Reprinted from JOURNAL OF COMPUTER AND SYSTEM SCIENCES
All Rights Reserved by Academic Press, New York and London

Vol. 33, No. 3, December 1986
Printed in Belgium

Membership for Growing Context-Sensitive Grammars Is Polynomial*

ELIAS DAHLHAUS

Technische Universität Berlin, Fachbereich Mathematik, Strasse des 17. Juni 135, 1000 Berlin, West 12, Germany

AND

MANFRED K. WARMUTH

Department of Computer and Information Sciences, Applied Sciences Building, University of California, Santa Cruz, California 95064

Received July 1, 1985; revised March 10, 1986

A context-sensitive grammar is a *growing* context-sensitive grammar, if the right-hand side of every production is strictly longer than the left-hand side. We show that for any fixed growing context sensitive grammar, the membership problem for the corresponding language is polynomial.

1. INTRODUCTION

Context-sensitive grammars (csgs) are one of the classical grammar families of formal language theory. They were introduced in [Ch59] and have been studied extensively since then (see [Bo73, Ha78] for an overview). Context-sensitive grammars are defined as rewriting systems, where the length of the right-hand side of every production is at least as large as the length of the left-hand side. This restriction on the productions is responsible for the fact that the question of membership for context-sensitive languages (csls) is equivalent to the question of acceptance for nondeterministic linear bounded automata [Ku64]. Therefore membership for csls is PSPACE complete [Ka72] and this is true even for certain fixed grammars. In this paper we show that if we restrict ourselves to "growing" productions, i.e., the right-hand side of every production is strictly longer than the left-hand side, then membership for fixed csls is polynomial.

This may appear surprising in view of the results obtained in [Bo78]. The growing csls are a subclass of $\text{LINEAR}_{\text{CS}}$ as defined in [Gl64, Bo71]. Languages of

* This research was done while the authors were visiting the Hebrew University of Jerusalem. The first author was supported by the Minerva Foundation and the second author by the United States–Israel Binational Foundation Grant 2439-82. A preliminary version of this appeared in the CAAP 86 Proceedings [DW86].

0022-0000/86 $3.00

$\text{LINEAR}_{\text{CS}}$ are given by an arbitrary csg which has the property that every word w in the language has a derivation of length at most $c\,|w|$,[1] for some overall constant c, which only depends on the grammar. In [Bo78] it was shown that there are *NP*-complete languages in $\text{LINEAR}_{\text{CS}}$. Thus our result that the family of growing csls is in *P* deserves an explanation.

Observe that in $\text{LINEAR}_{\text{CS}}$ "complex derivations" are allowed using nongrowing productions; then the final word may be padded such that the length of the overall derivation is linear in the length of the final word. In fact, for every language in *P* there is a polynomially padded version of this language which is in $\text{LINEAR}_{\text{CS}}$ [Bo78].

An arbitrary csg may be converted into a growing csg by adding a dummy symbol to the right-hand side of every nongrowing production. The grammar needs to be changed slightly so that the dummy symbols are "ignored." But now padding increases the length of the word exponentially. Each time a "signal" runs from one end of a sentential form to the other, the length increases by a constant factor.

Note that the question of emptiness for csls is undecidable [BPS61]. By padding a csg with dummy symbols a related growing csg is constructed. Clearly, emptiness for the corresponding growing csls is also undecidable. For the question of emptiness, the "exponential padding" is redundant.

The paper is outlined as follows. In Section 2 the basic notations are developed. Given a word w, we want to decide membership for a language defined by some fixed growing csg. A planar directed acyclic graph is associated with every derivation of the grammar. In Section 3 we show that if all productions are growing then there is a path of length $O(\log(|w|))$ from each vertex to some sink (not necessarily the same sink) in the graph. The sinks of the graph are labeled with the word w to be tested for membership. These short paths are then used in Section 4 in a polynomial cut-and-paste algorithm for deciding membership for a growing csl.

In the cut-and-paste algorithm each "piece" of a derivation graph is characterized by a tuple (called a frame) which contains all the essential information about the piece. Because of the short paths there is only a polynomial number of different frames which need to be considered. Frames were used extensively in [GS85, GW85, GW86a] for studying polynomial cases of k-parallel rewriting.

