A Model of Syntax-Directed Transduction of Unrestricted Grammars Using 2PDA with Multisymbol Matching Production Rules

Steven William Cooke
Western Michigan University

Follow this and additional works at: http://scholarworks.wmich.edu/masters_theses
Part of the Computer Sciences Commons

Recommended Citation
http://scholarworks.wmich.edu/masters_theses/1299

This Masters Thesis—Open Access is brought to you for free and open access by the Graduate College at ScholarWorks at WMU. It has been accepted for inclusion in Master's Theses by an authorized administrator of ScholarWorks at WMU. For more information, please contact maira.bundza@wmich.edu.
A MODEL OF SYNTAX-DIRECTED TRANSDUCTION OF UNRESTRICTED GRAMMARS USING 2PDA WITH MULTISYMBOL MATCHING PRODUCTION RULES

by

Steven William Cooke

A Thesis
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Master of Science
Department of Computer Science

Western Michigan University
Kalamasoo, Michigan
April 1986
Syntax-directed transduction of unrestricted grammars is modeled by 2PDA automata with multisymbol matching production rules. Syntax-directed transduction normal form grammars (SDT) are constructed for unrestricted grammars. Transduction of sequences of derivation directives to sequences of derivation steps is defined for SDT grammars. E2PDA, a superset of 2PDA with input, output, and multi-symbol matching production rules, is formalized. The class of languages generated by E2PDA is proven equivalent to the class of languages generated by unrestricted grammars. E2PDA syntax-directed transducers are constructed for SDT grammars.
ACKNOWLEDGEMENTS

I would like to dedicate this thesis to my wife, Suzanne, and my parents, Dean and Marjorie, who through their encouragement and patience are responsible for its completion. I would especially like to thank Professor Dionysios Kountanis for his support and understanding through the completion of this research.

Steven William Cooke
INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of “sectioning” the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.
Cooke, Steven William

A MODEL OF SYNTAX-DIRECTED TRANSDUCTION OF UNRESTRICTED GRAMMARS USING 2PDA WITH MULTISYMBOL MATCHING PRODUCTION RULES

Western Michigan University M.S. 1986

University Microfilms International 300 N. Zeeb Road, Ann Arbor, MI 48106
TABLE OF CONTENTS

ACKNOWLEDGEMENTS ................................................................. ii

CHAPTER

I. INTRODUCTION ................................................................. 1

II. SDT NORMAL FORM GRAMMARS ................................. 3
    Definition of a SDT Normal Form Grammar ...................... 5
    Construction ........................................................................ 5
    Definition of SDT Transduction ....................................... 6
    Example ................................................................................ 7

III. E2PDA AUTOMATA .......................................................... 11
    Definition of an E2PDA .................................................. 14
    Theorem I ............................................................................ 16
    Theorem II ........................................................................... 18
    Example ............................................................................... 19

IV. SDT TRANSDUCTION USING E2PDA ......................... 22
    Construction ....................................................................... 23
    Example ............................................................................... 24

V. EXTENSIONS ................................................................. 30

VI. CONCLUSIONS ............................................................... 34

REFERENCES ........................................................................ 36

BIBLIOGRAPHY ...................................................................... 37
CHAPTER I

INTRODUCTION

This paper describes a model of syntax-directed transduction of unrestricted grammars by two-stack pushdown automata (2PDA) with multisymbol matching production rules (E2PDA). A normal form grammar, which provides a framework for syntax-directed transduction, is generated for unrestricted grammars. An automaton which has stack operations permitting a finite number of stack symbols to be pushed onto or popped from its stacks is formally defined and shown to be equivalent in power to Turing machines. Syntax-directed transducers (SDT) are demonstrated to be effectively constructed with this automaton for any unrestricted grammar. Extensions to the basic automaton are described which permit it to function as a syntax-directed compiler incorporating semantic routines, symbol tables, and attribute evaluation rules, enabling it to recognize unrestricted attribute grammars.

Syntax-directed transduction is a methodology in which the terminal strings of a grammar are generated interactively as steps in their derivation sequences. The various sentential forms in the derivation sequence are translated to the next via directives specifying the location and production rule to apply to generate the next sentential form. The full computational complexity of Turing computability is provided by permitting the use of unrestricted grammars in the generation of strings. Using a grammar-based model of computation, systems based upon production rules may be easily modeled. The usual automata used to compute the class of unrestricted grammars are Turing machines. However, the strong connection between derivation steps and production rules of an automaton are not as obvious with Turing machines and unrestricted grammars as they are with pushdown automata and
context-free grammars in the context of parsing.
CHAPTER II

SDT NORMAL FORM GRAMMARS

Syntax-directed transduction is a grammar-based analog to editing where the curser represents the location of the next production rule to apply and the directives represent the production rule to apply at the current curser location. Acceptance of unrestricted grammars by Turing class automata requires nondeterminism on the part of the automata. However, by generating final strings of a language via transduction, where the input language selects the next derivation step to apply in the generation of the final strings, a deterministic machine is permitted. In transduction of unrestricted grammars, it is necessary to specify the location of the next production rule to apply since the generation of unrestricted grammars does not place a limitation on the location where a production rule may be applied. In contrast, parsing deterministic context-free grammars limits the location where a production rule may be applied to the leftmost nonterminal of the sentential form. It is also necessary to specify the production rule to apply since unrestricted grammars may have multiple symbols on the left hand side of a production rule and a number of the production rules may match the current sentential form in the context of the present curser location.

To model the behavior of a grammar under syntax-directed transduction, the grammar is converted to a normal form which explicitly represents the action of the curser and the specification of the production rules to apply. This normal form grammar will be referred to as the SDT Normal Form Grammar for the grammar. The full specification of the SDT Normal Form Grammar for any unrestricted grammar is detailed in the construction given in this chapter. The converted grammar includes productions to start the derivation by appending a curser to the left of
the start symbol of the original grammar, move the curser one symbol to the left or right in the sentential form, delete the curser and terminate the derivation, and productions corresponding to those from the original grammar with curser prefixes which force productions to occur only in the presence of the curser.

