An XML Viewer for Tabular Forms for use with Mechanical Documentation

Osamu Inoue and Kensei Tsuchida
Dept. Information & Computer Sciences
Toyo University
2100, Kujirai, Kawagoe
Saitama, 350-8585, Japan
email: inoue@tsu.cs.toyo.ac.jp, kensei@eng.toyo.ac.jp

Shun-ichi Nakagawa, Tomokazu Arita and Takeo Yaku
Dept. Computer Science & System Analysis
Nihon University
3-25-40, Sakurajosui, Setagaya
Tokyo, 156-8550, Japan
email: {nakagawa, arita, yaku}@am.chs.nihon-u.ac.jp

ABSTRACT
We deal with mechanical documentation in software development tools. First, we review tabular forms for program specification and their formal syntax by an attribute edNCE graph grammar. Next we explain a parser based on the syntactic definitions and attribute rules. Furthermore, we introduce an XML viewer for tabular forms based on the attribute graph grammar. Finally, we introduce a system structure to construct the whole processing system for mechanical documentation. These results are applied to mechanical manipulation of general tabular forms.

KEY WORDS
software documentation, XML, visualization, attribute graph grammars, tabular forms

1. INTRODUCTION

Mechanical documentation such as automatic drawing and editing of program specification forms is one of the important problems in software development tools. Software documentation includes tabular forms, such as program specification forms, and diagrams, such as program flowcharts. Furthermore, the tabular forms may be classified into: (1) nested-structured forms in which items are linked hierarchically to one another; and (2) tessellation-structured forms such as symbol tables and spread sheets. This paper deals with (1) (that is, the nested-structured tabular forms) together with their mechanical manipulating problems.

In mechanical manipulation of tabular forms, it is necessary to explicitly define both the syntax and drawing conditions (cf. [3]). Attribute graph grammars formulate syntactic structure. They also formulate visual structure among items in forms, universally by syntax with attribute rewriting rules. Franck [2] introduced precedence graph grammars and applied them to nested program tabular forms called PLAN2D. We formulated the hierarchically structured program tabular form [5, 8].

In 2000, we introduced part of a syntactic definition of program specification forms based on ISO6592 standard [7]. It is to be noted that those forms are represented by attribute marked graphs. For this purpose, we employed graph grammars. In this paper, we propose a universal processing system for tabular forms. Accordingly, we employ this system as the common models attribute NCE graph grammars.

On the other hand, it is to be noted that XML is widely recognized as one of the most influential standards concerning data exchange and Web-presentations today. Since XML is platform-independent [1], software documents expressed in XML are viewed and displayed on any readily available Web browsers. Thus, documents expressed in the form of XML can be shared with many other users, regardless of their computer environments. Here, we enhanced our system in such a manner as to be able to generate XML source codes of tabular forms by using the attribute edNCE graph grammar as well as to check their grammatical correctness.

In Section 2, we review tabular forms for program specification and a formal syntax of those forms by attribute edNCE graph grammars. In Section 3, we also review a parser for tabular forms, which provides a mechanical verifier. In Section 4, we introduce an XML viewer for tabular forms based on the attribute graph grammar. In Section 5, we propose a syntactic processing system using above methods. Section 6 provides concluding remarks.

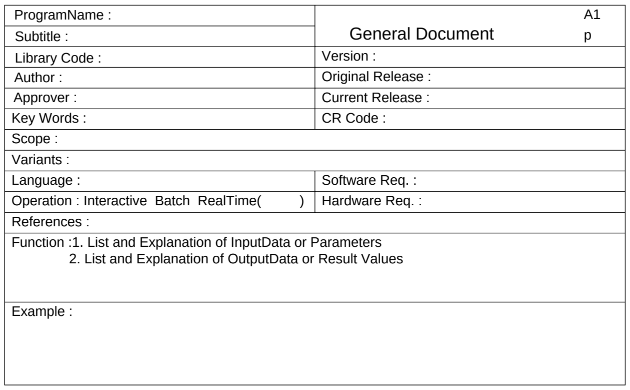
2. TABULAR FORMS AND THEIR SYNTAX

In this section, we review the tabular forms for program specification, by using some examples.

