A graph layout and a multi-focus perspective display in the flowgraph editor for the legal articles

Koji Miyagi Motoki Miura Jiro Tanaka

Institute of Information Sciences and Electronics
University of Tsukuba
Tennoudai 1-1-1, Tsukuba-shi, Ibaraki 305, Japan.
+81 298 53 5165
{miyagi,miuramo,jiro}@softlab.is.tsukuba.ac.jp

ABSTRACT

The legal articles are difficult to understand for a non-specialist. We think that understanding becomes easier if the legal articles are expressed in the form of the flowgraph. Since legal articles contain lots of information, the flowgraph size becomes large when this information is expressed in the form of the flowgraph. The display size and the resolution of the computers are limited. It is hard to handle such information in the computer display. Therefore, we need techniques which express the flowgraph in a compact manner.

In this paper, we propose a **graph layout technique** and a **multi-focus perspective display method** for the flowgraph editor.

INTRODUCTION

Generally speaking, a legal article has the structure of "legal necessary conditions → legal effect"[6]. It means that the law is effective if its necessary conditions are satisfied. This structure can be expressed easily by the flowgraph, where "legal necessary conditions" node and "legal effect" node are connected by "yes edge". The flow of the conditions, i.e. time progress, is very important. The flowgraph can be understood easily as it shows the course of the conditions with the arrow.

Editing becomes a hard work if a user edits a flowgraph by manual operations. Therefore, a computer must provide us some assistance. In this paper, we propose a graph layout technique and a multi-focus perspective display method for the flowgraph editor for such purpose.

Here we consider a flowgraph editor which handles the United Nations Convention for the International Sale of Goods (CISG)[3]. CISG has been adopted by about 50 nations included the USA, Germany, UK, Russia, China, Australia, etc. It is becoming a worldwide legal standard for international commercial transactions. The text of the CISG has been written in many languages, and its provisions are being applied in many different judicial systems[4].

Graph layout is a technique to rearrange a graph automatically and display the result. The technique helps a user to understand the complex graph. We need to rearrange it, following to the characteristics of the drawing-object, to the form which is easier to understand. Though it varies in the drawing-object, generally it must be rearranged so that nodes and edges will not overlap and all nodes are adjusted in a right manner.

The various drawing-objects, which are diagrams, texts, and so on, must be mapped by a mapping method and displayed on the display. It assists a user to understand the logical meaning of the flowgraph promptly by adopting a suitable display method.

ALGORITHM FOR THE FLOWGRAPH LAYOUT

The concept of a flowgraph is very general. The history of a flowgraph goes back to the paper[7] in 1947. There are many commercially available flowgraph editors. However, most of flowgraph editors like RFflow[8] has no automatic layout facilities.

We would like to equip the automatic layout algorithm to our flowgraph editor. In editing the flowgraph, we want to check the consistency of the flowgraph, in a certain extent.

The example of the flowgraph

An example of the flowgraph is shown in Figure 1. This flowgraph shows the validity of the contract. This example includes three round-shaped terminal nodes and seven rectangular-shaped nodes. The largest node which

