*Electronic Notes in Theoretical Computer Science 50 No. 3 (2001) { Proc. GT-VMT 2001 URL:* [*http://www.elsevier.nl/locate/entcs/volume50.html*](http://www.elsevier.nl/locate/entcs/volume50.html) *7 pages*

*Application of Attribute NCE* Graph Grammars to Syntactic Editing

*of Tabular Forms*

# *Tomokazu ARITA a;1 Kiyonobu TOMIYAMA a;2* Kensei TSUCHIDA b;3 Takeo YAKU a;4

*a Dept. Comput. Sci. and System Analysis, Nihon University 3{25{40, Sakurajosui, Setagaya, Tokyo, 156-8550, Japan*

*b Dept. Inf. and Comp. Sci., Toyo University 2100, Kujirai, Kawagoe, Saitama, 350{8585, Japan*

*Abstract*

*In this paper, we deal with editing tabular forms for program speci cations based on a particular graph grammar HNGG [2]. First, we formalize syntax-directed editing methods by extending of the notion of the Cornell Program Synthesizer [8] to attribute NCE graph grammars (cf. [1]). Next, we discuss the algorithms of the editing methods.*

*Key words: Graph Grammars, Visual Programming, Software Development, Syntax-Directed Editors*

# *1 Introduction*

*Mechanical editing of tabular forms is one of the important issues in software* engineering methodology. The Cornell Program Synthesizer is well-known and is often referred to as a structured and text-based editor which uses an at- tribute grammar successfully [8]. Tabular forms are represented by several di erent models (e.g., Pane [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 unbound and the order of items has some valid meaning, tab- ular forms are denoted by a graph grammar [2]. Accordingly, the mechanical

*1 Email:* [*arita@cssa.chs.nihon-u.ac.jp*](mailto:arita@cssa.chs.nihon-u.ac.jp)

*2 Email:* [*tomiyama@cssa.chs.nihon-u.ac.jp*](mailto:tomiyama@cssa.chs.nihon-u.ac.jp)

*3 Email:* [*kensei@eng.toyo.ac.jp*](mailto:kensei@eng.toyo.ac.jp)

*4 Email:* [*yaku@cssa.chs.nihon-u.ac.jp*](mailto:yaku@cssa.chs.nihon-u.ac.jp)

*c 2001 Published by Elsevier Science B. V.* Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

|  |  |
| --- | --- |
| Program Name : hanoi | |
| Subtitle : | |
| Library Code : cs-2001-02 | Version : 1.1 |
| Author : K. Tomiyama | Original Release : 2000/06/10 |
| Approver : | Current Release : 2000/10/01 |
| Keywords : hanoi tower | CR-Code : |
| Language : C | Software Req. : gcc |
| Operation : | Hardware Req. : |
| Function : | |
| Example : | |

Hardware Req.

*Fig. 1. Tabular form in a Hiform document and its corresponding graph.*

in

in ov Subtitle ov

ov Author ov Approver

Keywords

in ov

lf lf lf

lf

ov Operation lf

ov

lf

Function

ov

Example

Software Req.

Language

CR-Code

A1

Current Release

Original Release

Version

Library Code

Program Name

HEAD

*editing of tabular forms is supposed to be executed by some syntactic editing* methods.

*In this paper, we consider a programming documentation, Hiform, as an* example of tabular forms. Hiform document is a collection of 17 types of tabular forms and includes all items de ned in the ISO6592 guideline [9],[2]. It should be noted that certain ISO6592 tabular forms are regarded as tabular forms, since they have modular structures. Such tabular forms are represented by graphs. Fig. 1 illustrates a Hiform form and its corresponding graph. This graph is constructed as follows: (1) A node label of the 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 the meaning of `over', and*

*`in' denotes the meaning of `within'.*

*It is supposed that a mechanical processing of tabular forms can be realized* e ectively by syntactic manipulation of graphs. In [2], the inner structure of each form in Hiform is de ned by an attribute NCE graph grammar. Hiform forms are speci ed by graphs that are generated its grammar.

*The purpose of this paper is to extend Cornell Program Synthesizer mech-* anisms to graphs using the results reported in [1] and [2] and to formalize a syntactic editing mechanism for graphs. Insertion in HNGG [3] is de ned so that this manipulation is validly executed by the con uence [7] of HNGG.

