CONTEXT-FREE GRAMMARS ON TREES

William C. Rounds
Case Western Reserve University
Cleveland, Ohio

Summary

In this paper we discuss still another version of indexed grammars[1] and macro grammars[3], gaining some geometric intuition about the structure of these systems. An ordinary context-free grammar is a rewriting system for strings; we find that a macro grammar is a rewriting system for trees. CF grammars on strings form a special case since strings can be thought of as trees without branching nodes.

We consider the special case of finite-state grammars in this report. We define the tree analogue of a nondeterministic generalized sequential machine and obtain results about the domain and range of such a mapping. We relate these results to the theory of generalized finite automata[6].

Introduction

Let us consider a generalized sequential machine as a special sort of grammatical system. Suppose, for example, that for some state q and input symbol σ, the next state is q' and the next output word is w, where $\sigma \in \Sigma$, the input alphabet, and $w \in \Sigma^*$. We write this information in the form of a production

$$(q,\sigma) \to w(q').$$

As an example, consider the homomorphism which doubles every letter of the input word; we have a state-set $Q = \{q\}$, alphabet $\Sigma = \{\sigma,\tau\}$ and a set of productions P:

$$(q,\sigma) \to \sigma\sigma(q)$$
$$(q,\tau) \to \tau\tau(q)$$

As a "derivation" we have

$$(q)\sigma\tau\sigma \Rightarrow \sigma\sigma(q)\tau\sigma \Rightarrow \sigma\sigma\tau\tau(q)\sigma \Rightarrow \sigma\sigma\tau\tau\sigma\sigma(q).$$

To be precise, if we want the result to be free of the state at the end, we would need to tack an end marker # to the input string $\sigma\tau\sigma$ and introduce a production $(q,\#) \to \#$ to get rid of the state. Let us skip the details of beginning and ending a derivation, however, to see how to generalize our gsm to work on more general input-output structures. In particular, we will obtain mappings from trees to trees.

The basic idea is to let the input symbols, which are in a sense nonterminal, have arguments. This is done by Fischer[3] for macro grammars (without input). Each input symbol σ will have a fixed number of arguments $r(\sigma)$ associated with it. In particular, the end marker symbols will have no arguments. To specify our input set, we need a finite alphabet Σ and a function $r: \Sigma \to \omega$. Set

$$\Sigma_n = \{\sigma \in \Sigma \mid r(\sigma) = n\}$$

We define the input set $\mathcal{T}_\Sigma^o$ inductively: $\mathcal{T}_\Sigma^o$ is the smallest set containing Σ_o and such that whenever $t_o,\ldots,t_{n-1} \in \mathcal{T}_\Sigma^o$ and $\sigma \in \Sigma_n$ then $\sigma(t_o,\ldots,t_{n-1}) \in \mathcal{T}_\Sigma^o$.

<u>Example</u>. Let $\Sigma_1 = \{\sigma,\tau\}$, $\Sigma_o = \{\#\}$. Then $\sigma(\tau(\sigma(\#)))$ is an element of $\mathcal{T}_\Sigma^o$. We can visualize this element as the string $\sigma\tau\sigma\#$.

<u>Example</u>. Let $\Sigma_2 = \{\rho\}$, $\Sigma_1 = \{\sigma,\tau\}$, $\Sigma_o = \{\#,\lambda\}$. Then

$$t = \rho(\rho(\sigma(\#),\tau(\lambda)),\sigma(\tau(\lambda)))$$

is an element of $\mathcal{T}_\Sigma^o$. We can visualize t as a wide string:

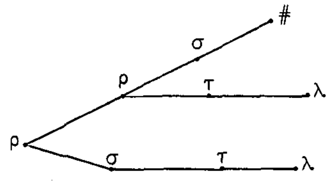


Fig. 1

Now we are going to define productions, as in the case of the gsm above, which will allow us to rewrite this tree from top to bottom (in Fig. 1, the top is at the left). Initially, we shall rewrite the symbol ρ as a small tree. Since ρ has two arguments, the production must tell us how to substitute these arguments into the small tree we get from rewriting ρ. It also must tell us the proper next-state transitions. We thus get a picture of a production as in Fig. 2.

