On sentence membership problem in context-sensitive languages

P.A. RYSZAWA

pawel.ryszawa@wat.edu.pl

Military University of Technology, Faculty of Cybernetics
Kaliskiego Str. 2, 00-908 Warsaw, Poland

A new type of graph is introduced, the grammar graph. The possibility of assigning labels to each node in such a graph extends it to the grammar net. The grammar net should be considered as a new graphical tool that helps in an analysis of whether a particular sentence belongs to a given context-sensitive grammar. Another concept, the derivation net, closely related to the grammar graph and of a similar structure, will be used to show an algorithm that is able to decide that some sentences do not belong to a language generated by a context sensitive grammar, while leaving others as a candidate members of it.

Keywords: context-sensitive grammar, non-contracting grammar, formal language, graph, parsing.

1. Introduction

In the formal language theory, every grammar can be expressed in terms of an alphabet (finite set of terminal symbols) denoted by V_T, finite set of non-terminal symbols denoted by V_N and a set of production rules – i.e. “prescriptions” of how to derive a “correct” sentence over the alphabet with those symbols, starting from some non--terminal symbol S. Each production is of the form $\mu_1\mu_2 \dots \mu_m \to \gamma_1\gamma_2 \dots \gamma_n$, where μ_i, γ_i are symbols. A particular class of grammars, where $m < n$ and at least one of μ_i is non--terminal, is called non-contracting. If, for each production of the above form, n does not exceed some N, we call this a non-contracting grammar of N-th order. Every such grammar has its equivalent context-sensitive grammar and vice--versa, in the sense that both generate the same language (weak equivalence). By definition, context-sensitive grammars are those with productions of the form $\alpha\mu\beta \to \alpha\gamma_1\gamma_2 \dots \gamma_n\beta$, where $\alpha = \alpha_1\alpha_2 \dots \alpha_p$, $\beta = \beta_1\beta_2 \dots \beta_q$, α_i, β_i, γ_i are some symbols (terminal or non-terminal) and μ is some non-terminal symbol. For the sake of simplicity, it is assumed throughout this paper that an immaterial and trivial type of production $P\colon U \to \varepsilon$ (deriving empty string), where $U \in V_N$, is not included in the considered grammars.

It has been already proved (see e.g. [3]) that for each non-contracting grammar there exists an equivalent grammar of 2nd order. Moreover, one can easily construct an equivalent grammar where every production containing a terminal symbol a can only be of the form $A \to a$. Thus, every context-sensitive grammar can be equivalently expressed via a grammar with productions of the form: $A \to BC$, $AB \to CD$, $A \to B$ and $A \to a$, where capital letters denote non-terminal and small letters denote terminal symbols.

So far, there have been found many algorithms for parsing different kind of grammars. One of them, C-Y-K (see e.g. [5], [3]) – for context-free grammars, was an inspiration to construct the one presented it this paper. This new algorithm will be illustrated with one of the best known context--sensitive language $\{a^n b^n c^n \mid n \geq 1\}$.

2. Grammar graph and net

To start with, we need a new graphical “tool” to express the ideas laying behind the derivation of grammar trees. This is “grammar graph” which is a special kind of the directed bipartite graph with all the edges ordered in its endpoints. Two set of nodes are represented by rectangles and circles. Single-lined circle nodes will represent non-terminal symbols, double-lined circles will represent terminal symbols and rectangles will represent productions. More formally, the grammar graph $\tilde{G}$ representing some grammar $G = \langle V_N, V_T, P, S\rangle$ is modelled as:

$$\tilde{G} = \langle V, \Gamma_S, \Gamma_N^{-1}\rangle \tag{1}$$

where:
$V = V_N \cup V_T \cup P$ is a set of nodes identified with symbols and production rules of G,
V_N – a set of non-terminal symbol nodes,
V_T – a set of terminal symbols nodes (alphabet),
P – a set of production rules,

$\Gamma_S: P \rightarrow (V_N \cup V_T)^+$ is a mapping representing an ordered list (tuple) of the directed edges outgoing from each $P_i \in P$ to their successor symbol nodes, as pointed by the graph arrows.

$\Gamma_N^{-1}: P \rightarrow V_N^+$ is a ("reverse") mapping from a production node to an ordered list (tuple) of its predecessor non-terminal symbol nodes. The tuple must contain at least one element because every production in the underlying grammar has at least one symbol on the left. Please note the notation: $X^+ = X \times X^* = = X \times (X \cup X^2 \cup \ldots) = X^2 \cup X^3 \cup \ldots$ (X^* is the Kleene closure). For all productions $P_i \in P$ of the form $P_i: \mu_1\mu_2 \ldots \mu_m \rightarrow \gamma_1\gamma_2 \ldots \gamma_n$, the maps Γ_S and Γ_N^{-1} are subject to the following constraints:

$$\Gamma_S(P_i) = \langle \gamma_1, \gamma_2, \ldots, \gamma_n \rangle \tag{2}$$

$$\Gamma_N^{-1}(P_i) = \langle \mu_1, \mu_2, \ldots, \mu_m \rangle \tag{3}$$

In this paper, though, examples will be based on non-contracting grammars of order 2.

It is emphasized here that the above defined graph is of special kind. All the arrows incoming to and outgoing from a production node are ordered. This is expressed via the special form of the mapping Γ_S and Γ_N^{-1} that maps the production nodes to a set of tuples of nodes and not a family of their subsets! Of course, each tuple, by its nature, holds the information about the order. Graphically, it will be depicted by the arrow starting points placed from left to right on the edge of a production node symbol as per Γ_S. The same concerns the arrows incoming to a production node, all of them are strictly ordered based on Γ_N^{-1}.

Example 1

The first example shows the basic parts of grammar graphs for productions:

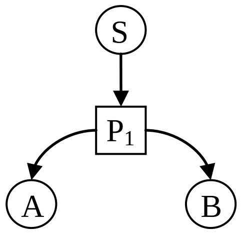

Fig. 1. A fragment of some grammar graph representing single production P_1: S⟶AB

The following productions P_i are considered: $P_1: S \rightarrow AB$, $P_2: S \rightarrow BA$, $P_3: A \rightarrow B$, $P_4: A \rightarrow a$, $P_5: AB \rightarrow CD$ and $P_6: BA \rightarrow CD$. The Figure 1 shows the first of them. Recall that the starting points of the arrows are ordered according to the order of symbols on the right-hand side in the corresponding production. Here, $\Gamma_S(P_1) = = \langle A, B \rangle$ and $\Gamma_N^{-1}(P_1) = \langle S \rangle$. The next graph, in Figure 2, although equivalent in terms of the classic graph definition to that of Figure 1, represents different production $S \rightarrow BA$. Here, the difference is that $\Gamma_S(P_2) = \langle B, A \rangle$.

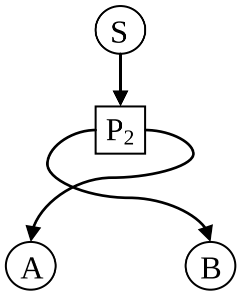

Fig. 2. A graph fragment representing P_2: S⟶BA

Next, a production of the form $A \rightarrow B$ is shown in Figure 3. Here, $\Gamma_S(P_3) = \langle B \rangle$ and $\Gamma_N^{-1}(P_3) = = \langle A \rangle$.

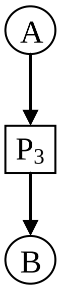

Fig. 3. A grammar graph representing single production P_3: A⟶B

A production of the form $A \rightarrow a$ is shown in Figure 4. Here, $\Gamma_S(P_4) = \langle a \rangle$ and $\Gamma_N^{-1}(P_4) = = \langle A \rangle$.

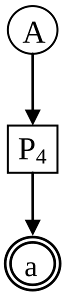

Fig. 4. A grammar graph representing single production P_4: A⟶a

A production of the form $AB \rightarrow CD$, specific to the non-contracting grammars, is shown in Figure 5.

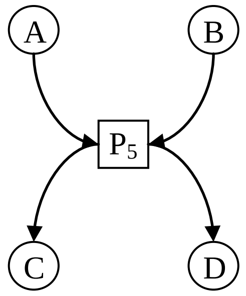

Fig. 5. A graph fragment representing P$_5$: AB→CD

In the above graph, the production node P_5 represents the order of the non-terminal nodes A and B by ordering the incoming arrows from left to right. $\Gamma_S(P_5) = \langle C, D\rangle$ and $\Gamma_N^{-1}(P_5) = \langle A, B\rangle$.

The last example, representing the production node P_6 is shown in Figure 6. It differs from P_5 in that it has $\Gamma_N^{-1}(P_6) = = \langle B, A\rangle$, i.e. the ordering of the predecessor nodes tuple is different.

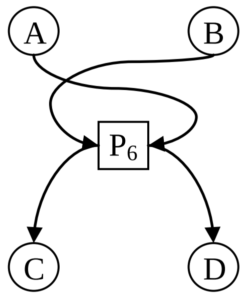

Fig. 6. A graph fragment representing P$_6$: BA→CD

▲

3. Labelling

Having defined the basic grammar graph structures, let us introduce an extension – a labelling on the grammar graph, thus, letting us derive the definition of the grammar net. A grammar net is a grammar graph with some labelling defined on it. The grammar graph $\tilde{G}$ **labelling**, denoted by $\mathfrak{L}_{\tilde{G}}$ is:

$$\mathfrak{L}_{\tilde{G}}: V \to 2^{(E^I)} \tag{4}$$

where:
E^I – the set of (so called) first order labels,
$V \to 2^{(E^I)}$ – a mapping that assigns a subset of first order labels to each grammar graph node.

Each **first order label** $e \in E^I$ is of the form $e = \langle \psi, a, b, \varphi\rangle$ (hereinafter shortly denoted as "label" e) belongs to the set:

$$E^I = \left\{ \langle \psi, a, b, \varphi\rangle \,\middle|\, \begin{matrix} 1 \le a \le b \le |\omega|; \\ a, b \in \mathbb{N}; \psi, \varphi \in E^{II} \end{matrix} \right\} \tag{5}$$

where:
a, b – correspond to the position of the first and the last character of some sentence ω, respectively,
E^{II} – the set of second order labels,
$\psi, \varphi \in E^{II}$ – some second order labels.

