

TR-88-17

A VISUAL APPROACH TO SIMULATION OF  
ICON-BASED CHINESE INPUT  
METHODS

參 考 書  
不 併 備

中研院資訊所圖書室



3 0330 03 000090 0

0090

# A Visual Approach to Simulation of Icon-Based Chinese Input Methods

K. Y. Cheng, R. K. Cheng, and M. S. Hwu

Institute of Information Science, Academia Sinica  
and  
Computer Science and Information Engineering Department,  
National Taiwan University

## ABSTRACT

This paper presents a visual programming environment for users to develop their own defined icon-based Chinese input methods. It will show the following four aspects. First, Chinese characters are concrete icons (decomposition graphs or webs). Second, a grammar that generates the above mentioned webs is a context-free web grammar. Third, an icon-based Chinese input method defines semantic rules to the production rules of the web grammar which then becomes an attributed web grammar. Fourth, a Chinese input simulator is a machine that accepts these webs. Developing such a simulator can be regarded as the construction of an user-defined application program under the proposed visual programming environment. This paper also shows that a form-based visual approach is included in the environment which can be used to specify ways of evaluation and simulation in the user-defined simulator. This demonstrates that visual approach to simulation of icon-based Chinese input methods is still form-based despite the fact that visual objects are concrete icons.

## I. INTRODUCTION

The programming technique that utilizes certain interactive devices such as screen cursor, joystick, mouse, touch screen, etc., and certain screen operations such as menu, window, form, diagram, icon etc., has become increasingly popular in recent personal computers and workstations. Under such a programming environment, an user who possesses little or no knowledge of traditional programming languages can still develop his/her own application programs effectively. This style of programming requires an user merely to express his own expertise in terms of certain visual objects and their relations.

The visual objects are often form-based, diagram-based, icon-based, or of their mixture. These visual objects have been used in office information[21,22,23,27], computer-aided instruction[7,8,9], computer-aided design[1], data base interface[13,14], computer graphics[10,11], image handling[5,15,25], scene understanding[6], ideograph understanding[4], ..., etc. Interestingly enough, when one examines these applications closely, one may find a common fact that form-based or diagram-based or icon-based visual objects can be represented in graphs[7,8,11,15,16,19,22,23,24,25,27,28]. In other words, we may view the human visual conception in these applications as composition of labeled graphs, each with its labeled node represents a visual object and a labeled arc a relationship.

In this paper, we are particularly interested in the discussion of a visual approach to simulation of icon-based Chinese input methods. An icon-based Chinese input method regards the visualization of a Chinese character as a decomposition graph. The graph represents the ideographic formation of the character code by method-defined rules. Different methods produce different sets of decomposition graphs. For example, if the rule applied is to decompose

from left to right, then the visual conception of the character 明 ( bright ) may be viewed as a decomposition graph with the character as its root node, the primitive icons 日 ( sun ) and 月 ( moon ) as the desendent nodes, and part-of and left-of predicates as it labeled arcs (Fig. 2.1 ). A software program which simulates user-defined Chinese input methods under the test of some pre-defined evaluation procedures is a simulation program. Different evaluation procedures produce different simulation programs. The realization of an icon-based Chinese input simulator is a typical case where the notion of visual programming can be applied since the programs may be sketched visually. The sketched simulation program ( or a simulator ) can provide an user ( or a designer expert ) the ability to define his/her own Chinese input method and get an immediate performance evaluation in a manner of what-being-sketched-is-what-being-get.

The main results of this paper are presented in the following sections. Section II gives a formal definition of the visual language. Section III shows that the decomposition graph formed in the realization of a Chinese character may be regarded as a sentence generated by a context-free web grammar. Section IV shows that meta rules of an icon-based Chinese input method are semantic rules to attribute each production rule of the web grammar. Section V gives a formal definition of the Chinese input simulator. Section VI shows the development of the simulation program for evaluating user-defined icon-based Chinese input methods under the visual programming environment. Section VII shows a program construction process and demonstrates portion of its execution results.

## II. VISUAL LANGUAGE DEFINITION

As mentioned previously, the visualization( visual conception ) of an object is a graph, called the decomposition graph of the object. A visual language is a

collection of visualized objects, which means a decomposition graph may be regarded as a representation of sentential forms of the visual language. Therefore, the visual language may be formally defined as

$$L_v = \{ x \mid x \text{ is a graph representation which can be generated by a visual grammar } G_v \},$$

the visual grammar  $G_v$  is a four tuple

$$G_v = ( V_N, V_T, P, S )$$

where  $V_N$  is a set of nonterminals ( composite visual objects ),  $V_T$  is a set of terminals ( primitive visual objects ),  $P$  is a set of production rules, and  $S$  is a set of starting symbols. Note that the realization of a visual application program is to infer and then manipulate the visual grammar or its equivalent to obtain a synthesized program which runs the application.

The language  $L_v$  is called a t-based visual language if and only if the graph representation is t-based, where t is substitutable for words such as "form"[7,8,21,22,23,27,28], "diagram"[2,12,16], or "icon"[15,19,25]. For example, the visualized Chinese characters in an icon-based Chinese input method are sentences of an icon-based visual language ( or an iconic language for briefly). Fig. 2.1(a) shows a visualization of the character 明 as a decomposition graph and Fig. 2.1(b) shows the visual grammar that generates it. In practice, more than one t-based visual languages are used simultaneously in the development of an application program.

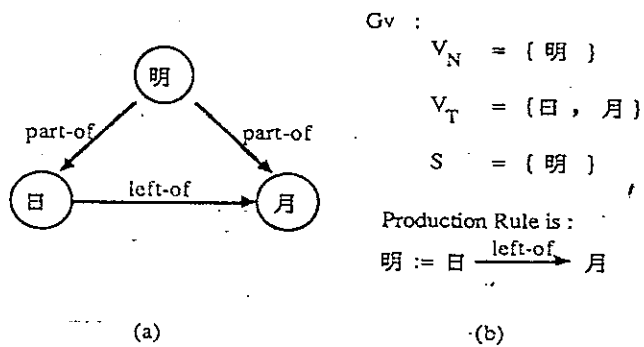


Fig. 2.1 The ideographic formation of the Chinese character 明  
 (a) the decomposition graph of 明, (b) the grammar that generates 明.

The visualization process that utilizes visual languages to construct an application program such that the constructed result is acceptable or realizable by a machine or an automaton is called visual programming. The environment under which the users can freely develop their visual application programs by visual programming is a visual programming environment. Note that the visual programming technique is a two-dimensional graphic construction which makes its programming style quite different from the conventional one-dimensional string description. As will be seen in later sections, the visual programming technique is easier to construct certain application programs such as the Chinese input simulator which would otherwise be rather difficult to implement in conventional ways.