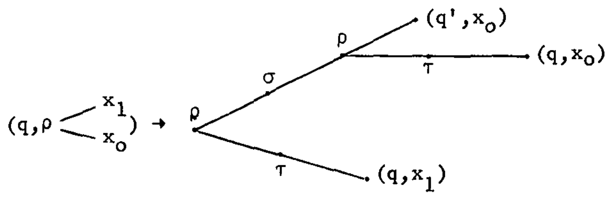


Fig. 2

To apply this production to the tree in Fig. 1, replace ρ by the right-hand side of the production, substitute the lower subtree in Fig. 1 for x_0 and the upper one for x_1 in the right-hand side. The result appears in Fig. 3.

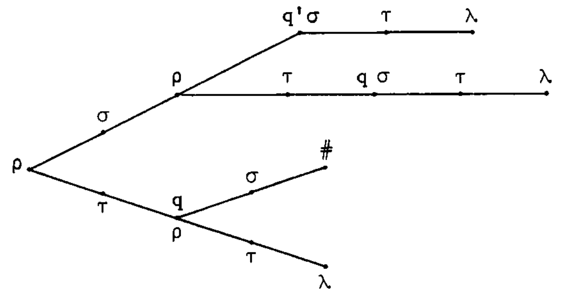


Fig. 3

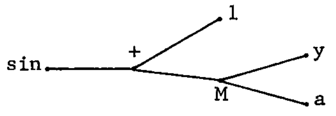
Now we are ready to process the subtrees of our original tree, starting in the states indicated in the branches of the tree in Fig. 3. Notice that the lower subtree σ-τ-λ was duplicated in the process of applying the given production. The feature of duplication will make tree rewriting systems stronger than context-free grammars.

<u>Examples</u>. (i) <u>The identity mapping</u>. Here we need a singleton state set $Q = \{q\}$, and productions

$$(q,\sigma(x_0,\ldots,x_{n-1})) \to \sigma((q,x_0),\ldots,(q,x_{n-1}))$$

for $\sigma \in \Sigma_n$ $(n > 0)$. For $\lambda \in \Sigma_0$, we need a production $(q,\lambda) \to \lambda$.

(ii) <u>Formal differentiation</u>. Let $\Sigma_0 = \{0,1,a,y\}$, $\Sigma_1 = \{\sin,\cos,\text{neg}\}$, $\Sigma_2 = \{M,+\}$. The tree



represents the term sin(ay + 1). We wish to differentiate this term with respect to y. Construct a set of productions to do the job syntactically: first let $Q = \{d,i\}$ be the set of states. d is the "differentiation" state and i is the "identity" state. The initial state will be d. The productions follow:

(a) $(d,a) \to 0$; $(d,0) \to 0$; $(d,1) \to 0$; $(d,y) \to 1$.

(the derivative of y is 1; the derivative of a constant is zero.)

(b) $(d,\sin(x)) \to M((d,x),\cos((i,x))$

$(d,\cos(x)) \to M((d,x),\text{neg}(\sin((i,x))))$

$(d,\text{neg}(x)) \to \text{neg}((d,x))$

(the chain rule.)

(c) $(d,+(x_0,x_1)) \to +((d,x_0),(d,x_1))$

$(d,M(x_0,x_1)) \to +(M((i,x_0),(d,x_1)),$
$M((d,x_0),(i,x_1)))$

(the sum and product rules.)

(d) productions $(i,\sigma(x_0,x_1)) \to \sigma((i,x_0),(i,x_1))$ just as in the first example.

<u>Formal Definitions</u>

In this section we formalize the ideas above.

Let X be a countable set of symbols $\{x_0,x_1,\ldots\}$. X is the set of <u>variables</u>. We wish to label the trees occurring on the right hand sides of productions with pairs (q,x) where q is a state and $x \in X$. This can be accomplished by the following general inductive definition.

