Parallel Hyperedge Replacement String Languages

Graham Campbell[*]

School of Mathematics, Statistics and Physics, Newcastle University
Newcastle upon Tyne, United Kingdom
`g.j.campbell2@newcastle.ac.uk`

There are many open questions surrounding the characterisation of groups with context-sensitive word problem. Only in 2018 was it shown that all finitely generated virtually Abelian groups have multiple context-free word problems, and it is a long-standing open question as to where to place the word problems of hyperbolic groups in the formal language hierarchy. In this paper, we introduce a new language class called the parallel hyperedge replacement string languages, show that it contains all multiple context-free and ET0L languages, and lay down the foundations for future work that may be able to place the word problems of many hyperbolic groups in this class.

1 Introduction

In general, the word problem is the question that asks if two strings (words) represent the same element in some structure. In the case of groups, this is the equivalent to asking if a given string represents the identity element, since if u, v are strings, then they are equal in a group if and only if uv^{-1} represents the identity in the group. Thus, given a presentation $\langle X \mid R \rangle$ for a group G, the word problem is equivalent to the membership problem for the string language $\mathrm{WP}_X(G) = \{w \in (X \cup X^{-1})^* \mid w =_G 1_G\}$. Viewing things geometrically, the word problem of a group can be identified with the set of loops based at the identity in the Cayley graph. A partial sketch of Cayley graphs of $\mathbb{Z}^2$ and F_2 is provided in Figure 1.
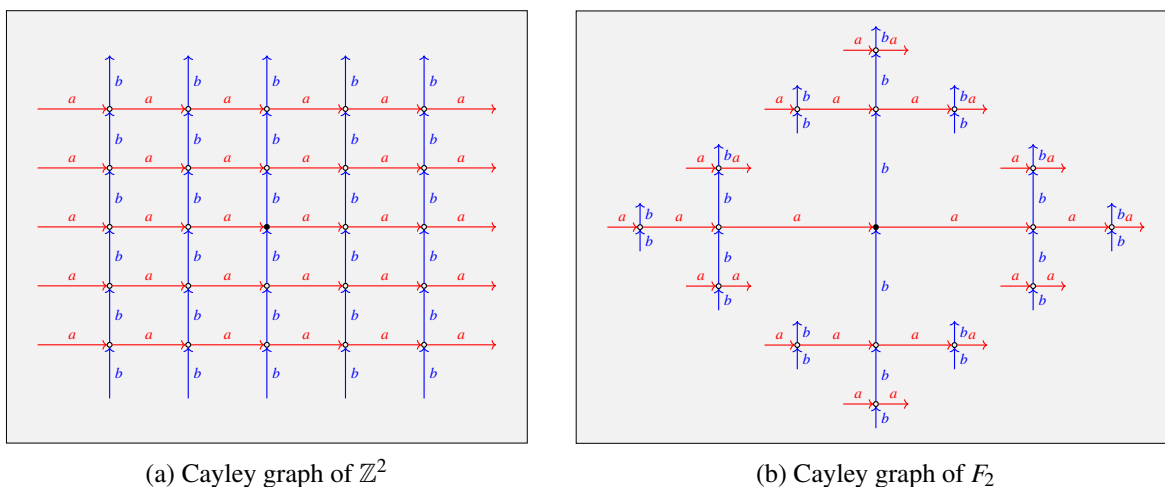

(a) Cayley graph of $\mathbb{Z}^2$ (b) Cayley graph of F_2

Figure 1: Example Cayley graphs

A natural question to ask is how hard the word problem is, in general, and for specific families of groups. Unsurprisingly, both the universal word problem and the word problem are undecidable in general, even for finite presentations [25]. It is well known that a presentation defines a finite group if and only if it admits a regular word problem [3], and defines a finitely generated virtually free group if and only if it admits a deterministic context-free word problem if and only if it admits a context-free word problem [22]. The multiple context-free (MCF) languages sit strictly in between the context-free and context-sensitive languages [29]. In 2015, a major breakthrough of Salvati was published, showing that the word problem of $\mathbb{Z}^2$ is an MCF language [28], and in 2018, Ho extended this result to all finitely generated virtually Abelian groups [16]. This is interesting since the MCF languages are exactly the string languages generated by hyperedge replacement grammars [10, 31]. It remains an open problem as to which other families of groups admit MCF word problems, however, we do at least know that the fundamental group of a hyperbolic three-manifold does not admit an MCF word problem [11].

There are of course, lots of other well-behaved language classes sitting in between the context-free and context-sensitive classes, such as the indexed languages [1] or the subclass of ET0L languages [27]. It is not known if there are any groups with indexed word problems, other than the virtually free groups, but it is known that a particular subclass of the indexed languages, not contained in ET0L, only contains word problems of virtually free groups [12]. We also do not know if any hyperbolic groups have ET0L word problems [6] (other than the virtually free groups), such as the fundamental group of the double torus. It is conjectured that every ET0L group language is admitted by a virtually free group [6]. Figure 2 shows the (group) language hierarchy, where necessarily strict inclusion uses a solid line, and $\mathcal{GP}$ denotes the class of all group languages (the class of word problems of all finitely generated groups).

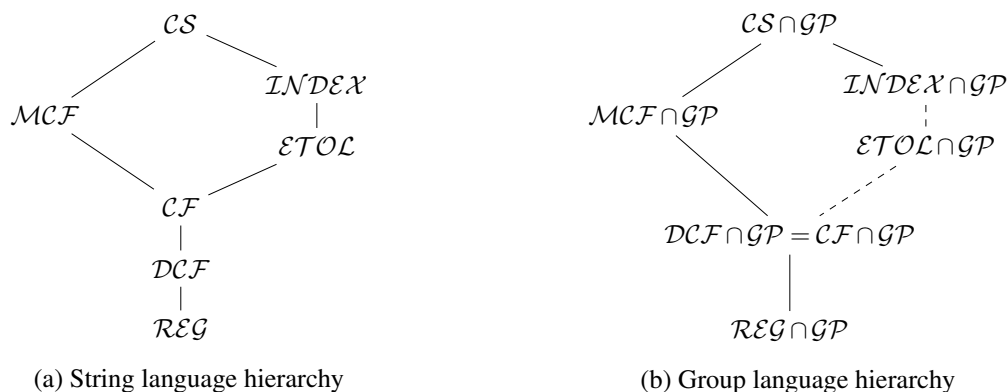(a) String language hierarchy   (b) Group language hierarchy

Figure 2: Previously known formal language hierarchies

In this paper, we define and study a new string language class, combining ideas from both ET0L and hyperedge replacement grammars. We call our new class the parallel hyperedge replacement string (PHRS) languages, and show that the class strictly contains both the classes of MCF and ET0L languages, that it is a substitution and iterated substitution closed full abstract family of languages, and that PHRS group languages are closed under free product. While parallel hyperedge replacement has been considered before, most notably by Habel and Kreowski (separately) [13, 18, 19], the work is not extensive and does not consider repetition-freeness, rational control, or string generational power. Our long term goal is to place the word problem for as many hyperbolic groups as possible in the PHRS class. Knowledge of (geometric) group theory and word problems is not required to read and understand this paper - it is purely motivational!

Figure 3 summarises how the PHRS and repetition-free PHRS languages fit into the string language hierarchy and also how we conjecture the hierarchy collapses when we restrict to group languages.
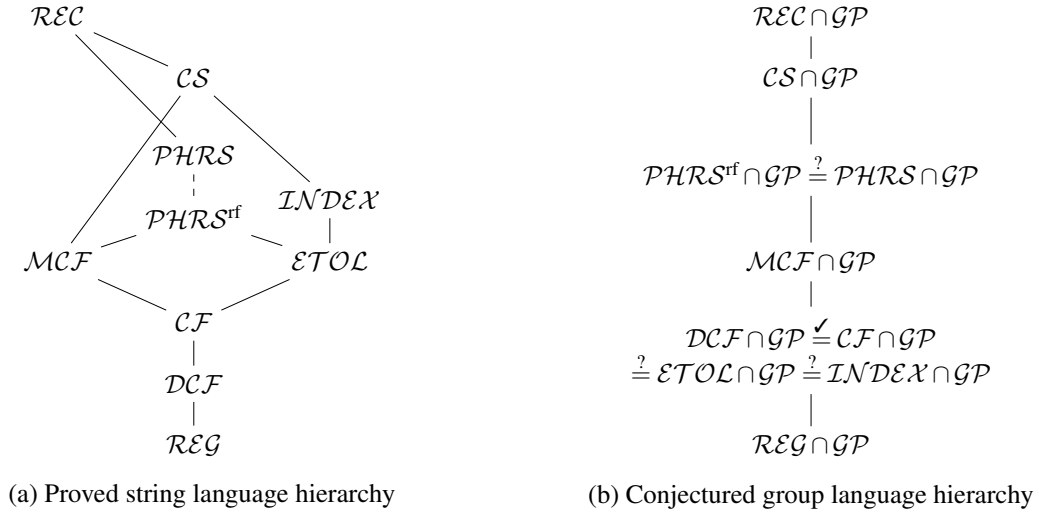

(a) Proved string language hierarchy   (b) Conjectured group language hierarchy

Figure 3: New formal language hierarchies

2   Preliminaries

By $\mathbb{N}$ we mean the natural numbers including zero, by $\underline{n}$ we mean $\{1,\ldots,n\}$, and $\oplus$ denotes relational override. In this paper, all alphabets and sequences will be finite. Formally, a sequence on a set S is a function $\sigma : \underline{n} \to S$. We view strings as sequences on an alphabet and denote the set of all sequences on S by S^*. By a coding we mean a letter-to-letter homomorphism of free monoids, and by a weak encoding we mean a coding which possibly sends letters to the empty string. In this section, we define hyperedge replacement and ET0L grammars, and recall some important known results.

2.1   Hyperedge Replacement

This subsection is mostly based on [13, 8]. By a signature we mean a pair $\mathcal{C} = (\Sigma, \text{type})$ where Σ is some alphabet, called the label set, and $\text{type} : \Sigma \to \mathbb{N}$ is a typing function which assigns to each label an arity called its type. We usually will assume some arbitrary but fixed signature $\mathcal{C} = (\Sigma, \text{type})$.

