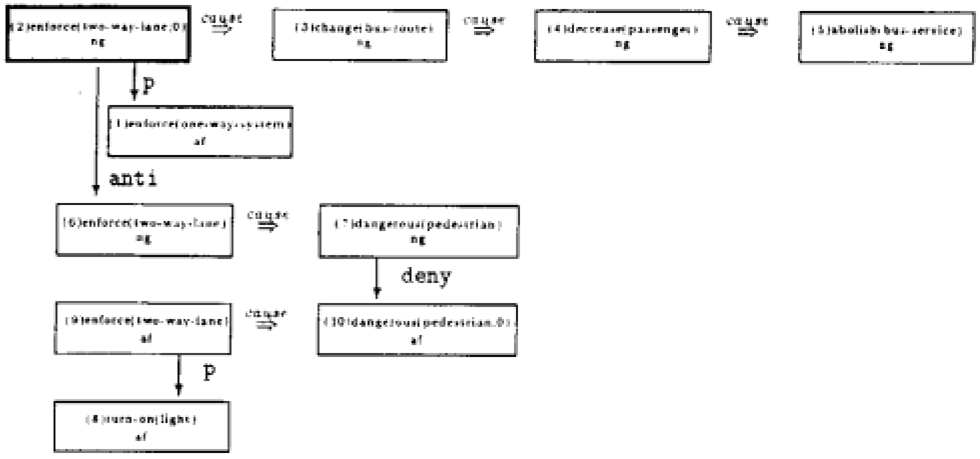


Figure 3: Argument Graph

Argument goal	Assertion	Correspondent Judgment
must(*A*)	It must be *A*	*ng*(*A*)
hb(*A*)	It had better be *A*	*g*(*A*)
may(*A*)	It may be *A*	¬*ng*(*A*)

Table 1: Argument Goals and Judgments

4.3 Argument Graph

The semantic contents that consist of the above parts are represented by the argument graph. Figure 3 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 **soa** (state of affairs). Ng in the nodes indicates that the soa 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 the next section.

4.4 Generating Semantic Contents of Arguments

Generation of semantic content of an argument consists of two major steps.

First, the content of the ground for the whole argument is generated from the given argument goal. If the generation of the ground content fails then the generation of the whole argument content fails, because the ground is indispensable to all arguments.

Second, the ground is reinforced by adding the refutations of the opposing arguments to the ground, and by giving examples to the ground and refutations. The module adds as many refutations and examples as possible. However refutations and examples are optional to an argument, so if none are found, an argument content with the ground only is generated. All these tasks are done by handling argument graphs.

1. Generation of content of ground
2. Reinforcement of the ground
   (a) add refutations of opposing arguments to the ground
   (b) give examples to the grounds and the refutations

In general, more than one argument content may be generated from one argument goal. In those cases, the module selects the first ground content that has been generated. On the other hand, all the refutations of the opposing arguments and examples are attached to the ground.

In order to generate persuasive arguments, the module has to select the most convincing combination of ground, refutations and examples based on some constraints. But the system does not have those in the state-of-the-art features. We will consider the matter at the conclusion of this paper.

In the reminder of this section, we will examine mechanisms for generating grounds, refutations of the opposing argument, and examples in turn.

4.4.1 Generation of grounds

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

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

   The module searches for a reason to believe a judgment corresponding to a given goal. If there is a rule in beliefs which predicts a result state *B* from a

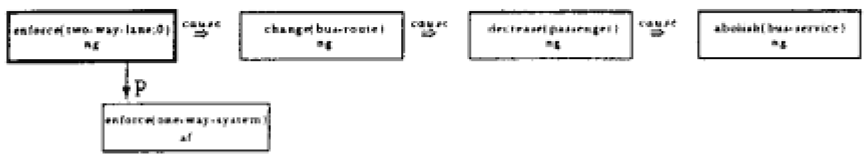


Figure 4: Generation of the Ground

state A, and state B is believed to be good ($g(B)$), then state A is also believed to be good ($g(A)$).

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

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.

$$\frac{A_1, A_2, \ldots, A_n \Rightarrow B \quad A_1 \sim A_n}{B}$$

The result of application of rules is represented in an argument graph. Figure 4 shows an example argument graph. This argument graph represents a ground of argument goal $must(cont = enforce(obj = twowaylane))$.

The process of generation starts by converting the goal in the form of judgments ("not enforcing a two-way-lane is bad"). Since the judgment is not directly supported by the system's belief, the module tries to apply a rule in the belief to predict what will occur if the two-way-lane is not enforced. Fortunately, the module finds the following rule in its belief.

> enforce[obj=one-way-system, loc=L],
> enforce[obj=two-way-lane, loc=L, pol=0]
> ⇒ change[obj2=bus-route, loc=L].

To apply the rule, the module tries to support a soa (state of affairs) "enforcing a one-way-system at some location" in its belief. Since it is not found, the module assumes that the soa is true in the course of this argument. Now, the module tries to support a judgment "change of the bus-route is bad". Since this is also not supported directly by the belief, application of rules succeeds until the judgment "abolition of bus-service is bad", which is included in the belief, is reached.