<u>Definition</u>. Let A be a set. The set $\mathcal{T}_\Sigma(A)$ of terms indexed by A is the smallest set (a) containing $A \cup \Sigma_0$ and (b) such that whenever $\sigma \in \Sigma_n$ and $t_0,\ldots,t_{n-1}$ are in $\mathcal{T}_\Sigma(A)$ then $\sigma(t_0,\ldots,t_{n-1}) \in \mathcal{T}_\Sigma(A)$.

Thus $$\mathcal{T}_\Sigma^o = \mathcal{T}_\Sigma(\Sigma_o) = \mathcal{T}_\Sigma(\emptyset) .$$

<u>Definition</u>. A (finite-state) production is a pair $((q,\sigma(x_0,\ldots,x_{n-1}),s)$, written $(q,\sigma(x_0,\ldots,x_{n-1})) \to s$, where $q \in Q$, the (finite) set of states, $\sigma \in \Sigma_n$, and $s \in \mathcal{T}_\Sigma(Q \times \{x_0,\ldots,x_{n-1}\})$.

In the intermediate stages of a derivation, states will occur attached to the top of subtrees of the original tree; in fact, the derivation will start with an initial state attached to the top of the input tree. Thus the intermediate configurations will be elements of

$$\mathcal{T}_\Sigma(Q \times \mathcal{T}_\Sigma^o)$$

where a pair $(q,t) \in Q \times \mathcal{T}_\Sigma^o$ is to be interpreted as a tree with a state attached to the top.

We are now ready for the definition of direct generation. As a consequence of our style of definition, we find that this can be done inductively.

<u>Definition</u>. The set of pairs (t,t') where t and t' are in $\mathcal{T}_\Sigma(Q \times \mathcal{T}_\Sigma^o)$ such that $t \Rightarrow t'$ (t directly generates t') is defined by induction on t:

(a) if $t \in \Sigma_o$ then $t' = t$;

(b) if $t \in Q \times \mathcal{T}_\Sigma^o$ then $t = (q,\bar{t})$ where $\bar{t} \in \mathcal{T}_\Sigma^o$. Depending on the form of $\bar{t}$ we have two cases:

(i) if $\bar{t} \in \Sigma_o$ then $(q,\bar{t}) \Rightarrow t'$ iff there is a production $(q,\overline{t}) \to t'$ in the given set of productions; (ii) if $\overline{t} = \sigma(\overline{t}_0,\ldots,\overline{t}_{n-1})$, then there must be a production $(q,\sigma(x_0,\ldots,x_{n-1}) \to s$ in the production set, and t' is obtained from s by substituting $\overline{t}_0$ for $x_0,\ldots,\overline{t}_{n-1}$ for x_{n-1} in the pairs (q',x) occurring as indices in s.

(c) if $t = \sigma(t_0,\ldots,t_{n-1})$ and
$t_0 \Rightarrow t_0',\ldots,t_{n-1} \Rightarrow t_{n-1}'$, then
$t \Rightarrow \sigma(t_0',\ldots t_{n-1}') = t'$.

Notice that there may be more than one tree t' such that $t \Rightarrow t'$; the system is nondeterministic. But there are only finitely many such, and the states "move down" the tree as in a nondeterministic gsm mapping.

Context-free productions. To extend the definition of finite-state to context-free rewriting systems, we return to the gsm mapping for motivation. Notice that a gsm mapping, thought of in the grammatical sense, is simply a left-to-right derivation using a finite-state grammar. If we use productions which introduce more nonterminal (input) symbols into the intermediate stages, we get left-to-right context-free derivations. This suggests defining context-free productions by insisting not that the states be placed at the bottom (variable) nodes, but that they be allowed in the interior of a tree, one state to a branch at most.

Thus if $(q,\sigma(x_0,\ldots,x_{n-1}) \to s$ is a CF production, then we require

$$s \in \mathcal{T}_\Sigma(Q \times \mathcal{T}_\Sigma(\{x_0,\ldots,x_{n-1}\})).$$

The definition of direct generation requires essentially no changes.

For the record:

Definition. A (top-down) context-free tree grammar is a tuple

$$G = (\Sigma, r, X, Q, S, \mathcal{P})$$

where Σ is finite, $r: \Sigma \to \omega$, Q is finite, S is a finite subset of $Q \times \mathcal{T}_\Sigma^o$ (the starting configurations), and $\mathcal{P}$ is a finite set of context-free productions.