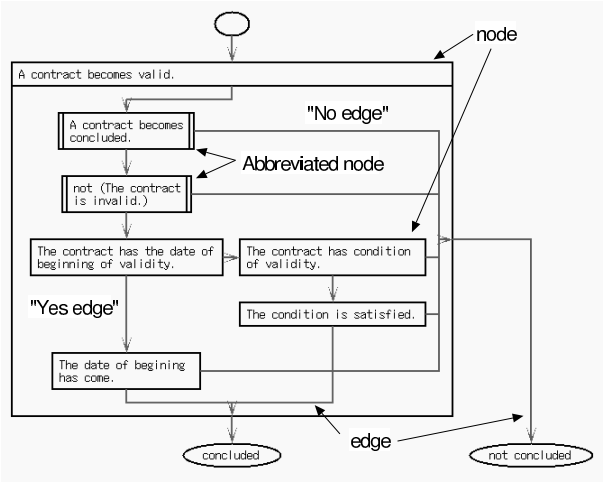


Figure 1: An example of flowgraph editor

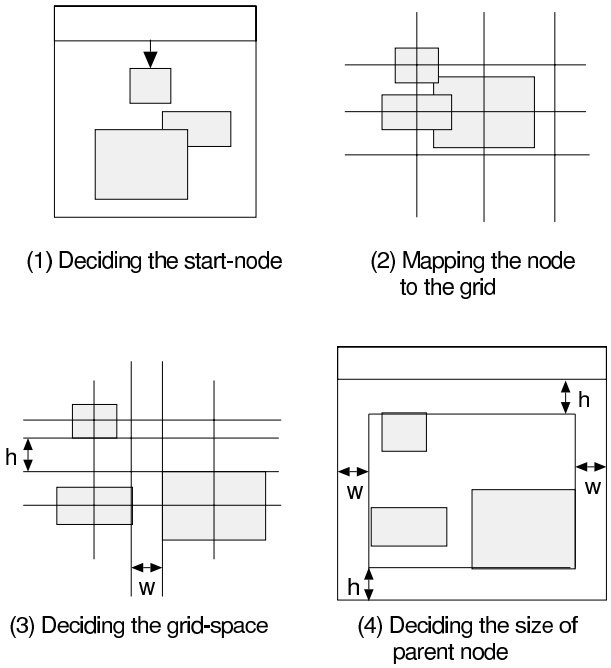


Figure 2: How to decide the size of nodes

is labeled "a contract becomes valid" has two abbreviated nodes and four nodes. The abbreviated nodes are shown by double-lined rectangles. Each abbreviated node includes some child nodes which are not displayed in this figure.

Several legal professors like to use such flowgraphs to show the legal articles. One of the typical characteristics common to all the flowgraphs is that the flow of the condition is regarded as an important factor. Especially, the nodes which connected with "yes edge" are arranged as a vertically straight mannar. The position of nodes which are connected with "no edge" depends on how the ones draw. There are two ways for drawing "no edge" which is preferred by legal professors. We call the way which draws "no edge" with a straight line as "Kagayama method," and the way which draws one with a turning line as "Yoshino method." Kagayama method locates the nodes which are connected to "no edge" to the right. Yoshino method locates the node to the right down. Yoshino method requires the larger area than Kagayama method to draw the same flowgraph. We have implemented both methods to make the flowgraph more flexible.

The layout of the flowgraph

The target flowgraph has the following features.

- The structure of the flowgraph is hierarchical. Some parent nodes have their child nodes inside of them.
- The size of a node is determined by the textural information which should be displayed and by the sizes of their child nodes.
- One node at most has two edges ("yes edge" and "no edge"). The "yes edge" stems from the bottom of the node, and the "no edge" from the right hand side of the node.
- The lines of the edges consist from horizontal and vertical lines only.

In the target flowgraph, the nodes can be nested to the arbitrary level. The child node connection is closed in the parent node. Therefore we can consider a process of node positioning as a localized problem.

The local positioning process is recursively carried out as follows (see Figure2): (1) deciding the start-node, (2) mapping the node to the grid, (3) deciding the grid-space, and (4) deciding the size of parent-node. To decide the start node, we look for an edge from parent to child. If the edge does not exist, we regard the node which does not have the incoming edge as a start node. If many start nodes are found, the mapping process is performed one by one. The mapping process will be described in the next section. After the mapping, each node belongs to a certain grid. We can assume that the size of the nodes has already been defined, since local positioning processes are performed in a bottom-up manner. To prohibit each node from overlapping, each grid-space is adjusted to the size of the largest node on the grid. The overall size of the parent node can be calculated by adding up the all grid-spaces.

After the local position of all node is determined, the global position of each node can be calculated. The global position of the child node is the sum of parent's

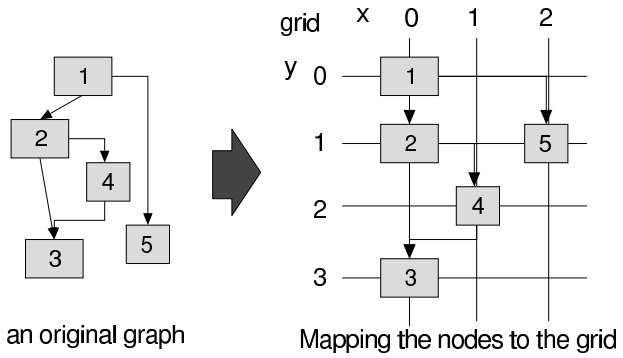


Figure 3: Mapping the node to the grid

global position and the child's local position. The process is executed from the top level of the node.

When all node positions are determined, the edges can be drawn. The "yes edge" is wired from the center of the bottom-line to the center of the top-line of the destination node. The "no edge" is wired from the center of the sideline to the center of the top-line (Yoshino method) or the left-line (Kagayama method).