2.1 Tabular Forms for Program Specification

We consider here a program specification language called Hiform [7] based on ISO6592 [4].

The International Organization for Standardization is-



ProgramName :	General Document A1
Subtitle :	p
Library Code :	Version :
Author :	Original Release :
Approver :	Current Release :
Key Words :	CR Code :
Scope :	
Variants :	
Language :	Software Req. :
Operation : Interactive Batch RealTime( )	Hardware Req. :
References :	
Function :1. List and Explanation of InputData or Parameters 2. List and Explanation of OutputData or Result Values	
Example :	

Figure 1. Hiform General Document form

sued a guideline in ISO6592 and described all items in program documentation in Annexes A, B and C. We considered the ISO6592 items and introduced Hiform, which includes all items defined in these Annexes. Hiform [7] is defined by 17 types of forms. Figure 1 shows a Hiform General Document form.

The order among tabular forms was already defined by a context-free string grammar [7]. The order and graphical structure of cells inside tabular forms will be discussed later.

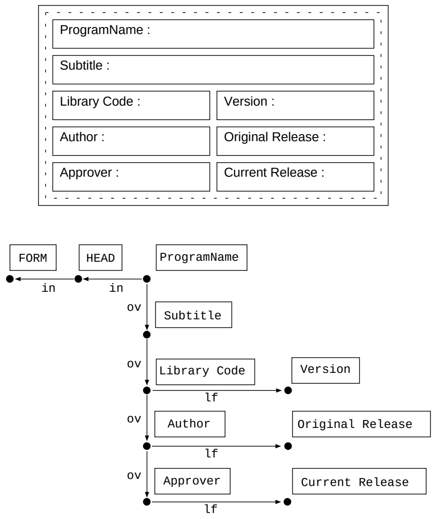


Figure 2. Nested tabular form and its corresponding marked graph

Hiform is characterized by graph grammars for graph syntax and attribute rules for drawing conditions. In the graph grammar of Hiform, a program form is represented by a marked graph with locations. We illustrate examples in Figure 2.

2.2 edNCE Graph Grammars [6]

The following provides a formal model of tabular forms.

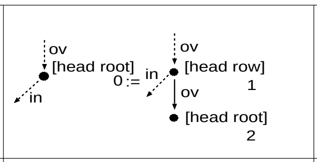
Definition An *edNCE graph grammar* is a 6-tuple $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$, where Σ is the *alphabet of node labels*, $\Delta \subseteq \Sigma$ denotes the alphabet of *terminal* node labels, Γ is the alphabet of *edge* labels, $\Omega \subseteq \Gamma$ is the alphabet of *final* edge labels, P is the finite set of *productions* and $S \in \Sigma - \Delta$ is the *initial nonterminal*. A production is of the form $X \to (D, C)$ 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. □

The type of *Nested* tabular forms (see Figure 2) plays an important role in software documentation. We characterize it by the following *Hiform Nested Graph Grammar* (HNGG).

2.3 HNGG [9, 10]

In this part, we consider an attribute edNCE graph grammar which characterizes the Hiform documents. The grammar is called *Hiform Nested Graph Grammar* (HNGG). We consider an *attribute edNCE* graph grammar **HNGG** $= < G_N, A_N, F_N >$ that generates nested tabular forms in Hiform *form*. *Underlying* graph grammar $G_N = (\Sigma_N, \Delta_N, \Gamma_N, \Omega_N, P_N, S_N)$ is an edNCE *context-free* graph grammar.

The following Figure 3 illustrates productions of HNGG. Each production has *attribute rules* for drawing information and generating XML source codes. The HNGG includes 280 productions and 1528 attribute rules for the definition of nested graph part such as the program form.