In Section 5 the polynomiality of the membership problem for fixed, growing csls is contrasted with the fact that there are *NP*-complete languages defined by fixed, growing scattered grammars. In scattered grammars (scgs) [GH68, GW86b] the symbols to be rewritten in parallel are not required to be adjacent. In every production each symbol on the left-hand side is rewritten into a string of length at least one. In growing scgs each symbol must be rewritten into a string of length strictly bigger than one. It is easy to see that in the derivation trees of growing scgs each node has an $O(\log(|w|))$ path to a leaf, where w is the word to be parsed. But since for scattered grammars the rewritten symbols do not need to be adjacent, we cannot cut and paste the derivation trees along the short paths.

[1] $|w|$ denotes the length of w.

The parallel complexity and the space complexity of the membership problem for fixed growing csls is discussed in the conclusion section. The main open problem is to determine the complexity of membership for "variable" growing csls, i.e., not only the word to be tested but also the growing csg is a variable of the input. The question is whether this problem can be solved in polynomial time or whether it is *NP*-complete.

2. Preliminaries

A *context-sensitive grammar* (csg) G is a quadruple (V, Σ, P, S) where:

(i) V is a finite set of symbols, Σ is the subset of V which contains the terminal symbols, and S is the startsymbol in $V-\Sigma$.

(ii) P is a finite set of productions of the form $\alpha \to \beta$, s.t. $\alpha, \beta \in V^+$ and $|\alpha| \leqslant |\beta|$. For two words u and v in V^*, u *derives* v, denoted $u \Rightarrow v$, if there exist $x, y, \alpha, \beta \in V^*$ s.t. $u = x\alpha y$, $v = x\beta y$ and $\alpha \to \beta \in P$. Let $\overset{*}{\Rightarrow}$ denote the reflexive and transitive closure of $\Rightarrow$. Using this notation we are ready to describe the *context-sensitive language* (csl) defined by the csg G: $L(G) = \{w \mid S \overset{*}{\Rightarrow} w \text{ and } w \in \Sigma^*\}$.

Now the *membership problem* for a csl $L(G)$ is defined as follows:

Input: a word $w \in \Sigma^*$, where Σ is the terminal alphabet of G;
Question: is $w \in L(G)$?

Note that G is fixed, i.e., it is not a variable of the input. There are fixed csgs for which this problem is PSPACE complete [Ku64, Ka72].

We restrict ourselves to a subclass of csgs for which the membership problem is in P (Sect. 4). A csg G is *growing* if for all productions $\alpha \to \beta$ of the grammar, $|\alpha| < |\beta|$.

Following [Lo70] each *derivation* is associated with a planar directed acyclic graph called a (*derivation*) *graph*. The vertices in such a graph will be labeled with the corresponding symbols and productions used in the derivation. Let $\omega(x)$ denote the label of vertex x, where ω is a function from the set of vertices of the derivation graph to $V \cup P$. Vertices labeled with symbols (resp. productions) are called *symbol* (resp. *production*) *vertices*. We inductively define the *derivation graph* $D_k = (V_k, E_k)$ which is associated with the derivation $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_k$:

Case $k = 1$. Let $\alpha_1 = a_1 a_2 \cdots a_p$, where $a_i \in V$. Then $D_1 = (V_1, E_1)$ has the vertices $V_1 = \{x_1, x_2, ..., x_p\}$ s.t. $\omega(x_i) = a_i$ and no edges, i.e., E_1 is empty.

