ELSEVIER

Theoretical Computer Science 223 (1999) 143–178

Theoretical Computer Science

www.elsevier.com/locate/tcs

HRNCE grammars – a hypergraph generating system with an eNCE way of rewriting [1]

Changwook Kim *, Tae Eui Jeong

School of Computer Science, University of Oklahoma, 200 Felgar street, Room 114, Norman, Oklahoma 73019, USA

Received September 1995; revised September 1997
Communicated by A. Salomaa

Abstract

We introduce a hypergraph-generating system, called HRNCE grammars, which is structurally simple and descriptively powerful. An HRNCE grammar replaces a handle (i.e., a hyperedge together with its incident nodes) by a hypergraph, whose nodes are connected to (or contained in) the hyperedges surrounding the replaced handle by using the eNCE rewriting mechanism. HRNCE grammars can generate all recursively enumerable languages. HRNCE grammars without edge erasing can generate PSPACE-complete languages. Separated HRNCE (S-HRNCE) grammars have many useful normal forms, including Chomsky, Greibach, and context-free normal forms. S-HRNCE languages are in NP and there is an NP-complete linear HRNCE (Lin-HRNCE) language satisfying any two of the following conditions: connected; degree-bounded; and rank-bounded. On the other hand, S-HRNCE (Lin-HRNCE) languages satisfying all these three conditions are in LOGCFL (NLOG) and there is such a language which is LOGCFL-complete (NLOG-complete). The equivalence problem is undecidable for Lin-HRNCE languages.

Keywords: Formal languages; Graph grammars; Normal forms; Complexity

1. Introduction

Graph grammars extend the traditional string grammars by replacing the left- and right-hand sides of a production by graphs and thus generate sets of graphs. They were originally introduced to describe picture patterns but soon found applications in many other subareas of computer science such as computer aided design, programming languages and compilers, databases, software specifications, concurrency, pattern

* Corresponding author. E-mail: kim@cs.ou.edu.

[1] A preliminary version of this paper was presented at the 5th International Workshop on Graph Grammars and their Application to Computer Science (Williamsburg, VA, USA, November 1994) and appeared in Lecture Notes in Computer Science, Vol. 1073 (Springer, Berlin, 1996), pp. 383–396.

0304-3975/99/$ - see front matter
PII: S0304-3975(99)00258-2

recognition and computer vision. Various applications and related research efforts of graph grammars can be found in the proceedings of the international workshops on "Graph Grammars and Their Application to Computer Science" [3, 4, 7–9].

It is important that any graph-grammar model be structurally simple, i.e., easy to use or understand, yet descriptively powerful, i.e., able to describe many interesting graph-theoretical properties. It is also desirable that such a system be flexible so that subclasses with nice features, such as tractable membership complexity or decidability, can be easily defined.

One of the most successful graph-grammar models in this sense is the node-label-controlled (NLC) grammars of Janssens and Rozenberg [23–25], in which a single node is replaced by a graph in a derivation step and embedding of the newly introduced graph into the existing graph is based on node labels only. They can describe PSPACE-complete languages. Many subclasses of NLC grammars with improved properties have appeared in the literature, mostly obtained by imposing certain structural restrictions or context-free conditions on NLC grammars. Examples are the boundary NLC (B-NLC) grammars of Rozenberg and Welzl [37–39] in which no two nonterminal nodes are allowed to be adjacent in any sentential form, the neighborhood-uniform NLC (NU-NLC) grammars of Janssens and Rozenberg [27] in which each node in the right-hand side of a production is connected either to all neighbors of the replaced node or to none, and the apex NLC (A-NLC) grammars of Engelfriet et al. [13] in which the embedding mechanism can establish edges between terminal nodes only. Extensions of NLC grammars have also been studied. Examples include the NCE grammars (NLC with neighborhood-controlled embedding) of Janssens and Rozenberg [26] in which the embedding mechanism makes use of the identity (rather than the label) of the nodes in the right-hand sides of productions, the eNCE grammars of Engelfriet et al. [11, 15] which extend the NCE grammars by adding edge labels, and the edNCE grammars of Engelfriet and Rozenberg [16] which extend the eNCE grammars by adding edge directions. Restrictions defined for NLC grammars can be easily defined for these extensions of NLC grammars. The eNCE grammars and their restrictions have been particularly successful since they are as simple as NLC grammars yet preserve or improve many nice features of NLC grammars.

For grammars generating hypergraphs, there are two well-known models: the context-free hypergraph (CFHG) grammars of Bauderon and Courcelle [2] and Habel et al. [19, 20], which replace a hyperedge by a hypergraph through their preidentified gluing points, and an extension of CFHG grammars called the handle-rewriting hypergraph (HH) grammars of Courcelle et al. [6], which replace a handle (i.e., a hyperedge together with its incident nodes) by a hypergraph through an extension of the CFHG rewriting mechanism that can also duplicate or delete the hyperedges surrounding the replaced handle. Both CFHG and HH grammars generate directed hypergraphs. CFHG grammars are structurally simple but are limited in description power, i.e., generate NP languages only. HH grammars are powerful but their rewriting mechanism seems to be rather complicated, in particular because of the duplication and deletion features.

We introduce a new hypergraph-generating system, called HRNCE grammars (hypergraph grammars with an eNCE way of rewriting), generating node- and hyperedge-labeled undirected hypergraphs. An HRNCE grammar replaces a handle by a hypergraph, as in HH grammars, whose nodes are connected to (or contained in) the hyperedges surrounding the replaced handle by using the eNCE rewriting mechanism. HRNCE grammars are as easy to use as eNCE grammars yet can generate all recursively enumerable languages. (eNCE grammars generate PSPACE languages only.) They can generate degree- and rank-unbounded hypergraphs, while CFHG and HH grammars can generate rank-bounded hypergraphs only. Furthermore, their subclasses possess many nice features comparable to their eNCE counterparts. We shall present basic properties of HRNCE grammars and their restrictions, emphasizing their normal forms and description power (or complexity).

The paper is organized as follows. Section 2 contains definitions of hypergraphs and their languages. Section 3 contains the formal definition of HRNCE grammars and their two subclasses, the separated HRNCE (S-HRNCE) grammars and the linear HRNCE (Lin-HRNCE) grammars, studied much in other graph-grammar models [6, 11, 12, 14–16, 30–32, 37–39]. (For NLC, eNCE and edNCE grammars, their separated subclasses are more often called boundary grammars. For HH grammars, the separated subclass and the boundary subclass are defined differently; the former properly includes the latter.) In Section 4, we show that HRNCE grammars describe all recursively enumerable languages and that HRNCE grammars without edge-erasing property describe PSPACE-complete languages. Section 5 discusses normal forms for S-HRNCE grammars. In particular, the edge-erasing property can be removed from S-HRNCE grammars and there are Chomsky and Greibach normal forms for S-HRNCE grammars. This implies that S-HRNCE languages are in NP. We also prove that S-HRNCE grammars can be put into context-free (i.e., confluent and associative) normal form. Section 6 further discusses the complexity of S-HRNCE languages. We show that there exists an NP-complete Lin-HRNCE language which satisfies any two of the following conditions: connected; degree-bounded; and rank-bounded. S-HRNCE languages satisfying all these three conditions are in LOGCFL, the class of languages log-space reducible to context-free languages [41], and there is such an S-HRNCE language which is LOGCFL-complete. On the other hand, Lin-HRNCE languages satisfying all these three conditions are in NLOG, the nondeterministic log-space class, and there is such a Lin-HRNCE language which is NLOG-complete. A corollary of our proof technique is the undecidability of the equivalence and intersection-emptiness problems for Lin-HRNCE languages. Section 7 proves a hierarchy of HRNCE language families. Section 8 contains some concluding remarks and possible future research problems.

2. Preliminaries

The reader is assumed to be familiar with formal language theory and complexity theory in the context of, e.g., [22]. This section contains definitions related to hyper-

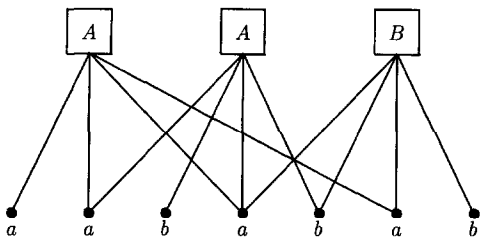


Fig. 1. A hypergraph.

graphs and their languages needed in this paper. In the sequel, the empty set is denoted by $\emptyset$ and, for a finite set A, its cardinality is denoted by $\#A$. The empty word is denoted by λ.

Let Σ, Γ be alphabets. A *hypergraph* over Σ and Γ is a system $H = (V, E, \phi, \psi, \iota)$, where V is a finite set of *nodes*, E is a finite set of *hyperedges* (or simply *edges*), $\phi : V \to \Sigma$ is a node-labeling function, $\psi : E \to \Gamma$ is an edge-labeling function, and $\iota : E \to 2^V$ is a mapping assigning a set of nodes to each edge in E. Thus, we handle undirected, node- and edge-labeled hypergraphs. Note that H is a *graph* if each edge consists of one or two nodes. For any hypergraph H, its five components are denoted by V_H, E_H, ϕ_H, ψ_H and ι_H.

A hypergraph can be pictorially described by a bipartite graph. Fig. 1 shows an example of such a description of a hypergraph with seven nodes (the dots) and three edges (the boxes) together with their labels. The membership of a node in an edge is indicated by a line connecting them.

A node v and an edge e in H are *incident* to each other if $v \in \iota_H(e)$. A node is an *isolated node* if it has no incident edge and an edge is an *empty edge* if it has no incident node. Two edges e and e' in H are *adjacent* (or *a-adjacent*) if there is a node $v \in \iota_H(e) \cap \iota_H(e')$ (with $\phi_H(v) = a$). The *degree* of a node v in H, denoted by $\deg_H(v)$, is the number of its incident edges; the degree of H, denoted by $\deg(H)$, is the maximum degree of its nodes. The *rank* of an edge e in H, denoted by $\mathrm{rank}_H(e)$, is the number of its incident nodes; the rank of H, denoted by $\mathrm{rank}(H)$, is the maximum rank of its edges.

A sequence $e_0, v_1, e_1, \ldots, v_k, e_k (k \geqslant 1)$ is a *path* (between e_0 and e_k) in H if $e_i \in E_H, 0 \leqslant i \leqslant k$, $v_j \in \iota_H(e_{j-1}) \cap \iota_H(e_j), 1 \leqslant j \leqslant k$, and $e_i \neq e_j$ and $v_i \neq v_j$ if $i \neq j$. H is a *chain* if it consists of a path and there is no other node, edge or incidence. H is *connected* if it consists of a single isolated node or a single empty edge or else it contains no isolated node or empty edge and there is a path between each pair of its edges. H is the *empty hypergraph*, denoted by Λ, if $V_H = \emptyset$ and $E_H = \emptyset$.

Two hypergraphs H and K are *isomorphic* if there are bijections α: $V_H \to V_K$ and β: $E_H \to E_K$ such that for all $v \in V_H$, $\phi_K(\alpha(v)) = \phi_H(v)$ and for all $e \in E_H$, $\psi_K(\beta(e)) = \psi_H(e)$ and $\iota_K(\beta(e)) = \{\alpha(v) \mid v \in \iota_H(e)\}$. H is a *subhypergraph* of K if $V_H \subseteq V_K$, $E_H \subseteq E_K$, $\phi_H(v) = \phi_K(v)$ for all $v \in V_H$, and $\psi_H(e) = \psi_K(e)$ and $\iota_H(e) \subseteq \iota_K(e)$ for all $e \in E_H$. H is the *handle* associated with an edge e in K, denoted by handle(e),

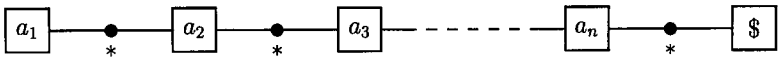


Fig. 2. An oriented chain.

if H is the subhypergraph of K consisting of e and all its incident nodes. H is the *dual* of K, denoted by dual(K), if $V_H = E_K$, $E_H = V_K$, $\phi_H = \psi_K$, $\psi_H = \phi_K$, and for all $e \in E_H, \iota_H(e) = \{e' \in E_K \mid e \in \iota_K(e')\}$. Note that dual(dual($H$)) $= H$ for every hypergraph H.

The set of all hypergraphs over Σ and Γ is denoted by $HGR_{\Sigma,\Gamma}$. A *hypergraph language* is any subset of $HGR_{\Sigma,\Gamma}$. A hypergraph language is *connected* if it contains connected hypergraphs only and is *degree-bounded* (*rank-bounded*) if the degree (rank) of each of its members is at most k, for some fixed $k \geqslant 0$.

A word $x = a_1 a_2 \cdots a_n$ can be described by a special chain called its *oriented chain*, shown in Fig. 2 and denoted by o-chain(x), where \$ is a special symbol to indicate orientation. For a word language L, o-chain(L) denotes the set $\{\text{o-chain}(x) \mid x \in L\}$.

3. HRNCE grammars

Existing (hyper)graph grammars rewrite a node (as in NLC grammars [23] and eNCE grammars [15]), an edge (as in edge-label-controlled grammars [36] and CFHG grammars [2, 19]), or a handle (as in handle NLC grammars [35] and HH grammars [6]) in a derivation step. As stated in [6], one can observe that, in general, handle-rewriting grammars are more powerful than node-rewriting grammars, which in turn are more powerful than edge-rewriting grammars. We choose to rewrite a handle in order to maximize description power. There are several ways to connect the newly introduced hypergraph into the neighborhood of the replaced handle: edges to nodes, nodes to edges, or both. One can also create new nodes or edges not in the right-hand side of a production when embedding takes place, as in HH grammars. Our choice is to simply connect nodes to edges without creating new nodes or edges. However, in order to perform such a connection selectively, we shall fully utilize all available local informations, such as node/edge labels and node identities, as in eNCE grammars. This combination is simple enough, comparable to the eNCE rewriting mechanism, yet produces power and flexibility as shall be demonstrated in the subsequent sections.

Definition 3.1. A *hypergraph grammar with an eNCE way of rewriting* (*HRNCE grammar*) is a system $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$, where

(1) Σ is an alphabet of *node labels*;

(2) Δ ($\subseteq \Sigma$) is the set of *terminal node labels* (the elements in $\Sigma - \Delta$ are *nonterminal node labels*);

(3) Γ is an alphabet of *edge labels*;

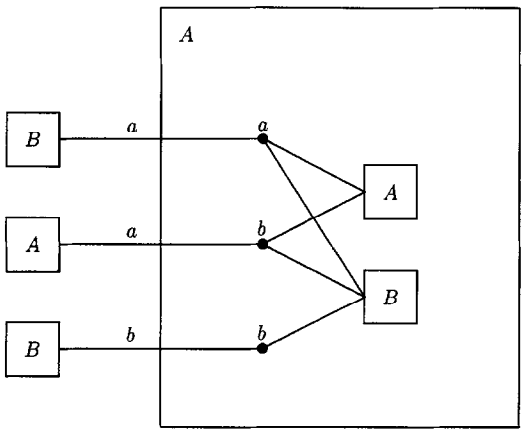


Fig. 3. A pictorial representation of a production rule.