H2	 ov
	[head root]
	in
	0 :=
	in
	ov
	[head row]
	1
	ov
	[head root]
	2
		x(1) = x(0) x(2) = x(0)
		y(1) = y(0) y(2) = y(0) + height(1)
		width(0) = max(width(1) , width(2))
		height(0) = height(1) + height(2)
		S_{XML} (0) = S_{XML} (1)· S_{XML}(2)

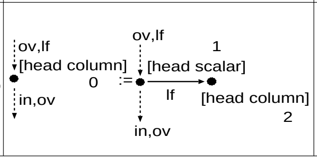
H5	 ov,lf
	[head column]
	0
	in,ov
	:=
	ov,lf
	1
	[head scalar]
	lf
	[head column]
	2
	in,ov
		x(1) = x(0) x(2) = x(0) + width(1)
		y(1) = y(0) y(2) = y(0)
		width(0) = width(1) + width(2)
		height(0) = max(height(1) , height(2))
		S_{XML} (0) = S_{XML} (1)· S_{XML}(2) ·</node>

Figure 3. Productions of HNGG

3. PARSING OF HIFORM [13]

3.1 Precedence Graph Grammar [2, 10]

We have modified the Franck's precedence relation [2] on Franck's context-free graph grammar and we hereby propose precedence relations on edNCE graph grammars for efficient parsing. Let *G* be an edNCE grammar. Then, precedence relations $\doteq$, $<\cdot$, $\cdot>$ and $<\cdot>$ are determined by a connection relation of each production of G .
We construct 5376 relations over the marks in HNGG as shown in Figure 4. It is shown that the relations are pairwise disjoint. HNGG syntax has the *precedence* property [2]. Then we can obtain the following property.

Property Grammar HNGG is a precedence graph grammar [2]. Thus, the parsing algorithm of Hiform is given by the Franck's linear time parsing algorithm [9].

Right / Left	[head scalar]			[head column]			[head row]			[head root]		
	in	ov	lf	in	ov	lf	in	ov	lf	in	ov	lf
[head scalar]		$<\cdot>$	$<\cdot$		$<\cdot>$	$\doteq$		$<\cdot>$			$<\cdot>$	
[head column]		$<\cdot>$			$<\cdot>$			$<\cdot>$			$<\cdot>$	
[head row]		$<\cdot$			$<\cdot$			$<\cdot$			$\doteq$	

Figure 4. A part of Precedence Realation of HNGG

3.2 Parsing Algorithm [2, 10, 13]

We describe here an outline of parsing algorithm (cf. [2, 10, 13]).
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. The algorithm 'reduce' principally consists of the following 3 steps.

1. It seeks for a handle in the graph inputted by an algorithm 'seekForHandle'. A handle is found using the precedence table.

2. The handle is replaced by a new node whose mark is a left hand side of production for this handle, by calling an algorithm 'replaceHandleToLeftHandSideOfProduction'.

3. 'reduce' returns a graph replaced by a handle.

The algorithm 'replaceHandleToLeftHandSideOfProduction' principally consists of the following 4 steps.

1. The handle is removed from the graph by calling 'moveHandleFromGraph'.

2. A production for the handle is searched by calling 'searchProduction'.

3. Derivation subtree is made by the handle and the production by calling 'makeDerivationTree'.

4. A new node's mark is the left hand side of the production.

4. ATTRIBUTE EVALUATION AND XML VIEWER

Layout problems can be solved by evaluation of attributes [13]. We use attributes x, y, $width$ and $height$. In addition, we introduce new attribute S_{XML} which contains XML source codes, as its value and representation corresponding to the tabular forms. The attribute S_{XML} is defined by using a concatenation operator '$\cdot$' and is computed by referring to other attributes. The XML source codes are generated by evaluating S_{XML}. Evaluation of attributes is performed in the bottom-up manner. We illustrate a process of evaluating the attributes in Figure 6.

First, by parsing for a marked graph, if the syntax of the marked graph is correct, then the derivation tree is generated. Next, by evaluating layout attributes for the derivation tree, the derivation tree with layout information is obtained. Then XML Viewer takes the derivation tree as an input and computes corresponding XML source codes by evaluating attributes of the derivation tree. Finally the viewer displays the tabular forms by referring to the XSL file pre-defined and using popular Web browser(e. g., IE). The number of attribute rules concerning the XML is 280. Figure 5 shows a processing flow of the parser and XML viewer, and Figure 6 is a derivation tree with XML source codes.

5. SYSTEM OVERVIEW

This system is constructed on a graph parsing engine for Hiform. This engine is comprised of three parts, which are (1) productions for tabular form syntax, (2) attribute rules for calculating values of tabular form's layout information and (3) a precedence table for tabular form parsing.

