15

Mathematical and Computational Models of Transformational Grammar

Joyce Friedman
Department of Computer and Communication Sciences
University of Michigan

INTRODUCTION

In this paper we compare three models of transformational grammar: the mathematical model of Ginsburg and Partee (1969) as applied by Salomaa (1971), the mathematical model of Peters and Ritchie (1971 and forthcoming), and the computer model of Friedman *et al.* (1971). All of these are, of course, based on the work of Chomsky as presented in *Aspects of the Theory of Syntax* (1965).

We were led to this comparison by the observation that the computer model is weaker in three important ways: search depth is not unbounded, structures matching variables cannot be compared, and structures matching variables cannot be moved. All of these are important to the explanatory adequacy of transformational grammar. Both mathematical models allow the first, they each allow some form of the second, one of them allows the third. We were interested in the mathematical consequences of our restrictions.

The comparison will be carried out by reformulating in the computer system the most interesting proofs to date of the ability of transformational grammars to generate any recursively enumerable set. These are Salomaa's proof that the Ginsburg-Partee model can generate any recursively enumerable (r.e.) set from a regular base, and the Peters-Ritchie proof that any r.e. set can be obtained from a minimal linear base. Although modifications are required, it is, as we shall show, possible to obtain these results within the weaker computer model.

Thus, every recursively enumerable language is generated by a transformational grammar with limited search depth, without equality comparisons of variables, and without moving structures corresponding to variables. The comparison reinforces the observation that transformational grammars can be excessively powerful in terms of generative capacity while at the same time lacking features necessary for explanatory adequacy.

An understanding of the role of variables in the structural description of a transformation is essential to the arguments of this paper. For the reader unfamiliar with the notion, we offer here an elementary explanation. Basically the idea is that in giving a structural description to match a tree, part of the description can be omitted in favor of a *variable* which will match any structure. In the simplest case consider the tree

$$S\langle NP\langle \text{rabbits}\rangle\ VP\langle V\langle \text{eat}\rangle\ NP\langle \text{lettuce}\rangle\rangle\rangle$$

where the brackets indicate a tree structure in which S dominates NP and VP, and VP dominates V and NP. Then either of the structural descriptions $NP\ VP$ or $NP\ V\ NP$ matches the tree. But these would not match

$$S\langle AUX\langle \text{do}\rangle\ NP\ \langle \text{rabbits}\rangle\ VP\langle V\rangle \text{eat}\rangle\ NP\ \langle \text{lettuce}\rangle\rangle\rangle$$

because an AUX precedes the leftmost NP. To match either of these trees, and any other tree ending in $V\ NP$, we introduce a variable as the left part of the structural description. The variable will match any initial structure. In the computer notation the percent sign is used for the variable, so we write $\%\ V\ NP$.

MATHEMATICAL AND COMPUTATIONAL MODELS

The two mathematical models limit themselves to the base component and the transformational component of the syntactic part of a grammar. That is, there is no consideration of either semantics or phonology, and furthermore, no mention of the lexical component in syntax. The computer model likewise ignores semantics, although it has now been extended to phonology. Within its syntactic part it has an important lexical component.

For purposes of comparison we restrict ourselves to the components that all three models have in common, that is, the base component and the transformational component. The base component consists of phrase structure rules, with a distinguished initial symbol S. This generates a base tree with S as root. The transformational component maps this tree onto a surface tree by application of a sequence of transformations.

The computer model differs from the mathematical models because of its intended use. It is not designed primarily as a mathematical object, but is the basis for a computer program used by linguists in writing and testing grammars and by students learning about transformational grammar. Some users are interested in the theory of syntax, and write grammars to illustrate points of theory. For example, the program is now being used to investigate case grammars. Other users are interested in describing little-known languages. For them, the grammar represents a hypothesis about the language which is tested by the program. For them transformational theory is simply the best currently available formalism for grammar.

In any case, the user writes a transformational component which is an input to the program. He may also provide a base component and study random sentences, or he may provide a partial or complete base tree. The output of the program is the output of the grammar, that is, a full derivation

from base tree to surface tree. The user can modify or accept his grammar based on this information.

Given this intended use of the program, there are certain natural consequences for the model. The computer form of a grammar should seem natural to a linguist: a computerized transformational grammar shouldn't look too different from those in the literature. The model must allow a choice of alternative expressions, and thus will contain redundant features, and also some features which purists might consider too powerful.