(4) Ω ($\subseteq \Gamma$) is the set of *terminal edge labels* (the elements in $\Gamma - \Omega$ are *nonterminal edge labels*);

(5) P is a finite set of *productions*, of the form $\pi = (A, X, C)$, where $A \in \Gamma - \Omega$ (the *left-hand side*), $X \in HGR_{\Sigma,\Gamma}$ (the *right-hand side*), and $C \subseteq V_X \times \Sigma \times \Gamma$ (the *embedding relation*); and

(6) Z ($\in HGR_{\Sigma,\Gamma}$) is the *axiom hypergraph*.

We shall discuss informally how a production $\pi = (A, X, C)$ is applied to a nonterminal edge e (with $\psi_H(e) = A$) in a hypergraph $H \in HGR_{\Sigma,\Gamma}$. It is very similar to the eNCE derivation step: First, remove handle(e), i.e., e and all its incident nodes, from H. Second, add X (or an isomorphic copy of it) to the resulting hypergraph. Now, for each $v \in V_X$ and each $(v, a, B) \in C$, add v to each edge labeled by B that is a-adjacent to e in H. Such a situation can be well described pictorially as used, e.g., in [15]; we show an example in Fig. 3, where the symbol A located in the left upper corner of the big box represents the left-hand side of π, the hypergraph located inside the big box (with two edges and three nodes) represents the right-hand side of π, and the three lines crossing the big box together with the three edges located outside the big box represent the embedding relation of π, i.e., (x, a, B), (y, a, A) and (z, b, B) if x, y, z are nodes of X read top-down in Fig. 3. If any edge adjacent to e in H contains no node after applying π to e, then it is removed immediately.

Formally, the HRNCE derivation step is defined as follows:

Definition 3.2. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be an HRNCE grammar. Let $H \in HGR_{\Sigma,\Gamma}$, $e \in E_H$, $\psi_H(e) = A \in \Gamma - \Omega, \pi = (A, X, C) \in P$, and let α, β be bijections on V_X and E_X, respectively, such that $V_H \cap \alpha(V_X) = \emptyset$ and $E_H \cap \beta(E_X) = \emptyset$. For each edge $e' \in E_H - \{e\}$, let $\gamma(e') = (\iota_H(e') - \iota_H(e)) \cup \{\alpha(v) \mid (v, a, \psi_H(e')) \in C$ and e, e' are a-adjacent in $H\}$. Then, H *directly derives* a hypergraph $K \in HGR_{\Sigma,\Gamma}$ in G, denoted

by $H \Rightarrow_{(e,\pi,\alpha,\beta)} K$ (or simply $H \Rightarrow_{(e,\pi)} K$ or $H \Rightarrow K$), if K is defined as follows:

$$V_K = (V_H - \iota_H(e)) \cup \alpha(V_X);$$

$$E_K = \{e' \in E_H - \{e\} \mid \iota_H(e) \cap \iota_H(e') \neq \emptyset \text{ implies } \gamma(e') \neq \emptyset\} \cup \beta(E_X);$$

$$\phi_K(x) = \phi_H(x) \text{ if } x \in V_H \text{ and } \phi_K(\alpha(x)) = \phi_X(x) \text{ if } x \in V_X;$$

$$\psi_K(y) = \psi_H(y) \text{ if } y \in E_H \text{ and } \psi_K(\beta(y)) = \psi_X(y) \text{ if } y \in E_X;$$

$$\iota_K(z) = \gamma(z) \text{ if } z \in E_H \text{ and } \iota_K(\beta(z)) = \iota_X(z) \text{ if } z \in E_X.$$

A sequence $Z \Rightarrow H_1 \Rightarrow H_2 \Rightarrow \cdots \Rightarrow H_n$ is called a *derivation* for H_n. The transitive reflexive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$. A hypergraph $H \in HGR_{\Sigma,\Gamma}$ such that $Z \Rightarrow^* H$ is called a *sentential form* of G. The *language generated by* G, denoted by $L(G)$, is the set $\{H \in HGR_{\Delta,\Omega} \mid Z \Rightarrow^* H\}$.

Example 3.3. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be an HRNCE grammar such that $\Sigma = \{a, b, *\}$, $\Delta = \{*\}$, $\Gamma = \{A, \#\}$, $\Omega = \{\#\}$, $Z = \boxed{A}$, and P consists of the two productions given in Fig. 4(a). G generates the set of all "ladders" of the form as shown in Fig. 4(b). In fact, G generates graphs only and is essentially identical to the HH grammar with no duplication or deletion given in [6, p. 226] and the eNCE grammar given in [15, p. 314].

Example 3.4. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be an HRNCE grammar such that $\Sigma = \Delta = \{*\}$, $\Gamma = \{A, B, \#\}$, $\Omega = \{\#\}$, $Z = \boxed{\#}\overset{*}{\bullet}\boxed{A}$, and P consists of the four productions given in Fig. 5(a). G generates the set of all "double stars" of the form as shown in Fig. 5(b), where the rank of the center of the left star can grow arbitrarily and the degree of the center of the right star can grow arbitrarily. This example shows that an HRNCE grammar can generate rank- and degree-unbounded hypergraphs.

Example 3.5. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be an HRNCE grammar such that $\Sigma = \Delta = \{*\}$, $\Gamma = \{A, \#\}$, $\Omega = \{\#\}$, $Z = \boxed{A}\overset{*}{\bullet}\boxed{\#}\overset{*}{\bullet}\boxed{A}$, and P consists of the two productions given in Fig. 6(a). G generates the set of all (duals of) "binary trees" of the form as shown in Fig. 6(b).

HRNCE grammars are not confluent, i.e., derived hypergraphs are dependent upon the order of application of production rules. Two subclasses of confluent (hyper)graph grammars that have been studied much in the literature are the separated and linear classes. We shall define their HRNCE versions, which can be easily seen to be confluent. Observe that the grammars given in Examples 3.3 and 3.4 are both linear HRNCE grammars and the grammar given in Example 3.5 is a separated HRNCE grammar.

Definition 3.6. An HRNCE grammar is a *separated HRNCE* (*S-HRNCE*) *grammar* if no two nonterminal edges are adjacent in the axiom and in the right-hand side of any production. (This implies that no two nonterminal edges are adjacent in any of its sentential forms, and vice versa; similar to other separated classes of grammars.) It is

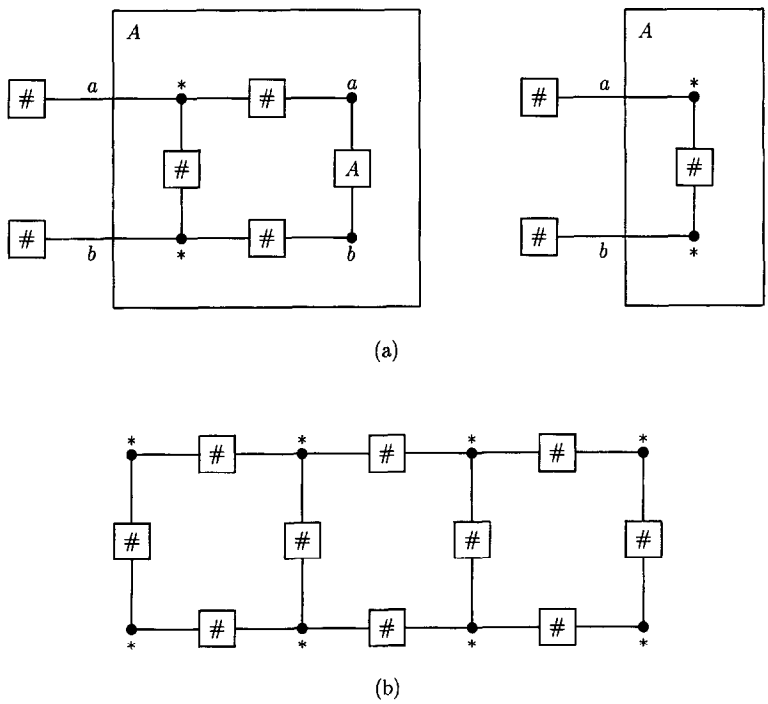


Fig. 4. (a) An HRNCE grammar generating all ladders; (b) an example of a ladder.

a *linear HRNCE* (*Lin-HRNCE*) *grammar* if there is at most one nonterminal edge in the axiom and in the right-hand side of each production.

4. Description power

We shall show that HRNCE grammars can generate all recursively enumerable (r.e.) languages. As each HRNCE language is clearly r.e., this implies that HRNCE languages are equivalent to r.e. languages. In fact, it turns out that the chain (or graph) generating power of HRNCE grammars is also r.e. We shall identify this r.e. language description power with the edge erasing capability of HRNCE grammars and show that HRNCE grammars without edge erasing characterize the PSPACE languages.

Theorem 4.1. *For every r.e. language L, we can construct an HRNCE grammar G such that $L(G)$ = o-chain(L).*

Proof. Let $M = (Q, I, T, \delta, q_0, \flat, F)$ be an arbitrary deterministic Turing machine with a single semi-infinite tape which is infinite to the right; Q is the set of states, I is the input alphabet, T is the total tape alphabet, $\delta : Q \times T \to Q \times T \times \{l, r\}$ is the transition function (l, r indicate the left and right moves), $q_0 \in Q$ is the initial state,

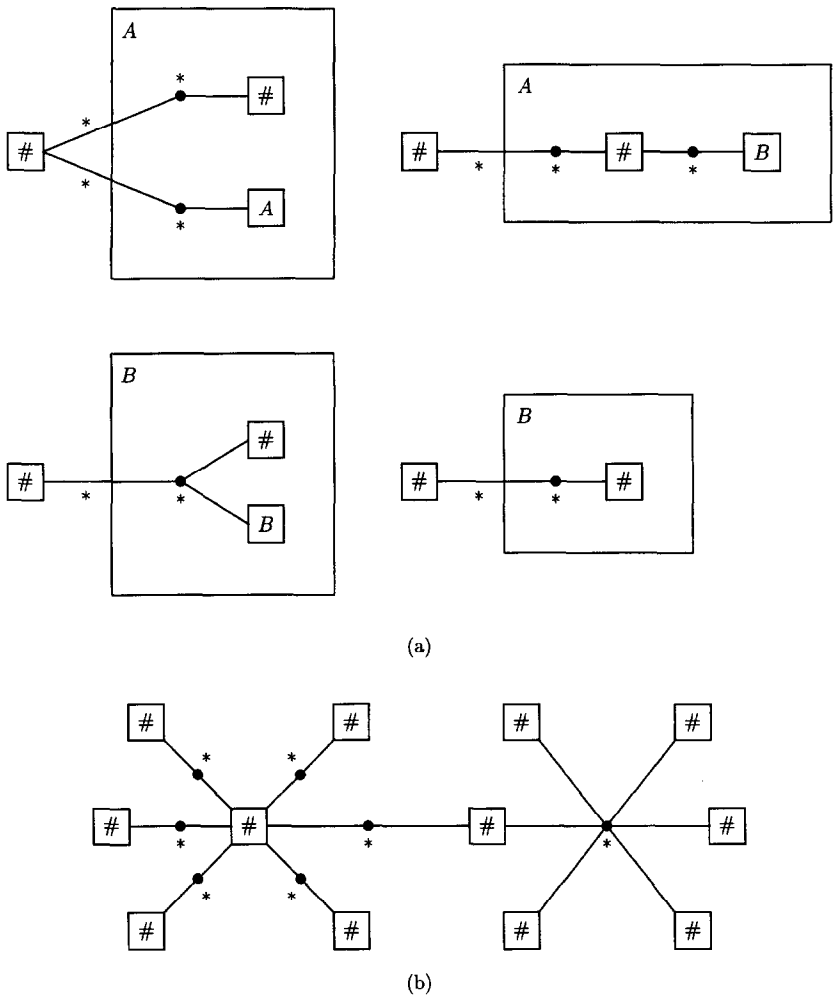


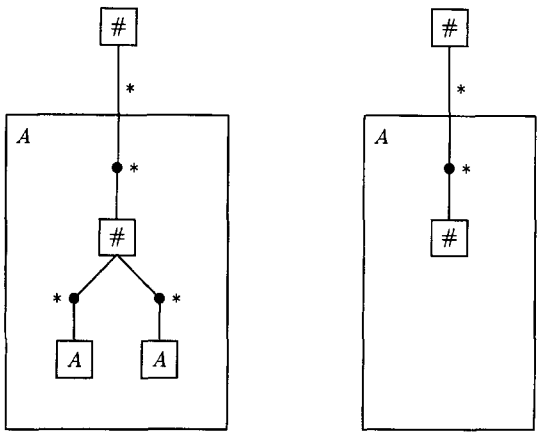
Fig. 5. (a) An HRNCE grammar generating all double stars; (b) an example of a double star.

$\flat \in T - I$ is the blank symbol, and $F \subseteq Q$ is the set of accepting states. We shall assume without loss of generality that (1) M does not accept any word of length zero or one and initially scans the leftmost input symbol (located in the leftmost tape cell), and (2) M does not print a blank symbol and accepts while scanning the leftmost blank symbol.

