Visualization with hierarchically structured trees for an explanation reasoning system

Mariko Sasakura
Okayama University

Susumu Yamasaki
Okayama University

Visualization with Hierarchically Structured Trees for an Explanation Reasoning System

Mariko Sasakura and Susumu Yamasaki
Graduate School of Natural Science and Technology,
Okayama University, Japan
sasakura@momo.it.okayama-u.ac.jp

Abstract

This paper is concerned with an application of drawing hierarchically structured trees. The tree drawing is applied to an explanation reasoning system. The reasoing is based on synthetic abduction (hypothesis) that gets a case from a rule and a result. In other words, the system searches a proper environment to get a desired result. In order that the system may be reliably related to the amount of rules which are used to get the answer, we visualize a process of reasoning to show how rules have concern with the process. Since the process of reasoning in the system makes a hierarchically structured tree, the visualization of reasoning is a drawing of a hierarchically structured tree. We propose a method of visualization that is applicable to the explanation reasoning system.

1. Introduction

As widely known, graphs are suitable to represent relations between objects. Especially they are often used in software visualizations that represent some features of programs such as static program structures or traces of executions of programs. In this paper, we apply a drawing of hierarchically structured trees to some visualization of a reasoning system.

We construct an explanation reasoning system based on logic programs. The system gives reasons or cases why the given goal (or result) holds. It is based on the abduction that gets a case from a rule and a result as in [5]. In other words, the system searches a proper environment to get a desired result.

An explanation reasoning system is significant, because it may help us to find out proper information from the large amount of data [11]. Currently, we have so much data that it is difficult to determine the information that we want. The explanation reasoing system may search information to get a desired result (or goal). For example, suppose that we have a problem in our computer, and want to know what is wrong. In this case, the explanation reasoning system will search for a cause of the problem using rules in the system.

However, how can we learn that the system is reliable or not? Although the correctness of the procedure is guaranteed, the reliability of the system is also related to the rules which are used to get what is wrong. Thus, a process of the system reaching the problem status can be an indicator of the reliability. Therefore, we propose to show the process of the reasoning as the way to judge the reliability of the answer.

A reasoning process consists of a hierarchically structured tree which is a tree whose nodes are also trees. The tree is a kind of a clustered graph [4], but according to [10], we call it a hierarchically structured tree. In this paper, we argue the drawing method of a hierarchically structured tree in order to visualize a reasoning process in the explanation reasoning system.

A tree is a popular data structure in computer science. Usually, a tree is drawn by the hierarchical approach [1]. There are other visualization methods. Treemap [12] and TennisViewer [8] are typical examples of the visualization methods which draw a tree in 2D.

A tree discussed in this paper is slightly different from the ordinary tree. Its nodes are also a tree. It is a special case of a hierarchically structured graph. A general form of a hierarchically structured graph is given as the higraph [7]. Some drawing algorithms for general use are already presented. Sugiyama et al. give the drawing algorithm to a compound graph which is a hierarchically structured graph that has two kinds of edges [13]. The clustered graph [4] is a hierarchically structured graph whose nodes are also graphs. Raitner provides a library for drawing hierarchically structured graphs [10].

We propose a new drawing method by applying the characteristic of the explanation system. A process of the explanation reasoning should be represented as a tree of which nodes are also trees. Our proposed method cannot visualize a general hierarchically structured graph, but it can vi-

Proceedings of the Eighth International Conference on Information Visualisation (IV'04)
1093-9547/04 $ 20.00 IEEE

sualize a reasoing process of the explanation reasoing system.

A process of the explanation reasoning is drawn as a combination of the hierarchical approach of graph drawing [1] and circles with inclusion. There are some previous works that draw hierarchically structured graphs with the combination. For example, Sugiyama et al. draw a whole tree as inclusion and a graph in a node using the hierarchical approach of graph drawing [13]. While we draw a whole tree as the hierarchical approach of graph drawing and a tree in a node as inclusion. In our application, since a node is a tree (not a graph), we can draw a node by inclusion. The reasons why we draw a process of the explanation reasoning in such a way are:

- The meanings of the two tree structures are different. One is caused by procedure calls, and the other is by recursions of a procedure. We would like to use different methods to represent different type of trees.
- A procedure which corresponds to node configures recursions. We think of some inclusion notion as suitable to represent recursion intuitively.

