



Syntactic Processing of Diagrams by NCE Graph Grammars

有田 友和,

富山 聖宣,

夜久 竹夫

日本大学

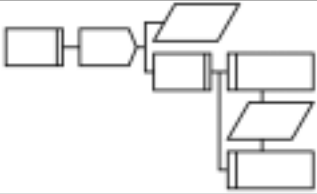
宮寺 庸造,
東京学芸大学

杉田 公夫,
東海大学

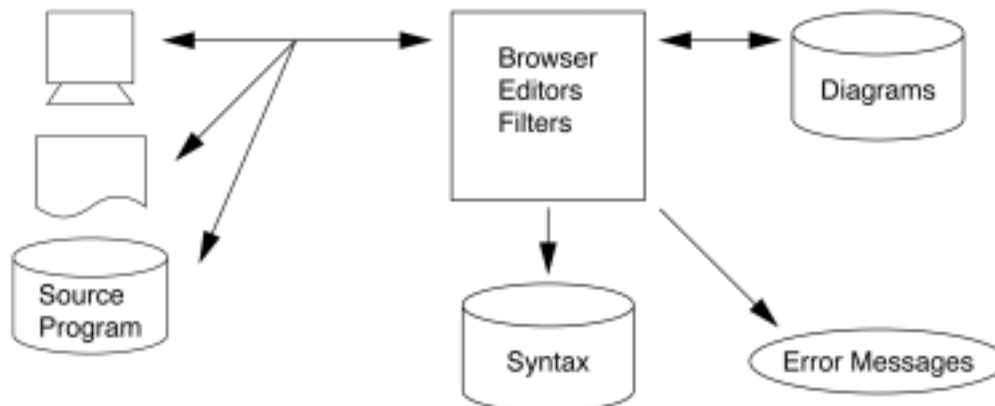
土田 賢省
東洋大学

Abstract

■ TARGET: Diagrams in Software Specification

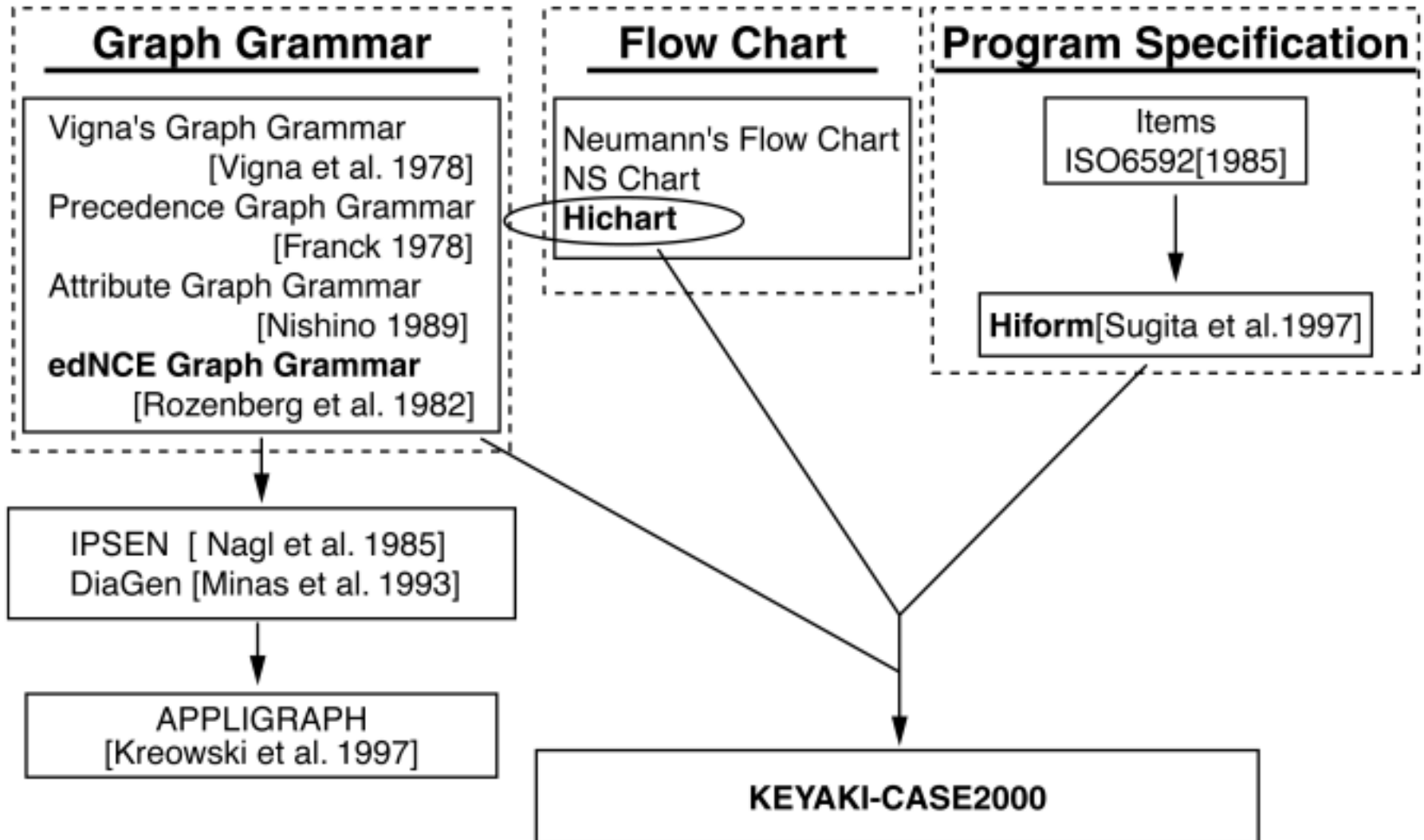
	Diagrams	Corresponding Graphs	Universal Models												
Hierarchical Diagram		Attribute Tree	Attribute NCE CFGG												
Nested Diagram	<table border="1" data-bbox="500 596 959 686"><tr><td>Program Code:</td><td rowspan="3">Program Specification</td></tr><tr><td>Program Name:</td></tr><tr><td>Library Code:</td><td>Version:</td></tr></table>	Program Code:	Program Specification	Program Name:	Library Code:	Version:	Attribute Marked Tree	Attribute NCE CFGG							
Program Code:	Program Specification														
Program Name:															
Library Code:		Version:													
Tessellation Diagram	<table border="1" data-bbox="500 739 959 829"><thead><tr><th>Name</th><th>Type</th><th>Size</th><th>G/L</th></tr></thead><tbody><tr><td>x</td><td>int</td><td>2</td><td>G</td></tr><tr><td>y</td><td>float</td><td>4</td><td>L</td></tr></tbody></table>	Name	Type	Size	G/L	x	int	2	G	y	float	4	L	Attribute Marked Tessellation Graph	Attribute NCE CSGG
Name	Type	Size	G/L												
x	int	2	G												
y	float	4	L												