### III. CONCRETE ICONS

An icon may be either primitive or compositive. A concrete icon can either be a primitive or a compositive one which, when represented in a decomposition graph, is a combination of primitive icons in the structure of spatial relations and descendent relations only. Let  $D_c = (V, E)$  denote a decomposition graph

which represents a concrete icon. In the node set ,  $V$  , the concrete icon is the roof node. Its primitive icons are the leaf nodes. The rest combined compositive icons are the nonterminal nodes. In the edge set ,  $E$  , there are two types of edges, the part of predicate and the spatial relation. The part-of predicate indicates the decendent relationship from one node to another, while the spatial relation shows the adjacency relationship between two icons.

A typical example of using the decomposition graph to represent the concrete icon is the visualization process of the Chinese character in an icon-based Chinese input method. Fig.3.1 shows such an example, where the Chinese character 腦 ( brain ) is represented as a decomposition graph. The character 腦 is a root node. The terminal nodes are primitive icons, and the rest are nonterminal nodes . The part-of predicate arcs are shown by dash lines and the spatial relation arcs by solid lines.

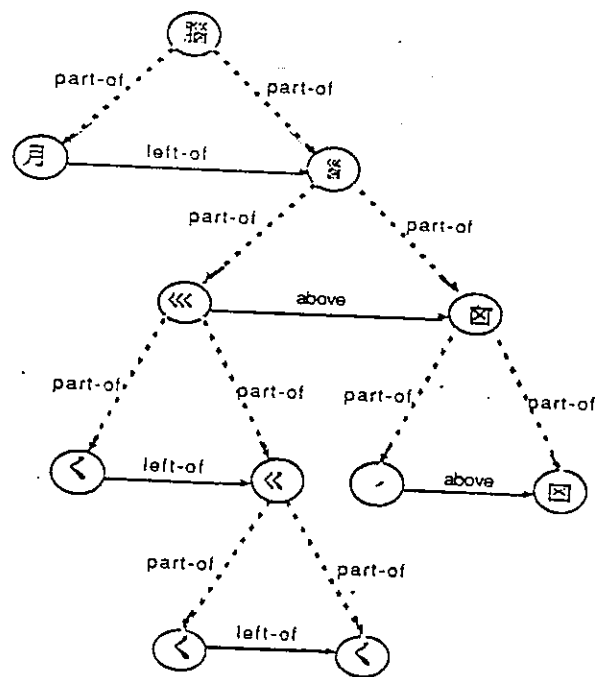


Fig. 3.1 The decomposition graph of the character 腦 .

To understand the formation steps in the visualization, we need to distinguish the part-of predicate from the spatial relations, thus we introduce a new graph called the derivation diagram as follows. A derivation diagram is obtained from a decomposition graph by preserving the inherited relationship of the spatial relations from an ancestor node to all of its descendant nodes. However, the readers are encouraged to consult [3,20] for a detailed procedure.

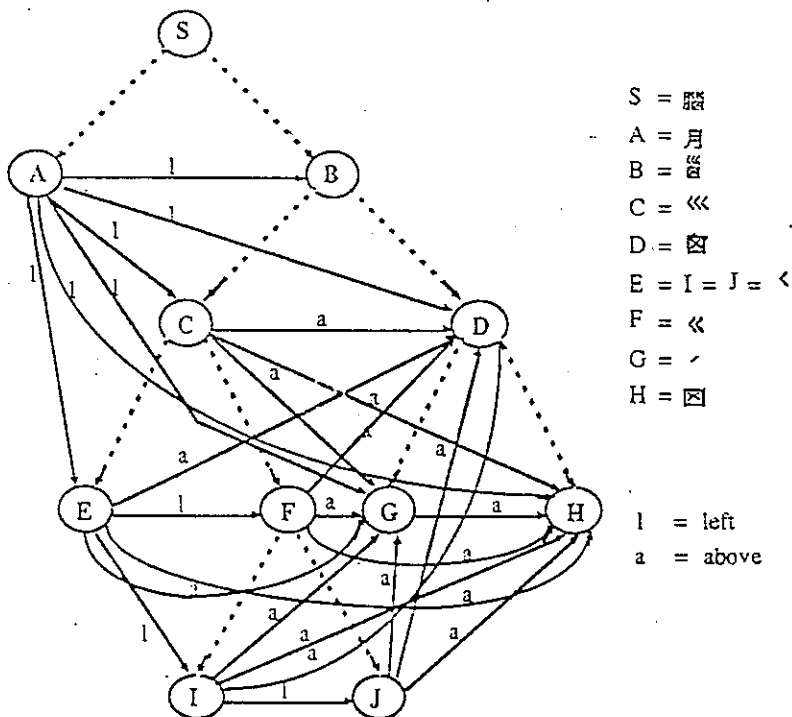


Fig. 3.2 The derivation diagram of the character 腦 .

Fig. 3.2 shows the derivation diagram of the character 腦. From a formal language theoretical point of view, a derivation diagram is a graphical expression of the derivation steps of a context-free web grammar[20]. A web production rule is defined as

$$\alpha \rightarrow \beta, \gamma$$



where  $\alpha$  and  $\beta$  are webs and  $\gamma$  is an embedding of  $\beta$ . In order to substitute a subweb  $\alpha$  of a web  $\omega$  by another subweb  $\beta$ , we have to specify how to embed  $\beta$  in  $\omega$  in place of  $\alpha$ . For example, the web grammar that generates the web 腦 is

$V_N = \{ \text{腦}, \text{管}, \text{函}, \text{《《}, \text{《} \}$ ,  $V_T = \{ \text{月}, \text{<}, \text{'}, \text{函} \}$ ,  $S = \{ \text{腦} \}$ ,  
and  $P =$  production rules shown in Fig. 3.3.

