Application of Attribute edNCE Graph Grammars to Syntactic Editing of Tabular Forms

冨山 聖宣 (Kiyonobu TOMIYAMA)[†] 有田 友和 (Tomokazu ARITA)[†]
土田 賢省 (Kensei TSUCHIDA)[††] 夜久 竹夫 (Takeo YAKU)[†]

[†] 日本大学文理学部応用数学科
Department of Applied Mathematics, Nihon University
{ tomiyama, arita, yaku }@am.chs.nihon-u.ac.jp
[††] 東洋大学工学部情報工学科
Department of Information and Computer Sciences, Toyo University
kensei@eng.toyo.ac.jp

Abstract Tabular forms such as program specification forms [7] are naturally formalized by the attribute graphs [8], in which the attribute denotes locations of items and while the edge labels denotes relations between items. Documents of the tabular forms are represented by graph grammars (e.g., see [8]). Accordingly, a syntactic formalization of document editing provides the foundation for mechanical documentation.

In this paper, we deal with particular graph grammar HNGG [7] for program specification tabular forms. We formalize syntax directed editing methods by extension of the notion of Cornell Program Synthesizer[2] to attribute NCE graph grammars (cf.[4]).

Keywords Graph Grammars, Visual Programming, Software Development, Syntax Directed Editors

1 Introduction

Mechanical editing of tabular forms is one of the important issues in the software engineering methodology. The Cornell Program Synthesizer (CPS) is well-known and is often referred to as a structured and text-based editor which uses an attribute grammar successfully [2]. Tabular forms are represented by several different models (e.g., Pane represents them by [6]). We assigned each item in the tabular form to an attributed node. This assignment naturally represents the order of items and location of items in the tabular form. Since the number of items in the form is generally infinite and the order of items has some valid meaning, tabular forms are denoted by graph grammars [7]. Accordingly, the mechanical editing of tabular forms supposed to be executed by some syntactic editing methods.

In this paper, we consider a programming documentation *Hiform* as an example of the tabular forms. Hiform document is a collection of 17 types of the tabular forms and includes all items defined in the guideline in ISO6592 [3],[7]. Those tabular forms are represented by graphs. Figure 1 illustrates a Hiform form and its corresponding graph. This graph is constructed as follows: (1) A node label of graph shows the type of an item of a tabular form. (2) An edge label shows relations between items. 'lf' denotes the meaning of 'left of'. 'ov' denotes 'over'. 'in' denotes 'whitin'.

A mechanical processing of tabular forms supposed to be realized effectively by syntactic manipulation of graphs. In [7], the inner structure of each form in Hiform is defined by an attribute NCE graph grammar.

The purpose of this paper is to extend CPS mechanisms to graphs using results in [4][7] and to formalize a syntactic editing mechanism for graphs, in this connection some examples are given in consideration with software documentation tabular forms. Definition is made as to insertion in HNGG [8] so that a manipulation is validly executed by the confluence[5] of HNGG.

2 Preliminaries

We review context-free edNCE graph grammars [5], and an attribute NCE graph grammar [7].

2.1 edNCE Graph Grammars [5]

Let Σ be an alphabet of *node labels* and Γ an alphabet of *edge labels*. A *graph* over *alphabets* Σ and Γ is a 3-tuple $H = (V, E, \lambda)$, where V is a finite nonempty *set of nodes*, $E \subseteq \{(v, \gamma, w) \mid v, w \in V, v \neq w, \gamma \in \Gamma\}$ is a *set of edges* and $\lambda : V \to \Sigma$ is a node *labeling function*.

2.1.1 Definition 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

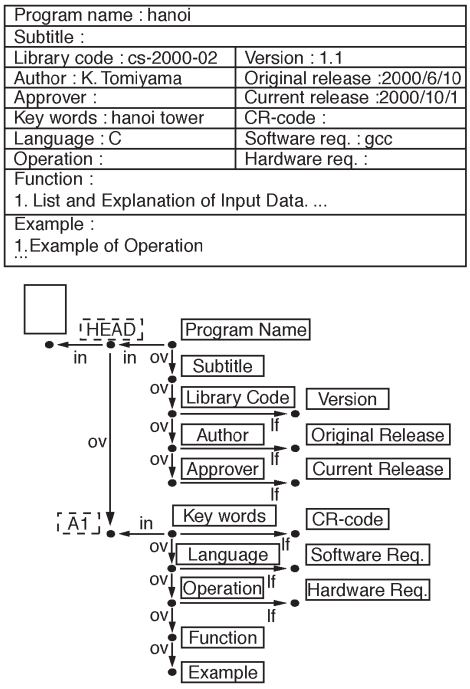

Figure 1. Tabular form in Hiform document and its corresponding graph.

initial nonterminal. A production is denoted by the form $p : X \rightarrow (D, C)$, where $X \in \Sigma - \Delta$, D is a graph over Σ and Γ, and $C \subseteq \Sigma \times \Gamma \times \Gamma \times V_D \times \{in, out\}$ is *connection relation*. □

2.2 Composition of Production Copies [4]

The composite representation of the production copies of an edNCE graph grammar is a theoretical and practical method for representing the graph-rewriting rules for embedding subgraphs of desired structures into a graph.