A **second order label** is of the form $\psi = \langle a, j, \Theta, b\rangle \in E^{II}$ and

$$E^{II} = \left\{ \langle a, j, \Theta, b\rangle \,\middle|\, \begin{matrix} 0 \le a \le b \le |\omega| + 1; \\ a \in \mathbb{Z}; b \in \mathbb{N}; \Theta \in \bar{P}; j \in \mathbb{N} \end{matrix} \right\} \tag{6}$$

where:
a, b – have the same meaning as for the first order labels,
$\bar{P} = P \cup \{\#\}$ – is a set of productions extended with a special mark #,
j – is some natural number,
Θ – is some production from the grammar G or the special mark #.
To simplify the notation, the left part of the label of the form $\langle\langle a-1, 1, \Theta, a\rangle, a \ldots$ will be shortly denoted by $\langle \Theta; a \ldots$. Meanwhile, the right part of the label of the form $\ldots b, \langle b, 1, \Theta, b+1\rangle\rangle$ will be denoted by $\ldots b; \Theta\rangle$.

Now, let $\mathfrak{L}_{\tilde{G}}$ be some labelling. Since the grammar graph $\tilde{G}$ is unambiguously defined by the corresponding grammar G, the labelling on $\tilde{G}$ can be identified with G as well, i.e.

$$\mathfrak{L}_G \overset{\text{def}}{=} \mathfrak{L}_{\tilde{G}}. \tag{7}$$

Definition 1
Let us also impose a partial order in the set of all possible labelling on $\tilde{G}$, denoted by $\succcurlyeq$, as follows: $\mathfrak{L}^1_{G'} \succcurlyeq \mathfrak{L}^2_{G''}$ if and only if $\mathfrak{L}^1_{G'}(v) \supseteq \mathfrak{L}^2_{G''}(v)$, for $v \in V' \supseteq V''$, where $V' = V'_N \cup V'_T$, $V'' = V''_N \cup V''_T$ for $G' = = \langle V'_N, V'_T, P', S'\rangle$ and $G'' = \langle V''_N, V''_T, P'', S''\rangle$. ●

The labelling closure $\overline{\mathfrak{L}_G}$ is the minimal labelling "generated" by $\mathfrak{L}_G$ according to the following rules:

a) (**initial rule**) The closure contains all of the elements from the generating labelling, i.e.

$$\overline{\mathfrak{L}_G} \succcurlyeq \mathfrak{L}_G. \tag{8}$$

b) (**joining rule**) If for a production $P_i \in P$ of the form $P_i: \gamma \to B_1 B_2 \ldots B_m$, where $\gamma \in (V_N)^*$ and $B_1, B_2, \ldots, B_m \in V_T \cup V_N$, there exist $\psi_0, \psi_1, \ldots \psi_m \in E^{II}$ and $a_1, \ldots, a_m \in \mathbb{N}$, $b_1, \ldots, b_m, \in \mathbb{N}$ such that for each $k = 1, 2, \ldots, m$:

$$e_k = \langle \psi_{k-1}, a_k, b_k, \psi_k\rangle \in \overline{\mathfrak{L}_G}(B_k) \tag{9}$$

and for each $k = 1,2,\dots,m-1$:

$$\psi_k = \langle b_k, j_k, \Theta_k, a_{k+1}\rangle \qquad (10)$$

for some $\Theta_k \in \bar{P}$, $j_k \in \mathbb{N}$, then:

$$e = \langle \psi_0, a_0, b_m, \psi_m\rangle \in \overline{\mathfrak{L}_G}(P_i). \qquad (11)$$

c) (**splitting rule**) If for a production $P_i \in P$ of the form $P_i: A_1A_2 \dots A_n \to \gamma$, where $A_1, A_2, \dots, A_n \in V_N$ and $\gamma \in (V_T \cup V_N)^*$ there exists a label e such that:

$$e = \langle \psi, a, b, \varphi\rangle \in \overline{\mathfrak{L}_G}(P_i) \qquad (12)$$

then:

$$\langle \psi, a, b, \langle a, 1, P_i, b\rangle\rangle \in \overline{\mathfrak{L}_G}(A_1),$$

$$\langle\langle a, j-1, P_i, b\rangle, a, b, \langle a, j, P_i, b\rangle\rangle \in \overline{\mathfrak{L}_G}\big(A_j\big),$$

for $j = 2, \dots, n-1$, and

$$\langle\langle a, n-1, P_i, b\rangle, a, b, \varphi\rangle \in \overline{\mathfrak{L}_G}(A_n). \qquad (13)$$

Example 2
Assume that the production rule $P_1: A \to a$ belongs to some grammar against which we test the sentence $aaba$ and the labelling for the terminal node contains 3 labels: $\langle\#;1,1;\#\rangle$, $\langle\#;2,2;\#\rangle$ and $\langle\#;4,4;\#\rangle$. After applying the labelling closure routine the production rule P_1 node and its predecessor, the non-terminal symbol node A, also contains those 3 labels – see Figure 7.

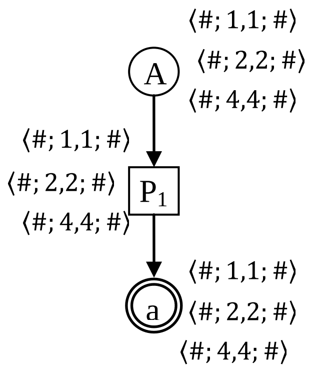

Fig. 7. Labelling example for production P_1: A→a

Assume also that the grammar contains the production rule $P_2: S \to AB$, the node B is labelled with $\langle\#;3,3;\#\rangle$ and the node A is labelled with $\langle\#;1,1;\#\rangle$, $\langle\#;2,2;\#\rangle$ and $\langle\#;4,4;\#\rangle$.

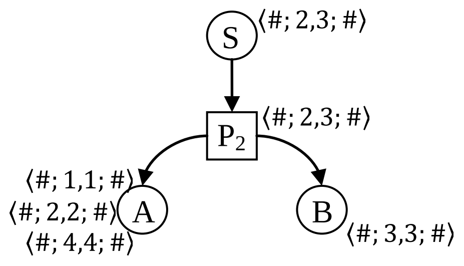

Fig. 8. Labelling example for production P_2: S→AB

According to the joining rule the labelling closure must contain also $\langle\#;2,3;\#\rangle$ for P_2 as $\langle\#;2,2;\#\rangle$ and $\langle\#;3,3;\#\rangle$ meet at the arrows outgoing from P_2. This is further copied to the node S, according to the splitting rule (without the actual splitting as there is only one predecessor). See Figure 8. ▲

Example 3
Assume we have some grammar with a production rule $P_3: AB \to CD$. The symbol node D is labelled with $\langle\langle 4,1,P_1,5\rangle, 4,7, \langle 6,1,P_2,7\rangle\rangle$ and the symbol node C is labelled with $\langle\#;3,3;\#\rangle$ and $\langle\#;3,5,\langle 4,1,P_1,5\rangle\rangle$ – see Figure 9.

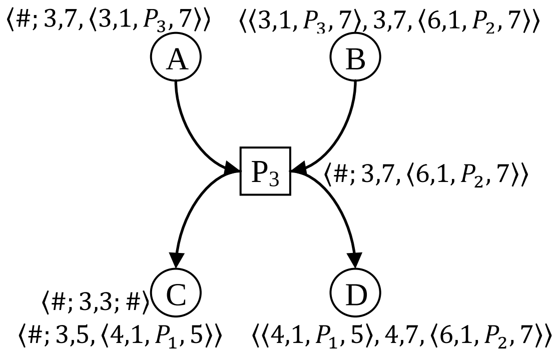

Fig. 9. Labelling example for production P_3: AB→CD

Two labels, namely $\langle\#;3,5,\langle 4,1,P_1,5\rangle\rangle$ and $\langle\langle 4,1,P_1,5\rangle, 4,7, \langle 6,1,P_2,7\rangle\rangle$, meet the joining criteria as they share second order label $\langle 4,1,P_1,5\rangle$ on their appropriate sides. Thus, they are entitled to be joined together in the production rule node P_3 to produce $\langle\#;3,7,\langle 6,1,P_2,7\rangle\rangle$. Finally, according to the splitting rule, this further produces labels $\langle\#;3,7,\langle 3,1,P_3,7\rangle\rangle$ and $\langle\langle 3,1,P_3,7\rangle, 3,7,\ \langle 6,1,P_2,7\rangle\rangle$ for the nodes A and B, respectively. ▲

The overall algorithm of labelling closure on a 2nd order non-contracting grammar net is that:

a) It copies all the labels from the initial labelling.
b) It joins two labels L and R, one from the left and one from the right successor symbol node of some production node, respectively, if their second order labels match in that the right second order label of L is the same as the left second order label of R. It means that $L = \langle\theta_L, a_L, b_L, \eta\rangle$ and $R = \langle\eta, a_R, b_R, \theta_R\rangle$ produce $\langle\theta_L, a_L, b_R, \theta_R\rangle$ in their common parent node.
c) It splits any label in a production node having two predecessors into two labels. The left predecessor receives a copy of the left second order label and the two numbers but with a new right second order label. The same second order label is received by the new label assigned to the right predecessor as his left one second order label, the two numbers and the right second order label are copied from the original label. It means that $\langle\theta_L, a, b, \theta_R\rangle$ in a production node P_i with two predecessors (left and right ones) produces $L = \langle\theta_L, a, b, \langle a, 1, P_i, b\rangle\rangle$ for the left predecessor and $R = \langle\langle a, 1, P_i, b\rangle,\ a, b, \theta_R\rangle$ for the right predecessor.
d) Copies all the labels unchanged from a node to its unique predecessor, i.e. if there exists one predecessor only.

Intuitively, splitting of a label into two, giving them matching second order labels $\langle a, 1, P_i, b\rangle$, expects that later another two labels with matching second order label $\langle a, 1, P_i, b\rangle$, derived from the above two ones, will join again. The "left" label will grow further on its left-hand side, while $\langle a, 1, P_i, b\rangle$ as its right second order label will "wait for meeting" another such second order label as the left one of another label that, possibly, could have "grown up" on its right-hand side. It means that, applying subsequently the splitting and joining rules, $\langle\theta_L, a, b, \langle a, 1, P_i, b\rangle\rangle$ may grow up on the left to some $\langle\theta_L{}', a', b, \langle a, 1, P_i, b\rangle\rangle$, where $a' \leq a$, and $\langle\langle a, 1, P_i, b\rangle, a, b, \theta_R\rangle$ may grow up to $\langle\langle a, 1, P_i, b\rangle, a, b', \theta_R{}'\rangle$, where $b \leq b'$. Finally, the two grown up labels may meet at some production node to join again, thus producing $\langle\theta_L{}', a', b', \theta_R{}'\rangle$. It will be described later.