In Section 2, we introduce a procedure of the explanation reasoning. In Section 3, we explain the algorithms of drawing a process of the explanation reasoning and present examples. In Section 4, we give concluding remarks.

2. The Explanation Reasoning Procedure

2.1. Overview

In this paper, we deal with a general logic program which is a set of clauses of the form:

$$A \leftarrow L_1, \ldots, L_n \ \ (n \geq 0)$$

where A is an atom and L_i are literals. A literal is an atom (positive literal) B or a negation of an atom (negative literal) $\sim B$. The atom A is called the head, while the literal list $L_1, \ldots, L_n$ is the body. A goal is an expression of the form $\leftarrow M_1, \ldots, M_l$ $(l \geq 0)$ where M_j are literals. If $l = 0$, then the goal is said an empty clause.

The original abductive procedure, which is a basis of the explanation reasoning system, is described in [5]. The procedure is not in general sound with respect to the 2-valued stable model semantics [5]. The correctness of the procedure is guaranteed in 3-valued logic in that it is sound with respect to 3-valued stable models [2, 3, 9].

The procedure consists of two derivations: a succeeding derivation and a failing derivation. The succeeding derivation aims at a proof for the given goal by SLD-resolution and negation as failure. The failing derivation aims at no proof of the given goal set.

The behaviour of the succeeding derivation is described as follows.

- **Step 1:** In a succeeding derivation, a positive literal in a goal is replaced by the literal list which is the body of a clause whose head is just the literal. (This is caused by so called SLD-resolution.)
- **Step 2:** If there is a negative literal $\sim l$ in a goal, a failing derivation for the goal set $\{\leftarrow l\}$ is invoked to prove the success of a goal $\leftarrow\sim l$. (Note any negative literal does not become the head of a clause of a general logic program.) If the goal $\leftarrow\sim l$ has a proof for the success, it is removed from the goal.
- There is a proof for a given goal if the goal becomes an empty clause by **Steps 1** and **2**.
- A succeeding derivation from a goal succeeds if the goal is reduced to the empty clause such that there is a proof for a given goal.

The behaviour of the failing derivation is described as follows.

- A failing derivation for the goal set $\{\leftarrow l\}$ holds if there is no proof of a goal $\{\leftarrow l\}$. In a failing derivation, we should check all cases of the goals that may have the proof of the goal $\leftarrow l$. Therefore, a failing derivation is invoked for a set of goals.
- **Step 1:** A positive literal in a goal of the set of goals is replaced by the literal list which is the body of a clause whose head is just the literal. (This is a derivation by SLD-resolution.) If there are more than one clause whose heads are just the literal, we add all goals that are generated by SLD-resolutions to the set of goals.
- **Step 2:** If there is no clause whose head is a positive literal in a goal g of the set of goals, the goal g is removed from the goal set.
- **Step 3:** If there is a negative literal $\sim l$ in a goal g, a succeeding derivation for the goal $\leftarrow l$ is invoked to prove the goal g. If the goal $\leftarrow l$ has a proof, the goal g is removed from the goal set.
- There is no proof of $\{\leftarrow l\}$ if the goal set becomes an empty set by **Steps 1, 2** and **3**.

EXAMPLE 1 *If there is a propositional logic program:*

$$P_1: \begin{array}{l} A \leftarrow\sim B \\ B \leftarrow C \\ B \leftarrow\sim D \\ D \leftarrow \end{array}$$

then the procedure operates on the goal $\leftarrow A$ *for* P_1 *as follows.*

1. At first, the succeeding derivation is invoked with the goal $\leftarrow A$.

2. By SLD-resolution, the goal is reduced to a goal $\leftarrow\sim B$.
3. Since $\sim B$ is a negative literal, the failing derivation is invoked with a goal set $\{\leftarrow B\}$.
4. By the **Step 1** of the failing derivation, the goal set is transformed to a set $\{\leftarrow C, \leftarrow\sim D\}$.
5. Since there is no rule whose head is C, there is no proof of $\leftarrow C$.
6. Since $\sim D$ of the goal $\leftarrow\sim D$ is a negative literal, the succeeding derivation is invoked with a goal $\leftarrow D$.
7. By SLD-resolution, the goal $\leftarrow D$ succeeds. Hence there is no proof of the goal $\leftarrow\sim D$.
8. By 6 and 7, there is no proof of the goal set $\{\leftarrow B\}$.
9. It follows that the goal $\leftarrow\sim B$ succeeds.
10. Finally the goal $\leftarrow A$ succeeds.

