COLING 82, J. Horecký (ed.)
North-Holland Publishing Company

Composition of Translation Schemes with D-Trees

Martin Plátek

Charles University, Faculty of Mathematics and Physics
Mal[illegible]ranské nám. 25
118 00 Prague
Czechoslovakia

Generative systems (GS) are defined in this paper as a composition of simple translation schemes with dependency trees. The following issues are discussed: (a) explicative power of GS, (b) the time complexity for the analysis and synthesis for GS.

INTRODUCTION

A generative system for Czech was presented in Sgall [6] . The concept of a generative system was studied by Plátek [4] and Plátek and Sgall [5] . In this paper we use a similar approach as that presented by Hajičová, Plátek and Sgall in [3] .

We define generative systems as a fundamental device for construction of grammars of natural languages. We give here some mathematical results to illustrate the usefulness of the new concept. We try first to formulate the necessary requirements on a grammar G of a natural language L. The grammar G must determine:

a) The set of all correct sentences of the language L. The set will be denoted by LC.

b. The set of the correct structural descriptions (SD) of the language L. SD represents all meanings of all sentences of LC.

c) The relation SH between LC and SD. The relation SH describes the ambiguity and the synonymy of L.

By a structural description we understand a dependency tree (D-tree).

The concept of a simple translation scheme from [1] is a generalisation of context-tree grammar. We introduce here a similar concept of a translation scheme, in this case as a generalisation of dependency grammar (see [2] , [5]).

A generative system (GS) is defined as a sequence of translation schemes with a special asymmetric property.

We show that the explicative power of GS increases with the length of GS. We present results concerning on algorithm for the analysis and synthesis of GS and show that its time complexity is independent on the length of GS.

Moreover for a given GS we can construct a similar GS, for which a fast algorithm for synthesis exists.

<u>Definitions.</u>

<u>Notation.</u> The vocabulary, sets of nodes, edges and rules are here <u>nonempty</u> and finite sets.

Let R be a relation. We denote

Dom (R) $= \{a \; ; \; [a,b] \in R\}$ and

Range (R) $= \{b \; ; \; [a,b] \in R\}$

By $f : U \rightarrow V$ we denote a total mapping from U into V.

Def. A string over a vocabulary V is a triple S=(U,LR,o), where U is a set of nodes, LR a linear ordering of U, $o:U \rightarrow V$. Let o(u)=A. We say that A is the value of node u. Let S=(U,LR,o), S1=(U1,LR1,o1), S2=(U2,LR2,o2) be the strings and $u \in U$. We say that S2 is produced from S by replacing u by S1, when the string S1 is placed between the predecesor and the succesor of node u and otherwise S2 does not differ from S. We denote as V^+ the set of all nonempty strings over V.

Def. Let S1 = (U1,LR1,o1), S2 = (U2,LR2,o2) be strings. Let U1 = $\{ul_1,\dots,ul_n\}$ and U2 = $\{u2_1,\dots,u2_n\}$ and $ul_1,\dots,ul_n$ be in the ordering LR1, and $u2_1,\dots,u2_n$ in the ordering LR2 and $o1(ul_i)= o2(u2_i)$ for all i between 1 and n. Then we say that S1 and S2 are equivalent.

We shall not distinguish between equivalent strings.

Def. A quintuple SR=(U,LR,B,r,o) is called a D-tree over V, when S(SR)=(U,LR,o) is a string and $o:U \rightarrow V$, B(SR)=(U,B.r) is a tree with the root r and when the following condition holds: The nodes of every path in B(SR), which begins with a leaf, are nodes of a substring of S(SR). We say that S(SR) is a projection of SR.

Def. Let SR1=(U1,LR1,B1,r1,o1) and SR2=(U2,LR2,B2,r2,o2) be D-trees. Let strings S(SR1) and S(SR2) be equivalent. Let f be a one-to-one mapping from U1 on U2, which preserves the ordering LR1 to the ordering LR2. Let f(r1)=r2 and let it hold that $[u,v] \in B1$ iff $[f(u), f(v)] \in B2$. Then we say that SR1 and SR2 are equivalent. We shall not distinquish between equivalent D-trees.

Def. Let D=(U,LR,B,r,o), D1=(U1,LR1,B1,r1,o1) and D2=(U2,LR2,B2,r2,o2) be D-trees and $u \in U$. We say, that D2 is produced from D by replacing u by D1, when S(D2) is produced from S(D) by replacing u by S(D1) and the neighbours of r1 in B(D2) are the same as neighbours of u in B(D). Otherwise D2 does not differ from D.

