Flexible Diagram Generation from Tagged Texts

Masashi MURAYAMA Yuichi NAKAMURA Yuichi OHTA
IEMS, University of Tsukuba, 305-8573 Japan

Abstract. This paper introduces a novel diagram generation scheme for presenting the inner structures of a text. We first describe how semantic structures of a text can be translated into a diagram: tagging to a text, followed by translation to a diagram. The generated diagrams are tightly linked to the texts, and the media complex of the text and the diagram effectively explains the contents of a document. In this paper, we focus on generating a variety of diagrams according to both the contents and the user's intention. This approach allows us to obtain comprehensible diagrams that serve the user's purposes.

1 Introduction

Recently, the Internet has given us easy access to vast amount of electronic texts. Although the benefits of this are great, it is often difficult to locate the desired portions, since reading is often time-consuming work. The linear structure of a natural language often forces us to write or read complicated texts.

A diagram, such as a flow chart or a graph, can be an effective solution to this problem. A diagram that uses structures of two or more dimensions can represent complicated notions or relationships. Based on this idea, we have proposed a novel scheme of diagram generation that translates the inner structures of a text[4]. This method is composed of the following processes:

- Embedding tags for marking up the semantic structure of a text.
- Generating diagrams according to translation rules that reflect a user's intuition and customs. Generated diagrams are tightly linked to the text.
- Providing a user interface that presents both the text and the diagrams, and enables quick access to the essence of the text.

In this paper, we introduce an important extension of this framework. Since in our previous research the text-to-diagram translation rules had a one-to-one correspondence and there were only a few of them, the diagrams were generated in a fixed way even when they were confusing or might not meet the user's purpose. As a solution to this problem, we prepared a variety of translation rules, allowing us to obtain comprehensible diagrams that serve the user's purposes.

In the following section, we describe the framework of our research. We then present the correspondence between diagrammatic structures and our XML tagset, and propose the use of many-to-many correspondence. Finally, we show some experiments using our method.

2 Framework of Diagrammatic Representation

2.1 Key Idea

The key idea of our framework is to automatically generate a comprehensible diagram that corresponds well with the semantic structures in a text. Diagram representation can be a useful tool for explaining the contents of a text, if we maintain conventional rules[3] for drawing diagrams. For humans this is not difficult, since the rules are loose and diagrams can be designed without seriously breaking the rules. Often, however, this is time-consuming work, and it would be prohibitive to prepare a diagram to explain every portion of a text.

In this sense, our scheme of diagram generation can contribute to information presentation and management to prevent information flood. Figure 1 shows an example of our system output. The left half of the figure shows a tagged text [1] and the right half shows a diagram translated from the text. In this example, as the user specifies some portion that has drawn his or her attention, the corresponding portion of the text is highlighted.

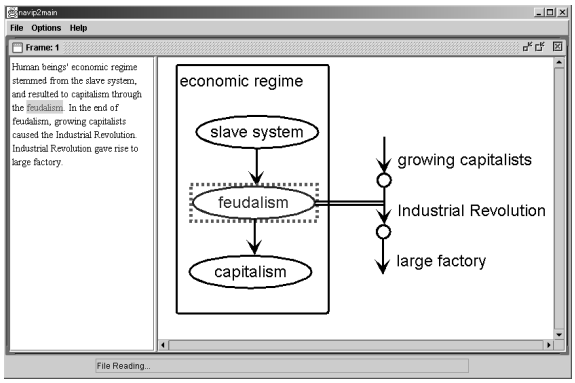


Figure 1: Our framework for diagram generation

For this purpose, we have developed our system mainly on the following points:

- We defined a tagset for marking up portions that can be effectively represented by diagrammatic expressions.
- We determined and implemented the translation rules from the semantic structures to diagrammatic expressions. Our system also allows and assists the user to modify a generated diagram, since it is still difficult for the system to arrange a perfect diagram.
- We developed a GUI that holds and shows tight links between the diagram and the text. Since a diagram does not have enough information to completely substitute for a text, the media complex of texts and diagrams is the most effective representation.