The construction in this chapter demonstrates that the SDT Normal Form Grammar for any grammar generates the same language as the language of the original grammar. If we have a sequence of derivation steps in the original grammar for a string, then corresponding to each of these steps is a set of derivation steps in the SDT Normal Form Grammar. The first step in the set is the application of the production rule corresponding to the production rule from the original grammar and the remaining steps in the set are the movements of the curser to the location required for the next application of a production. The initial step in the derivation sequence using the normalized grammar generates the curser and goal symbol of the original grammar. The final derivation step for the normal form grammar removes the curser.

The last section in this chapter defines the term SDT transduction. If G' is the SDT Normal Form grammar for G, then a syntax-directed transduction of a string, Alpha, by G' is a relation, R, with input specifying the production rules in G' to apply to the sentential forms and output specifying the sequence of sentential forms generated in the derivation sequence of Alpha. The format of the input language provides for application, curser motion, and curser removal directives. The output language is composed of the sequence of sentential forms generated in the derivation with end markers for each sentential form. The output terminates with the final string of the derivation. The relation, R, of input to output, defining the transduction of directives to terminal strings of the grammar, G, is modeled by a 2PDA with multisymbol matching production rules.
Definition of an SDT Normal Form Grammar

Let \( G = (V, T, P, S') \) be an unrestricted grammar. \( G \) is an SDT Normal Form Grammar if it includes exactly the productions:

1) \( S' \rightarrow S \quad S \in V \),
2) \( a S \rightarrow a \quad \forall a \in V \cup T \),
3) \( S a \rightarrow a S \quad \forall a \in V \cup T \),
4) \( S \rightarrow \epsilon \),
and a finite number of productions of the form

5) \( SX \rightarrow SY \quad X \in (V \cup T)^+ \quad Y \in (V \cup T)^* \).

Construction

For every grammar, \( G \), an SDT Normal Form Grammar, \( G_{\text{alt}} \), can be constructed s.t. \( L(G) = L(G_{\text{alt}}) \).

Let \( G = (V, T, P, S) \) be an unrestricted grammar.

Construct \( G_{\text{alt}} = (V' = (V \cup \{S', \}$\}), T' = T, P', S') \) where the productions, \( P' \), are of the form:

1) \( S' \rightarrow S \),
2) \( a S \rightarrow a \quad \forall a \in V \cup T \),
3) \( S a \rightarrow a S \quad \forall a \in V \cup T \),
4) \( S \rightarrow \epsilon \), and
5) \( SX \rightarrow SY \quad \forall X \rightarrow Y \in P \).

\( G_{\text{alt}} \) does the following:

1) Generates the goal symbol of the grammar, \( G \), with prefix marker, $;
2) Applies a production $\beta \rightarrow $ s.t. $ \rightarrow \gamma \in P$;
3) If the final string, \( \alpha \), is generated, then removes the marker, $,
and terminates; or

4) moves the marker, $, left or right until it is positioned to the left of the
leftmost symbol of the next production to apply.

Returns to step 2.

Let $S = a_1 \Rightarrow a_2 \Rightarrow \ldots \Rightarrow a_n$ be a derivation sequence for $\alpha \in L(G)$, represent
each $\alpha_i$ in the form $V_i\beta_iW_i$, where $\beta_i \rightarrow \beta_{i+1}$ is a production rule in $G$ and $V_i\beta_iW_i \Rightarrow
V_{i+1}\beta_{i+1}W_{i+1}$ is a derivation step for $\alpha$.

Corresponding to each derivation step, $\alpha_i \Rightarrow \alpha_{i+1}$ $i \in 1 - (n - 1)$, in $G$ there is
the sequence of derivation steps in $G'$:

$V_i\beta_iW_i \Rightarrow V_i\gamma_iW_i \Rightarrow V_{i+1}\beta_{i+1}W_{i+1}$,

where

$k_i = |V_i| - |V_{i+1}|$.

$S' \Rightarrow a_1$ by $S' \rightarrow \$ S$.

$\alpha_n \Rightarrow \\alpha$ by $S \rightarrow \epsilon$.

Definition of SDT Transduction

Let $G = (V, T, P, S)$ be an unrestricted grammar with productions numbered
$1 - m$.

Let $G'$ be an SDT Normal Form Grammar for $G$.

Let $S' = a_0 \Rightarrow a_1 \Rightarrow a_2 \Rightarrow \ldots \Rightarrow a_n$ be a derivation sequence for $\alpha \in L(G')$.

A Syntax-Directed Transduction of $\alpha$ by $G'$ is a relation $R_{sd} (\omega, \phi)$, where $\omega$ is
of the form:

$\omega_1\omega_2\ldots\omega_{n-1}$,

where

$\omega_i = j$ if $\alpha_i \Rightarrow \alpha_{i+1}$ by $\$X_j \rightarrow \$Y_j, \ X_j \rightarrow Y_j \in P$,

$\omega_i = L$ if $\alpha_i \Rightarrow \alpha_{i+1}$ by a $\$ \rightarrow \$ a,
\[ \omega_i = R \text{ if } \alpha_i \Rightarrow \alpha_{i+1} \text{ by } S \rightarrow a S, \]
\[ \omega_i = H \text{ if } \alpha_i \Rightarrow \alpha_{i+1} \text{ by } S \rightarrow \epsilon, \]

and \( \phi \) is of the form
\[ \alpha_0 \# \alpha_1 \# \alpha_2 \# \ldots \alpha_n \#. \]
\[ \alpha_n \Rightarrow \alpha_1 \text{ by } S' \rightarrow S. \]

Example

Let \( G = (V = \{ S, X, Y, Z, A, C \}, T = \{ a, b, c \}, P, S) \).

The productions, \( P \), are:

1) \( S \rightarrow XbZY \),
2) \( S \rightarrow \epsilon \),
3) \( Z \rightarrow AbZC \),
4) \( Z \rightarrow \epsilon \),
5) \( bA \rightarrow Ab \),
6) \( XA \rightarrow aX \),
7) \( X \rightarrow a \),
8) \( CY \rightarrow Yc \), and
9) \( Y \rightarrow c \).

A sequence of derivation steps for the string \( aabbc \) is:

\[ S, \]
\[ XbZY, \]
\[ XbAbZCY, \]
\[ XbAbCY, \]
\[ XAbbCY, \]
\[ aXbbCY, \]
\[ aabbbCY, \]
a a b b Y c, and
a a b b c c.