Given the relation $\Rightarrow$ defined above, let $\overset{*}{\Rightarrow}$ be its reflexive transitive closure.

Definition. Let G be a tree grammar. The tree language of G, written $\mathcal{L}(G)$, is the set

$$\{t \in \mathcal{T}_\Sigma^o \mid (\exists(q,s) \in S)((q,s) \overset{*}{\Rightarrow} t)\}$$

Evaluations and Recognizable Sets

Before studying tree languages, we want to give a definition which will allow us to obtain indexed languages from tree languages and which will also allow us to define recognizable sets. These definitions can be found in Thatcher and Wright[6].

Definition. By a (uniform) Σ-algebra we mean a set A together with a function $f_\sigma: A^n \to A$ for each $\sigma \in \Sigma_n$. A finite Σ-algebra is one for which A is finite.

Definition. The $\mathcal{A}$-evaluation function of a Σ-algebra $\mathcal{A}$ is the mapping

$$\| \ \|_{\mathcal{A}}: \mathcal{T}_\Sigma^o \to A$$

defined inductively by the rules

(i) $\|\lambda\|_{\mathcal{A}} = f_\lambda$ if $\lambda \in \Sigma_0$ (recall $f_\lambda \in A$)

(ii) $\|\sigma(t_0,\ldots,t_{n-1})\|_{\mathcal{A}} = f_\sigma(\|t_0\|_{\mathcal{A}},\ldots,\|t_{n-1}\|_{\mathcal{A}})$.

The concatenation algebra C is the algebra $(\Sigma_0^*,\{f_\sigma\})$ where

$$f_\lambda = \lambda \ , \quad \lambda \in \Sigma_0$$

$$f_\sigma(w_1,\ldots,w_{n-1}) = w_1\cdot\ldots\cdot w_{n-1}$$

Definition. Let $\mathcal{S} \subseteq \mathcal{T}_\Sigma^o$. The string language defined by $\mathcal{S}$ is the set $\|\mathcal{S}\|_C$.

Definition. The (string) language defined by a tree grammar G is the set

$$L(G) = \|\mathcal{L}(G)\|_C.$$

Theorem (M. Fischer[3]). For every tree grammar G we may effectively find an indexed grammar G' and an OI macro grammar G'' so that $L(G) = L(G') = L(G'')$.

As we have defined them, tree grammars never deal with the empty string. Thus the converse theorem is not true, but given a OI macro grammar G with no productions involving the empty string, we may find an equivalent tree grammar. Alternatively, we may introduce the empty string into tree grammars. This will not be done here.

We remark that IO macro grammars also have a natural analogue in tree grammars; this would involve instead of a "top-down" definition, a "bottom-up" one. OI grammars seem to be more interesting, so we have restricted our attention to these.

Let us now consider evaluation of trees by finite algebras, as is done by Thatcher and Wright[6].

Definition. A finite Σ-automaton is a system $\langle A,\{f_\sigma\},A_F\rangle$ where $\langle A,\{f_\sigma\}\rangle$ is a finite Σ-algebra, and $A_F \subseteq A$. (A_F is the set of designated final states.)

Definition (Thatcher and Wright[6]). $\mathcal{R} \subseteq \mathcal{T}_\Sigma^o$ is recognizable iff there is a Σ-automaton $\mathcal{A}$ so that

$$\mathcal{R} = \{t \in \mathcal{T}_\Sigma^o \mid \|t\|_{\mathcal{A}} \in A_F\}.$$

Theorem (Thatcher[7]). Let $\mathcal{R}$ be recognizable. Then $\|\mathcal{R}\|_C$ is context-free. Conversely, given a context-free grammar G we may find a set Σ and recognizable set $\mathcal{R} \subseteq \mathcal{T}_\Sigma^o$ such that $\|\mathcal{R}\|_C = L(G)-\{\varepsilon\}$.

