SYNTACTIC TABULAR FORM PROCESSING BY PRECEDENCE ATTRIBUTE GRAPH GRAMMARS

TOMOKAZU ARITA[†] KIMIO SUGITA[††] KENSEI TSUCHIDA[†††] TAKEO YAKU[†]

Dept. App. Math.[†]
Nihon University
3–25–40, Sakurajosui, Setagaya, Tokyo, 156-8550, Japan

Dept. Math.[††]
Tokai University
1117, Kitakaname, Hiratsuka, Kanagawa, 259–1292, Japan

Dept. Inf. & Comp. Sci.[†††]
Toyo University
2100, Kujirai, Kawagoe, Saitama, 350–8585, Japan

Abstract We deal with mechanical documentation in software development. First, we introduce a mathematical model, called precedence attribute NCE graph grammars, for mechanical documentation. Next we introduce syntactic definition of nested tabular forms by an attribute NCE graph grammar, and show that the grammar has precedence property. Furthermore, we introduce parsing methods for nested tabular forms.

Keywords Software Methodologies, Software Documentation, Graph Grammars

1 INTRODUCTION

Mechanical documentation such as automatic drawing and editing of software specification (see e.g. [2]) forms is considered one of the important issues in software development tools.

Software documents include tabular forms such as software specification forms and diagrams such as program flowcharts. Furthermore, the tabular forms are classified into (1) nested structured form in which items are linked hierarchically to each other, and (2) tessellation structured form such as symbol tables and spread sheets (see e.g. [8]). This paper deals with nested structured form and its mechanically manipulating problems.

In mechanical documentation, it is necessary to formally define tabular forms and the drawing conditions. Tabular forms generally have infinitely many items inside. And the order and the location of the items are valid. Accordingly, syntactic definition is effective in mechanical manipulation of the tabular forms. Attribute graph grammars also formulate universally syntactic structure and the visual structure among items in the tabular forms by attributed productions.

Franck [1] introduced precedence graph grammars and applied them to nested program diagrams called PLAN2D. Nishino [4] introduced an attribute graph grammar with respect to a drawing problem of tree-like diagrams and formalized transformation of tree-like diagrams. In [4], the drawing problems were specified by semantic rules of attributes. We have also studied syntactic and algorithmic manipulation of diagrams [12] [11], [10], [6]. We formulated hierarchical structured program diagram by an attribute graph grammar [6]. In 2000, we partly introduced a syntactic definition of software specification forms based on ISO6592 standard using graph grammars [12].

The purpose of this paper is to characterize graph grammars which provide formal definition of program specification forms with respect to syntactic manipulation and mechanical drawing. In Section 2, we review a program specification form. In Section 3, we introduce a precedence attribute NCE graph grammar as a model. In Section 4, we introduce an attribute NCE graph grammar for program specification forms and show that the grammar has precedence property. Section 5 provides parsing algorithms. Section 6 provides conclusions.

2 PROGRAM DOCUMENTATION LANGUAGE

We review here a program documentation lanuage *Hiform*. Hiform was originally developed for the purpose of assisting the programming education. We took the ISO6592 items into consideration and introduced Hiform which includes all items defined in Annexes of ISO6592 [3]. Hiform is defined by 17 different types of tabular forms [9]. The following Fig. 2.1 shows a Hiform program documentation form.

We employ a marked graph as an expression of a tabular form. We introduce below a marked graph which represents a tabular form. A *marked graph* is defined as follows: (1)*A node label* of graph shows an *item* of a tabular form.