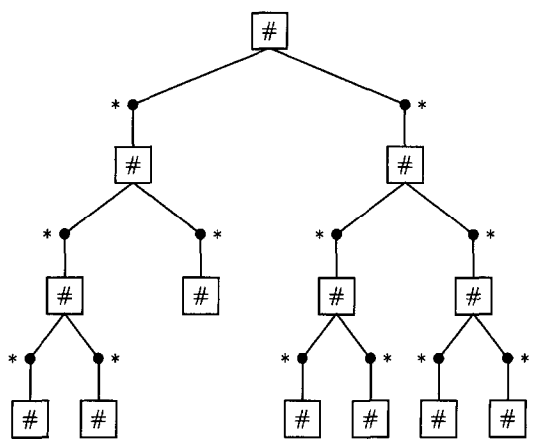
Let $\bar{I} = \{\bar{a} \mid a \in I\}$ and $\sigma(n) = (n+3) \bmod 3$ for all $n \geqslant -1$. Construct an HRNCE grammar $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ such that

$$\Sigma = \{*, @\} \cup T \cup \{\langle q, a\rangle \mid q \in Q, a \in T\};$$

$$\Delta = \{*\};$$



(a)



(b)

Fig. 6. (a) An HRNCE grammar generating all binary trees; (b) an example of a binary tree.

$$\Gamma = \{A, B, C, \bar{\$}, \$, \#\} \cup I \cup \{[\bar{a}, 1] \mid a \in I\}$$
$$\cup \{[a, i], (a, i) \mid a \in I \cup \{\bar{b}\}, i \in \{0, 1, 2\}\};$$
$$\Omega = \{\$\} \cup I;$$
$$Z = \boxed{A};$$

and P consists of the following productions:

(1) the productions in Fig. 7 for all $a, b \in I$ and all $i \in \{0, 1, 2\}$;

(2) the productions in Fig. 8(a) for all $a, f \in I \cup \{\bar{b}\}$, all $b, d \in T$, all $c \in T - \{\bar{b}\}$, all $e \in I \cup \bar{I} \cup \{\bar{b}\}$, all $g \in \Sigma$, all $p, q \in Q$, and all $i \in \{0, 1, 2\}$, provided that δ: $(p, b) \mapsto (q, c, r)$ is a right move of M;

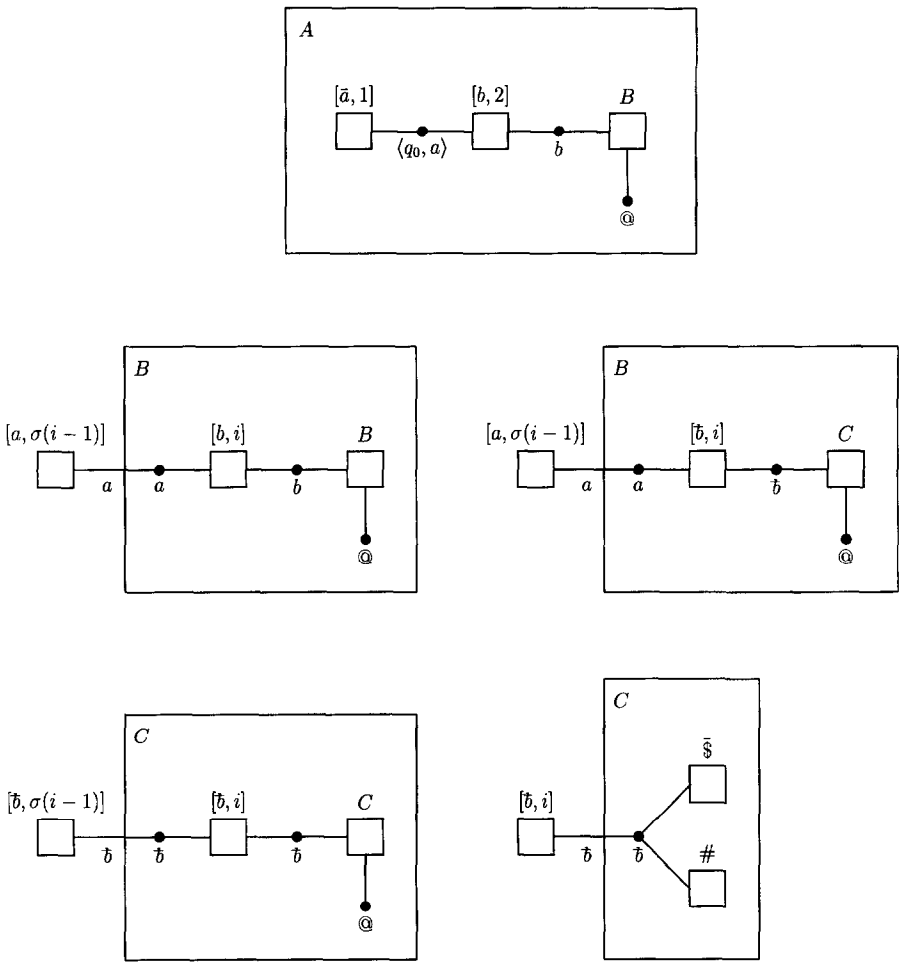


Fig. 7. The productions to create an initial configuration of M.

(3) the productions in Fig. 8(b) for all $a, f \in I \cup \{\bar{b}\}$, all $b \in T$, all $c, d \in T - \{\bar{b}\}$, all $e \in I \cup \bar{I} \cup \{\bar{b}\}$, all $g \in \Sigma$, all $p, q \in Q$, and all $i \in \{0, 1, 2\}$, provided that δ: $(p, b) \mapsto (q, c, l)$ is a left move of M;

(4) the productions in Fig. 9 for all $a, e, f \in I \cup \{\bar{b}\}$, all $b \in T - \{\bar{b}\}$, all $c \in T \cup \{\langle p, \bar{b}\rangle \mid p \in F\}$, all $d, s \in I$, all $g \in \Sigma$, all $h \in \Sigma - \{\langle p, \bar{b}\rangle \mid p \in F\}$, all $q \in F$, all $t \in I \cup \{\$, \#\}$, and all $i \in \{0, 1, 2\}$.

Let $x = a_1 a_2 \cdots a_n$ be any word in $L(M)$. Then, the hypergraph in Fig. 10(a), called a *chainar form* (a chain with two tails labeled by $\bar{\$}$ and #) of x, can be generated by using the productions in Fig. 7. This chainar form is interpreted as the initial configuration of M on x, where node labels simulate the tape configuration of M, edge labels keep the original input word, the symbols 0, 1, 2 in edge labels indicate orientation, and m ($> n \geqslant 2$) is the number of tape cells to be visited by M during

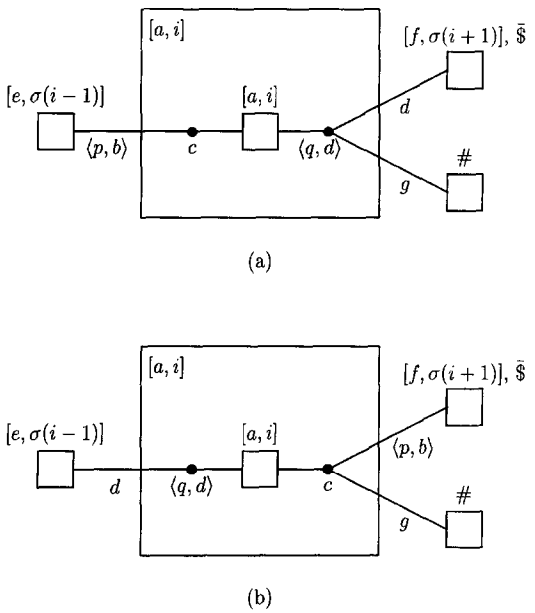


Fig. 8. The productions to simulate a move of M: (a) right move; (b) a left move.

the computation on x. Suppose that, at any point of computation, M is in state q and scans the i-th symbol b_i of the tape content $b_1 b_2 \cdots b_m$. This situation is described by the chainar form in Fig. 10(b) by G. Starting with any such chainar form, G can simulate a move of M by using the productions in Fig. 8, preserving the chainar form of the sentential form. When M enters an accepting state (scanning the leftmost blank symbol, i.e., the m-th symbol, of the tape), G can first traverse the chainar form from left to right, by using the first type of production in Fig. 9, in order to relabel each edge labeled by $[a,i]$, $a \in I \cup \{\bar{b}\}$ and $i \in \{0,1,2\}$, by (a,i), and then traverse it again from right to left, by using other productions in Fig. 9, in order to erase unnecessary nodes and edges and recover the original input word x in its oriented chain form. We show an example of a task that traverses the chainar form from right to left in Fig. 11, where q is assumed to be in F. It follows that o-chain$(L(M)) \subseteq L(G)$.

Now, let H be a hypergraph in $L(G)$ and consider any derivation D for H in G. Each sentential form in D is a forest of chains and/or a chainar form. In D, there is a derivation step that introduces two edges labeled by $\bar{\$}$ and #. (See Fig. 7. G must use each of the A-, B- and C-productions at least once in D since no edge labeled by B or C in a sentential form can be removed by rewriting a neighboring handle because of the @-labeled node.) Consider the derivation step that removes the nonterminal edge label #. It is not difficult to observe that this can be done only by the second production in Fig. 9, i.e., by rewriting the $\bar{\$}$-labeled edge by \$ and, at the same time, its incident node labeled by $\langle q, \bar{b}\rangle$, for some $q \in F$, by @. (If the node incident to the $\bar{\$}$-labeled edge is not labeled by $\langle q, \bar{b}\rangle$ for some $q \in F$, then the edge label # and the newly created node

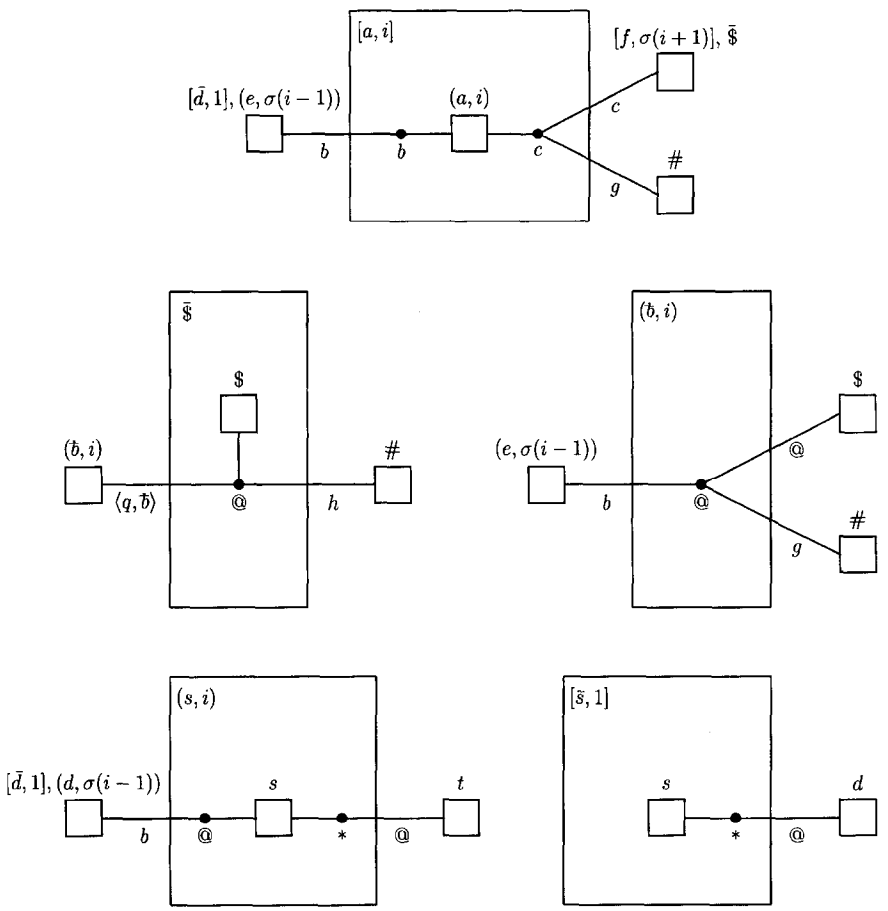


Fig. 9. The productions to recover the input word of M.

label @ cannot be removed in the future derivation step.) Now, the nonterminal node label @ created in this derivation step can be removed only by rewriting the handle associated with a $(\bar{b}, i)$-labeled edge, for some i, which contains the @-labeled node, via the third type of production in Fig. 9. (A $(\bar{b}, i)$-labeled edge and its two incident nodes are replaced by a single node labeled by @.) It is easy to observe now that the label @ created initially while removing the #-labeled edge must traverse from right to left on a chain and be removed at the left end, by applying the third type of production in Fig. 9 as many as the number of edges labeled by $(\bar{b}, i)$, $i \in \{0, 1, 2\}$, to remove all such edges, and then the fourth type of production in Fig. 9 as many as the number of edges labeled by (d, i), $d \in I$ and $i \in \{0, 1, 2\}$, to relabel them by the corresponding symbols from I, and finally the fifth type of production in Fig. 9 to complete the derivation with an oriented chain form. Note that this is possible only if the starting sentential form for this process (to which the $\bar{\$}$-production is applied) is connected and each of its rank-two edges is labeled by some (a, i), $a \in I \cup \{\bar{b}\}$ and

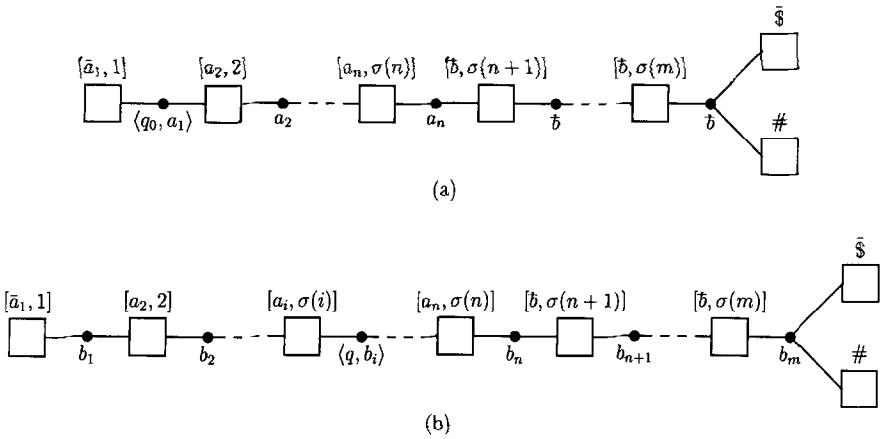


Fig. 10. Sentential forms for configurations of M: (a) an initial configuration; (b) a typical intermediate configuration.

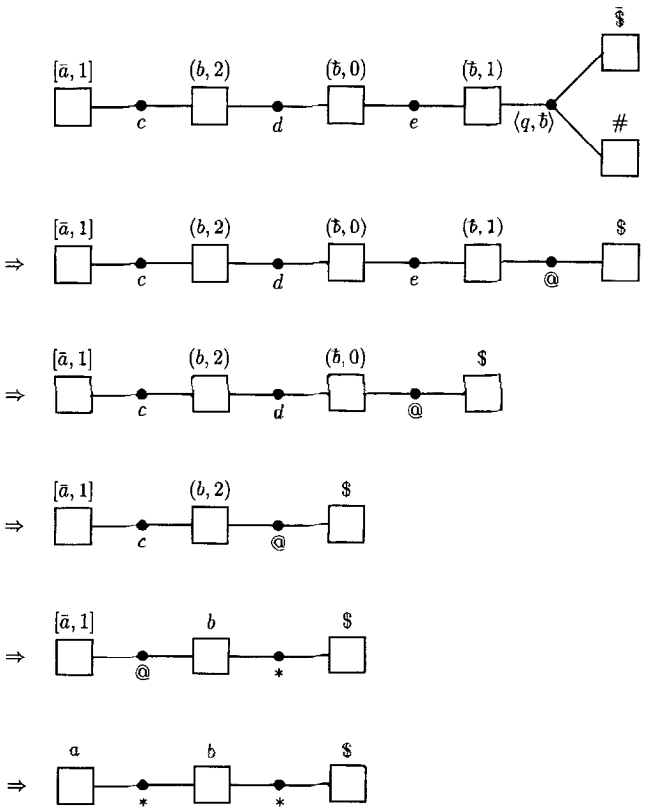


Fig. 11. A derivation that recovers the orginal input word of M.