The sequence of production rules applied in the derivation sequence is:

1, 3, 4, 5, 6, 7, 8, 9.

Construct $G_{ul}=(V'=V \cup \{S', \$\}), T'=T, P', S')$.

The productions, $P'$, are:

1) $S' \rightarrow \$ S,$
2) $S \$ \rightarrow \$ S,$
3) $\$ S \rightarrow S \$,$
4) $X \$ \rightarrow \$ X,$
5) $\$ X \rightarrow X \$,$
6) $Y \$ \rightarrow \$ Y,$
7) $\$ Y \rightarrow Y \$,$
8) $Z \$ \rightarrow \$ Z,$
9) $\$ Z \rightarrow Z \$,$
10) $A \$ \rightarrow \$ A,$
11) \$ A \rightarrow A \$,$
12) $C \$ \rightarrow \$ C,$
13) $\$ C \rightarrow C \$,$
14) $a \$ \rightarrow \$ a,$
15) $a \$ \rightarrow \$ a,$
16) $b \$ \rightarrow \$ b,$
17) $b \$ \rightarrow \$ b,$
18) $c \$ \rightarrow \$ c,$
19) $c \$ \rightarrow \$ c,$
20) $\rightarrow \epsilon,$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
21) $ S \rightarrow$ $ X b Z Y,$
22) $ S \rightarrow,$
23) $ Z \rightarrow$ $ A b Z C,$
24) $ Z \rightarrow,$
25) $ b A \rightarrow$ $ A b,$
26) $ X A \rightarrow$ $ a X,$
27) $ X \rightarrow$ $ a,$
28) $ C Y \rightarrow$ $ Y c,$ and
29) $ Y \rightarrow$ $ c.$

A sequence of derivation steps for $a b c$ in $G_{sc}$ is:

$S' ,
S,$
$ S ,
X b Z Y ,
X b Z Y ,
X b Z Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
X A b Z C Y ,
a X b b C Y ,
a a X b b C Y ,
a a X b b C Y ,
a$
a \$ a b b C Y,
a a \$ b b C Y,
a a b \$ b C Y,
a a b b \$ C Y,
a a b b \$ Y c,
a a b b \$ c c, and
a a b b c c.

The sequence of production rules applied in the derivation sequence is:

1, 21, 5, 17, 23, 11, 17, 24, 16, 10, 16, 25, 4, 27, 15, 27, 15, 17, 17, 28, 29, 20.

The sequence of syntax-directed transduction directives is:

CHAPTER III

E2PDA AUTOMATA

The class of unrestricted languages is accepted by automata which are able to scan left and right on their tapes and replace single instances of symbols under their read write heads. Both the Turing machine and the 2PDA have this mechanism. The two stacks of the 2PDA are just special cases of tapes. However, in comparison to the multisymbol matching and replacement capability of a deterministic context-free parser, the computational ability of this class of automata does not easily model the transitions made in the derivation steps of the class of languages generated by this class of automata. The necessary element needed in a machine used for modeling transitions in a derivation is the ability to make transitions of the machine corresponding to replacement of the left hand side of a production rule by the right hand side of the production rule. The strategy used to prove that the class of languages generated by unrestricted grammars is equivalent to the class of languages accepted by Turing machine requires a shifting over scheme. This shifting over technique is necessary to permit the string on the tape to shrink or grow to account for the differing lengths of the left and right hand sides of the applied production rule. In comparison, a machine which permits a direct move corresponding to a derivation step can easily be seen to be based upon a multisymbol matching scheme analogous to that of the deterministic context-free parser. The deterministic context-free parser is a single stack machine which matches the topmost portion of the stack and an input symbol with symbols specified in a production rule for the machine. Since the class of Turing computable languages is given by 2PDA, it is natural to extend the types of production rules used for this type of automata to include multisymbol matching in the same way as those of the deterministic context-
free parser. By doing this the combined advantages of Turing computability and string matching are achieved.

This chapter defines a model which provides both Turing computability and multisymbol matching production rules. This automaton is referred to as an E2PDA since it has an extended scan of the two stacks. E2PDA is a generalization of the usual 2PDA in that it also has a finite number of states, two pushdown stacks, and it defines a mapping of states and stack elements to other states and stack elements. However, in the E2PDA model the restriction on the number of symbols matched by a production rule is lifted and an arbitrary finite sequence of symbols may be matched on both the left and right stacks. For full generality the definition describes a transducer which implies an input and output tape and these tapes also have multisymbol matching capabilities in the production rules. The production rules are mappings from $Q \times I^* \times S^* \times S^*$ to $Q \times O^* \times S^* \times S^*$ where $Q$ is a state, $I^*$ is the set of input strings, $O^*$ is the set of output strings, and $S^*$ is the set of stack symbol strings. Note that the $I^*$ and $S^*$ strings refer to a prefix of the input string and the topmost symbols from the left and right stacks respectively. An instantaneous description of the state of a machine is given in terms of the current state, the remaining input string, the output string generated, and the current contents of the left and right stacks. Let $(q : i0 \ i1 : o0 : a0 \ a1 : b0 \ b1)$ be the state of a 2PDA and $(q : i0 \ a1 : b1 : p : o1 : a2 : b2)$ be a production rule for the E2PDA, then the next state of the machine would be $(p : i1 : o0 \ o1 : a0 \ a2 : b0 \ b2)$ where $q$ and $p$ are states and all the other elements are strings of symbols. The relation, $R$, for a machine, $M$, is the set of input output pairs such that for a given element of the set the input is accepted by the machine and the output is generated.

The theorems in this chapter prove that the class of languages generated by
an unrestricted grammar is exactly the same class of languages as those accepted by E2PDA. This result would be expected since E2PDA is an extension of 2PDA which is known to be equivalent to Turing machines. Turing machines accept exactly those languages generated by an unrestricted grammar. Proof of this can be found in Hopcroft and Ullman Theorems 9.3 and 9.4\textsuperscript{1}. The theorems given in this chapter of the paper are modeled after the theorems in Hopcroft and Ullman\textsuperscript{1}.