The node mapping process

In this section, we describe the node mapping process. We define an orthogonal grid in each parent-node, and the child-nodes are assigned to the grid (Figure 3). The node assignment algorithm which is expressed in Java-like notation is shown in Figure4.

Figure4 defines class `Edge`, class `Node`, and class `Layout`. Procedure `mapping()` is defined in class `Layout`. In the previous section, we have decided the start-node. The procedure `mapping()` is called

```
mapping(start_node, 0, 0);
```

initially. So the start-node is assigned the grid(0,0). After that, we search the neighbor node which is connected by "yes edge" and assume the node as a start-node. This operation is continued while the node has the "yes edge." If the node does not have "yes edge," the nodes which are linked the "no edge" will be mapped. The grid position of node which is linked by "no edge" is depend on the way to draw the edge (see Figure5). We search first the "yes edge" rather than the "no edge" in each node, because the "yes edge" should be drawn as vertical straight mannar.

DISPLAY METHOD

The screen size and the resolution are the problems when a diagram is displayed on a computer. At present, because there is a limit on the screen size and the resolution

```
class Edge {
  int type;    // YES or NO
  Node d_node; // destination node
  int turning; //  Yoshino = 1 , Kagayama = 0
}

class Node {
  Edge yes_edge, no_edge; // link information
  Node parent_node;
  Node[] child_nodes;
  Size node_size;
  int grid_x, grid_y; // grid-point of this node
}

class Layout {
  static int indent;      // static variable

  void deciding_start_node(Node node){ ... }

  void mapping(Node node, int gx, int gy){
    Edge yes = node.yes_edge;
    Edge no = node.no_edge;

    node.grid_x = gx;   // assignment
    node.grid_y = gy;
    if (indent < gx) indent = gx;  // max of gx

    if (yes != null)
       mapping (yes.d_node, gx, gy+1);
    if (no != null)
       if (gx <= indent)
           mapping (no.d_node, indent+1, gy+no.turning);
         else
           mapping (no.d_node, gx+1, gy+no.turning);
  }
}
```

Figure 4: The node assignment algorithm

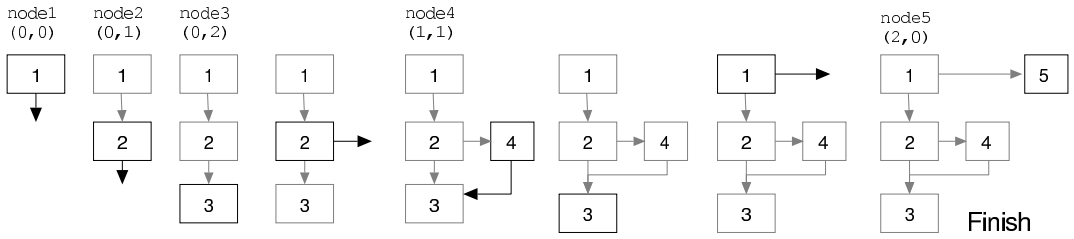
of a computer, a large figure cannot be looked over on one screen.

Requirements for the display method

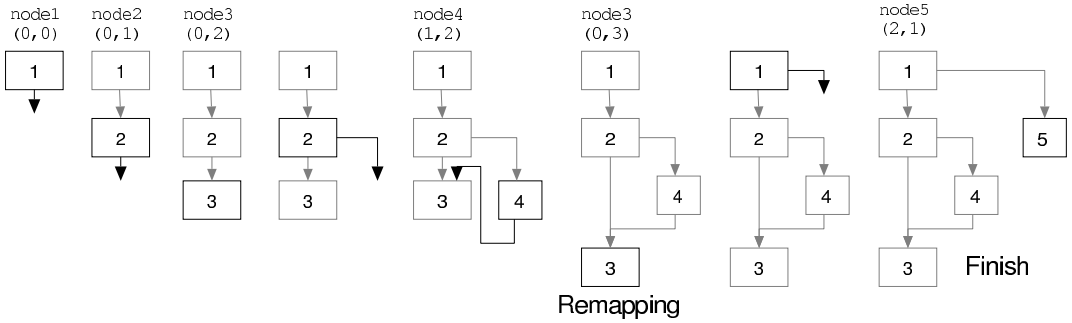
Five requirements have been proposed for the display method in [2], which are: detaility, totality, simultaneity, singleness and the fitness of the mapping. A flowgraph editor should satisfy these five requirements.

Generally, a display method is characterized by the nature of the mapping from the original figure to the display. There are various display methods, such as detailed method, multi-window method, screen switching method and multi-focus perspective display method[2]. However these display methods do not satisfy the above five requirements, except for multi-focus perspective display method.