We see from this theorem and the indexed-grammar theorem above that

$$\frac{\text{CF sets of trees}}{\text{indexed languages}} = \frac{\text{recognizable sets of trees}}{\text{context-free languages}}$$

Notice that using algebras to evaluate trees provides us with a notion of semantics discussed by Petrone[5] and Knuth[4].

Translations

Syntax-directed translations involve operations on trees associated with a context-free grammar. It seems natural, therefore, to consider the effect of a finite state mapping on a recognizable set of input trees.

Definition. A (finite-state) translation system with input set $\mathcal{R}$ is a tree grammar G which uses only finite state productions together with a subset $\mathcal{R}$ of $\mathcal{T}_\Sigma^o$. (Only the initial states of G need be specified; $\mathcal{R}$ is to be recognizable.)

Definition. The tree translation defined by the translation system $(G,\mathcal{R})$ is the set

$$T = \{(s,t) \in \mathcal{R} \times \mathcal{T}_\Sigma^o \mid (\exists q \in Q_o)((q,s) \overset{*}{\Rightarrow} t)\}$$

Definition. The string translation defined by G and $\mathcal{R}$ is the set

$$\{(\|s\|_C, \|t\|_C) \mid (s,t) \in T\}$$

Theorem. If $\mathcal{R}$ is recognizable, then the domain of T is recognizable (effectively).

Proof: It suffices to show that the domain of the translation defined by G and $\mathcal{T}_\Sigma^o$ is recognizable because the domain of T is the intersection of $\mathcal{R}$ with the new domain, and recognizable sets are effectively closed under intersection. To this end we construct an appropriate Σ-automaton.

Let $G = (\Sigma, r, X, Q, Q_o, \mathcal{P})$.

Define $\mathcal{A} = (A, \{f_\sigma\}, A_F)$, where

(i) $A = 2^Q$

(ii) $A_F = \{Q' \subseteq Q \mid Q_o \cap Q' \neq \emptyset\}$

(iii) $f_\lambda = \{q \in Q \mid \text{for some } p \in \mathcal{P},\ p \text{ is } (q,\lambda) \to t\}$

$f_\sigma(Q_o,\ldots,Q_{n-1}) = \{q \in Q \mid$ for some $p \in \mathcal{P}$, p is $(q,\sigma) \to s$ where for each (q',x_i) indexing s, we have $q' \in Q_i\}$

We may check by induction on t that for each $q \in Q$:

$$q \in \|t\|_{\mathcal{A}} \Leftrightarrow (\exists t' \ \mathcal{T}_\Sigma^o)((q,t) \overset{*}{\Rightarrow} t'). \qquad \text{Q.E.D.}$$

Definition. A (finite-state) surface set is the range of a finite-state tree translation. A (finite-state) target language is a set $\|\mathcal{S}\|_C$ where $\mathcal{S}$ is a surface set.

Corollary. The emptiness problem for surface sets and target languages is solvable.

Corollary. The domain of a string translation is context-free.

The theorem just given remains true even when the transforming tree grammar is context-free. Proving the effectiveness involves using the solvability of the emptiness problem for OI macro grammars, so we have omitted the proof here. We do want to state a converse to the above theorem, however.

Theorem. Let $\mathcal{R}$ be recognizable. Then the identity relation restricted to $\mathcal{R}$ is a finite-state tree translation: $I_{\mathcal{R}} = T$ where T is a finite state translation defined by a grammar G with input set $\mathcal{T}_\Sigma^o$.

The composition of two relations R and S is written $R \circ S$ and is defined to be

$$\{(x,y) \mid (\exists z)(x,z) \in S \text{ and } (z,y) \in R\}.$$

Theorem. Finite-state tree translations are closed under composition (effectively). More precisely, given two translations R and S both with input set $\mathcal{T}_\Sigma^o$, we may effectively find a grammar G" so that the translation defined by G" with input set $\mathcal{T}_\Sigma^o$ is just $R \circ S$.

