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 finitely generated virtually Abelian groups are multiple context-free, and it is a long standing open question where to place the hyperbolic groups in the formal language hierarchy. In this paper we introduce a new language class, the parallel hyperedge replacement string languages, generalising the multiple context-free and ET0L languages, and lay down the foundations for future work that may be able to place the hyperbolic groups in this class.

1 Introduction

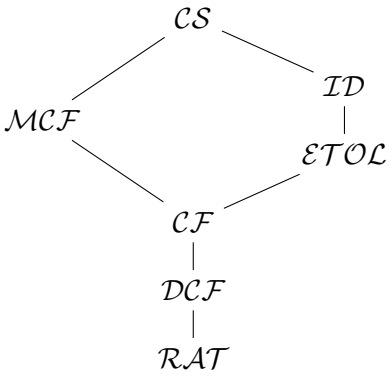
In general, the word problem is the question that asks whenever two strings represent the same element in some structure. In the case of groups, this is the equivalent to asking if a single string is equivalent to the identity, since if u, v are strings, then they are equal in a group if and only if uv^{-1} is equal to the identity in the group. Thus, given a group presentation $\langle X \mid R \rangle$ for a group G, the word problem is equivalent to the membership problem for the string language $\mathrm{WP}_X(G) = \{w \in (X \cup X^{-1})^* \mid w =_G 1_G\}$.

A natural question to ask is how hard is the word problem, 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 [15]. It is elementary that a presentation defines a finite group if and only if it admits a regular word problem [2], 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 [13]. In 2015, a major breakthrough of Salvati was published, showing that the word problem of $\mathbb{Z}^2$ is a multiple context-free (MCF) language [19, 18], and in 2018, Ho extended this result showing that all finitely generated virtually Abelian groups admit MCF word problems [10]. This is exciting because the MCF languages are exactly the string languages of hyperedge replacement grammars, which are strictly contained in the context-sensitive string languages [6, 20]. 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 a MCF word problem [7].

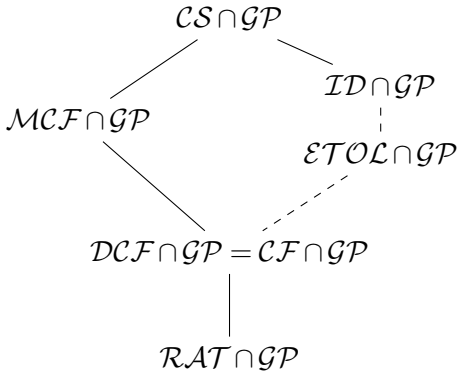
There are of course, lots of other well-behaved language classes sitting in between the context-free and context-sensitive classes, such as the indexed languages [1] or the subclass of ET0L languages [17]. It is not known if there are any groups with indexed word problems other than the virtually free groups, although it has been shown this is the case for a strict subclass of the indexed languages not contained in ET0L [8]. In particular, we don't know if any hyperbolic groups have ET0L word problems [4] (other than the virtually free groups), such as the fundamental group of the double torus. It has been conjectured that all ET0L group languages are only admitted by virtually free groups. We show the (group) language hierarchy below, where necessarily strict inclusion uses a solid line, $\mathcal{GP}$ denotes the class of all group languages (the class of word problems of all finitely generated groups).

*Supported by a Doctoral Training Grant from the Engineering and Physical Sciences Research Council (EPSRC) Grant No. (2281162) in the UK.