A hypergraph is a tuple $H = (V_H, E_H, \text{att}_H, \text{lab}_H, \text{ext}_H)$ where V_H is a finite set of nodes, E_H is a finite set of hyperedges, $\text{att}_H : E_H \to V_H^*$ is the attachment function, $\text{lab}_H : E_H \to \Sigma$ is the labelling function, and $\text{ext}_H \in V_H^*$ are the external nodes, such that labelling is compatible with typing $(\text{type} \circ \text{lab}_H = |\cdot| \circ \text{att}_H)$. In an abuse of notation, we write $\text{type}(H) = |\text{ext}_H|$ for the type of H, and define $\text{type}_H : E_H \to \mathbb{N}$ by $\text{type}_H = \text{type} \circ \text{lab}_H$ for the type of a hyperedge. For any hyperedge $e \in E_H$, whenever $m = \text{type}_H(e)$ we call e a type m hyperedge, and call e proper whenever $\text{att}_H(e)$ is injective (contains no repeated nodes). Call H proper if every $e \in E_H$ is proper and repetition-free if ext_H is injective. The class of all hypergraphs (repetition-free hypergraphs) over $\mathcal{C}$ is denoted $\mathcal{H}_{\mathcal{C}}$ ($\mathcal{H}_{\mathcal{C}}^{\text{rf}}$). We say two hypergraphs $G, H \in \mathcal{H}_{\mathcal{C}}$ are isomorphic $(G \cong H)$ if there is a pair of bijective functions $(g_V : V_G \to V_H, g_E : E_G \to E_H)$ such that $\text{att}_H \circ g_E = g_V^* \circ \text{att}_G$, $\text{lab}_H \circ g_E = \text{lab}_G$, and $g_V \circ \text{ext}_G = \text{ext}_H$.

Given a string $w \in \Sigma^*$ of length n, its string graph is $w^{\bullet} = (\{v_0, \dots, v_n\}, \{e_1, \dots, e_n\}), \text{att}, \text{lab}, v_0 v_n)$ where $\text{att}(e_i) = v_{i-1} v_i$ and $\text{lab}(e_i) = w(i)$ for all $i \in \underline{n}$ (Figure 4(a)). If $H \cong w^{\bullet}$ for some $w \in \Sigma^*$, we call H a string graph representing w. We also use the superscript bullet to denote the handle of a label. If $X \in \Sigma$ is of type n, then the handle of X is the hypergraph $X^{\bullet} = (\{v_1, \dots, v_n\}, \{e\}, \text{att}, \text{lab}, v_1 \cdots v_n)$ where $\text{att}(e) = v_1 \cdots v_n$ and $\text{lab}(e) = X$ (Figure 4(b)). These two definitions coincide for a type 2 label, considered either as a string of length 1 or as a label, so there can be no confusion.
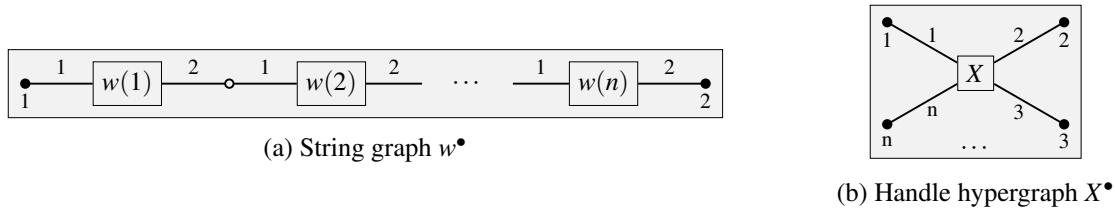

(a) String graph $w^{\bullet}$

(b) Handle hypergraph $X^{\bullet}$

Figure 4: Example hypergraphs

Let $H \in \mathcal{H}_C$ be a hypergraph and $B \subseteq E_H$ be a selection of hyperedges. Then $\sigma : B \to \mathcal{H}_C$ is called a replacement function if $\text{type} \circ \sigma = \text{type}_H|_B$. The replacement of B in H using σ is denoted by $H[\sigma]$, and is the hypergraph obtained from H by removing B from E_H, disjointly adding the nodes and hyperedges of $\sigma(e)$, for each $e \in B$, and identifying the i-th external node of $\sigma(e)$ with the i-th attachment node of e, for each $e \in B$ and $i \in \text{type}_H(e)$. The external nodes of $H[\sigma]$ remain exactly those of H and all hyperedges keep their original attachments and labels. $H[\sigma]$ exists exactly when $\sigma : B \to \mathcal{H}_C$ is a replacement function, and is unique up to isomorphism. If $B = \{e_1, \dots, e_n\}$ and $R_i = \sigma(e_i)$ for all $i \in \underline{n}$, then we write $H[e_1/R_1, \dots, e_n/R_n]$ in place of $H[\sigma]$. Figure 5 shows an example replacement.
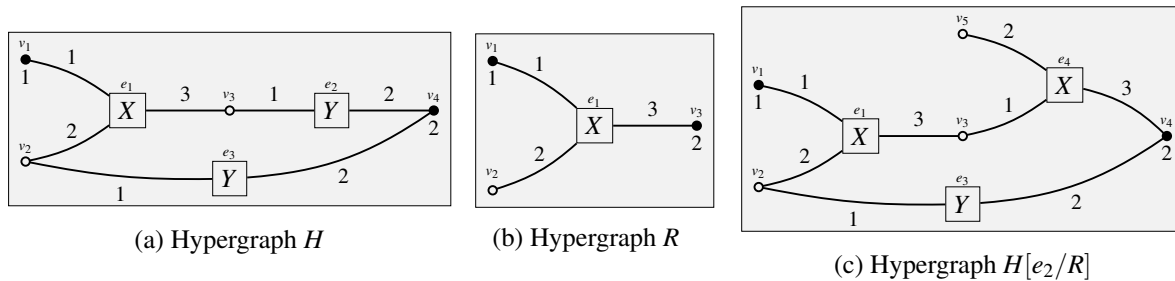

(a) Hypergraph H

(b) Hypergraph R

(c) Hypergraph $H[e_2/R]$

Figure 5: Example hyperedge replacement

Let $N \subseteq \Sigma$ be a set of non-terminals. A type n rule over N is a pair (L, R) with $L \in N$, $R \in \mathcal{H}_C$, and $\text{type}(L) = \text{type}(R) = n$. Call a rule (L, R) repetition-free (proper) if R is repetition-free (proper). Given a hypergraph $H \in \mathcal{H}_C$ and a set of rules $\mathcal{R}$, if $e \in E_H$ and $(\text{lab}_H(e), R) \in \mathcal{R}$, then we say that H directly derives $H' \cong H[e/R]$, and write $H \Rightarrow_{\mathcal{R}} H'$. For a given hyperedge e and choice of rule, H' is unique up to isomorphism. Clearly $\Rightarrow_{\mathcal{R}}$ is a binary relation on $\mathcal{H}_C$. We say $H \in \mathcal{H}_C$ derives H' if there is a sequence $H \Rightarrow_{\mathcal{R}} H_1 \Rightarrow_{\mathcal{R}} \cdots \Rightarrow_{\mathcal{R}} H_k = H'$ for some $k \geq 1$ or $H \cong H'$. We write $H \Rightarrow_{\mathcal{R}}^k H'$ or $H \Rightarrow_{\mathcal{R}}^* H'$. Clearly, (direct) derivations cannot delete nodes, and (direct) derivations made using repetition-free rules cannot merge nodes. We have the following result for repetition-free rules:

Theorem 2.1 (HR Context-Freeness [13]). *Let $\mathcal{R}$ be a set of repetition-free rules over N, $H \in \mathcal{H}_C$, $X \in N$, and $k \in \mathbb{N}$. Then there is a derivation $X^{\bullet} \Rightarrow^{k+1} H$ if and only if there is a rule $(X, R) \in \mathcal{R}$ and a mapping $\sigma : \text{lab}_R^{-1}(N) \to \mathcal{H}_C$ such that $H = R[\sigma]$, $\forall e \in \text{lab}_R^{-1}(N), \text{lab}_R(e)^{\bullet} \Rightarrow^{k(e)} \sigma(e)$, and $\sum_{e \in \text{lab}_R^{-1}(N)} k(e) = k$.*

A hyperedge replacement grammar of order k (k-HR grammar) is a tuple $\mathcal{G} = (\mathcal{C}, N, S, \mathcal{R})$ where $\mathcal{C} = (\Sigma, \text{type})$ is a signature, $N \subseteq \Sigma$ is the set of non-terminal labels, $S \in N$ is the start symbol, and $\mathcal{R}$ is a finite set of rules over N, with $\max(\{\text{type}(r) \mid r \in \mathcal{R}\}) \leq k$. We call $\Sigma \setminus N$ the terminal labels and call $\mathcal{G}$ repetition-free (proper) if all its rules are repetition-free (proper). The language generated by $\mathcal{G}$ is $\text{L}(\mathcal{G}) = \{H \in \mathcal{H}_\mathcal{C} \mid S^\bullet \Rightarrow_\mathcal{R}^* H \text{ with } \text{lab}_H^{-1}(N) = \emptyset\} \subseteq \mathcal{H}_\mathcal{C}$. $L \subseteq \mathcal{H}_\mathcal{C}$ is called a (repetition-free) hyperedge replacement language of order k ((repetition-free) k-HR language) if there is a (repetition-free) k-HR grammar such that $\text{L}(\mathcal{G}) = L$. The class of (repetition-free) HR languages is the union of all (repetition-free) k-HR languages for $k \geq 0$. Denote these $\mathcal{HR}_k$ and $\mathcal{HR}$ ($\mathcal{HR}_k^{\text{rf}}$ and $\mathcal{HR}^{\text{rf}}$). All such languages are isomorphism-closed and homogeneous (all hypergraphs have the same type).