A computer model can be too strong without harm, provided only that the user can specialize it by rejecting options which he feels are unnecessary. A simple example is the choice of repetition modes for transformations. A transformation applies if, on testing, a structural analysis of the tree is found which matches the structural description of the transformation. Four different 'repetition modes' can be used to determine whether one or all such structural analyses will be found, and if and when the corresponding structural changes will be made. Using mnemonics constructed of 'A' for 'analyze' and 'C' for 'change', we can represent these as AC (find the first analysis and apply the change), AACC (find all analyses, then do all changes), ACAC (find the first analysis, do the change, repeat), and AAC (find all analyses, do one randomly selected change). Some linguists might argue that not all of these are necessary; indeed many might feel that one mode suffices for all transformations. For a computer model, however, it is advisable to allow all reasonable possibilities, so that a user may make his own choice. The user is free to experiment, without being committed to the use of excessive power.

Similarly, there may be technical weaknesses in a computer model which are desirable for practical reasons. Although the base grammar specifies that the sentence symbol *S* may be introduced recursively, the computer program will not introduce embedded sentences in the base unless the user has specifically called for one in the input to a particular run. This device is necessary if the output of the generation process is to remain within practical limits. In fact, for transformational grammar, the relation between embedding and embedded sentences must be well specified if a sentence is to result.

In a mathematical mode, on the other hand, it becomes very important to be neither too weak nor too strong, because the investigation of power is a prime purpose in constructing the model. Unbounded processes must be expressed as such, wherever the linguistic theory allows them. That is, results would immediately be suspect if what is really an unbounded process in language were simulated by a bounded process in the mathematical model. Thus, both the Ginsburg-Partee and Peters-Ritchie models attempt to be faithful to linguistic theory in a way that the computer model does not.

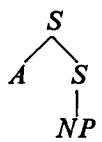
Restrictions of the use of variables

We have mentioned above several important ways in which the computer model is apparently too weak for explanatory adequacy. These are

1. bounded depth of analysis
2. lack of equality comparisons on variables
3. inability to move structures corresponding to variables.

On the other hand, both mathematical models allow unbounded depth of analysis; both allow equality comparisons of variables, although the Ginsburg-Partee model compares terminals only; Peters-Ritchie, but not Ginsburg-Partee, allows movement of structures corresponding to variables.

The bounded depth constraint in the computer model requires that in analyzing a tree to match a structural description the search never goes below an S unless that S is explicitly mentioned. (If S is the only recursive symbol in the base, then an equivalent statement is that no essential use of variables is allowed. Any occurrence of a variable could then be replaced by a finite choice of definite symbols. Thus, all these restrictions are restrictions on variables.) Consider a structural description $S/\langle A\ S\ \langle NP\rangle\rangle$ and the subtree



If the search begins at the top S, the daughters A and S are found and then the NP below this S can be matched. But no match will be found if the structural description is just $S/\langle A\ NP\rangle$, which does not allow the search to go below the intermediate S. This restriction was made because it is in practice useful. It is convenient not to have to consider more than the current S unless that consideration is part of the argument. For example, in the Peters-Ritchie proof there are several structural descriptions (e.g. T3) with the condition *1 is not an S*; in the computer version the condition can be omitted because of the bounded depth constraint (see Appendix B). Further, the user can study the effect of unbounded search by writing the alternation of the case of depth 1, depth 2, and so on up to any finite limit. This of course does not give an exact representation of the transformation, but is adequate for all practical purposes.

A consequence of the decision to block search whenever an S is encountered in the tree but not in the structural description is that it is not possible for a transformation to pull a constituent out of an arbitrarily deeply embedded subtree, raise it over any number of sentence boundaries, and bring it up to a higher sentence. Postal (1971) has argued convincingly that transformations must have this power. It must be possible to write a single transformation that will find a noun phrase and bring it up from an arbitrarily deeply embedded subsentence. Postal's example of the type of sentence whose derivation requires unbounded depth is

Who did Jack find out that Mary believed that Bill said that you thought you saw?

Here the deep structure corresponds roughly to

Q(Jack found out that (Mary believed that (Bill said that (you thought (you saw who)))))

where the parentheses indicate sentence embedding. The final 'who' must be moved up to the top sentence from a sentence arbitrarily far down. There is in general no upper bound to this depth.

There are results that can be proved about transformational grammars if search depth is bounded, that are not easily proved, and are possibly not even true, otherwise. Hamburger (1971) was able to extend to transformational grammars some of Gold's results (1967) on identification of languages in the limit. One of Hamburger's crucial assumptions was that the search depth was bounded.

The remaining two points in which the mathematical models appear more powerful, the equality comparison for variables and moving of structure matched by variables, are primarily motivated by the analysis of conjunction (which indeed poses many problems for transformational grammar).