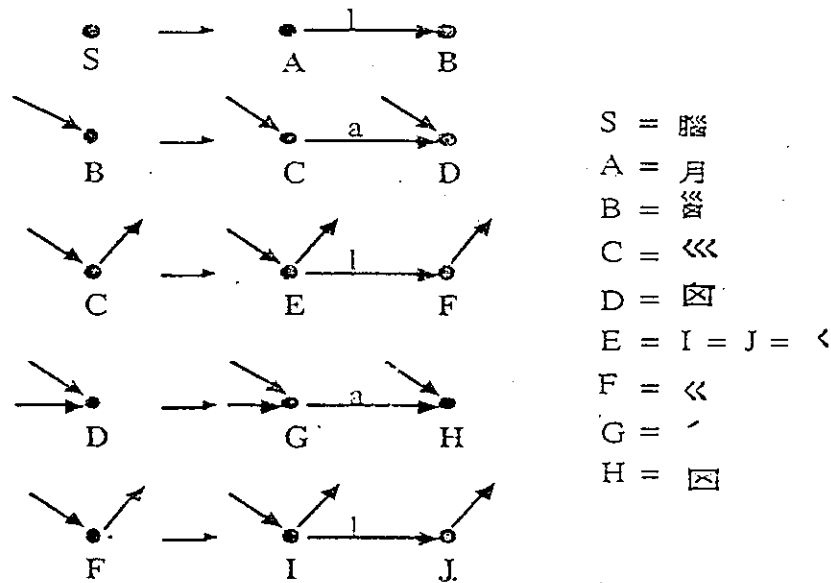


Fig. 3.3. The web production rules that generate the character 腦 .

To derive web production rules from a derivation diagram, we shall consider two graphs. One is called the skeleton and the other is called the section. A skeleton is a subdiagram obtained from a given derivation diagram by removing all of its spatial relation edges and leaving only part-of predicate edges and their connecting nodes. For example, Fig. 3.4 shows a skeleton of the diagram of Fig. 3.2. Notice that the skeleton is a tree structure whose root or subroot corresponds to a web  $\alpha$  and its decedents to another web  $\beta$  and thus a rewriting ( or production ) rule  $\alpha \rightarrow \beta$  where  $\alpha \in V_N$  and  $\beta \in V_N \cup V_T$  is obtained. The grammar thus obtained is context-free because of the nature of

tree structure.

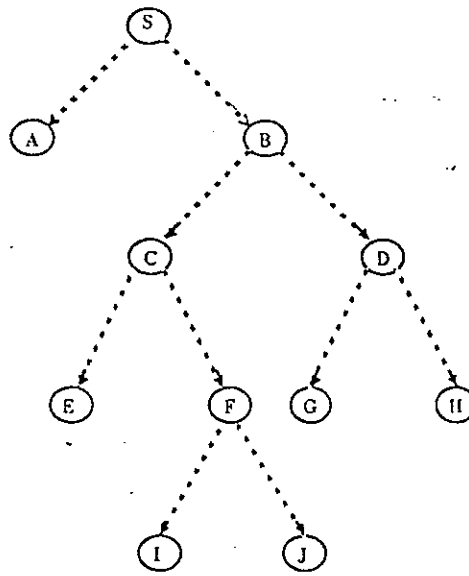


Fig. 3.4 The skeleton of fig. 3.2.

A section is a subdiagram that corresponds to a sentential form of the language generated by the web grammar. We are only interested in the terminal section that shows the external code symbols in order. For example, Fig. 3.5 shows the terminal section of Fig. 3.2. Suppose that  $n_0, \dots, m_i$  is a path from root node  $n_0$  to a frontier node ( or leave node )  $m_i$  of the skeleton, and let  $m_1, m_2, \dots, m_k$  be all the frontier nodes, then a set  $C$  of nodes of the derivation diagram is a cross-cut set if  $C \cap [n_0, \dots, m_i]$  is a singleton for all  $1 \leq i \leq k$ . A cross-cut set  $C$ , together with all the edges of the derivation diagram between nodes of  $C$ , is called a section.

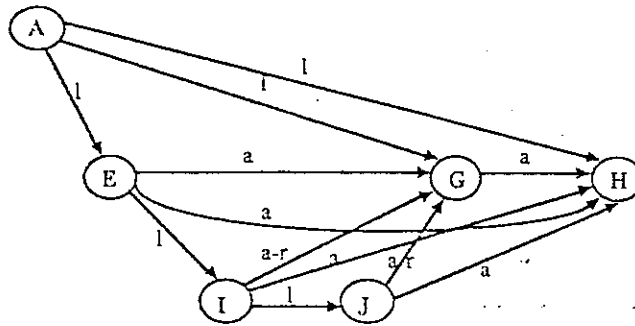


Fig. 3.5 The terminal section of fig. 3.2.

To obtain the web grammar from a given derivation diagram, a skeleton is first obtained to derive the production rules with  $\alpha \rightarrow \beta$ ,  $\alpha \in V_N$  and then another section is obtained to describe the embedding relationship of a production rule. Notice that the number of production rules with  $\alpha \rightarrow \beta$ ,  $\alpha \in V_N$  is equal to the number of internal nodes in the skeleton. Also notice that the nodes in the terminal section are primitive icons in  $V_T$ .

#### IV. ICON-BASED CHINESE INPUT METHOD

There are two types of meta rules adopted exclusively in every icon-based Chinese input method. One is that of the decomposition rule and the other is of the decoding rule. The decomposition rule provides an operator a selection order for the decomposition of each character. For example, suppose that the general rule to decompose a character is from left-to-right then from top-to-down then from outside-to-inside, then the character 腦 may be decomposed into a string of primitive icons 月, <, <, <, -, ☒, this, in turns, identifies the character.

The decoding rule sets conditions for an operator to follow since the number of primitive icons is limited to at most the number of terminal keys.

Therefore, the selected primitive icons can be considered as a reduced version of the original ones. For example, one may use 田 to represent 囧 and 女 to represent 𡥉 ( see Fig. 4.2 ). Other conditions include the external code length is limited to within a maximum number, only the first and last external codes are selected for decoding the same repeated external code, and so forth. Formally speaking, the decoding rules attribute each web production rule with a meaning in parsing a nonterminal. From Knuth's formal semantics, there are two kinds of attributes, the synthesized attribute and the inherited attribute respectively [17,18]. For each production in a context-free grammar, semantic rules are specified by defining (1) all the synthesized attributes of the nonterminal on the left side of the production, and (2) all the inherited attributes of the nonterminals on the right side of the production.

To illustrate an example, let Exc be the external code, Len the code length, and Lmt the limited true length. Then, the decoding rules of an icon-based Chinese input method, called the Tsang-Chi method, may be specified by defining the following semantic rules

- a. S is a starting symbol,  $S \in V_N$ ,  $A, B, C \in V_N \cup V_T$ .