Preliminary Report. Final version to appear in:
TERMGRAPH 2020



(a) String Language Hierarchy

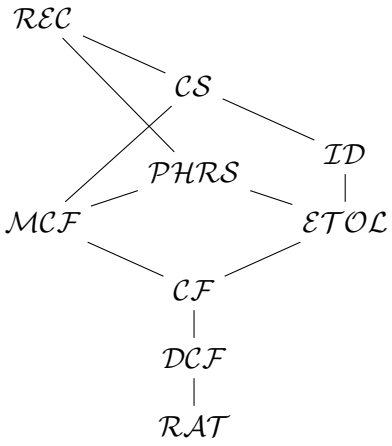


(b) Group Language Hierarchy

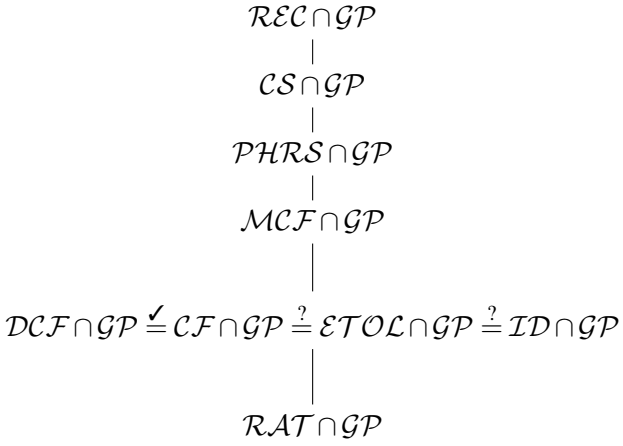
Figure 1: Known Formal Language Hierarchies

The aim of this paper is to lay the foundations for another language class, combining the ideas from ET0L and hyperedge replacement grammars to yield a genuinely new language class: the parallel hyperedge replacement string (PHRS) languages, strictly containing the MCF and ET0L languages. While parallel hyperedge replacement has been considered before [9, 12], the work is not extensive and does not consider rational control or string generational power. Ultimately, our hope is that we might, one day, be able to place the hyperbolic groups within this class. Note that knowledge of (geometric) group theory is not required to read and understand this paper - it is purely motivational!

We summarise the string language hierarchy below, with what we know from results, previous and from this paper, 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 2: New Formal Language Hierarchies

2 Preliminaries

This section is mostly based on the survey [5]. A signature is a pair $\mathcal{C} = (\Sigma, \text{type})$ where Σ is some finite alphabet (we only consider finite alphabets in this paper, but the definition works, in principle, without this restriction), 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. For the rest of paper, we will assume some arbitrary but fixed signature $\mathcal{C} = (\Sigma, \text{type})$. We also denote by $\text{seq}(X)$ ($\text{iseq}(X)$) all finite sequences (finite injective sequences) on a set X.

An n-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; $\text{att}_H : E_H \to \text{seq}(V_H)$ is the attachment function; $\text{lab}_H : E_H \to \Sigma$ is the labelling function; $\text{ext}_H : \text{iseq}(V_H)$ are the external nodes; such that labelling is compatible with typing ($\text{type} \circ \text{lab}_H = |\cdot| \circ \text{att}_H$) and n is the type of H, where in an abuse of notation, we write $\text{type}(H) := |\text{ext}_H|$. We also define $\text{type}_H : E_H \to \mathbb{N}$ by $\text{type}_H := \text{type} \circ \text{lab}_H$, and for all $e \in E_H$, whenever $m = \text{type}_H(e)$ we call e an m-hyperedge, and call a hyperedge $e \in E_H$ proper whenever $\text{att}_H(e)$ is injective. The class of all hypergraphs over $\mathcal{C}$ is denoted $\mathcal{H}_\mathcal{C}$, and $G, H \in \mathcal{H}_\mathcal{C}$ 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 $\text{att}_H \circ g_E = g_V^* \circ \text{att}_G$; $\text{lab}_H \circ g_E = \text{lab}_G$; $g_V \circ \text{ext}_G = \text{ext}_H$.