THEOREMS OF SALOMAA AND PETERS-RITCHIE

In spite of these linguistically weak aspects of the computer model, it retains the full power demonstrated for the mathematical models. We examine two major mathematical results on transformational grammar, and then show that both proofs can be reproduced in the computer system.

Theorem (Salomaa). For an alphabet A, there is a set R of transformational rules such that any recursively enumerable λ-free language L_0 over A is generated by a (restricted) Ginsburg-Partee transformational grammar with regular base and with R as the set of transformational rules. Salomaa's proof is based on the theorem that every recursively enumerable language L_0 can be expressed as

$$L_0 = h(h_1(D \cap K_1) \cap h_1(D \cap K_2))$$

where h and h_1 are homomorphisms, D is a Dyck language, and K_1 and K_2 are regular languages. D, h, and h_1 are determined by A only, and are independent of L_0. The base component generates the language $K_1 a_0 K_2$, where a_0 is a marker, and the transformations carry out the homomorphisms and check whether or not substrings belong to D and whether they belong to the required intersections.

The Salomaa proof seems intuitively to differ from a normal linguistic derivation. In particular, the lack of a transformational cycle seem unnatural. This is closely related to the arbitrary recursion in the base. Not only is S not a recursive symbol, but most other nonterminals of the base can be recursive. Thus, the Salomaa proof seems to use a grammar which is different in obvious ways from the grammar required for natural languages.

Theorem (Peters and Ritchie). L is a recursively enumerable language on the

alphabet $A=\{a_1, a_2, \ldots, a_n\}$ if and only if L is generated by a transformational grammar with the base rules $S \rightarrow S\#$ and $S \rightarrow a_1 a_2 \ldots a_n b\#$.

The Peters-Ritchie proof begins with the fact that every r.e. language is enumerated by some Turing machine Z. They construct a transformational grammar which simulates the Turing machine Z. The terminal string of the tree as it goes through the derivation contains a substring that represents the instantaneous description of the Turing machine. The transformational grammar is set up so that at each cycle exactly one Turing machine instruction is applied. As the derivation proceeds up the tree, the Turing machine is simulated step-by-step, and its instantaneous description is carried along. Each time a cycle is completed, a boundary symbol is erased, and one of the S's given by the first rule of the base component is pruned. The initial (base) tree has enough sub-trees so that there will be one for each instruction to be used. Finally, a very clever scheme is used to erase all the boundary symbols just in case the Turing machine has been adequately simulated. The language of the grammar, that is, those surface strings which do not contain the boundary symbol, corresponds to the set of sentences in the language enumerated by the Turing machine.

These results show that transformational grammars as usually formulated are too powerful. Peters and Ritchie (1969) observe that their result makes impossible an empirically testable statement of the universal base hypothesis for natural languages, unless one enlarges the range of data to be accounted for by a grammar.

There are, of course, other results that show that transformational grammars under certain restrictions generate restricted subclasses of languages. For example, Petrick (1965) showed that, for a particular definition of transformational grammar, only recursive languages are obtained if the depth of embedding is bounded by a function of the length of the sentence. Refined results along these lines are given in Peters and Ritchie (1971).

COMPARISON OF THE THREE MODELS

In comparing the three models we emphasize features used in these two proofs. For each proof, we constructed a computer version that was run on several examples. Appendix A lists our reproduction of Salomaa's proof; Appendix B is the Peters-Ritchie proof. In the discussion we show how these versions of the proofs differ from the originals, and in particular we show that neither of them makes unavoidable use of the more powerful concepts of variable which are lacking in the computer model. Specific transformations of the proofs will be referred to frequently; they will be found in the appendixes.

Base component

No attempt was made to simulate on the computer the base components of the two proofs. The base rules are listed in the Appendixes for completeness

only. All computer runs were made starting with completed base trees. The computer model treats base rules as ordered context-free rules with recursion on the sentence symbol only.

In the absence of any specification of the base component in the Ginsburg-Partee model, Salomaa obtains the base trees by a regular grammar in which the rules are unordered and there are many recursive symbols; however, the sentence symbol is not reintroduced by any rule.

The Peters-Ritchie model allows an unordered set of context-sensitive rules as the base. In the proof under consideration the base is a two-rule minimal linear grammar; recursion is on the sentence symbol only.