Theorem 2.2 (Repetition-Free HR Generational Power [10])**.** Given an HR grammar $\mathcal{G}$ over $\mathcal{C}$, one can effectively construct a repetition-free HR grammar $\mathcal{G}'$ with $\text{L}(\mathcal{G}') = \text{L}(\mathcal{G}) \cap \mathcal{H}_\mathcal{C}^{\text{rf}}$.

Theorem 2.3 (HR Linear-Growth [13])**.** Given an infinite HR language L, there exists an infinite sequence of hypergraphs in L, say $H_0, H_1, H_2, \ldots$ and constants $c, d \in \mathbb{N}$ with $c + d \geq 1$, such that for all $i \in \mathbb{N}$, $|V_{H_{i+1}}| = |V_{H_i}| + c$ and $|E_{H_{i+1}}| = |E_{H_i}| + d$.

The partial function $\text{STR} : \mathcal{H}_\mathcal{C} \rightharpoonup \Sigma^*$ sends string graphs to the strings they represent, and is undefined elsewhere. A language $L \subseteq \mathcal{H}_\mathcal{C}$ is said to be a string graph language if it only contains string graphs. Given an HR grammar $\mathcal{G}$ that generates a string graph language, we write $\text{STR}(\text{L}(\mathcal{G}))$ for the actual string language it generates. A string language $L \subseteq A^*$ is called a (repetition-free) hyperedge replacement string language of order k ((repetition-free) k-HRS language) if there is a (repetition-free) k-HR grammar $\mathcal{G}$ such that $\mathcal{G}$ generates a string graph language and $\text{STR}(\text{L}(\mathcal{G})) = L \setminus \{\varepsilon\}$. The class of (repetition-free) HRS languages is the union of all (repetition-free) k-HRS languages for $k \geq 2$.

Theorem 2.4 (HR String Generative Power)**.** The following classes are equivalent, for any $k \geq 1$:

1. $\mathcal{HRS}_{2k} = \mathcal{HRS}_{2k+1} = \mathcal{HRS}_{2k}^{\text{rf}} = \mathcal{HRS}_{2k+1}^{\text{rf}}$: string languages of (repetition-free) hyperedge replacement grammars of order $2k$ or $2k+1$;

2. $\text{OUT}(\mathcal{DTWT}_k)$: output languages of deterministic tree walking transducers of crossing number at most k (see [2]);

3. $\mathcal{LCFR}_k$: string languages of linear context-free rewriting systems of rank at most k (see [30]);

4. $\mathcal{MCF}_k$: languages of k-multiple context-free grammars (see [29]);

5. $\mathcal{RTSA}_k$: languages of k-restricted tree stack automata (see [7]).

Proof. $\mathcal{HRS}_k = \mathcal{HRS}_k^{\text{rf}}$ for all $k \geq 2$ is due to Theorem 2.2 and $\mathcal{HRS}_{2k+1}^{\text{rf}} \subseteq \text{OUT}(\mathcal{DTWT}_k) \subseteq \mathcal{HRS}_{2k}$ for all $k \geq 1$ is due to Engelfriet et al. [10], which gives us the equalities in (1) and (1) = (2). (2) = (3) is due to Weir [31], (3) = (4) is due to Seki et al. (1991) [29], and (4) = (5) is due to Denkinger [7]. □

Call a set $S \subseteq \mathbb{N}^d$ linear if it is of the form $\{p + a_1 p_1 + \cdots a_n p_n \mid a_1, \ldots a_n \in \mathbb{N}\}$ for some fixed $p, p_1, \ldots p_n \in \mathbb{N}^d$. Call S semilinear if it is a finite union of linear sets. Given $A = \{a_1, \ldots, a_d\}$ and $w \in A^*$, define $\psi_A(w) = (|w|_{a_1}, \ldots, |w|_{a_d})$ where $|w|_{a_i}$ counts the number of occurrences of a_i in w. A string language L is called semilinear if $\psi_A(L)$ is semilinear. Two string languages $L_1, L_2 \subseteq A^*$ are called letter-equivalent if $\psi_A(L_1) = \psi_A(L_2)$. In 1966, Parikh showed that a language is semilinear if and only if it is letter-equivalent to a regular language and that all context-free langauges are semilinear [26]. In 1991, Seki et al. showed that all MCF languages are also semilinear and that the classes of k-MCF languages are substitution closed full AFLs [29]. This gives us the following result:

Theorem 2.5 (HRS Closure Properties)**.** For all $k \geq 2$, $\mathcal{HRS}_k$ is a substitution closed full AFL containing only semilinear languages.

2.2 ET0L Languages

Lindenmayer systems (L systems) were introduced in 1968 by Aristid Lindenmayer. We direct the reader to [27] for a comprehensive introduction to the topic. In this paper, we are interested in the class of string languages called the ET0L languages, described by a specific type of L system.

A table over Σ is a left-total finite binary relation $T \subseteq \Sigma \times \Sigma^*$, and can be associated to a substitution σ_T such that for any $L \subseteq \Sigma^*$, we define $\sigma_T(L) = \bigcup_{w \in L} \sigma_T(w)$ and $\sigma_T(a_1 \dots a_n) = \{w_1 \dots w_n \mid (a_1, w_1), \dots, (a_n, w_n) \in T\}$. An ET0L grammar is a tuple $\mathcal{G} = (\Sigma, A, S, \mathcal{T})$ where Σ is an alphabet, $A \subseteq \Sigma$ is the terminal alphabet, $S \in \Sigma$ is the start symbol and $\mathcal{T}$ is a finite set of tables over Σ. The language generated by $\mathcal{G}$ is $L(\mathcal{G}) = \bigcup_{n \in \mathbb{N}} \{\sigma_{T_1} \cdots \sigma_{T_n}(S) \mid T_1, \dots, T_n \in \mathcal{T}\} \cap A^*$. A language $L \subseteq A^*$ is called an ET0L language if there exists an ET0L grammar $\mathcal{G}$ such that $L(\mathcal{G}) = L$. It will be convenient to think of table entries as rules and substitutions as parallel replacement. We will make this formal Subsection 3.1.

Finally, call an ET0L grammar propagating if each table is contained in $\Sigma \times \Sigma^+$ (rather than just $\Sigma \times \Sigma^*$). That is, rules have non-empty right-hand sides. The following results are useful to us:

Theorem 2.6 (Propagating ET0L Generative Power [27]). Given an ET0L grammar $\mathcal{G}$, one can effectively construct a propagating ET0L grammar $\mathcal{G}'$ such that $L(\mathcal{G}) \setminus \{\varepsilon\} = L(\mathcal{G}')$.

Theorem 2.7 (ET0L Closure Properties [27]). $\mathcal{ET0L}$ is a substitution closed full AFL.

Theorem 2.8 (MCF and ET0L Incomparable).

1. $K = \{wh(w) \mid w \in D\}$ is a 2-MCF language which is not ET0L, where $D \subseteq \Sigma^*$ is the Dyck language [9], $\bar{\Sigma}$ is a disjoint copy of Σ, and $h : \Sigma^* \to \bar{\Sigma}^*$ is defined by sending each $a \in \Sigma$ to its copy $\bar{a} \in \bar{\Sigma}$.

2. $L = \{a^{2^n} \mid n \in \mathbb{N}\}$ is an ET0L language but not MCF. Moreover, L is not semilinear.

Proof. The first part follows from Theorem 8 of [24]. For the second part, it is easy to see that $\mathcal{G} = (\{a\}, \{a\}, a, \{\{(a, aa)\}\})$ is an ET0L grammar with $L(\mathcal{G}) = L$. Recall from Subsection 2.1 that a language is semilinear if and only if it is letter-equivalent to a regular language. Since L is a language on only one symbol, it must be semilinear if and only if it is a regular language, but clearly, it is not a regular language. But all MCF languages are semilinear, so it must be the case that L is not MCF. $\square$

3 New Results

3.1 Parallel Hyperedge Replacement

We start by introducing parallel derivations and parallel hyperedge replacement grammars and languages, equivalent to those defined by Habel in Chapter VIII.3 of [13]. The most fundamental notion to us is that of a parallel direct derivation, where every hyperedge is necessarily replaced.

In order to ensure progress can always be made, we are only interested in sets of rules that are tables:

Definition 3.1 (Table). Given $\mathcal{C} = (\Sigma, \text{type})$, a table T over Σ is a finite set of rules over Σ such that for each $L \in \Sigma$ there is at least one $R \in \mathcal{H}_\mathcal{C}$ with $(L, R) \in T$. Call T repetition-free (proper) if all its rules are repetition-free (proper).

Definition 3.2 (Parallel Direct Derivation). Given $\mathcal{C} = (\Sigma, \text{type})$, $H \in \mathcal{H}_\mathcal{C}$ with $E_H = \{e_1, \dots e_n\}$, and T a table over Σ, if for each $e_i \in E_H$, there is a $R_i \in \mathcal{H}_\mathcal{C}$ such that $(\text{lab}_H(e_i), R_i) \in T$, then we say that H parallelly directly derives $H' \cong H[e_1/R_1, \dots, e_n/R_n]$, and write $H \Rightarrow_T H'$.

Definition 3.3 (Parallel Derivation). Given $C = (\Sigma, \text{type})$, $H, H' \in \mathcal{H}_C$, and a finite set of tables $\mathcal{T} = \{T_i \mid i \in I\}$ over Σ indexed by I, we say H parallelly derives H' if there is a sequence of parallel direct derivations $H \Rightarrow_{T_{i_1}} H_1 \Rightarrow_{T_{i_2}} \cdots \Rightarrow_{T_{i_k}} H_k = H'$ or $H \cong H'$. We write $H \Rightarrow_{\mathcal{T}}^{i_1 i_2 \cdots i_k} H'$, $H \Rightarrow_{\mathcal{T}}^k H'$, or $H \Rightarrow_{\mathcal{T}}^* H'$. Call $i_1 i_2 \cdots i_k \in I^*$ the trace of the derivation, defined to be ε when $H \cong H'$.