4. Derivation graph and net

Let us now construct a new kind of graph for the sentence ω, denoted by $D_\omega(G)$ – a derivation graph. Assume first that the following chain of direct derivations is the derivation of ω in G:

$$S = \gamma_0 \to \gamma_1 \to \gamma_2 \to \cdots \to \gamma_m = \omega. \quad (14)$$

where i-th sentential form γ_i, of length J_i, is:

$$\gamma_i = \gamma_{i,1}\gamma_{i,2} \dots \gamma_{i,J_i} \in (V_N \cup V_T)^{J_i}, \quad \gamma_{i,j} \in V_N \cup V_T \quad (15)$$

Without loss of generality we can assume that there exist no two identical sentential forms in the derivation of ω, i.e.

$$i_1 \neq i_2 \Rightarrow \gamma_{i_1} \neq \gamma_{i_2}. \quad (16)$$

Indeed, if there existed some $\gamma_a = \gamma_{a_1} = \gamma_{a_2}$, such that

$$S = \gamma_0 \to \cdots \to \gamma_{a_1} \to \cdots \to \gamma_{a_2} \dots \to \gamma_m = \omega \quad (17)$$

then the mid-derivation $\dots \to \gamma_{a_1} \to \cdots \to \gamma_{a_2} \to \cdots$ could be just contracted to $\dots \to \gamma_a \to \cdots$, hence

$$\gamma_0 \to \cdots \to \gamma_a \to \cdots \to \gamma_m. \quad (18)$$

It must be noted here, that there could exist more than one correct derivation for the sentence ω! Until this is immaterial in this paper, let it be any of them. Moreover, assume that the above derivation does not contain any "weak cycle" in the following sense:

Definition 2
Let $\gamma_i \xrightarrow{P_{a_{i+1}}} \gamma_{i+1} \xrightarrow{P_{a_{i+2}}} \dots \xrightarrow{P_{a_j}} \gamma_j$ be a part of some derivation from the i-th to the j-th step with productions $P_{a_{i+1}}, P_{a_{i+2}}, \dots, P_{a_j}$. Assume that a concatenation of 3 parts can be distinguished in every sentential form: $\gamma_k = \varphi_k\chi_k\psi_k$, $k \in \overline{i,j}$, $\varphi_k, \chi_k, \psi_k \in (V_N \cup V_T)^*$, possibly of length 0, where $\varphi_k = \varphi_{k,1}\varphi_{k,2} \dots \varphi_{k,|\varphi_k|}$, $\chi = \chi_{k,1}\chi_{k,2} \cdots \chi_{k,|\chi_k|}$, $\psi_k = \psi_{k,1}\psi_{k,2} \dots \psi_{k,|\psi_k|}$. Assume also that the 3 parts φ, χ or ψ were chosen such that every direct derivation $\varphi_k\chi_k\psi_k \xrightarrow{P_{a_{k+1}}} \varphi_{k+1}\chi_{k+1}\psi_{k+1}$ affects either χ-part or one of the remaining two ones – φ- or ψ-part. That is, one of the following cases occurs:
a) a production $P_{a_{k+1}}: \chi_{k,x+1}\chi_{k,x+1} \cdots \chi_{k,x+m} \to \chi_{k+1,y}\chi_{k+1,y+1} \cdots \chi_{k+1,y+n}$ starts with a subpart of the part χ_k and derives another subpart within χ_{k+1}; $\varphi_k = \varphi_{k+1}$ and $\psi_k = \psi_{k+1}$,
b) either $P_{a_{k+1}}: \varphi_{k,x'+1}\varphi_{k,x'+1} \cdots \varphi_{k,x'+m} \to \varphi_{k+1,y'+1}\varphi_{k+1,y'+2} \cdots \varphi_{k+1,y'+n}$ (or $P_{a_{k+1}}: \psi_{k,x''}\psi_{k,x''+1} \dots \psi_{k,x''+m} \to$

$\psi_{k+1,y''+1}\psi_{k+1,y''+2}\cdots\psi_{k+1,y''+n})$ starts with a subpart of the part φ_k (or ψ_k) and derives another subpart within φ_{k+1} (or ψ_{k+1}, respectively) while leaving $\chi_k = \chi_{k+1}$.

Finally, assume that $\chi_i = \chi_j$. If so, the sequence of all the steps where the first of the above two cases takes place is called **weak cycle**. ●

Observe that stripping a derivation from weak cycles does not affect it as a whole. This is because all the productions in some weak cycle process sequential forms only within their χ-parts, finally yielding the same χ-part as the initial one, while leaving other two parts φ and ψ intact. In parallel, all of the other productions still does not affect χ-part within the boundaries of such a weak cycle (between i-th and j-th steps). Observe also that the productions from a weak cycle may be interleaved with all the other productions within the respective boundaries in more than one way, provided that the order within each of the two classes remains unchanged. This is the result of the observation that the two classes are completely independent within those boundaries.

Now, another restriction can be imposed on the sentence derivations. Unless stated otherwise, all the derivations considered in this paper are assumed not to have weak cycles. If a derivation contained any, it could be shortened, of course, and the weak cycle would disappear.

Definition 3

The following structure is a **derivation graph** of the sentence ω over the grammar G for the derivation as defined in (14):

$$D_\omega(G) = \langle \hat{V}, \hat{E}, \hat{L}, \hat{M} \rangle \tag{19}$$

where:

$\hat{V} = \hat{V}_S \cup \hat{V}_P$ – set of nodes, $\hat{V}_S \cap \hat{V}_P = \emptyset$, where $\hat{V}_S$ represents some symbols and $\hat{V}_P$ represents some productions of G,

$\hat{E} \subseteq \hat{V}_S \times \hat{V}_S \cup \hat{V}_S \times \hat{V}_P \cup \hat{V}_P \times \hat{V}_S \subseteq \hat{V} \times \hat{V}$ – a set of directed edges where none of them joins directly two nodes representing productions,

$\hat{L} = \langle \hat{L}_0, \hat{L}_1, \ldots, \hat{L}_m \rangle$ – a sequence of layers, each consisting of a sequence of nodes $\hat{L}_i = \langle \hat{\gamma}_{i,1}, \hat{\gamma}_{i,2}, \ldots, \hat{\gamma}_{i,J_i} \rangle$, where $\hat{\gamma}_{i,j} \in \hat{V}_S$ for all i, j, so each node from $\hat{V}_S$ belongs to an exactly one layer and all the nodes in a particular layer are different; this is ordering information for the nodes,

$\hat{M}: \hat{V} \to V_N \cup V_T \cup P$ – a mapping from nodes to the set of symbols (terminal and non-terminal ones) and productions in the underlying grammar G, such that

$$\forall_{\zeta \in \hat{V}_P} \hat{M}(\zeta) \in P,$$
$$\forall_{\xi \in \hat{V}_S} \hat{M}(\xi) \in V_N \cup V_T, \tag{20}$$

subject to the following restrictions: Let $\gamma_k = \varphi\chi_k\psi$, $\gamma_{k+1} = \varphi\chi_{k+1}\psi$, $P_{a_{k+1}}: \chi_k \to \chi_{k+1}$, where $\varphi, \chi_k, \chi_{k+1}, \psi \in (V_N \cup V_T)^*$. If $\varphi = \gamma_1\gamma_2\cdots\gamma_x$, $\chi_k = \gamma_{x+1}\gamma_{x+2}\cdots\gamma_{x+p}$, $\psi = \gamma_{x+p+1}\gamma_{x+p+2}\cdots\gamma_{J_k}$ and $\chi_{k+1} = \gamma'_1\gamma'_2\cdots\gamma'_q$ then

$$\hat{L}_k = \langle \hat{\gamma}_{k,1}, \ldots, \hat{\gamma}_{k,x}, \hat{\gamma}_{k,x+1}, \ldots, \hat{\gamma}_{k,x+p}, \hat{\gamma}_{k,x+p+1}, \ldots, \hat{\gamma}_{k,J_k} \rangle$$
$$\hat{L}_{k+1} = \langle \hat{\gamma}_{k+1,1}, \ldots, \hat{\gamma}_{k+1,x}, \hat{\gamma}_{k+1,x+1}, \ldots, \hat{\gamma}_{k+1,x+q}, \hat{\gamma}_{k+1,x+q+1}, \ldots, \hat{\gamma}_{k+1,J_{k+1}} \rangle \tag{21}$$

and

$$\begin{aligned}
\hat{M}(\hat{\gamma}_{k,1}) &= \hat{M}(\hat{\gamma}_{k+1,1}) = \gamma_1,\\
&\vdots\\
\hat{M}(\hat{\gamma}_{k,x}) &= \hat{M}(\hat{\gamma}_{k+1,x}) = \gamma_x,\\
\hat{M}(\hat{\gamma}_{k,x+1}) &= \gamma_{x+1},\\
&\vdots\\
\hat{M}(\hat{\gamma}_{k,x+p}) &= \gamma_{x+p},\\
\hat{M}(\hat{\gamma}_{k+1,x+1}) &= \gamma'_1,\\
&\vdots\\
\hat{M}(\hat{\gamma}_{k+1,x+q}) &= \gamma'_q,\\
\hat{M}(\hat{\gamma}_{k,x+p+1}) &= \hat{M}(\hat{\gamma}_{k+1,x+q+1}) = \gamma_{x+p+1},\\
&\vdots\\
\hat{M}(\hat{\gamma}_{k,J_k}) &= \hat{M}(\hat{\gamma}_{k+1,J_{k+1}}) = \gamma_{J_k}.
\end{aligned} \tag{22}$$

Of course,

$$J_{k+1} = J_k - p + q, \tag{23}$$

must hold. Next, $\hat{V}_P = \{\hat{P}_1, \hat{P}_2, \ldots, \hat{P}_m\}$ and

$$\hat{M}(\hat{P}_{k+1}) = P_{a_{k+1}} \tag{24}$$

for $\gamma_k \xrightarrow{P_{a_{k+1}}} \gamma_{k+1}$. Finally, the set of edges contains pairs of the form:

$$\begin{gathered}
\langle \hat{\gamma}_{k,1}, \hat{\gamma}_{k+1,1} \rangle \in \hat{E},\\
\cdots\\
\langle \hat{\gamma}_{k,x}, \hat{\gamma}_{k+1,x} \rangle \in \hat{E},\\
\langle \hat{\gamma}_{k,x+1}, \hat{P}_{k+1} \rangle \in \hat{E},\\
\cdots\\
\langle \hat{\gamma}_{k,x+p}, \hat{P}_{k+1} \rangle \in \hat{E},
\end{gathered}$$

$$\langle \hat{P}_{k+1}, \hat{\gamma}_{k+1,x+1} \rangle \in \hat{E},$$
$$\dots$$
$$\langle \hat{P}_{k+1}, \hat{\gamma}_{k+1,x+q} \rangle \in \hat{E},$$

$$\langle \hat{\gamma}_{k,x+p+1}, \hat{\gamma}_{k+1,x+q+1} \rangle \in \hat{E},$$
$$\dots$$
$$\langle \hat{\gamma}_{k,J_k}, \hat{\gamma}_{k+1,J_{k+1}} \rangle \in \hat{E}, \tag{25}$$

for each k, where x – as defined for $\hat{L}_k$ and $\hat{L}_{k+1}$ in (21). ●

The following figure illustrates a part of derivation graph with two adjacent layers:

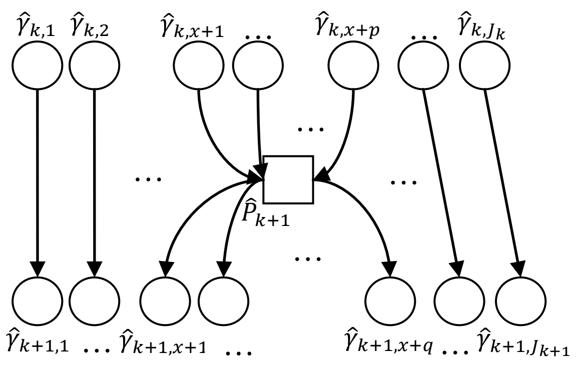

Fig. 10. Two neighbouring layers of a derivation graph

The general idea of connecting the nodes with arrows is as follows:

a) A node $\hat{\gamma}_{k,l}$ in the k-th layer that corresponds to a symbol being part of the left side of the production rule $P_{a_{k+1}}$ in the $(k+1)$-th derivation step points to the production rule node following the layer, i.e. this is a case where $x+1 \le l \le x+p$ (note that x depends on k).

b) A node $\hat{\gamma}_{k,l}$ that does not corresponds to a symbol being part of the left side of the production rule $P_{a_{k+1}}$ points to the corresponding node in the next layer, this a the case $l \le x$ or $x+p+1 \le l$. Moreover, both such nodes maps to the same symbol in the underlying grammar G.

c) The node $\hat{P}_{k+1}$ between the layers k-th and $(k+1)$-th points to the nodes in the $(k+1)$-th layer that correspond to the right-hand side symbols of the production rule $P_{a_{k+1}}$ in the $(k+1)$-th derivation step.

Example 4

Assume that some grammar G consists of the following production rules: $P_1: S \to AB$, $P_2: A \to YX$, $P_3: XB \to BZ$, $P_4: Z \to B$, $P_5: B \to X$, $P_6: Y \to y$, $P_7: X \to x$ and $P_8: B \to b$. Consider a possible derivation of the sentence yxb, for example $S \overset{P_1}{\to} AB \overset{P_2}{\to} YXB \overset{P_3}{\to} YBZ \overset{P_5}{\to} YXZ \overset{P_6}{\to} yXZ \overset{P_4}{\to} yXB \overset{P_7}{\to}$ $\overset{P_7}{\to} yxB \overset{P_8}{\to} yxb$. The obtained derivation graph is depicted in Figure 11.

Inside each node there is written a value that the node maps to in the grammar G. For instance, $\hat{M}(\hat{P}_6) = P_4$ and $\hat{M}(\hat{\gamma}_{8,1}) = y$.

The derivation steps represented by $\hat{P}_3$, $\hat{P}_4$ and $\hat{P}_6$ form a weak cycle in the derivation. The subnet starting in the layer 2 with 2 nodes mapping to X and B ends up with another two nodes in the layer 6 mapping to exactly the same X and B. This subnet is absolutely independent of the other part of the derivation net. No step in this subnet could affect the steps outside it and *vice versa*. Hence, the steps in question can be omitted without affecting the whole derivation. Thus, a reduced derivation graph can be obtained – as depicted in Figure 12 (nodes are reindexed). The derivation itself shortens to $S \overset{P_1}{\to} AB \overset{P_2}{\to} YXB \overset{P_6}{\to} yXB \overset{P_7}{\to} yxB \overset{P_8}{\to} yxb$.

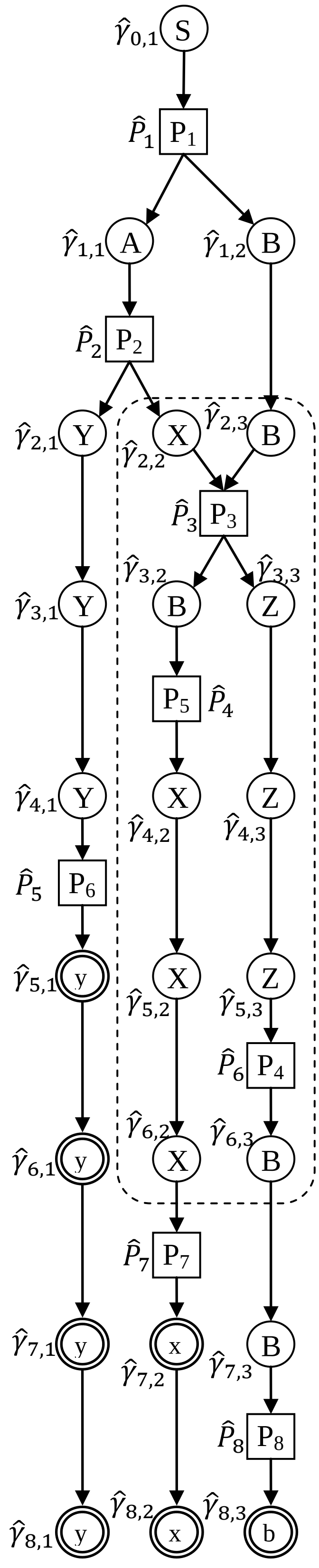

Fig. 11. A derivation graph example

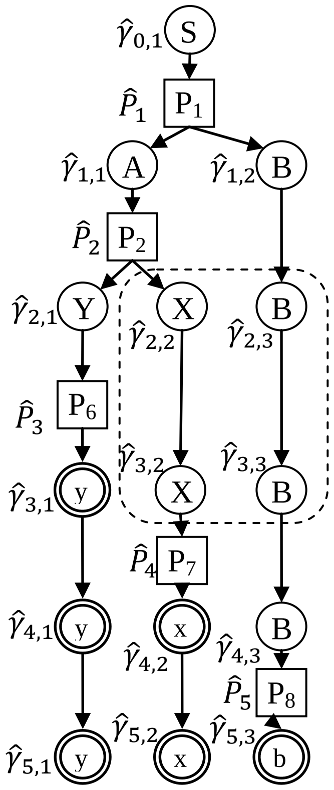

Fig. 12. A derivation graph reduction example

▲

Lemma 1
The derivation graph $D_\omega(G)$ is planar.

Proof
Indeed, the first layer of nodes (consisting of the starting node only) is planar by its nature. Assume now that the first k layers, with production nodes between each two adjacent of them, form a planar graph, say G_k. Assume also that no arrow of G_k protrudes from the area limited by the "very outer" nodes of G_k and that the nodes of the k-th layer are amongst those very outer ones. Now, extend G_k to G_{k+1} by adding the production node $\hat{P}_{k+1}$, symbol nodes $\gamma_{k+1,1}, \gamma_{k+1,2}, \dots, \gamma_{k+1,J_{k+1}}$ and all the arrows between them, as depicted in figure 10. Next, join them to $\gamma_{k,1}, \gamma_{k,2}, \dots, \gamma_{k,J_k}$ with appropriate arrows. Since G_k is planar and the new nodes and arrows, which itself would form a planar graph, are connected to "outer" nodes $\gamma_{k+1,1}, \gamma_{k+1,2}, \dots, \gamma_{k+1,J_k}$, not by-passed by any arrow of G_k, the resulting G_{k+1} is planar. Moreover, now all the nodes of the $(k+1)$-th layer are those amongst the most outer ones. By induction, we conclude as in the lemma. ■

Similarly to the definition of grammar net, that extended the grammar graph with some labelling, we can now extend the definition of derivation graph to the derivation net. A labelling of the form

$$\widehat{\mathfrak{L}}_G^{\omega}: \hat{V} \to E^I \tag{26}$$

over the derivation graph $D_\omega(G)$ together with $D_\omega(G)$ itself is a **derivation net**.

Each node in the derivation graph has a label assigned to it, thus forming a derivation net. The interpretation is similar to the labelling of grammar graphs except that it allows each node to have exactly one label. Similar are also the rules of deriving new labels from those already existing, except that:

a) new labels are successively derived upwards layer by layer, starting from the last one,
b) the labels derived in the $(k-1)$-th layer are based only on the labels from the k-th layer,
c) the labels in the last layer are chosen arbitrarily.

It will be shown in more details in the next sections along with a possible way to do such labelling, i.e. assigning exactly one label to each node.

Let us now define another concept – quotient derivation net. Let a pair $\langle D_\omega(G), \widehat{\mathfrak{L}}_G^{\omega}\rangle$ of a derivation graph and a labelling on it be a derivation net. The **quotient derivation net** over a mapping $\widehat{M}$ is a net constructed by contracting those nodes from $D_\omega(G)$ that are mapped by $\widehat{M}$ to the same symbol or production in G. It is worth noting here that the derivation graph $D_\omega(G)$, and hence a derivation net defined on it, holds an implicit information on the incoming and outgoing arrows in the order of nodes in each layer. Likewise in the grammar graph, it is required that the order of all the arrows incoming and outgoing from production-related nodes after the contraction is preserved. Thus, the quotient derivation net is defined as a pair $\langle \tilde{G}_{\omega,\widehat{M}}, \mathfrak{L}_{\tilde{G}_{\omega,\widehat{M}}}\rangle$ where

$$\tilde{G}_{\omega,\widehat{M}} = \langle \tilde{V}_{\omega,\widehat{M}}, \tilde{\Gamma}_{S,\omega,\widehat{M}}, \tilde{\Gamma}^{-1}_{N,\omega,\widehat{M}}\rangle \tag{27}$$

is a graph of a structure similar to that of the grammar graph and $\mathfrak{L}_{\tilde{G}_{\omega,\widehat{M}}}$ is a labelling over it. The above graph is constructed as follows:

$$\tilde{V}_{\omega,\widehat{M}} = \tilde{V}_{S,\omega,\widehat{M}} \cup \tilde{V}_{P,\omega,\widehat{M}} \tag{28}$$

is the set of nodes in $\tilde{G}_{\omega,\widehat{M}}$ where

$$\tilde{V}_{S,\omega,\widehat{M}} = \{v \in V_N \cup V_T \mid \exists_{\hat{v}\in\hat{V}} \widehat{M}(\hat{v}) = v\} \tag{29}$$

is the subset of those contracted nodes that correspond to symbols in the grammar G and

$$\tilde{V}_{P,\omega,\widehat{M}} = \{p \in P \mid \exists_{\hat{v}\in\hat{V}} \widehat{M}(\hat{v}) = p\} \tag{30}$$

is, accordingly, the subset of those that correspond to productions in G. Likewise in the definition of grammar graph, the nodes are identified with productions and symbols in the underlying grammar as this should never lead to ambiguity. Next,

$$\tilde{\Gamma}_{S,\omega,\widehat{M}}: \tilde{V}_{P,\omega,\widehat{M}} \to \left(\tilde{V}_{S,\omega,\widehat{M}}\right)^+ \tag{31}$$

is a mapping that assigns ordered sequences of symbol-related nodes to each production-related ones. Every sequence represents the order in which arrows outgoing from a production node point to symbol nodes. Similarly,

$$\tilde{\Gamma}^{-1}_{N,\omega,\widehat{M}}: \tilde{V}_{P,\omega,\widehat{M}} \to \left(\tilde{V}_{S,\omega,\widehat{M}}\right)^+ \tag{32}$$

is a “reversed” mapping for arrows incoming into the production-related nodes and represent their order. Moreover, for all $P_i \in \tilde{V}_{P,\omega,\widehat{M}}$ of the form $P_i: \mu_1\mu_2 \dots \mu_m \to \gamma_1\gamma_2 \dots \gamma_n$ we have

$$\tilde{\Gamma}_{S,\omega,\widehat{M}}(P_i) = \langle \gamma_1, \gamma_2, \dots, \gamma_n\rangle \tag{33}$$

and

$$\tilde{\Gamma}^{-1}_{N,\omega,\widehat{M}}(P_i) = \langle \mu_1, \mu_2, \dots, \mu_m\rangle. \tag{34}$$