The first theorem constructs a grammar which models the actions of an E2PDA with no input or output tapes. In this case the input is positioned on the right stack initially and no output is generated since the machine acts as an accepter. The essential details of the proof are that the grammar nondeterministically generates two copies of a string and simulates the actions of the machine on one of the copies. If the machine accepts the string, then the grammar removes one of the copies and generates a terminal string. If the machine does not accept the string, then the grammar never generates a terminal string. The copy of the strings that the grammar acts upon has the state of the machine embedded in it and the grammar provides productions simulating the actions of the machine on this copy. If a final state is entered, then the grammar provides production rules which remove the manipulated copy, the embedded state, and terminate with the final string accepted by the machine.

The second theorem constructs a machine which simulates the production rules of an unrestricted grammar. The idea of the construction is to use the ability of the E2PDA to map strings of symbols to strings of symbols to simulate the productions of the grammar. The machine has an input tape which contains the input string. Initially the right stack contains the start symbol of the grammar. Then the machine repeatedly performs the following routine. First it nondeterministically shifts some number of elements from the right stack to the left stack. Then it
nondeterministically selects a grammar rule to apply. If the grammar rule matches
the top of the right stack, then the machine replaces that portion of the stack which
matches the left hand side of the production rule with the right hand side of the
production. It then shifts the elements from the left stack back to the right stack.
If the input matches the string of symbols on the right stack, the machine accepts
the string. If not, the machine repeats the routine.

Definition of an E2PDA

An E2PDA Transducer has the following characteristics:
1) A finite number of states \( \text{state} \),
2) two push down stacks \( \text{stack}_i \text{ and stack}_r \),
3) an input tape which is read only and contains the finite input for the E2PDA
\( \text{input} \), and
4) an output tape which is write only and contains the output generated by the
E2PDA \( \text{output} \).

An E2PDA is formally denoted by:
\[ M = (Q, I, O, \delta, S, B, q_0, F) \]
where
\( Q \) is the finite set of states,
\( I \) is the finite set of input symbols,
\( O \) is the finite set of output symbols,
\( S \) is the finite set of stack symbols,
\( B \) is the blank symbol (an element of \( I, O, S \)),
\( q_0 \) is the start state, and
\( F \) is the set of final accepting states.

The production rules, \( \delta \), are mappings
\[ Q \times I^* \times S^* \times S^* \rightarrow Q \times O^* \times S^* \times S^*. \]

An instantaneous description of the current state of an E2PDA is denoted by

\[(q_i : i : o : a : b),\]

where

\[q_i \in Q,\]
\[i \in I^*,\]
\[o \in O^*,\]
\[a, b \in S^*.\]

Let \((q_i : i_0 : a_1 : b_1)\) be an instantaneous description of an E2PDA.

Let \((q_i : i_0 : a_1 : b_1 : q_j : o_1 : a_2 : b_2)\) be a production rule for the E2PDA.

Then the next state of the E2PDA is \((q_j : i_1 : o_0 a_0 : a_0 a_2 : b_0 b_2)\).

This is denoted as

\[(q_i : i_0 i_1 : o_0 : a_0 a_1 : b_0 b_1) \rightarrow (q_j : i_1 : o_0 a_1 : a_0 a_2 : b_0 b_2),\]

where

\[q_i, q_j \in Q,\]
\[i_0, i_1 \in I^*,\]
\[o_0, o_1 \in O^*,\]
\[a_0, a_1, o_0, a_2, b_0, b_1, b_2 \in S^*.\]

\[\omega \in \mathcal{L}(M), (\omega, \phi) \in \mathcal{R}(M) \text{ iff } (q_0 : \omega : \epsilon : B : B) \overset{*}{\rightarrow} (q_i : \epsilon : \phi : \alpha_0 : \alpha_1),\]

where

\[q_i \in F,\]
\[\omega \in I^*,\]
\[\phi \in O^*,\]
\[\alpha_0, \alpha_1 \in S^*.\]
Theorem I

If $L$ is $L(M)$ for an $E2PDA$, then $L$ is $L(G)$ for an unrestricted grammar $G$.

Proof:

Let $M = (Q, \delta, S, B, q_0, F)$.

Construct $G = (V, S, P, A0)$.

$V = (A0, A1, A2, A3) \cup Q \cup S'$, the productions in $P$ are:

1) $A0 \rightarrow A1 A2$,
2) $A1 \rightarrow B A1$,
3) $A1 \rightarrow \epsilon$,
4) $A2 \rightarrow a' A2 a \quad \forall a \in S^*$,
5) $A2 \rightarrow q_0 A3$,
6) $A3 \rightarrow A3 B$,
7) $A3 \rightarrow \epsilon$,
8) $y_1 ... y_l q z'_l ... z'_m \rightarrow t'_l ... t'_1 p z'_1 ... z'_k \quad \forall (q, x, ..., z, y_1, ..., y_l : p, x, ..., z, t_1, ..., t_l) \in \delta$,
9) $X' q \rightarrow q \quad \forall X \in S, \forall q \in F$,
10) $q X' \rightarrow q \quad \forall X \in S, \forall q \in F$, and
11) $q \rightarrow \epsilon \quad \forall q \in F$.

Grammar $G$ nondeterministically generates two copies of a string in $S^*$ then simulates the action of $M$ on one copy. If $M$ accepts the string, then $G$ converts the second copy to a terminal string. If $M$ does not accept the string, then the derivation never results in a terminal string.

Using rules 1 – 7 the initial state of the machine is generated.

If $M$ accepts $a_1 a_2 a_3 ... a_n$ then for some $m_1, m_2$, $M$ uses no more than $m_1$ cells from the bottom of $stack_r$ and no more than $m_2$ cells from the bottom of $stack_l$.

The string which models the initial state of the machine is
$B^{m_1}a_n'a_{n-1}...a_3'a_2'a_1'q_0B^{m_2}a_1a_2a_3...a_n$.

It can be shown by the number of moves made by $M$ that if

$$(q_0 : \varepsilon : \alpha^{-1}) \vdash (q : X_0 : X_1),$$

then

$$B^{m_1}a^{-1}q_0B^{m_2}a \triangleright X_1qX_0^{-1}.$$  

Assume $\alpha, X_i, Y_i \in S^\star$.