Def. A translation scheme of type string - D-trees (TS [S,D]) is a quadruple T=(VN,VT,S,P), where VN is a the vocabulary of nonterminals, VT the vocabulary of terminals, $VN \cap VT=\emptyset$, $S \in VN$ and P is a set of rules of the following type: $LS \leftarrow A \rightarrow RS$, where $A \in VN$ (the middle of the rule) LS (the lefthand side) is a string over $VN \cup VT$, RS(the righthand side) is a D-tree over $VN \cup VT$ and the following condition holds: When all nodes with terminals are erased from S(RS) and LS, then we get two equal strings.
Let $p=LS \leftarrow A \rightarrow RS$. We write $[LS1,RS1] \; p\!\Longrightarrow [LS2,RS2]$, when (i):the leftmost nonterminal node of LS1 is some u with the value A, (ii):the leftmost nonterminal node of RS1 is some v with the value A and (iii):LS2 is produced from LS1 by replacing u by LS and RS2 is produced from RS1 by replacing v by RS.

$\bigcup_{p \in P} p\!\Longrightarrow$ is denoted as $\Longrightarrow$ and $\overset{*}{\Longrightarrow}$ is the transitive closure of $\Longrightarrow$.

We denote as $TR(T)= \{[LS,RS] \; ; \; [S,S] \overset{*}{\Longrightarrow} [LS,RS] \; , \; LS,S\,(RS) \in VT^+\}$.

Remark. Analogically as a translation scheme of the type string - D-tree was defined, also definitions of the type string - string (TS [S,S]) or of the type D-tree - D-tree (TS [D,D]) can be given. By TS [S,S] the lefthand side and righthand side of a rule is always a string. By TS [D,D] both sides of a rule are always D-trees. As TS we denote the set of all translation schemes of all the three types.

Def. Let T1,...,Tn be a sequence of TS. We denote as TR(T1,...,Tn)=TR(T1).TR(T2)...TR(Tn). The main definition of this paper is the following:

Def. A generative system (GS) is a sequence T1,...,Tn of TS, where TR(T1,...,Tn) is a relation between strings and D-trees and for every [d1,d2] ∈ TR(Tn) there exists a s1,so,such that s1,d2] ∈ TR (T1,...Tn). The set AN (T1,...,Tn;v)= {[v,d] ∈ TR(T1.. ....,Tn) } is called the analysis of v. The set ST(T1,...,Tn;d)= = {[s,d] ∈ TR(T1,...,Tn)} is called the full synthesis of D-tree d.

Remark. Let GS1=$T_1,\ldots,T_n$ be a GS. Then

Range(TR(T_1)) ⊃ Dom (TR(T_2))⊃...Range(TR(T_{n-1})) ⊃ Dom(TR(T_n)).

We call this property of GS1 an asymetric property of GS.

Def. Let GS1 be a GS. We say that the function MS is a function of the minimal synthesis of GS1, if the following conditions are fulfiled:

a) $MS^{-1} \subset$ TR(GS1)

b) Dom(MS)=Range(TR(GS1)).

Def. D-grammar (DG) is a T ∈ TS [S,D], where T=(VN,VT,S,P) and for every p ∈ P, p=LS⟵ A ⟶RS there holds, that LS=S(RS).

Def. We denote DR_o= { TR(T);T ∈ DG} and DR_j= { TR(T1,...,Tj); T1,...,Tj ∈ GS} for j ∈ N. For j ∈ N ∪ {0} we write $1DR_j$= {F ∈ DR_j; F is a function } .

Note. We need also one more concept. It is the concept of an h-morphic generative system for another one.

Def. Let V1,V2 be two alphabets and h:V1⟶V2. Let S1=(U1,LR1,o1), S2=(U2,LR2,o2) be two strings, where o1:U1⟶V1, o2:U2⟶V2. We say that a tuple (f,h) is an h-morphism from S1 to S2, when f:U1⟶U2 is a one-to-one mapping which preserves the ordering on nodes and for every u ∈ U1 there holds that h(o1(u))=o2(f(u)). We say that S1 is h-morphic for S2, if there exists an h-morphism from S1 to S2.