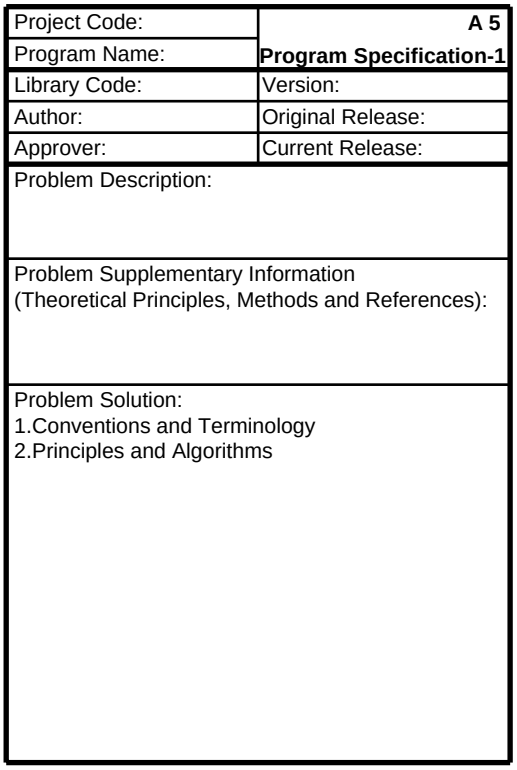

Fig.2.1 A Hiform Document

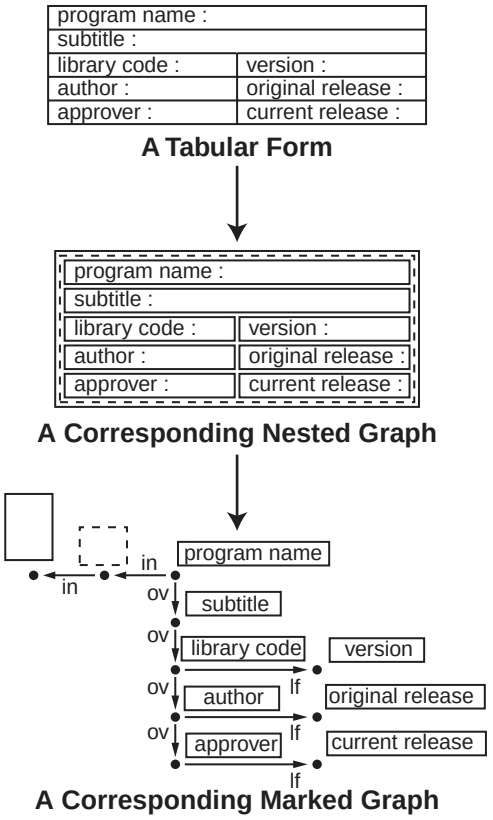

Fig. 2.2 A Tabular Form, a Nested Graph and a Marked Graph

A node label is called a *mark*. (2)An *edge label* shows relations between items. 'lf' denotes the meaning of 'left of'. 'ov' denotes the meaning of 'over'. 'in' denotes the meaning of 'within' [1]. Fig. 2.2 shows an expression of a tabular form.

3 ATTRIBUTE NCE GRAPH GRAMMAR

In this section, we introduce attribute NCE graph grammars. We note that Franck's graph grammars could not represent general tabular forms such as tessellation forms [12], since the graph grammar could not execute edge rewriting. We consider here universal models which are commonly applied to general tabular forms. For this purpose, we employ edNCE graph grammars and introduce the following models. However, it is noted that Franck's graph grammar has capability to formalize tabular forms if the forms are restricted to "nested" type we deal with later.

3.1 ATTRIBUTE NCE GRAPH GRAMMARS

We add attributes to edNCE graph grammars and introduce an attribute NCE graph grammar, as following.

Definition 3.1.1 [7] An *edNCE graph grammar* is a 6–tuple $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$, where Σ is the *alphabet* of node labels, $\Delta \subseteq \Sigma$ is the alphabet of *terminal* node labels, Γ is the alphabet of edge labels, $\Omega \subseteq \Gamma$ is the alphabet of *final* edge labels, P is the finite set of *productions*, and $S \in \Sigma - \Delta$ is the *initial nonterminal*. A production is of the form $X \rightarrow (D, C)$ with $X \in \Sigma - \Delta$, D is a graph over the Σ and Γ, and $C \subseteq \Sigma \times \Gamma \times \Gamma \times V_D \times \{in, out\}$ is the *connection relation*, where V_D is a set of nodes on D. □

Definition 3.1.2 An *attribute NCE graph grammar* is a 3–tuple $AGG = \langle G, Att, F \rangle$, where

1. $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ is called an *underlying graph grammar* of AGG. Each production p in P is denoted by $p = X \rightarrow (D, C)$. $Lab(D)$ denotes the set of all occurrences of the node symbols labeling the nodes in the graph D.