Let

$$(g_0 : \varepsilon : \alpha^{-1}) \overset{k-1}{\vdash} (q : X_0X_1 : X_2X_3) \vdash (p : X_0X_4 : X_2X_5)$$

and

$$B^{m_1}a^{-1}q_0B^{m_2}a \overset{k-1}{\triangleright} X_1X_2^{-1}X_3^{-1}.$$  

By rule 8,

$$(q; X_1; X_2 : p; X_4; X_5)$$

implies

$$X_3qX_1^{-1} \rightarrow X_5pX_4^{-1}.$$  

So

$$X_2X_3qX_1^{-1}X_0^{-1} \Rightarrow X_2X_5pX_4^{-1}X_0^{-1}.$$  

Therefore,

$$(g_0 : \varepsilon : \alpha^{-1}) \overset{k}{\vdash} (p : X_0X_4 : X_2X_5)$$

implies

$$B^{m_1}a^{-1}q_0B^{m_2}a \overset{k}{\triangleright} X_2X_5pX_4^{-1}X_0^{-1}.$$  

Let

$$(g_0 : \varepsilon : \alpha^{-1}) \vdash (p : Y_0 : Y_1), p \in F.$$  

Then

$$B^{m_1}a^{-1}q_0B^{m_2}a \triangleright Y_1pY_0^{-1}a.$$  

By rules 9, 10, and 11; if $p \in F$, then

$$Y_1pY_0^{-1}a \Rightarrow a.$$
Therefore, \( \alpha \in L(G) \).

**Theorem II**

If \( L \) is \( L(G) \) for unrestricted grammar \( G = (V, T, P, S) \), then \( L \) is \( L(M) \) for an E2PDA \( M \).

**Proof:**

Construct an E2PDA automaton \( M \) to recognize \( L \).

Let \( M = (Q = \{ q_0, q_1, q_2, q_3, q_4, q_5 \}, I = T, \delta, S = V \cup T \cup \{ $, B \}, B, q_0, F = \{ q_6 \} \) \), the transition rules in \( \delta \) are:

1. \((q_0;\ldots; q_1; $$; $$, S)\),
2. \((q_1;\ldots; a: q_1; a) \quad \forall a \in V \cup T \cup \{ B \}, \)
3. \((q_1;\ldots; q_2; i)\),
4. \((q_2;\ldots; ^{-1}: q_3; \gamma ^{-1}) \quad \forall \beta \rightarrow \gamma \in P, \)
5. \((q_3; a: q_3; a) \quad \forall a \in V \cup T \cup \{ B \}, \)
6. \((q_3; $$; q_4; $$)\),
7. \((q_4;\ldots; q_1; i)\), and
8. \((q_4; \omega; $$; \omega ^{-1} : q_5; ;)\).

Input contains \( \omega \in T^* \). \( Stack_l \) and \( stack_r \) contain the sentential forms \( \alpha \) of \( G \). \( M \) initializes \( stack_r \) to \( S \).

Then \( M \) repeatedly does the following:

1. Nondeterministically shifts \( i \) elements from \( stack_r \) to \( stack_l \) such that any \( 0 \leq i \leq |\alpha| \) may be chosen;
2. nondeterministically selects a grammar rule \( \beta \rightarrow \gamma \) to apply;

If the grammar rule matches the top of \( stack_r \), then replaces the top \( |\beta| \) elements of \( stack_r \) with \( \gamma \).

3. shifts the elements from \( stack_l \) to \( stack_r \); and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
4) compares \( \omega \) with \( \alpha \). If they match, then accepts; \( \omega \in L(G) \).

Otherwise returns to step 1.

Example

Let \( G \) be the grammar for the example in chapter 2.

Construct an E2PDA to determine if \( aabbcc \) is a string in the grammar \( G \).

Let \( M = (Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}, I = \{a, b, c\}, \delta, S = \{S, X, Y, Z, A, C, a, b, c, \$, B\}, B, q_0 \)

\( F = \{q_5\} \).

The transition rules in \( \delta \) are:

1) \( (q_0; ; ; ; q_1; \$; \$, S), \)
2) \( (q_1; ; ; ; S : q_1; S), \)
3) \( (q_1; ; ; ; X : q_1; X), \)
4) \( (q_1; ; ; ; Y : q_1; Y), \)
5) \( (q_1; ; ; ; Z : q_1; Z), \)
6) \( (q_1; ; ; ; A : q_1; A), \)
7) \( (q_1; ; ; ; C : q_1; C), \)
8) \( (q_1; ; ; ; a : q_1; a), \)
9) \( (q_1; ; ; ; b : q_1; b), \)
10) \( (q_1; ; ; ; c : q_1; c), \)
11) \( (q_1; ; ; ; B : q_1; B), \)
12) \( (q_1; ; ; ; q_2; ; ), \)
13) \( (q_2; ; ; ; S : q_3; ; Y, Z, b, X), \)
14) \( (q_2; ; ; ; S : q_3; ; ), \)
15) \( (q_2; ; ; ; Z : q_3; ; C, Z, b, A), \)
16) \( (q_2; ; ; ; Z : q_3; ; ), \)
17) \( (q_2; ; ; ; A, b : q_3; ; b, A), \)
The initial state of the machine is

\((q_0 : a, a, b, b, c, c : B : B)\).

Intermediate states of the machine before and after grammar rule productions are:

\((q_2 : a, a, b, b, c, c : B, \$, B, \$, S)\),

\((q_2 : a, a, b, b, c, c : B, \$, X, b : B, \$, Y, Z)\),

\((q_2 : a, a, b, b, c, c : B, \$, X, b, A, b : B, \$, Y, C, Z)\),

\((q_2 : a, a, b, b, c, c : B, \$, X : B, \$, Y, C, b, A, b)\),

\((q_2 : a, a, b, b, c, c : B, \$, b : B, \$, Y, C, b, b, A, X)\),

\((q_2 : a, a, b, b, c, c : B, \$, a : B, \$, Y, C, b, b, X)\).
(q₃ : a, a, b, b, c : B, $, a, a, b, b : B, $, Y, C),
(q₃ : a, a, b, b, c : B, $, a, a, b, b : B, $, c, Y), and
(q₄ : a, a, b, b, c : B, $ : B, $, c, c, b, b, a, a).

The sequence of grammar production rules applied in the derivation sequence is;

13, 15, 16, 17, 18, 19, 20, 21.