Def. Let D1=(U1,LR1,B1,r1,o1) and D2=(U2,LR2,B2,r2,o2) be D-trees. Let (t,h) be an h-morphism S(D1) to S(D2). Let there hold that [u,v] ∈ B1 iff [t(u), t(v)] ∈ B2 and t(r1)=r2. We say that (t,h) is a h-morphism from D1 to D2. We say that D1 is h-morphic to D2, when there exists an h-morphism from D1 for D2.

Def. Let T1=(VN1,VT1,S1,P1) and T2=(VN2,VT2,S2,P2) be TS. Let h:VN1 ∪ VT1⟶VN2 ∪ VT2, where h(VN1)=VN2, h(VT1)=VT2. Let there exist a one-to-one mapping MP from P1 on P2 such, that if p=LS1⟵ A1⟶ RS1 and MP(p)=LS2⟵ A2⟶ RS2, then LS1 is h-morphic to LS2, RS1 is h-morphic to RS_2 and h(A1)=A2. We then say, that T1 is h-morphic for T2.

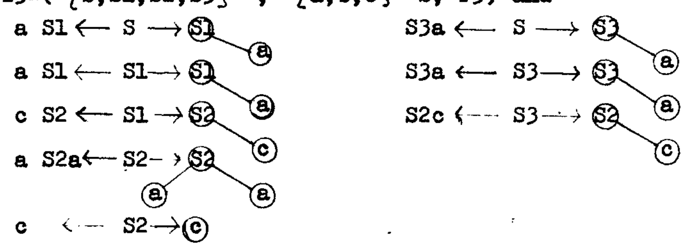
Def. Let GS1=$T1_1,\dots,T1_n$ and GS2=$T2_1,\dots,T2_n$ both be GS. Let $T1_1$ be h_1-morphic to $T2_1$, $T1_2$ h_2-morphic to $T2_2$,... and so on to n; we say then, that GS1 is h-morphic for GS2, where $h=(h_1,\dots,h_n)$.

Examples

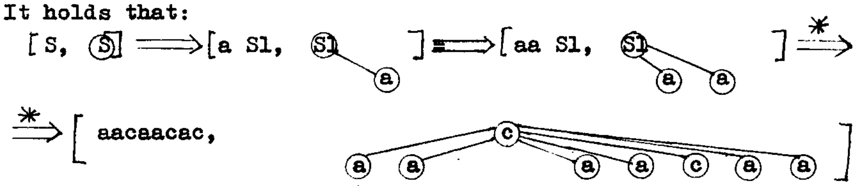
Example 1.

Let us have an example of a translation scheme.
Let T3=({S,S1,S2,S3} , {a,b,c} S, P3) and

P3:



It holds that:



We can sea that Dom(TR(T3))=
$=\{a^n\ c\ a^n\ c\ a^j\ ;\ n,j \in N\} \cup \{a^j\ c\ a^n\ c\ a^n\ ;\ j,n \in N\}$
and that TR(T3) is a function.

Example 2.

We present in this example some interesting set of translation schemes.

G4 = ({S,A} , {a,c} , S, P4) where

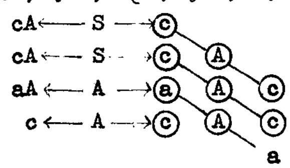
P4: caAac ⟵ S ⟶ caAac
aAa ⟵ A ⟶ aAa
c ⟵ A ⟶ c

Then TR(G4)= $\{[c\ a^n\ c\ a^n\ c,\ c\ a^n\ c\ a^n\ c]\ ;\ n \in N\}$

T5=({ S,A} , {a,c} , S, P5)

P5: cA ⟵ S ⟶ cAc
cA ⟵ A ⟶ cAc
aA ⟵ A ⟶ aAa
c ⟵ A ⟶ c

and
T6= ({S,A} , {a,c} , S, P6)
P6:



and let

$R4(k)=TR(G4, \underbrace{T5,\ldots,T5}_{k\text{-times}})$,

then $R4(k)= \{ [c\ a^n\ c\ a^n\ c,\ (c\ a^n)^{2^{k+1}} c] ; n \in N \}$

and if $TR(G4,\underbrace{T5,\ldots,T5}_{(k-1)\text{times}},T6)= I\,T(k)$

then $R4(k)= \{ [a,b] ; [a,c] \in I\,T(k) \text{ and } b=S(c) \}$

Results.

Assertion 1. For $j \geq 0$ it holds that $DR_j \subsetneqq DR_{j+1}$ and $1DR_j \subsetneqq 1DR_{j+1}$.

Notation. $|s|$ denotes the length of the string s, which is the card (U), where U is the set of nodes of s.