2.2.1 Definition[4] Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ be an edNCE graph grammar. Let $p_1 : X_1 \rightarrow (D_1, C_1)$ $(D_1 = (V_{D_1}, E_{D_1}, \lambda_{D_1}))$ and $p_2 : X_2 \rightarrow (D_2, C_2)$ $(D_2 = (V_{D_2}, E_{D_2}, \lambda_{D_2}))$ be production copies of G. If $u \in V_{D_1}$, $X_2 = \lambda_{D_1}(u)$, and D_1 and D_2 are disjoint, then a *composite production copy* (with a connection relation) $p : X_1 \rightarrow (D_1, C_1)[u/(D_2, C_2)]$

The composite production copy p composed by p_1 and p_2, and denoted by $p_1 \circ p_2$. □

2.3 Confluence Property [5]

The confluence property guarantees that the result of a derivation shall not depend on the order of the left applications of the production copies. Confluence is a very important property because it guarantees the validity of the left application of the composite production copies. The confluency is also important in the case of developing efficient parsing algorithms.

2.3.1 Definition [5] An edNCE graph grammar $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ is *dynamically confluent* if the following holds for every intermediate graph H generated by G:
if $H \Rightarrow_{u_1,p_1} H_1 \Rightarrow_{u_2,p_2} H_{12}$ and $H \Rightarrow_{u_2,p_2} H_2 \Rightarrow_{u_1,p_1} H_{21}$ $(p_1, p_2 \in P)$ are (creative) derivation of G with $u_1, u_2 \in V_H$ and $u_1 \neq u_2$, then$H_{12} = H_{21}$. □

2.4 Attribute NCE Graph Grammars [7]

We review an attribute graph grammar for the mechanical drawing. An *attribute NCE graph grammar* is as follows.

2.4.1 Definition [7] An attribute NCE Graph Grammar is a 3-tuple $AGG =< G, Att, F >$ where $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ is context-free edNCE graph grammar, called an *underlying graph grammar* of AGG. *Att* is the *set of attributes* of *AGG*. F is *the set of semantic rules of AGG*. □

2.5 HNGG [7] [8] [9]

In this section, we consider an attribute NCE graph grammar. The grammar is called *Hiform Nested tabular form Graph Grammar* (HNGG). **HNGG** $=< G_N, A_N, F_N >$ that generates modular tabular forms called *Hiform form*. Each production has attribute rules for drawing information. The HNGG includes 280 productions and 1248 attribute rules. Figure 2 illustrates a part of productions with attribute rules of HNGG.

3 Editing of Modular Tabular Form

In this section, we deal with a formal definition for editing manipulation using production instance of HNGG, and we also show the validity of our definition using confluency of HNGG.

3.1 Production Instance

We introduce editing manipulations in latter part. The editing manipulations are exactly defined by production instance as follows. In this part, we introduce production instance.

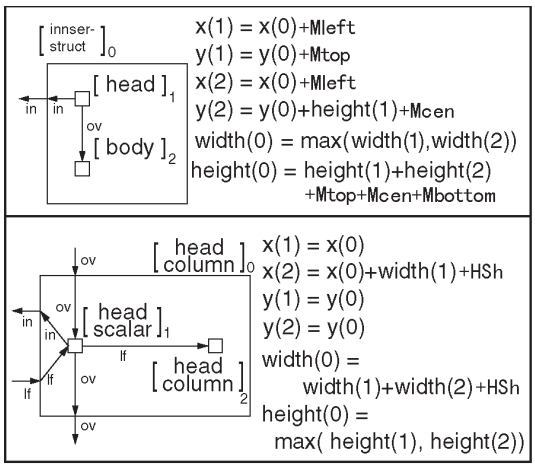

Figure 2. A part of productions of HNGG.

3.1.1 Definition A *production instance* ("instance" for short) is a 3-tuple (ω, p_i, H'_{p_i}), where
1. $\omega \in V_{D_{i-1}}$ is a node removed during the derivation $D_{i-1} \Rightarrow_{p_i} D_i$.
2. $p_i : X_{p_i} \rightarrow (H_{p_i}, C_{p_i}) \in P$ is a production.
3. H'_{p_i} is an embedded graph isomorphic to H_{p_i} during $D_{i-1} \Rightarrow_{p_i} D_i$.