The **multi-focus perspective display** keeps the size of the whole figure unchangeable and still expands the focused nodes. The multi-focus perspective display method is further classified into fisheye display method, orthogonal fisheye display method and biform display



(a) Kagayama method



(b) Yoshino method

Figure 5: Yoshino method and Kagayama method

method. The degree of requirement satisfaction is high in multi-focus perspective display methods.

Fisheye display method and orthogonal fisheye display method have a tendency that the original form is not kept. The tendency becomes strong as the size of the object becomes large. A distant part from the viewpoint in the infinity is reduced by mapping the area from infinite to finite. In reality, the end of the screen is reduced too much and cannot be recognized.

Biform display method was invented to get over the faults of other multi-focus perspective display methods. "Biform display method" magnifies the viewpoint's areas uniformly and reduces other area uniformly. Biform display method keeps the contour-shape and the circumference-part is never reduced to the infinity. This display method maps the rectangle-area to the rectangle-area.

We pay attention to biform display method. But, even if a flowgraph is displayed by the method, the size of the flowgraph which a computer can display, while satisfying the five requirements, is limited. Under the present condition, the size is not enough to handle the flowgraph for an expert. Therfore we think that abbreviation display facility is necessary. By using this facility, we can abbreviate the arbitrary part of the figure to use the limited screen size effectively. Therefore we have implemented abbreviation-type biform display method by adding the abbreviation display facility to the biform display method.

Biform display method

Figure 6 shows the biform display method. A user can choose multiple nodes which he wants to be magnified as viewpoints and designate the magnifying power of the viewpoint-nodes. In the result, the figure is mapped transformed as is shown in Figure 6. Biform display method is performed as follows:

Step (1) : Dividing the areas

An original figure consists of nodes and edges. We pay attentions only to the nodes. We designate viewpoint rectangles first. The upper-left and the bottom-right coordinates of the viewpoint rectangles are looked for. The values of a1, a2, a3, ..., b0, b1, b2, ..., c1, c2, c3, ..., d0, d1, d2, ... are calculated in this step.

Step (2) : Unifying the areas

When areas overlap, they must be unified and handled as one area as (4) of Figure 6. The algorithm of the area

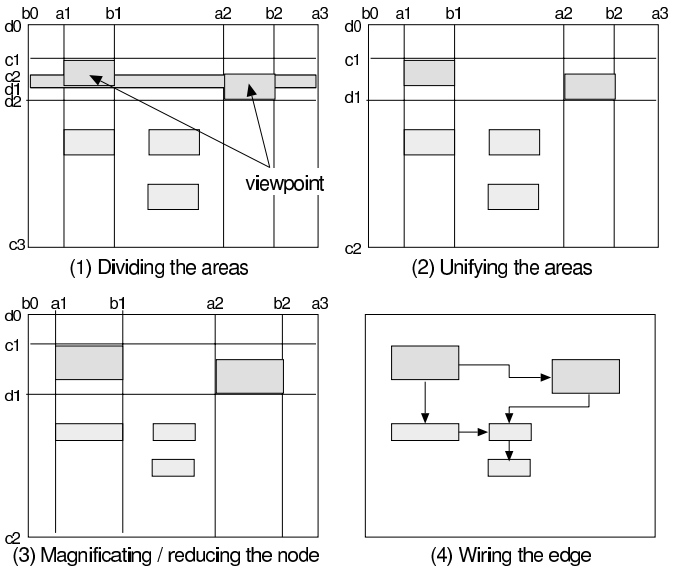


Figure 6: Biform display method

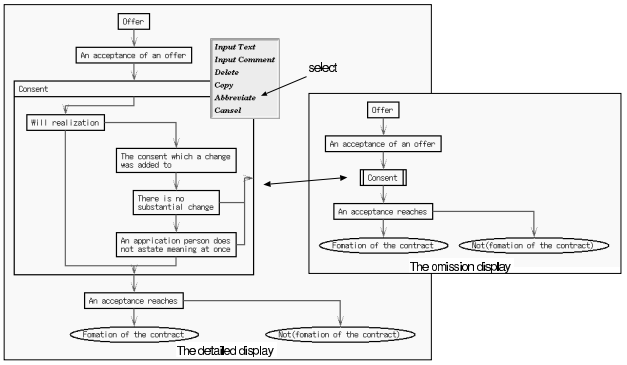


Figure 7: Abbreviation display and Detailed display

concatenation is shown in Formulae (1) and (2).