Below we describe the first point and then, in the next section, we describe the second point and the extension of the present study to the previous research.

2.2 Tagging to a Text

Texts contain various kinds of relations, from syntactic structures to deep semantics, some of which can be effectively represented by diagrams. We have proposed a tagset for marking up such semantic structures in a text. A tag describes the type of the semantic structure that each portion has, and the system illustrates the portion based on the type.

We designed the tagset based on the XML standard, and we can use ordinary tools for XML documents. Table 1 shows actual tags. `<node>`, or `<n>`, is used to mark an the element such as a word, a phrase, or a sentence. This tag attaches an ID, a role, and a reference to the element. `<relation>` marks up the relation between elements. It has two important attributes: `structure` shows the structural category of the relation, and

[1] The tags are invisible in this viewer, since we assume that the users are not good at reading mark-up languages.

Table 1: Tagset for describing semantic structures of a text

Tag	Note and `attributes`
`<node>` or `<n>`	Mark up An element. `id`= identifier, `nref`= reference to other ID `role`= role of the element in composing relation
`<relation>`	Mark up a relation `structure`= structural category of the relation `semantics`= semantic category of the relation

Table 2: Structural categories

value	Type of relation	Required roles of element
`order`	order, series, causality, hierarchy, inclusion, subordination, etc.	upper , medium, lower
`equivalence` (`equiv`)	equivalence, equality, coordination, etc.	obj(object), eq(or neq)
`modification` (`mod`)	explanation, attribution, etc.	obj(object), mod(modifier)
`other`	general relations(excepting the above)	obj(object)

Table 3: Semantic categories

`time`	chronology, time sequence, etc.
`cause`	causality. cause and effect, or reason and result
`space`	location or locus in physical world
`inout`	means "Input/Output". material, product, input and output, etc.
`process`	process, flow of things, flow of topic, etc.
`set`	organization, theory of sets, subordination, etc.
`other`	other semantics

`semantics` shows the semantic category of the relation. The structural category classifies the relation based on algebraic characteristics. Table 2 shows this classification. `order` means the most common category that expresses an order relation having both upper and lower elements. `equivalence` means the equivalent relation. `modification` means the relation between a modifier and the word that is modified. `other` is the category selected when none of the above types fits. A semantic category denotes what kind of property a relation specifies. Table 3 shows the semantic categories that we are currently using.

By the combination of these two types of categorization, we can flexibly denote each relation. For example, a relation may refer to chronological order by using "order" as its structural category and "time" as its semantic category[2].

This formalism is useful also for information filtering. If we want some information on chronological order, other structural categories or other kinds of `order`s (such as a spatial order, hierarchy, or subordinate) are useless, and we can easily filter them out. Thus, we can emphasize elements that are the focus of our interest.

[2]Hereafter, we denote the relations type by a tuple of structural category and semantic category, such as order&time.

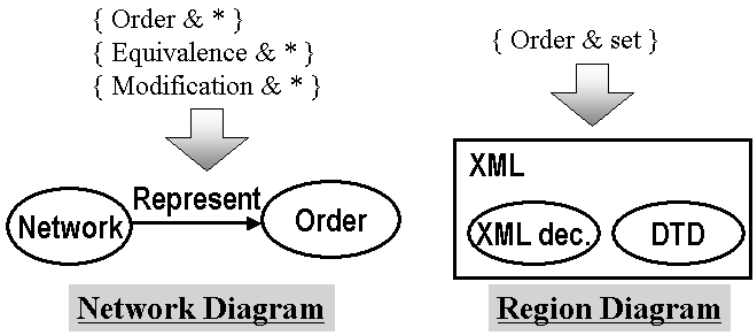


Figure 2: Simple correspondence used in our previous research

(a)alignment	(b)simple link	(c)double link	(d)arrow link	(e)triangle link	(f)consecutive link
A B	A B	A B	A B	A B	A B
(g)enclosure	(h)branch	(i)balloon	(j)stack	(k)arrow box	(l)tree
A B	A B	A B	A B	A B	A B C