■ GOAL: Syntactic Processing



1 Introduction

Background

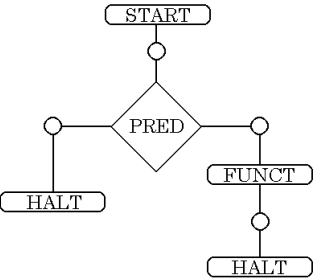
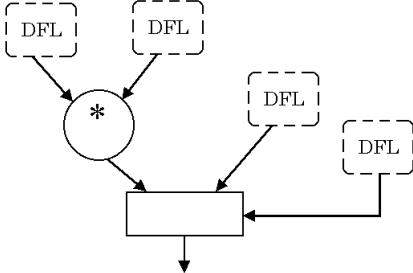
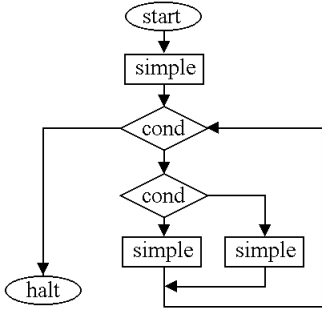




Related Works

Project Name	Program Semantics by GG	Drawing by Combinatorial Algorithm	Drawing by AGG	Syntax by Graph Grammar
IPSEN	○	—	—	○
DiaGen	—	○	—	○
KEYAKI-CASE2000	—	○	○	○

Related Works (continued)

Diagrams	Known Models	
<p data-bbox="91 287 373 339">Flowchart</p> 	<p data-bbox="795 287 1068 418">CF PLEX Grammar</p>	<p data-bbox="1348 287 1747 347">K.S.Fu (1982)</p>
	<p data-bbox="795 461 1081 592">Relational Grammar</p>	<p data-bbox="1348 461 1719 611">K.Wittenburg Et al.(1991)</p>
<p data-bbox="91 646 253 882">Data Flow Chart</p> 	<p data-bbox="795 646 1073 778">Positional Grammar</p>	<p data-bbox="1348 646 1753 801">G. Costagliola et al. (1990)</p>
<p data-bbox="91 989 388 1225">Structured Flow Chart</p> 	<p data-bbox="795 989 1264 1139">Symbol Relation Grammar</p>	<p data-bbox="1348 989 1690 1143">F.Ferruci et al. (1996)</p>

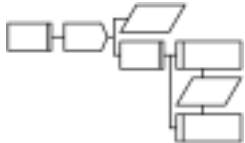


Motivation

- Formalism of diagram's structure and layout information
- Formalism of diagram processing method

Results

■ Types of graph grammars

Graph Grammar	Grammar's type (Rewriting rule, Attribute rule)	Diagram												
HCGG	<i>Context-free</i> (67, 723)													
HNGG	<i>Context-free, precedence</i> (280, 1248)	<table border="1" data-bbox="1271 905 1734 996"> <tr> <td>Program Code:</td> <td>Program Specification</td> </tr> <tr> <td>Program Name:</td> <td></td> </tr> <tr> <td>Library Code:</td> <td>Version:</td> </tr> </table>	Program Code:	Program Specification	Program Name:		Library Code:	Version:						
Program Code:	Program Specification													
Program Name:														
Library Code:	Version:													
HTGG	<i>Context-sensitive</i> (69, 308)	<table border="1" data-bbox="1271 1053 1728 1162"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Size</th> <th>G/L</th> </tr> </thead> <tbody> <tr> <td>x</td> <td>int</td> <td>2</td> <td>G</td> </tr> <tr> <td>y</td> <td>float</td> <td>4</td> <td>L</td> </tr> </tbody> </table>	Name	Type	Size	G/L	x	int	2	G	y	float	4	L
Name	Type	Size	G/L											
x	int	2	G											
y	float	4	L											

■ Integrated processing methods of diagrams

Diagram Processing System **KEYAKI-CASE2000**



Contents

1. Introduction

2. Program Flowcharts and Program Specification Forms

Hichart

Hiform

3. An Attribute Graph Grammar for Hierarchical Diagrams

4. Attribute Graph Grammars for Tabular Diagrams

Nested
Diagram

Tessellation
Diagram

5. Diagram Processing System

HichartED HiTS LIVE

HiformED

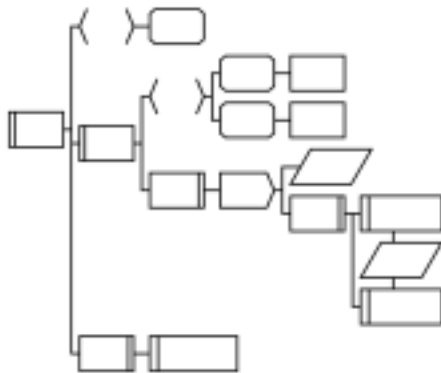
6. Conclusion

2. Program Flowcharts and Program Specification Forms

Diagram in Program Specifications

Hichart

Hierarchical Diagram



Hiform

(Tabular Diagrams)