$$if\ b_i \le a_{i+1}\ (i = 1, \cdots, m)$$

$$\begin{cases} b_j & = & b_{j+1} & (j = i, \cdots, m-1) \\ a_{j+1} & = & a_{j+2} & (j = i, \cdots, m-1) \\ m & = & m-1 & \\ i & = & i-1 & \end{cases} \tag{1}$$

$$if\ d_k \le c_{k+1}\ (k = 1, \cdots, m)$$

$$\begin{cases} d_l & = & d_{l+1} & (l = k, \cdots, m-1) \\ c_{l+1} & = & c_{l+2} & (l = k, \cdots, m-1) \\ m & = & m-1 & \\ k & = & k-1 & \end{cases} \tag{2}$$

Step (3) : Magnifying / reducing the node

The reduction rate of the reduction area must be calculated. A reduction rate is calculated by Formulae (3) and (4). λ and μ are the reduction rates of the x-axis and the y-axis direction of the reduction area. σ shows the magnifying power of the viewpoint-area.

$$\lambda = \frac{(a_{m+1} - b_0) - \sigma\Sigma_{k=1}^{m}(b_k - a_k)}{\Sigma_{k=1}^{m+1}(a_k - b_{k-1})} \tag{3}$$

$$\mu = \frac{(c_{n+1} - d_0) - \sigma\Sigma_{l=1}^{n}(d_l - c_l)}{\Sigma_{l=1}^{n+1}(c_l - d_{l-1})} \tag{4}$$

New coordinates are calculated for the points at the upper-left and the bottom-right of the nodes. Coordinates are calculated by Formulae (**??**) and (6).

$$f(x) = \begin{cases} b_0 + \lambda(x - b_i) + \sigma\Sigma_{k=1}^{i}(b_k - a_k) + \lambda\Sigma_{k=1}^{i}(a_k - b_{k-1}) \\ \quad if\ b_i < x < a_{i+1}\ (i = 0, 1, \cdots, m) \\ b_0 + \sigma(x - a_i) + \sigma\Sigma_{k=1}^{i-1}(b_k - a_k) + \lambda\Sigma_{k=1}^{i}(a_k - b_{k-1}) \\ \quad if\ a_i < x < b_i\ (i = 1, 2, \cdots, m) \end{cases} \tag{5}$$

$$g(y) = \begin{cases} d_0 + \mu(y - d_j) + \sigma\Sigma_{l=1}^{j}(d_l - c_l) + \mu\Sigma_{l=1}^{j}(c_l - d_{l-1}) \\ \quad if\ d_j < y < c_{j+1}\ (j = 0, 1, \cdots, n) \\ d_0 + \sigma(y - c_j) + \sigma\Sigma_{l=1}^{j-1}(d_l - c_l) + \mu\Sigma_{l=1}^{j}(c_l - d_{l-1}) \\ \quad if\ c_j < y < d_j\ (j = 1, 2, \cdots, n) \end{cases} \tag{6}$$

Step (4) : Wiring the edge

An edge is wired to connect the newly arranged nodes.

Abbreviation-type biform display method

We propose "abbreviation-type biform display method" to add abbreviation display facility. When the size of the figure becomes large, it is necessary to abbreviate the unnecessary part. As for abbreviation display, a user can specify the arbitrary node for the abbreviation. This method deletes the child node of the abbreviation node. The edges which connect the child nodes are also deleted. The abbreviation-node are changed to the abbreviated icon. It was invented while referring to fisheye of the abbreviation type[5].

THE SNAPSHOTS OF THE EXECUTION

Figure 8 is the screen outlook of the flowgraph editor. This editor is composed of Menu-bar, Node-box, Edge-box and Canvas. The flowgraph of the legal articles can be made by using this flowgraph editor. The nodes are expressed by rectangles and show the legal necessary condition or the legal effect. The edges show relations between the legal necessary condition and the legal effect or the relations among the legal necessary conditions.