2.2. A Hierarchically Structured Tree by the Explanation Reasoning Procedure

In this section, we describe the way that presents reasoning processes by the explanation reasoning procedure as a hierarchically structured tree.

The outline of the way is as follows.

- A transition of goals produced by a succeeding derivation is represented as a tree that we call a goal tree. There is a unique leaf on a goal tree by a succeeding derivation.
- A transition of goals by a failing derivation is represented as a tree that we also call a goal tree. A goal tree for a failing derivation can have several leaves.
- Derivation calls from succeeding and failing derivations make a tree that we call a derivation tree. Nodes of a derivation tree are goal trees and edges are derivation calls between succeeding and failing derivations.

A derivation tree made by the explanation reasoning procedure is a hierarchically structured tree $DT = (GT, E)$, where GT is a set of goal trees and E is a set of edges between goal trees. The depth of a node is defined as the numbers of edges from the root node. The depth of the root node is 0.

A derivation tree is constructed as follows:

- The root node is the goal tree produced by a succeeding derivation from a given goal, since the explanation reasoning procedure starts from a succeeding derivation. There is only one root node for a derivation tree.
- The nodes of the depth 1 are the goal trees produced by failing derivations that are invoked by the succeeding derivation.
- The nodes of the depth 2 are the goal trees produced by succeeding derivations that are invoked by the failing derivations.
- If the nodes of the depth n are goal trees produced by succeeding derivations, the nodes of the depth n+1 are the goal trees by failing derivations.
- If the nodes of the depth n are goal trees produced by failing derivations, the nodes of the depth n+1 are the goal trees by succeeding derivations.

This construction is derived from the procedure represented in [15].

A derivation tree has the following properties.

- The nodes of the even depth are produced by succeeding derivations.
- The nodes of the odd depth are produced by failing derivations.

These are recursively constructed because in the explanation reasoning procedure, a succeeding derivation invokes failing derivations and a failing derivation invokes succeeding derivations.

3. Drawing a Process of Explanation Reasoning

A hierarchically structured tree that is produced by the explanation reasoning system is visualized by two methods. A goal tree is drawn as inclusion and a derivation tree is drawn using the hierarchical approach to graph drawing in three-dimensions.

The aim of our visualization is to give the way to check the correctness of the reasoning performed by the explanation reasoning system. There may be two kinds of way to check the correctness: a precise way and an intuitive way. An example of a precise way is to check if all clauses of rules is proper or not. An intuitive way gives a heuristic one that a reasoning is proper. In this paper, we point out some heuristics.

1. If there are many steps in a derivation, the possibility of uses of improper rules may increase, because there are many rules concerned with the derivation.
2. If there is no candidate in a failing derivation, the possibility of incorrectness may increase because there may be cases that are not explicitly described in the rules.

We design a derivation tree to show users the whole structure, and goal trees to show users the complexity of the trees at a glance. We draw a derivation tree using the hierarchical approach to graph drawing in three-dimensions, and a goal tree using circles with inclusions.

There are already some works that draw tree using circles [14]. Our method is different from them. The idea of circles with inclusion is based on Euler/Venn diagrams [6].

3.1. Drawing a Goal Tree

A goal tree by a succeeding derivation is represented as a sequence of goals: $(G_0, \ldots, G_n)$. We draw a goal tree as inclusion of circles. The points of our method are as follows:

- G_i is represented as a circle.
- A circle for G_i includes a circle for G_{i+1}.
- The centers of G_i and G_{i+1} have the same coordinates, to use space effectively.

The drawing algorithm is constructed in the following. The circle for G_i is represented as a triplet (x_i, y_i, r_i), where (x_i, y_i) is the coordinates of the center of the circle and r_i is the radius. Then $(x_{i+1}, y_{i+1}, r_{i+1})$ for G_{i+1} is

$$\begin{aligned} x_{i+1} &= x_i \\ y_{i+1} &= y_i \\ r_{i+1} &= r_i * \alpha \end{aligned}$$

where the value α is a fixed value for $0 \leq \alpha \leq 1$. α decides a space between circles.