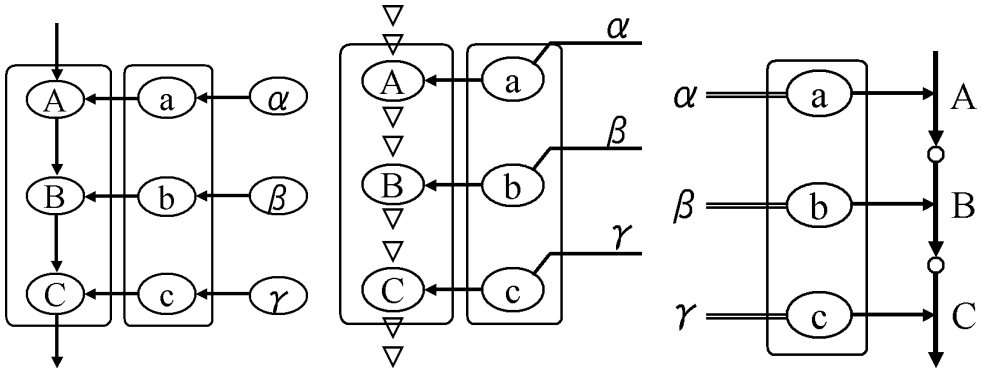
Figure 3: Examples of widely used diagrammatic expressions

type of relation	1st candidate	2nd candidate	3rd. candidate	4th candidate
`order & time`	(d)	(f)	(e)	(b)
`order & set`	(g)	(h)	(d)	(b)
`equiv & time`	(c)	(b)	none	none
`mod & term`	(d)	(h)	(c)	none

Figure 4: Examples of the ordered correspondence

3 Diagrammatic Expressions with the Order of Priority

There are various implicit rules and customs in drawing diagrams. A primitive in a diagram expresses a notion, or a combination of notions. A diagrammatic structure, such as that of alignment, often represents the relations among them. For example, area inclusion shows hierarchy or subordination. An arrow shows direction, order, or modification. A row of primitive diagrams shows flow, order, or a weak relation. Although those linguistic functions are not strict, the user can easily and quickly grasp the outline when the usage of the diagrammatic expressions is consistent with the semantics of a text.



(a) A diagram by our previous research.

(b) A diagram by our new system.

(c) A diagram generated according to some options.

Figure 5: Sample diagrams for explaining a process flow. A-C are the steps, a-c are the inputs for the steps, and $\alpha - \gamma$ are the explanations for a-c.

In our previous research, we considered a few diagrammatic expressions for which translation rules were defined as shown in Fig.2. The simple correspondence is used for expressing relations as follows: While the pattern of network diagram is always used for order, equivalence, and modification, the pattern of region diagram is used for `{order & set}`. Although a few patterns of diagrammatic expressions may correctly represent semantic structures, generated diagrams are not always comprehensible. Since the same diagrammatic expression is often used for two or more different types of semantic structure in a text, such as "causality" and "modification", we might feel difficulties in discriminating between them. This reduces the advantages of utilizing diagrams.

To cope with this problem, we introduce a more flexible translation by many-to-many correspondence between semantic structures in a text and diagrammatic expressions. Figure 3 shows examples of widely used diagrammatic expressions. By preparing a variety of diagrammatic expressions with the order of priority, the system can flexibly choose appropriate expressions. An example of the ordered preference is shown in Fig. 4. For example, "chronological order", which is `{order&time}`, can be represented by the pattern of arrow link (d). When this expression pattern is already used for representing other relations, another pattern (f) is used in order to show the differences. This mechanism has the following advantages:

- By representing different things in different ways, generated diagrams can be comprehensible and less confusing.
- As the focus of interest or the important portion varies from situation to situation, it may be emphasized by choosing the most prominent expression from among several candidates.
- The system may adjust diagrammatic description according to the user's intention or preference.

We first devised the default preference by examining the diagrammatic expressions taken from various articles and books [3]. The actual pattern selection is based on the default preference if the user gives no other options. Thus, pattern (d) is the first candidate for representing `{order&time}`, and the next candidate is pattern (f).