Proof: All we have to do is imitate the proof for nondeterministic gsm mappings. Let G define S and G' define R. Give G the state set Q and G', Q'. Our new grammar will have state set $Q \times Q'$, and we will combine the productions of G and G' to get those for G". We make use of the fact that both mappings can be carried out at the same time. Formally, let n be the maximum rank of any symbol in Σ. Define a new terminal alphabet

$$\Sigma_o' = \Sigma_o \cup (Q \times \{x_o,\ldots,x_{n-1}\}) \cup ((Q \times Q') \times \{x_o,\ldots,x_{n-1}\}).$$

Let Σ' be the expanded alphabet obtained from Σ by adding the new letters in Σ_o'. We first generate the productions of G'' by using a grammar $\bar{G}$: $\bar{G}$ has the productions of G' together with the productions $(q',(q,x)) \to ((q,q'),x)$ for all $q \in Q$, $q' \in Q$, $x \in \{x_o,\ldots,x_{n-1}\}$. Let $(q,\sigma) \to s$ be a production of G. For each $q' \in Q'$ consider the set

$$A(q',s) = \{t \in \mathcal{T}^o_{\Sigma'} \mid (q',s) \underset{\bar{G}}{\overset{*}{\Rightarrow}} t\}$$

Then G'' will have the productions

$$\{((q,q'),\sigma) \to t \mid t \in A(q',s), q' \in Q'\}.$$

Do this for each production in G. Finally, take the set of initial states of G to be $Q_o \times Q'_o$. We may then prove by induction that

$$((q,q'),t) \underset{G''}{\overset{*}{\Rightarrow}} t'$$

if and only if for some $s \in \mathcal{T}^o_{\Sigma}$,

$$(q,t) \underset{G}{\overset{*}{\Rightarrow}} s \quad \text{and} \quad (q',s) \underset{G'}{\overset{*}{\Rightarrow}} t'.$$

Corollary. The above theorem remains true even when the input sets are arbitrary recognizable sets.

Proof: The identity $I_{\mathfrak{R}}$ is the translation defined by a grammar $G_{\mathfrak{R}}$ with input set $\mathcal{T}^o_{\Sigma}$.

Closure Properties

Theorem. A surface set need not be recognizable.

Proof: $\mathcal{T}^o_{\Sigma}$ is recognizable, hence a surface set. Let $\tau \notin \Sigma$, $r(\tau) = 2$. Then

$$\mathcal{S} = \{\tau(s,s) \mid s \in \mathcal{T}^o_{\Sigma}\}$$

is a surface set (trivial) but

$$\|\mathcal{S}\|_C = \{ww \mid w \in \Sigma^*_o\}$$

is not context free. However, we have

Theorem. Surface sets are closed under intersection with recognizable sets (effectively).

Proof: Corollary to the theorem on composition.

Corollary. Surface sets are recursive.

Proof: $t \in \mathcal{S} \Leftrightarrow \mathcal{S} \cap \{t\} = \emptyset$.

Although surface sets are closed under intersection with recognizable sets, they are not closed under full intersection. Our method of proving this result will lead us to a solution of the infiniteness problem both for surface sets and for target languages.

Definition. Let Σ be an alphabet with valence function r. Define a new function $r' : \Sigma \to \omega$ by setting

$$r'(\sigma) = \begin{cases} 1 & \text{if } r(\sigma) > 0; \\ 0 & \text{otherwise.} \end{cases}$$

Denote the new alphabet (Σ, r') by Σ'.
Let $t \in \mathcal{T}^o_{\Sigma}$. Then $\Pi(t)$, the set of paths through t, is the subset of $\mathcal{T}^o_{\Sigma'}$ defined by

(i) $\Pi(\lambda) = \{\lambda\}$ for $\lambda \in \Sigma_o$
(ii) $\Pi(\sigma(t_o,\ldots t_{n-1})) = \{\sigma(t') \mid t' \in \bigcup_i \Pi(t_i)\}$

Let $\mathcal{S} \subseteq \mathcal{T}^o_{\Sigma}$. The set of paths through $\mathcal{S}$ is

$$\Pi[\mathcal{S}] = \bigcup_{t \in \mathcal{S}} \Pi(t).$$

Lemma 1. If $\mathcal{S}$ is a surface set, then so is $\Pi[\mathcal{S}]$.

Proof: Π is a finite state translation.