Forming the node

A node is formed by choosing an appropriate figure from the node-box and clicking the left button of the mouse putting the mouse cursor in the canvas(Figure 9). Then

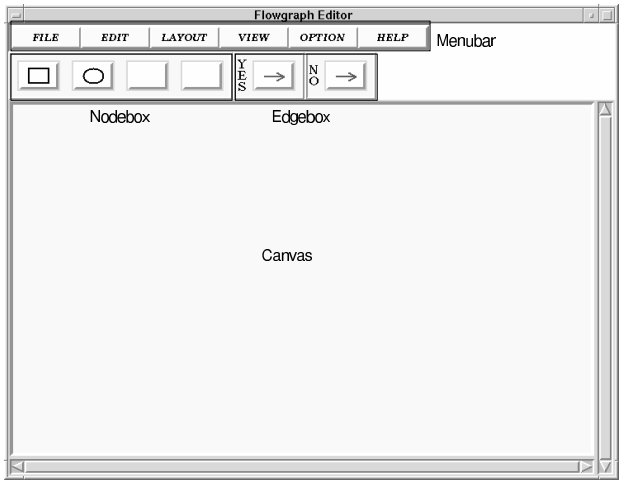


Figure 8: The screen outlook of the flowgraph editor

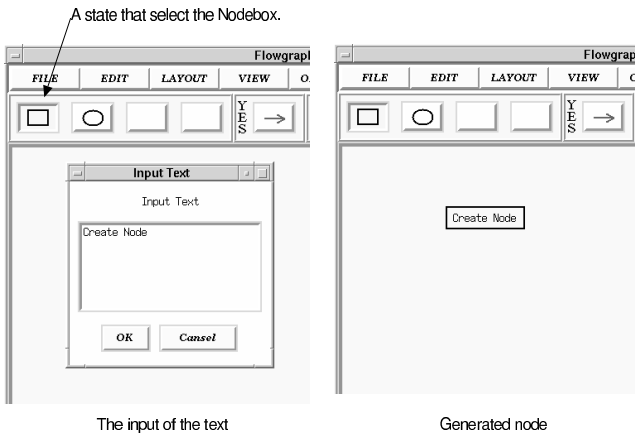


Figure 9: Forming the node

a dialog box appears. A user fills in the legal necessary condition element, the legal effect with a keyboard and clicks OK button.

Wiring the edge

An edge can be wired by dragging with the right button of the mouse from the start-node to the end-node of the edge. The kind of the edge can be chosen from the edge box.

Layout

A layout is carried out automatically when a user choose "layout" from the menu-bar or the following cases.

- When nodes are overlapped.
- When abbreviation or detailed display is specified.

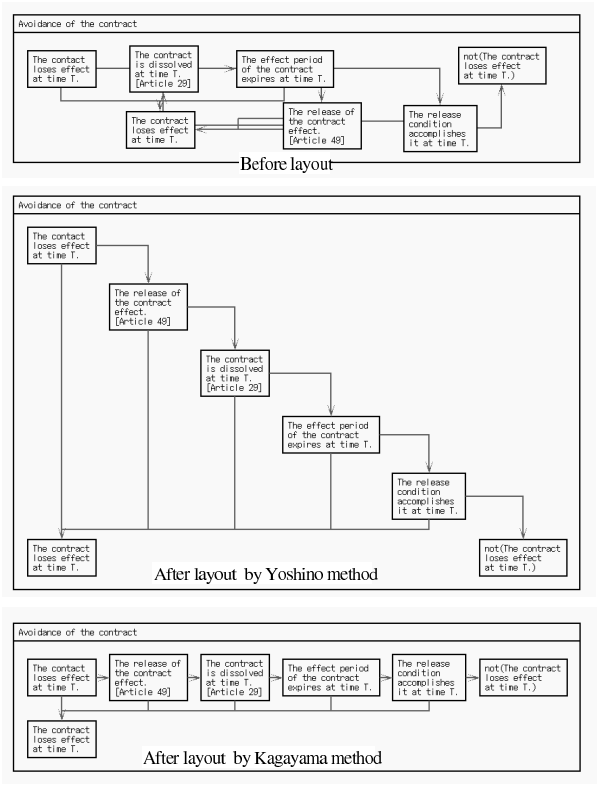


Figure 10: The layout of the flowgraph editor

- When child-node is taken out from the parent-node.

This system has methods for the layout of the flowgraph. Their methods of the layout are "Yoshino method" and "Kagayama method". Figure 10 shows a state of the layout by their methods.

Abbreviated display, detailed display

A node-menu is displayed by double-clicking on the node. Abbreviation or details is carried out by choosing "Abbreviation" or "Detail" from the node-menu.

Biform display

A user chooses "Biform Display" from the node-menu, and follows the instructions of the dialog-box, and designates a viewpoint. Next, the user manipulates a scale, and designates a magnifying power. Biform display is performed according to the change in the magnifying power in real time. When the "OK" button is pressed the text is written in the figure. Figure 11 shows a state of biform display.