$i \in \{0,1,2\}$. (The second condition guarantees that no @-labeled node is mistakenly removed by simulating a move of M.) Suppose that $[\bar{a}_1, 1]$, $[a_2, 2]$, ... , $[a_n, \sigma(n)]$ is the sequence of edge labels, with each $a_i \in$ I, which are introduced by applying the productions in Fig. 7. Let $\pi_1, \pi_2, \ldots, \pi_k$ be the sequence of productions from Fig. 8 which are introduced in D. Then, M can clearly accept the word $a_1 a_2 \cdots a_n$ by using the transitions corresponding to $\pi_1, \pi_2, \ldots, \pi_k$ since G can preserve the chainar form of its sentential form only by simulating moves of M accurately. It follows that $L(G) \subseteq$ o-chain($L(M)$). □

The r.e. description power of HRNCE grammars originates from their selective *edge-erasing capability* that can filter out undesired derivations from generating terminal hypergraphs by use of a so-called *blocking edge* such as one labeled by # or a so-called *blocking node* such as one labeled by @, as used in the above proof. Note that an HRNCE grammar can remove an edge in a sentential form in two ways:

(1) by an explicit edge-erasing mechanism via an edge-erasing production whose right-hand side contains no edge; and

(2) by an implicit edge-erasing mechanism that removes an edge by removing the incident nodes/surrounding handles in a derivation.

The grammar G in the proof of Theorem 4.1 uses both types of edge erasing (i.e., erasing of the edge labeled by $(\bar{b}, i)$ and #, respectively), but the first type of edge erasing can be removed: simply attach a $\bar{b}$-labeled, rank-one edge to the @-labeled node in the right-hand side of the third production in Fig. 9, where $\bar{b}$ is a new nonterminal edge label (this $\bar{b}$-labeled edge disappears when @ moves left). G also uses both a blocking edge and a blocking node (labeled by # and @, respectively), but the latter can be replaced by a blocking edge if we remove the first type of edge erasing as described above and similarly attach a $\bar{b}$-labeled, rank-one edge to each @-labeled node in the right-hand sides of the productions in Fig. 7. Thus, we can simulate r.e. languages by using HRNCE grammars with the edge-erasing mechanism of the second type only, together with a blocking edge.

It is also possible to simulate r.e. languages by using HRNCE grammars with the edge-erasing mechanism of the first type only, together with a blocking node. For this, let us describe a word $x = a_1 a_2 \cdots a_n$ by its *modified oriented chain*, denoted by m-o-chain(x), which is obtained from the oriented chain in Fig. 2 by attaching a $*$-labeled, degree-one node to each edge. Now, consider the hypergraph obtained from the sentential form in Fig. 10(a) by removing the #-labeled edge and by attaching a $*$-labeled, degree-one node to each edge and the hypergraph obtained from the sentential form in Fig. 10(b) by doing a similar modification. Such a structure can be created and preserved while simulating the Turing machine M by an HRNCE grammar G' that can be obtained by a simple modification of G and, intuitively, no erasing of the second type occurs. Thus, $L(G') =$ m-o-chain($L(M)$) and M' uses the edge-erasing mechanism of the first type only, together with a blocking node.

NLC grammars can filter out undesired derivations by disconnecting the finally generated graphs [24]. On the other hand, eNCE grammars use a blocking edge, which

is permanent once it is created, for the same purpose [15]. Our HRNCE model uses a blocking edge and/or a blocking node that can or cannot be removed depending on a particular derivation by its selective erasing capability. If we eliminate both types of edge erasing from HRNCE grammars, then their languages are in PSPACE. Such a grammar without edge erasing can generate a PSPACE-complete language, by using a blocking node only. See Theorem 4.4 below. This situation is comparable to NLC and eNCE grammars that also characterize the PSPACE languages in a similar way.

Definition 4.2. An *N-HRNCE grammar* is an HRNCE grammar without edge erasing, i.e., it never erases an already created edge except for rewriting of its associated handle by a hypergraph containing at least one edge.

Lemma 4.3. *For every context-sensitive language L, we can construct an N-HRNCE grammar G such that $L(G) = $ m-o-chain(L).*

Proof. Let M be a linear-bounded automaton accepting L. If we simulate M by an HRNCE grammar G without edge erasing, along the method to simulate a Turing machine by an HRNCE grammar with edge-erasing productions and a blocking node as discussed earlier, then $L(G) =$ m-o-chain(L) and G has no edge-erasing property. □

Theorem 4.4. *Every N-HRNCE language is in PSPACE. There exists a PSPACE-complete N-HRNCE language.*

Proof. Let G be an N-HRNCE grammar and H an arbitrary input hypergraph. Assume without loss of generality that the axiom of G consists of a single nonterminal edge without nodes. Suppose that $H \in L(G)$ and let D be a derivation for H in G. Consider an arbitrary intermediate sentential form K in D. Note first that each isolated node in K remains isolated in all subsequent sentential forms in D. Now, each nonempty edge e in K has two types of incident nodes: the nodes introduced together with e in a single derivation step, from the right-hand side of a production, and the nodes introduced in later derivation steps and connected to e via embedding relations. The sum of all nodes of the former type over all edges in K, together with the isolated nodes in K, is exactly the node set of K. Let c be the number of isolated nodes in H and let d be the maximum rank of the edges in the right-hand sides of the productions of G. Then, clearly $\#V_K \leqslant c + d \cdot \#E_K$. The number of edges in a sentential form never decreases in D. This means that K contains at most $\#E_H$ edges and at most $c + d \cdot \#E_H$ nodes. Namely, each intermediate sentential form in D is size-bounded by a polynomial function in the size of H. Therefore, such a derivation can be guessed in polynomial space since we need only store two consecutive sentential forms on the worktape of a Turing machine. It follows that $L(G) \in$ PSPACE.

There exists a context-sensitive language L_0 which is PSPACE-complete [17]. By Lemma 4.3, there exists an N-HRNCE grammar G_0 such that $L(G_0) =$ m-o-chain(L_0). A word x can be easily transformed into its modified oriented chain. Clearly, $x \in L_0$

if and only if m-o-chain(x) $\in L(G_0)$. Namely, L_0 reduces to $L(G_0)$. Therefore, $L(G_0)$ is a PSPACE-complete N-HRNCE language. □

We note that the edge-erasing property of either type, that an HRNCE grammar ever erases an already created edge (except for rewriting of its associated handle by a hypergraph containing at least one edge) to generate a terminal hypergraph, is an undecidable property. This can be esaily observed by considering a reduction from the Turing machine emptiness problem, which is undecidable, along the construction of G in the proof of Theorem 4.1 or its modified version G' as discussed earlier. In the next two sections, we shall study some basic properties (normal forms and complexity) of S-HRNCE grammars, for which this property can be effectively removed.

5. Normal forms

We shall present some useful normal form results for S-HRNCE grammars. In particular, the edge-erasing property can be removed from S-HRNCE grammars and there exist Chomsky and Greibach normal forms for S-HRNCE grammars. We shall also observe that S-HRNCE languages are context-free in the sense of Courcelle [5], i.e., they are generated by confluent and associative S-HRNCE grammars. The reader can observe that many normal forms discussed in this section are known for B-NLC, B-eNCE and/or B-edNCE grammars and the results stated in this section hold for Lin-HRNCE grammars as well.

Definition 5.1. An S-HRNCE grammar G is *simple* if, for each sentential form H in G and each nonterminal edge e in H, the nodes incident to e are pairwise distinctly labeled.

Lemma 5.2. *Every S-HRNCE language can be generated by a simple S-HRNCE grammar.*

Proof. Let G be an S-HRNCE grammar. Each production (A,X,C) in G is transformed into another one as follows. We first modify X by using node contraction. Let e be a nonterminal edge in X and let $x_1,x_2,\ldots,x_n$ $(n\geqslant 1)$ be the nodes incident to e and labeled by the same symbol, say a. Remove $x_1,x_2,\ldots,x_n$ and add a new node $x_{1,2,\ldots,n}$ labeled by a to e and to all other (terminal) edges e' in X containing at least one x_i. Replace each triple $(x_i, b, B) \in C$, $1 \leqslant i \leqslant n$, by $(x_{1,2,\ldots,n}, b, B)$. Repeat this for all nonempty nonterminal edges in X and all node labels, in any order. Now, modify the axiom of G by using the same node contraction. It is easy to see that the resulting S-HRNCE grammar generates $L(G)$ and has the property that nodes incident to a nonterminal edge are pairwise distinctly labeled in each sentential form. □

Definition 5.3. The *context* of a nonterminal edge e in a hypergraph H, denoted by $\text{context}_H(e)$, is the set $\{(a, B) \mid$ an edge e' with $\psi_H(e') = B$ is a-adjacent to $e\}$.

An S-HRNCE grammar $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ is *context-consistent* if there is a function $\eta : \Gamma - \Omega \to 2^{\Sigma\times\Omega}$ such that, for each sentential form H in G and for each nonterminal edge e in H, $\text{context}_H(e) = \eta(\psi_H(e))$. The function η satisfying this property is called the *context-describing function* of G.

Lemma 5.4. *Every S-HRNCE language can be generated by a context-consistent S-HRNCE grammar.*

Proof. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be an S-HRNCE grammar and consider an arbitrary derivation step $H \Rightarrow_{(e,\pi)} K$ in G. The context of nonterminal edges created from the right-hand side of π can be calculated from the context of e in H and the context of a nonterminal edge does not change until its associated handle is rewritten. This observation was used in [37] to transform B-NLC grammars into the context-consistent normal form and we shall follow the same transformation technique. Relabel each nonterminal edge label A in Z by A_r if the context of this edge is r. Transform each production (A, X, C) of G into as many as $2^{\#(\Sigma\times\Omega)}$ productions of the form (A_r, X', C), where $r \subseteq \Sigma \times \Omega$ and X' is obtained from X by relabeling each nonterminal edge e by B_u if $\psi_X(e) = B$ and $u = \text{context}_X(e) \cup \{(\phi_X(v), B') \mid v \in \iota_X(e), (v, a, B') \in C, \text{ and } (a, B') \in r\}$. The proof for the fact that the resulting grammar is a context-consistent S-HRNCE grammar generating $L(G)$ is fully analogous to the B-NLC case and is left to the reader. □

Definition 5.5. A *Λ-production* is one whose right-hand side is Λ. A *chain production* is one whose right-hand side is a single nonterminal handle. (A handle is a *nonterminal handle* if its associated edge is labeled by a nonterminal symbol; otherwise, it is a *terminal handle*.)

Lemma 5.6. *Every S-HRNCE language without Λ can be generated by an S-HRNCE grammar without Λ- and chain productions.*

Proof. Analogous to the cases of context-free grammars [22] and eNCE grammars [15]. Probably, we need only note the following. In the case of Λ-productions, each handle in the right-hand side of a production that can derive Λ is optionally removed. If such removal of handles results in an empty terminal edge in the right-hand side of a production, then it must also be removed. In the case of chain productions, the embedding relations of the productions replacing a chain production can be calculated by inductively combining the embedding relations of two productions that can be used consecutively into one. See Definitions 5.17 and 5.18 for such a calculation. □

Definition 5.7. An S-HRNCE grammar $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ is *neighborhood preserving* if, for all $H, K \in HGR_{\Sigma,\Gamma}$ such that $Z \Rightarrow^* H \Rightarrow_{(e,\pi)} K$, each edge adjacent to e in H is incident to at least one node from the right-hand side of π in K.

Lemma 5.8. *Every S-HRNCE language without Λ can be generated by a neighborhood-preserving S-HRNCE grammar.*

Proof. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be a context-consistent S-HRNCE grammar generating a language without Λ. Assume without loss of generality that G contains no Λ-production and Z contains one edge only, which is nonterminal. Let η be the context-describing function of G.

We construct an S-HRNCE grammar $G' = (\Sigma, \Delta, \Gamma', \Omega, P', Z')$ such that, for every derivation $Z \Rightarrow^* H \Rightarrow^* K$ $(\in HGR_{\Delta,\Omega})$ in G, there is a corresponding derivation $Z' \Rightarrow^* H' \Rightarrow^* K'$ $(= K)$ in G' such that, if e is a nonterminal edge adjacent to a terminal edge e' in H and e' is not incident to any node created from e in K, then e and e' are not adjacent in H' in the first place. Such a disconnection between e and e' in H' is done by removing the nodes shared by e and e' from the node set of e' and, as a result, e' may also be removed if its node set becomes empty. To construct such a grammar G', we shall adopt the method used for B-NLC grammars [37]. (An essentially identical method was also used for B-edNCE grammars in [16].) Augment each nonterminal edge label A by adding an index $r \subseteq \eta(A)$: the modified label A_r means that the edge with this label has not established a connection to B-labeled edges via a-labeled nodes, for all $(a, B) \in r$, even though it should have if followed the productions of G. The information contained in r can be transferred to newly introduced nonterminal edge labels by using the embedding relation of a production and each "forbidden" embedding relation (a, B) in r can be eventually verified and deleted when applying a production whose right-hand side does not establish connection to B-labeled edges via a-labeled nodes.

Formally, G' is defined as follows. Let $\Gamma' = \{A_r \mid A \in \Gamma - \Omega, r \subseteq \eta(A)\} \cup \Omega$ and let Z' be the same as Z except that the only edge in Z, which is nonterminal, is augmented with $\emptyset$. To define P', let $\pi = (A, X, C)$ be a production in P. Let X' be any hypergraph obtained from X by replacing the label Y of each nonterminal edge e in X by Y_u for some $u \subseteq \eta(Y)$ and by deleting all nodes labeled by a from the node set of each B-labeled edge that is a-adjacent to e in X if $(a, B) \in u$. (Note that no node of X is removed in this process; only the membership of a node in the node set of a terminal edge can be removed. As the result, some terminal edges may disappear if they lose all incident nodes.) Then, each production of the form (A_r, X', C) with $r \subseteq \eta(A)$ is in P' if the following conditions hold:

(1) for each $(a, B) \in \eta(A) - r$, $(v, a, B) \in C$ for some node $v \in V_X$;

(2) for each node v and each nonterminal edge e incident to v in X, if $(a, B) \in r$, $(v, a, B) \in C$, and $\psi_{X'}(e) = Y_u$ then $(\phi_X(v), B) \in u$; and

(3) for each node v not incident to any nonterminal edge in X, if $(a, B) \in r$ then $(v, a, B) \notin C$.