2. Each node symbol $Y \in \Sigma$ of G is associated with two disjoint finite sets $Inh(Y)$ and $Syn(Y)$ of *inherited* and *synthesized attributes*, respectively. We denote the set of all attributes of nonterminal node symbols X by $Att(Y) = Inh(Y) \cup Syn(Y)$. $Att = \bigcup_{Y \in V} Att(Y)$ is called the set of attributes of AGG. We assume that $Inh(S) = \emptyset$. An attribute a of Y is denoted by $a(Y)$, and set of possible

values of a is denoted by $V(a)$.

3. Associated with each production $p = X_0 \rightarrow (D, C) \in P$ is a set F_p of *semantic rules* which define all the attributes in $Syn(X_0) \cup \bigcup_{Y \in Lab(D)} Inh(Y)$. A semantic rule defining an attribute $a_0(X_{i_0})$ has the form $a_0(X_{i_0}) := f(a_1(X_{i_1}), ..., a_m(X_{i_m}))$, $0 \leq i_j \leq |Lab(D)|$, $X_{i_j} \in Lab(D), 0 \leq j \leq m$. Here $|Lab(D)|$ denotes the cardinality of the set $Lab(D)$, and f is a mapping from $V(a_1(X_{i_1}) \times ... \times a_m(Xi_m))$ into $V(a_0(X_{i_0}))$. In this situation, we say that $a_0(X_{i_0})$ depends on $a_j(X_{i_j})$ for $j, 0 \leq j \leq m$ in p. The set $F = \bigcup_{p \in P} F_p$ is called the *set of semantic rules* of AGG. □

3.2 PRECEDENCE RELATIONS

We modify the Franck's precedence relations [1] on Franck's context-free graph grammar and propose precedence relations on edNCE graph grammars for efficient parsing. In this part, we deal with a following edNCE grammar. Let G be an edNCE grammar. All production of G is of the form $X \rightarrow (D, C)$ and $\forall(\sigma, \alpha, \alpha, x, d) \in C$ (with $\sigma \in \Sigma$, $\alpha \in \Gamma$, $x \in V_D$, and $d \in \{in, out\}$).

Notation 3.2.1 For every $m \in \Gamma$ and $\forall\# \in \Sigma$ let

$$\doteq_m \stackrel{def}{=} \left\{ (A, B) \;\middle|\; \begin{array}{l} P \ni p : X \rightarrow (D, C), \\ \text{there is an edge } (x, y) \text{ on } D \\ \text{where } x \text{ is marked } A, \\ y \text{ is marked } B \text{ and} \\ (x, y) \text{ is labeled } m. \end{array} \right\}$$

$$\rightarrow_m \stackrel{def}{=} \left\{ (A, B) \;\middle|\; \begin{array}{l} P \ni p : A \rightarrow (D, C), \\ C \ni (\#, m, m, x, in), \\ \text{and the mark of } x \text{ is } B. \end{array} \right\}$$

$$\leftarrow_m \stackrel{def}{=} \left\{ (B, A) \;\middle|\; \begin{array}{l} P \ni p : A \rightarrow (D, C), \\ C \ni (\#, m, m, y, out), \\ \text{and the mark of } y \text{ is } B. \end{array} \right\}$$

□

Notation 3.2.2 [1] For every $m \in \Gamma$ let $\lessdot_m \stackrel{def}{=} \doteq_m \cdot \stackrel{+}{\rightarrow}_m$, $\gtrdot_m \stackrel{def}{=} \stackrel{+}{\leftarrow}_m \cdot \doteq_m$, and $\lessdot\cdot\gtrdot_m \stackrel{def}{=} \stackrel{+}{\leftarrow}_m \cdot \doteq_m \cdot \stackrel{+}{\rightarrow}_m$, where $+$ denotes transitive closure. □

Definition 3.2.3 Precedence relations are $conflictless$ if and only if for every $m \in \Gamma$ the relations $\lessdot_m, \doteq_m, \gtrdot_m$ and $\lessdot\cdot\gtrdot_m$ are pairwise disjoint [1]. □

Definition 3.2.4 [1] A context-free attribute NCE graph grammar is called a *precedence attribute NCE grammar* if and only if (i) the precedence relations are conflictless, (ii) all rules are uniquely invertible, and (iii) there is no reflexive nonterminal symbol in the grammar. □

A language is called a precedence attribute NCE graph language if it is generated by a precedence attribute NCE graph grammar.