Lemma 2. If $\mathcal{S}$ is a surface set of strings, then $\mathcal{S}$ is regular (in Kleene's sense).

Proof: If G is a grammar defining $\mathcal{S}$, then no productions of G will be used which have other than strings on their right-hand sides. It is easy to make up a nondeterministic gsm whose range will be $\mathcal{S}$, and so $\mathcal{S}$ is regular.

Corollary. Surface sets are not closed under intersection.

Proof: Let $\Sigma = \Sigma_o \cup \Sigma_1 \cup \Sigma_2$, where $\Sigma_2 = \{\rho\}$, $\Sigma_1 = \{\sigma,\tau\}$, $\Sigma_o = \{\lambda\}$. Define $\sigma^1(x) = \sigma(x)$, $\sigma^{i+1}(x) = \sigma(\sigma^i(x))$. Put

$$\mathcal{S}_1 = \{\rho(\sigma^j(\lambda), \tau^i(\sigma^j(\lambda))) \mid i,j \geq 1\}$$

$$\mathcal{S}_2 = \{\rho(\sigma^j(\lambda), \tau^j(\sigma^i(\lambda))) \mid i,j \geq 1\}$$

$\mathcal{S}_1$ and $\mathcal{S}_2$ are both surface sets. But

$$\Pi[\mathcal{S}_1 \cap \mathcal{S}_2] = \{\rho\sigma^j(\lambda)\} \cup \{\rho\sigma^j\tau^j(\lambda)\}$$

which is not regular; hence $\mathcal{S}_1 \cap \mathcal{S}_2$ is not a surface set by Lemma 2.

Theorem. The infiniteness problem is solvable for surface sets and for target languages.

Proof: The first statement follows immediately from Lemma 2 and the fact that infiniteness is solvable for regular sets. The second will follow as well if we can show that any target language L can be obtained from a surface set which is finite if and only if L is finite. To show this, let G generate the surface set from which L is obtained. Delete all symbols σ such that $r(\sigma) = 1$ from the right-hand side of all productions in G. Thus, in our new grammar, the trees on the right-hand sides of productions will be in the 2-SPG form of Bar-Hillel[2]. No tree will be generated unless it is a single symbol or has nothing but branching nodes in it. Since the new target language is still L, we have the desired result.

We conclude by stating some properties of target languages and a conjecture.

Theorem. Target languages form an AFL.

Theorem. Target languages are recursive.

Conjecture. With respect to inclusion, (finite-state) target languages are incomparable with indexed languages.

Acknowledgment

This material is a revision of the results obtained in my doctoral dissertation[8], where I proposed finite-state translations as models for transformational grammars. I am indebted to Prof. Dana Scott, my thesis adviser, and to Prof. Joyce Friedman and Mr. William Ogden for their valuable suggestions and guidance.

This research was supported in part by USAF contract F 44 620-68-C-0030.

References

1. Aho, A. V., Indexed grammars--an extension of context-free grammars. IEEE Conf. Record, Symp. on Switching and Automata Theory (Oct., 1967).
2. Bar-Hillel, Y., M. Perles and E. Shamir, On formal properties of simple phrase structure grammars. Z. Phonetik Sprachwiss. Kommunikat 14 (1961), 143-172.
3. Fischer, M., Grammars with macro-like productions. IEEE Conf. Record, Symp. on Switching and Automata Theory (Oct., 1968).
4. Knuth, D., Semantics of context-free languages. Math. Systems Theory, vol. 2, no. 2 (June, 1968), pp. 127-158.
5. Petrone, L., Syntax-directed mappings of context-free languages. IEEE Conf. Record, Symp. on Switching and Automata Theory (Oct., 1968).
6. Thatcher, J. W. and J. B. Wright, Generalized finite automata theory with an application to a decision problem of second-order logic. Math. Systems Theory, vol. 2, no. 1 (March, 1968), pp. 57-81.
7. Thatcher, J. W., Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. J. Comp. Sys. Sci., vol. 1, no. 4 (Dec., 1967), pp. 317-322.
8. Rounds, W. C., Trees, transducers, and transformations. Ph.D. thesis, Stanford Univ., 1968.