Case $k > 1$. Assume $\alpha_1 \Rightarrow \alpha_2 \cdots \Rightarrow \alpha_{k-1}$ corresponds to the graph $D_{k-1} = (V_{k-1}, E_{k-1})$ and $\alpha_{k-1} = u\alpha v \Rightarrow u\beta v = \alpha_k$. From D_{k-1} and the production $\alpha \to \beta$ the graph D_k is constructed for $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_k$. For the word $\beta = b_1 b_2 \cdots b_q$ create the vertices $V_\beta = \{y_i \mid 1 \leqslant i \leqslant q\}$ and for the production $\alpha \to \beta$ create an additional vertex y. Choose the vertices s.t. V_{k-1}, V_β and $\{y\}$ are distinct. The vertices of V_β

are labeled with the symbols of β, i.e., $\omega(y_i) = b_i$, and y is labeled with the production $\alpha \to \beta$. Let V_α be the sinks (symbol vertices) of D_{k-1} corresponding to α. Now $V_k = V_{k-1} \cup V_\beta \cup \{y\}$ and $E_k = E_{k-1} \cup V_\alpha \times \{y\} \cup \{y\} \times V_\beta$.

An example is given in Fig. 1. The planarity of the derivation graphs follows from the fact that only sinks are connected to the new production vertex. The sources of the graph D_k correspond to α_1 and the sinks to α_k. We say that D_k *derives* α_k. Since the graph is planar there is a natural left to right order amongst the sources: let $\alpha_1 = a_1 a_2 \cdots a_p$, then for $1 \leqslant i < j \leqslant p$ the vertex corresponding to a_i is to the *left* of the vertex of a_j and the vertex of a_j is to the *right* of a_i. Similarly, there is a natural left to right order amongst the sinks of a derivation graph, and amongst the predecessors and successor of every production vertex. Two sources (sinks) are called *adjacent* if they are adjacent in the left to right order of the sources (sinks).

In a derivation graph D a *path* π is defined to be a sequence $x_1, x_2, \ldots, x_{e+1}$ of vertices of D. The paths π *starts* at x_1, *finishes* x_{e+1} and has *length* e. Note that a path which contains one vertex has length zero. In this paper we assume that non-empty paths always end at a sink of D. The *distance* $d_x(y)$ of y from x is the length of the shortest paths which start at x and finish at y. Note that $d_x(x) = 0$.

In the following lemma we will to show that there exists a shortest paths from each vertex to some sink s.t. no pair of paths is "crossing." To construct such a set of paths we use the following definition of consistency. Two paths are *consistent* if they have no common vertices, or if starting from the first common vertex the paths are identical. A set of paths is consistent if each pair is. Note that since derivation graphs are planar two consistent paths cannot "cross."

LEMMA 1. *For any derivation graph there exists a set of shortest paths from all vertices to sinks such that this set of paths is consistent.*

Proof. Let v_i, for $1 \leqslant i \leqslant m$, be the vertices of a derivation graph D and let π_i be a shortest path starting at v_i (and finishing at a sink of D). We now inductively construct paths π_i' (for $1 \leqslant i \leqslant m$), where π_i' starts at v_i, s.t. $\{\pi_j' \mid 1 \leqslant j \leqslant i\}$ is a consistent set of shortest paths.

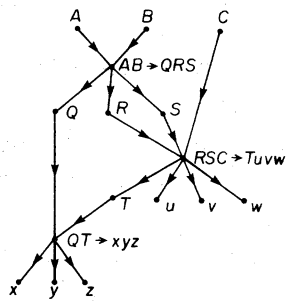


FIG. 1. The derivation graph corresponding to the derivation $\underline{ABC} \Rightarrow \underline{QRSC} \Rightarrow \underline{QTuvw} \Rightarrow xyzuvw$ (the rewritten symbols are underlined).

Assume the set $\{\pi'_j \mid 1 \leqslant j \leqslant \bar{m} < m\} = \Pi'$ is consistent. If $\pi_{\bar{m}+1}$ is consistent with Π' we set $\pi'_{\bar{m}+1} = \pi_{\bar{m}+1}$ and there is nothing to show. Otherwise, let x be the first common vertex of $\pi_{\bar{m}+1}$ with some path π' of Π'. Since both π' and $\pi_{\bar{m}+1}$ are shortest paths, the suffixes of π' and $\pi_{\bar{m}+1}$ which start with x have the same length. Let $\pi'_{\bar{m}+1}$ be the path which agrees with $\pi_{\bar{m}+1}$ up until x and then follows π' to the sink. Clearly, $\pi'_{\bar{m}+1}$ has the same length as $\pi_{\bar{m}+1}$ and is consistent with Π'. This completes the description of the inductive construction. ∎

3. Short Paths in Derivation Graphs

Consider derivation graphs for a growing csg which derive a word w. In this section we show that in such graphs there is a path of length $O(\log(|w|))$ from each vertex to a sink. We prove this by assigning weights to the vertices, s.t. big weights will correspond to short paths.