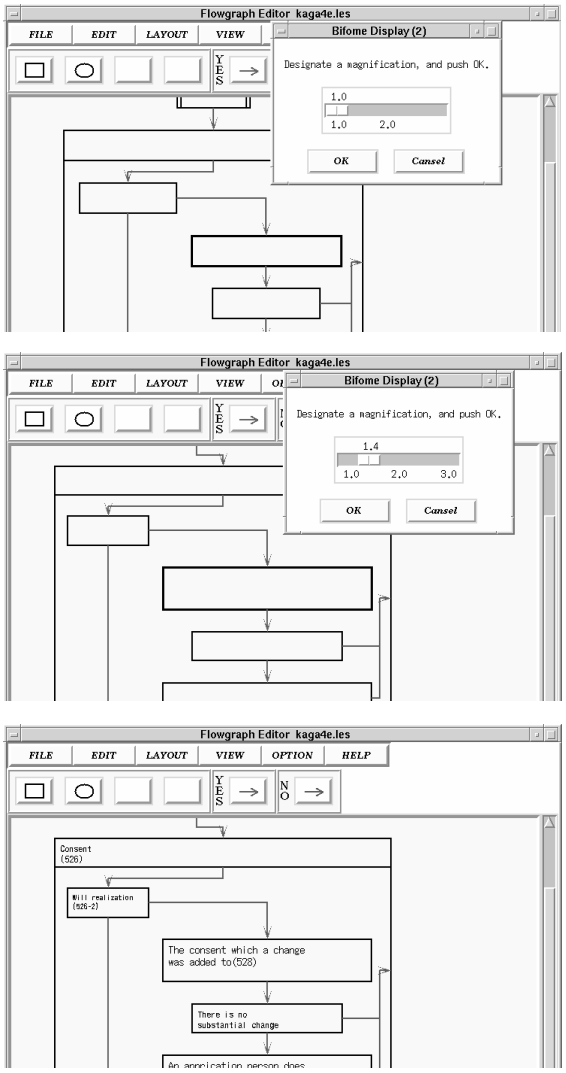


Figure 11: Biform display method

IMPLEMENTATION OF THE FLOWGRAPH EDITOR

We used Tcl/Tk[1] to implement the flowgraph editor. In implementing the editor, we have designed the intermediate code for the flowgraph editor.

The intermediate code format

The format of the intermediate code for "node" and "edge" is as follows.

< *Node* > ID, text, comment, color, figure, absolute coordinates, size, relative coordinates, grids, parent-node-ID, child-node-IDs, edge-IDs, abbreviation/detailed