Structural description

In reproducing the Salomaa proof in the computer system, a few changes to the transformations were necessary. There is no difficulty with S-bounded search depth, since the base structure is a simple sentence with no embedded S. However, the transformation T8, which checks whether $h_1(k_1)=h_1(k_2)$, does so by an equality test on variables. Equality of variables here is equality of corresponding terminal strings only. T8 cannot be transcribed directly because comparison of variables is not allowed in the computer model. However, by using a device that Salomaa uses elsewhere in the proof, T8 is replaced by a sequence of transformations T8A, T8B, and T8C (see Appendix A), which create an extra copy of the relevant subtree and then compare its halves node by node, deleting if the comparison is satisfactory.

Another difference, though not relevant to our main discussion, is that for Salomaa a structural description is a boolean combination of proper analyses with no explicit mention of substructure. His proof uses boolean combination in three transformations, T5, T7, and T11. In all three the form is a single proper analysis which carries the main burden of the transformation, conjoined with a negation which specifies that no letter of a particular alphabet occurs. This is done to ensure that the previous transformation has applied as often as it can and is now no longer applicable. Thus, if the transformations had been taken as ordered, the negation would be unnecessary. In the computer model a structural description is a sequence of structures and negation is available only with reference to the subanalysis of a mentioned node. This was adequate to the purposes of T7 and T11 (see Appendix A). T5 was rewritten more simply since the boolean combination was in fact unnecessary.

The Peters-Ritchie model has a much richer notion of structural description, specified in terms of a boolean combination of conditions on a factorization of the labelled bracketing representing the tree. The model allows equality comparisons of variables; these compare structures rather than terminal strings. However, this device is not used in their proof.

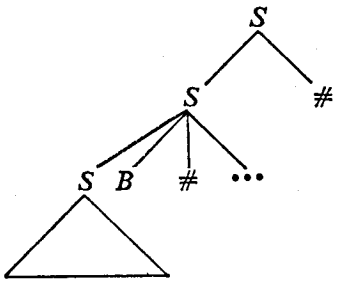
The Peters-Ritchie model does allow unbounded search depth, and it was on this point that some changes were necessary in transcribing the trans-

formations. It was necessary to introduce additional structure in the statements of the transformations; however, some supplementary conditions were thereby eliminated.

The basic scheme for the transformations of the Peters-Ritchie proof can most easily be seen by examining transformation T5. For the case of an alphabet with the three symbols $A1$, $A2$, $A3$, the structural description of T5 can be written in the notation of the computer model as

S⟨S⟨# # # # A1 A2 A3 B # # # # (# #) B⟩B# %⟩#.

Because of the bounded search depth discussed above, the computer version of this transformation mentions two sentence symbols S rather than just the one mentioned in the original proof. The tree matched by this structural description can be represented schematically as:



The sentence tree whose contents are fully given by the structural description is the lowest one in the figure. The dots indicate the position in the tree of the instantaneous description of the Turing machine. The top S in the figure is the one at the top of the current step of the cycle.

All of the structural descriptions of the Peters-Ritchie proof can be rewritten in this form or, as in T3, as choices of a finite set of these. Thus unbounded search depth, although allowed by the notation, is not needed for the proof.

Structural change

The computer model disallows any operations on variables, but is otherwise able to reproduce all changes allowed by the other two models. The Peters-Ritchie model allows deletion, substitution, and adjunction, all of sequences of factors; the Ginsburg-Partee model is similar except that sequences corresponding to variables cannot be moved. Since the Peters-Ritchie proof does not use the ability to move variables, there is no difficulty in transcribing its structural changes to the computer notation.

Sequencing of transformations

The three models differ in the way they specify the order of application of transformations. The computer model provides a language in which a

control program can be written. The Peters-Ritchie model assumes the standard bottom-to-top transformational cycle. In Appendix B, the computer control program is used to provide the transformational cycle for the Peters-Ritchie proof.

The control device of the Ginsburg-Partee model is also quite general, but Salomaa's proof uses only a restricted model in which the control device is ignored. Salomaa's transformations are unordered and are formulated so that at any point in the derivation at most one of them can apply. The computer system requires some order for the rules; in Appendix A the control program simply applies them in the order written. By marking some rules as ACAC the full derivation is carried out in one pass through the rules. An alternative would be to specify all rules as AC and have the control program invoke them repeatedly until none applies.

Parameters

The repetition mode parameter of the computer model was discussed above as an example of deliberate excess power. Although the mathematical model of Ginsburg and Partee provides directly for AC transformations, and indirectly, through the control device, for ACAC transformations, Salomaa's proof uses only AC. In simulating his proof it is convenient, though not necessary, to use both AC and ACAC. The Peters-Ritchie model treats all transformations as AAC. In their proof, transformations T3 and T4 must be regarded as AAC; the others could be indifferently taken as any of the four modes.