[3]Currently, we are investigating this preference by the subjective evaluations of more than 20 users. The results will be reported in the near future.

```
...
First, <n id="X3">attach <n id="B2">tires</n></n> to <n id="B1">chassis</n>
And then, <n id="A3">attach <n id="B3">the body</n></n>
...
<relation structure="order" semantics="inout">
 <n role="upper" nref="#B2" /> <n role="obj" nref="#X3" /> </relation>
<relation structure="order" semantics="inout">
 <n role="upper" nref="#B3" /> <n role="obj" nref="#A3" /> </relation>
<relation structure="order" semantics="process">
 <n role="upper" nref="#X3" /> <n role="lower" nref="#A3" /> </relation>
...
```

Figure 7: A portion of tagged text: assembling a toy car.

Figure 5 shows sample diagrams for explaining a process flow that requires some inputs. Figure 5(a) is that is generated by using our previous rules. By using other patterns of diagrammatic expressions, we can obtain Figure 5(b). This diagram is more comprehensible, and we can easily grasp the semantic structures. By giving some options, we obtain Figure 5(c), in which some portions are emphasized and some other portions are suppressed.

4 Experiments

An overview of our system is shown in Fig. 6. The system takes a tagged text with XML format as input, parses the text, extracts semantic descriptions from it, and then translates those descriptions into a diagram. After generating a diagram, the system serves as a diagram editor, which helps the user to modify the generated diagram according to the diagram's purposes or the user's intentions.

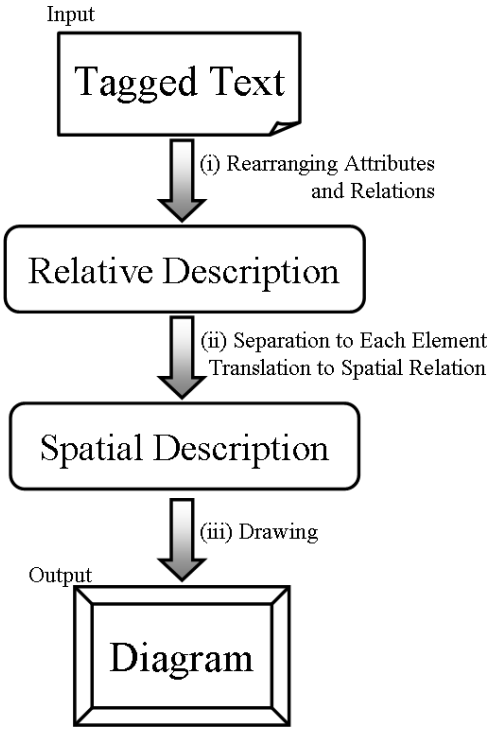
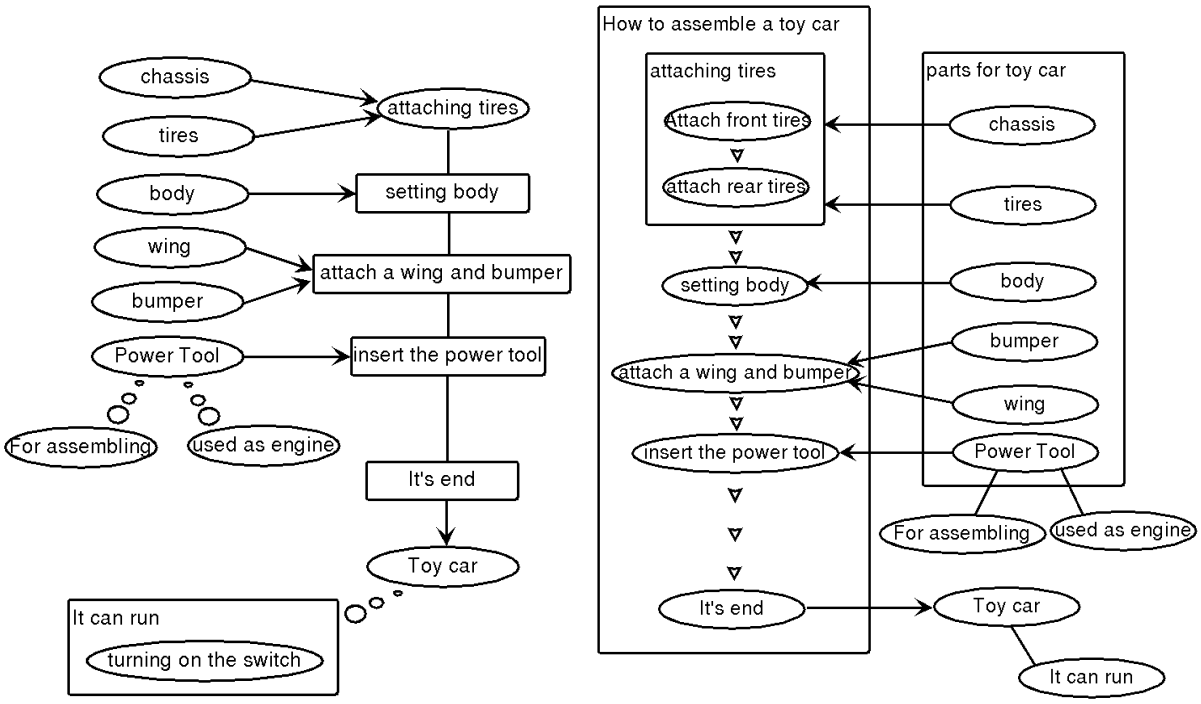