*In Section 2, preliminary de nitions are given. In Section 3, a formal* de nition for editing mechanisms is given by using instance sequences [1]. And we also show the validity of our de nition by using the con uence of HNGG. Section 4 is devoted to our concluding remarks.

# *2 Preliminaries*

*2.1 edNCE Graph Grammars [7]*

*Let be an alphabet of node labels and be an alphabet of edge labels. A* graph over alphabets and is a 3-tuple H = (V; E; ), where V is a nite nonempty set of nodes, E f(v; ; w) j v; w 2 V; v 6= w; 2 g is a set of edges, and : V ! is a node labeling function.

*De nition 2.1 An edNCE graph grammar is a 6-tuple G = ( ; ; ; ; P; S),* where is the alphabet of node labels, is the alphabet of terminal node labels, is the alphabet of edge labels, is the alphabet of nal edge labels, P is the nite set of productions, and S 2 is the initial nonterminal. A production is denoted by the form p : X ! (D; C), where

*X 2 , D is a graph over and , and C VD fin; outg*

*is the connection relation.* *2*

*2.2 Composition of Production Copies [1]*

*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.

*De nition 2.2 [1] Let G = ( , , , , P , S ) be an edNCE graph grammar. Let p1 : X1 ! (D1, C1) (D1 = (VD1 , ED1 , D1 )) and p2 : X2 ! ( D2, C2*

*) (D2 = (VD2 , ED2 , D2 )) be production copies of G. If u 2 VD1 and X2 =*

*D1 (u), and D1 and D2 are disjoint, then a composite production copy (with* a connection relation) p : X1 ! (D; C) is de ned as follows: D is a graph as VD = fVD1 -fugg [ VD2 about nodes. C = f ( , = , !, d) 2 C1 j ! 2 VD1 - fug g [ f ( , = Æ, y, d) j 9 2 , ( , = , u, d ) 2 C1, ( , = Æ, y, d) 2 C2 g The composite production copy p composed by p1 and p2, and denoted

*by p1 Æ p2.* *2*

*2.3 Con uence Property [7]*

*The con uence property guarantees that the result of a derivation shall not* depend on the order of the applications [7] of the production. Con uence is a very important property because it guarantees the validity of the application of the composite production copies. The con uence is also important when developing eÆcient parsing algorithms.

*De nition 2.3 [7] An edNCE graph grammar G = ( ; ; ; ; P; S) is dynamically con uent if the following holds for every intermediate graph H generated by G: if H )u1;p1 H1 )u2;p2 H12 and H )u2;p2 H2 )u1;p1 H21 (p1; p2 2 P ) are derivations of G with u1; u2 2 VH and u1 6= u2, then H12 = H21.* *2*

*2.4 Attribute NCE Graph Grammars [2]*

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

*De nition 2.4 [2] An attribute NCE Graph Grammar is a 3-tuple AGG*

*=< G; Att; F > where G = ( ; ; ; ; P; S) is a context-free edNCE graph* grammar, called an underlying graph grammar of AGG. Here Att is the set of attributes of AGG, and F is the set of semantic rules of AGG. *2*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P2 | inner- | Semantic Rule | Ph5 | head  ov [ column ] | Semantic Rule |
| [ struct ]0  in in [ head ]1 ov  [ body ]2 | | x(1) = x(0)  y(1) = y(0)  x(2) = x(0)  y(2) = y(0)+height(1) width(0)  =max(width(1),width(2)) height(0)  =height(1)+height(2) | 0  ov  in head  in [scalar]1  lf head  lf lf [ column ]  ov 2  ov | | x(1) = x(0)  y(1) = y(0)  x(2) = x(0)+width(1) y(2) = y(0)  width(0)  =width(1)+width(2) height(0)  =max(height(1), height(2)) |

*Fig. 2. Part of the productions of HNGG.*

in  [ head ]

P2 ov



[ inner- ]

struct

in

 [ body ]

*Fig. 3. An example of applying a production P2.*

*2.5 HNGG [2] [3] [4]*

*We review an attribute NCE graph grammar for tabular forms. The grammar* is called a Hiform Nested tabular form Graph Grammar (HNGG). HNGG =

*< GN ; AN ; FN > that generates modular tabular forms called Hiform form* where GN = ( N ; N ; N ; N ; PN ; SN ) is the underling edNCE graph gram- mar. Each production has a semantic rule for drawing information. The HNGG includes 280 productions and 1248 attribute rules. Fig. 2 illustrates a part of the productions with attribute rules of HNGG. We write productions like style of the edNCE graph grammar. Fig. 3 is an example of applying a production P2. By applying a production P2, a node labeled \inner{struct" is replaced to a graph of right hand side of P2.