We denote $D_{i-1} \overset{\omega H'_{p_i}}{\Rightarrow_{p_i}} D_i$ if D_{i-1} is directly derived D_i by applying the instance (ω, p_i, H'_{p_i}). □

If there is a production sequence $p = (p_1, \cdots, p_n)$ and instance $(\omega_i, p_i, H'_{p_i})$ for each production p_i $(1 \leq i \leq n)$, an *instance sequence* is a sequence of $((\omega_1, p_1, H'_{p_1}), \cdots, (\omega_n, p_n, H'_{p_n}))$.

3.2 Syntactic Insertion

In this part, we define the syntactic insertion, and denote the flow of insertion manipulation. This manipulation is based on HNGG. Syntax directed editing is executed by sequences of production instances.

3.2.1 Definition For an derivation sequence $D_0 \overset{\omega_1 H'_{p_1}}{\Rightarrow_{p_1}} \cdots \overset{\omega_{i-1} H'_{p_{i-1}}}{\Rightarrow_{p_{i-1}}} D_{i-1} \overset{\omega H'_{p_i}}{\Rightarrow_{p_i}} D_i \overset{\omega_{i+1} H'_{p_{i+1}}}{\Rightarrow_{p+1}} \cdots \overset{\omega_n H'_{p_n}}{\Rightarrow_{p_n}} D_n$ with instance $(p_j : X_{p_j} \rightarrow (H_{p_j}, C_{p_j}), 1 \leq j \leq n)$, we say that q is *insertable* (for p_i) if there is an instance (ω, q, H'_q) $(q : X_q \rightarrow (H_q, C_q) \in P_N)$ such that $D_{i-1} \overset{\omega H'_q}{\Rightarrow_q} Q$ and if there is a derivation sequence $D_{i-1} \overset{\omega H'_q}{\Rightarrow_q} Q \overset{\omega' H'_{p_i}}{\Rightarrow_{p_i}} D'_i \overset{\omega_{i+1} H'_{p_{i+1}}}{\Rightarrow_{p_{i+1}}} \cdots \overset{\omega_n H'_{p_n}}{\Rightarrow_{p_n}} D'_n$ with instance. □

3.2.2 Definition For a production $q : X_q \rightarrow (H_q, C_q) \in P_N$ which is insertable for $p_i : X_{p_i} \rightarrow (H_{p_i}, C_{p_i})$ and $\bigcup_{i=1}^{n} H'_{p_i} \cap H'_q = \phi$, an instance sequence S is obtained by *insertion* of an instance (ω, q, H'_q) into an instance sequence $((\omega_1, p_1, H'_{p_1}), \cdots, (\omega_n, p_n, H'_{p_n})) \overset{def}{\Leftrightarrow}$
$S = ((\omega_1, p_1, H'_{p_1}), \cdots, (\omega_{i-1}, p_{i-1}, H'_{p_{i-1}}), (\omega, q, H'_q), (\omega', p_i, H'_{p_i}), \cdots, (\omega_n, p_n, H'_{p_n}))$
There is the instance sequence S as follows.

1. Trace the derivation sequence with instance D_n back to D_{i-1}.
2. Apply the instance (ω, q, H'_q) to D_{i-1} , and obtain the resultant graph Q.
3. Apply the instance sequence ((ω', p_i, H'_{p_i}), $(\omega_{i+1}, p_{i+1}, H'_{p_{i+1}}), \cdots, (\omega_n, p_n, H'_{p_n})$) to Q, and get the resultant graph D'_n. □

Inserting some instances into an instance sequence bring a new item into existence. That is, they correspond to a manipulation to insert a new item into a permissible place in a Hiform document.

3.2.3 Remark In the same manner as the editing by the instance for a production, we can further define *insertable by composite production copy.* □

3.2.4 Definition A graph H' is *obtained by syntactic insertion* of a graph A at an edge x in a graph H. $\overset{def}{\Leftrightarrow}$
1. A composite production copy q for the graph A and the edge x exists. Furthermore, there exists an instance sequence i_q for q and x.
2. There exists an instance sequence i_H for H. An instance sequence S is obtained by insertion of i_q into an instance sequence i_H.
3. The graph H' is derived by the instance sequence S. □

3.2.5 Proposition Let H be the graph obtained from G by the insertion of nodes a and b at an edge x and an edge y respectively in this order, in HNGG. Let H' be the graph obtained from G by the insertion of nodes b and a at the edge y and x respectively in this order, in HNGG. Then, $H = H'$. □

3.2.6 Proposition Insertion in HNGG is executed in linear time. □

We illustrate an example of an insertion based on HNGG by Figure 3. Here, we consider to insert a form F_2 into a form F_1. Let G_1 be a graph for F_1, and let G_2 be a graph for F_2. Then, a syntactic insertion of G_2 at an edge e in G_1 is done as following:

1. By e and G_2, a composite production copy q for G_2 is obtained. By q and e, An instance sequence i_q is also obtained.