The proof for the fact that G' is a context-consistent, neighborhood-preserving S-HRNCE grammar (with the context-describing function η' such that $\eta'(A_r) = \eta(A) - r$ for all $A_r \in \Gamma'$) generating $L(G)$ is fully analogous to the one in [16] or [37] and is left to the reader. □

Lemma 5.9. *Every S-HRNCE language without Λ can be generated by an S-HRNCE grammar without edge erasing.*

Proof. Recall that an HRNCE grammar can erase an edge either by an edge-erasing production or by removing its incident nodes/surrounding handles, as discussed in Section 4. Lemma 5.8 implies that the edge erasing of the second type can be removed from S-HRNCE grammars. For the first type of edge erasing, we need only remove productions whose right-hand sides contain isolated nodes only since Λ-productions can be removed from S-HRNCE grammars (Lemma 5.6). This can be done easily by using a standard production substitution technique (similar to the Λ- and chain-production removal) and is left to the reader. □

Definition 5.10. An S-HRNCE grammar is *reduced* if it is simple, context-consistent, chain-production free, neighborhood-preserving, and edge-erasing free.

Theorem 5.11. *Every S-HRNCE language without Λ can be generated by a reduced S-HRNCE grammar.*

Proof. Let G be an S-HRNCE grammar not generating Λ. Using the methods discussed in this section, successively transform G into a grammar which is simple, Λ-production free, context-consistent, neighborhood-preserving, chain-production free, and edge-erasing free (it is sufficient to remove productions whose right-hand sides contain isolated nodes only, as discussed in the proof of Lemma 5.9), in this sequence. Then the resulting grammar is a reduced S-HRNCE grammar generating $L(G)$. □

Definition 5.12. An S-HRNCE grammar is in *Chomsky normal form* if the right-hand side of each production consists of (1) either isolated nodes or a terminal handle but not both, and optionally a nonterminal handle, or (2) two nonterminal handles with no isolated node.

Theorem 5.13. *Every S-HRNCE language without Λ can be generated by a reduced S-HRNCE grammar in Chomsky normal form.*

Proof. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be an S-HRNCE grammar without edge erasing. We construct an S-HRNCE grammar $G' = (\Sigma', \Delta, \Gamma', \Omega, P', Z)$ containing only chain productions and productions in Chomsky normal form such that $L(G') = L(G)$, by using a method similar to the one for B-eNCE grammars [15]. G' can be further transformed into an equivalent reduced form by using the techniques discussed earlier in this section; the resulting grammar preserves the Chomsky normal form.

The idea is simple. If any production of G violates the rule for G', i.e., its right-hand side contains (1) isolated nodes together with a terminal handle or more than one handle or (2) no isolated node together with two terminal handles or more than two handles, then it is transformed into a set of productions for G' that must be executed in sequence, as in the Chomsky normal-form transformation for context-free grammars. To establish appropriate embedding relations for new productions, we introduce special

node and edge labels:

$$\Sigma' = \Sigma \cup \{\bar{v} \mid (A, X, C) \in P \text{ and } v \in V_X\};$$
$$\Gamma' = \Gamma \cup \{[\pi, i] \mid \pi \in P \text{ and } 1 \leqslant i \leqslant \#E_X \text{ if } X \text{ is the right-hand side of } \pi\}.$$

To define P', let $\pi = (A,X,C)$ be any production in P that is not in the desired form and let $e_1, e_2, \ldots, e_n$ be any fixed ordering of the edges in X such that all terminal edges, if any, come before nonterminal edges. We construct $n+1$ productions, π_0, $\pi_1, \ldots, \pi_n$, for π. The first production π_0 rewrites a handle whose associated edge is labeled by A and creates a nonterminal edge ξ_0, labeled by $[\pi, 1]$, and all nodes in X, that can be connected to outside edges exactly as done by π by using the embedding relation C. In the right-hand side of π_0, all isolated nodes of X are labeled as in X and remain isolated. Other nodes are contained in the node set of ξ_0 and are labeled by their (barred) names so that connection to their incident edges in X, introduced by other productions, can be made appropriately. For $i = 1, 2, \ldots, n-1$, the production π_i rewrites a handle whose associated edge is labeled by $[\pi, i]$ and creates two edges, e_i and a new nonterminal edge ξ_i labeled by $\psi_X(e_i)$ and $[\pi, i+1]$, respectively, and re-creates all nodes incident to ξ_{i-1} in the right-hand side of the production π_{i-1}. Each node incident to e_i but not to any edge indexed higher than i in X is incident to e_i only in the right-hand side of π_i and is labeled by its original label, i.e., as in X. Each node incident to e_i and some edge indexed higher than i in X is incident to both e_i and ξ_i in the right-hand side of π_i and is labeled by its barred name. All other nodes are incident to ξ_i only in the right-hand side of π_i and are labeled by their barred names. Each of these nodes will be connected to all previously incident edges, by using the special labels (the barred node names) of the nodes incident to ξ_{i-1} to which π_i is applied. Finally, π_n rewrites a handle whose associated edge is labeled by $[\pi, n]$ and creates the last edge e_n and its incident nodes, labeled as in X, and establishes connection between these nodes and their previously incident edges.

Formally, the $n+1$ productions in P' that correspond to $\pi = (A,X,C)$ in P are defined as follows:

(1) $\pi_0 = (A, X_0, C)$, where $E_{X_0} = \{\xi_0\}$, $V_{X_0} = V_X$, $\iota_{X_0}(\xi_0) = \{v \in V_X \mid v \text{ is not isolated}\}$, $\psi_{X_0}(\xi_0) = [\pi, 1]$, $\phi_{X_0}(v) = \phi_X(v)$ if $v \notin \iota_{X_0}(\xi_0)$ and $\phi_{X_0}(v) = \bar{v}$ if $v \in \iota_{X_0}(\xi_0)$ for all $v \in V_{X_0}$;

(2) $\pi_i = ([\pi, i], X_i, C_i)$, $1 \leqslant i \leqslant n-1$, where $E_{X_i} = \{e_i, \xi_i\}$, $V_{X_i} = \iota_{X_{i-1}}(\xi_{i-1})$, $\iota_{X_i}(e_i) = \iota_X(e_i)$, $\iota_{X_i}(\xi_i) = \bigcup_{j>i} \iota_X(e_j)$, $\psi_{X_i}(e_i) = \psi_X(e_i)$, $\psi_{X_i}(\xi_i) = [\pi, i+1]$, $\phi_{X_i}(v) = \phi_X(v)$ if $v \notin \iota_{X_i}(\xi_i)$ and $\phi_{X_i}(v) = \bar{v}$ if $v \in \iota_{X_i}(\xi_i)$ for all $v \in V_{X_i}$, and $C_i = \{(v, \bar{v}, B) \mid v \in V_{X_i} \text{ and } B \in \Omega\}$; and

(3) $\pi_n = ([\pi, n], X_n, C_n)$, where $E_{X_n} = \{e_n\}$, $V_{X_n} = \iota_{X_n}(e_n) = \iota_X(e_n)$, $\psi_{X_n}(e_n) = \psi_X(e_n)$, $\phi_{X_n}(v) = \phi_X(v)$ for all $v \in V_{X_n}$, and C_n is the same as in (2) with $i = n$.

It is easy to see that the productions $\pi_0, \pi_1, \ldots, \pi_n$ must be applied in sequence in any derivation. As S-HRNCE grammars are confluent, we can in fact assume that they are applied in consecutive derivation steps. π_0 creates all nodes in X and establishes their connection to outside edges exactly as done by π. Each π_i ($1 \leqslant i \leqslant n$) creates

e_i, recovering the original labels of some of its incident nodes (not incident to any e_j, $j > i$) and preserving each node's incidence to other edges. Terminal edges in X must be generated before any nonterminal edge in X is generated since, otherwise, some nonterminal edge in X will be adjacent to some ξ_i in P'. With all these observations, it is not difficult to see that $L(G') = L(G)$ and G' contains only chain productions and productions in Chomsky normal form; a formal proof is left to the reader. As indicated before, G' can be transformed into the reduced form, preserving the Chomsky normal form. □

Definition 5.14. A Lin-HRNCE grammar is a *Lin1-HRNCE grammar* if the right-hand side of each production consists of either isolated nodes or a terminal handle, and optionally a nonterminal handle.

Theorem 5.15. *Every Lin-HRNCE language without Λ can be generated by a reduced Lin1-HRNCE grammar.*

Proof. Identical to the proof of Theorem 5.13. □

Definition 5.16. An S-HRNCE grammar is in *Greibach normal form* if the right-hand side of each production consists of either isolated nodes or a terminal handle, and optionally other nonterminal handles.

To transform an S-HRNCE grammar G into the Greibach normal form, we shall use a method similar to the one for context-free grammars. This involves two principal transformations on the productions of G: the production substitution and recursion removal. We shall define the production substitution by using a so-called hypergraph substitution function. (This function is similar to the one defined for B-edNCE grammars in [6] and will also be used later in this section to show context-freeness of S-HRNCE languages.) Then, with the notion of a recursive production appropriately defined, we shall show that a process essentially identical to the one for context-free grammars [22] works for S-HRNCE grammars.

Definition 5.17. A *hypergraph with embedding* over Σ and Γ is a pair (H, emb), where $H \in HGR_{\Sigma,\Gamma}$ and $\text{emb} \subseteq V_H \times \Sigma \times \Gamma$. Let $HGR^e_{\Sigma,\Gamma}$ be the set of all hypergraphs with embedding over Σ and Γ. Each hypergraph with embedding (H, emb) such that $\text{emb} = \emptyset$ is identified with H. Thus, $HGR_{\Sigma,\Gamma} \subseteq HGR^e_{\Sigma,\Gamma}$. Let (H, emb_H), $(K, \text{emb}_K) \in HGR^e_{\Sigma,\Gamma}$, where $V_H \cap V_K = \emptyset$ and $E_H \cap E_K = \emptyset$, and let $e \in E_H$. For each $e' \in E_H - \{e\}$, let $\gamma(e') = (\iota_H(e') - \iota_H(e)) \cup \{v \in V_K \mid (v, a, \psi_H(e')) \in \text{emb}_K$ and e, e' are a-adjacent in $H\}$. Then, the *hypergraph substitution* of (K, emb_K) for e (or handle(e) to be more precise) in (H, emb_H), denoted by $(H, \text{emb}_H)[e \leftarrow (K, \text{emb}_K)]$, is the hypergraph with embedding (J, emb_J) such that: $V_J = (V_H - \iota_H(e)) \cup V_K$; $E_J = \{e' \in E_H - \{e\} \mid \iota_H(e) \cap \iota_H(e') \neq \emptyset$ implies $\gamma(e') \neq \emptyset\} \cup E_K$; $\iota_J(e') = \gamma(e')$ if $e' \in E_H$ and $\iota_J(e') = \iota_K(e')$ if $e' \in E_K$; the node and edges labels in J are the same as in H or

K; and $\mathrm{emb}_J = (\mathrm{emb}_H - (\iota_H(e) \times \Sigma \times \Gamma)) \cup \{(v, a, B) \mid (v, \phi_H(v'), B) \in \mathrm{emb}_K$ for some $v' \in \iota_H(e)$ such that $(v', a, B) \in \mathrm{emb}_H\}$.

Definition 5.18. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be an S-HRNCE grammar. A *production substitution* in G is to remove a production (A,X,C) with $\psi_X(e) \in \Gamma - \Omega$ for some $e \in E_X$ and add a production $(A, \bar{X}, \bar{C})$ such that $(\bar{X}, \bar{C}) = (X,C)[e \leftarrow (Y,D)]$, for each production $(\psi_X(e), Y, D)$ in G.

Lemma 5.19. *A production substitution does not affect the language generated by an S-HRNCE grammar.*

Proof. A production substitution combines two derivation steps that can occur consecutively into one. It is not difficult to check its accuracy. Now, the confluence property of S-HRNCE grammars implies the lemma. □

Definition 5.20. A production (A,X,C), with at least one edge but no isolated node in X, is *recursive* if all edges in X are labeled by A.

Lemma 5.21. *Every S-HRNCE language without Λ can be generated by an S-HRNCE grammar without recursive productions.*

Proof. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be a reduced S-HRNCE grammar. Let η be the context-describing function of G. Suppose that $\pi_{1,i} = (A, X_i, C_i)$, $1 \leqslant i \leqslant m$, are all recursive A-productions and $\pi_{2,i} = (A, Y_i, D_i)$, $1 \leqslant i \leqslant n$, are all nonrecursive A-productions. (We assume that $m, n \geqslant 1$.) We show how these productions can be replaced by nonrecursive productions, independently of other productions in P and without changing the generated language.

For each $i = 1, 2, \ldots, n$, let Y_i' be the hypergraph obtained from Y_i by adding a new nonterminal edge δ_i with one node y_i, labeled by $\bar{A}$ and $*$, respectively, where $\bar{A}$ and $*$ are new labels. For each $i = 1, 2, \ldots, m$, let e_i be any fixed edge in X_i. Let X_i' be the hypergraph obtained from X_i by removing e_i and its incident nodes and adding a new nonterminal edge ξ_i with one node x_i, labeled by $\bar{A}$ and $*$, respectively. Let X_i'' be the hypergraph obtained from X_i by simply removing e_i and its incident nodes.

We shall keep all nonrecursive A-productions. Now, remove all recursive A-productions and add the following productions:

(1) $\pi_{2,i}' = (A, Y_i', D_i')$ for all $i = 1, 2, \ldots, n$, where $D_i' = D_i \cup (\{y_i\} \times \Sigma \times \Omega)$;

(2) $\pi_{1,i}' = (\bar{A}, X_i', C_i')$ for all $i = 1, 2, \ldots, m$, where $C_i' = \{(\phi_{X_i'}(v), *, B) \mid v \in V_{X_i} - \iota_{X_i}(e_i), B \in \Omega$ and $(\phi_{X_i'}(v), B) \in \eta(A)\} \cup (\{x_i\} \times \Sigma \times \Omega)$; and

(3) $\pi_{1,i}'' = (\bar{A}, X_i'', C_i'')$ for all $i = 1, 2, \ldots, m$, where $C_i'' = C_i' - (\{x_i\} \times \Sigma \times \Omega)$.