Rather than replacing only non-terminals, as is usual in hyperedge replacement grammars, we allow all hyperedges to be replaced, and have a special set of terminal symbols to allow us to say when it is that a hypergraph is terminally labelled, just like ET0L grammars.

Definition 3.4 (PHR Grammar). A parallel hyperedge replacement grammar of order k (k-PHR grammar) is a tuple $\mathcal{G} = (C, A, S, \mathcal{T})$ where $C = (\Sigma, \text{type})$ is a signature, $A \subseteq \Sigma$ is the set of terminal labels, $S \in \Sigma$ is the start symbol, and $\mathcal{T} = \{T_i \mid i \in I\}$ is a non-empty, finite set of tables over Σ indexed by I with $\max(\{\text{type}(r) \mid r \in \bigcup_{T_i \in \mathcal{T}} T_i\}) \leq k$. We call $\Sigma \setminus N$ the terminal labels and call $\mathcal{G}$ repetition-free (proper) if all its tables are repetition-free (proper). The language generated by $\mathcal{G}$ is $L(\mathcal{G}) = \{H \in \mathcal{H}_C \mid S^\bullet \Rightarrow_{\mathcal{T}}^* H \text{ with } \text{lab}_H^{-1}(A) = E_H\} \subseteq \mathcal{H}_C$.

Definition 3.5 (PHR Language). $L \subseteq \mathcal{H}_C$ is called a (repetition-free) parallel hyperedge replacement language of order k ((repetition-free) k-PHR language) if there is a (repetition-free) k-PHR grammar $\mathcal{G}$ such that $L(\mathcal{G}) = L$. The class of (repetition-free) PHR languages is the union of all (repetition-free) k-PHR languages for $k \geq 0$. Denote these $\mathcal{PHR}_k$ and $\mathcal{PHR}$ ($\mathcal{PHR}_k^{\text{rf}}$ and $\mathcal{PHR}^{\text{rf}}$).

Just like languages generated by hyperedge replacement, parallel hyperedge replacement languages are closed under hypergraph isomorphism and are homogeneous in the sense that all hypergraphs in a language have the same type. Next, we confirm that PHR languages strictly contain the HR languages:

Theorem 3.6 (PHR Generalises HR). For all $k \geq 0$, $\mathcal{HR}_k \subsetneq \mathcal{PHR}_k$ and $\mathcal{HR}_k^{\text{rf}} \subsetneq \mathcal{PHR}_k^{\text{rf}}$.

Proof. Suppose $C = (\Sigma, \text{type})$ and $\mathcal{G} = (C, N, S, \{r_1, \ldots, r_n\})$ is a (repetition-free) k-HR grammar. Then we construct a (repetition-free) k-PHR grammar $\mathcal{G}' = (C, A, S, \mathcal{T})$ with $A = \Sigma \setminus N$ and $\mathcal{T} = \{T_1\}$ where $T_1 = \{r_1, \ldots, r_n\} \cup \{(X, X^\bullet) \mid X \in \Sigma\}$. Clearly every parallel direct derivation with start hypergraph G can be decomposed into at most $|E_G|$ direct derivations where if an edge is replaced by itself, we omit it, and if a genuine replacement from $\mathcal{R}$ occurs, we use that. So, by induction on derivation length, we see that every parallel derivation in $\mathcal{G}'$ can actually be written as a derivation in $\mathcal{G}$. Similarly, every direct derivation in $\mathcal{G}$ can be lifted to a parallel derivation in $\mathcal{G}'$ by replacing all but one edge by itself. So, by induction on derivation length, we see that every derivation in $\mathcal{G}$ can be written as a parallel derivation in $\mathcal{G}'$. Thus, together with the fact that the terminal symbols and start symbol coincide, $L(\mathcal{G}) = L(\mathcal{G}')$.

To see strictness, we are inspired by the fact that the string language $\{a^{2^n} \mid n \in \mathbb{N}\}$ is ET0L but not MCF (Theorem 2.8). We will show there is a repetition-free 0-PHR language that is not k-HR for any $k \geq 0$. Let $\mathcal{G} = (C, A, S, \mathcal{T})$ be the repetition-free 0-PHR grammar with $C = (\{\square\}, \{(\square, 0)\})$, $A = \{\square\}$, $S = \square$, and $\mathcal{T} = \{T_1\}$ where $T_1 = \{(\square, \square^\bullet \sqcup \square^\bullet)\}$ ($\sqcup$ denotes disjoint union of hypergraphs). Clearly $L(\mathcal{G})$ is the language of hypergraphs over C with 2^n hyperedges. By Theorem 2.3, $L(\mathcal{G})$ is not HR. $\square$

Corollary 3.7. PHR languages need not have only linear growth, in the sense of Theorem 2.3.

Figure 6 shows an example derivation using the grammar from the proof of Theorem 3.6.

3.2   Rational Control of Traces

It is often convenient to restrict the sequences of allowed traces when defining a language using a PHR grammar, leading to better readability of grammars and possibly shorter proofs. A popular choice in L systems is so-called rational control, and was considered for ET0L in 1975 by Nielsen [23] and later
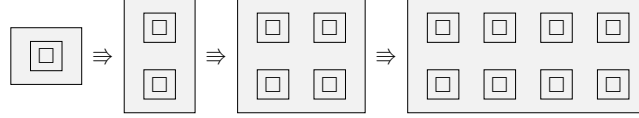
Figure 6: Example parallel derivation

by Asveld [4]. We will make precise a notion of rational control for PHR grammars, and show that generational power actually remains the same because we can always encode the rational control.

Definition 3.8 (Controlled Parallel Derivation). Given $\mathcal{C} = (\Sigma, \text{type})$, $H, H' \in \mathcal{H}_\mathcal{C}$, a finite set of tables $\mathcal{T} = \{T_i \mid i \in I\}$ over Σ indexed by I, and $\mathcal{M}$ an FSA over I, we say H ($\mathcal{M}$-)parallelly derives H' if H parallelly derives H' with trace $i_1 i_2 \cdots i_k \in \text{L}(\mathcal{M})$. We write $H \Rightarrow_\mathcal{T}^{i_1 i_2 \cdots i_k} H'$, $H \Rightarrow_\mathcal{T}^k H'$, or $H \Rightarrow_\mathcal{T}^\mathcal{M} H'$.

Definition 3.9 (PHR Grammar with Control). A (repetition-free) parallel hyperedge replacement grammar with control of order k ((repetition-free) k-PHR grammar with control) is a tuple $\mathcal{G} = (\mathcal{C}, A, S, \mathcal{T}, \mathcal{M})$ where $(\mathcal{C}, A, S, \mathcal{T})$ is a (repetition-free) k-PHR grammar (called the underlying grammar) with $\mathcal{T}$ indexed by I, and $\mathcal{M}$ is an FSA over I (called the rational control). The generated language is $\text{L}(\mathcal{G}) = \{H \in \mathcal{H}_\mathcal{C} \mid S^\bullet \Rightarrow_\mathcal{T}^\mathcal{M} H \text{ with } \text{lab}_H^{-1}(A) = E_H\} \subseteq \mathcal{H}_\mathcal{C}$.

Theorem 3.10 (PHR Grammar Control Removal). Given a (repetition-free) k-PHR grammar with control $\mathcal{G}$, one can effectively construct a (repetition-free) k-PHR grammar $\mathcal{G}'$ such that $\text{L}(\mathcal{G}) = \text{L}(\mathcal{G}')$.

Proof. Let $\mathcal{G} = ((\Sigma, \text{type}), A, S, \{T_1, \ldots T_l\}, \mathcal{M})$. Without loss of generality, we can assume that $\mathcal{M} = (Q, \underline{n}, \delta, i, F))$ is deterministic and full, and $Q \cap \Sigma = \emptyset$. We construct the (repetition-free) k-PHR grammar $\mathcal{G}' = ((\Sigma', \text{type}'), A, S', \mathcal{T}')$. First, make a disjoint (from Σ_1) copy of A, $\bar{A}$, and to each $X \in A$, associate a unique $\bar{X} \in \bar{A}$. Moreover, given a hypergraph H, denote by $\bar{H}$ the same hypergraph but with its labelling function composed with the function that sends $X \in A$ to $\bar{X}$ and leaves everything else in place. Next, choose some additional fresh symbols: S', $F_0, \ldots, F_k$. Using these, we define $\Sigma' = \{S'\} \cup Q \cup \Sigma \cup \bar{A} \cup \{F_0, \ldots, F_l\}$ where $\text{type}(S') = \text{type}(S)$, $\text{type}(q) = 0$ for all $q \in Q$, $\text{type}'(X) = \text{type}(X)$ for all $X \in \Sigma$, $\text{type}'(\bar{X}) = \text{type}(X)$ for all $X \in A$, and $\text{type}'(F_j) = j$ for all $j \in \underline{k}$.

Finally, let $\mathcal{T}' = \{T_0', T_1', \ldots T_l'\}$ where $T_0' = \{(S', \overline{S^\bullet} \sqcup i^\bullet)\} \cup \{(q, \emptyset) \mid q \in F\} \cup \{(\bar{X}, X^\bullet) \mid X \in A\} \cup \{(X, F_{\text{type}'(X)}) \mid X \in (Q \setminus F) \cup \Sigma \cup \{F_0, \ldots, F_k\}\}$ and for each $j \in \underline{l}$, $T_j' = \{(\bar{L}, \bar{R}) \mid (L, R) \in T_j \wedge L \in A\} \cup \{(L, \bar{R}) \mid (L, R) \in T_j \wedge L \notin A\} \cup \{(q, \delta(q, j)^\bullet) \mid q \in Q\} \cup \{(X, X^\bullet) \mid X \in \{S'\} \cup A \cup \{F_0, \ldots F_k\}\}$.