2. **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 $ng(A)$. But, in fact, there is a refutation to the argument"

   A scheme for creation of refutations of an argument is the same as a scheme for creating refutations to an opposing argument.

4.4.2 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
$must(A)$ $(= ng(\neg A))$	$ng(A), g(A)$
$hb(A)$ $(= g(A))$	$ng(A), g(A)$

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

In this example, the opposing argument is

> "enforcement of a two-way-lane is bad, because it will put pedestrians in danger".

This opposing argument is a one step application of the following rule.

> enforce[obj=two-way-lane, loc=L]
> ⇒ dangerous[obj=pedestrian, loc=L].

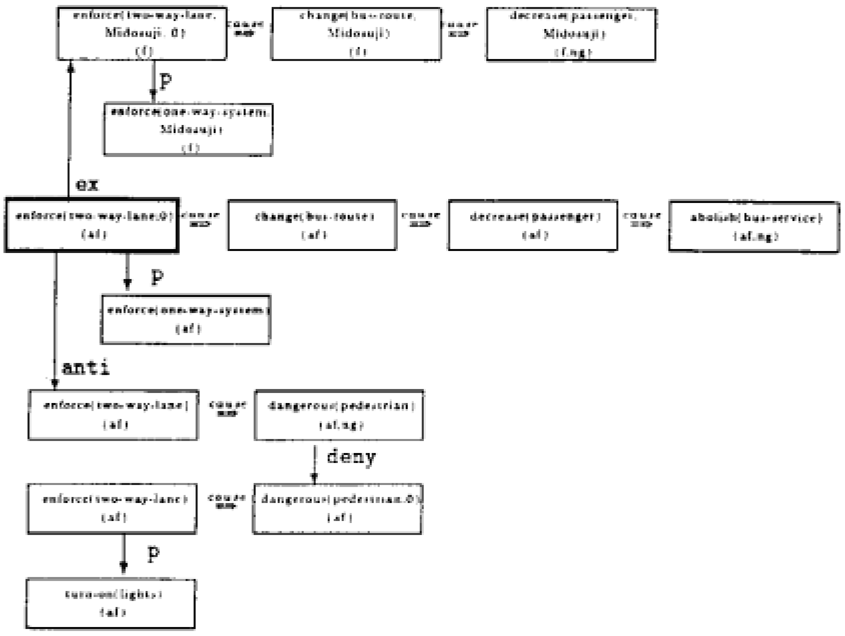
To refute this argument, the module tries to find exceptions to the rule. Since, we have the following rule in the belief, the opposing argument can be refuted.

> enforce[obj=two-way-lane, loc=L],
> turn-on[act=bus, obj=lights, loc=L]
> ⇒ dangerous[obj=pedestrian, loc=L, pol=0].

4.4.3 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).

Finally, we get the argument graph with a ground, refutations of opposing arguments, and examples (Figure 7).



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 7: complete argument graph

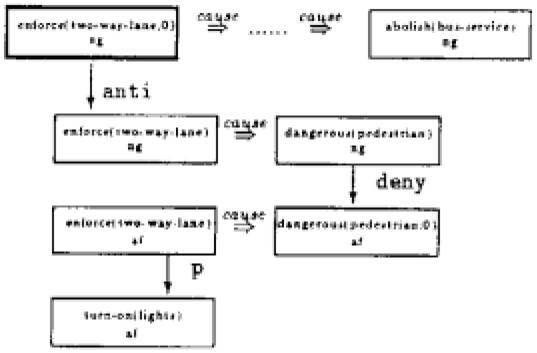
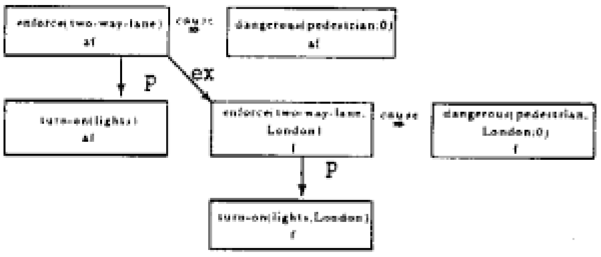


Figure 5: Generation of Refutation of Opposing Argument



enforce(two-way-lane) af
cause
dangerous(pedestrian,0) af
P
ex
turn-on(lights) af
enforce(two-way-lane, London) f
cause
dangerous(pedestrian, London,0) f
P
turn-on(lights,London) f