2. An instance sequence i_{G_1} is existed. If q is insertable for p in i_{G_1}, an instance sequence i_G is obtained by insertion of i_q into an instance sequence i_{G_1}.

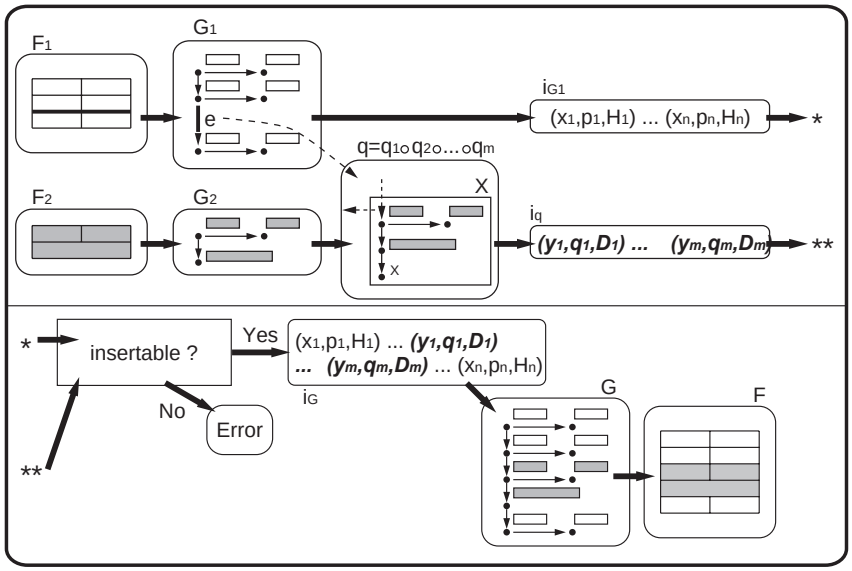

Figure 3. A flow of an insertion process.

3. By this instance sequence i_G, a graph G is obtained. G is a new form which is obtained by inserting F_2 into F_1.

4 Conclusion

We proposed editing method for tabular forms, based on attribute NCE graph grammar of tabular forms with a homogenous cell size. Our editing method also includes attribute rules for mechanical drawing. By using this editing method, we can exactly edit valid tabular forms defined by edNCE graph grammar. Linear time editing algorithm with attribute rules for primitive drawing exists.

These syntactic editing methods could be applied to syntactic manipulation of spreadsheet languages. We are reconstructing attribute rules for more sophisticated drawing. We are investigating other edit manipulations that are a division manipulation, a combination manipulation and so on. Furthermore we are now developing a tabular form editor system by using this approach.

Acknowledgment

We thank Mr. S. Kanai's advice in the course of preparing the manuscript. We also thank to Mr. S. Nakagawa and Mr. K. Ruise for valuable discussions.

References

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

[2] Tim Teitelbaum and Thomas Reps, The Cornell Program Synthesizer: A Syntax-Directed Programming Environment, *Comm. ACM*, Vol.24, 563-573, (1981).

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

[4] Y.Adachi, K.Anzai, et al. Hierarchical Program Diagram Editor Based on Attribute Graph Grammar, *Proc. COMPSAC*96, 205-213, (1996).

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

[6] John F. Pane, Brad A. Myers, Tabular and Textual Methods for Selecting Objects from a Group, *Proc. 2000 IEEE Symp. on Visual Language*, 157-164, (2000).

[7] T. Arita, K. Tomiyama, T. Yaku, Y. Miyadera, K. Sugita, K. Tsuchida, Syntactic Processing of Diagrams by Graph Grammars, *Proc. IFIP WCC ICS 2000*,145-151, (2000).

[8] T. Arita, K. Sugita, K. Tsuchida, T. Yaku, Syntactic Tabular Form Processing by Precedence Attribute Graph Grammars, *Proc. IASTED Applied Informatics 2001*,637-642, (2001).

[9] T. Arita et al., A Precedence Attribute NCE Graph Grammar for Hiform, Http://www.hichart.org/keyaki/archive/HC00-001