Let us first discuss why short paths do not always exist for derivation graphs of grammars which define languages in $\text{LINEAR}_{\text{CS}}$. In [Gl64] it was shown that $L = \{ucu^R cu: u \in \{a, b\}^*\}$ is not $\text{LINEAR}_{\text{CS}}$.[2] Since growing csls are a subclass of $\text{LINEAR}_{\text{CS}}$ [Bo73] the language L is not a growing csl. Intuitively, only $O(\log(|w|))$ bits can be transmitted across paths of length $O(\log(|w|))$. But in L, $O(|w|)$ bits need to be transmitted to synchronize the production of the words u, u^R and u in $w = ucu^R cu$.

It is crucial that in the definition of L the word u is over a two symbol alphabet. Just producing three blocks of equal size as in the language $L' = \{a^n b^n c^n: n \geqslant 1\}$ is much easier. One can show that L' is a growing csl. In L' only $O(\log(|w|))$ bits need to be transmitted. It is easy to see that $\hat{L} = \{a^{2n} b^{2n} c^{2n}: n \geqslant 0\}$ is a growing csl. We let a special symbol scan the word. During each complete scan the number of symbols a, b, and c is doubled. From this it is easy to see that L' is also a growing csl. To produce the word $a^n b^n c^n$, $\lfloor \log n \rfloor$ scans are used. Each scan corresponds to a bit in the bit representation of n. Again we double the number of symbols in each scan, but we also add an additional symbol if the corresponding bit is one.

We mentioned already in the introduction that for every language in P there is a padded version [Bo71] which is in $\text{LINEAR}_{\text{CS}}$. Thus even though $\{ucu^R cu: u \in \{a, b\}^*\}$ is not in $\text{LINEAR}_{\text{CS}}$, the language $\{ucu^R cud^{(|u|)^2}: u \in \{a, b\}^*\}$ is.

We proceed to prove the existence of short paths in derivation graphs of a growing csg. Let D be such a derivation graph deriving the word w. Each vertex x of D is associated with a subgraph of D. Let D_x be the subgraph induced by all vertices reachable from x.

The length of the paths will depend on the *growth ratios* of the productions in the grammar. The growth ratio of a production $\alpha \to \beta$ is the ratio $|\beta|/|\alpha|$. The minimum growth ratio of all productions of a grammar is the growth ratio of the grammar.

[2] The word u^R denotes the reverse of the word u.

Throughout the paper this minimum is denoted by g. Note that $g > 1$ for growing csgs.

The growth ratio of a production vertex is the ratio between the number of immediate successors over the number of immediate predecessors. Thus g is a lower bound on the growth ratios of the production vertices of D. Since each production vertex of D_x has at least as many immediate predecessors and the same number of immediate successors in D as in D_x, the growth ratios of the production vertices of D_x are also bounded by g.

We now assign weights $t_x(\cdot)$ to the vertices of D_x according to the following scheme:

(i) $t_x(x) = 1$.

(ii) For a production vertex p, the weight $t_x(p)$ is the sum of all the weights of the immediate predecessors of p.

(iii) If p is a production vertex with k immediate successors, then each of these receives a weight of $t_x(p)/k$.

Note that $\Sigma_{s \text{ is sink of } D_x} t_x(s) = 1$. Since D_x has at most $|w|$ sinks, there is a sink in D_x of weight at least $1/|w|$.

The following lemma shows that big weights correspond to short paths.