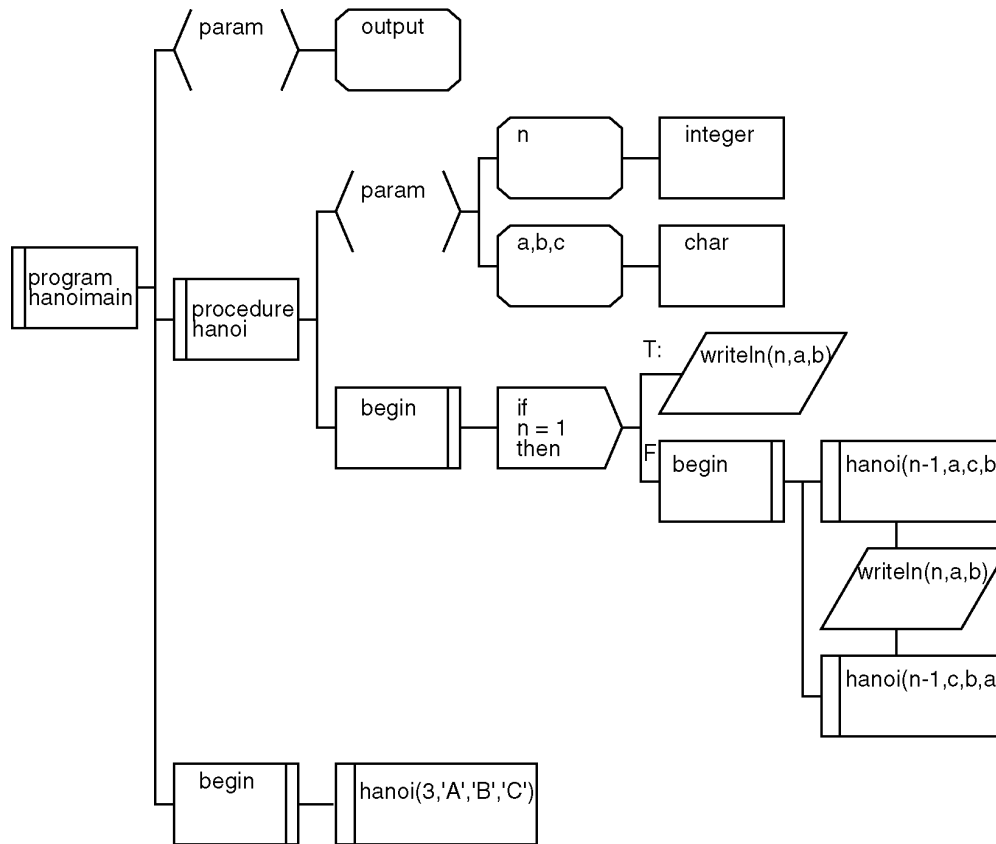
Nested Diagram

Program Code:	Program Specification
Program Name:	
Library Code:	Version:

Tessellation Diagram

Name	Type	Size	G/L
x	int	2	G
y	float	4	L

2.1 Hierarchical Diagram for Program Flowchart



A Hichart program flowchart (Tower of Hanoi).

2.2 Tabular Diagrams for Program Specification Forms

Hiform

(a program specification language)

- 17 types of Forms based on ISO6592
- A collection of tabular forms

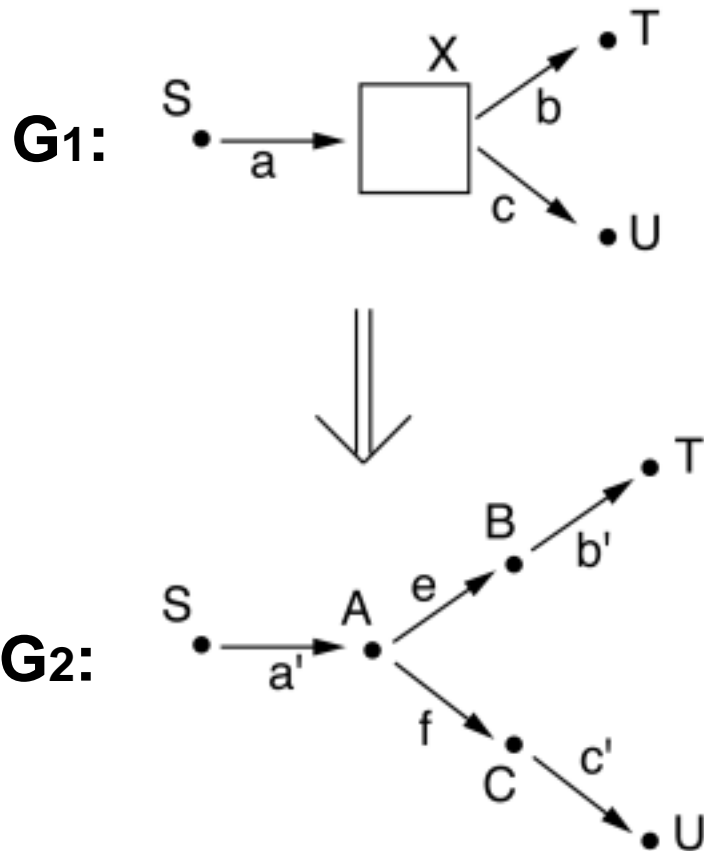
Project Code:	A 5
Program Name:	Program Specification-1 p
Library Code:	Version:
Author:	Original Release:
Approver:	Current Release:
Problem Description:	
Problem Supplementary Information (Theoretical Principles, Methods and References):	
Problem Solution: 1. Conventions and Terminology 2. Principles and Algorithms	



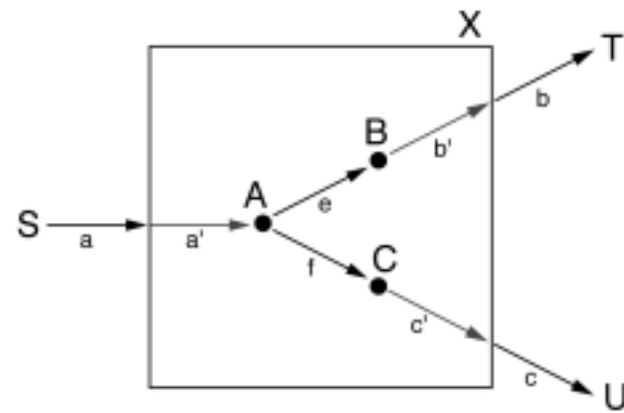
3. An Attribute Graph Grammar for Hierarchical Diagrams

Attribute Context-sensitive NCE (Neighborhood Controlled Embedding) Graph Grammar [Rozenberg et al. 1982]

Derivation :

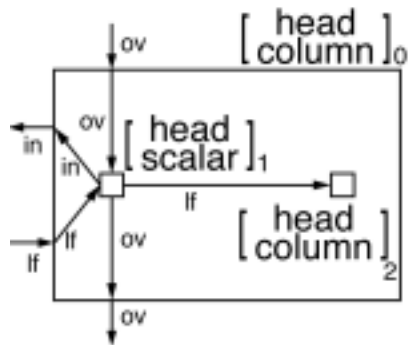


Rewriting Rule:



Attribute Context-free NCE Graph Grammar

Production with attribute rules:



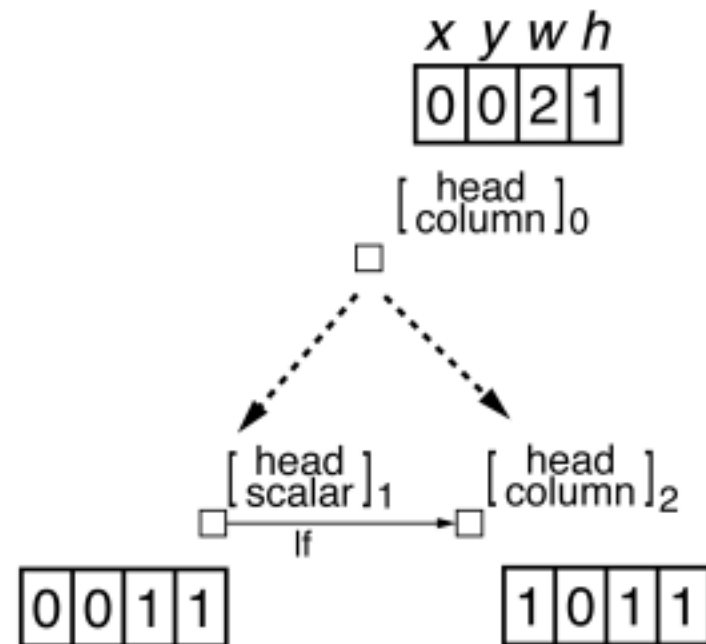
Semantic Rule

```

x(1) = x(0)
x(2) = x(0)+width(1)
y(1) = y(0)
y(2) = y(0)

width(0) = width(1)+width(2)
height(0) = max( height(1), height(2))
    
```

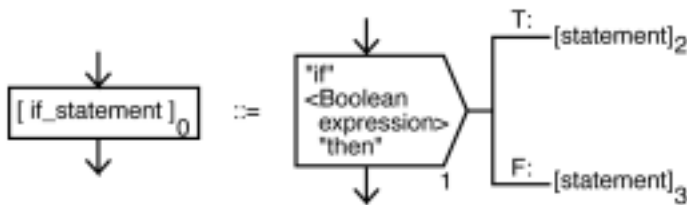
Attribute rule evaluation in CF NCE GG :



■ Grammar 3.1 HCGG

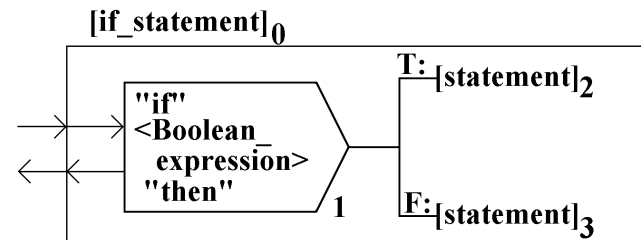
HCGG (HiChart Graph Grammar) is an attribute context-free NCE graph grammar for hierarchical diagrams in Hichart such as :

Production Example of HCGG



if_statement (1)

Production



Semantic Rules

$top(2)=top(0)$	$cl(2)=""T:"$
$top(3)=bottom(2)+GapY$	$cl(3)=""F:"$
$x(1)=x(0)$	$id(1)=id(0)$
$x(2)=x(0)+w(1)+GapX$	$id(2)=id(1)+1$
$x(3)=x(0)+w(1)+GapX$	$id(3)=id(2)+nc(2)$
$y(0)=(y(2)+y(3))/2$	$nc(0)=1+nc(2)+nc(3)$
$bottom(0)=max(bottom(1),bottom(3))$	

$w(1)=MinW$

$h(1)=get_height(["if",<Boolean_expression>,"then"])$

$cell(1)=""exclusive_selection"$

$string(1)=get_str(["if",<Boolean_expression>,"then"])$

$lines(1)=get_line(1,[2,3])$

Features of HCGG

GG	Type	Rewriting Rule	Attribute Rule
HCGG	Context-free	67	723



■ Property 3.2

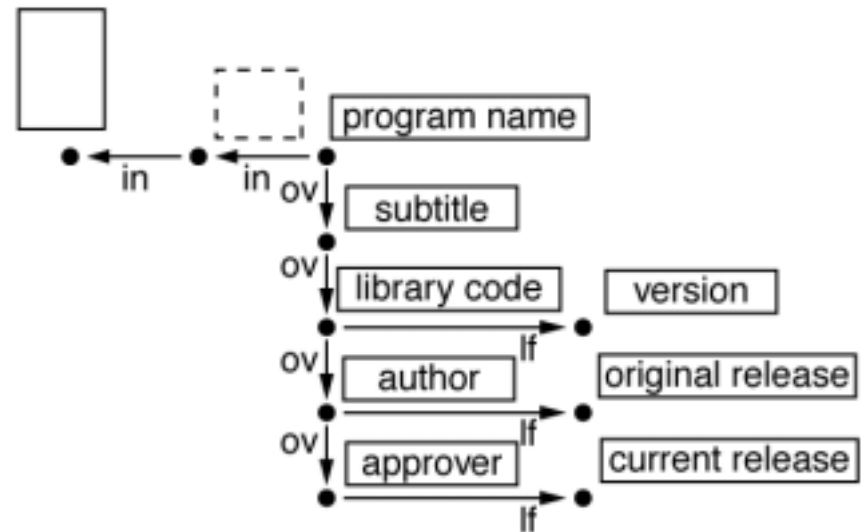
Attribute rules in HCGG are evaluated in linear time.