We can see that the purpose of table 0 is to start and stop the derivation process, with the rest of the tables simulating the original system, while also simulating the automaton. So, if $S^\bullet \Rightarrow_\mathcal{T}^w H$ is a derivation in $\mathcal{G}$ with H terminally labelled and $w \in \text{L}(\mathcal{M})$, then there is a corresponding derivation $S'^\bullet \Rightarrow_\mathcal{T}^{0w0} H$ in $\mathcal{G}'$. That is $\text{L}(\mathcal{G}) \subseteq \text{L}(\mathcal{G}')$. To see the reverse inclusion, we analyse all derivations of the form $S'^\bullet \Rightarrow_\mathcal{T}^w H$. If w does not contain 0 at least twice, then H is necessarily not terminally labelled. So, useful derivations must have trace $x0y0z$ where $x, y \in \{1, \ldots l\}^*$, $z \in \{0, \ldots l\}^*$. Clearly if $S'^\bullet \Rightarrow_\mathcal{T}^x H'$, then $H' \cong S'^\bullet$, so we can assume $x = \varepsilon$. Similarly, if $S'^\bullet \Rightarrow_\mathcal{T}^{0y0} H'$ and $H' \Rightarrow_\mathcal{T}^z H$, then either $H' \cong H$ if H was terminally labelled, and so we could assume $z = \varepsilon$, or H' is labelled by at least non-terminal which has no terminally labelled successor hypergraph, and so it doesn't matter what H is. Finally, we analyse y. If $y \in \text{L}(\mathcal{M})$, then we proceed as in the analysis of the other direction of inclusion. If $y \notin \text{L}(\mathcal{M})$, then the final step sends the type zero symbol tracking the machine state to F_0 which forces all successors of the hypergraph to be not terminally labelled. □

Thus, the hypergraph languages that can be generated by k-PHR grammars are exactly those that can be generated by k-PHR grammars with control, since certainly no control can be simulated.

3.3 PHRS Languages

We now turn our attention to string languages. We believe the class of parallel hyperedge replacement string languages is a genuinely new class of languages, containing all multiple context-free and ET0L languages. It is not simply equal to the (parallel) multiple context-free languages because these are known to be incomparable with ET0L [24]. Recall that the hyperedge replacement string languages are exactly the multiple context-free languages. In this subsection, we confirm that parallel hyperedge replacement string languages contain all of these and also all of the ET0L languages.

Definition 3.11 (PHR String Language). A string language $L \subseteq A^*$ is called a (repetition-free) parallel hyperedge replacement string language of order k ((repetition-free) k-PHRS language) if there is a (repetition-free) k-PHR grammar $\mathcal{G}$ such that $\mathcal{G}$ generates a string graph language and $\mathrm{STR}(\mathrm{L}(\mathcal{G})) = L \setminus \{\varepsilon\}$. The class of (repetition-free) PHRS languages is the union of all (repetition-free) k-PHRS languages for $k \geq 2$. Denote these $\mathcal{PHRS}_k$ and $\mathcal{PHRS}$ ($\mathcal{PHRS}_k^{\mathrm{rf}}$ and $\mathcal{PHRS}^{\mathrm{rf}}$).

Notice that we exclude the case $k < 2$, since $\mathcal{PHRS}_0^{\mathrm{rf}} = \mathcal{PHRS}_0 = \mathcal{PHRS}_1^{\mathrm{rf}} = \mathcal{PHRS}_1 = \{\emptyset, \{\varepsilon\}\}$. It is clear that $\mathcal{PHRS}_k^{\mathrm{rf}} \subseteq \mathcal{PHRS}_k$ for $k \geq 2$, and that we can iteratively compute the unreachable symbols in a similar way as for context-free grammars (see Section 7.1 of [17]):

Definition 3.12 (Unreachable Symbol). Given a PHR grammar over (Σ, type), $X \in \Sigma$ is called unreachable if there is no derivation starting at $S^{\bullet}$ containing a hypergraph with a hyperedge labelled by X.

Lemma 3.13. For $k \geq 2$, given a (repetition-free) k-PHR grammar $\mathcal{G}$, one can effectively construct a (repetition-free) k-PHR grammar $\mathcal{G}'$ with no unreachable symbols and $\mathrm{L}(\mathcal{G}) = \mathrm{L}(\mathcal{G}')$.

The following lemma is also clear, using Lemma 3.13, enabling us to prove Theorem 3.15:

Lemma 3.14. For $k \geq 2$, given a k-PHR grammar $\mathcal{G}$ generating a string graph language, one can effectively construct a proper k-PHR grammar $\mathcal{G}'$ such that there are no unreachable symbols, all terminals are type 2, all non-terminals are type at least 2, and $\mathrm{L}(\mathcal{G}) = \mathrm{L}(\mathcal{G}')$.

Theorem 3.15 (PHRS Generalises ET0L). For all $k \geq 2$, $\mathcal{ET0L} \subseteq \mathcal{PHRS}_k^{\mathrm{rf}}$. When $k \geq 4$, $\mathcal{ET0L} \subsetneq \mathcal{PHRS}_k^{\mathrm{rf}}$. Moreover, $\mathcal{ET0L} = \mathcal{PHRS}_2^{\mathrm{rf}} = \mathcal{PHRS}_2$.

Proof. First we show $\mathcal{ET0L} \subseteq \mathcal{PHRS}_k^{\mathrm{rf}}$ for all $k \geq 2$. Suppose L is an ET0L language, then by Theorem 2.6, there exists a propagating ET0L grammar $\mathcal{G} = (\Sigma, A, S, \{T_i \mid i \in I\})$ such that $L \setminus \{\varepsilon\} = \mathrm{L}(\mathcal{G})$. It follows that every rule can be encoded as an HR rule over $\mathcal{C}' = (\Sigma, \Sigma \times \{2\})$ giving us a repetition-free 2-PHR grammar $\mathcal{G}' = (\mathcal{C}', A, S, \{\{(L, R^{\bullet}) \mid (L, R) \in T_i\} \mid i \in I\})$ with $\mathrm{L}(\mathcal{G}) = \mathrm{STR}(\mathrm{L}(\mathcal{G}'))$.

Next, we show that $\mathcal{PHRS}_2 \subseteq \mathcal{ET0L}$. Suppose L is a 2-PHRS language, then there is a 2-PHR grammar $\mathcal{G} = (\mathcal{C}, A, S, \{T_i \mid i \in I\})$ generating a string graph language such that $L \setminus \{\varepsilon\} = \mathrm{STR}(\mathrm{L}(\mathcal{G}))$. Lemma 3.14 allows us to assume a lot about the form of RHSs of rules. It is easy to see that all RHSs must actually be string graphs, or could be transformed to string graphs, since any non-conformant pieces can just be inlined into the string graph because it will ultimately be deleted and the nodes merged in any terminally labelled derived hypergraph. So the system can be converted into an ET0L grammar. Thus, we have $\mathcal{PHRS}_2 \subseteq \mathcal{ET0L} \subseteq \mathcal{PHRS}_2^{\mathrm{rf}}$ and $\mathcal{PHRS}_2^{\mathrm{rf}} \subseteq \mathcal{PHRS}_2$, so the inclusions must be equalities.

Finally, strictness follows from Theorems 2.4, 2.8, and 3.6. That is, we can construct a repetition-free 4-PHR grammar $\mathcal{G}'$ generating a string graph language with $\mathrm{STR}(\mathrm{L}(\mathcal{G}')) = K \setminus \{\varepsilon\}$ (from Theorem 2.8) which is not ET0L. $\square$

Corollary 3.16. There are repetition-free 2-PHRS languages that are not semilinear.

Proof. Theorem 3.15 gives us a 2-PHRS language which is not semilinear by Theorem 2.8. $\square$

Theorem 3.17 (PHRS Generalises MCF). *For all $k \geq 2$, $\mathcal{HRS}_k^{\mathrm{rf}} \subsetneq \mathcal{PHRS}_k^{\mathrm{rf}}$.*

Proof. Inclusion from Theorem 2.4, and Theorem 3.6 and its proof. We get strictness from Theorem 2.8 together with Theorem 3.15. □

3.4 Formal Language Closure Properties

Recall that a full AFL is a non-empty class of string languages closed under rational operations (union, concatenation, Kleene plus), rational intersection, homomorphisms, and inverse homomorphisms. In this subsection, we show that the class of PHRS languages is an (iterated) substitution closed full AFL, and that the class of repetition-free PHRS languages is closed under non-erasing (iterated) substitution with closure under rational operations and non-erasing homomorphisms following from this as a corollary.

Theorem 3.18 (PHRS Closed Under Substitutions). *Let $L \subseteq A^*$ be a k-PHRS language (repetition-free k-PHRS language) and h be a k-PHRS substitution (non-erasing repetition-free k-PHRS substitution) on A. Then $h(L)$ and $\bigcup_{n \in \mathbb{N}} h^n(L)$ are k-PHRS languages (repetition-free k-PHRS languages).*

Proof. There is a (repetition-free) k-PHR grammar $\mathcal{G} = (\mathcal{C}, A, S, \mathcal{T})$ such that $A = \{a_1, \ldots, a_m\}$ and $\mathrm{L}(\mathcal{G})$ is a string graph language, and $\mathrm{STR}(\mathrm{L}(\mathcal{G})) = L \setminus \{\varepsilon\}$. Similarly for each $i \in \underline{m}$, there is a (repetition-free) k-PHR grammar $\mathcal{G}_i = (\mathcal{C}_i, B, S_i, \mathcal{T}_i)$ such that $\mathrm{L}(\mathcal{G})$ is a string graph language and $\mathrm{STR}(\mathrm{L}(\mathcal{G}_i)) = h(a_i)$. Without loss of generality, we assume both that each S_i does not appear as a label in any RHS apart from possibly a rule $(S_i, S_i^\bullet)$ and that the following sets are pairwise disjoint: $\Sigma, B, \Sigma_1 \setminus B, \ldots, \Sigma_m \setminus B$. For each $i \in \underline{m}$, let $\overline{\Sigma_i}$ be a copy of Σ_i consisting of fresh symbols, and identify each $x \in \Sigma_i$ with its copy $\bar{x} \in \overline{\Sigma_i}$. Given a hypergraph H, by $\overline{H}$ we mean H but with its labelled function composed with the function which takes any $x \in \Sigma_i$ to $\bar{x}$ and leaves everything else fixed.