LEMMA 2. *Let y be a symbol vertex of D_x and let d be a non-negative integer. If $t_x(y) \geq g^{-d}$ then $d_x(y) \leq 2d$.*

Proof. We prove this by an induction on d. The base case of $d = 0$ is trivial. Assume the lemma holds for all $d' < d$ and let y be a symbol vertex of D_x s.t. $1 > t_x(y) \geq g^{-d}$. Let p be the production vertex which precedes y. Assume p has a immediate predecessors and b immediate successors. Since $t_x(p) = b \cdot t_x(y)$, p must have an immediate predecessor y' with weight at least $(b/a)\, t_x(y)$. By the above remarks $t_x(y') \geq g \cdot t_x(y) \geq g^{1-d}$. Applying the inductive hypothesis it follows that $d_x(y') \leq 2d - 2$ and $d_x(y) \leq 2d$. ∎

Since there is a sink of weight at least $1/|w|$ in D_x, the lemma implies the existence of a path of length at most $2\lceil \log_g(|w|) \rceil$ from x to a sink. For fixed growing csgs this bound is $O(\log(|w|))$ since $g > 1$ and since g only depends on the grammar. Paths are called *bounded* if they are of length at most $2\lceil \log_g(|w|) \rceil$. A derivation graph is *bounded* if there is a consistent set of bounded paths from all vertices to sinks. Combining the above remarks with Lemmas 1 and 2, we get the basis for the polynomial algorithm:

THEOREM 1. *Every derivation graph of a growing* csl *which derives the word w is bounded.*

4. Membership of Growing csl Is Polynomial

In the previous section we showed there are short paths from all vertices to sinks in derivation graphs. We now use these paths to "cut" derivation graphs into "pieces." Each piece is bordered on the left and on the right by a path of length $O(\log(|w|))$. There is an exponential number of derivation graphs and pieces. We therefore gather the essential information about a piece in a frame. Part of this information will be a description of the left and the right path. There will be only a polynomial number of valid frames, which are all found by the algorithm. The information gathered in the frames will be sufficient to decide membership. The same technique was used extensively in [GS85, GW85, GW86a] to show that membership for various problems of k-parallel rewriting is polynomial. Also the classic Cocke–Kasami–Younger algorithm [Ka65, Yo67, Ha78] for context-free language membership can be described using a simple notion of frames. We first rewrite the Cocke–Kasami–Younger algorithm using all the essential notions of this section: *frames*, *valid* frames, the valid frames of a derivation tree, *instance* of a valid frame, *basic* frames, the set VAL of all valid frames. This will help the reader who is familiar with the Cocke–Kasami–Younger algorithm to understand this section.

Assume that we are given a context-free grammar in Chomsky normal form [Ha78]. Let w be the word to be tested for membership. A frame is a tuple (A, l, r) s.t. A is a symbol of the alphabet and $1 \leqslant l \leqslant r \leqslant |w|$. Note that there is only a polynomial number of frames. A frame (A, l, r) is *valid* if $A \overset{*}{\Rightarrow} w_l w_{l+1} \cdots w_r$ in the context-free grammar. We call the derivation tree that corresponds to such a derivation an instance of (A, l, r). Observe that valid frames parameterize all derivation trees that derive subwords of w. The valid frames of an arbitrary derivation tree D are all frames for which there is a subtree of D that derives a subword of w. It is easy to see that $S \overset{*}{\Rightarrow} w$ iff $(S, 1, |w|)$ is valid. (This corresponds to Theorem 2.)

The Cocke–Kasami–Younger algorithm simply computes the set VAL of all valid frames:

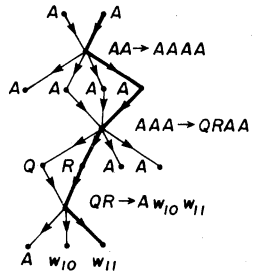
0. Initialize VAL with the set of all *basic* valid frames: $\{(w_i, i, i): 1 \leqslant i \leqslant |w|\}$.

Repeat

1. Add the frame (A, i, i) to VAL, if the grammar contains the production $A \to b$ and if (b, i, i) is in VAL.
2. Add the frame (A, l, r) to VAL, if the gramar contains the production $A \to BC$ and if (B, l, m) as well as (C, m, r) are in VAL.

Until no new frames can be added to VAL.