The production rule is  $S \rightarrow \{ A, B \}$

If A and B are separable, then

$$\text{Exc}(S) = \text{Exc}(A) \cdot \text{Exc}(B)$$

$$\text{Len}(S) = \text{Len}(A) + \text{Len}(B)$$

$$\text{Lmt}(A) = 2$$

$$\text{Lmt}(B) = 3,$$

else

$$\text{Exc}(S) = \text{Exc}(A) \cdot \text{Exc}(B)$$

$$\text{Len}(S) = \text{Len}(A) + \text{Len}(B)$$

$$\text{Lmt}(A) = 3$$

$$\text{Lmt}(B) = 4 - \text{Len}(A)$$

b. The production rule is  $A \rightarrow \{ B, C \}$

If  $\text{Lmt}(A) \geq 2$ , then

$$\text{Exc}(A) = \text{Exc}(B) \cdot \text{Exc}(C)$$

$$\text{Len}(A) = \text{Len}(B) + \text{Len}(C)$$

$$\text{Lmt}(B) = \text{Lmt}(A) - 1$$

$$\text{Lmt}(C) = \text{Lmt}(A) - \text{Len}(B)$$

If  $\text{Lmt}(A) = 1$  and B and C are mutually inclusive, then

$$\text{Exc}(A) = \text{Exc}(B) \cdot \text{Exc}(C)$$

$$\text{Len}(A) = \text{Len}(B) + \text{Len}(C)$$

$$\text{Lmt}(B) = \text{Lmt}(A)$$

$$\text{Lmt}(C) = \text{Lmt}(A) - \text{Len}(B)$$

If  $\text{Lmt}(A) = 1$  and B and C are exclusive, then

$$\text{Exc}(A) = \text{Exc}(B) \cdot \text{Exc}(C)$$

$$\text{Len}(A) = \text{Len}(B) + \text{Len}(C)$$

$$\text{Lmt}(C) = \text{Lmt}(A)$$

$$\text{Lmt}(B) = \text{Lmt}(A) - \text{Lmt}(C)$$

c.  $A \in V_T$

If  $\text{Lmt}(A) > 0$ , then

$$\text{Exc}(A) = \text{Mk}(A)$$

$$\text{Len}(A) = 1$$

else

$$\text{Exc}(A) = \lambda$$

$$\text{Len}(A) = 0$$

where  $\cdot$  denotes a concatenate operator,  $M_k(A)$  denotes a pattern matching of the  $Exc(A)$  to the  $k$ -th primitive icon, and  $\lambda$  is a null element. It is seen that  $Exc$  and  $Len$  are two synthesized attributes while  $Lmt$  is an inherited attribute.

The grammar with each production rule being associated with a set of semantic rules is called an attributed grammar[18,26]. Fig.4.1 shows the attributed web grammar which describes the visualization of the character 腦 in the Tsang-Chi method. Fig.4.2 shows its derivation tree( only the skeleton is shown ).

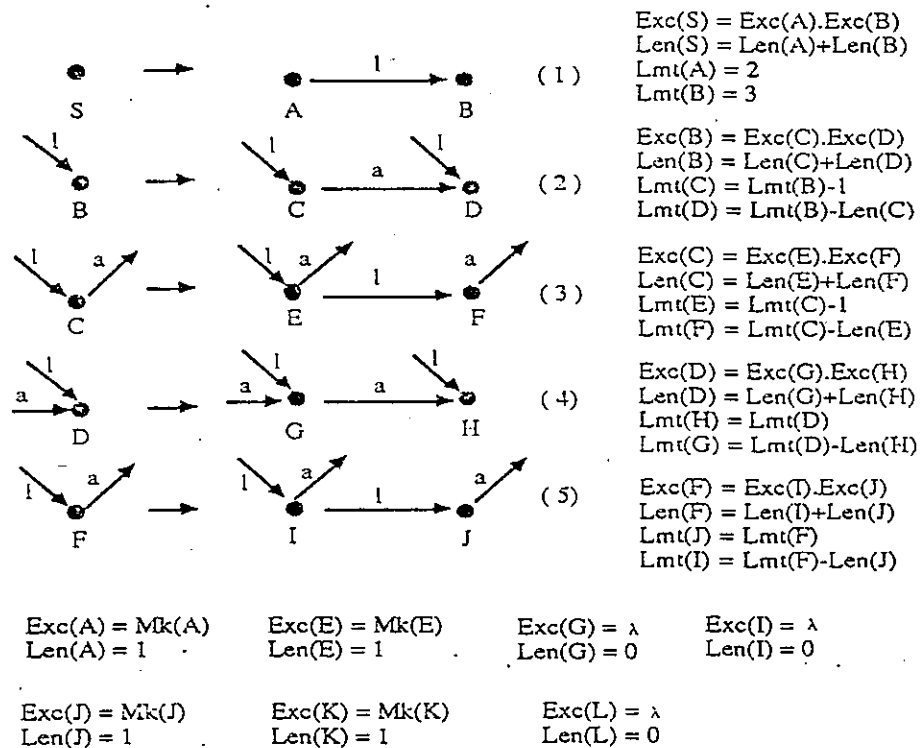


Fig. 4.1 Semantic rules associated in the web productions of 腦 .

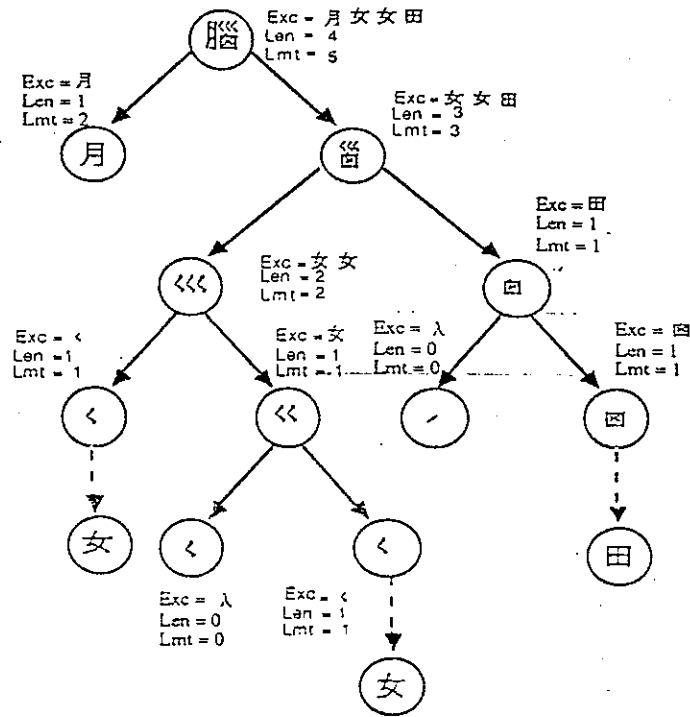


Fig. 4.2 The derivation tree of the character 腦 .