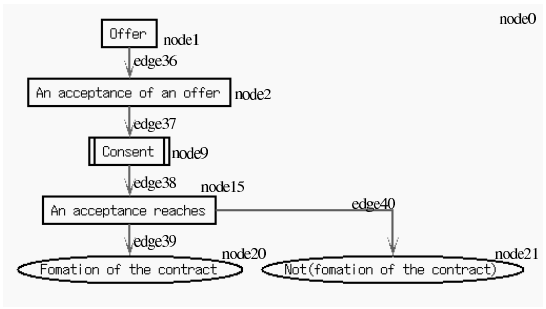


Figure 12: The graph of the intermediate code

```
Node:
0 {} {} honeydew s1 {7 7} {706 586 0} {7 7} {} {} {1 2 9 15 20 21}
{} no
1 Offer {} black s1 {200 27} {52 26 26} {193 20} {1 1} 0 {} 36 no
2 {An acceptance of an offer} {} black s1 {130 83} {192 26 26} {123
76} {1 2} 0 {36 37} no
9 Consent {} black s1 {27 139} {398 322 26} {20 132} {1 3} 0 {10 11
12 13} {37 38 12 13 24 41 42} abb
10 {Will realization} {} black s1 {48 195} {129 26 26} {21 30} {1 1}
9 {} {12 13 14} no
11 {The consent which a change was added to} {} black s1 {197 251}
{199 40 40} {170 86} {2 2} 9 {} {14 15} no
12 {There is no substantial change} {} black s1 {225 321} {143 40 40}
{198 156} {2 3} 9 {} {15 16 41} no
13 {An application person does not state meaning at once} {} black
s1 {197 391} {199 40 40} {170 226} {2 4} 9 {} {16 24 42} no
15 {An acceptance reaches} {} black s1 {144 491} {164 26 26} {137
484} {1 4} 0 {} {38 39 40} no
20 { Formation of the contract } {} black c9 {120 547} {213 26 26}
{113 540} {1 5} 0 {} 39 no
21 { Not(formation of the contract) } {} black c9 {445 547} {248 26
26} {438 540} {2 5} 0 {} 40 no

Edge:
12 {} {} blue e4 9 10 y {226 165 226 180 112 180 112 195} -15
13 {} {} blue e4 10 9 y {112 221 112 446 226 446 226 461} 15
14 {} {} red e4 10 11 n {177 208 296 208 296 251} {}
15 {} {} blue e4 11 12 y {296 291 296 306 296 306 296 321} {}
16 {} {} blue e4 12 13 y {296 361 296 376 296 376 296 391} {}
24 {} {} blue e4 13 9 y {296 431 296 446 226 446 226 461} 15
36 {} {} blue e4 1 2 y {226 53 226 68 226 68 226 83} {}
37 {} {} blue e4 2 9 y {226 109 226 124 226 124 226 139} {}
38 {} {} blue e4 9 15 y {226 461 226 476 226 476 226 491} {}
39 {} {} blue e4 15 20 y {226 517 226 532 226 532 226 547} {}
40 {} {} red e4 15 21 n {308 504 569 504 569 547} {}
41 {} {} red e4 12 9 n {368 341 410 341 410 300 425 300} 15
42 {} {} red e4 13 9 n {396 411 410 411 410 300 425 300} 15
```

Figure 13: Intermediate code

< *Edge* > ID, text, comment, color, figure, start-node-ID, end-node-ID, yes or no, list of the coordinates, constraint

In implementing this flowgraph editor, we had to realize the graph drawing of the class structure, the automatic graph layout and the abbreviation-type biform display method. We have designed the intermediate code which realizes these facilities. To draw the class structure, we must handle the parent and child relations. Therefore we have made the editor that can record "parent-node-ID" and "child-node-ID". In the layout algorithm of the editor, it must handle the absolute loca-

tion and the relative location. The information of the grid location is also necessary. To handle these information, the editor records "absolute coordinates", "relative coordinates" and "grids". In abbreviation-type display method, we must handle the information whether a node is abbreviated or not. We have also added the element "abbreviation / detailed". When a node is abbreviated, "abb" is contained in the element, and when not abbreviated, "no" is shown.

Example of the intermediate code

The intermediate code for Figure 12 is shown in Figure 13. The node 0 expresses the screen of the editor. We can see "abb" at the tail of the node 9. It denotes node 9 is abbreviated. The nodes (10 ~ 13) and the edges (12 ~ 24,41,42) are not displayed because the parent node 9 has been abbreviated.

CONCLUSIONS

We have implemented the flowgraph editor for the legal articles. The editor has the graph layout and the abbreviation-type biform display facilities. Considering the characteristics of the flowgraph, we have designed the new graph layout. The user can easily rearrange the figure, and understand the flowgraph. The abbreviation-type biform display has effectively been implemented. It satisfies the five requirements for the display method and has the abbreviation display facility. A user can understand the meaning of the flowgraph better than before. We are asking legal professors, such as Professor Yoshino and Professor Kagayama, for the user test. We would like to improve the editor reflecting their results.

REFERENCES

1. Jhon K. Ousterhout: Tcl and the Tk Toolkit, Addison-Wesley Publishing Company, 1994.
2. Kazuo Misue, Kozo Sugiyama : On Multi-focus Perspective Display Method of Figures for Computer Aided Diagrammatical Thinking, International Institute for Advanced Study of Social Information Science, Fujitsu Laboratories, IIAS-RR-91-4J(1991.01).
3. Kazuaki Sono, Masashi Yamate: The United Nations Convention on Contracts for the International Sale of Goods (Kokusaibaibaihou), Seirinshoin, 1993 (*in Japanese*).
4. Kervin D. Ashley and Hajime Yoshino ed.: The Fourth International Workshop on a Legal Expert System for the CISG, Legal Expert System Association, 1997.
5. Kozo Sugiyama: Automatic graph drawing methods and their applications, The Society of Instrument and Control Engineers, 1993 (*in Japanese*).
6. Hajime Yoshino: Expression method of legal knowledge, Report of research and development on legal expert system, pp. 124-138, 1994 (*in Japanese*).
7. H. H. Goldstein and von Neumann: Planning and coding problems for an electronic computing instrument, part II, in *von Neumann Collected Works*, Vol.5, McMillan, New York, pp. 80-151.
8. "RFflow Professional Version 3.0J" Simphony,1994.