4 AN ATTRIBUTE GRAPH GRAMMAR FOR HIFORM

We propose an attribute graph grammar which characterizes the Hiform documents. The grammar is called *Hiform Nested Graph Grammar*(HNGG). We show productions of HNGG in Fig. 4.1 HNGG consists of 280 productions. The label of the start graph is "[struct]." We also construct 1248 attribute rules of HNGG as shown in Fig. 4.1 Each production is associated with attribute rules. These attribute rules are mainly used for evaluating the positions and the sizes of items. The set of all productions is in [13].

Proposition 1 The grammar HNGG above is a precedence attribute NCE graph grammar.

Proof. We can construct 5376 relations over the marks in HNGG as shown in Fig. 4.3 The relation are shown to be pairwise disjoint. □

5 PARSING OF PRECEDENCE ATTRIBUTE NCE GRAPH LANGUAGE

5.1 SYNTACTIC ANALYSIS

We introduce an outline of parsing algorithms (cf. [1]).

The algorithm 'parser' includes repetition of the algorithm 'reduce' while an input graph to the algorithm 'reduce' is not a start graph of the graph grammar. An algorithm 'reduce' finds a handle in input graph and replace this handle to new node which has a mark of left hand side of a production for this handle. (a handle is a graph which is isomonophic for a graph in a right hand side of a production.)

The 'reduce' seeks for a handle in inputted graph by an algorithm 'seekForHandle' in the first step. A handle is found by using the precedence table. In the second step, the handle is replaced by a new node which mark is a left hand side of production for this handle by calling an algorithm 'replaceHandleToLeftHandSideOfProduction'. Finally, 'reduce' returns a graph replaced by a handle.

The algorithm 'replaceHandleToLeftHandSideOfProduction' is consists of 4 steps. It is necessary for this algorithm to input a graph and a handle in the graph. First, the handle is removed from the graph by calling 'moveHandleFormGraph'. Second, a production for the handle is searched by calling 'searchProduction'. Third, derivation subtree is made by the handle and the production by calling 'makeDerivationTree'. Finally, a new node is embeded in the graph by 'embedLeftHandSideToGraph'. This new node's mark is the left hand side of the production.

Algorithm parser
Input: Marked Graph and Precedence Table
Output: Derivation Tree

Method
void parser(MarkedGraph graph, PrecedenceTable ptable){
    while(a mark of any node in graph is not start_graph){

