NCEグラフ文法によるダイアグラムの構文的処理

Syntactic Processing of diagrams by NCE Graph Grammrs

有田　友和[†]　冨山　聖宣[†]　杉田　公生[††]　宮寺　庸造[††††]　土田　賢省[†††]　夜久　竹夫[†]
Tomokazu Arita　Kiyonobu Tomiyama　Kimio Sugita　Youzou Miyadera　Kensei Tsuchida　Takeo Yaku
日本大学[†]　東海大学[††]　学芸大学[††††]　東洋大学[†††]
Nihon University　Tokai University　Gakugei University　Toyo University

1. Introduction

We deal with syntactic definitions and processing of program diagrams based on graph grammars with respect to the mechanical drawing. We consider diagrams which are program flowchart diagram and tabular diagrams. Furthermore, we introduce an integrated diagram processing method based on NCE graph grammars. The results could be applied to general diagram processing.

2. Diagrams Used in Program Specifications

We review diagrams appeared in the software visualization. We consider two types of diagrams. One is hierarchical diagrams for program flowchart and The other is tabular diagrams for program specification forms.

2.1 Hierarchical Diagrams for Program Flowchart

We introduce a program flowchart description language Hichart (Hierarchical flow CHART description language). Hichart is of a tree structured program flowchart type.

2.2 Tabular Diagrams for Program Specification Forms

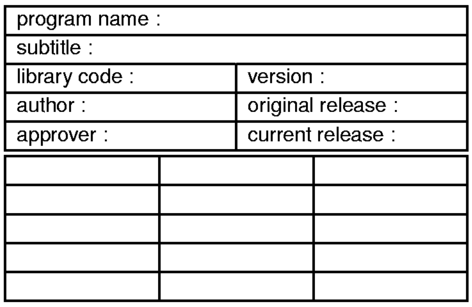
We introduce here a program specification language called Hiform based on ISO6592. The International Organization for Standardization issued a guideline in ISO6592 and described all items in program documentation in Annexes A, B and C. We considered the ISO6592 items and introduced Hiform96, which includes all items defined in these Annexes. Hiform is defined by 17 types of forms.

Hiform was originally developed for the purpose of facilitating the software development at schools. Hiform Specification is a collection of tabular diagrams.

The order among tabular forms is defined by a context-free string grammar. The order and graphical structure of cells inside tabular diagrams is defined by graph grammars.

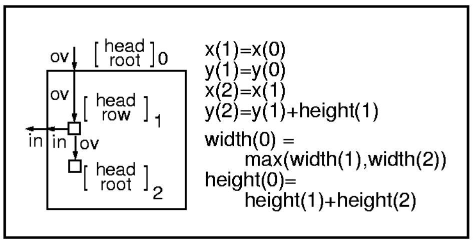
3. Attribute Graph Grammars for Diagrams

The layout information for these diagrams in program specifications is defined by attribute graph grammars. By defining these layout information using graph grammars, we obtain the following advantages. (1) It is possible to draw diagrams automatically. (2) It is possible to edit diagrams by syntactic methods. (3) It is possible to define layout conditons for diagrams declaratively.



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

Fig.1 A specification form



x(1)=x(0)
y(1)=y(0)
x(2)=x(1)
y(2)=y(1)+height(1)
width(0) =
max(width(1),width(2))
height(0)=
height(1)+height(2)

Fig. 2 A *production* with attribute rules

4. Diagram Processing System

We here describe a diagram processing system which is called KEYAKI-CASE2000. KEYAKI-CASE2000 consists of the following components: (1) Hichart program diagram editing component (HichartED), (2) Hichart program diagram filtering component (HiTS), (3) Program variable analyzing component (LIVE), and (4) Hiform diagram component (HiformED).

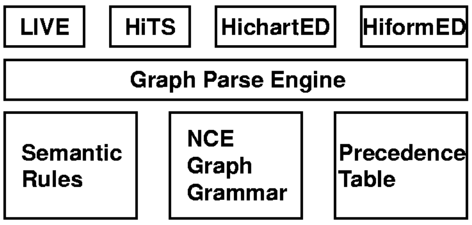


Fig. 3 KEYAKI - CASE2000

5. Conclusion

We proposed syntactic processing of diagrams by NCE graph grammars. Then we developed diagram editor system. The system allows users to edit in syntax-directed mannar and draw diagrams mechanical based ongraph grammatical method.

References

[1]