The computer model allows transformations to be specified as optional or obligatory. Since the optionality parameter is relegated by Ginsburg and Partee to the control device, which Salomaa does not use, Salomaa's transformations are all obligatory. In the Peters-Ritchie system all transformations are obligatory, although the effect of optionality can be obtained.

Special conditions

The computer model was designed to be neutral with respect to certain special conditions on transformational grammars so that a user might simulate them if desired but would not be required to include them. The condition on recoverability of deletions is required by the Peters-Ritchie model; it is preserved by our version of their proof. Their automatic 'pruning convention' is simulated by the transformation TPRUNE in the computer version. The filtering condition is a condition on the output of a grammar: for Peters and Ritchie the presence of the boundary marker # signals a failed derivation; Salomaa uses the markers $a_0, \ldots, a_6$ for this purpose.

Acknowledgement

This research was supported in part by the National Science Foundation under Grant GS 31309 to The University of Michigan.

REFERENCES

Chomsky, N. (1965) *Aspects of the Theory of Syntax*. Cambridge, Mass.: MIT Press.

Friedman, J. *et al.* (1971) *A Computer Model of Transformational Grammar*. New York: American Elsevier.

Ginsburg, S. & Partee, B. (1969) A mathematical model of transformational grammars. *Information and Control*, **15**, 297–334.

Gold, E.M. (1967) Language identification in the limit. *Information and Control*, **10**, 447–74.

Hamburger, H.J. (1971) *On the Learning of Three Classes of Transformational Grammars*. Ph.D. dissertation, University of Michigan.

Peters, P.S. & Ritchie, R.W. (1969) A note on the universal base hypothesis. *Journal of Linguistics*, **5**, 150–2.

Peters, P.S. & Ritchie, R.W. (1971) On restricting the base component of a transformational grammar. *Information and Control*, **18**, 483–501.

Peters, P.S. & Ritchie, R.W. (forthcoming) On the generative power of transformational grammars.

Petrick, S.R. (1965) *A Recognition Procedure for Transformational Grammars*. Ph.D. dissertation, Massachusetts Institute of Technology.

Postal, P.M. (1971) *Cross-Over Phenomena*. New York: Holt, Rinehart and Winston.