Figure 6: Overview of our system

Figure 7 shows a tagged text for a system input, and Figure 8(a) shows a generated diagram in which several diagrammatic expressions are effectively used in the same diagram. The left half of the diagram represents the process flow of assembling a toy car, and the right half represents the parts required for the processes. The enclosure represents `{order&set}`, and the steps of assembling parts and the required parts are visually grouped by placing them in boxes.

While the above example would be satisfactory in many situations, we can obtain different diagrams by giving some additional options. If we want to focus on the parts of a toy car, we can emphasize them by the following modifications: Using the enclosure pattern only for the parts; omitting other `{order&set}` type relations; not using patterns having the same shapes of diagrammatic elements. Thus, the diagrammatic representation shown in Fig.8(b) is obtained by choosing different patterns from those in the previous example.



(a) Example of a generated diagram for a tagged text in Fig.7

(b) Example of another diagram generated for the same text: Processes are show by the directions of the arrow-heads, and input-output are shown by the arrow connection

Figure 8: Generated diagram

5 Conclusion

We proposed a novel scheme for generating diagrams that effectively represent the semantic structures of tagged texts. First we presented the basic framework of the translation to diagrammatic representations and the tagset for marking up texts. Next, we proposed the use of many-to-many correspondence between semantic structures in texts and diagrammatic expressions. We showed that our system generates comprehensible explanations for a text when the user chooses appropriate diagrammatic expressions.

Although the system works as shown in the experiments, we have many topics to tackle. There are many useful diagrammatic expressions that are not yet implemented in our system. The preference settings among expressions are still ongoing in our research. Better arrangement algorithms for generating diagrams are necessary in order to make it easier for the user to modify obtained diagrams.

6 References

[1] David Harel. On visual formalism. *Communications of ACM*, 31(5):514–530, 1988.

[2] Tomihisa Kamada and Satoru Kawai. A general framework for visualizing abstract objects and relations. *ACM Trans. on Graphics*, 10(1):1–39, 1 1991.

[3] Corey Kosak, Joe Marks, and Stuart Shiebar. Automating the layout of network diagrams with specified visual organization. *IEEE Trans. on Systems, Man, and Cybernetics*, 24(3):440–454, 1994.

[4] M. Murayama, Y. Nakamura, and Y. Ohta. Diagram generation from tagged texts toward document navigation. *Proc. ICME*, 2001,(http://www.image.esys.tsukuba.ac.jp/members/murayama/work/ICME2001_murayama.pdf).