Figure 6: Generation of the Example

5 Experiments

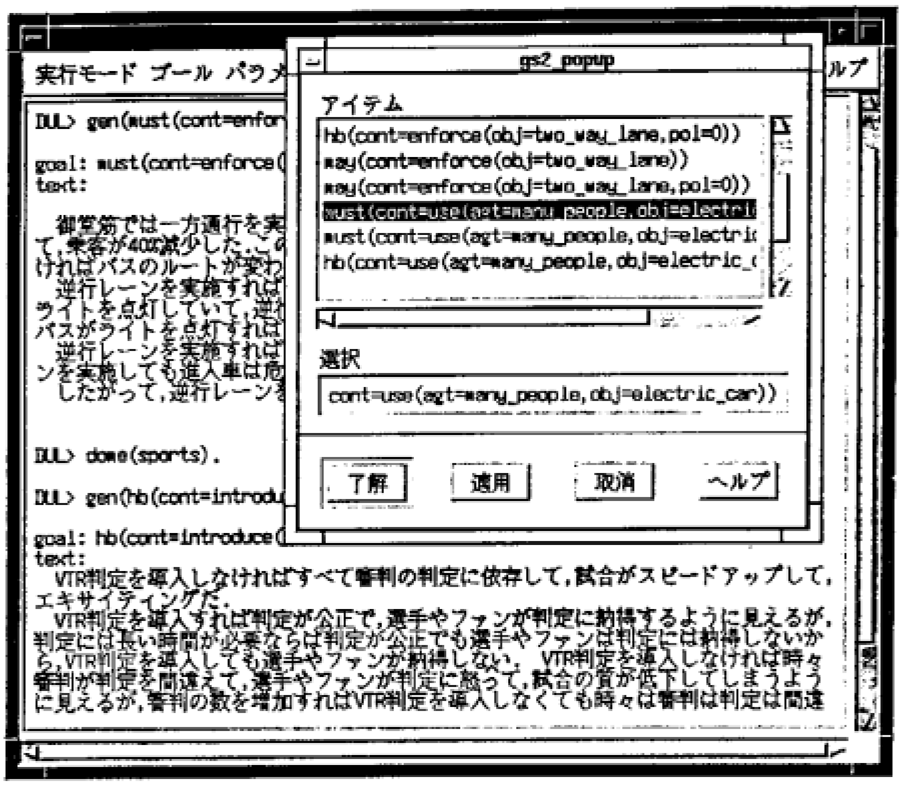
Dulcinea has been built upon KL1[13] and runs on PIM (Parallel Inference Machine)[10]. The GUI (Graphic User Interface) for Dulcinea has also been built (Figure 8). Using the GUI, we can set and modify parameters of argument strategies, and we can retrieve the data (Beliefs, Argument Graphs, and FTS).,

A number of experiments on argument generations have been done on the system. We have tested the generation mechanism on six domains (Table 2). Each domain's belief consists of about 10 facts, 20 rules and 5 judgments.

As a result, we have realized that our generation mechanism of argument contents is general enough to generate a large number of arguments in various domains.

6 Conclusion

We have described the mechanism for generating argument text contents. The mechanism reflects the schemas we use for writing argument texts. In the generation of persuasive contents, we found that the reinforcement of the ground argument by adding refutations of opposing



```
DUL> gen(must(cont=enfor

goal: must(cont=enforce(
text:

  御堂筋では一方通行を実
て,乗客が40%減少した.こ
けれはバスのルートが変わ
  逆行レーンを実施すれば
ライトを点灯していて,逆
バスがライトを点灯すれは
  逆行レーンを実施すれば
ンを実施しても進入車は危
  したがって,逆行レーンを


DUL> dome(sports).

DUL> gen(hb(cont=introdu

goal: hb(cont=introduce(
text:
  VTR判定を導入しなければすべて審判の判定に依存して,試合がスピードアップして,
エキサイティングだ.
  VTR判定を導入すれば判定が公正で,選手やファンが判定に納得するように見えるが,
判定には長い時間が必要ならば判定が公正でも選手やファンは判定には納得しないか
ら,VTR判定を導入しても選手やファンが納得しない.  VTR判定を導入しなければ時々
審判が判定を間違えて,選手やファンが判定に怒って,試合の質が低下してしまうよう
に見えるが,審判の数を増加すればVTR判定を導入しなくても時々は審判は判定は間違
```

```
hb(cont=enforce(obj=two_way_lane,pol=0))
may(cont=enforce(obj=two_way_lane))
may(cont=enforce(obj=two_way_lane,pol=0))
must(cont=use(agt=many_people,obj=electri
must(cont=use(agt=many_people,obj=electri
hb(cont=use(agt=many_people,obj=electric_

cont=use(agt=many_people,obj=electric_car))
```

Figure 8: User interface of Dulcinea

Domain	Theme
traffic	enforcement of two-way-lane
	development of electric cars
whaling	prohibition of commercial whaling
labor	reduction of working hours
house	purchase of own house
	living with one's parents
medicine	admission of mercy killing
	admission of brain death as human death
sports	introduction of VTR judgment
	doping on players

Table 2: Domains and Themes

arguments and giving examples to them is important.

To generate more persuasive argument contents, we have to make use of the reader's belief model.

References

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

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

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

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

[9] Kubo Yukihiro Hagiwara Kaoru, Ikeda Teruo, and Kotani Akira. Planning strategy of text structures in argument text generation system dulcinea. In *IEICE, NLC*, 1992.

[10] Taki Kazuo. Parallel inference machine PIM. In *International Conference on Fifth Generation Computer Systems 1992*, volume 1, pages 50–72. ICOT, 1992.

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

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

[13] Chikayama Takashi. Operating system PIMOS and kernel language KL1. In *International Conference on Fifth Generation Computer Systems 1992*, volume 1, pages 73–88. ICOT, 1992.

[14] 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.