Let G' be the S-HRNCE grammar obtained from G by performing the above transformation for each edge label $A \in \Gamma - \Omega$ that defines recursive productions. Clearly, G' does not contain any recursive production. (Each Y_i' contains an $\bar{A}$-labeled edge. Each of X_i' and X_i'' contains at least one A-labeled edge since G is chain-production

free.) To understand our transformation, consider a short derivation in G that uses recursive A-productions $\pi_{1,i}$, $\pi_{1,j}$, $\pi_{1,k}$ and then breaks recursion by using a nonrecursive A-production $\pi_{2,l}$:

$$H_0 \Rightarrow_{(e_0,\pi_{1,i})} H_1 \Rightarrow_{(e_i,\pi_{1,j})} H_2 \Rightarrow_{(e_j,\pi_{1,k})} H_3 \Rightarrow_{(e_k,\pi_{2,l})} H_4,$$

where e_i (e_j, e_k) is an A-labeled edge in the right-hand side of $\pi_{1,i}$ ($\pi_{1,j}$, $\pi_{1,k}$). Note that all A-labeled edges created from e_0 in this derivation are pairwise node-disjoint since G is separated and that each such A-labeled edge has exactly the same context as e_0 since G is context-consistent. As no terminal edge or isolated node is created in this derivation except in the last step, each of these A-labeled edges must preserve the neighborhood of e_0 for its accurate context. With this observation, it is not difficult to see that the following is an accurate simulation of the above derivation:

$$H_0 \Rightarrow_{(e_0,\pi'_{2,l})} H'_1 \Rightarrow_{(\delta_i,\pi'_{1,i})} H'_2 \Rightarrow_{(\xi_i,\pi'_{1,j})} H_3 \Rightarrow_{(\xi_j,\pi''_{1,k})} H_4.$$

As such a simulation is clearly possible for a derivation of any length in G that involves recursive A-productions and G is confluent, it follows that $L(G) \subseteq L(G')$. The converse argument that $L(G') \subseteq L(G)$ can be similarly observed based on the fact that G is confluent and context-consistent. We shall leave it to the reader. □

Theorem 5.22. *Every S-HRNCE language without Λ can be generated by a reduced S-HRNCE grammar in Greibach normal form.*

Proof. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be a reduced S-HRNCE grammar in Chomsky normal form. We construct a reduced S-HRNCE grammar in Greibach normal form generating $L(G)$, by using a method similar to the one for context-free grammars [22]. Enumerate the nonterminal edge labels: $A_1, A_2, \ldots, A_n$, where $n = \#(\Gamma - \Omega)$. Now, construct an S-HRNCE grammar G' by performing the following in sequence:

(1) For $i = 1, 2, \ldots, n$ in sequence, transform all A_i-productions of G so that each A_i-production (A_i, X, C) has the property that X contains an isolated node, a terminal edge, or a nonterminal edge labeled by A_j for some $j > i$. This can be done by using a sequence of production substitutions (Lemma 5.19) followed by a recursion removal (Lemma 5.21). Recall that recursion removal introduces a new nonterminal edge label $\bar{A}_i$ and the right-hand side of each $\bar{A}_i$-production contains at least one edge with its label from Γ.

(2) The right-hand side of each A_n-production contains an isolated node or a terminal edge. Transform all other A_i-productions into such form by using production substitutions, for $i = n-1, n-2, \ldots, 1$ in sequence. Now, transform all $\bar{A}_i$-productions ($1 \leqslant i \leqslant n$) into such form by using production substitutions.

This transformation algorithm is identical to the one for context-free grammars in [22] except that the notion of production substitution and recursion has been modified. With Lemmata 5.19 and 5.21 and the fact that G is a reduced S-HRNCE grammar in Chomsky normal form, it should not be difficult to see that $L(G') = L(G)$ and G' contains some isolated nodes and/or terminal edges in the right-hand side of each

production. The productions of G' can be further modified so that the right-hand side of each production contains either some isolated nodes or exactly one terminal edge, by using a method similar to the Chomsky normal-form transformation (to replace redundant terminal edges by nonterminal edges). Finally, the resulting grammar can be converted into the reduced form as explained in the proof of Theorem 5.11. □

Our final subject in this section is the context-freeness of S-HRNCE languages, along the definition of context-free rewriting systems given in [5]. This requires a hypergraph substitution function satisfying the so-called preservation axiom (stated in Lemma 5.24 below) with which the S-HRNCE rewriting step can be defined. Let [ ] be the substitution function defined in Definition 5.17. The following lemmata can be proved in a straightforward way and their proofs are left to the interested reader, referring to Lemma 5.3 in [5] and Lemma 3.2 in [6] for a very similar proof.

Lemma 5.23. *For each reduced S-HRNCE grammar G, $H \Rightarrow_{(e,\pi)} K$ in G if and only if $K = H[e \leftarrow (X, C)]$, where $\pi = (A, X, C)$ and $\psi_H(e) = A$.*

Lemma 5.24. *The hypergraph substitution [ ] restricted to reduced S-HRNCE grammars satisfies the following context-free conditions as stated in* [5]:

(1) *Preservation Axiom: the object substituted (i.e., a handle) preserves the existence of other objects;*

(2) *Confluence Axiom: $H[e \leftarrow K][e' \leftarrow K'] = H[e' \leftarrow K'][e \leftarrow K]$ for all $H, K, K' \in HGR^e_{\Sigma,\Gamma}$ and all e, $e' \in E_H$;*

(3) *Associativity Axiom: $H[e \leftarrow K][e' \leftarrow K'] = H[e \leftarrow K[e' \leftarrow K']]$ for all $H, K, K' \in HGR^e_{\Sigma,\Gamma}$, all $e \in E_H$, and all $e' \in E_K$.*

Theorem 5.25. *Every S-HRNCE language without Λ can be generated by a context-free S-HRNCE grammar (in reduced, Chomsky or Greibach normal form).*

Proof. Follows from Lemmata 5.23 and 5.24 (and Theorems 5.13 and 5.22). □

6. Complexity of S-HRNCE Languages

We showed in Section 4 that HRNCE and N-HRNCE languages characterize the r.e. and PSPACE respectively languages. We continue investigation of the description power of S-HRNCE grammars along this line and show that S-HRNCE and Lin-HRNCE languages characterize the NP languages. More specifically, S-HRNCE languages are in NP and there is an NP-complete Lin-HRNCE language which satisfies any two of the following conditions: connected; degree-bounded; and rank-bounded. When all these three conditions are imposed, S-HRNCE and Lin-HRNCE languages characterize the classes LOGCFL and NLOG, respectively. This situation is similar to the complexity of B-edNCE languages which are in NP. There is an NP-complete Lin-edNCE language satisfying any two of the following conditions: connected; in-degree

bounded; and out-degree bounded [1], and every B-edNCE (Lin-edNCE) language satisfying all these three conditions is in LOGCFL (NLOG) [12]. Note also that CFHG languages characterize the NP languages and connected CFHG graph languages of bounded degree are in LOGCFL [33]. In fact, B-edNCE grammars, CFHG grammars, and separated HH grammars generate the same graph languages of bounded degree [10, 16].

Theorem 6.1. *Every S-HRNCE language is in* NP. *There exists an* NP*-complete Lin-HRNCE language which satisfies any two of the following conditions*: *connected*; *degree-bounded*; *and rank-bounded.*

Proof. The existence of reduced Greibach normal form for S-HRNCE grammars, as stated in Theorem 5.22, implies that S-HRNCE languages are in NP. For NP-hardness part, we shall use a reduction from the set CB_2 of all graphs with cyclic bandwidth at most two, which is NP-complete [28]. (A graph has cyclic bandwidth at most k if there exists a cyclic ordering of its nodes such that the cyclic distance between each pair of adjacent nodes is at most k. For convenience in our reduction, we shall assume without loss of generality that every graph in CB_2 has at least six nodes.) Turán [42] constructed a so-called monotone NLC grammar, in which the right-hand side of each production contains at least two nodes, that generates CB_2. There also exists a B-NLC grammar generating CB_2 [37]. We shall construct a Lin-HRNCE grammar G generating dual(CB_2), the set of all duals of the graphs in CB_2, using an idea similar to the ones in [37, 42].

Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$, where $\Sigma = \{b_1, b_2, b_{-1}, b_{-2}, *\}$, $\Delta = \{*\}, \Gamma = \{A, B, \#\}, \Omega = \{\#\}, Z = \boxed{A}$ and P consists of the following productions:

(1) the thirty two productions obtained from the production shown in Fig. 12(a) by removing or retaining each node labeled by $*$;

(2) the four productions obtained from the production shown in Fig. 12(b) by removing or retaining each node labeled by $*$; and

(3) the one hundred twenty eight productions obtained from the production shown in Fig. 12(c) by removing or retaining each node labeled by $*$.

It is easy to observe that $L(G) = \text{dual}(CB_2)$, e.g., by following a few derivation steps of G. As each graph (an instance of CB_2) can be easily converted to its dual (an instance of the membership problem for G), it follows, together with $L(G) \in$ NP as discussed before, that $L(G)$ is NP-complete. Furthermore, each hypergraph in $L(G)$ has degree at most two and rank at most four.

Now, consider the following modification G' of G. The hypergraph in Fig. 12(a) has an additional node labeled by a new terminal symbol $\bar{*}$ which is connected to all five edges. The productions in Figs. 12(b) and 12(c) can be easily modified so that this special node exists uniquely in each sentential form, being incident to all edges. Clearly,

Similarly, consider G'' obtained from G as follows: Add a new terminal edge labeled by $\bar{\#}$ consisting of five new terminal nodes labeled by $\bar{*}$ to the hypergraph in Fig. 12(a)

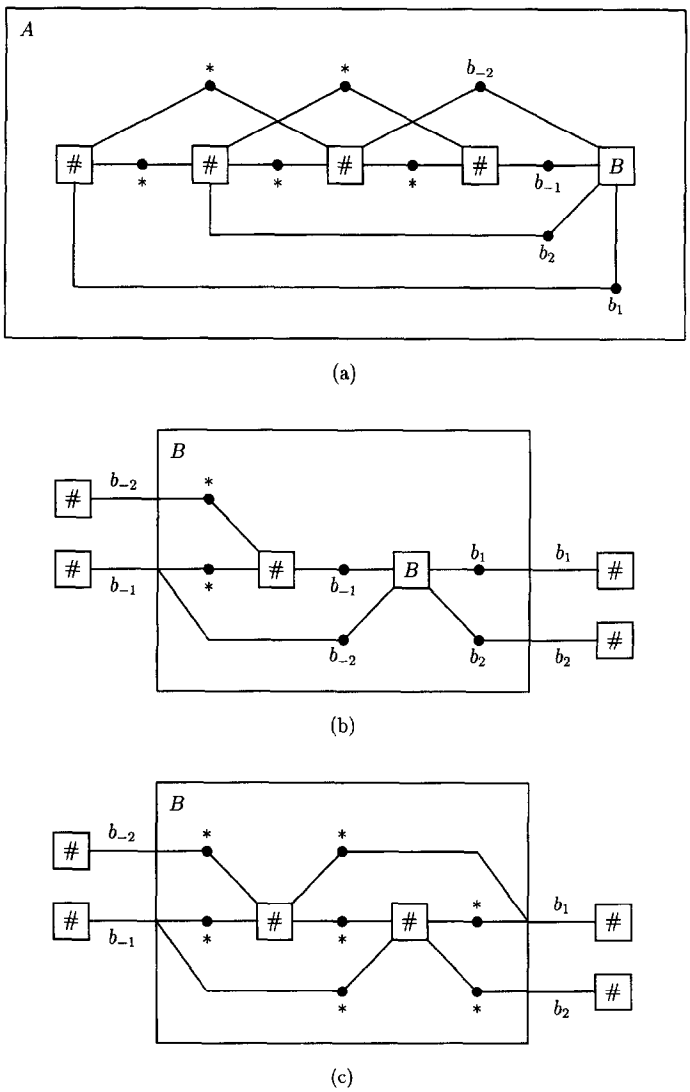


Fig. 12. The productions of G generating dual (CB_2).

and include these five nodes to the five edges in Fig. 12(a) in a one-to-one manner. Modify the productions in Figs. 12(b) and (c) so that each newly introduced edge is connected to this $\bar{\#}$-labeled edge via a $\bar{*}$-labeled node. Then, $L(G'')$ is NP-complete and consists of connected and degree-bounded hypergraphs only. □

Lemma 6.2. *For every context-free (linear) language L, we can construct an S-HRNCE (Lin-HRNCE) grammar G such that $L(G) = o\text{-}chain(L)$.*

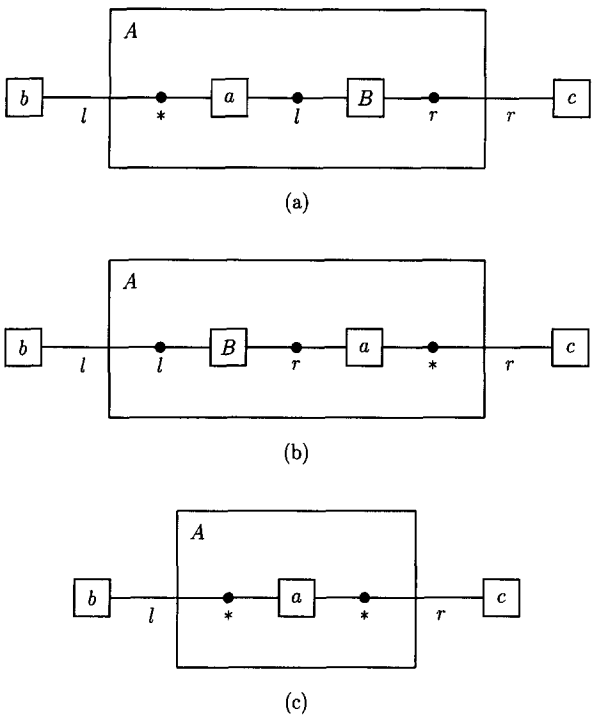


Fig. 13. The productions to simulate linear grammars.