We now construct the (repetition-free) k-PHR grammar $\mathcal{G}' = (\mathcal{C}', B, \mathcal{T}', S)$ such that $\mathrm{L}(\mathcal{G}')$ is a string graph language and $\mathrm{STR}(\mathrm{L}(\mathcal{G}')) = h(L) \setminus \{\varepsilon\}$. Let $\mathcal{C}' = (\Sigma', \mathrm{type}')$ be the union of all the above signatures also including the disjoint copies, where copies are of the same type as their original symbols, together with the fresh symbols $F_0, \ldots F_k$ of type $0, \ldots, k$, respectively. Finally, let $\mathcal{R} = \{(X, X^\bullet) \mid X \in \Sigma'\}$, $\mathcal{F} = \{(X, F_{\mathrm{type}'(X)}^\bullet) \mid X \in \Sigma'\}$, and $\mathcal{T}' = \bigcup_{0 \leq i \leq m} \mathcal{T}_i'$, where $\mathcal{T}_0' = \{\mathcal{R} \oplus T \mid T \in \mathcal{T}\} \cup \{\mathcal{F} \oplus \{(a_i, \overline{S_i}^\bullet) \mid i \in \underline{m}\}\}$, and for each $i \in \underline{m}$ let $\mathcal{T}_i = \{\mathcal{R} \oplus (\{(\overline{L}, \overline{R}) \mid (L, R) \in T\} \cup \{(\overline{S_i}, \overline{S_i}^\bullet)\}) \mid T \in \mathcal{T}_i\} \cup \{\mathcal{R} \oplus (\{(\overline{X}, X^\bullet) \mid X \in B\} \cup \{(\overline{X}, F_{\mathrm{type}'(X)}^\bullet) \mid X \in \Sigma_i \setminus (B \cup \{\overline{S_i}\})\})\}$.

One can see that derivations that make progress start with $S^\bullet$ then apply tables from the first part of $\mathcal{T}_0'$, simulating $\mathcal{G}$. At some point, the final table of $\mathcal{T}_0'$ may be applied, which immediately rewrites all the terminals to encoded start symbols for their respective grammars for substitution and sends all non-terminals to failure non-terminals. If a hypergraph contains any failure non-terminals at this point, non terminally labelled hypergraph can be derived in future. Derivations can now simulate the $\mathcal{G}_i$ totally independently, with choice of delaying start, giving total freedom over the simulated derivation sequences for each instance of the encoded start symbol. Finally, the encoded systems can end their simulation at any point by sending their encoded terminals to real terminals in B and their encoded non-terminals to failure non-terminals. It is now clear that $\mathrm{STR}(\mathrm{L}(\mathcal{G}')) = h(L) \setminus \{\varepsilon\}$, as required.

Showing $\bigcup_{n \in \mathbb{N}} h^n(L)$ is similar, modifying the above proof, adding another table which can be used to send all terminals to restart the process and all non-terminals to a failure symbol. □

Corollary 3.19 (PHRS Closed Under Homomorphisms). *Let $L \subseteq A^*$ be a k-PHRS language (repetition-free k-PHRS language) for any $k \geq 2$ and $\varphi : A^* \to B^*$ be a homomorphism (non-erasing homomorphism). Then $\varphi(L)$ is a k-PHRS language (repetition-free k-PHRS language).*

Next, we show closure under rational operations, which can be seen via the following general result:

Lemma 3.20. Let $\mathcal{F}$ be a class of string languages containing all regular languages, which is closed under non-erasing substitution. Let $L_1, L_2 \subseteq A_1^*$ be $\mathcal{F}$ languages. Then:

1. $L_1 \cup L_2$ is an $\mathcal{F}$ language;   (closure under union)

2. $L_1 L_2$ is an $\mathcal{F}$ language;   (closure under concatenation)

3. L_1^+ is an $\mathcal{F}$ language.   (closure under Kleene plus)

Proof. To see (1), notice that $L_1 \cup L_2$ is simply $h(K)$ where $K = \underline{\text{if}} \ \varepsilon \in L_1 \cup L_2 \ \underline{\text{then}} \ \{X, Y, \varepsilon\} \ \underline{\text{else}} \ \{X, Y\}$, and h is a non-erasing substitution with $h(X) = L_1 \setminus \{\varepsilon\}$ and $h(Y) = L_2 \setminus \{\varepsilon\}$. Thus we have $L_1 \cup L_2 = h(K)$, and since, in either case, K is a regular language, $h(K) \in \mathcal{F}$. (2) and (3) are similar.   □

Theorem 3.21 (PHRS Closed Under Rational Operations). Let $L_1, L_2 \subseteq A_1^*$ be (repetition-free) k-PHRS languages for any $k \geq 2$. Then:

1. $L_1 \cup L_2$ is a (repetition-free) k-PHRS language;   (closure under union)

2. $L_1 L_2$ is a (repetition-free) k-PHRS language;   (closure under concatenation)

3. L_1^+ is a (repetition-free) k-PHRS language.   (closure under Kleene plus)

Proof. Combine Theorem 3.18 and Lemma 3.20.   □

We now show closure under rational intersection, inspired by the proof of Theorem V.1.7(iv) of [27]:

Theorem 3.22 (PHRS Closed Under Rational Intersection). Let $L \subseteq A^*$ be a (repetition-free) k-PHRS language and $K \subseteq B^*$ be a regular language, for any $k \geq 2$. Then $L \cap K$ is a (repetition-free) k-PHRS language.

Proof. There is a (repetition-free) k-PHR grammar $\mathcal{G} = (\mathcal{C} = (\Sigma, \text{type}), A, \mathcal{T} = \{T_1, \ldots T_n\}, S)$ such that $\text{L}(\mathcal{G})$ is a string graph language and $\text{STR}(\text{L}(\mathcal{G})) = L \setminus \{\varepsilon\}$. Without loss of generality, we assume that $(\Sigma \setminus A) \cap B = \emptyset$. There must also be a deterministic full FSA $\mathcal{M} = (Q, B, \delta, p, F)$ such that $\text{L}(\mathcal{M}) = K \setminus \{\varepsilon\}$. We will now construct a (repetition-free) k-PHR grammar with control $\mathcal{G}' = (\mathcal{C}', A \cap B, \mathcal{T}', S, \mathcal{M}')$ such that $\text{L}(\mathcal{G}')$ is a string graph language and $\text{STR}(\text{L}(\mathcal{G}')) = (L \cap K) \setminus \{\varepsilon\}$, thus proving that $L \cap K$ is a (repetition-free) k-PHRS language, using Theorem 3.10.

First, we define the signature $\mathcal{C}'$. Let $\Delta = (\bigcup_{0 \leq i \leq k} \{\text{type}^{-1}(\{i\}) \times Q^i\})$, $\Sigma' = \Delta \cup \{S\} \cup (A \cap B)$, $\text{type}'((X, q_1, \ldots q_i)) = \text{type}(X)$ for all $(X, q_1, \ldots q_i) \in \Delta$, $\text{type}'(S) = 2$, and $\text{type}'(X) = 2$ for all $X \in A \cap B$.

In order to define $\mathcal{T}'$ it will be useful to introduce the intermediate notion of a hypergraph with node labels. In particular, we are interested in labelling the nodes by states of $\mathcal{M}$. A node labelled hypergraph over $(\mathcal{C}, Q)$ is a pair (H, l) where H is a hypergraph over $\mathcal{C}$ and l is a function $V_H \to Q$. Notice that any such node labelled hypergraph can be encoded as a hypergraph over $(\Delta, \text{type}'|_\Delta)$: for each $e \in E_H$, the new hyperedge labelling function is defined by sending e to $(\text{lab}_H(e), q_1, \ldots, q_i)$ where $i = \text{type}_H(e)$ and $q_j = l(\text{att}_H(e)(j))$ for $1 \leq j \leq i$. Call this injective encoding function enc. Next, given a type t hypergraph H over $\mathcal{C}$ and a sequence $\sigma : \underline{t} \to Q$, define $\text{CHOICES}_Q(H, \sigma) = \{\text{enc}((H, l)) \mid l : V_H \to Q, l \circ \text{ext}_H = \sigma\}$.

We now define $\mathcal{T}' = \{T_0', T_1', \ldots T_n'\}$ where:

1. $T_0' = \mathcal{R} \oplus \{((X, q_1, q_2), Y^\bullet) \mid X \in A \cap B, \delta(q_1, X) = q_2\}$;

2. $T_i' = \mathcal{R} \oplus (\bigcup_{(L,R) \in T_i} \{((L, \sigma(1), \ldots, \sigma(t)), H) \mid t = \text{type}(L), \sigma : \underline{t} \to Q, H \in \text{CHOICES}_Q(R, \sigma)\} \cup \{(S, (S, p, q)^\bullet) \mid q \in F\})$, for $1 \leq i \leq n$;

   where $\mathcal{R} = \{(X, X^\bullet) \mid X \in \Sigma'\}$.

Finally, let $\mathcal{M}'$ be an FSA defined by the regular expression $\{1,\ldots,n\}^+0$. Correctness follows from the fact that the application of the final table T_0 will produce a terminal string graph $(x_1 x_2 \cdots x_m)^\bullet$ if and only if the previous hypergraph was a string graph of the form $((x_1,q_1,q_2)(x_2,q_2,q_3)\cdots(x_m,q_m,q_{m+1}))^\bullet$ and $\delta(q_i,x_i) = q_{i+1}$ for $1 \le i \le m$, $q_1 = p$, and $q_{m+1} \in F$. That is, we have traced out an accepting path in the FSA $\mathcal{M}$, having simulated $\mathcal{G}$. $\qquad\square$

Finally, we show closure under inverse homomorphisms, via the following general result:

Lemma 3.23. Let $\mathcal{F}$ be a class of string languages which is closed under rational substitution and rational intersection. Let $L \subseteq A^*$ be an $\mathcal{F}$ language and $\varphi : B^* \to A^*$ a homomorphism. Then $\varphi^{-1}(L)$ is an $\mathcal{F}$ language too.