The membership algorithm for growing context-sensitive languages will follow the same outline. The frames will parameterize pieces of the derivation graphs. We need to be able to describe the paths bordering a piece on the left and on the right. One notation for paths of derivation graphs is given in Fig. 2. The productions are the labels of the production vertices on the path and the numbers specify which successors and predecessors are on the path. These numbers are necessary because for



$2/AA \to AAAA/(4,3)/AAA \to QRAA/(2,2)/QR \to Aw_{10}w_{11}/3$

FIG. 2. The description of a path (in boldface).

a given production $\alpha \to \beta$ in the grammar some symbols might have multiple occurrences in α or β. We could present the algorithm using the notation of Fig. 2 which would be more efficient. But for the sake of simplicity of the presentation we assume that the grammar is in a special form.

A grammar is called a *one-grammar* if for each production $\alpha \to \beta$ in the grammar each symbol of the alphabet occurs at most once in α and at most once in β. Using standard methods of Formal Language Theory, it is easy to construct an equivalent one-grammar for a given grammar by increasing the size of the alphabet and by adding chain productions. For *chain* productions $|\alpha| = |\beta| = 1$ must hold.

In the following we outline the construction of an equivalent one-grammar. Details are left to the reader. Assume there is a production $\alpha \to \beta$ in which some symbol A (terminal or nonterminal) appears twice in α. In this case the two occurrences of A in the production are replaced by two new nonterminals A_1 and A_2. Furthermore two new productions are added to the grammar: $A \to A_1$ and $A \to A_2$. By repeatedly applying the above, double occurences of symbols from the left-hand sides of productions are eliminated. With a similar construction we can eliminate double occurrences from the right-hand sides of productions.

Since for the membership problem we assume that the grammar is fixed, the size of the equivalent one-grammar will also be fixed and independent of the input word to be tested for membership. Observe that the original grammar and the equivalent one-grammar define the same language. Furthermore derivation graphs for the original grammar translate into derivation graphs for the corresponding one-grammar and vice versa. A path in the derivation graph of the one-grammar is at most three times as long as the corresponding path in the "equivalent" derivation of the original grammar.

This motivates the following assumptions for the rest of this section. The fixed growing csg of the membership problem is given by its equivalent one-grammar. Paths in derivation graphs of the latter grammar are *bounded* if they are of length $6\lceil \log_g(|w|)\rceil$ instead of $2\lceil \log_g(|w|)\rceil$. Theorem 1 also holds for the equivalent one-grammars with the new bound. (Recall that g is the growth-ratio of the original growing csg.)

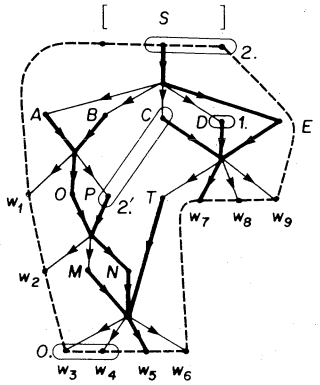
The input word w which is to be tested for membership is denoted as $w_1 w_2 \cdots w_{|w|}$. To get a simple description of the algorithm we add dummy symbols to the beginning and end of w. Let [and] denote two symbols which are not in the alphabet of the grammar. Set $w_0 = [$ and $w_{|w|+1} =]$.

To decribe a path $\pi = x_1, x_2, \ldots, x_e$ in a derivation graph of a one-grammar, it is now sufficient to use the sequence $\omega(x_1)/\omega(x_2)/\cdots/\omega(x_e)$ which is called the *labeling sequence* of π and is denoted by $\Omega(\pi)$.

Similarly to paths, a labeling sequence is *bounded* if it is of length at most $6\lceil \log_g(|w|)\rceil$. A *frame* is a tuple (t, λ, ρ, l, r) s.t. $t \in V^2 \cup V$, λ and ρ are bounded labeling sequences starting with the first respectively last symbol of t, and $0 \leqslant l \leqslant r \leqslant |w| + 1$.