The above formulas are similar to that of the grammar graph, compare (2) and (3). We can always contract the underlying derivation graph (or net) in this way because all the production nodes in $D_\omega(G)$ that maps with $\widehat{M}$ to the same production in G must have predecessors and successors that, in their order, map to the same symbols in G. $D_\omega(G)$ always correspond to a correct derivation of ω, so the graph itself must have the above property.

On the other hand, the contracted graph does not contain any arrow between two different symbol nodes. This, in turn, is a result of the fact that the only such arrows in the underlying derivation graph (or net) are those that link two corresponding nodes in adjacent layers. Both nodes in such pairs, though, always map with $\widehat{M}$ to the same symbol in G. Thus, $\tilde{\Gamma}_{S,\omega,\widehat{M}}$ and $\tilde{\Gamma}^{-1}_{N,\omega,\widehat{M}}$ cover all the necessary arrows in $\tilde{G}_{\omega,\widehat{M}}$.

Due to its definition, production related nodes in $\tilde{G}_{\omega,\widehat{M}}$ are also production related nodes in $\tilde{G}$. So do the symbol nodes and, moreover, the predecessors and successors of each production related node, along with their orders, are the same as in $\tilde{G}$. Thus, we obtain the following result:

Corollary 1
$\tilde{G}_{\omega,\hat{M}}$ is a subgraph of $\tilde{G}$.

As regards to the labelling $\mathfrak{L}_{\tilde{G}_{\omega,\hat{M}}}$, this is defined as follows:

$$\mathfrak{L}_{\tilde{G}_{\omega,\hat{M}}}(v) = \{\hat{\mathfrak{L}}_G^{\omega}(v') \mid \hat{M}(v') = v\}. \quad (35)$$

All the nodes in $D_\omega(G)$ that map with $\hat{M}$ to the same node in $\tilde{G}_{\omega,\hat{M}}$ "transfer" their labels into the set of labels assigned to the resulting contracted node. The above definitions of the contracted derivation graph and the labelling on it both form the definition of the quotient derivation net.

5. Membership problem

Let $\mathfrak{L}_G^{\omega}$ denote a "minimal" labelling over $\tilde{G}$ with respect to the sentence $\omega = w_1 w_2 \dots w_p \in (V_T)^p$ of length p such that

$$\mathfrak{L}_G^{\omega}(w_l) = \{\langle \#; l, l; \# \rangle\} \quad (36)$$

and

$$\mathfrak{L}_G^{\omega}(s) = \emptyset \quad (37)$$

where $s \neq w_l$, for all $l = 1, 2, \dots, p$. Every label from the initial labelling defined above can be unambiguously assigned to a particular node in the last (z-th) layer of $D_\omega(G)$. It means that $\langle \#; 1,1; \# \rangle$, $\langle \#; 2,2; \# \rangle$, …, $\langle \#; J_z, J_z; \# \rangle$, where $J_z = p$, correspond to the nodes $\hat{\gamma}_{z,1}$, $\hat{\gamma}_{z,2}$, …, $\hat{\gamma}_{z,J_z}$, respectively. So, the labelling $\hat{\mathfrak{L}}_G^{\omega}$ is correctly defined for this layer, i.e.

$$\hat{\mathfrak{L}}_G^{\omega}(\hat{\gamma}_{z,l}) = \langle \#; l, l; \# \rangle \quad (38)$$

Now, the following rules applied recursively to the nodes of $D_\omega(G)$ layer by layer will correctly define all the values of $\hat{\mathfrak{L}}_G^{\omega}$:

a) (**rewriting rule**) If a symbol node $\hat{\gamma}_{i+1,j'}$ is the direct successor of another symbol node $\hat{\gamma}_{i,j}$, i.e. $\langle \hat{\gamma}_{i,j}, \hat{\gamma}_{i+1,j'} \rangle \in \hat{E}$, then we let $\hat{\mathfrak{L}}_G^{\omega}(\hat{\gamma}_{i,j}) = \hat{\mathfrak{L}}_G^{\omega}(\hat{\gamma}_{i+1,j'})$. Recall that this is possible only if both of them maps with $\hat{M}$ to the same symbol in G and they correspond to those parts of γ_i and γ_{i+1} in the derivation step $\gamma_i \xrightarrow{P_{a_{i+1}}} \gamma_{i+1}$ that are not affected by $P_{a_{i+1}}$. Moreover, after the contraction to a quotient derivation net, both nodes will reduce to the same node with one label.

b) (**joining rule**) If symbol nodes $\hat{\gamma}_{i+1,x+1}$, $\hat{\gamma}_{i+1,x+2}$, …, $\hat{\gamma}_{i+1,x+m}$ are all direct successors (in the above order) of a production node $\hat{P}_{i+1}$ and the labels assigned to them by $\hat{\mathfrak{L}}_G^{\omega}$ are
$\langle \psi_{x+1}, a_{x+1}, b_{x+1}, \varphi_{x+1} \rangle$,
$\langle \psi_{x+2}, a_{x+2}, b_{x+2}, \varphi_{x+2} \rangle$,
⋮
$\langle \psi_{x+m}, a_{x+m}, b_{x+m}, \varphi_{x+m} \rangle$, respectively, then assign $\langle \psi_{x+1}, a_{x+1}, b_{x+m}, \varphi_{x+m} \rangle$ to $\hat{P}_{i+1}$, i.e. we let $\hat{\mathfrak{L}}_G^{\omega}(\hat{P}_{i+1}) = \langle \psi_{x+1}, a_{x+1}, b_{x+m}, \varphi_{x+m} \rangle$.

c) (**splitting rule**) If symbol nodes $\hat{\gamma}_{i,x+1}$, $\hat{\gamma}_{i,x+2}$, …, $\hat{\gamma}_{i,x+m}$ are all direct predecessors (in the above order) of a production node $\hat{P}_{i+1}$ that is assigned a label $\langle \psi, a, b, \varphi \rangle$ then assign the labels $\langle \psi, a, b, \langle a, 1, \hat{p}, b \rangle \rangle$,
$\langle \langle a, 1, \hat{p}, b \rangle, a, b, \langle a, 2, \hat{p}, b \rangle \rangle$,
$\langle \langle a, 2, \hat{p}, b \rangle, a, b, \langle a, 3, \hat{p}, b \rangle \rangle$,
⋮
$\langle \langle a, m-2, \hat{p}, b \rangle, a, b, \langle a, m-1, \hat{p}, b \rangle \rangle$,
$\langle \langle a, m-1, \hat{p}, b \rangle, a, b, \varphi \rangle$,
where $\hat{p} = \hat{M}(\hat{P}_{i+1})$, to those nodes, respectively.

Definition 4
Let $\hat{\gamma}_{i,j}$, $\hat{\gamma}_{i,j+1}$ be two neighbouring symbol nodes in the i-th layer of some derivation net. Let $\langle \psi_{i,j}, a_{i,j}, b_{i,j}, \varphi_{i,j} \rangle$ and $\langle \psi_{i,j+1}, a_{i,j+1}, b_{i,j+1}, \varphi_{i,j+1} \rangle$ be two labels assigned to them. The property of **neighbouring second order labels equality** holds, if $\varphi_{i,j} = \psi_{i,j+1}$. ●

Lemma 2
For each two neighbouring labels, in every layer, the neighbouring second order labels equality holds. Moreover, the left second order label in the first node in each layer is the same through all the layers, so do the right second order labels of the last nodes.

Proof
First, observe that the lemma is true for the last layer – recall (36). Next, assume, that the above property holds for the $(i+1)$-th layer. Let $\hat{\gamma}_{i,1}$, $\hat{\gamma}_{i,2}$, …, $\hat{\gamma}_{i,x}$ be direct predecessors of $\hat{\gamma}_{i+1,1}$, $\hat{\gamma}_{i+1,2}$, …, $\hat{\gamma}_{i+1,x}$. It is easily seen that the very first second order labels put on the left are the same. Since $\hat{\mathfrak{L}}_G^{\omega}(\hat{\gamma}_{i,l}) = \hat{\mathfrak{L}}_G^{\omega}(\hat{\gamma}_{i+1,l})$, for $l = 1, 2, \dots, x$, the property of neighbouring second order labels equality still holds within those first x nodes in the k-th layer. Let now $\hat{\gamma}_{i+1,x+1}$, $\hat{\gamma}_{i+1,x+2}$, …, $\hat{\gamma}_{i+1,x+q}$ be all the direct successors (in that order) of $\hat{P}_{i+1}$. According to the joining rule for the derivation nets the left second order label of $\hat{P}_{i+1}$ is the same as the left second order label of $\hat{\gamma}_{i+1,x+1}$ and the right second order label of $\hat{P}_{i+1}$ is the same as the right second order label of $\hat{\gamma}_{i+1,x+q}$. Let $\hat{\gamma}_{i,x+1}$, $\hat{\gamma}_{i,x+2}$, …, $\hat{\gamma}_{i,x+p}$ be directs predecessors of $\hat{P}_{i+1}$. Now, according to