The terminal state of the machine is

(q₅ : B, $ : B).
CHAPTER IV

SDT TRANSDUCTION USING E2PDA

This chapter demonstrates that E2PDA can effectively model the transduction of SDT Normal Form Grammars. In a construction similar to the last proof in chapter 3, the mechanical details of syntax-directed transduction by an E2PDA are presented. Elements of the construction which are different from those in the second proof in chapter 3 are the motions of the curser, the removal of non-determinism, and the use of SDT transduction as the input output relationship of the machine. The E2PDA constructed in this chapter models interactive entry of directives and generates trace output which provides the user with a method of selecting the specific sequence of derivation steps and thus a deterministic generation of the final strings of a language. Therefore, the construction is a derivation of an E2PDA to accept the syntax-directed directives of a grammar and generate the resulting strings of that grammar. The actions of the machine constructed for a given grammar are as follows. The machine initializes the right stack to the start symbol of the grammar with a curser prefix and repeatedly performs the following routine. It moves all of the symbols from the left stack to the right stack. Then it moves all of the symbols from the right stack back to the left stack and also copies the symbols to the output as a record of the next sentential form generated in the derivation of the final string. Next it appends a separator marker to the output to denote the end of the sentential form and moves the symbols from the left stack to the right stack until the curser marker is again positioned at the top of the right stack. Finally it applies one of the directives. The directives will be either to apply production number i, move the marker one symbol left or right in the sequence of symbols of the sentential form, or halt and remove the curser. If the directive was
an application of a production, then replace the portion of the right stack matching
the left hand side of the production with the right hand side of the production. If
the directive was a curser move, then reposition the curser one position to the left
or right of the current location. If the directive was a halt, then if the sentential
form contains only terminal symbols for the grammar, then output the final string
and accept. If the directive was a halt but nonterminals are still present in the
sentential form, then reject the input. If the directive was not a halt, then repeat
the routine again.

Construction

If R is $R_{nlt}(G)$ for unrestricted grammar, G, then an E2PDA automaton M can
be constructed s.t. R is $R_{nlt}(M)$.

Let $G = (V, T, P, S)$ be an unrestricted grammar with productions numbered 1
- m.

Construct an E2PDA M to transduce R.

Let $M = (Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_7\}, I = \{L, R, H, j\epsilon 1..m\}, O = V \cup T \cup \{\#, \}, \delta,
S = V \cup T \cup \{\%, \$, B\}, B, q_0, F = \{q_7\}$, the transition rules in $\delta$ are:

1) $(q_0; ;; q_1; ;; \%; \%; S; \$),$
2) $(q_1; ;; q_2; ;; \%),$
3) $(q_1; a; q_1; ;; a) \ \forall a \in V \cup T \cup \{\},$
4) $(q_2; ;; \%; q_3; ;; \%),$
5) $(q_2; a; q_2; a; a) \ \forall a \in V \cup T \cup \{\},$
6) $(q_3; ;; q_4; ;; \$),$
7) $(q_3; a; q_3; ;; a) \ \forall a \in V \cup T,$
8) $(q_4; j; ;; \beta^{-1}; \$; q_4; ;; \gamma^{-1}; \$) \ \forall \beta \rightarrow \gamma \in P,$
9) $(q_4; L; a; \$; q_4; ;; a; \$) \ \forall a \in V \cup T,$
10) \((q_4; R; \alpha, \Sigma; q_i; \alpha; \Sigma) \quad \forall \alpha \in V \cup T,\)

11) \((q_4; H; \Sigma; q_0; \Sigma),\)

12) \((q_0; \%; q_0; \%),\)

13) \((q_0; \alpha; q_0; \alpha) \quad \forall \alpha \in T,\)

14) \((q_0; \%; q_7; \#; \%),\) and

15) \((q_0; \alpha; q_0; \alpha) \quad \forall \alpha \in T.\)

*Input* contains \(\omega \in \Sigma^*\). *Output* contains \(\phi \in \Omega^*\). *Stack* and *stackr* contain the sentential forms, \(\alpha,\) of \(G'\), the SDT Normal Form Grammar of \(G\). *M* initializes *stackl* to \(\%\) and *stackr* to \(\%, S, \$.\)

Then *M* does the following:

1) Outputs the next \(\alpha\) from the derivation sequence;

2) repositions the marker, \(\$\), to the top of *stackr*; and

3) applies one of the following directives.

If the directive is an application of a production, then it replaces the top \(|$X|\)
elements of *stackr* with \(\$Y\). \(X \rightarrow Y \in P.\)

If the directive is a move, then it repositions the marker, \(\$.\)

If the directive is a halt, then it removes the marker, \(\$,\) and goes to step 4.

Otherwise, it returns to step 1.

4) Outputs the terminal string \(\alpha.\)

**Example**

Let \(G\) be the grammar for the example in chapter 2.

Construct an E2PDA *M* to transduce the relation \(R\) given by \(G_{s_{\Sigma\#}}\).

Let \(M=(Q=q_0, q_1, q_2, q_3, q_4, q_5, q_7), I=\{L, R, H, j \in \{1, 9\}\}, O=\{S, X, Y, Z, A, C, a, b, c, \$, \#, \}, \delta, S=(S, X, Y, Z, A, C, a, b, c, \%, \$, B), B, q_0, F=\{q_7\}).\)

The transition rules in \(\delta\) are:
1) \((q_0; \cdot q_1; \cdot \times; \cdot \times; S, \cdot S)\),
2) \((q_1; \cdot \times; q_2; \cdot \times)\),
3) \((q_1; S; \cdot q_1; \cdot S)\),
4) \((q_1; X; \cdot q_1; \cdot X)\),
5) \((q_1; Y; \cdot q_1; \cdot Y)\),
6) \((q_1; Z; \cdot q_1; \cdot Z)\),
7) \((q_1; A; \cdot q_1; \cdot A)\),
8) \((q_1; C; \cdot q_1; \cdot C)\),
9) \((q_1; a; \cdot q_1; \cdot a)\),
10) \((q_1; b; \cdot q_1; \cdot b)\),
11) \((q_1; c; \cdot q_1; \cdot c)\),
12) \((q_1; \cdot S; \cdot q_1; \cdot S)\),
13) \((q_2; \cdot S; \cdot q_2; \cdot \#; \cdot \%)\),
14) \((q_2; S; \cdot q_2; S; S)\),
15) \((q_2; X; \cdot q_2; X; X)\),
16) \((q_2; Y; \cdot q_2; Y; Y)\),
17) \((q_2; Z; \cdot q_2; Z; Z)\),
18) \((q_2; A; \cdot q_2; A; A)\),
19) \((q_2; C; \cdot q_2; C; C)\),
20) \((q_2; a; \cdot q_2; a; a)\),
21) \((q_2; b; \cdot q_2; b; b)\),
22) \((q_2; c; \cdot q_2; c; c)\),
23) \((q_3; \cdot S; \cdot q_2; \cdot S)\),
24) \((q_3; \cdot S; \cdot q_4; \cdot S)\),
25) \((q_3; S; \cdot q_3; \cdot S)\),
26) \((q_3; X; \cdot q_3; \cdot X)\),