Figure 7 illustrates the System Overview, and the Table 1 shows the features of the system. An execution screen of the XML viewer is shown in Figure 8 in the Appendix.

Editor		under development
Drawer	3k	lines
Parsing Engine	2k	lines
Production	280	rules
Attribute Tree	1528	rules
Precedence Table	5376	relations

Table 1. Feature of our system

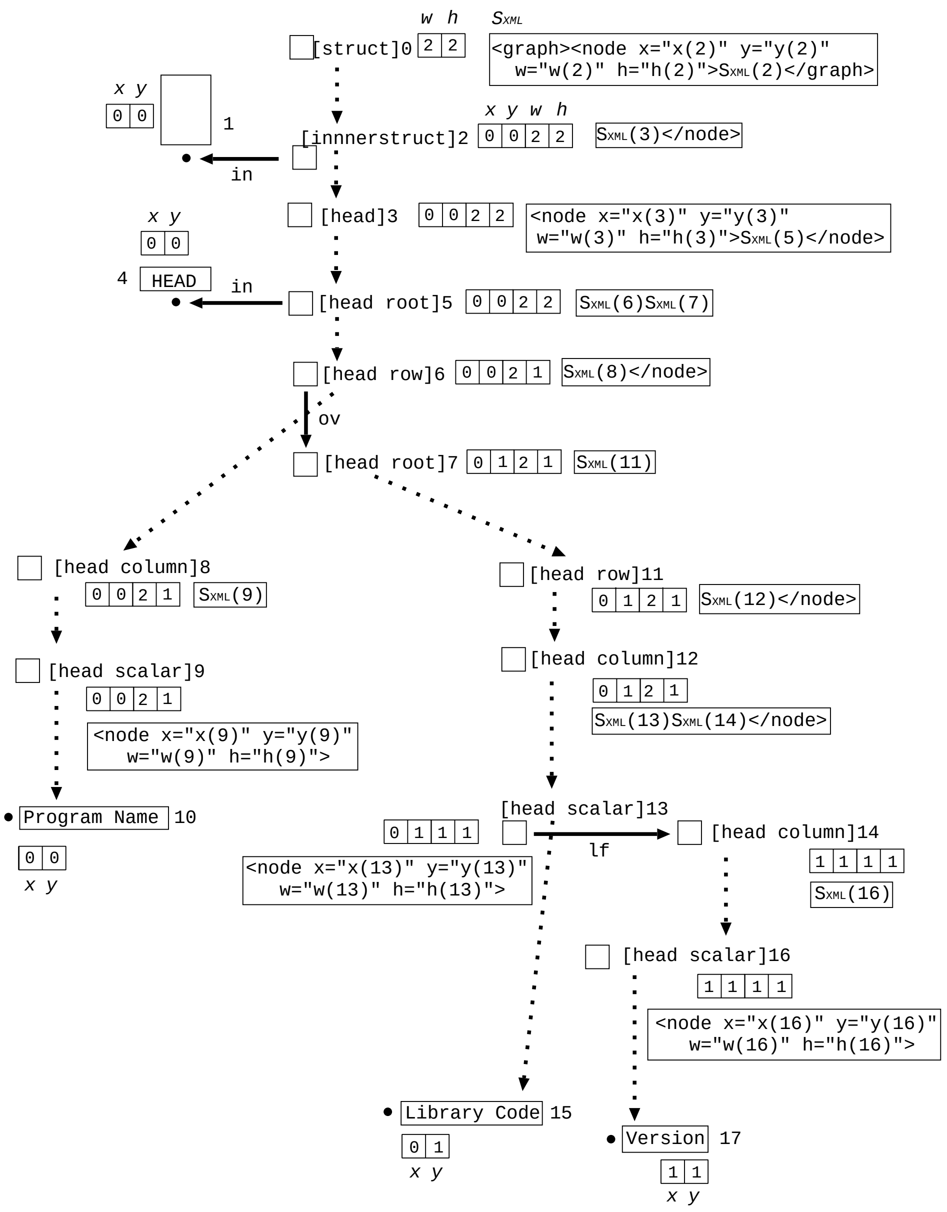


Figure 6. Derivation Tree with XML source codes

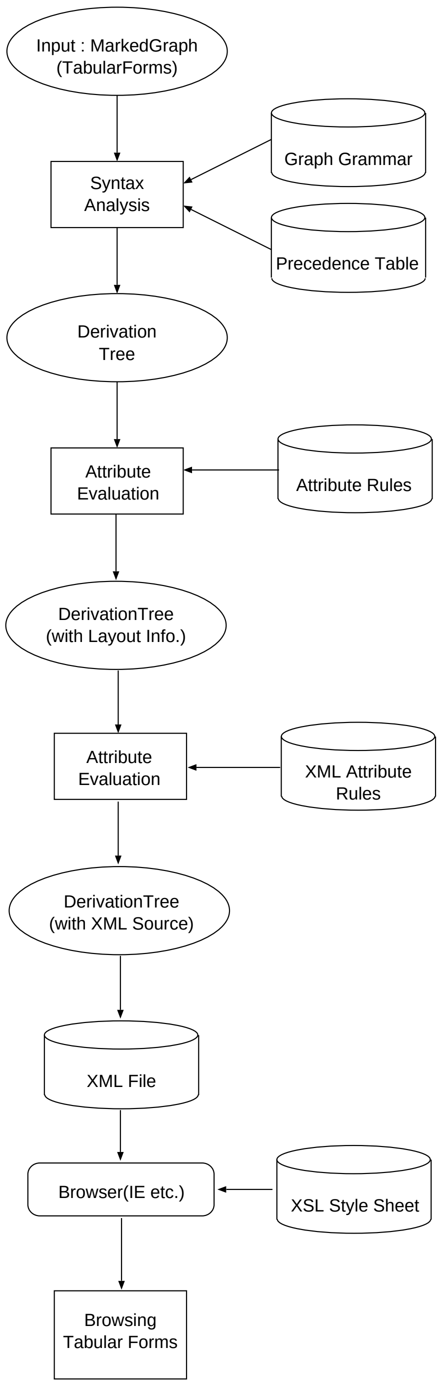


Figure 5. Processing Flow of the parser and XML viewer

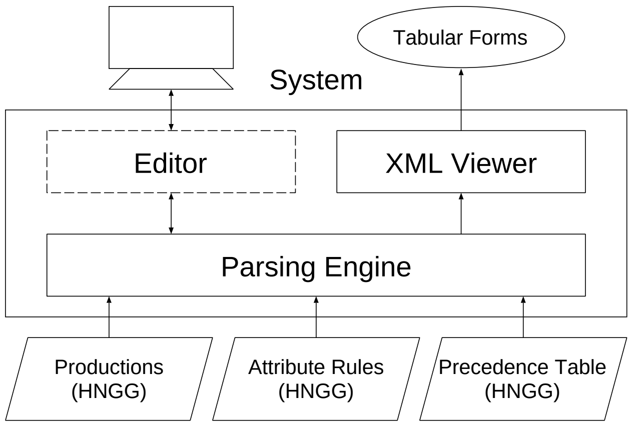


Figure 7. System Overview

6. RELATED WORKS

Graph manipulating systems such as graph editors and graph drawing systems using combinatorial and constraint algorithms have been developed[6]. Along with the development of the graph grammar theory, syntactic graph manipulating systems such as DIAGEN were also developed. Among them, several large projects such as APPLIGRAPH have been also developed. Nagl et al. introduced IPSEN systems.

7. CONCLUSIONS

We defined an attribute edNCE graph grammar with 280 productions and 1528 attribute rules for ISO 6592 documentation with 137 items. We proposed syntactic tabular form designing environment (cf. [9, 11, 12, 14]), based on the attribute edNCE graph grammar. Then we developed XML viewer for tabular forms, which generates XML source codes based on the grammar and displays tabular forms with IE. Our whole system takes a marked graph as input, verifies the syntax of it and displays it on a common use Web browser. We are now developing a Syntax Editor.

ACKNOWLEDGEMENTS

We thank Mr. Kanai for his advise in the course of preparing the manuscript. We also thank Prof. Sugita for valuable discussions.