A goal tree caused by a failing derivation is represented as a sequence of goal sets. To draw a goal tree for a failing derivation, we use two methods: one is for a node which has an outgoing branch, the other is for a node which has plural outgoing branches. The method for the former node is the same as the method for a goal tree by a succeeding derivation. The method for the latter node is demonstrated below:

Assume that G is the goal of a node which has plural outgoing branches. $g_1, \ldots, g_m$ are nodes that are the destination of the branches. A triplet (x_0, y_0, r_0) denotes the circle for G. Then (x_j, y_j, r_j) for g_j is:

$$\begin{aligned} x_j &= x_0 + \tfrac{r_0}{2} * cos(2\pi * j/m) \\ y_j &= y_0 + \tfrac{r_0}{2} * sin(2\pi * j/m) \\ r_j &= \tfrac{r_0}{4} \end{aligned}$$

A goal tree is drawn as concentric circles in the case that there is an outgoing branch. It is on a white background for a succeeding derivation and on a gray background for a failing derivation. We easily know whether a derivation has many steps or not, by the complexity of concentric circles at a glance. We also learn a goal tree for which the derivation is made by the background colour of the tree.

3.2. Drawing a Derivation Tree

The points of a method for drawing a derivation tree are as follows:

- A derivation tree is drawn using the hierarchical approach to graph drawing in three-dimensions.
- A node is drawn as a plane in which the corresponding goal tree is drawn. We call a plane to be a *paper*.

The algorithm of calculating coordinates of a node of a derivation tree is constructed. A triplet (x_i^d, y_i^d, z_i^d) denotes the center of i-th paper of the depth d. If the paper has n (≥ 1) children, $(x_j^{d+1}, y_j^{d+1}, z_j^{d+1})$ $(1 \leq j \leq n)$ is:

(i) If $n = 1$**,** then

$$x_1^{d+1} = x_i^d, y_1^{d+1} = y_i^d, z_1^{d+1} = z_i^d + \beta$$

(ii) IF $n > 1$**,** then

$$\begin{aligned} x_j^{d+1} &= x_i^d + r * sin(2\pi * j/n) \\ y_j^{d+1} &= y_i^d + r * cos(2\pi * j/n) \\ z_j^{d+1} &= z_i^d + \beta \end{aligned}$$

The value β and r are fixed values. β decides a space between the depth d and the depth $d+1$. The value r is concerned with a space between papers which have the same parent.

3.3. Examples

We show two examples one of which is simple and the other is more complicated. We explain the detail of the visualization using the simple example.

EXAMPLE 2 *There is a propositional logic program:*

$$P_2: \begin{array}{ll} A \leftarrow B, \sim C, \sim D, \sim I, \sim L & B \leftarrow \\ C \leftarrow \sim B & D \leftarrow E \\ E \leftarrow F & F \leftarrow G \\ G \leftarrow H & H \leftarrow I \\ C \leftarrow I & C \leftarrow J \\ C \leftarrow K, L & L \leftarrow M \end{array}$$

Given $\leftarrow A$ *as a goal, the explanation reasoning system is shown in Figure1.*

Figure 2 is the root node of the tree as in Figure 1. Figure 3 is the node for the failing derivation from a goal $\leftarrow C$. The background colour of each node represents the sort of derivation: it is white for a succeeding derivation and gray for a failing derivation. A light gray circle means that it invokes a derivation. A dark gray circle means that the corresponding goal holds. As well, a black circle means that there is no proof to the corresponding goal. In this example, we know a goal $\leftarrow A$ succeeds since the innermost circle for the root node is dark gray. By this figure, we can know that the root derivation has many steps, where the failing derivation for a goal $\leftarrow C$ is complex and the failing derivations for goals $\leftarrow D, \leftarrow I, \leftarrow L$ are simple.