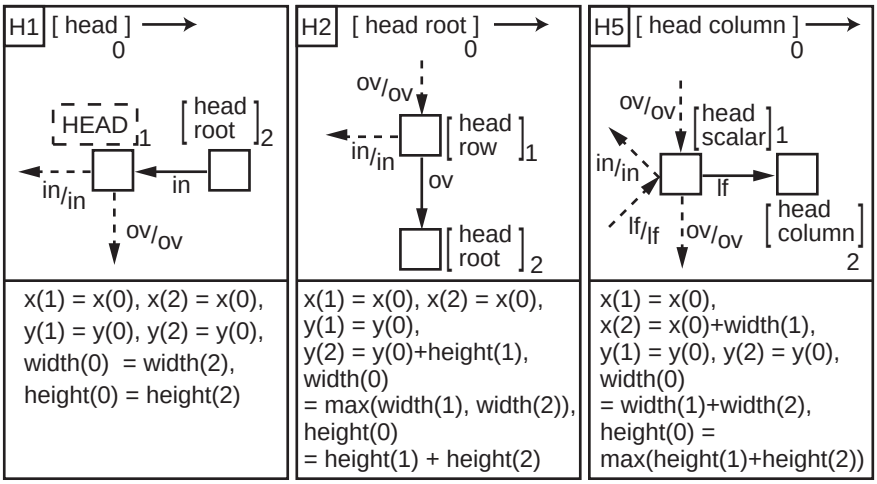

Fig. 4.1 Productions of HNGG

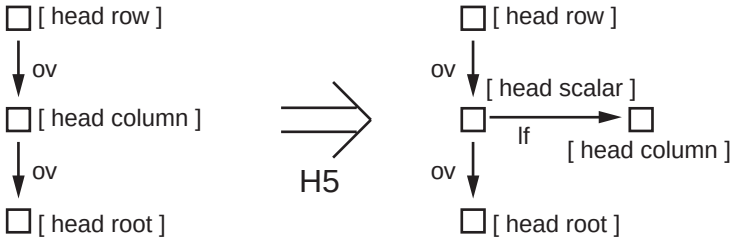

Fig. 4.2 An example of applying a production

Left \ Right	[head scalar]			[head column]			[head row]			[head root]		
	in	ov	lf	in	ov	lf	in	ov	lf	in	ov	lf
[head scalar]		<.>	<		<.>	=		<.>			>	
[head column]		<.>			<.>			<.>			>	
[head row]		<			<			<			=	

Fig. 4.3 A Part of Precedence Relation of HNGG

```
            graph = reduce(graph, ptable);
        }
}

MarkedGraph reduce(
        MarkedGraph graph,
        PrecedenceTable ptable
        ){

        MarkedGraph handle = seekForHandle(graph, ptable);
        graph = replaceHandleToLeftHandSideOfProduction(
                graph, handle );
        return graph;
}

MarkedGraph replaceRightWithLeft(
        MarkedGraph graph,
        MarkedGraph handle){

        moveHandleFormGraph(graph, handle);
        Production production = searchProduction(handle);
        makeDerivationTree( handle, production);
        graph = embedLeftHandSideToGraph(
                graph, production);
        return graph;
}
```

□

5.2 ATTRIBUTE EVALUATION FOR LAYOUT

Layout problems of nested diagrams are solved by attribute evaluation [4]. We use attributes which are *x*, *y*, *width* and *height*. Symbols *x* and *y* are used to calculate *x* coordinate

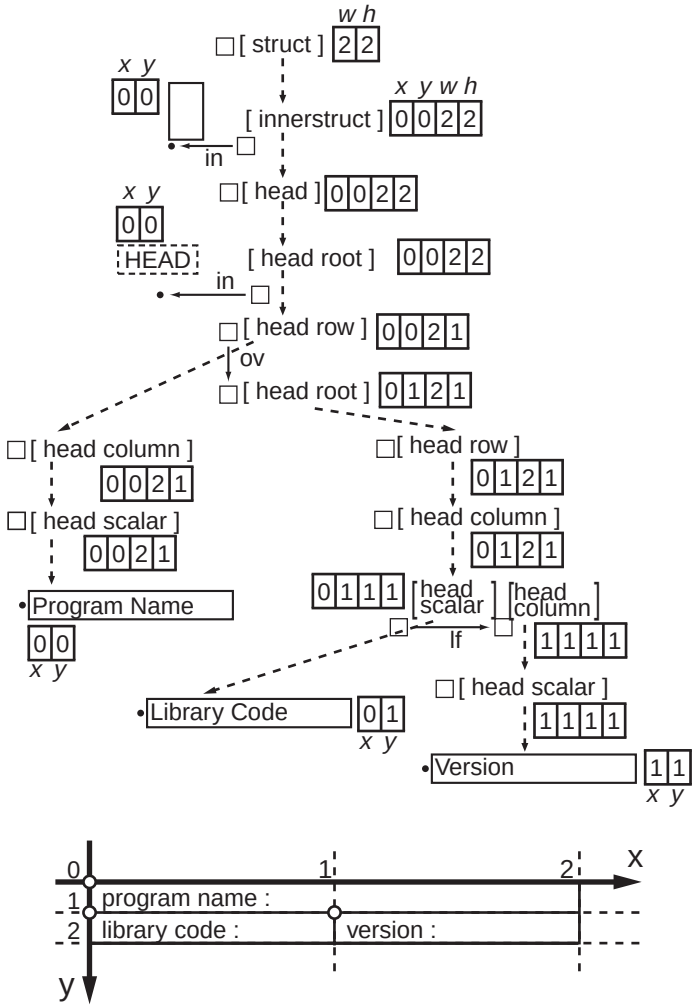

Fig. 5.1 A derivation tree (top) by HNGG with evaluated attribute values and a form (bottom) drawn based on their values