Intuitively, the above frame specifies a "piece" of a planar derivation graph which might appear in the cut-and-paste process. This piece is bordered on the left (resp. right) by a path labeled with λ (resp. ρ). The piece derives $w_l, w_{l+1}, \ldots, w_r$, i.e., λ ends at a sink labeled with w_l, ρ ends at a sink labeled with w_r and the sinks in between are labeled accordingly. The word t specifies how the "piece" starts. If the left and right path start at the same vertex (*unary* frame) then t is the label of that vertex. In the case where the paths start at different vertices (*binary* frame), t consists of the labels of both vertices. See Fig. 3 for examples. The polynomial running time of the membership algorithm for fixed growing csls heavily relies on the fact that the number of frames is polynomial in $|w|$. Note that there is only a polynomial number of labeling sequences of bounded paths (length up to $6\lceil \log_g(|w|)\rceil$) since g is a positive constant.

Not every frame corresponds to a piece of a derivation graph, only valid frames do. A frame is *valid* if and only if it is a valid frame w.r.t. a bounded derivation



0. $(w_3 w_4, w_3, w_4, 3, 4)$
1. $(D, D/CDE \to Tw_7 w_8 w_9/w_7, D/CDE \to Tw_7 w_8 w_9/w_7, 7, 7)$
2. $(S], S/S \to ABCDE/E/CDE \to Tw_7 w_8 w_9/w_7,], 7, 10)$
2'. $(PC, P/OP \to w_2 MN/N/MNT \to w_3 w_4 w_5 w_6/w_5,$
$C/CDE \to Tw_7 w_8 w_9/w_7, 5, 7)$

FIG. 3. Some valid frames with respect to a derivation graph and a set of bounded consistent paths (in boldface); the symbols of the first component of each frame are encircled.

graph D and a consistent set of bounded paths $\Pi = \{\pi_y \mid \pi_y$ starts with the symbol vertex y of $D\}$ from each symbol vertex of D to a sink.[3]

DEFINITION. The valid frames of (D, Π) are given as follows:

(1) The unary frame $(\omega(v), \Omega(\pi_v), \Omega(\pi_v), l, l)$ is valid if

(i) v is a symbol vertex of D;

(ii) π_v ends at a sink labeled with w_l.

(2) The binary frame $(\omega(u)\,\omega(v), \Omega(\pi_u), \Omega(\pi_v), l, r)$ is valid if

(i) u and v are symbol vertices of D s.t. adding the edge (u, v) to D does not violate the planarity of D;

(ii) the edge (v, w) does not leave the planar circle which encloses all edges of D and is defined by the edges between adjacent sources, the edge between the rightmost source and the rightmost sink, the edges between adjacent sinks, and the edge between the leftmost sink and the leftmost source (see broken circle of Fig. 3);

(iii) there is no path from u to v and vice versa;

(iv) π_u ends at some sink s;

(v) the $r-l+1$ sinks starting from s going to the right are labeled with $w_l, w_{l+1}, \ldots, w_r$;

(vi) the $(r-l+1)$th such sink is the one at which π_v ends.

There are many valid frames beloning to (D, Π). For a particular frame we want to specify the subgraphs of derivation graphs which correspond to that frame. Let $F = (t, \lambda, \rho, l, r)$ be a valid frame of some tuple (D, Π). If F is unary then $\rho = \lambda$ and the path of vertices in D which corresponds to λ is an *instance* of F. In the case where F is binary then the subgraph I induced by the vertices v of D for which the following conditions hold is called an *instance* of the frame F:

(i) v has a predecessor amongst the two vertices corresponding to t;

(ii) π_v ends at a sink corresponding to w_m, where $l \leqslant m \leqslant r$;

(iii) if v is not on the path corresponding to λ but π_v and λ have some vertex x as their first common vertex, then the predecessor of x on λ is he left of the predecessor of x on π_v;

(iv) if v is not on the path ρ but π_v and ρ have some vertex x as their first common vertex, then the predecessor of x on ρ is to the right of the predecessor of x on π_v.

Intuitively I consists of all vertices of D "below" T, to the "right" of λ and to the "left" of ρ. Applying the above definition of valid frames gives the following equivalence.

[3] Note that D does not necessarily derive the whole word w, but in the case where w and the word derived by D have no subword in common then no valid frames belong to D.