APPENDIX A

```
"COMPUTER EXPERIMENTS IN TRANSFORMATIONAL GRAMMAR:"
"SALOMAA- THE GENERATIVE CAPACITY OF TRANSFORMATIONAL GRAMMARS"
"         OF GINSBURG AND PARTEE"
"INFORMATION AND CONTROL, 18, 227-232 (1971)"


          "THE BASE GENERATES A WORD  K1 A0 K2 "
PHRASESTRUCTURE
     S = Q1.
          "LET G1 BE A REGULAR GRAMMAR GENERATING K1"
          "WITH INITIAL SYMBOL Q1"
          "WITH NONTERMINALS  U1, U2, SAY"
     "**** INSERT HERE ALL RULES  A = X B, B NONTERMINAL, OF G1"
     "**** INSERT HERE A RULE  A = X S2, FOR EACH RULE A = X, X IN I2"
          "LET G2 BE A REGULAR GRAMMAR GENERATING K2"
          "WITH INITIAL SYMBOL Q2"
          "WITH NONTERMINALS  V1, V2, SAY"
     S2 = A0 Q2.
     "**** INSERT HERE ALL RULES OF G2"
     $ENDPSG

     "LET I BE D1, D2"
     "LET I1 BE C1,C2,C3,C4"
     "LET THE SYMBOLS OF I2 BE B1,...,B6, WITH INVERSES BI1,...,BI6

TRANSFORMATIONS

          "T1 DUPLICATES THE BASE WORD"
          "AND INTRODUCES THE MARKER A1 BETWEEN THE TWO COPIES"
 TRANS T1 AC.
   SD I Q1.
   SC 1 ADRIS 1, A1 ARISE 1.

          "THE NEXT RULES OPERATE ON THE LEFTMOST COPY"
          "T2 CHECKS WHETHER OR NOT  K1  BELONGS TO  D "
          "AND IF IT DOES ERASES K1. "
 TRANS T2 ACAC.
   SD % (1 B1 2 BI1, 1 B2 2 BI2, 1 B3 2 BI3, 1 B4 2 BI4, 1 B5 2 BI5,
         1 B6 2 BI6) % S2 A1 Q1.
   SC ERASE 1, ERASE 2.

          "T3 CHANGES THE MARKER TO A2"
 TRANS T3 AC.
   SD S2 2A1 Q1.
   SC A2 SUBSE 2.

          "T4 CHECKS WHETHER OR NOT  K2  BELONGS TO D,"
          "AND, IF IT DOES, ERASES  K2"
 TRANS T4 ACAC.
   SD A0 % (1 B1 2 BI1, 1 B2 2 BI2, 1 B3 2 BI3, 1 B4 2 BI4, 1 B5 2 BI5,
            1 B6 2 BI6) % A2 Q1.
   SC ERASE 1, ERASE 2.

          "T5 CHANGES THE MARKER TO A3, "
          "PROVIDED BOTH  K1  AND  K2  BELONG TO  D "
          "T5 ALSO ERASES THE LEFT MOST BRANCH OF THE TREE"
 TRANS T5 AC.
 "BOOLEAN COMBINATION EXCLUDING LETTERS OF I2 IS UNNECESSARY"
   SD 1A0 2A2 Q1.
   SC ERASE 1, A3 SUBSE 2.

          "THE REMAINING RULES OPERATE ON THE RIGHT-MOST BRANCH"
          "T6 APPLIES THE HOMOMORPHISM  H1  TO BOTH  K1  AND  K2 "
```

```
"TRANS T6"
      "THE ACTUAL SC WILL DEPEND ON H1. THE STRING H1(X) IS ADJOINED"
      "TO THE LEFT OF 3, THEN 3 IS ERASED."
      "TO REPRESENT THE HOMOMORPHISM H1 WE BREAK T6 INTO 2R+8 "