## V. CHINESE INPUT SIMULATOR

Due to the fact of employing the constraint decoding rules in the decomposition of the Chinese characters, a string of primitive icon codes thus obtained may correspond to many different characters which results the so called collision problem. An useful Chinese input method shall keep the collision rate low. In addition, requirements, such as the average number of key-in operations shall be less, the decomposition speed shall be faster, the decomposition rules shall be easier to use, etc., all need to be considered. In particular, the decomposition speed should be fast enough to meet the ordinary typing speed standard.

As far as the decomposition act is concerned, the decomposition speed depends on the number of derivation steps ( or the number of production rules

applied ) in character decoding as well as the frequency of the decomposition rules being used. With fewer derivation steps and easier applications of the decomposition rules, the decomposition speed will be faster. As stated in the previous section, the number of derivation steps can be obtained from counting the number of production rules with  $\alpha \rightarrow \beta$ ,  $\alpha \in V_N$  and  $\beta \in V_N \cup V_T$  being applied, and the number of keys depressed is the number of the terminal nodes. The decomposition speed and the easier use of the decomposition rules may be directly measured from counting the nodes and the appearance rate of the decomposition rules used. In other words, one may use the decomposition graphs as media not only to simulate a Chinese input method but also to derive production rules from them ( from parsing of the decomposition graphs ) in order to obtain its performance evaluation.

A Chinese input simulator is a machine that accepts user-defined decomposition graphs and corresponding decomposition rules as the input and produces the evaluation result as the output. Thus a simulator is a six tuple

$$\text{CIS} = ( Q, S, \Delta, \delta, q_0, F )$$

where:

1.  $Q$  is a finite set of states.
2.  $S = (P, E)$  is a finite set of decomposition graphs, the node set  $P$  is a finite set of external code symbols and the edge set  $E$  consists of two types of edge, a dash-line arrow specifies a part-of predicate and a solid-line arrow specifies a spatial relation.
3.  $\Delta$  is a finite set of output values.
4.  $\delta$  is a mapping from  $Q \times (S \cup \{\lambda\})$  to a finite subset of  $Q \times \Delta^*$ , and  $\lambda$  denotes a null element.



5.  $q_0 \in Q$  is the initial state.
6.  $F \subseteq Q$  is the set of final states.

It is worthwhile to note that some values may be generated and accumulated at each intermediate state so that the evaluation result is obtained simultaneously at the state. In the following section, we shall show a visual programming environment under which an user can define an application software for developing Chinese input simulators.

## VI. A VISUAL PROGRAMMING ENVIRONMENT

To support the development of an user-defined Chinese input simulator, a visual programming environment provides some facilities and tools for the user to select during his/her construction of the application program. An user may first select a table ( the decomposition rule table ) for the description of his/her own set of decomposition rules, then select another table ( the primitive icon table ) to define a set of primitive icons as terminal nodes for the decomposition graphs, and then select another table ( the spatial relation table ) to define spatial relations for arcs construction. After that, the user can then create a working area ( decomposition graph window ) to construct each decomposition graph of the corresponding Chinese character according to decomposition rules and a reporting form window to display the intermediate evaluation results. The system then converts each decomposition graph into a derivation diagram and records the information of skeleton and terminal section subgraphs into some system generated tables for later use in the evaluation process. A build-in function set defined in the evaluation procedure table is called by the system to generate values to measure the efficiency of the input method just defined. The evaluation result then shows in the reporting form window with format defined in a reporting form schema for recording and displaying both the intermediate

evaluation result during execution and the final evaluation result after execution.

Fig. 6.1 shows the system organization of a visual programming synthesizer. There are four definition languages for users to define the decomposition rules, the primitive icons, the spatial relations, and the evaluation procedures. The results are some tables, the decomposition rule table (DRT), the primitive icon table (PIT), the spatial relation table (SRT), and the evaluation procedure table (EPT). Notice that they are defined by a form-based visual language although they are going to be used in the construction of icons and icon related information. After these tables are defined, another form-based language can be used to define the reporting form schemata for recording and displaying the intermediate and final evaluation results. A schema defines the specification of the reporting form which consists of a form heading ( defining the form structure ) with form fields, the form field sources ( where are they coming from ), field data types, and specification items. Each specification item ( in a set of build-in function items ) selects some form fields for its specification. Finally, the program structure is described into an internal representation of the application program via a V-Form interface language. The internal structure of the application program defined in this way is an extended version of our previously designed visual programming synthesizer, the EVIPS [7,8], because by now we have to include two more V-Form types DG ( stands for decomposition graph ) and REPORT. DG is a type of window for the user to define decomposition graphs and to select the corresponding decomposition rules, while REPORT is a type of window where the intermediate and final reports show.

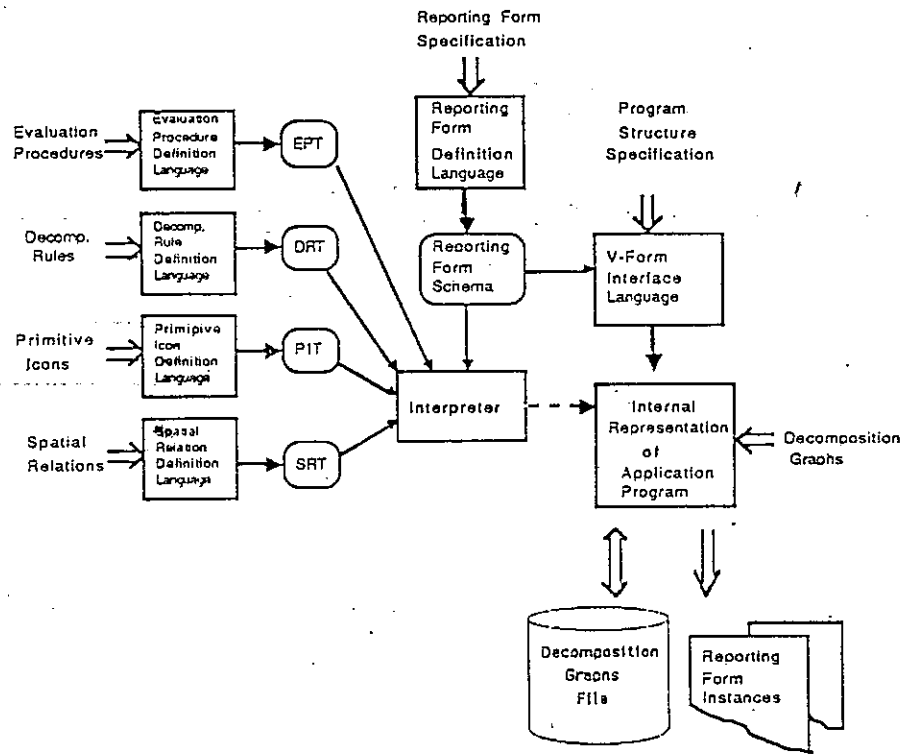


Fig. 6.1 The system organization of a visual programming synthesizer.