3.2 Attribute Graph Grammars for Tabular Diagrams

■ Nested Diagram and Its Corresponding Marked Graph

program name :	
subtitle :	
library code :	version :
author :	original release :
approver :	current release :

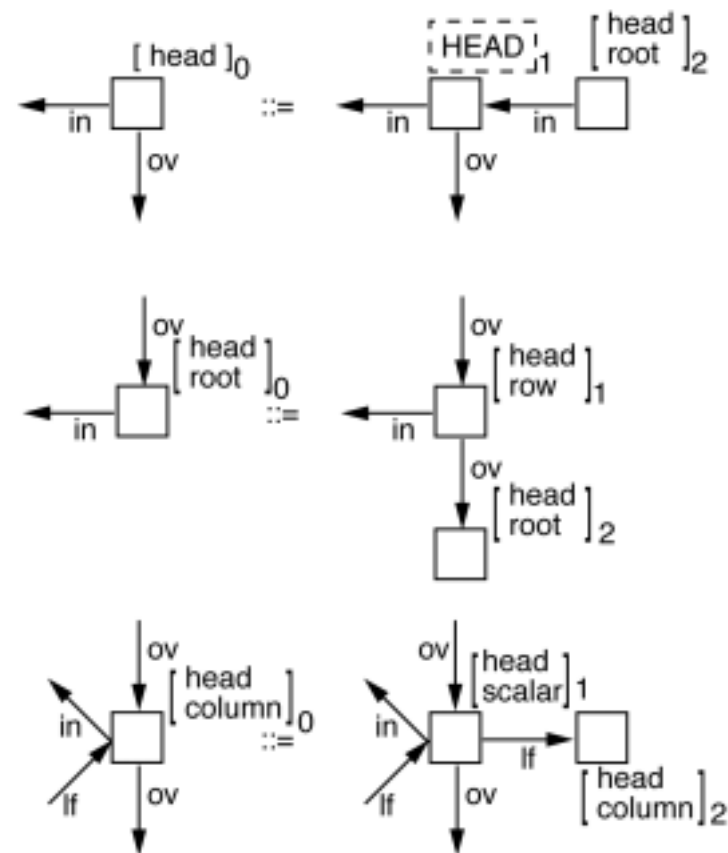
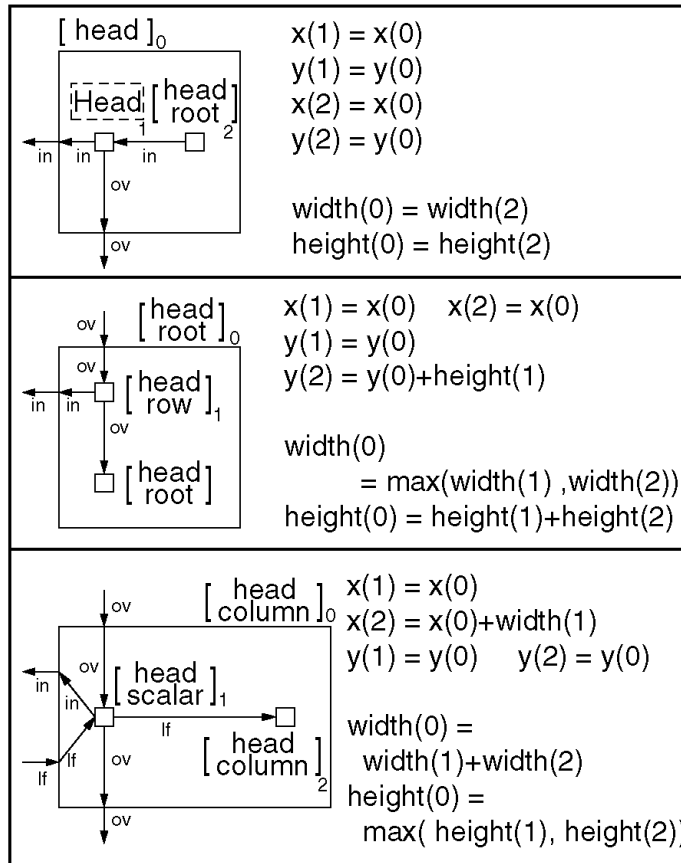
program name :	
subtitle :	
library code :	version :
author :	original release :
approver:	current release :



Grammar 4.1 HNGG

HNGG (Hiform Nested Graph Grammar) is an attribute context-free NCE graph grammar for the nested diagrams such as:

Production Examples of HNGG



Features of HNGG

GG	Type	Rewriting Rule	Attribute Rule
HNGG	Context-free	280	1248

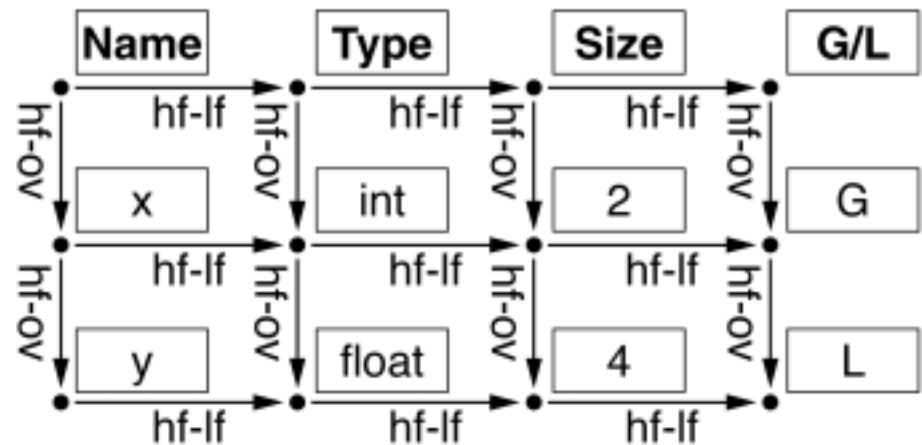


■ Property 4.2

HNGG is a **precedence graph grammar** (see e.g. Franck 1978).