and *y* coordinate, respectively. And *width* and *height* are also used to calculate width and height, respectively. We illustrate a process of an attribute evaluation in Fig. 5.1 The yield of the derivation tree of Fig.5.1(top) is a program document shown in Fig 5.1(bottom). We have:

Proposition 2 Attributes in HNGG are evaluated in linear time. □

6 CONCLUSION

We proposed an attribute NCE graph grammar for tabular forms that are program documentation forms based on ISO6592. This graph grammar is necessary in order to form tabular forms and to develop a processing system of tabular forms. This graph grammar can formalize with respect to both the logical and visual structures. We note that ISO6592 ANNEX documentation requires at most 280 productions with 1248 attribute rules.

Furthermore, we proposed a parsing algorithm of this attribute graph grammar. This parser uses precedence relations for effective parsing. We are now developing a software documentation system utilizing our proposed approach in this paper.

Acknowledgments We thank Mr. S. Kanai's advise in the course of preparing the manusicript. We also thank to Mr. K. Tomiyama and S. Nakagawa for valuable discussions.

References

[1] Reinhold Franck, A Class of Linearly Parsable Graph Grammars, *Acta Infomatica*, 10, 1978, 175-201.

[2] G. Engels, R. Call, M. Nagl, et al., Software Specification Using Graph Grammars, *Computing*, 31, 1983, 317-346.

[3] ISO6592-1985, Guidelines for The Documentation of Computer-Based Application Systems, 1985.

[4] T. Nishino, Attribute Graph Grammars with Applications to Hichart Program Chart Editors, *Advances in Software Science and Technology*, 1, 1989, 426-433.

[5] F. Haro, Graph Grammars and Their Application to Computer Mathematics, *Master's Thesis, Tokai University*, 1989 (in Japanese).

[6] Y. Adachi, K. Anzai, K. Tsuchida and T. Yaku, Hierarchical Program Diagram Editor based on Attribute Graph Grammar, *Proc. IEEE COMPSAC*, 21, 1996, 205-213.

[7] Grsegorz Rozenberg (Ed.), *Handbook of Graph Grammar and Computing by Graph Transformation*, World Scientific Publishing ,1997.

[8] G. Santucci and L. Tarantino, A Hypertabular Visualizer of Query Results, *Proc. of the 1997 IEEE Symp. Visual Language*, 1997, 193-200.

[9] K. Sugita, Y. Adachi, Y. Miyadera, K. Tsuchida and T. Yaku, Proc. of Advanced Software Mechanisms for Computer-Aided Instruction information Literacy, *APEC-CIL'97*, 1997, session 8–4.

[10] K. Sugita, A. Adachi, Y. Miyadera, K. Tsuchida and T. Yaku, A Visual Programming Environment based on Graph Grammars and Tidy Graph Drawing, *Proc. Internat. Conf. Software Engin. (ICSE '98)*, 20-II, 1998, 74-79.

[11] A. Adachi, T. Tsuchida and T. Yaku, Program Visualization Using Attribute Graph Grammars, CD-ROM Book, IFIP World Computer Congress 98, (1998).

[12] T. Arita, K. Tomiyama, T. Yaku, Y. Miyadera, K. Sugita, K. Tsuchida, Syntactic Processing of Diagrams by Graph Grammars, *Proc. IFIP World Computer Congress ICS2000*, 2000, 145-151.

[13] T. Arita, A Precedence Attribute NCE Graph Grammar for Hiform,
http://www.hichart.org/keyaki/archive/HC00–001/