Fig. 6.2 shows a summary of the command menu used in the visual programming environment. On the top is a main menu for the selection of commands such as defining tables, specifying reporting form schemata, constructing application programs, executing the constructed application program, setting up or exiting from the system, or asking for help. The table command menu is for the construction of four tables, the decomposition rule table, the primitive icon table, the spatial relation table, and the evaluation procedure table with some editing operations, build-in function calls, and file manipulations. The reporting form schema command menu is for the construction of reporting form heading and its body. The program construction command menu is very much the same as that of EVIPS except the tables are PIT, DRT, EPT and SRT.

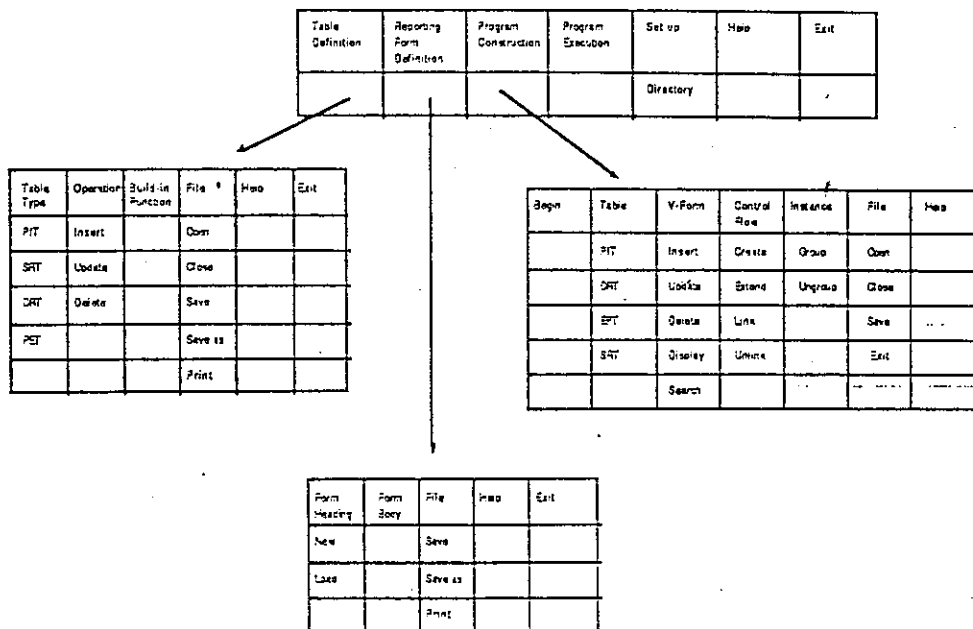


Fig. 6.2 The summary of the command menu.

## VII. PROGRAM CONSTRUCTION

From the user's point of view, the construction of a Chinese input simulator, under the previously described visual programming environment, must follow certain steps in order. Fig. 7.1 shows such an application program construction flow diagram.

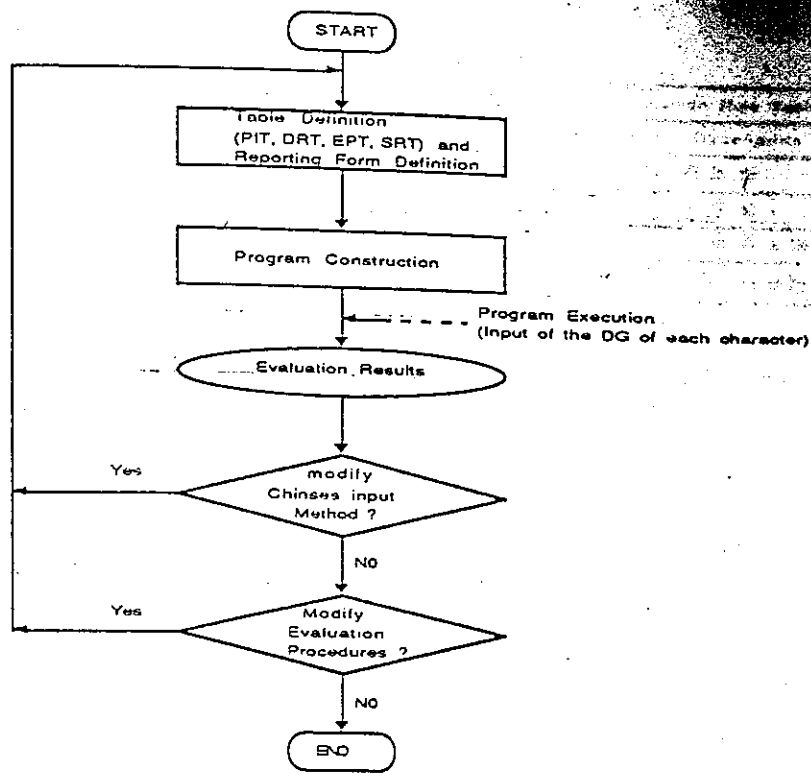





Fig.7.1 Application program construction flow.

As mentioned above, the four tables : DRT, PIT, SRT, and EPT ( Fig. 7.2 shows their appearance ) and a reporting form schema ( Fig. 7.3 shows its appearance ) are first defined. To define a reporting form schema, the user can load in a reporting form heading if it has been defined earlier and select the proper form fields for each specification item.

Icon	Internal Code
月	02
女	11
田	20
口	24
土	30

Icon	Description
	left-of
	right-of
	above-of

ID	Type	Description
dr1	D	左右分
dr2	D	上下分
er2	E	全部取五码
er3	E	字首取用码

Name	Parameters	Definition
$f$	$s$	$\text{SORT} \{ 0.8 + 0.2 * (\text{NO-POC}(\text{IN-RULE}(s)) / \text{NO-PR}) \}$
$ds$	$t, x$	$1, \forall x \in \text{Terminal Nodes}$
$ds$	$h, x$	$\text{SORT} \{ \text{SUM}(\text{ds}(h-1, y)) * f / \text{SUM}(\text{HEIGHT}(y)), \forall y \in \text{CHILD}(x) \}$

Fig.7.2 An example of part of four tables.