■ Tessellation Diagram and Its Corresponding Graph

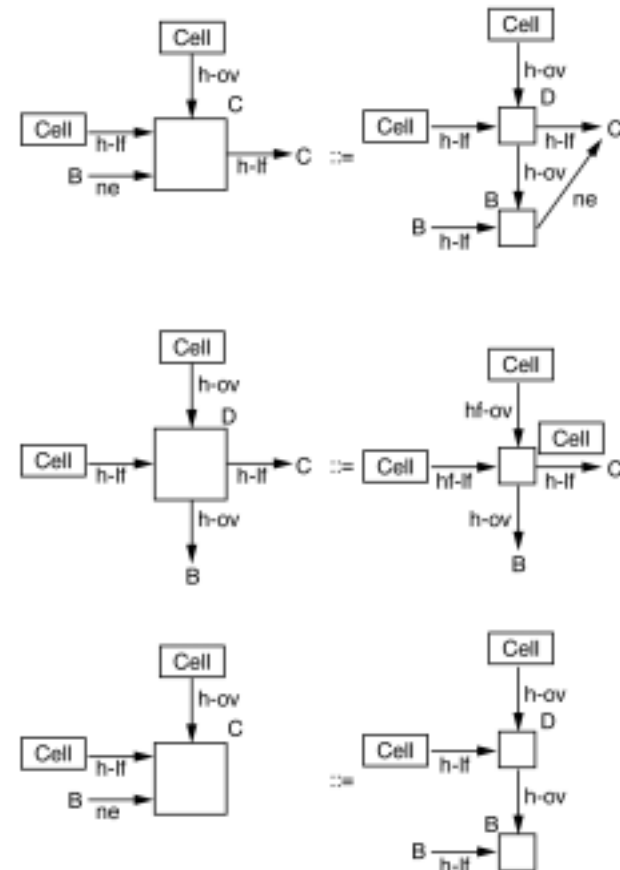
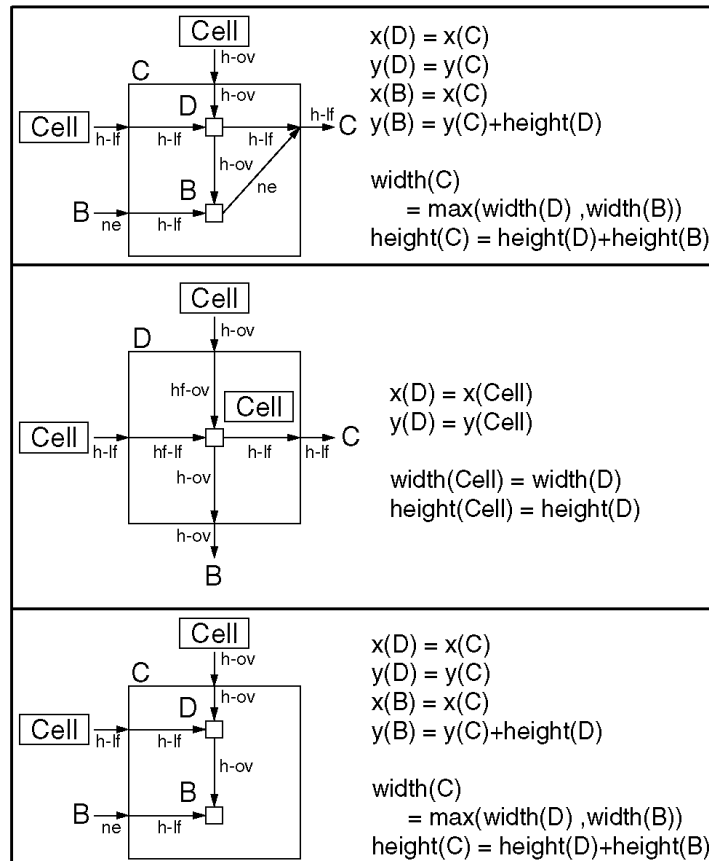
Name	Type	Size	G/L
x	int	2	G
y	float	4	L



■ Grammar 4.3 HTGG

HTGG (Hiform Tessellation Graph Grammar) is an attribute context-sensitive NCE graph grammar for the tessellation diagrams such as:

Production Example of HTGG



Features of HTGG

GG	Type	Rewriting Rule	Attribute Rule
HTGG	Context-sensitive	69	308





5. Diagram Processing System

KEYAKI – CASE2000 Concept

1. HichartED

Hichart program diagram editing component

2. HiTS

Hichart program diagram filtering component

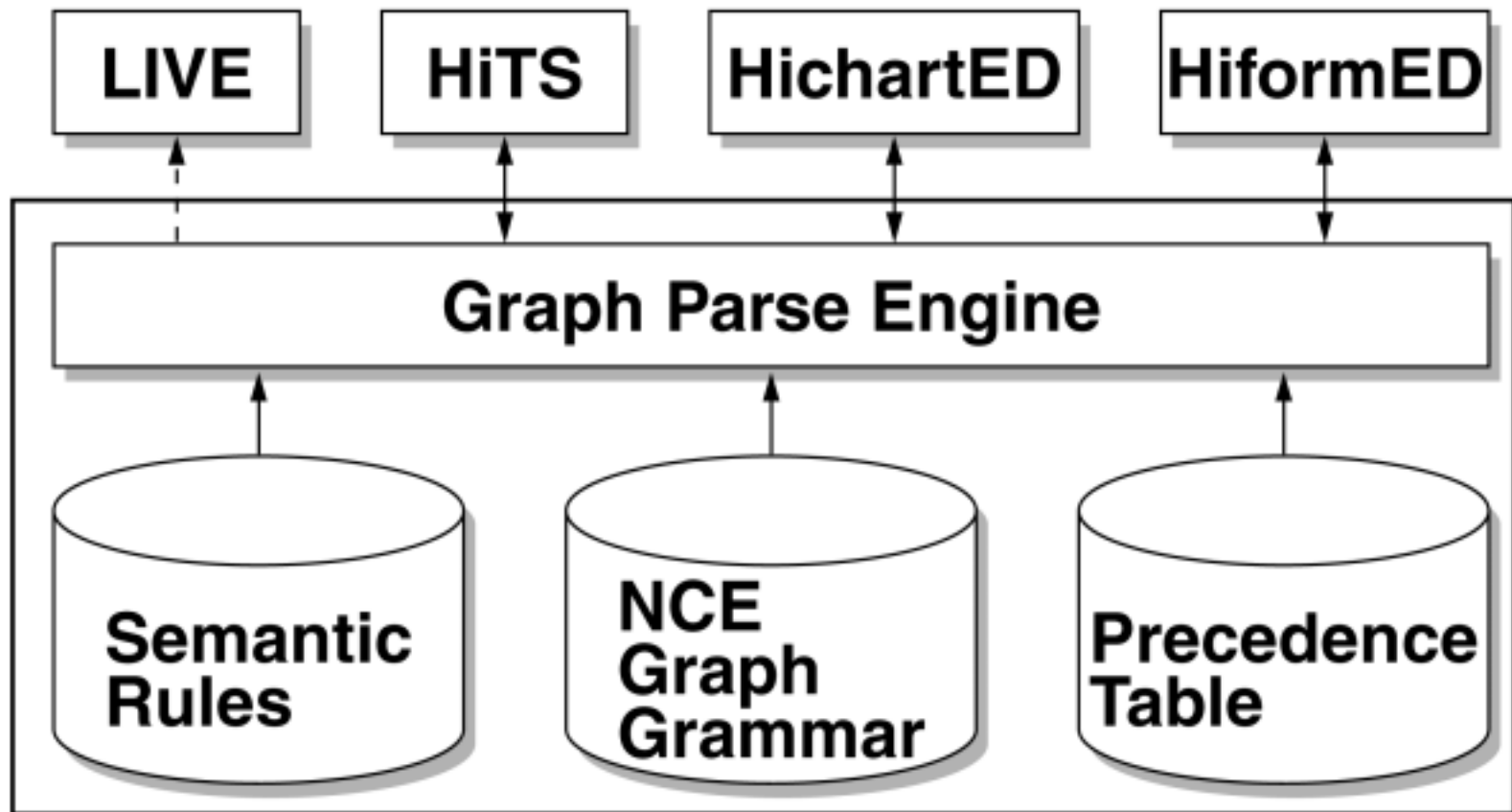
3. LIVE

Program variable analyzing component

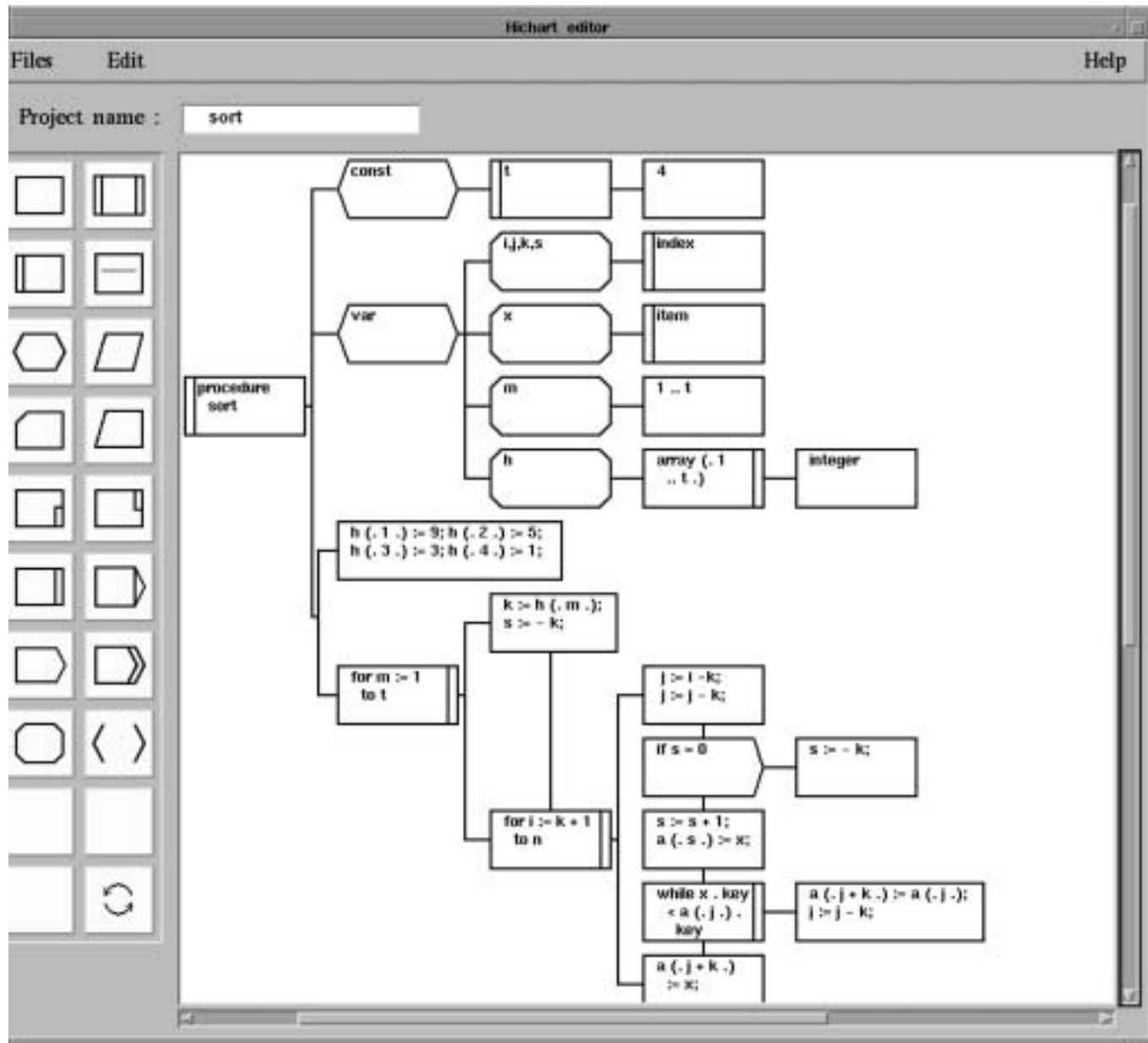
4. HiformED

Hiform diagram component

KEYAKI-CASE2000 Inside



Screen Concept of HichartED



Execution Screen in HiTS

The screenshot displays the HiChart Translation Service interface. On the left is a control panel with the following elements:

- Language selection: English / Japanese
- Input a target source file:
- Select Layout type:
- Input Cell's size: horizontal length vertical length
- Select objects if you need:
 - Picture
 - HiChart flowchart (.pc file)
 - Abstract HiChart flowchart (.a.h file)
 - C-code
 - Abstract H-Codes
 - Source code
-

At the bottom of the control panel, it reads: "Computer Software Lab./ Data Base Division" and "Takeshi Miyazaki e-mail: miyazaki@i.s.u.t.ac.jp".

The main execution screen on the right is divided into two sections:

- Top Section:** A large flowchart representing the execution process. It consists of numerous rectangular and oval-shaped nodes connected by lines, illustrating the flow of data and control. Two large, hand-drawn eyes are superimposed on the top right of this section.
- Bottom Section:** A code editor window showing the following C code:

```
#include <stdio.h>
#include <limits.h>

typedef enum {FALSE, TRUE}
BOOL;

#define DF INT_MAX
#define N 8
#define START 0

int lr[N], ld[N];

void di_extra(int start,
(
    BOOL alist[N]);
int l, current, s

for (i = 0; i < N
    alist[i] = I
    ld[i] = I
    lr[i] = -
}
alist[start] = TR
count = 1;
ld[start] = 0;

do {
    sin = DF
    for (i =
        ld[i] < sin) {
        ...
    }
}
```