the splitting rule for the derivation nets, the left second order label of $\hat{\gamma}_{i,x+1}$ is the same as the left second order label of $\hat{P}_{i+1}$ and hence the same as the left one of $\hat{\gamma}_{i+1,x+1}$. Since the neighbouring second order labels equality holds by inductive hypothesis for the pair $\hat{\gamma}_{i+1,x}$ and $\hat{\gamma}_{i+1,x+1}$ we deduce that the same property holds for the pair $\hat{\gamma}_{i,x}$ and $\hat{\gamma}_{i,x+1}$. The splitting rule also imposes the neighbouring second order labels equality on each neighbouring pair within $\hat{\gamma}_{i,x+1}$, $\hat{\gamma}_{i,x+2}$, …, $\hat{\gamma}_{i,x+p}$. The right second order label of $\hat{\gamma}_{i,x+p}$ is the same as that of $\hat{P}_{i+1}$ and hence – that of $\hat{\gamma}_{i+1,x+q}$. By inductive hypothesis, the neighbouring second order labels equality holds for the pair $\hat{\gamma}_{i+1,x+q}$ and $\hat{\gamma}_{i+1,x+q+1}$, from which it can be deduced that this property also holds for the pair $\hat{\gamma}_{i,x+p}$ and $\hat{\gamma}_{i,x+p}$. Finally, since $\hat{\gamma}_{i,x+p+1}$, $\hat{\gamma}_{i,x+p+2}$, …, $\hat{\gamma}_{i,J_i}$ are all the direct predecessors of $\hat{\gamma}_{i+1,x+q+1}$, $\hat{\gamma}_{i+1,x+q+2}$, …, $\hat{\gamma}_{i+1,J_{i+1}}$, respectively, and they inherit the labels according to the rewriting rule, the property holds also for each neighbouring pair within $\hat{\gamma}_{i,x+p+1}$, $\hat{\gamma}_{i,x+p+2}$, …, $\hat{\gamma}_{i,J_i}$. Thus, by induction, it can be deduced that the property of neighbouring second order labels equality holds within each layer of the derivation net. The very last second order labels in each layer are the same because they are rewritten.

In a special case where $x = 0$ it is easily seen that the left second order labels of the first nodes in the two layers are the same indeed. Symmetrically, it is also easily seen for the case where $\hat{P}_{i+1}$ is incident with both last nodes in those layers, i.e. for $x = J_{i+1} - q + 1 = = J_i - p + 1$. ■

Example 5

Let the grammar G be as in the example 4. Figure 13 shows the layers of the derivation graph for sentence yxx. The nodes in each layer are labelled with subsequent generations of labels, some of which are just copied from the previously labelled layer (rewriting rule). Figure 14, on the other hand, shows graph G', a subgraph of G, that was obtained after the contraction of the derivation graph. Its labelling is only a part of how can G be labelled. ▲

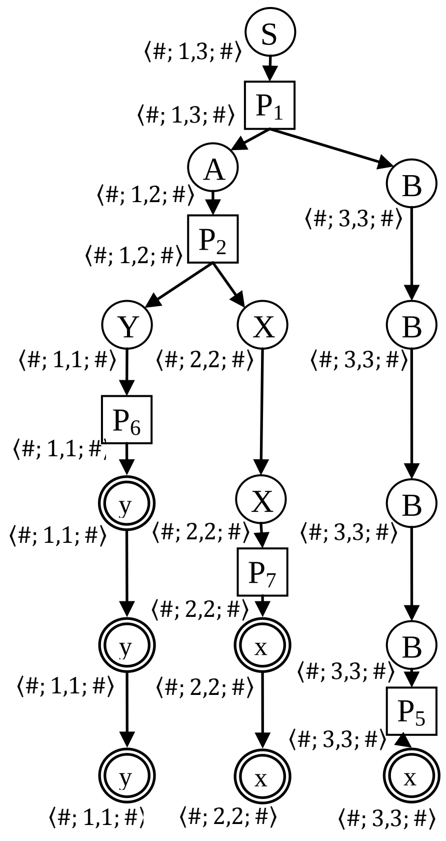

Fig. 13. An example of labels on a derivation graph

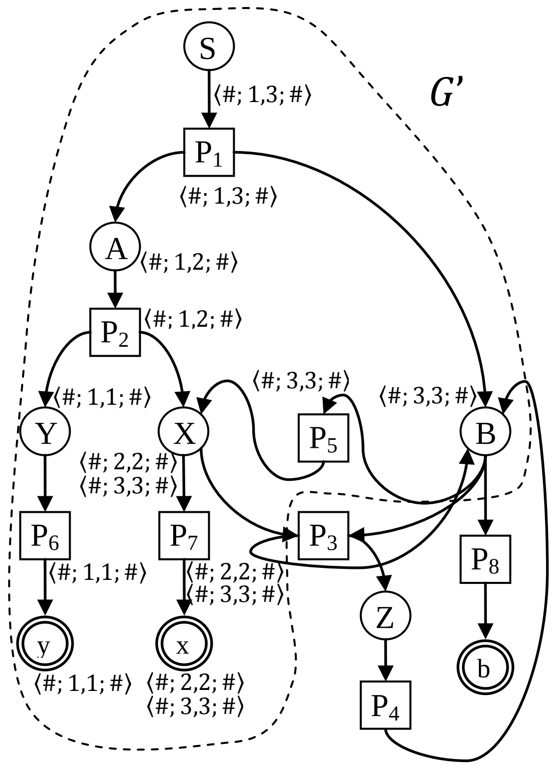

Fig. 14. An example of derivation graph contraction

Lemma 3

If the label assigned to some node in $D_\omega(G)$, according to the labelling closure routine for derivation graphs, is of the form $\langle\psi, a, b, \varphi\rangle$, then a is the index of the leftmost and b is the index of the rightmost node in the last layer that are accessible via directed paths from this labelled node. Moreover, for a symbol node $\hat{\gamma}_{i,j}$ with assigned label $\langle\psi_j, a_j, b_j, \varphi_j\rangle$ and a symbol node $\hat{\gamma}_{i,l}$ with assigned label $\langle\psi_l, a_l, b_l, \varphi_l\rangle$, where $j < l$, we have $a_j \leq a_l$ and $b_j \leq b_l$.

Proof

The property is obvious for the last layer as it results from the initial labelling structure. Now, assume that the property holds for all the nodes in the $(i+1)$-th layer. The nodes $\hat{\gamma}_{i,1}$ through $\hat{\gamma}_{i,x}$ have only one successor each, i.e. $\hat{\gamma}_{i+1,1}$ through $\hat{\gamma}_{i+1,x}$, respectively, and inherit their labels. Hence, their labels still hold the property in question. Similarly, the nodes $\hat{\gamma}_{i,x+p+1}$ through $\hat{\gamma}_{i,J_i}$ inherit the labels from their single successors $\hat{\gamma}_{i+1,x+q+1}$ through $\hat{\gamma}_{i+1,J_{i+1}}$, respectively, and the property in question holds also within this group. Now, the production node $\hat{P}_{i+1}$ between the two layers also receives a label of the form $\langle\psi^{(P)}_{i+1}, a^{(P)}_{i+1}, b^{(P)}_{i+1}, \varphi^{(P)}_{i+1}\rangle$. It results from the joining rule for the derivation nets that

$$a^{(P)}_{i+1} = \min\{a_{i+1,x+1}, a_{i+1,x+2}, \dots, a_{i+1,x+q}\}$$
$$b^{(P)}_{i+1} = \max\{b_{i+1,x+1}, b_{i+1,x+2}, \dots, b_{i+1,x+q}\} \tag{39}$$

where $a_{i+1,x+u}$, $b_{i+1,x+u}$, are determined by the labels $\langle\psi_{i+1,x+u}$, $a_{i+1,x+u}$, $b_{i+1,x+u}$, *φi+1,x+u* assigned to *γi+1,x+u*, for $u = 1,2,\dots,q$. Indeed, by the inductive hypothesis, $a^{(P)}_{i+1} = a_{i+1,x+1} \leq a_{i+1,x+2} \leq \cdots \leq a_{i+1,x+q}$ and $b_{i+1,x+1} \leq b_{i+1,x+2} \leq \cdots \leq b_{i+1,x+q} = b^{(P)}_{i+1}$. Since all the directed paths that start in $\hat{P}_{i+1}$ and end up in the last layer go through one of $\hat{\gamma}_{i+1,x+1}$, $\hat{\gamma}_{i+1,x+2}$, …, $\hat{\gamma}_{i+1,x+q}$, the property in question holds for $\hat{P}_{i+1}$ and its label. Finally, all the direct predecessors of $\hat{P}_{i+1}$, say $\hat{\gamma}_{i,x+1}$, $\hat{\gamma}_{i,x+2}$, …, $\hat{\gamma}_{i,x+p}$, are assigned labels $\langle\psi_{i,x+v}, a^{(P)}_{i+1}, b^{(P)}_{i+1}, \varphi_{i,x+v}\rangle$, $v = 1, 2, \dots, p$. Since the pair of values $a^{(P)}_{i+1}$ and $b^{(P)}_{i+1}$ are inherited by them from $\hat{P}_{i+1}$ and the only paths starting from them and ending up in the last layer go through $\hat{P}_{i+1}$, the property holds for them as well. Thus, by induction, all the layers of nodes have the property stated in the lemma. ■

Corollary 2

$\langle\#; 1, |\omega|; \#\rangle$ is the root label in a derivation net $\langle D_\omega(G), \hat{\mathfrak{L}}^\omega_G\rangle$, i.e. assigned to the “starting” node S, if the net is initially labelled as in (38). This is a direct result from the last two lemmas. The left and right second order labels are inherited from the leftmost and the rightmost second order labels in the last layer, while 1 and $|\omega|$ are, of course, the indices of the first and the last symbol in $|\omega|$ and hence the indices of the first and the last node in the last layer.

Lemma 4

Let ω be any correct sentence in G and $D_\omega(G)$ with $\hat{\mathfrak{L}}^\omega_G$ be its derivation net, where the labelling $\hat{\mathfrak{L}}^\omega_G$ was generated from the initial labelling using rewriting, joining and splitting rules for derivation nets. If $\hat{\mathfrak{L}}^\omega_G(v) = \langle\psi, a, b, \varphi\rangle$ for some node v, then $\langle\psi, a, b, \varphi\rangle \in \overline{\mathfrak{L}^\omega_G}\left(\hat{M}(v)\right)$. In other words, $\mathfrak{L}_{\tilde{G}_{\omega,\hat{M}}} \preccurlyeq \overline{\mathfrak{L}^\omega_G}$, where $\tilde{G}_{\omega,\hat{M}}$ with $\mathfrak{L}_{\tilde{G}_{\omega,\hat{M}}}$ is the quotient derivation net of $D_\omega(G)$ with $\hat{\mathfrak{L}}^\omega_G$ over $\hat{M}$.