# *3 Editing of Modular Tabular Forms*

*In this section, we present a formal de nition for editing manipulation by* using production instances of HNGG, and we also show the validity of our de nition by using con uence of HNGG.

*3.1 Production Instance*

*The editing manipulations are de ned by production instance as follows.*

*De nition 3.1 A production instance (\instance" for short) is a 3-tuple (!,* pi, Hpi ), where (1) ! 2 VDi 1 is a node removed during the derivation Di 1 )pi Di, (2) pi : Xpi ! (Hpi ; Cpi ) 2 P is a production, and (3) Hpi is an embedded graph isomorphic to Hpi during Di 1 )pi Di.

*We denote Di 1*

*!Hpi*

*)pi*

*Di if Di*

*is directly derived from D*

*i 1*

*by applying*

*the instance (!; pi; Hpi ).* *2*

F1

G1

e

iG1

(x1,p1,H1) ... (xn,pn,Hn)

\*

q=q1 q2 ... qm

F2

G2

X

iq

***(y1,q1,D1) ... (ym,qm,Dm)***

\*\*

X

\*

Yes

(x1,p1,H1) ... ***(y1,q1,D1)***

***... (ym,qm,Dm)*** ... (xn,pn,Hn)

iG

G

No

F

Error

\*\*

insertable ?

*Fig. 4. Flow of an insertion process.*

*If there is a production sequence p = (p1; ; pn) and instance (!i; pi; Hpi )* for each production pi (1 i n), an instance sequence is a sequence of ((!1; p1; Hp1 ); ; (!n; pn; Hpn )).

*A derivation of an edNCE graph grammar is represented by a derivation* tree. However, a derivation sequence is convenient as a representation of a derivation for a prosessing model. Therefore, we represent a derivation with a production sequence.

*3.2 Syntactic Insertion*

*In this section, we de ne the syntactic insertion. This manipulation is based* on HNGG. Syntax-directed editing is de ned by using instance sequences.

*De nition* *3.2*

*For a derivation sequence D0*

*!1Hp1*

*)p1*

*!i 1 Hpi 1*

*)pi 1 D*

*!Hpi*

*i 1 )pi Di*

*!i+1Hpi+1*

*!n Hpn*

*)pi+1 )pn Dn with instance (pj : Xpj ! (Hpj ; Cpj ); 1 j n), we*

*assume that q is insertable (for pi) if there is an instance (!; q; Hq) (q : Xq !*

*!Hq*

*(Hq; Cq) 2 PN ) such that Di 1 )q Q and if there is a derivation sequence*

*!H !0 Hp !i+1Hp ! H*

*Di 1 )q Q pi*

*q*

*D*

*n*

*D*

*)*

*i*

*)*

*pn*

*n*

*0 i+1*

*i pi+1*

*)pn*

*0 where !0 is a node in Q, a node*

*label of !0 is left hand side of p .* *2*

*i*

*De nition 3.3 For a production q : Xq ! (Hq; Cq) 2 PN , which is insertable*

*for pi*

*i=1*

*: Xpi*

*! (Hpi*

*; Cpi*

*) and Sn*

*Hpi*

*\ Hq*

*= , an instance sequence S*

*is obtained by insertion of an instance (!; q; Hq) into an instance sequence*

*def*

*((!1; p1; Hp1 ), , (!n; pn; Hpn )) , S = ((!1; p1; Hp1 ), , (!i 1; pi 1; Hpi 1 ),*

*(!; q; Hq), (!; pi; Hpi ), , (!n; pn; Hpn )). The instance sequence S is given*

*as follows. (1) Trace the derivation sequence with instance Dn back to Di 1.*

*(2) Apply the instance (!; q; Hq) to Di 1, and obtain the resultant graph Q.*

*(3) Apply the instance sequence ( (!0 ;p ; H*

*i*

*n*

*pi*

*), (!*

*i+1*

*; pi+1*

*; Hpi+1*

*), , (!n*

*; pn;*

*Hpn*

*) ) to Q, and get the resultant graph D0 .* *2*

*Inserting some instances into an instance sequence brings 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.