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
27) \((q_3; Y; q_3; Y)\),
28) \((q_3; Z; q_3; Z)\),
29) \((q_3; A; q_3; A)\),
30) \((q_3; C; q_3; C)\),
31) \((q_3; a; q_3; a)\),
32) \((q_3; b; q_3; b)\),
33) \((q_3; c; q_3; c)\),
34) \((q_4; 1; S, $: q_1; Y, Z, b, X, $)\),
35) \((q_4; 2; S, $: q_1; $)\),
36) \((q_4; 3; Z, $: q_1; C, Z, b, A, $)\),
37) \((q_4; 4; Z, $: q_1; $)\),
38) \((q_4; 5; A, b, $: q_1; b, A, $)\),
39) \((q_4; 6; A, X, $: q_1; X, a, $)\),
40) \((q_4; 7; X, $: q_1; a, $)\),
41) \((q_4; 8; Y, C, $: q_1; c, Y, $)\),
42) \((q_4; 9; Y, $: q_1; c, $)\),
43) \((q_4; L; S, $: q_1; S, $)\),
44) \((q_4; L; X, $: q_1; X, $)\),
45) \((q_4; L; Y, $: q_1; Y, $)\),
46) \((q_4; L; Z, $: q_1; Z, $)\),
47) \((q_4; L; A, $: q_1; A, $)\),
48) \((q_4; L; C, $: q_1; C, $)\),
49) \((q_4; L; a, $: q_1; a, $)\),
50) \((q_4; L; b, $: q_1; b, $)\),
51) \((q_4; L; c, $: q_1; c, $)\),
52) \((q_4; R; S, $: q_1; S, $)\),

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The generation of a a b b c c.


The initial state of the machine is

\[(q_0 : 1, R, 3, R, R, 4, L, L, 5, L, 6, R, 7, R, R, 8, 9, H : B : B).\]

Intermediate states of the machine before and after grammar rule productions are;


The sequence of grammar production rules applied in the derivation is:

\[ B, \%, X : B, \%, Y, Z, b, \$ \),


\[(q_4 : L, 5, L, 6, R, 7, R, R, 8, 9, H : O_8, X, b, A, b, \$, C, Y, \# = O_9 : B, \%, X, b, A : B, \%, Y, C, b, \$),\]

\[(q_4 : 5, L, 6, R, 7, R, R, 8, 9, H : O_9, X, b, A, b, \$, C, Y, \# = O_{10} : B, \%, X, b : B, \%, Y, C, b, A, \$),\]

\[(q_4 : L, 6, R, 7, R, R, 8, 9, H : O_{10}, X, b, A, b, \$, C, Y, \# = O_{11} : B, \%, X : B, \%, Y, C, b, A, \$),\]

\[(q_4 : 6, R, 7, R, R, 8, 9, H : O_{11}, X, b, A, b, \$, C, Y, \# = O_{12} : B, \%, X : B, \%, Y, C, b, b, A, \$),\]

\[(q_4 : R, 7, R, R, 8, 9, H : O_{12}, X, b, b, b, C, Y, \# = O_{13} : B, \%, X : B, \%, Y, C, b, b, A, \$),\]

\[(q_4 : R, 7, R, R, 8, 9, H : O_{13}, X, b, b, b, C, Y, \# = O_{14} : B, \%, X : B, \%, Y, C, b, b, X, \$),\]

\[(q_4 : 7, R, R, R, 8, 9, H : O_{14}, a, b, b, C, Y, \# = O_{15} : B, \%, a : B, \%, Y, C, b, b, X, \$),\]

\[(q_4 : R, R, R, 8, 9, H : O_{15}, a, b, b, C, Y, \# = O_{16} : B, \%, a : B, \%, Y, C, b, b, a, \$),\]

\[(q_4 : R, 8, 9, H : O_{16}, a, b, b, C, Y, \# = O_{17} : B, \%, a : B, \%, Y, C, b, b, \$),\]

\[(q_4 : R, 8, 9, H : O_{17}, a, b, b, C, Y, \# = O_{18} : B, \%, a : B, \%, Y, C, b, \$),\]

\[(q_4 : 8, 9, H : O_{18}, a, b, b, C, Y, \# = O_{19} : B, \%, a : B, \%, Y, C, b, \$),\]

\[(q_4 : 9, H : O_{19}, a, b, b, C, Y, \# = O_{20} : B, \%, a : B, \%, Y, C, b, \$),\]

\[(q_4 : H : O_{20}, a, b, b, C, Y, \# = O_{21} : B, \%, a : B, \%, Y, C, \$).\]
The terminal state of the machine is

$$(q_f : O_{21}, a, a, b, b, c, c, \# = O_{22} : B, \%, a, a, b, b, c, c : B, \%)$$.
CHAPTER V

EXTENSIONS

The use of E2PDA as an effective model of syntax-directed transduction may be extended to incorporate the concept of semantic routines associated with production rules as they are used in compilers. The routines of a compiler are divided into two segments: a syntax analyzer which derives an implicit derivation tree of the terminal strings of a grammar, and the semantic routines associated with each of the production rules of the grammar which are performed as each of the production rules is applied. Since the production rules of an E2PDA syntax-directed transducer may be formulated to correspond to the production rules of a grammar, it is natural to extend the E2PDA to a generator compiler with semantic stacks paired with the syntax stacks, semantic routines to be performed when a production is applied, symbol tables, and attribute evaluation rules.