Proof

First, observe that for the last (k-th) layer $\hat{\mathfrak{L}}^\omega_G(\hat{\gamma}_{k,j}) = \langle\#; j, j; \#\rangle$ and this belongs to $\mathfrak{L}^\omega_G\left(\hat{M}(\hat{\gamma}_{k,j})\right)$, for $j = 1,2,\dots,J_k$ and $J_k = |\omega|$. Of course, $\overline{\mathfrak{L}^\omega_G} \succcurlyeq \mathfrak{L}^\omega_G$, so $\langle\#; j, j; \#\rangle \in$ $\in \overline{\mathfrak{L}^\omega_G}\left(\hat{M}(\hat{\gamma}_{k,j})\right)$ as well. Assume now that $\hat{\mathfrak{L}}^\omega_G(\hat{\gamma}_{i,j}) \in \overline{\mathfrak{L}^\omega_G}\left(\hat{M}(\hat{\gamma}_{i,j})\right)$, for all $j = 1,2,\dots,J_i$, where i is an index of some layer in $D_\omega(G)$. Let $\hat{\gamma}_{i,x_i+1}, \hat{\gamma}_{i,x_i+2}, \dots, \hat{\gamma}_{i,x_i+q_i}$ be all the direct successors (in that order) of $\hat{P}_i$ and
$\langle\psi_{i,x_i+1}, a_{i,x_i+1}, b_{i,x_i+1}, \varphi_{i,x_i+1}\rangle$,
$\langle\psi_{i,x_i+2}, a_{i,x_i+2}, b_{i,x_i+1}, \varphi_{i,x_i+2}\rangle$,
…
$\langle\psi_{i,x_i+q_i}, a_{i,x_i+q_i}, b_{i,x_i+q_i}, \varphi_{i,x_i+q_i}\rangle$ be their labels, respectively. Recall that the above labels hold the neighbouring second order label equality property, i.e. $\varphi_{i,x_i+1} = \psi_{i,x_i+2}$, $\varphi_{i,x_i+2} = \psi_{i,x_i+3}$, … , $\varphi_{i,x_i+q_i-1} = \psi_{i,x_i+q_i}$. The joining rule for the derivation nets assigns $\langle\psi_{i,x_i+1}$, a_{i,x_i+1}, b_{i,x_i+q_i}, $\varphi_{i,x_i+q_i}\rangle$ to $\hat{P}_i$. On the other hand, $\langle\psi_{i,x_i+j}$, a_{i,x_i+j}, b_{i,x_i+j}, $\varphi_{i,x_i+j}\rangle \in \overline{\mathfrak{L}^\omega_G}\left(\hat{M}(\hat{\gamma}_{i,x_i+j})\right)$, for $j = 1,2,\dots,J_i$. Since $\hat{M}(\hat{\gamma}_{i,x_i+1})$, $\hat{M}(\hat{\gamma}_{i,x_i+2})$, …, $\hat{M}(\hat{\gamma}_{i,x_i+q_i})$ are direct successors (in that order) of $\hat{M}(\hat{P}_i)$ in $\tilde{G}$, the joining rule for grammar nets produces a new label (of course, unless it has already existed) for $\hat{M}(\hat{P}_i)$. This is $\langle\psi_{i,x_i+1}$, a_{i,x_i+1},

$b_{i,x_i+q_i}, \varphi_{i,x_i+q_i}\rangle$ and is equal to $\widehat{\mathfrak{L}}_G^{\omega}(\hat{P}_i)$. Hence, $\widehat{\mathfrak{L}}_G^{\omega}(\hat{P}_i) \in \overline{\mathfrak{L}_G^{\omega}}\left(\widehat{M}(\hat{P}_i)\right)$ holds. Finally, let $\hat{\gamma}_{i-1,x_{i-1}+1}$, $\hat{\gamma}_{i-1,x_{i-1}+2}$, $\cdots$, $\hat{\gamma}_{i-1,x_{i-1}+p_{i-1}}$ be all the direct predecessors (in that order) of $\hat{P}_i$. According to the splitting rule for derivation nets those nodes are assigned labels $\langle\psi_{i-1,x_{i-1}+1}, a_{i,x_i+1}, b_{i,x_i+q_i}, \varphi_{i-1,x_{i-1}+1}\rangle$, $\langle\psi_{i-1,x_{i-1}+2}, a_{i,x_i+1}, b_{i,x_i+q_i}, \varphi_{i-1,x_{i-1}+2}\rangle$, $\dots$ $\langle\psi_{i-1,x_{i-1}+p_{i-1}}, a_{i,x_i+1}, b_{i,x_i+q_i}, \varphi_{i-1,x_{i-1}+p_{i-1}}\rangle$, respectively. On the other hand, the splitting rule for the grammar nets adds exactly the same labels to the predecessors of $\widehat{M}(\hat{P}_i)$ in $\tilde{G}$, preserving, of course, their respective order. All the other nodes in the $(i-1)$-th layer of $D_\omega(G)$ inherit their labels from respective nodes in the i-th layer and maps with $\widehat{M}$ to the same node in $\tilde{G}$ as the node they inherited their label from. These labels, though, have already been taken into account in $\overline{\mathfrak{L}_G^{\omega}}$. By induction, it is easily deduced that $\widehat{\mathfrak{L}}_G^{\omega}(v) \in \overline{\mathfrak{L}_G^{\omega}}\left(\widehat{M}(v)\right)$ for every node v in $D_\omega(G)$. ■

Theorem 1
Let ω be a correct sentence in the language generated by a grammar G, i.e. $\omega \in L(G)$, then $\langle\#; 1, |\omega|; \#\rangle \in \overline{\mathfrak{L}_G^{\omega}}(S)$.

Proof
This directly results from the previously proved lemmas. $\widehat{\mathfrak{L}}_G^{\omega}(\hat{\gamma}_{1,1}) = \langle\#; 1, |\omega|; \#\rangle$ and $\widehat{\mathfrak{L}}_G^{\omega}(\hat{\gamma}_{1,1}) \in \overline{\mathfrak{L}_G^{\omega}}\left(\widehat{M}(\hat{\gamma}_{1,1})\right)$, where $\widehat{M}(\hat{\gamma}_{1,1}) = S$. ■

The above theorem gives a tool to check if a particular sentence can potentially be a correct one in G, i.e. having any correct derivation starting from S. If $\langle\#; 1, |\omega|; \#\rangle$ is not found for S, then the sentence ω obviously does not belong to $L(G)$. On the other hand, if $\langle\#; 1, |\omega|; \#\rangle$ is found for S indeed, then further procedure must be undertaken to check its correctness.

6. Algorithm

The algorithm is based on the labelling closure routine for the initial labelling of a given grammar graph. The following quasi-language describes its version for a given 2nd order non--contracting grammar G:

```
input txt: String;
input grammar G;
type Label2nd: (left: Int, prod: Production,
                right: Int);
type Label: (leftlbl: Label2nd, left: Int,
             right: Int, rightlbl: Label2nd);
let lst: List of (lbl: Label, nod: Node);
let crs: Int = 1;
let prods = set of all production nodes;
let symbs = set of all symbol nodes;
let lbl: Label;
for each i in 1..|ω|
  for each s in symb
    if txt(i) = s then
      append (((i-1, nil, i), i, i,
              (i, nil, i+1)), s)
          to lst;
    end if;
  end for;
end for;
while crs <= length of lst
  let n := lst(crs).nod;
  if n is symbol node then
    for each p in prods
      if p has two children then
        for each c in 1..crs-1
          let s := lst(c).nod;
          let newlbl: Label = nil;
          if n is left child of p and
            s is right child of p then
            if n.rightlbl = s.leftlbl then
              let lbl := (n.leftlbl, n.left,
                          s.right, s.rightlbl);
              if (lbl, p) not in lst then
                append (lbl, p) to lst;
              end if;
            end if;
          elsif n is right child of p and
                s is left child of p then
            if n.leftlbl = s.rightlbl then
              let lbl := (s.leftlbl, s.left,
                          n.right, n.rightlbl);
              if (lbl, p) not in lst then
                append (lbl, p) to lst;
              end if;
            end if;
          end if;
        end for;
      elsif p has one child then
        if n is child of p then
          let lbl := (n.leftlbl, n.left,
                      n.right, n.rightlbl);
          if (lbl, p) not in lst then
            append t (lbl, p) to lst;
          end if;
        end if;
      end if;
    end for;
  elsif n is production node then
    for each s in symbs
      if s is left parent of n then
        let lbl := (n.leftlbl, n.left, n.right,
                    (n.left, n, n.right));
        if (lbl, s) not in lst then
          append (lbl, s) into lst;
        end if;
      elsif s is right parent of n then
        let lbl := ((n.left, n, n.right),
                    n.left, n.right, n.rightlbl);
        if (lbl, s) not in lst then
          append (lbl, s) into lst;
        end if;
      elsif s is the only parent of n then
        let lbl := (n.leftlbl, n.left,
                    n.right, n.rightlbl);
        if (lbl, s) not in lst then
          append (lbl, s) into lst;
        end if;
      end if;
    end for;
  end if;
  crs := crs + 1;
end while;
```

The main data structure in the above algorithm is a list of pairs consisting of a label and node reference (whether it is production node, symbol node or empty reference). A label $\langle\langle a,1,\zeta,b\rangle,c,d,\langle e,1,\xi,f\rangle\rangle$ is represented by ((a,ζ,b),c,d,(e,ξ,f)) in this data structure. If ζ or ξ is #, then it is represented by the empty reference (**nil**). The input string `txt` is first split into single characters corresponding to terminal symbol nodes. All the pairs of labels with **nil** in the second order labels (equivalent to #) and unit range corresponding to a particular character combined with the corresponding terminal symbol node references are put to list. Thus, the initial labelling is represented there.

Next, the algorithm finds new labels based on the previously found ones. This is done according to the labelling closure rules. Newly found labels are added at the end of the list. In parallel, a cursor `crs` (marker) is moved step by step through the list to mark the last element in the list for which there have been generated new labels based on all the already existing in the list. That is to say, for a label currently pointed by the marker, all the previous ones in the list are tested if they “match” to produce new ones. The new ones, are appended to the list if they have not existed yet there. This is similar to the breadth-first search algorithm.

The number of possible labels is finite. The number M^{II} of possible second order labels is limited with the following formula:

$$M^{II} \leq |\omega|^2(1+|P|). \tag{40}$$

The number M^{II} of possible first order labels is limited with the next formula:

$$M^{I} \leq |\omega|^2(M^{II})^2 \tag{41}$$

Thus the number of labels is not greater than $|\omega|^2\big(|\omega|^2(1+|P|)\big)^2$. Since, the number of production nodes in the grammar graph is finite, the number of possible pairs consisting of a label and a node is also finite. This number will never exceed $|\omega|^2\big(|\omega|^2(1+|P|)\big)^2(|N|+|\Sigma|+|P|)$. This implies that the algorithm will always stop at some point as it never produces any pair of label and node twice.