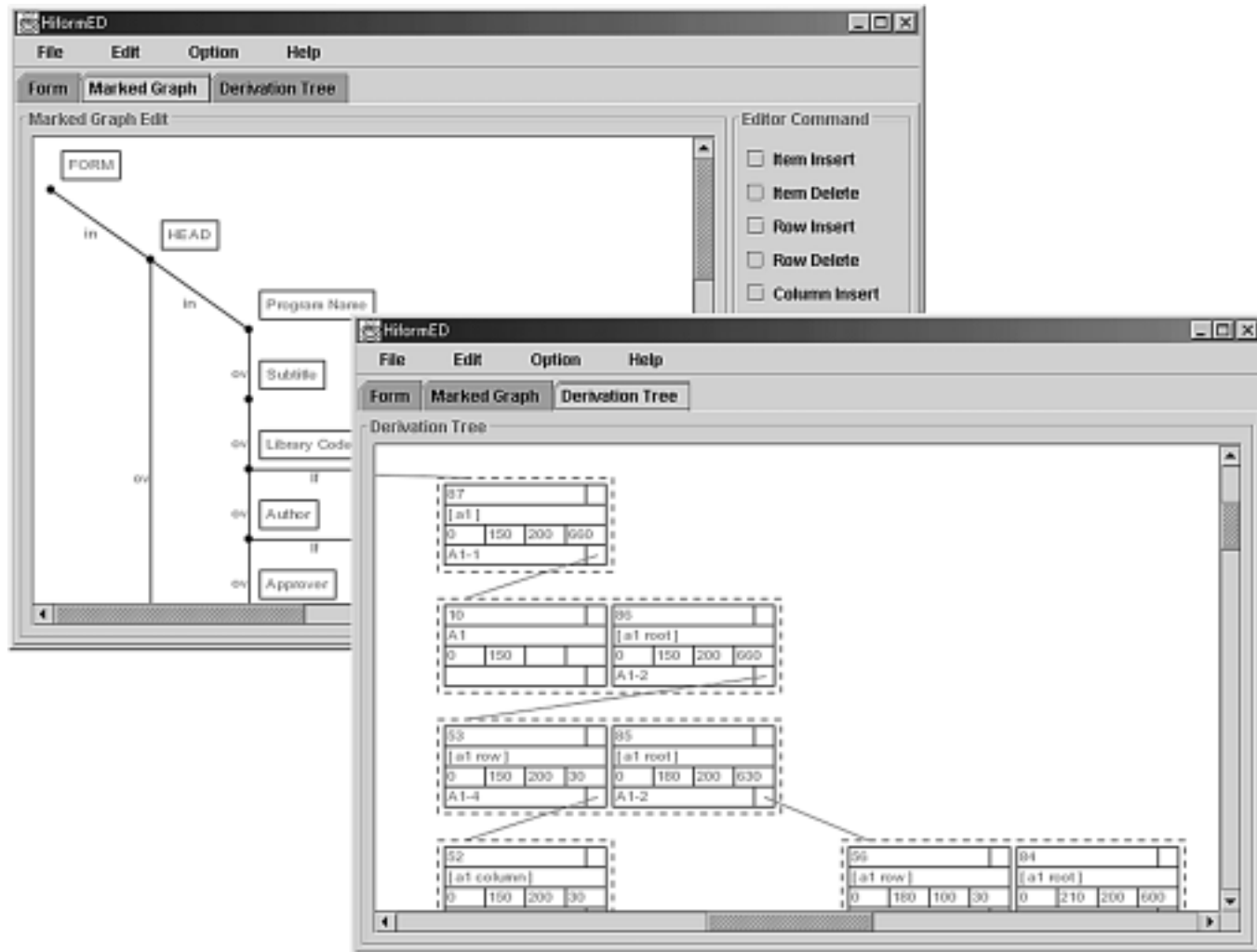
Execution Screen in LIVE

The screenshot displays the LIVE (Live Implementation of Visual Execution) environment. It features several windows:


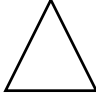
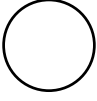
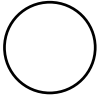
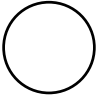
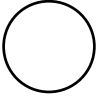
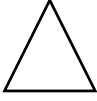

- Source Code Window:** Shows the Pascal-like source code for a matrix multiplication program. The code includes constants, variable declarations, and nested loops for reading matrix A, matrix B, and calculating the product matrix C.
- File Browser Window:** Displays a file list with columns for file names and sizes. The selected file is `matrimul.liv`. Buttons for "了解" (Info), "フィルター" (Filter), and "取消" (Cancel) are visible.
- Data Flow Graph Window:** Shows a complex graph of nodes and edges representing the execution flow of the program. Nodes are labeled with code line numbers (e.g., 00010, 00011, etc.) and represent operations like reading elements, calculating sums, and writing to the output matrix.
- System Windows:** Includes a clock, a task manager window titled "タスクマネージャ", and a window titled "Issue01" showing a waveform.

```
00001 program Matrimul(Input, Output);
00002 const
00003   M = 4;
00004 var
00005   I: 1..M;
00006   J: 1..M;
00007   K: 1..M;
00008   Sum, Element: Integer;
00009   A: array[1..M, 1..M] of Integer;
00010   B: array[1..M, 1..M] of Integer;
00011   C: array[1..M, 1..M] of Integer;
00012
00013 begin
00014   for I := 1 to M do
00015     begin
00016       for J := 1 to M do
00017         begin
00018           Read(Element);
00019           A[I, J] := Element;
00020         end;
00021       end;
00022     for I := 1 to M do
00023       begin
00024         for J := 1 to M do
00025           begin
00026             Read(Element);
00027             B[I, J] := Element;
00028           end;
00029         end;
00030       for I := 1 to M do
00031         begin
00032           for J := 1 to M do
00033             begin
00034               Sum := 0;
00035               for K := 1 to M do
00036                 Sum := Sum + A[I, K] * B[K, J];
00037             end;
00038             C[I, J] := Sum;
00039           end;
00040         end;
00041       for I := 1 to M do
00042         begin
00043           for J := 1 to M do
00044             begin
00045               write(C[I, J]);
00046             end;
00047             writeln;
00048           end;
00049         end;
00050       end;
00051     end;
00052   end;
00053 end;
```

Screen Concept of HiformED



6. Conclusion

System	NCE GG	System with NCE GG	System without NCE GG
HichartED			
HiTS		Not yet	
LIVE	Not yet	Not yet	
HiformED		Not yet	

Our Project Web Site :

Including detailed description of Graph Grammars

■ **URL:**

<http://www.hichart.org/>