Proof. Let $\bar{B}$ be a copy of B such that $(A \cup B) \cap \bar{B} = \emptyset$, and let $\bar{\cdot} : B \to \bar{B}$ identify each $b \in B$ with its copy $\bar{b} \in \bar{B}$. For each $a \in A$, define the regular language $L_a = \{w_1 a w_2 \mid w_1, w_2 \in \bar{B}^*\} \subseteq (A \cup \bar{B})^*$ and the rational substitution h on A by $a \mapsto L_a$. Also define $K = \bigcup_{n \in \mathbb{N}} \{\varphi(x_1)\overline{x_1}\varphi(x_2)\overline{x_2} \cdots \varphi(x_n)\overline{x_n} \mid x_1, x_2, \ldots x_n \in B\}$ and the homomorphism $\psi : (A \cup \bar{B})^* \to B^*$ by $\psi(a) = \varepsilon$ for each $a \in A$ and $\psi(\bar{b}) = b$ for each $b \in B$.

Notice $h(L) \cap K = \bigcup_{n \in \mathbb{N}} \{\varphi(x_1)\overline{x_1}\varphi(x_2)\overline{x_2} \cdots \varphi(x_n)\overline{x_n} \mid x_1, \ldots, x_n \in B$ and $\varphi(x_1)\varphi(x_2)\cdots\varphi(x_n) \in L\}$, so we have $\varphi^{-1}(L) = \psi(h(L) \cap K)$. Now, $h(L)$ is an $\mathcal{F}$ language since $\mathcal{F}$ is closed under rational substitution, $h(L) \cap K$ is an $\mathcal{F}$ language since $\mathcal{F}$ is closed under rational intersection, and $\psi(h(L) \cap K)$ is an $\mathcal{F}$ language since $\mathcal{F}$ is closed under homomorphisms (a special case of rational substitution). Thus, $\varphi^{-1}(L)$ is an $\mathcal{F}$ language, as required. $\qquad\square$

Theorem 3.24 (PHRS Closed Under Inverse Homomorphisms)**.** For all $k \ge 2$, $\mathcal{PHRS}_k$ is closed under inverse homomorphisms.

Proof. The result follows from Theorems 3.18 and 3.22 and Lemma 3.23. $\qquad\square$

3.5 Group Word Problem Closure Properties

Since the class of k-PHRS languages is a full AFL for any $k \ge 2$, it satisfies the following important properties:

Theorem 3.25 (WP Independent Of Presentation [15])**.** Let $\mathcal{F}$ be a class of string languages which is closed under inverse homomorphisms, and let $\langle X \mid R \rangle$ be a presentation of a group G such that $\mathrm{WP}_X(G)$ is an $\mathcal{F}$ language. Then all presentations $\langle X' \mid R' \rangle$ of G are such that $\mathrm{WP}_{X'}(G)$ is an $\mathcal{F}$ language.

Theorem 3.26 (WP Subgroup and Supergroup Closure [11])**.** Let $\mathcal{F}$ be a full AFL and G be a group with word problem in $\mathcal{F}$. Then every finitely generated subgroup and every finite index supergroup of G has word problem in $\mathcal{F}$.

In 2019, Kropholler and Spriano showed that a graph of groups with vertex groups with MCF word problem and edge groups finite, yields a group with an MCF word problem [20]. A special case of this construction is a free product of groups. We now show that a free product of groups with (repetition-free) PHRS word problems is a group with a (repetition-free) PHRS word problem. Our strategy is entirely different to Kropholler and Spriano's approach, which relied on Denkinger's automata characterisation of MCF languages (Theorem 2.4).

The following easy lemma, where presentations of groups are written as monoid presentations, gives us a recursive description of the word problem of free products, enabling us to prove Theorem 3.28.

Lemma 3.27. Let G_1, G_2 be finitely generated groups over disjoint alphabets $A_1 = \{a_1, \ldots a_n\}$, $A_2 = \{b_1, \ldots b_m\}$, respectively. If $X = A_1 \cup A_2$ and $L_i = \mathrm{WP}_{A_i}(G_i)$ for $i = 1, 2$, then $\mathrm{WP}_X(G_1 * G_2)$ is the smallest set L such that $\varepsilon \in L$ and $\forall i \in \{1, 2\}, \forall w \in L_i, \forall u, v \in X^*, uv \in L \Rightarrow uwv \in L$.

Theorem 3.28 (WP Free Product Closure). Let $\mathcal{F}$ be a class of string languages containing all finite languages, closed under union and concatenation, and closed under nested iterated substitution. Then if G_1, G_2 are groups with presentations admitting a $\mathcal{F}$ word problem, $G_1 * G_2$ has a presentation admitting a $\mathcal{F}$ word problem.

Proof. Let A_1, A_2, X, L_1, L_2, L be as in Lemma 3.27, then it is immediate that iterated application of the nested non-erasing $\mathcal{F}$-substitution h of strings on $A_1 \cup A_2$, defined by $h(a_i) = \{a_i\} \cup a_i L_1 \cup L_1 a_i$ and $h(b_j) = \{b_j\} \cup b_j L_2 \cup L_2 b_j$ for all $i \in \underline{n}, j \in \underline{m}$, to L, gives us exactly $\mathrm{WP}_X(G_1 * G_2)$. The result them follows from the assumed closure properties. $\qquad\square$

4 Conclusion and Future Work

We have shown some foundational properties of parallel hyperedge replacement grammars, with a focus on string generational power, showing that the class of parallel hyperedge replacement string languages is a substitution and iterated substitution closed full AFL, containing all MCF and ET0L languages. Theorem 2.2 tells us that the string generational power of HR grammars is not restricted by requiring grammars to be repetition-free. It remains future work to determine if a similar result holds in the parallel replacement setting. If it turns out that there is no such result, there is still a middle-ground where one can obtain all of the closure properties we have shown, but without allowing merging of nodes by derivations. Call the below equivalent classes the repetition-free weak-coded k-PHRS languages ($\mathcal{WPHRS}_k^{\mathrm{rf}}$):

1. The class of string languages generated by repetition-free k-PHR grammars under the image of some weak coding.

2. The class of string languages generated by repetition-free k-PHR grammars with a special type 2 label `empty`, interpreted as the empty string by STR.

Using the results and proofs from Subsection 3.4, it is not too difficult to see that $\mathcal{WPHRS}_k^{\mathrm{rf}}$ is a substitution and iterated substitution closed full AFL, and that $\mathcal{PHRS}_2^{\mathrm{rf}} = \mathcal{WPHRS}_2^{\mathrm{rf}} = \mathcal{PHRS}_2$, due to the proof of Theorem 3.15. We conjecture this holds for all $k \geq 2$:

Conjecture 4.1 (PHR String Generational Power). For all $k \geq 2$, $\mathcal{PHRS}_k^{\mathrm{rf}} = \mathcal{WPHRS}_k^{\mathrm{rf}} = \mathcal{PHRS}_k$.

Figure 7 summarises the closure properties we know. It remains future work to show that the class of PHRS languages is a strict subclass of the context-sensitive languages. We conjecture this to be true, and we also conjecture that only even increments in order increase string generative power. Figure 8 summarises both our known and conjectured string language hierarchies.

Conjecture 4.2 (CS Generalises PHRS). $\mathcal{PHRS} \subsetneq \mathcal{CS}$.

Conjecture 4.3 (PHRS Grouping). For all $k \geq 1$, $\mathcal{PHRS}_{2k} = \mathcal{PHRS}_{2k+1}$.

Because $\mathcal{PHRS}$ is closed under inverse homomorphisms, we know that the property of having a PHRS word problem is independent of the presentation. We have additionally shown that PHRS groups are closed under free product. We also conjecture the following, which has a wide-reaching corollary:

Conjecture 4.4 (PHRS WP Double Torus). The fundamental group of the double torus admits a PHRS word problem which is neither an MCF nor ET0L language.

Corollary 4.5. If Conjecture 4.4 is true, then the word problem of any surface group is PHRS.

Proof. By a *surface* here, we mean a closed, connected, orientable, 2-manifold, and by a *surface group*, we mean the fundamental group of a surface. Any surface always has a finite genus. The genus 0 surface (the sphere) gives us the trivial group, and 1 (the torus), $\mathbb{Z}^2$ (see for example [21]). We know both of these groups are regular, 2-MCF [16], respectively, so certainly PHRS (Theorem 3.17).

For higher genuses, it follows from the Fundamental Theorem of Covering Spaces (Theorem 1.38 of [14]) that the fundamental group appears as a finitely generated subgroup of the fundamental group of a genus 2 surface such as a double torus. Since $\mathcal{PHRS}$ is a full AFL, if the double torus has fundamental group with PHRS word problem, then all its finitely generated subgroups do too (Theorem 3.26). □

Highly related to the word problem is the consideration of sets of solutions of more general equations over groups or other structures. It is a recent result that solution sets (of fixed normal forms) of finite systems of equations in hyperbolic groups are EDT0L languages [5]. We are yet to consider deterministic parallel hyperedge replacement, but it may be possible to establish that other classes of groups have solution sets that are deterministic parallel hyperedge replacement string languages.

It remains future work to consider the effect of tables on generative power. It is a long-standing result that ET0L grammars with only one table have less generative power than those with two tables, and that allowing more than two tables does not increase generative power any further [27]. It is likely that a similar result holds for PHR and PHRS languages. Other more general future work would include investigating both the tree and graph generational power of PHR grammars, and investigating decidability and complexity results for basic problems relating to PHR grammars. We do not know if the emptiness or finiteness problems for PHR grammars are decidable, but we conjecture that they are.