Form Heading	Form Body	File	Help	Exit			
IRPT : Chinese_input_evaluation							
(Character Result)							
	character symbol name	occur. rate (%)	decomp. speed	(Radical) name total occur. times	(Rule) name total occur. times		
SOURCE	symbol(cic)	bratel(cic)	ds(cic)	symbol(cic)	occ(cic)	rule(cic)	rate(cic)
DATA TYPE							
KEY	✓			✓		✓	
COUNT	✓						
AVG		✓	✓		✓		✓
MIN		✓	✓		✓		✓
MAX		✓	✓		✓		✓

Fig. 7.3 The definition of the reporting form schema.

Fig. 7.4 shows an insertion process of a V-Form with type REPORT and name RESULT onto the V-Form page, the VFORM:Decom\_Result ( a program segment ), the result is showing in Fig. 7.5.

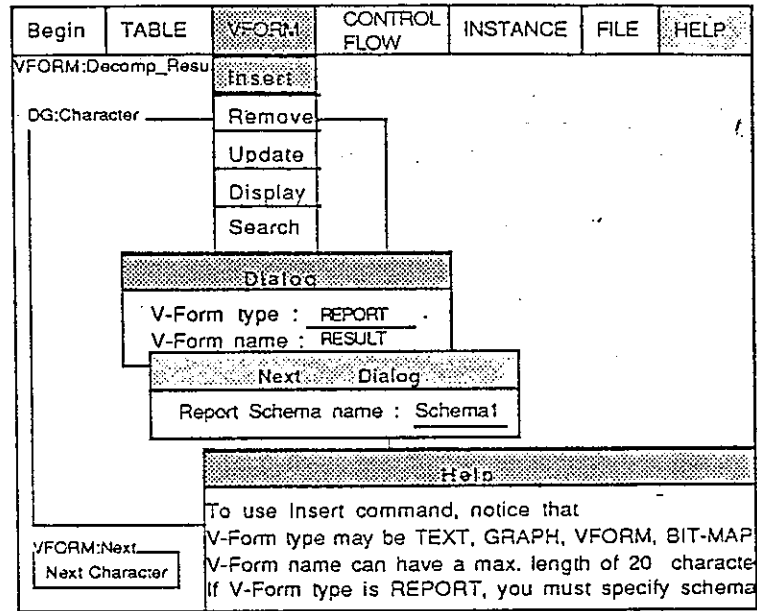


Fig. 7.4 The insertion process of a V-Form with type REPORT.

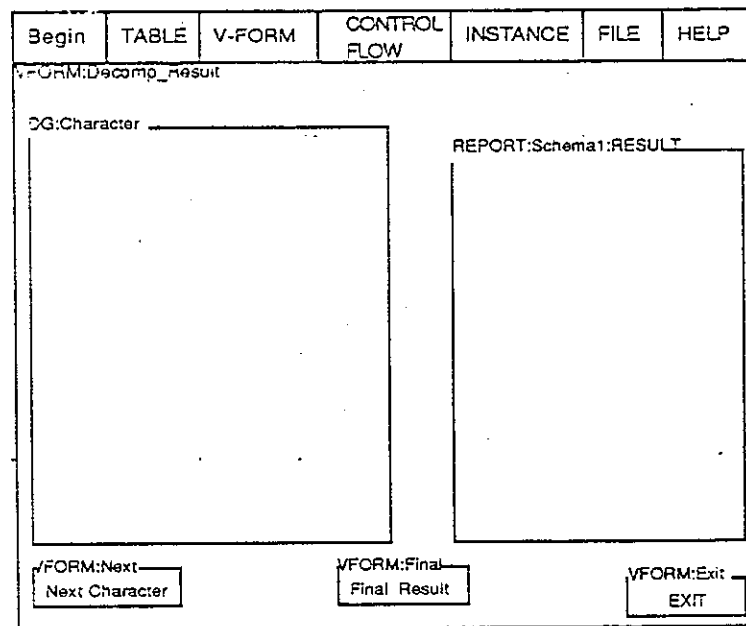


Fig. 7.5 The result after V-Form REPORT:Schema1:RESULT is inserted.

Fig.7.6 shows the execution of the program segment VFORM:Decom\_Result. As can be seen, the user is using two tables, the

primitive icon table and the spatial relation table, to construct the decomposition graph of the Chinese character 腦 ( brain ) and describing the decomposition rules being used associated with the character in a table ( the Corresponding Rule Table ). After the construction of the decomposition graph of the character, the evaluation result ( accumulative updated values ) appears in the right reporting form. To continue the construction of the next decomposition graph, the V-Form Next Character is selected. When all the decomposition graphs have been constructed, the V-Form Final Result is then selected to get the overall evaluation result of the Chinese input method just designed ( the result shows in Fig. 7.7 ).

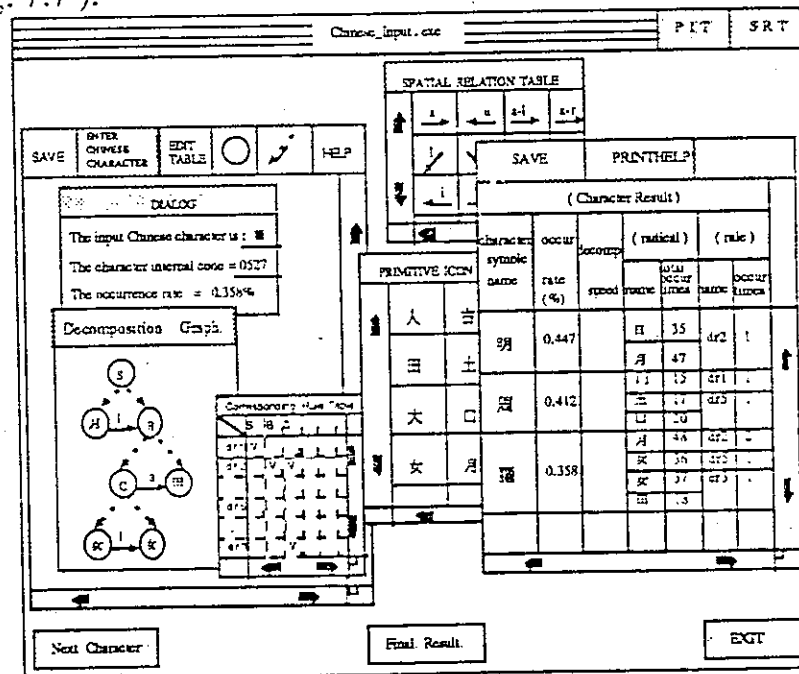


Fig.7.6 The execution of VFORM:Decomp\_Result.



SAVE				PRINT		HELP	
Total No. of characters	Average decomposition speed	Total no. of primitive icons used	Average no. of keys depressed	Average collision rate	Average no. of rule used in one character		
13900	0.825	105	3.2	0.27	4.5	↑	↓

The following reporting form is the summary of the evaluation results

Chinese\_Input . exe      P L I T S      T

Next Input Method      END

Fig.7.7 The overall evaluation result of the Chinese input method designed.