As soon as the algorithm stops, one can check if there exists the special label ((0,**nil,1)**,1,|ω|,(|ω|,**nil,**|ω|+1)) in the list. If it does indeed, then the given string `txt` might be a correct sentence in the given grammar G. If it does not, then the sentence is obviously incorrect.

To find more accurate limit for the number of possibly generated labels, note that second order labels will always have form $\langle a,1,\theta_i,b'\rangle$, if put on the left of $\langle a,b\rangle$, $a \leq b' \leq b$, and $\langle a',1,\theta_j,b\rangle$, if put on the right of $\langle a,b\rangle$, $a \leq a' \leq b$; θ_i and θ_j – some productions or the special mark # (represented by the empty reference **nil**). Thus, the overall form of a first order label with two second order labels is

$$\langle\langle a,1,\theta_i,b'\rangle,a,b,\langle a',1,\theta_j,b\rangle\rangle. \tag{42}$$

There are four numbers that are linearly bounded by $|\omega|$. Thus, the algorithm time and space is not worse than $O(n^4 \cdot m^2)$, where $n = |\omega|$ and $m = |P| + 1$.

Example 6

Let the grammar G consist of the following production rules: $P_1\colon S \to ZC$, $P_2\colon Z \to AB$, $P_3\colon B \to XY$, $P_4\colon X \to HB$, $P_5\colon Y \to BG$, $P_6\colon GB \to BG$, $P_7\colon GC \to CC$, $P_8\colon BH \to HB$, $P_9\colon AH \to AA$, $P_{10}\colon A \to a$, $P_{11}\colon B \to b$ and $P_{12}\colon C \to c$. G contains the sentences of the form $a^n b^n c^n$, $n \geq 1$. Let us trace the algorithm for $\omega = abc$.

$|\omega| = 3$. The following table shows the content of the main list filled with initial labelling data and the values of the cursor after the each position:

Pos.	Newly added label	Target node	Cursor pos.
1.	((0,**nil**,1),1,1,(1,**nil**,2))	a	1
2.	((1,**nil**,2),2,2,(2,**nil**,3))	b	1
3.	((2,**nil**,3),3,3,(3,**nil**,4))	c	1

After the 3rd step the list contains 3 initial labels, paired with their corresponding nodes, and the cursor points to the first pair. Stopping condition is not met and the label ((0,**nil**,1),1,1,(1,**nil**,2)) in the first position will be used to produce new ones, potentially matching it with any other label existing in the list. The target node a is a symbol, thus a subsequent loop iterates over all the productions. For each production with two children all the previous elements of the list are compared if the two refer to 2 target nodes being children of this production with matching second order labels. Of course, there are no previous elements in the list as the cursor is at the first one. For one-child production it may, however, produce new pair of label and node. The node a is the only child of P_{10}. The newly produced label ((0,**nil**,1),1,1,(1,**nil**,2)) for P_{10}, which in fact is the same as for a, is added to the list as this has not been added to the list yet. There are no more labels produced in the step and the cursor moves to the next position, i.e. the second one. So, the list is given new entry:

4.	((0,**nil**,1),1,1,(1,**nil**,2))	P_{10}	2

The same routine for the second and third position in the list gives the following:

5.	((1,**nil**,2),2,2,(2,**nil**,3))	P_{11}	3
6.	((2,**nil**,3),3,3,(3,**nil**,4))	P_{12}	4

Note that the cursor moved to beyond the initial labelling entries in the list. The stop condition is not met as the list was enlarged in the meantime with newly added labels. The algorithm continues.

In the fourth step, the algorithm finds the target node P_{10} to be a production. Since its only parent is the symbol node A, new label ((0,**nil**,1),1,1,(1,**nil**,2)) is proposed for this node. It is appended indeed to the list, paired with A, as it has not existed yet there. Similarly, ((1,**nil**,2),2,2,(2,**nil**,3)) paired with B and ((2,**nil**,3),3,3,(3,**nil**,4)) paired with C are added to the list:

7.	((0,**nil**,1),1,1,(1,**nil**,2))	A	5
8.	((1,**nil**,2),2,2,(2,**nil**,3))	B	6
9.	((2,**nil**,3),3,3,(3,**nil**,4))	C	7

Now, with the cursor pointing to the 7th position, that corresponds to a label assigned to A, the algorithm iterates over all productions and finds that A is the left child of P_2 and both the left and the right child of P_9. Nevertheless, it does not match to any previously generated label, hence the cursor moves on to the next position without producing any new label:

-	-	-	8

The symbol node B is the right child of P_2, the left child of P_5 and the left child of P_6. For P_2, when iterating over the previous positions, the algorithm finds that the label ((0,**nil**,1),1,1,(1,**nil**,2)) assigned to A in the 7th position matches with ((1,**nil**,2),2,2,(2,**nil**,3)) in the current position and produces new label ((0,**nil**,1),1,2,(2,**nil**,3)) for the node P_2. This is the only matching for the 8th position, hence:

10.	((0,**nil**,1),1,2,(2,**nil**,3))	P_2	9

The label ((2,**nil**,3),3,3,(3,**nil**,4)) assigned to C does not match to any previous label assigned to the children of P_1 and P_7, the only parents of C. The cursor moves to the next position:

-	-	-	10

Similarly, the next steps of the algorithm produces:

11.	((0,**nil**,1),1,2,(2,**nil**,3))	Z	11
12.	((0,**nil**,1),1,3,(3,**nil**,4))	P_1	12
13.	((0,**nil**,1),1,3,(3,**nil**,4))	S	13
-	-	-	14

and the algorithm stops finding the special label ((0,**nil**,1),1,3,(3,**nil**,4)) for the initial symbol S, stating that $\omega = abc$ can potentially be a correct sentence in G. ▲

Example 7
Let the grammar G be as in the example 6. Let also $\omega = abcc$. The following table shows all the steps of the algorithm:

Pos.	Newly added label	Target node	Cursor pos.
1.	((0,**nil**,1),1,1,(1,**nil**,2))	a	1
2.	((1,**nil**,2),2,2,(2,**nil**,3))	b	1
3.	((2,**nil**,3),3,3,(3,**nil**,4))	c	1
4.	((3,**nil**,4),4,4,(4,**nil**,5))	c	1
5.	((0,**nil**,1),1,1,(1,**nil**,2))	P_{10}	2
6.	((1,**nil**,2),2,2,(2,**nil**,3))	P_{11}	3
7.	((2,**nil**,3),3,3,(3,**nil**,4))	P_{12}	4
8.	((3,**nil**,4),4,4,(4,**nil**,5))	P_{12}	5
9.	((0,**nil**,1),1,1,(1,**nil**,2))	A	6
10.	((1,**nil**,2),2,2,(2,**nil**,3))	B	7
11.	((2,**nil**,3),3,3,(3,**nil**,4))	C	8
12.	((3,**nil**,4),4,4,(4,**nil**,5))	C	9
-	-	-	10
13.	((0,**nil**,1),1,2,(2,**nil**,3))	P_2	11
-	-	-	12
14.	((2,**nil**,3),3,4,(4,**nil**,5))	P_7	13
15.	((0,**nil**,1),1,2,(2,**nil**,3))	Z	14
16.	((2,**nil**,3),3,4,(3, P_7,4))	G	-
17.	((3,P_7,4),3,4, (4,**nil**,5))	C	15
18.	((0,**nil**,1),1,3,(3,**nil**,4))	P_1	16
-	-	-	17
-	-	-	18
19.	((0,**nil**,1),1,3,(3,**nil**,4))	S	19
-	-	-	20

In the above routine the algorithm did not find the special label ((0,**nil**,1),1,4,(4,**nil**,5)) for the initial symbol S. This implies that $abcc$ is not a correct sentence in the given grammar. ▲

7. Conclusion

The idea of the algorithm presented in this paper lays behind the labelling closure generated by the initial labelling of the terminal nodes. A necessary condition for a particular sentence to belong to some non-contracting grammar was shown. Some properties related to new concepts were described.

Further research on the grammar and derivation nets might be conducted to improve their “selectiveness”, i.e. to show a more restricting necessary condition. Such a condition should reject more incorrect sentences than the algorithm presented here. One of the possible ways to improve the algorithm would be to discover potential derivations, based on how successive labels have been obtained, and check their correctness.

8. Bibliography

[1] Aho A.V., Lam M.S., Sethi R., Ullman J.D., *Compilers: Principles, Techniques and Tools*, 2nd Edition, Pearson Education, 2006.
[2] Chomsky N., “On Certain Formal Properties of Grammars”, *Information and Control*, Vol. 2, 137–167 (1959).
[3] Foryś M., Foryś W., *Teoria automatów i języków formalnych*, EXIT, 2005.
[4] Hopcroft J.E., Motwani R., Ullman J.D., *Introduction to Automata Theory, Languages and Computation*, 2nd Edition, Pearson Education, 2001.
[5] Kuroda S.-Y., “Classes of Languages and Linear-Bounded Automata”, *Information and Control*, Vol. 7, 207–223 (1964).
[6] Linz P., *An Introduction to Formal Languages and Automata*, 5th Edition, Jones & Bartlett Learning, 2012.
[7] Martin J.C., *Introduction to Languages and the Theory of Computation*, 4th Edition, McGraw-Hill, 2011.
[8] Szwoch M., *Języki formalne, automaty i translatory*, PWNT, 2008.
[9] Waite W.M., Goos G., *Compiler Construction*, Springer-Verlag, 1984.
[10] Younger D.H., “Recognition and Parsing of Context-Free Grammars in Time n^3”, *Information and Control*, Vol. 10, Issue 2, 189–208 (1967).

O problemie przynależności zdań do języków kontekstowych

P.A. RYSZAWA

W artykule wprowadzony został nowy rodzaj grafu – graf gramatyczny. Możliwość przypisywania etykiet do węzłów daje rozszerzenie do tzw. sieci gramatycznej. Sieć gramatyczną należy traktować jako nowe narzędzie graficzne w analizie przynależności zdań do danego języka kontekstowego. Inna koncepcja, sieć wywodu, ściśle związana z grafem gramatycznym i o podobnej strukturze, została wykorzystana do pokazania algorytmu, który potrafi wstępnie wyselekcjonować niektóre zdania nienależące do danego języka generowanego przez gramatykę kontekstową, pozostawiając inne jako potencjalnie w nim zawarte.

Słowa kluczowe: gramatyka kontekstowa, gramatyka nieskracająca, język formalny, graf, rozbiór zdań.