The extension of a grammar to incorporate attributes connected to the symbols of the grammar is referred to as an attributed grammar. The normal use of attributed grammars is as a way of restricting the elements of a language to those which, in addition to being generated by the production rules of the grammar, satisfy a set of auxiliary functions computed on the attributes of the symbols in a derivation. Each of the symbols which appear in a production rule has a set of attributes which are computed from functions of the values of attributes of its superior and subordinate nodes in the derivation diagram. Inherited attributes are those which are computed from the values of superior nodes, and synthesized attributes are those which are computed from the values of subordinate nodes. The mechanism used for maintaining the values of the attributes of the symbols is a semantic stack. Each of the cells of the semantic stack is paired with a cell of the
syntax stack, and the value of the symbol found in a given cell of the syntax stack is the value found in the corresponding cell of the semantic stack. A notable use of attribute grammars is in the generation of the output language of a compiler.

With attribute grammars, values must be explicitly moved from node to node of the derivation tree by passing them through semantic values managed by the semantic stack. Parsers for topdown deterministic context-free grammars construct an implicit derivation tree of an input string with a left to right scan of the input and a leftmost derivation. Alternatively, this type of parser is said to perform a topdown parsing of the input. Since the values of the syntax stack are derived by topdown generation and the evaluation of attributes for a symbol is performed only when the production of that symbol is performed, only inherited attributes may be generated by a topdown parser which uses only attribute evaluation rules. Note that attributed grammars are not defined in the context of parsing of the languages generated by their production rules. Rather, attribute grammars define a set of mutually necessary conditions for a string to be part of a language; the attribute evaluation functions define the set of conditions to be satisfied by the component symbols.

The limitation of topdown generation to inherited attributes is a function of defining a derivation sequence and requiring that evaluation of attributes occurs only when the relevant productions are applied. It is for this reason that symbol tables are used by parsers. Symbol tables provide a mechanism for transferring attributes from one subtree to another when the attributes for their common superior node have previously been evaluated in the course of topdown parsing thereby providing no route for transmission of attributes between the two subtrees. A symbol table is a table of values maintained separately from the values maintained by the semantic stack. In normal use, pointers from the semantic stack are directed
to values held in the symbol table. This strategy is used as a way of permitting both inherited and synthesized attributes to be evaluated by a topdown parser. During each of the derivation steps in the generation of a string of a grammar, the corresponding derivation tree component is appended to the full derivation tree being generated in the symbol table. Once the final string has been generated, the inherited and synthesized attributes are evaluated since the full derivation tree is available for this purpose.

Syntax-directed compilers based upon the syntax-directed transduction of an attributed unrestricted grammar would be analogous to topdown parsing compilers. In a scheme similar to that of topdown semantic routines for topdown deterministic context-free parsers, the syntax-directed compiler would be based upon inherited attributes and the production rules for the grammar are augmented to include associated semantic routines which evaluate the values of the attributes of the symbols. Although the use of unrestricted grammars makes it possible for attribute values to be transmitted from any location to subordinate nodes in a derivation diagram and therefore attributes may be inherited, it is not possible for attributes to be synthesized by a syntax-directed compiler without symbol tables because the semantic routines must be performed in the same order as the generation of symbols and synthesized attributes must be generated in the opposite order. This is the same situation as with the topdown parser for a deterministic context-free grammar. However, by generating attribute evaluation rules which construct a symbol table which maintains a representation of the derivation diagram for a string, and by having the semantic stack values be pointers to the occurrence of the syntax symbols on the constructed derivation diagram, syntax-directed compilers can be devised which provide for evaluation of both inherited and synthesized attributes of symbols. The compiler would construct the derivation diagram and whatever at-
tribute evaluations are appropriate during the generation phase of a string would be performed. During a termination phase the evaluation of the remaining attributes would be performed. The mechanism used to construct the derivation diagram is as follows. When a production rule is applied to the sentential form maintained by the syntax stack, a corresponding addition is made to the derivation diagram. The values maintained by the semantic stack are pointers to the corresponding element in the derivation diagram. A new set of nodes for the diagram is generated which represents the symbols on the right hand side of the production. These nodes have a parent pointer which points to the leftmost node in the set of nodes which comprise their parents. The parent nodes have sibling pointers which point to their right sibling in the derivation step. In the reverse direction, the parent nodes have children pointers which point to the leftmost child in the derivation. The child nodes have sibling pointers which point to rightmost sibling in the derivation step. The derivation diagrams constructed in this manner permit traversing the diagram in any desired direction and facilitate the evaluation of attributes associated with each of the symbols on the diagram. The values of the attributes would be maintained as additional values of the nodes representing the symbols of the derivation diagram.
CHAPTER VI

CONCLUSIONS

In conclusion, this paper has developed a strategy for using E2PDA as the basis for syntax-directed transducers. The construction of an E2PDA automaton for syntax-directed transduction from a grammar is effectively performed and the resulting production rules for the machine define actions which model the actions of a syntax-directed transduction normal form grammar. The extension of the 2PDA model of computation to multisymbol matching production rules has been shown to produce a one-to-one mapping of production rules for the grammar to productions modeling grammar rule applications by the machine.

The methodology used by compilers based upon parsers has been converted to one which has syntax-directed translators as the foundation. The generation of derivation trees and evaluation of attribute grammars with a syntax-directed translation compiler is accomplished in the same way as it is for parsing compilers by the use of semantic stacks, semantic routines, and symbol tables.

Further work in this area consists of developing a generator for syntax-directed transduction compilers from specifications of grammar rules and their attribute evaluation rules. The format of the generator would be expected to be similar to that of compiler generators now in use. The generator would produce a program which would simulate the actions of the production rules as they are formulated by the construction of an E2PDA syntax-directed transducer from a grammar. Exact modeling of the individual steps of the machine are not necessary but rather the replication of the major states of the machine as they are generated is sufficient. These steps include generating output for the current state of the sentential form as each directive is accepted, recognition of directives with appropriate modification.
of the sentential form, and performance of the correct semantic routine during application of a production. Additional work is in the area of finding appropriate applications for the mechanism defined in the paper.

The area of application for transducer compilers generated by the methodology developed in this paper is confined to those situations where the nature of the productions of a grammar and their associated attribute evaluation rules require an attributed unrestricted grammar. The existence of such grammars is expected in the future and the strategy developed by this paper should provide a reasonable way of working with these grammars when they are discovered.
REFERENCES

BIBLIOGRAPHY