EXAMPLE 3 *Figure 4 shows the figure for a complicated reasoning. This reasoning is performed on a rule that is produced at random.*

3.4. Discussion

We can prove that the proposed visualization method draws all transitions of goal and a derivation tree for the explanation reasoning procedure, because there is a one-to-one relation between the explanation reasoning procedure and the drawing process. The detail of the proof is out of the scope of this paper.

The advantages of the visualization method are:

- A derivation tree shows the structure of derivation calls.
- A goal tree is represented in a node of a derivation tree. We can see how complicated each derivation is.
- We can easily distinguish a derivation tree and a goal tree because of the way of visualization : a derivation tree is drawn as the hierarchical approach of graph drawing and a goal tree is drawn by inclusions of circles.
- The result of reasoning can be found in the root node. We can easily learn whether the given goal holds or not, by looking at the root node.
- If we want to investigate the details of a derivation, we can scale up the node and check the derivation.

4. Concluding Remarks

We describe a visualization of a reasoing process for the explanation reasoning system. Since the reasoning process makes a hierarchically structured tree, the visualization is an application of the drawing hierarchically structured trees. We visualize the hierarchically structured tree as a three-dimensional graph whose node is also a tree. Each node makes a plane and we visualize a tree of the node in the plane. The tree in the plane is drawn by using inclusions. We can easily distinguish a derivation tree and a goal tree because of the way of visualization.

We think that the contribution of this paper is to display an application of hierarchically structured trees. In general, behaviours or structures of software systems can be represented as graphs. Most of current systems visualize them as simple graphs. However, hierarchically structured trees may be useful to represent more complex relations and help the users to develop systems.

References

[1] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Plentice Hall, 1999.

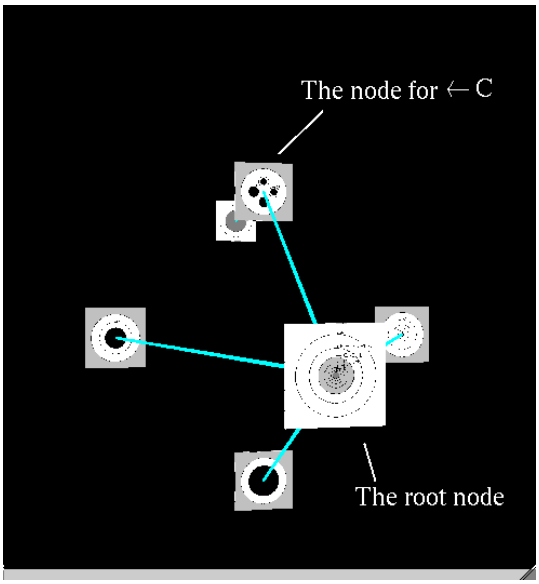

Figure 1. A derivation tree for a goal: This figure shows 4 failing derivations and 1 succeeding derivation are invoked during the reasoning process.

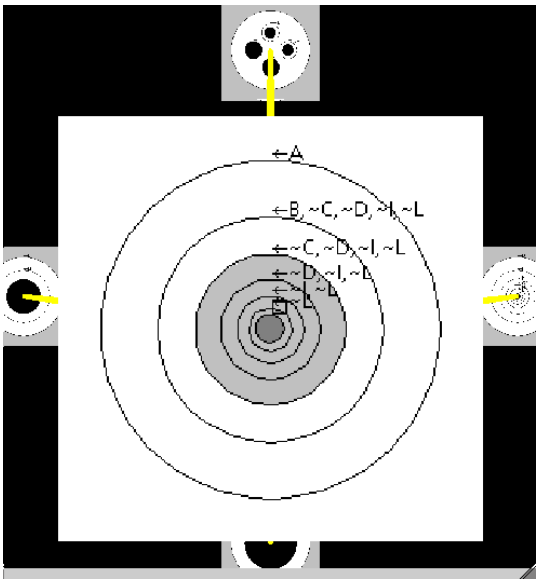

Figure 2. A goal tree of the root node: This figure shows that the goal $\leftarrow A$ holds on the program P_2.