Proof. Let $G_1 = (\Gamma, \Omega, P, S)$ be a linear grammar in $\{A \to aB, A \to Ba, A \to a\}$-normal form, where $A, B \in \Gamma - \Omega$ and $a \in \Omega$. Construct a Lin-HRNCE grammar $\bar{G}_1 = (\Sigma, \Delta, \Gamma', \Omega', P', Z)$ such that $\Sigma = \{l, r, *\}$, $\Delta = \{*\}$, $\Gamma' = \Gamma \cup \{\bar{A} \mid A \in \Gamma - \Omega\} \cup \{\$\}$, $\Omega' = \Omega \cup \{\$\}$, $Z = \boxed{\bar{S}} \overset{r}{—} \boxed{\$}$, and P' consists of the following productions for all $b \in \Omega$ and all $c \in \Omega \cup \{\$\}$:

(1) if $A \to aB$ is a production of G_1, then the production in Fig. 13(a) and its modified version, with A replaced by $\bar{A}$ and the $*$-labeled node removed, are in P';

(2) if $A \to Ba$ is a production of G_1, then the production in Fig. 13(b) and its modified version, with A and B replaced by $\bar{A}$ and $\bar{B}$, respectively, and the l-labeled node removed, are in P'; and

(3) if $A \to a$ is a production of G_1, then the production in Fig. 13(c) and its modified version, with A replaced by $\bar{A}$ and the left $*$-labeled node removed, are in P'.

Each sentential form of $\bar{G}_1$ is an oriented chain containing exactly one nonterminal edge. This nonterminal edge is labeled by $\bar{A}$ for some $A \in \Gamma - \Omega$ if and only if it is the leftmost edge. Rewriting of $\bar{G}_1$ accurately simulates a derivation step of G_1 by using the labels l and r as orientation. It is straightforward to see that $L(\bar{G}_1) =$ o-chain$(L(G_1))$.

Similarly, starting with a context-free grammar G_2 in operator normal form (i.e., no two nonterminals are adjacent in the right-hand side of any production) [21], it is easy

to construct an S-HRNCE grammar $\bar{G}_2$ such that $L(\bar{G}_2) = \text{o-chain}(L(G_2))$ by using the same method as described above. □

Theorem 6.3. *Every Lin-HRNCE language which is connected, degree-bounded, and rank-bounded is in NLOG. There exists such a Lin-HRNCE language which is NLOG-complete.*

Proof. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be a reduced Lin1-HRNCE grammar (Theorem 5.15) generating a connected hypergraph language of degree at most d and rank at most r. To prove that $L(G) \in$ NLOG, we shall use a technique introduced in [18] for graph layout analysis, and subsequently adopted in [1, 11, 12] for efficient parsing of connected, linear or boundary eNCE languages of bounded degree.

Consider any input hypergraph $H \in HGR_{\Delta,\Omega}$. That H is connected, of degree at most d, and of rank at most r can be easily checked in log space nondeterministically. Suppose that $H \in L(G)$ and let K be any intermediate sentential form in a derivation for H in G. Let e be the nonterminal edge in K. One finds two types of terminal edges in K:

(1) a completed edge, not adjacent to e, which should exist in H in such a way that its incident nodes in K are exactly its incident nodes in H; and

(2) a partially completed edge, adjacent to e, which should also exist in H (since G has no edge-erasing property) but its incident nodes have not been completely generated.

The edges of the second type are called *boundary edges*. Let e' be any boundary edge. The nodes incident to e' but not to e in K are permanent and must match with some of the nodes incident to e' in H. The rest of the nodes incident to e' in H must be generated by e via the nodes shared by e and e'. These nodes in H, incident to e' but not generated yet in K, are called *critical nodes* of e' and are denoted by critical(e').

Consider the subhypergraph K' of K consisting of the nonterminal edge e, all its incident nodes, and all its adjacent (boundary) edges. The *window* of K, denoted by window(K), is the hypergraph obtained from K' by relabeling each boundary edge e' with $[\psi_K(e'), \text{critical}(e')]$. Note that there are only a bounded number of boundary edges and their critical nodes in each K such that $Z \Rightarrow^* K \Rightarrow^* H$ since G is simple and neighborhood-preserving and H is degree- and rank-bounded. Therefore, window(K) can be stored in log space for all such K.

Our NLOG algorithm is based on the fact that it is sufficient to keep the window of a sentential form while simulating a derivation for H in G. It is easy to calculate window(Z) nondeterministically. Starting with an arbitrary window W, the next window W' can be obtained as follows:

(1) Guess a production $\pi = (A, X, C)$ of G such that A is the nonterminal edge label in W. If X contains isolated nodes (thus, no terminal edge) then go to Step (3).

(2) Let t be the terminal edge in X. Guess an edge e in H but not in W such that $\psi_H(e) = \psi_X(t)$, which is identified with t. Verify that e has not been generated yet by checking that, in H, there exists a path e_0 (= e), $v_0, \ldots, e_{k-1}, v_{k-1}, e_k$ ($k \geqslant 1$) such that e_k is a boundary edge in W, v_{k-1} is a critical node of e_k, and there is no other boundary edge or critical node in this path.

(3) Calculate W' by applying π to W. This involves guessing which node in X not incident to the nonterminal edge (if any) corresponds to which critical node, removing each terminal edge in W with all its incidence completed as in H from the window, adding e to the window (in case where Step (2) is executed) if it is adjacent to the nonterminal edge in X, checking that no boundary edge is connected to a node which is not critical, and adjusting the labels of the boundary edges in the new window W'. Meanwhile, if the new window W' is empty, then accept H. Otherwise, let $W := W'$ and go to Step (1).

It should be clear how the algorithm works. Probably, we need only note that the verification done in Step (2) is correct because H is connected, and so, every edge in H is generated exactly once. Each window is log-space bounded as observed earlier. Therefore, the path checking in Step (2) and the calculation of W' from W in Step (3) can be clearly done in log space. Thus, this is an NLOG algorithm accepting $L(G)$. We leave the detailed implementation of this algorithm to the reader, referring to [11] and [12] for an essentially identical algorithm.

The NLOG-completeness result stated in the theorem follows from the inclusion result proved above and Lemma 6.2 since there exists an NLOG-complete linear language [40]. □

Theorem 6.4. *Every S-HRNCE language which is connected, degree-bounded, and rank-bounded is in LOGCFL. There exists such an S-HRNCE language which is LOGCFL-complete.*

Proof. Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be a reduced S-HRNCE grammar in Greibach normal form (Theorem 5.22) generating a connected hypergraph language of degree at most d and rank at most r. We show that $L(G) \in$ LOGCFL by extending the NLOG algorithm given in the proof of Theorem 6.3. (This extension is similar to the LOGCFL algorithm for connected B-eNCE languages of bounded degree given in [12].) Let $H \in HGR_{\Delta,\Omega}$ be an input hypergraph. It can be checked in log space that H is connected, of degree at most d and of rank at most r. Assume that $H \in L(G)$ and let K be any sentential form in a derivation for H in G. As in the Lin-HRNCE case, K contains two types of terminal edges: completed edges, not adjacent to any nonterminal edge, and other partially completed (or boundary) edges. The critical nodes of a boundary edge have not been generated yet and must be generated by nonterminal edges adjacent to this edge. Now, the subhypergraph of K consisting of all nonterminal edges in K, all their incident nodes, and all their adjacent boundary edges augmented with their critical nodes is called the *window* of K, denoted by window(K), following the notion in the Lin-HRNCE case.

It is not possible to store window(K) in log space since the number of nonterminal edges in K is not bounded. We shall decompose window(K) into a set of small windows of the same type as in the Lin-HRNCE case as follows. For each boundary edge, its critical nodes are nondeterministically partitioned and distributed to its

adjacent nonterminal edges. Intuitively, this means that each adjacent nonterminal edge will generate the assigned critical nodes in the future derivation. Clearly, each critical node is generated by exactly one adjacent nonterminal edge, and so, this assumption is consistent with the S-HRNCE rewriting mechanism. Now, for each nonterminal edge e in window(K), consider the small window consisting of e, all its incident nodes (none of them is shared by any other nonterminal edge since G is separated), and all its adjacent edges (each of them is incident to a node generated from e because of the neighborhood-preserving property of G) together with the critical nodes assigned to e; call it window(K, e). Our modified window, denoted by window(K, $*$), consists of these small windows, as many as the number of nonterminal edges in K, i.e., window(K, $*$) $=$ $\{\text{window}(K, e) \mid e \in E_K, \psi_K(e) \in \Gamma - \Omega\}$. Note that a boundary edge in K must be duplicated into as many copies of the number of its adjacent nonterminal edges in this process.

With this preparation, we state now our LOGCFL algorithm. It is sufficient to prove that $L(G)$ is accepted by a log-space bounded auxiliary pushdown automaton in polynomial time. (An auxiliary pushdown automaton is a Turing machine with an additional unbounded pushdown store and its space bound is imposed on the work tape only. The class of languages accepted by log-space bounded and simultaneously polynomial-time bounded auxiliary pushdown automata is identical to the class of LOGCFL languages [41]. The LOGCFL algorithm for connected B-eNCE languages of bounded degree in [12] used log-space bounded alternating Turing machines with polynomial tree size that also characterize the LOGCFL languages.) Construct an auxiliary pushdown automaton M that does the following, when given a hypergraph H in a suitably encoded form on its input tape:

(1) Nondeterministically compute window(Z, $*$) and push its elements into the pushdown store.

(2) Pop the top element, W, of the pushdown store and execute the first step of the NLOG lagorithm, that guesses a production $\pi = (A, X, C)$ applicable to W.

(3) Execute the second step of the NLOG algorithm if X contains a terminal edge t (which is identified with $e \in E_H$). This ensures that e should indeed be generated by the nonterminal edge in W since G is a separated grammar. Namely, if e were any other terminal edge in H, the only way it can be connected to a boundary edge in W is via a path not containing any critical node in W.

(4) Execute the third step of the NLOG algorithm to obtain the new window W'. If W' is not empty, then decompose it into small windows, as discussed earlier, and push them into the pushdown store. Now, if the pushdown store is empty, then accept H. Otherwise, go to Step (2).

A small window containing one nonterminal edge can be stored in log space as observed in the Lin-HRNCE case. Such a window creates a new window containing a bounded number of nonterminal edges in Step (4). Thus, this new window and its modified version (a set of small windows) can be clearly stored in log space. As G is separated, the critical nodes of a boundary edge can be handled separately by its adjacent nonterminal edges, and so, the decomposition of a window into small windows

is consistent with the S-HRNCE rewriting mechanism. As G is confluent, these small windows can be added to the pushdown store in any order for later executions. A boundary edge may exist in many small windows, stored in the pushdown store of M, but it is eventually removed from the window when its last set of critical nodes are generated from a small window. As H is connected, each edge in H is certainly identified exactly once. This observation should convince the reader that M accepts $L(G)$ and M is log-space bounded. Steps (2)-(4) of the above algorithm are iterated no more than $\#V_H + \#E_H$ times (since G is in Greibach normal form), and each such iteration clearly takes polynomial time. Therefore, M runs in polynomial time. It follows that $L(G) \in$ LOGCFL.

The LOGCFL-completeness result stated in the theorem follows from the inclusion result proved above and Lemma 6.2 since there exists a LOGCFL-complete context-free language (i.e., the hardest context-free language of Greibach) [29]. □

Major decision problems such as equivalence are undecidable for linear grammars [22]. Therefore, a direct consequence of Lemma 6.2 is that these decision problems are also undecidable for Lin-HRNCE grammars. Among others, we shall state the following:

Theorem 6.5. *It is undecidable whether or not* $L(G_1)=L(G_2)$ *and* $L(G_1)\cap L(G_2)=\emptyset$ *for Lin-HRNCE grammars* G_1 *and* G_2.

7. An HRNCE language hierarchy

For each grammar type X, let $\mathscr{L}(X)$ denote the family of all languages generated by X grammars. We shall prove that $\mathscr{L}$(Lin-HRNCE) $\subsetneq$ $\mathscr{L}$(S-HRNCE) $\subsetneq$ $\mathscr{L}$(N-HRNCE) $\subsetneq$ $\mathscr{L}$(HRNCE). As inclusion relations in this hierarchy are obviously true, we shall only prove their properness.

Theorem 7.1. $\mathscr{L}$(*Lin-HRNCE*) $\subsetneq$ $\mathscr{L}$(*S-HRNCE*).

Proof. Consider the language L of all duals of binary trees introduced in Example 3.5, which is generated by an S-HRNCE grammar. We show that L is not a Lin-HRNCE language, by using a method similar to the one for separating Lin-eNCE and A-eNCE classes [11]. Namely, we use the fact that a complete binary tree of depth $2k$ has cutwidth $k+1$ for all $k \geqslant 1$ [34]. This implies that dual(L) is not of bounded cutwidth. Thus, it is sufficient to prove that, for every Lin-HRNCE language of bounded degree and bounded rank, its dual is of bounded cutwidth.

We shall define the cutwidth of a hypergraph as follows. Let H be a hypergraph. A *linear layout of* H is a bijection l: $V_H \cup E_H \to \{1, 2, \ldots, n\}$, where $n = \#(V_H \cup E_H)$. For $1 \leqslant i \leqslant n$, the *i-th cut of* H *under* l is $\mathrm{cut}_i(H, l) = \#\{(v, e) \in V_H \times E_H \mid v \in \iota_H(e)$ and either $l(v) \leqslant i$ and $l(e) > i$ or $l(v) > i$ and $l(e) \leqslant i\}$. (Note that $\mathrm{cut}_n(H, l) = 0$.) The *cutwidth of* H *under* l is $\mathrm{cw}(H, l) = \max\{\mathrm{cut}_i(H, l) \mid 1 \leqslant i$

$\leqslant n\}$ and the *cutwidth of* H is $\mathrm{cw}(H) = \min\{\mathrm{cw}(H, l) \mid l \text{ is a linear layout of } H\}$. A hypergraph language is of *bounded cutwidth* if each of its members is of cutwidth at most k, for some fixed $k \geqslant 1$. Note that the cutwidth of a hypergraph language is identical to the cutwidth of its dual and, for every graph language, its cutwidth as defined here is no smaller than its cutwidth as usually defined for graphs, i.e., the one counting the number of edges that cross over the boundary between two adjacent nodes in the layout, with nodes only laid out on a line (see, e.g., [34]).

Let $G = (\Sigma, \Delta, \Gamma, \Omega, P, Z)$ be an arbitrary reduced Lin1-HRNCE grammar (Theorem 5.15) generating a hypergraph language of bounded degree and bounded rank. Assume without loss of generality that Z consists of a single nonterminal edge without nodes. Let $d = \max\{\deg(H) \mid H \in L(G)\}$ and $r = \max\{\mathrm{rank}(H) \mid H \in L(G)\}$. Let $N = dr \cdot \#\Omega \cdot \#\Sigma + r$. Let H be any hypergraph in $L(G)$. We claim that $\mathrm{cw}(H) \leqslant N$, or equivalently, $\mathrm{cw}(\mathrm{dual}(H)) \leqslant N$.