      "TRANSFORMATIONS, AND CARRY OUT THE MAPPING SYMBOL BY SYMBOL"
TRANS T6B1 ACAC.
  SD A3 % 3B1 %.
  SC C1 ALESE 3, C1 ALESE 3, ERASE 3.  "H1(B1)=C1 C1"
TRANS T6B2 ACAC. SD A3 % 3B2 %. SC ERASE 3. "H1(B2)=EMPTY"
TRANS T6B3 ACAC. SD A3 % 3B3 %. SC C2 ALESE 3, ERASE 3. "H1(B3)=C2"
TRANS T6B4 ACAC. SD A3 % 3B4 %. SC C3 ALESE 3, C4 ALESE 3, ERASE 3.
TRANS T6B5 ACAC. SD A3 % 3B5 %. SC C1 ALESE 3, ERASE 3.
TRANS T6B6 ACAC. SD A3 % 3B6 %. SC C2 ALESE 3, C2 ALESE 3, ERASE 3.
TRANS T6BI1 ACAC. SD A3 % 3BI1 %. SC C3 ALESE 3, ERASE 3.
TRANS T6BI2 ACAC. SD A3 % 3BI2 %. SC ERASE 3.
TRANS T6BI3 ACAC. SD A3 % 3BI3 %. SC C4 ALESE 3, C1 ALESE 3, ERASE 3.
TRANS T6BI4 ACAC. SD A3 % 3BI4 %. SC C2 ALESE 3, ERASE 3.
TRANS T6BI5 ACAC. SD A3 % 3BI5 %. SC C3 ALESE 3, C3 ALESE 3, ERASE 3.
TRANS T6BI6 ACAC. SD A3 % 3BI6 %. SC ERASE 3.

      "T7 CHANGES THE MARKER TO  A4, "
      "AFTER HAVING CHECKED THAT ALL APPLICATIONS OF H1 ARE MADE"
TRANS T7 AC (A3).
  SD 1 A3 Q1¬/<% (B1,B2,B3,B4,B5,B6,BI1,BI2,BI3,BI4,BI5,BI6) %>.
  SC A4 SUBSE 1.

      "T8 CHANGES THE MARKER TO A5, PROVIDED H1(K1)=H1(K2)"
"TRANS T8 CANNOT BE TRANSCRIBED DIRECTLY"
"BECAUSE WE DO NOT HAVE THE EQUALITY TEST FOR VARIABLES X"
"INSTEAD WE COPY THE TREES OF INTEREST AND COMPARE BY DELETION"
TRANS T8A AC (A4).
  SD A4  1 Q1 .
  SC 1 ADRIS 1, A5 ARISE 1.
TRANS T8B ACAC (A4).
  SD A4 Q1 A5 % 5(C1,C2,C3,C4) A0 % 6(C1,C2,C3,C4), WHERE 5 EQ 6.
     "COULD AVOID CONDITION 5 EQ 6 BY DOING 4 CASES"
  SC ERASE 5, ERASE 6.
TRANS T8C AC (A4).
  SD 1 A4 2 Q1  3 A5 4 A0.
  SC A5 SUBSE 1, ERASE 3, ERASE 4.

      "T9 ERASES THE BRANCH DOMINATED BY THE NODE S2,"
      "AND CHANGES THE MARKER TO A6"
TRANS T9 AC (A5).
  SD 1 A5 % 3 S2.
  SC A6 SUBSE 1, ERASE 3.

      "T10 APPLIES THE HOMOMORPHISM H"
TRANS T10 ACAC.
      "AN ALTERNATIVE WAY TO DO A HOMOMORPHISM"
  SD A6 % (1 C1,2 C2, 3 C3, 4 C4) %.
  SC D1 ALESE 1, D2 ALESE 1, ERASE 1,
     D1 ALESE 2, ERASE 2,
     ERASE 3,
     D2 ALESE 4, D2 ALESE 4, ERASE 4.

      "T11 ERASES THE MARKER, THUS LEAVING A WORD OF  L0 "
TRANS T11 AC.
  SD 1 A6 Q1¬/<% (C1,C2,C3,C4) %>.
  SC ERASE 1.

    CP I;TREE;II.
  $ENDTRA

  $MAIN FTRIN TRAN.

   "INPUT BASE TREES FOLLOW"
S<Q1<B1 U1<B3 U2<BI3 U1<BI1 S2<A0 Q2<B1 V1<B3 V2<BI3 V2<BI1>>>>>>>>>>.
S<Q1<B1 U1<BI1 U2<B3 S2<A0 Q2<B1 V1<BI1 V2<BI3>>>>>>>>.
  $
```