Conjecture 4.6 (Decidable PHR Emptiness). The following problem is decidable:

    Instance: A PHR grammar $\mathcal{G} = (\mathcal{C}, A, S, \mathcal{T})$.
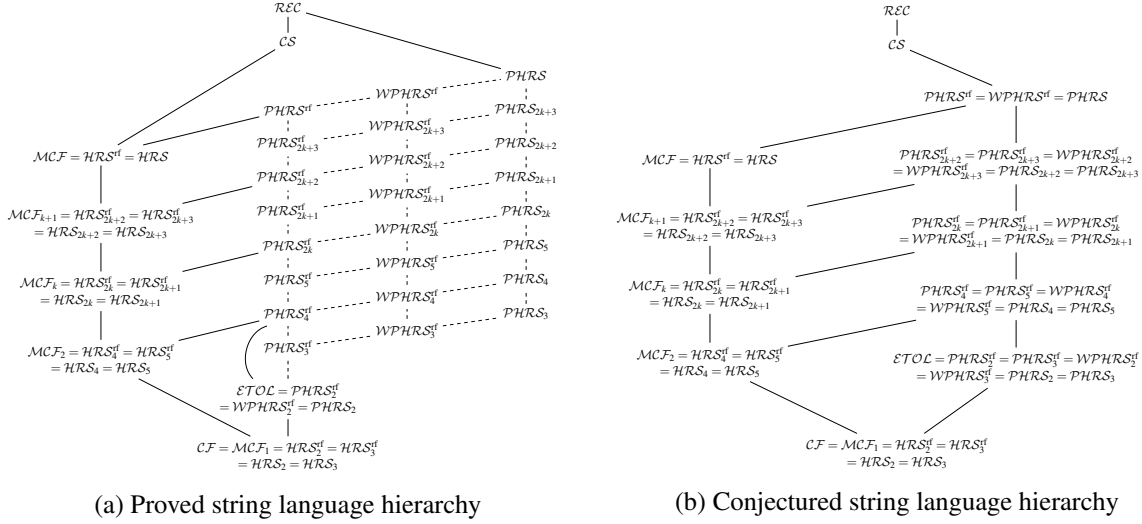   **Question:** Is $L(\mathcal{G}) = \emptyset$?

Conjecture 4.7 (Decidable PHR Finiteness). The following problem is decidable:

    Instance: A PHR grammar $\mathcal{G} = (\mathcal{C}, A, S, \mathcal{T})$.
    Question: Does $L(\mathcal{G})$ contain only finitely many non-isomorphic hypergraphs?

Operation/Class	$\mathcal{HRS}_k^{\mathrm{rf}}$ $= \mathcal{HRS}_k$	$\mathcal{PHRS}_k^{\mathrm{rf}}$	$\mathcal{WPHRS}_k^{\mathrm{rf}}$	$\mathcal{PHRS}_k$
Rational Operations	✓	✓	✓	✓
Rational Intersection	✓	✓	✓	✓
Inverse Homomorphisms	✓	?	✓	✓
Non-Erasing Homomorphisms	✓	✓	✓	✓
Arbitrary Homomorphisms	✓	?	✓	✓
Non-Erasing Substitutions	✓	✓	✓	✓
Arbitrary Substitutions	✓	?	✓	✓
Iterated Nested Non-Erasing Substitutions	✓	✓	✓	✓
Iterated Nested Arbitrary Substitutions	✓	?	✓	✓
Iterated Non-Erasing Substitutions	✗	✓	✓	✓
Iterated Arbitrary Substitutions	✗	?	✓	✓

Figure 7: Summary of formal language closure properties ($k \geq 2$)

(a) Proved string language hierarchy   (b) Conjectured string language hierarchy

Figure 8: Detailed formal language hierarchies ($k \geq 3$)

References

[1] Alfred Aho (1968): *Indexed Grammars – An Extension of Context-Free Grammars*. Journal of the ACM 15(4), pp. 647–671, doi:10.1145/321479.321488.

[2] Alfred Aho & Jeffrey Ullman (1972): *Translations on a Context Free Grammar*. Information and Control 19(5), pp. 439–475, doi:10.1016/S0019-9958(71)90706-6.

[3] Anatoly Anisimov (1971): *Group languages*. Cybernetics 7, pp. 594–601, doi:10.1007/BF01071030.

[4] Peter Asveld (1977): *Controlled iteration grammars and full hyper-AFL's*. Information and Control 34(3), pp. 248–269, doi:10.1016/S0019-9958(77)90308-4.

[5] Laura Ciobanu & Murray Elder (2019): *Solutions Sets to Systems of Equations in Hyperbolic Groups Are EDT0L in PSPACE*. In: *Proc. 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019), Leibniz International Proceedings in Informatics (LIPIcs)* 132, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, pp. 110:1–110:15, doi:10.4230/LIPIcs.ICALP.2019.110.

[6] Laura Ciobanu, Murray Elder & Michal Ferov (2018): *Applications of L systems to group theory*. International Journal of Algebra and Computation 28(2), pp. 309–329, doi:10.1142/S0218196718500145.

[7] Tobias Denkinger (2016): *An Automata Characterisation for Multiple Context-Free Languages*. In: *Proc. International Conference on Developments in Language Theory (DLT 2016), Lecture Notes in Computer Science* 9840, Springer, pp. 138–150, doi:10.1007/978-3-662-53132-7_12.

[8] Frank Drewes, Hans-Jörg Kreowski & Annegret Habel (1997): *Hyperedge Replacement Graph Grammars*, pp. 95–162. World Scientific, doi:10.1142/9789812384720_0002.

[9] Andrzej Ehrenfeucht & Grzegorz Rozenberg (1977): *On some context-free languages that are not deterministic ET0L languages*. R.A.I.R.O. Informatique théorique 11(4), pp. 273–291, doi:10.1051/ita/1977110402731.

[10] Joost Engelfriet & Linda Heyker (1991): *The string generating power of context-free hypergraph grammars*. Journal of Computer and System Sciences 43(2), pp. 328–360, doi:10.1016/0022-0000(91)90018-Z.

[11] Robert Gilman, Robert Kropholler & Saul Schleimer (2018): *Groups whose word problems are not semilinear*. Groups Complexity Cryptology 10(2), pp. 53–62, doi:10.1515/gcc-2018-0010.

[12] Robert Gilman & Michael Shapiro (1998): *On groups whose word problem is solved by a nested stack automaton*. Available at https://arxiv.org/abs/math/9812028.

[13] Annegret Habel (1992): *Hyperedge Replacement: Grammars and Languages*. Lecture Notes in Computer Science 643, Springer, doi:10.1007/BFb0013875.

[14] Allen Hatcher (2002): *Algebraic Topology*. Cambridge University Press.

[15] Thomas Herbst & Richard Thomas (1993): *Group presentations, formal languages and characterizations of one-counter groups*. Theoretical Computer Science 112(2), pp. 187–213, doi:10.1016/0304-3975(93)90018-O.

[16] Meng-Che Ho (2018): *The word problem of $\mathbb{Z}^n$ is a multiple context-free language*. Groups Complexity Cryptology 10(1), pp. 9–15, doi:10.1515/gcc-2018-0003.

[17] John Hopcroft, Rajeev Motwani & Jeffrey Ullman (2006): *Introduction to Automata Theory, Languages, and Computation*, 3rd ed. edition. Addison-Wesley.

[18] Hans-Jörg Kreowski (1992): *Parallel Hyperedge Replacement*, pp. 271–282. Springer, doi:10.1007/978-3-642-58117-5_17.

[19] Hans-Jörg Kreowski (1993): *Five facets of hyperedge replacement beyond context-freeness*. In: Proc. 9th International Conference on Fundamentals of Computation Theory (FCT 1993), Lecture Notes in Computer Science 710, Springer, pp. 69–86, doi:10.1007/3-540-57163-9_5.

[20] Robert Kropholler & Davide Spriano (2019): *Closure properties in the class of multiple context-free groups*. Groups Complexity Cryptology 11(1), pp. 1–15, doi:10.1515/gcc-2019-2004.

[21] William Massey (1977): *Algebraic Topology: An Introduction*. Graduate Texts in Mathematics 56, Springer.

[22] David Muller & Paul Schupp (1983): *Groups, the Theory of Ends, and Context-Free Languages*. Journal of Computer and System Sciences 26(3), pp. 295–310, doi:10.1016/0022-0000(83)90003-X.

[23] Mogens Nielsen (1975): *EOL systems with control devices*. Acta Informatica 4, pp. 373–386, doi:10.1007/BF00289618.

[24] Taishin Nishida & Shigeko Seki (2000): *Grouped partial ET0L systems and parallel multiple context-free grammars*. Theoretical Computer Science 246(1–2), pp. 131–150, doi:10.1016/S0304-3975(99)00076-6.

[25] Pyotr Novikov (1955): *Über die algorithmische Unentscheidbarkeit des Wortproblems in der Gruppentheorie*. Trudy Matematicheskogo Instituta imeni V.A. Steklova 44, pp. 1–143.

[26] Rohit Parikh (1966): *On Context-Free Languages*. Journal of the ACM 13(4), pp. 570–581, doi:10.1145/321356.321364.

[27] Grzegorz Rozenberg & Arto Salomaa (1980): *The Mathematical Theory of L Systems*. Pure and Applied Mathematics 90, Academic Press.

[28] Sylvain Salvati (2015): *MIX is a 2-MCFL and the word problem in $\mathbb{Z}^2$ is captured by the IO and the OI hierarchies*. Journal of Computer and System Sciences 81(7), pp. 1252–1277, doi:10.1016/j.jcss.2015.03.004.

[29] Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii & Tadao Kasami (1991): *On multiple context-free grammars*. Theoretical Computer Science 88(2), pp. 191–229, doi:10.1016/0304-3975(91)90374-B.

[30] Vijay Shanker, David Weir & Aravind Joshi (1987): *Characterizing structural descriptions produced by various grammatical formalisms*. In: Proc. 25th Annual Meeting on Association for Computational Linguistics (ACL '87), Association for Computational Linguistics, pp. 104–111, doi:10.3115/981175.981190.

[31] David Weir (1992): *Linear context-free rewriting systems and deterministic tree-walking transducers*. In: Proc. 30th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, pp. 136–143, doi:10.3115/981967.981985.