Assertion 2. Let GS1 be a generative system,

a) Then there exist an algorithm that computes for every string v the set AN(GS1,v) (analysis) with the time complexity bound by a function $K1. |v|^3 . \max \{card\ (AN\ (T_1,\ldots,T_j;v)\}_{j=1}^{n}$, where K1 depends only on GS1.
b) Then there exists an algorithm that computes for every D-tree d the set ST(GS1,d) (full synthesis) with the time complexity bound by function $K2. |S(d)|^3$. card (ST(GS1)), where K2 depends only on GS1.

Assertion 3. Let GS1 be a GS. Then there exists an h-morphic generative system GS2 for GS1 and an algorithm that for every D-tree d computes ST(GS2,d) with a time complexity bound by function $K . |S(d)|$. card (ST(GS1,d)) where K depends only on GS2 and Dom(TR(GS1))=Dom(Tr(GS2))

Assertion 4. Let GS1 be a GS. Then there exists an h-morphic generative system GS2 for GS1 and an algorithm such that for every D-tree d computes MS(d) with a time complexity bound by function $K . |S(d)|$, where MS is the function of minimal syntesis of GS2, Dom(TR(GS1))=Dom(TR(GS2)) and K depends only on GS2.

Remarks.

Remark to Assertion 1.

We sketch here a proof of Ass.1. We see that $DR_0 \subset DR_1$ and $1DR_0 \subset 1DR_1$. Dikovskij and Modina have shown in [2] , that TR (T3) from Example 1 cannot be in DR_0. We see that T3 is a TS. Thus $DR_0 \subsetneqq DR_1$. Since TR(T3) is a function , we see that $1DR_0 \subsetneqq 1DR_1$.

In the Example 2 we have shown that $IT(k) \in 1DR_k$. From the results on composition of pushdown transducers (PST) in [4] and from the equivalence theorem between TS's and PST's from [1] it follows, that $IT(k+1) \notin DR_k$. Thus $DR_j \subsetneq DR_{j+1}$ and $1DR_j \subsetneq 1DR_{j+1}$.

Remark to Assertion 2.
The algorithm for analysis and synthesis for a GS is based on the idea of Cocke-Younger-Kasami algorithm. For a seguence of simple translation schemes of the type string-string the algorithm is presented in Suchomel [7] . The difference between the upper boundary of the time complexity of the full synthesis and analysis is given by the asymmetric property of a GS.

Remark to Assertion 3.
The basic idea of the proof is a construction of a new GS to GS1. The new GS, denoted GS2, has full information in the alphabets for a straightforward algorithm for a full synthesis.

Remark to Assertion 4.
The idea of the proof is analogous to that of Assertion 3. When we have a partition of Dom(TR(GS1)) in the clases of synonymous sen - tences, the function of minimal synthesis chooses always only one representant of his class. Therefore the algorithm can be so fast.

Conclusion remarks.
When formulating a grammar for natural language, we can use with advantage the modularity of GS. We have shown that the time comple-xity of the analysis and synthesis for DR_j, $j \geq 2$ is independent on j. Otherwise the explicative power of DR_j is increasing with j. We have also shown, that to any generative system there can be con - structed an h-morphic generative system with the full information for a fast algorithm of the minimal synthesis.

References.

[1] Aho A.V.-Ullman J.D.: The Theory of Parsing Translation and Compiling, Vol.1:Parsing (Prentice-Hall, Englewood Cliffs,1972)

[2] Dikovskij A.J.-Modina L.S.: O trech typach odnoznačnosti kon - těkstnosvobodnych jazykov, in Matematičeskaja lingvistika i teoria algoritmov. Kalinin 1978.

[3] Hajičová E.- Plátek M.-Sgall P.: Komunikace s počítačem v češtině [Man-machine communication in Czech] in SOFSEM (VVS Bratislava 1980)

[4] Plátek M.: On serial and parallel compositions of, PST'S Thesis Faculty of mathematics and physics, Charles University, Prague (December 1979)

[5] Plátek M.-Sgall P.: A scale of context sensitive languages: Aplication to natural language, Information and Control,vol38 N.1.(1978)

[6] Sgall P.: Generativní popis jazyka a česká deklinace (Academia Prague 1967)

[7] Suchomel K.: Generativní systémy a rozpoznávání jazyků [Generative systems and language recognition] Diploma work, Faculty of mathematics and physics, Charles University, Prague (April 1981)