APPENDIX B

```
"COMPUTER EXPERIMENTS IN TRANSFORMATIONAL GRAMMAR:"
"PETERS AND RITCHIE- ON RESTRICTING THE BASE COMPONENT"
"INFORMATION AND CONTROL, 18, 483-501 (1971) "

PHRASESTRUCTURE
   S = ( S #, A1 A2 A3 B # ) .
     $ENDPSG

          "NOTE: BOUNDED DEPTH OF SEARCH, NO ESSENTIAL VARIABLES "
          "NOTE: NO NUMBERED VARIABLES "
          "NOTE: NO RESTRICTIONS ARE USED "

TRANSFORMATIONS
 TRANS T12."COMBINES T1 AND T2"
          "PRODUCES R BOUNDARIES FOR INNERMOST SENTENCE,"
          "AND POSITIONS THEM CORRECTLY AND ADDS B AS RIGHTMOST SYMBOL"
          "LET N=NUMBER OF TM SYMBOLS"
          "LET R=NUMBER OF TM STATES=4"
   SD 1A1 A2 A3 2B 3# .
   SC 3 ADLES 1, 3 ADLES 1, 3 ADLES 1, 3 ADLES 1, "(R TIMES)"
      2 ADRIS 3.
 TRANS T3 AAC .
          "T3 AND T4 PRODUCE THE OTHER 3 #'S IN THE INNERMOST SENTENCE"
          "AND THE STRING B**U # Y B**V "
          "T3 PRODUCES B**V, THEN T4 PRODUCES #Y, THEN T3 PRODUCES B**U"
  SD (9S< 1# # # # A1 A2 A3 3B (4#,5#) (# #) B > % 8#,
     S<9S<1# # # # A1 A2 A3 3B (4#,5#) (# #) B > % >   8#).
          "CONDITION THAT 1 IS NOT AN S IS UNNEEDED IN THIS FORMAT"
   SC 4 ADRIS 4, 3 ADRIS 9, ERASE 8.
 TRANS T4 AAC .
   SD   S<
      10S<#### (3A1 A2 A3,A1 3A2 A3,A1 A2 3A3) B (5#,#) 6# B > %> 9# .
   SC 5ADLES 6, 3 ADRIS 10, 5 ADRIS 10, ERASE 9.
 TRANS T5  II .
          "INCREASES T (INITIALLY 4) BY 1 WHEN LEFTMOST USABLE SQUARE"
          "OF TAPE IS REACHED FOR THE FIRST TIME"
   SD S< S< # # # # A1 A2 A3 B # # # 2# (# #) B> B # % > # .
   SC 2 ADRIS 2.
 TRANS T6 .
          "INCREASES T BY 2 WHEN RIGHTMOST USABLE SQUARE IS REACHED"
          "FOR THE FIRST TIME"
          "NOTE: A LEMMA IS NEEDED TO THE EFFECT THAT EVERY R.E. "
          "LANGUAGE IS ENUMERATED BY SOME TM THAT SCANS ITS WHOLE"
          "INPUT- THIS IS OF COURSE EASY TO PROVE"
   SD S< S< # # # # A1 A2 A3 B # # 2# # (#) B> %.# (A1,A2,A3,B) B > # .
   SC 2 ADRIS 2, 2 ADRIS 2.
 TRANS T7B.
          "TURING MACHINE ZB: REPLACES ALL A3 BY A2 AND GOES HOME "
          "(S1 A1 R S1) (S1 A2 R S1) (S1 A3 A2 S1) (S1 B L S2) "
    "     (S2 A1 L S2) (S2 A2 L S2) (S2 B R S3) "
   SD S<
     S<#### A1  3A2 A3 B ####(##)(#)B> %  (A1,A2,A3,B) 10# (18(A1,A2),
                                                          14A3       )
     %.B > 17 # .
   SC 18 ALESE 10, 9 ARISE 10, 3 SUBST 14, 2 ADLES 10, 19 ADLES 10,
      4 ADLES 10, ERASE 11, ERASE 12, ERASE 13, ERASE 17 .
 TRANS T7B2.
   SD S<
    S<# 2### A1 A2 A3 B ####(##)(#)B> % 9(A1,A2,A3,B) 10# (11# (A1,A2),
                                                          B        )
     % B > 17 # .
   SC 18 ALESE 10, 9 ARISE 10, 3 SUBST 14, 2 ADLES 10, 19 ADLES 10,
      4 ADLES 10, ERASE 11, ERASE 12, ERASE 13, ERASE 17 .
```

```
TRANS T7B3.
  SD S<
    S<# 2# 19## A1 A2 A3 B ####(##)(#)B>%(A1,A2,A3,B) 10# 11# 188
    % B > 17 # .
  SC 18 ALESE 10, 9 ARISE 10, 3 SUBST 14, 2 ADLES 10, 19 ADLES 10,
     4 ADLES 10, ERASE 11, ERASE 12, ERASE 13, ERASE 17 .
TRANS T8 .
         "T8,T9,T10 ARE CLEAN-UP TRANSFORMATIONS"
         "T8 CHECKS THAT THE SQUARES AT EACH END OF THE TAPE ARE NOT"
         "BEING SCANNED, AND THAT T=7 SO NO EXCESS TAPE WAS PROVIDED"
         "IF OK, T8 ERASES RIGHT-END B AND HALTING STATE, REPLACES"
         "DEEPEST SENTENCE BY S<#> AND POSITIONS A # TO SIGNAL T9"
  SD S<1S< # # # # A1 A2 A3 B # # # # # # #  # B> % (A1,A2,A3,A4,B)
     6# (10# (11# (12#))) (A1,A2,A3,B) % 9B > 13# .
         "   SECOND CHANGE OPERATION HERE COULD BE DONE BY ERASURES"
  SC 6 ADRIS 1, (S|+LOW +DONE|<#>) SUBSE 1,
     ERASE 6, ERASE 10, ERASE 11, ERASE 12,
     ERASE 9, ERASE 13.
TRANS T9 .
         "T9 PASSES A # ACROSS THE TAPE FROM L TO R, ONE SQUARE AT A"
         "TIME, ERASING EACH B ENCOUNTERED"
  SD S< S<#> % 3# (4B,5(A1,A2,A3)) 6* % > 7#  .
  SC ERASE 4, 3 ALESE 6, ERASE 7.
TRANS T10.
         "T10 ERASES FINAL # AND RIGHTMOST SYMBOL IF IT IS B "
  SD S<1S<#> % 3# (4B,A1,A2,A3) > 6# .
  SC ERASE 1, ERASE 3, ERASE 4, ERASE 6.
TRANS TPRUNE .
         "REDUCTION CONVENTION FOR LABELED BRACKETINGS"
  SD 1S<2S> , WHERE 1 DOMBY S .
  SC 2 SUBSE 1.
"TRANSFORMATIONAL CYCLE"
TRANS SMARK  V.
  SD 1S, WHERE 1 NDOM S.
  SC |+LOW| MERGEF 1.
TRANS LOWESTS  V.
  SD (1S|+LOW|, 1S<S|+LOW +DONE| %>, 1S<S|+DONE|<S|+LOW +DONE| %> %>),
     WHERE 1 NINC1 |+DONE|.
  SC |+DONE| MERGEF 1.
CP
  SMARK;
  IN LOWESTS(1) DO< I; II ; TREE > .
    $ENDTRA
```