References

[1] Extensible Markup Language (XML), The World Wide Web Consortium (W3C), http://www.w3.org/XML

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

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

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

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

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

[7] 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, 74-79 (1998).

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

[9] 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).

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

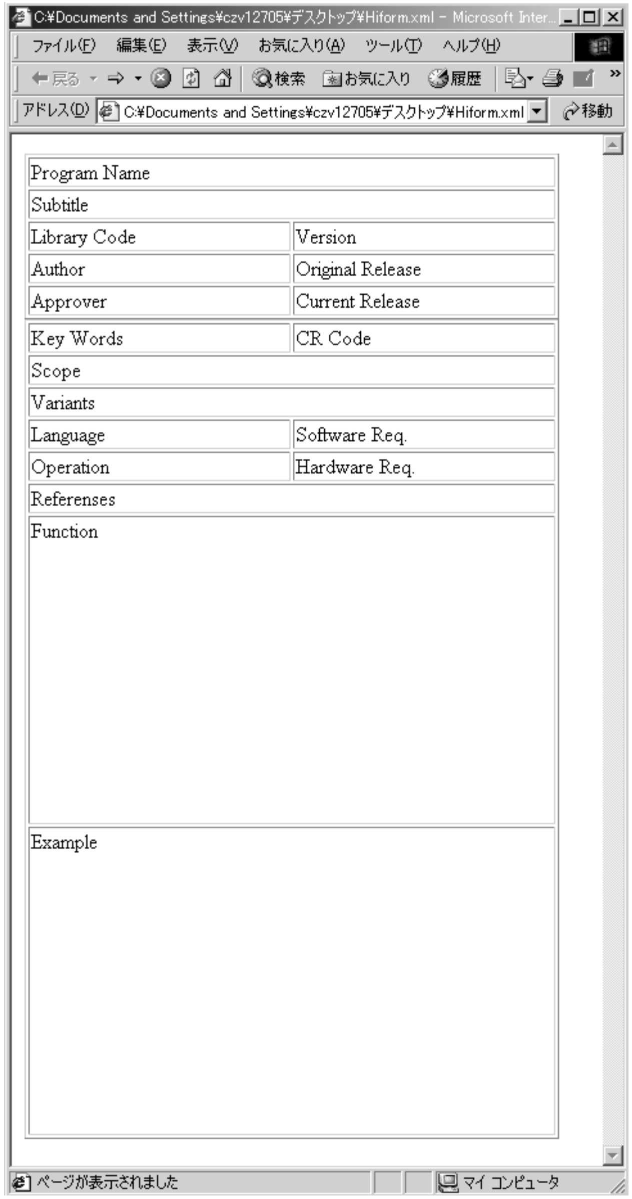
[11] K. Tomiyama et al., Syntactic Editing for Nested Tabular Forms,
http://www.hichart.org/keyaki/archive/HC00-002

[12] T. Arita et al., A Context-Sensitive Attribute NCE Graph Grammar for Tabular Form,
http://www.hichart.org/keyaki/archive/HC00-003

[13] T. Arita, K. Sugita, K. Tsuchida and T. Yaku, Syntactic Tabular Form Processing By Precedence Attribute Graph Grammar, *Proc. IASTED AI 2001,pp.637-642 (2001)*

[14] T. Arita, K. Tomiyama, K. Tsuchida and T. Yaku, Application of Attribute NCE Graph Grammars to Syntactic Editing of Tabular Forms, Electric Notes in Theoretical Computer Science, Vol. 50, 3, (2001).

APPENDIX


C:¥Documents and Settings¥czv12705¥デスクトップ¥Hiform.xml - Microsoft Inter...
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 検索 お気に入り 履歴
アドレス(D) C:¥Documents and Settings¥czv12705¥デスクトップ¥Hiform.xml
移動
ページが表示されました
マイ コンピュータ

Program Name	
Subtitle	
Library Code	Version
Author	Original Release
Approver	Current Release
Key Words	CR Code
Scope	
Variants	
Language	Software Req.
Operation	Hardware Req.
Referenses	
Function	
Example	

Figure 8. The XML file which specified XSL is browsed by Internet Explorer