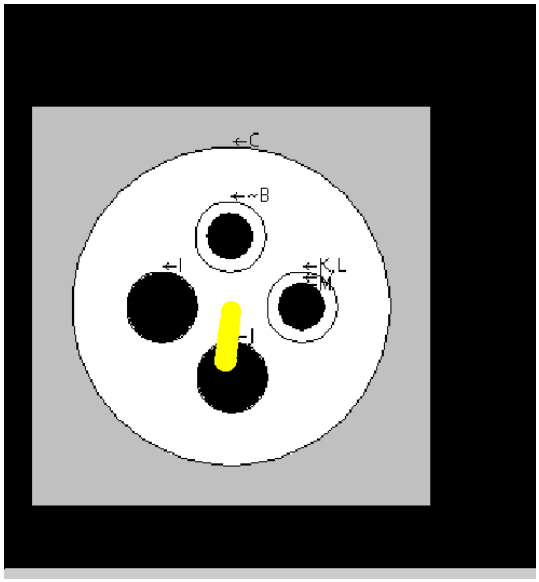

Figure 3. A goal tree for the failing derivation: This figure shows that there are four clauses in the program P_2, whose head is C. For all the clauses, $\leftarrow C$ has no proof. We can know it from the four black circles in the figure.

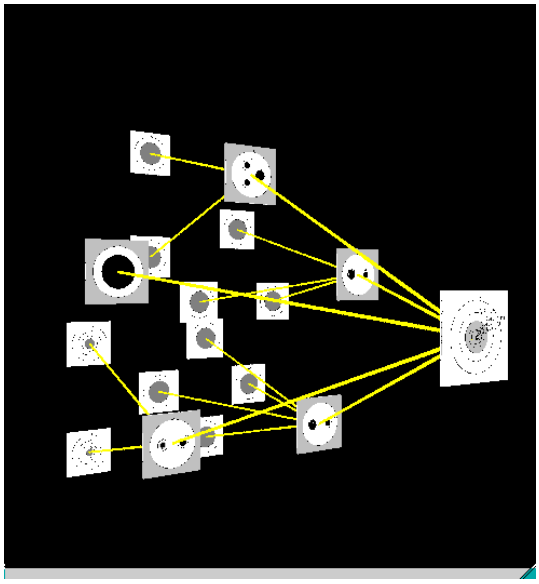

Figure 4. A derivation tree of a reasoning

[2] P. M. Dung. Negation as hypotheses: An abductive foundation for logic programming. *Proceedings of the Eighth International Conference on Logic Programming*, pages 3–17, 1991.

[3] P. M. Dung. An argumentation-theoretic foundation for logic programming. *Journal of logic programming*, 22:151–177, 1995.

[4] P. Eades and Q.-W. Feng. Multilevel visualization of clustered graphs. *GD1996, LNCS 1190*, pages 101–112, 1996.

[5] K. Eshghi and R. Kowalski. Abduction compared with negation by failure. *Proc. of 6th ICLP*, pages 234–255, 1989.

[6] E. Hammer and S.-J. Shin. Euler and the role of visualization in logic. *Languages, Logic and Computation: The 1994 Moraga Proceedings*, pages 271–286, 1994.

[7] D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988.

[8] L. Jin and D. Banks. Tennisviewer: a browser for competition trees. *IEEE Comput. Graph. Appl.*, 17(4):63–65, 1997.

[9] A. Kakas and P. Mancarella. Preferred extensions are partial stable models. *J. of Logic and Computation*, 2:719–770, 1992.

[10] M. Raitner. Hgv: A library for hierarchies, graphs, and views. *GD2002, LNCS 2528*, pages 236–243, 2002.

[11] M. Sasakura and S. Yamasaki. An explanation reasoning procedure applicable to loop transformation in compiler. *Proc. of ACM ESEC/FSE International Workshop on Intelligent Technologies for Software Engineering, WITSE 03*, pages 34–39, 2003.

[12] B. Shneiderman. Tree visualization with treemaps: A 2d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[13] S. Sugiyama and K. Misue. Visualization of structural information: Automatic drawing of compound digraphs. *IEEE Transaction on Systems, Man and Cybernetics*, 21(4):876–892, 1991.

[14] S. Teoh and M. K.-L. Rings: A technique for visualizing large hierarchies. *GD2002 LNCS 2528*, pages 268–275, 2002.

[15] S. Yamasaki and M. Sasakura. Towards distributed programming systems with visualizations based on nonmonotonic reasoning. *International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (CD-ROM) 76*, 2001.