Let $\#E_H = n$ and consider a derivation for H in G, of the form

$$\begin{aligned} D:\ Z &\Rightarrow^* K_1 \Rightarrow_{(x_1,\pi_1)} H_1 \\ &\Rightarrow^* K_2 \Rightarrow_{(x_2,\pi_2)} H_2 \\ &\cdots \\ &\Rightarrow^* K_n \Rightarrow_{(x_n,\pi_n)} H_n \Rightarrow^* H \end{aligned}$$

such that the right-hand side of π_i contains a terminal edge e_i, $1 \leqslant i \leqslant n$. Then, $E_H = \{e_1, e_2, \ldots, e_n\}$. Let $I = \{u_1, u_2, \ldots, u_m\}$ be the set of isolated nodes in H. Identify the nodes incident only to the terminal edge e_i in the right-hand side of π_i with $v_{i1}, v_{i2}, \ldots, v_{i,t_i}$, $1 \leqslant i \leqslant n$. Then, certainly $0 \leqslant t_i \leqslant r$ for each i and $V_H = \bigcup_{1 \leqslant i \leqslant n}\{v_{ij} | 1 \leqslant j \leqslant t_i\} \cup I$. Consider the linear layout l_D of H defined by the following sequence: e_1, v_{11}, $\ldots$, v_{1,t_1}, $\ldots$, e_n, v_{n_1}, $\ldots$, $v_{n,t_n}, u_1, \ldots, u_m$. For each $i \in \{1, 2, \ldots, n\}$, consider $\mathrm{cut}_{\delta(i)}(H, l_D)$, where $\delta(i) = \sum_{1 \leqslant k \leqslant i-1} (t_k + 1) + 1$, i.e., the cut located immediately to the right of the edge e_i in l_D. Each incidence between a node v and an edge e ($\neq e_i$) in H that contributes to $\mathrm{cut}_{\delta(i)}(H, l_D)$ has the property that $l_D(v) > \delta(i)$ and $l_D(e) < \delta(i)$ because of the HRNCE rewriting mechanism. This implies that $\mathrm{cut}_{\delta(i)}(H, l_D) \geqslant \mathrm{cut}_{\delta(i)+j}(H, l_D)$ for all $i \in \{1, 2, \ldots, n\}$ and all $j \in \{1, 2, \ldots, t_i\}$. Thus, $\mathrm{cw}(H, l_D) = \max\{\mathrm{cut}_{\delta(i)}(H, l_D) \mid 1 \leqslant i \leqslant n\}$.

Now, assume to the contrary that $\mathrm{cut}_{\delta(i)}(H, l_D) > N$ for some i. Call the incidences in H that contribute to $\mathrm{cut}_{\delta(i)}(H, l_D)$ but are not associated with the edge e_i *critical lines*. The edges associated with the critical lines are called *left-critical edges* (they are located to the left of e_i in l_D) and the nodes associated with the critical lines are called *right-critical nodes* (they are located to the right of e_i in l_D). All left-critical edges are adjacent to the nonterminal edge x_i in K_i. As e_i is incident to at most r nodes and each left-critical edge is incident to at most r right-critical nodes, there must be more than $(N-r)/r = d \cdot \#\Omega \cdot \#\Sigma$ left-critical edges, all adjacent to x_i in K_i. For each left-critical edge e, choose an *active node* in K_i, which is shared by e and x_i and is used later to establish the connection between e and a right-critical node in l_D. As there are $\#\Omega$ terminal edge labels and at most $\#\Sigma$ active nodes (because G is simple), there must be

more than d left-critical edges labeled by the same symbol that share the same active node with x_i. This means that some right-critical node located to the right of e_i in l_D is incident to more than d edges because of the HRNCE embedding mechanism. This is a contradiction to our assumption that $\deg(H) \leqslant d$. It follows that dual($L(G)$) has a bounded cutwidth, and this proves the theorem. □

Theorem 7.2. $\mathscr{L}(S\text{-}HRNCE) \subsetneq \mathscr{L}(N\text{-}HRNCE)$.

Proof. Consider the word language $L = \{a^{2^n} \mid n \geqslant 1\}$. L can be certainly accepted by a linear-bounded automaton. Therefore, by Lemma 4.3, there exists an N-HRNCE grammar G such that $L(G)$ = m-o-chain(L). We shall prove, however, that there is no S-HRNCE grammar G' such that $L(G')$ = m-o-chain(L), by using a pumping argument.

Suppose to the contrary that $L(G')$ = m-o-chain(L) for some S-HRNCE grammar $G' = (\Sigma, \Lambda, \Gamma, \Omega, P, Z)$. Assume without loss of generality that G' is a reduced S-HRNCE grammar in Chomsky normal form (Theorem 5.13) and that Z contains one edge, which is nonterminal. Let $N = 2^{\#(\Gamma - \Omega)+2}$ and let H be any hypergraph in $L(G')$ with at least N edges. Then there exists a derivation D for H in G' that can be written as $Z \Rightarrow^* H_1 \Rightarrow^* H_2 \Rightarrow^* H$, where (1) E_{H_1} contains at least one terminal edge; (2) all edges in $E_{H_2} - E_{H_1}$ are generated starting from one nonterminal edge $e \in E_{H_1}$; and (3) $E_{H_2} - E_{H_1}$ contains at least one terminal edge and exactly one nonterminal edge e' which is labeled by $\psi_{H_1}(e)$. This can be easily observed by drawing a tree structure showing edge rewritings, similar to a derivation tree for a context-free grammar, and by using the fact that G' is an edge-erasing free confluent grammar. Let t be the number of terminal edges in $E_{H_2} - E_{H_1}$. Then, $1 \leqslant t \leqslant \#E_H - 2$. Now, the derivation D' obtained from D by applying again the sequence of productions used to obtain H_2 from H_1 to e' and by using other productions as used in D derives a hypergraph in $L(G')$. Consider, in particular, the case where H = m-o-chain(a^{2^N}) and let H' be the corresponding hypergraph generated by D'. Then, $H' \in L(G')$ and $\#E_{H'} = \#E_H + t$. However, this implies that $2^N + 2 \leqslant \#E_{H'} \leqslant 2^{N+1}$ since $\#E_H = 2^N + 1$. This is a contradiction to $H' \in$ m-o-chain(L). It follows that there is no S-HRNCE grammar generating m-o-chain(L), and this completes the proof. □

Theorem 7.3. $\mathscr{L}(N\text{-}HRNCE) \subsetneq \mathscr{L}(HRNCE)$.

Proof. Follows from Theorems 4.1 and 4.4 since PSPACE is properly included in the class of r.e. languages. □

8. Concluding Remarks

HRNCE grammars are structurally simple, as easy to use as NLC/eNCE grammars, yet descriptively powerful; they generate all r.e.languages and can simulate string grammars and automata in a very straightforward way. They are flexible in that subclasses with nice features, such as S-HRNCE and Lin-HRNCE grammars, can be well defined. Known techniques and results can also be well extended to these classes without

difficulty. Thus, HRNCE grammars are a sound generative device in this sense. There are many properties of HRNCE grammars and their languages that remain to be investigated. We shall list a few of them below.

(1) There are other language-theoretical properties of HRNCE languages not covered in this paper, e.g., their combinatorial and closure properties, as studied for B-NLC grammars in [38, 39].

(2) There are other subclasses and extensions of HRNCE grammars, e.g., the neighborhood-uniform and apex subclasses and the directed version that can be defined along their eNCE counterparts [13, 16, 27].

(3) There are other language-describing mechanisms for HRNCE grammars, e.g., via squeezing with graph languages or other interesting hypergraph languages, as studied for NLC languages in [32].

(4) There may be other tight relations between HRNCE subclasses and traditional complexity classes not discussed in this paper.

(5) It is necessary to compare HRNCE grammars with other (hyper)graph grammars, such as eNCE, CFHG and HH grammars, for a unified study of graph-generating systems.

References

[1] IJ.J. Aalbersberg, J. Engelfriet, G. Rozenberg, The complexity of regular DNLC graph languages, J. Comput. System Sci. 40 (1990) 376–404.

[2] M. Bauderon, B. Courcelle, Graph expressions and graph rewritings, Math. Systems Theory 20 (1987), 83–127.

[3] V. Claus, H. Ehrig, G. Rozenberg (Eds.), Proc. 1st Internat. Workshop on Graph Grammars and Their Application to Computer Science and Biology, Lecture Notes in Computer Science, vol. 73, Springer, Berlin, 1979.

[4] J. Cuny, H. Ehrig, G. Engels (Eds.), Proc. 5th Internat. Workshop on Graph Grammars and their Application to Computer Science, Lecture Notes in Computer Science, vol. 1073, Springer, Berlin, 1996.

[5] B. Courcelle, An axiomatic definition of context-free rewriting and its application to NLC graph grammars, Theoret. Comput. Sci. 55 (1987) 141–181.

[6] B. Courcelle, J. Engelfriet, G. Rozenberg, Handle-rewriting hypergraph grammars, J. Comput. System Sci. 46 (1993) 218–270.

[7] H. Ehrig, H.-J. Kreowski, G. Rozenberg, A. Rosenfeld (Eds.), Proc. 4th Internat. Workshop on Graph Grammars and their Application to Computer Science, Lecture Notes in Computer Science, vol. 532, Springer, Berlin, 1991.

[8] H. Ehrig, M. Nagl, G. Rozenberg (Eds.), Proc. 2nd Internat. Workshop on Graph Grammars and their Application to Computer Science, Lecture Notes in Computer Science, vol. 153, Springer, Berlin, 1983.

[9] H. Ehrig, M. Nagl, G. Rozenberg, A. Rosenfeld (Eds.), Proc. 3rd Internat. Workshop on Graph Grammars and Their Application to Computer Science, Lecture Notes in Computer Science, vol. 291, Springer, Berlin, 1987.

[10] J. Engelfriet, L. Heyker, Hypergraph languages of bounded degree, J. Comput. System Sci. 48 (1994) 58–89.

[11] J. Engelfriet, G. Leih, Linear graph grammars: power and complexity, Inform. and Comput. 81 (1989) 88–121.

[12] J. Engelfriet, G. Leih, Complexity of boundary graph languages, Theoret. Inform. Appl. 24 (1990) 267–274.

[13] J. Engelfriet, G. Leih, G. Rozenberg, Apex graph grammars and attribute grammars, Acta Inform. 25 (1988) 537–571.

[14] J. Engelfriet, G. Leih, G. Rozenberg, Nonterminal separation in graph grammars, Theoret. Comput. Sci. 82 (1991) 95–111.
[15] J. Engelfriet, G. Leih, E. Welzl, Boundary graph grammars with dynamic edge relabeling, J. Comput. System Sci. 40 (1990) 307–345.
[16] J. Engelfriet, G. Rozenberg, A comparison of boundary graph grammars and context-free hypergraph grammars, Inform. and Comput. 84 (1990) 163–206.
[17] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.
[18] E. Gurari, I.H. Sudborough, Improved dynamic programming algorithms for bandwidth minimization and the min cut linear arrangement problem, J. Algorithms 5 (1984) 531–546.
[19] A. Habel, H.-J. Kreowski, Some structural aspects of hypergraph languages generated by hyperedge replacement, Proc. STACS 87, Lecture Notes in Computer Science, vol. 247, Springer, Berlin, 1987, pp. 207–219.
[20] A. Habel, H.-J. Kreowski, W. Vogler, Metatheorems for decision problems on hyperedge replacement graph languages, Acta Inform. 26 (1989) 657–677.
[21] M.A. Harrison, Introduction to Formal Language Theory. Addison-Wesley, Reading, MA., 1978.
[22] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation Addison-Wesley, Reading, MA., 1979.
[23] D. Janssens, G. Rozenberg, On the structure of node-label-controlled graph languages, Inform. Sci. 20 (1980) 191–216.
[24] D. Janssens, G. Rozenberg, Restrictions, extensions, and variations of NLC grammars, Inform. Sci. 20 (1980) 217–244.
[25] D. Janssens, G. Rozenberg, Decision problems for node label controlled graph grammars, J. Comput. System Sci. 22 (1981) 144–177.
[26] D. Janssens, G. Rozenberg, Graph grammars with neighbourhood-controlled embedding, Theoret. Comput. Sci. 21 (1982) 55–74.
[27] D. Janssens, G. Rozenberg, Neighborhood-uniform NLC grammars, Comput. Vis. Graphics Image Process. 35 (1986) 131–151.
[28] D.S. Johnson, The NP-completeness column: an ongoing guide, J. Algorithms 3 (1982) 288–300.
[29] D.S. Johnson, A catalog of complexity classes, in: J. van Leeuwen, (Ed.), Handbook of Theoretical Computer Science, vol. A, Elsevier, Amsterdam, 1990, Chap. 2.
[30] C. Kim, A hierarchy of eNCE families of graph languages, Theoret. Comput. Sci. 186 (1997) 157–169.
[31] C. Kim, D.H. Lee, Separating k-separated eNCE graph languages, Theoret. Comput. Sci. 120 (1993) 247–259.
[32] C. Kim, D.H. Lee, Node replacement graph languages squeezed with chains, trees, and forests, Inform. and Comput. 117 (1995) 63–77.
[33] C. Lautemann, The complexity of graph languages generated by hyperedge replacement, Acta Inform. 27 (1990) 399–421.
[34] T. Lengauer, Upper and lower bounds on the complexity of the min-cut linear arrangement problem on trees, SIAM J. Algebraic Discrete Meth. 3 (1982) 99–113.
[35] M.G. Main, G. Rozenberg, Handle NLC grammars and r.e. languages, J. Comput. System Sci. 35 (1987) 192–205.
[36] M.G. Main, G. Rozenberg, Edge-label controlled graph grammars, J. Comput. System Sci. 40 (1990) 188–228.
[37] G. Rozenberg, E. Welzl, Boundary NLC graph grammars-Basic definitions, normal forms, and complexity, Inform. and Control 69 (1986) 136–167.
[38] G. Rozenberg, E. Welzl, Graph theoretic closure properties of the family of boundary NLC graph languages, Acta Inform. 23 (1986) 289–309.
[39] G. Rozenberg, E. Welzl, Combinatorial properties of boundary NLC graph languages, Discrete Appl. Math. 16 (1987) 59–73.
[40] I.H. Sudborough, A note on tape-bounded complexity classes and linear context-free languages, J. ACM 22 (1975) 499–500.
[41] I.H. Sudborough, On the complexity of deterministic context-free languages, J. ACM 25 (1978) 405–414.
[42] Gy. Turán, On the complexity of graph grammars, Acta Cybernet. 6 (1983) 271–280.