## VIII. CONCLUSION

In this paper presented, we showed that the visualization of the Chinese characters in an icon-based Chinese input method is an iconic language. An icon-based Chinese input method is defined in terms of a set of decomposition graphs and that of decoding rules. A decomposition graph ( or a web ) can be transformed into a derivation diagram. A web is a derivation diagram for a contex-free web grammar. The decoding rules assign meaning to each production of the grammar which then becomes an attributed context-free web grammar. The number of production rules is the number of decomposition steps in parsing the visual routines to decode a Chinese character. Based upon these observations, a Chinese input simulator can be described as a machine that accepts the specification languages of the decomposition graphs and the decoding rules. We then showed that a visual programming environment can be established for users to define their own Chinese input simulators ( application programs ) and run their own defined Chinese input methods on it.

## REFERENCES

- [ 1 ] M. Beretta, P. Mussio, and M. Protti, "Icons: Interpretation and Use", Second IEEE Computer Society Workshop on Visual Language, June 1986, pp.149-158.
- [ 2 ] Alfs T. Berztiss, "Specification of Visual Representations of Petri Nets", Third IEEE Computer Society Workshop on Visual Language, Aug. 1987, pp.225-233.
- [ 3 ] J. M. Brayer and K. S. Fu, "Some multidimensional grammar inference methods", in Pattern Recognition and Artificial Intelligence, (C. H. Chen, ed.), Academic Press, New York, 1976.
- [ 4 ] S. K. Chang, "Icon Semantics - A Formal Approach to Icon System Design", Int'l Jour. of Pattern Recognition and Artificial Intelligence, World Scientific Press, 1987.
- [ 5 ] S. K. Chang, E. Jungert, S. Leviardi, G. Tortora, and T. Ichikawa, "An Image Processing Language with Icon-Assisted Navigation", IEEE Transactions on Software Engineering, August 1985, pp.811-819.
- [ 6 ] S. K. Chang, Q. Y. Shi, and C. W. Yan, "Iconic Indexing By 2D strings", second IEEE Computer Society Workshop on Visual Language, June 1986, pp.12-21.
- [ 7 ] K. Y. Cheng, C. C. Hsu, I. P. Lin, M. C. Lu, and M. S. Hwu, "VIPS: A Visual Programming Synthesizer", Second IEEE Computer Society Workshop on Visual Language, June 1986, pp.92-98.
- [ 8 ] K. Y. Cheng, M. S. Hwu, and C. C. Hsu, "An Extended Visual Programming Synthesizer for Computer Aided Instruction Applications", Third IEEE Computer Society Workshop on Visual Language, Aug. 1987, pp.147-160.
- [ 9 ] W. Finzer and L. Gould, "Programming by Rehearsal", Byte, Vol. 9, No. 6, June 1984, pp.187-210.
- [ 10 ] G. F. McCleary, Jr. , "An Effective Graphic "Vocabulary"", IEEE Computer Graphics and Applications, March / April 1983, pp.46-53.
- [ 11 ] F. S. Montalvo, "Diagram Understanding: Associating Symbolic Descriptions with Images", Second IEEE Computer Society Workshop on Visual Language, June 1986, pp.4-11.
- [ 12 ] E. P. Glinert and S. L. Tanimoto, "Pict : an Interactive Graphical Programming Environment", IEEE computer Magazine, November 1984, pp.7-25.
- [ 13 ] C. F. Herot, "Spatial Management of Data", ACM Trans. on Database Systems, Vol. 5, No. 4, 1980, pp.493-514.
- [ 14 ] K. Hoehne, "The ISQL Language - A unified Tool for Managing Images and Non-Images Data Management System", in Visual Languages, edited by S. K. Chang et. al., Plenum Pub. Co. 1986.
- [ 15 ] T. Ichikawa, "HI-VISUAL: A Language Supporting Visual Interaction in Programming", in

Visual Languages, edited by S. K. Chang et. al., Plenum Pub. Co. 1986.

- [16] R. J. K. Jacob, "A State Transition Diagram Language of Visual programming", IEEE Computer, Vol. 18, No. 8, Aug. 1985, pp.51-59.
- [17] D. E. Knuth, "Semantics of context-free languages," *J. Math. Syst. Theory* 2, 127-46 (1968).
- [18] P. M. Lewis II, D. J. Rosenkrantz, and R. E. Stearns, *Compiler Design Theory*, Addison-Wesley, Reading, Mass., 1976.
- [19] M. A. Musen, L. M. Fagan, and E. H. Shortliffe, "Graphical Specification of Procedural Knowledge for an Expert System", Second IEEE Computer Society Workshop on Visual Language, June 1986, pp.167-178.
- [20] J. L. Pfaltz and A. Rosenfeld, "Web grammars", Proc. First Int. Joint Conf. Artif. Intell., May 1969, Washington, D.C., pp.609-619.
- [21] N. C. Shu, et al., "Specification of Forms Processing and Business Procedures for Office Automation", IEEE Transactions on Software Engineering, Vol. SE-8, No. 5, Sep. 1982, pp.499-512.
- [22] N. C. Shu, "A Forms-oriented and Visual-directed application development system for non-programmers", First IEEE Computer Society Workshop on Visual Language, Dec.1984, pp.162-170.
- [23] N. C. Shu. "FORMAL : A Form-Oriented, Visual-Directed Application Development System", IEEE Computer, Vol. 18, No. 8, 1985, pp.38-49.
- [24] Tadashi Ae and Reiji Aibara, "A Rapid Prototyping of Real-Time Software Using Petri Nets", third IEEE Computer Society Workshop on Visual Language, Aug. 1987, pp.234-241.
- [25] I. Yoshimoto, N. Monden, M. Hirakawa, M. Tanaka, and T. Ichikawa, "Interactive Iconic Programming Facility in HI-VISUAL", Second IEEE Computer Society Workshop on Visual Language, June 1986, pp.34-41.
- [26] K. C. You and K. S. Fu, "A syntactic approach to shape recognition using attributed grammars," *IEEE Trans. Syst. Man Cybern.*, SMC-9(6), June 1979.
- [27] M. M. Zloof, "Query-by-Example: A Data Base Language", IBM System Journal, Vol. 16, No. 4, 1977, pp.324-343.
- [28] M. M. Zloof, "QBE/OBE: A Language for Office and Business Automation," IEEE Computer, Vol. 14, No. 5, 1981, pp.13-22.