*Remark 3.4 In the same manner as the editing by using the instance of a* production, we can further de ne insertable by composite production copy. 2

*De nition 3.5 A graph H0 is obtained by syntactic insertion of a graph A* at an edge x in a graph H, if the following conditions hold: (1) A composite production copy q for graph A and edge x exists. (2) There exists an instance sequence iq for q and an instance sequence iH for H. An instance sequence S is obtained by insertion of i into an instance sequence i . (3) The graph H0 is derived from instance sequence S. *2*

*q H*

*Proposition 3.6 Let H be the graph obtained from G by the insertion of* graph a and graph b at edge x and edge y respectively in this order in HNGG. Let H0 be the graph obtained from G by the insertion of b and a at y and *x*

*respectively in this order in HNGG. Then, H = H0 .*

*Proof. HNGG has a con uence property. Thus, the proposition is veri ed.2* Proposition 3.7 Insertion in HNGG is executed in linear time.

*Proof. An insert point of a production instance is found in linear time for* the length of an embedded production sequence. Let n be the number of nodes in a target graph. In a derivation of our HNGG, any node in a target graph is changing to a terminal node by at most ve application. Therefore, the maximum length of an instance sequence for the target graph is 5n. Since HNGG is a precedence edNCE graph grammar [2], syntax analysis is executed in linear time [5]. Attribute evaluation is also executed in linear time [2]. 2

*An example of an insertion is given in Fig. 4. Here, we insert a form F2* into a form F1. Let G1 be a graph for F1, and let G2 be a graph for F2. Then, a syntactic insertion of G2 at edge e in G1 is done as follows: (1) A composite production copy q for G2 is obtained from e and G2. (2) An instance sequence iq for q and an instance sequence iG1 existed. If q is insertable for p in iG1 , an instance sequence iG is obtained by insertion of iq into an instance sequence iG1 . (3) A graph G is generated from this instance sequence iG. G is a new form which is obtained by inserting F2 into F1.

# *4 Conclusion*

*Spread sheets and software documents are used for software visualization* widely. Our results are a proposal of a theoretical model such visualization.

*We proposed an editing method, based on attribute NCE graph grammar of* tabular forms with a homogenous cell size. This method includes attribute rules for mechanical drawing. It can exactly edit valid tabular forms de ned by edNCE graph grammar. A linear time editing algorithm with attribute rules for primitive drawing exists.

*These syntactic editing methods could be applied to syntactic manipula-* tion of spreadsheet languages. We are reconstructing attribute rules for more sophisticated drawing. We are investigating other edit manipulations such as a division manipulation, a combination manipulation and so on. Furthermore we are now developing a tabular-form editor system.

# *Acknowledgement*

*We thanks Prof. K. Sugita for his valuable suggestions. We thank Mr. S.* Kanai for his advice in the course of preparing the manuscript. We also thank Mr. S. Nakagawa and Mr. K. Ruise for their valuable discussions.

# *References*

*[1] Adachi, Y., K.Anzai, et al. Hierarchical Program Diagram Editor Based on Attribute Graph Grammar, Proc. COMPSAC96 (1996), 205-213.*

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

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

*[4] Arita, T., A Precedence Attribute NCE Graph Grammar for Hiform, (2000), URL:* [*http://www.hichart.org/*](http://www.hichart.org/) *or* [*http://www.cssa.chs.nihon-u.ac.jp/*](http://www.cssa.chs.nihon-u.ac.jp/)

*~yaku/keyaki/archive/HC00-001*

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

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

*[7] Rozenberg, Grzegorz(Ed.), \Handbook of Graph Grammar and Computing by Graph Transformation", World Scienti c Publishing, (1997).*

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

*[9] ISO 6592{1985, Guidelines for the Documentation of Computer{Based Application Systems, (1985).*