Given a string $w \in \Sigma^+$ of length n, its string graph is $w^\bullet = (\{v_0, \ldots, v_n\}, \{e_1, \ldots, e_n\}), \text{att}, \text{lab}, v_0 v_n)$ such that $\text{att}(e_i) = v_{i-1} v_i$ and $\text{lab}(e_i) = w(i)$ for all $1 \leq i \leq n$. We call v_0, begin, and v_n, end. We denote the class of all string graphs over $\mathcal{C}$ by $\mathcal{S}_\mathcal{C}$. 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 $X^\bullet = (\{v_1, \ldots, v_n\}, \{e\}, \text{att}, \text{lab}, v_1 \cdots v_n)$ such that $\text{att}(e) = v_1 \cdots v_n$ and $\text{lab}(e) = X$. Note that these two definitions coincide for a type 2 label, considered either as a label or as a string of length 1, thus there can be no confusion.

Let $H \in \mathcal{H}_\mathcal{C}$ be a hypergraph and $B \subseteq E_H$ be a set of hyperedges. Then $\sigma : B \to \mathcal{H}_\mathcal{C}$ 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 $e \in B$; identifying the i-th external node of $\sigma(e)$ with the i-th attachment node of e, for each $e \in B$ and $i \in \underline{\text{type}_H(e)}$, where $\underline{\lambda} = \{1, \ldots, \lambda\}$ for any $\lambda \in \mathbb{N}$. The external nodes of $H[\sigma]$ remain exactly those of H. All hyperedges keep their original attachments and labels.

If $H \in \mathcal{H}_\mathcal{C}$, $B \subseteq E_H$ and $\sigma : B \to \mathcal{H}_\mathcal{C}$, then $H[\sigma]$ exists exactly when σ is a replacement function, and is unique up to isomorphism. Moreover, nodes of H are never deleted or merged. If $H \in \mathcal{H}_\mathcal{C}$, $B = \{e_1, \ldots, e_n\} \subseteq E_H$, $\sigma : B \to \mathcal{H}_\mathcal{C}$ be a replacement function, and $R_i = \sigma(e_i)$ for all $i \in \underline{n}$. Then we write $H[e_1/R_1, \ldots, e_n/R_n]$ in place of $H[\sigma]$.

Let $N \subseteq \Sigma$ be a set of non-terminals. A rule (or production) over N is a pair $p = (L, R)$ with $L \in N$, $R \in \mathcal{H}_\mathcal{C}$, and $\text{type}(L) = \text{type}(R)$. Given $H \in \mathcal{H}_\mathcal{C}$ and $\mathcal{R}$ a set of rules, if $e \in E_H$ and $(\text{lab}_H(e), R) \in \mathcal{R}$, then we say that H directly derives $H' \cong H[e/R]$ (using $\mathcal{R}$), and write $H \Rightarrow_\mathcal{R} H'$. For a given e and choice of rule, H' is unique up to isomorphism. Clearly this is a binary relation on $\mathcal{H}_\mathcal{C}$. We say $H \in \mathcal{H}_\mathcal{C}$ derives H' if there is a sequence $H \Rightarrow_\mathcal{R} H_1 \Rightarrow_\mathcal{R} \cdots \Rightarrow_\mathcal{R} H_k \cong H'$ for some $k \in \mathbb{N}$. We write $H \Rightarrow_\mathcal{R}^k H'$ or $H \Rightarrow_\mathcal{R}^* H'$.

A hyperedge replacement grammar of order k (k-HR grammar) is a system $\mathcal{G} = (\mathcal{C}, N, S, \mathcal{R})$ 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; $\mathcal{R}$ is a finite set of rules over N; with $\max(\{\text{type}(R) \mid (L, R) \in \mathcal{R}\}) \leq k$. We call $\Sigma \setminus N$ the terminal labels. The language generated $\mathcal{G}$ 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 hyperedge replacement language of order k (k-HR language) if there is a k-HR grammar such that $\text{L}(\mathcal{G}) = L$. The class of HR languages is the union of all k-HR languages for $k \in \mathbb{N}$.

Theorem 2.1 (Context-Freeness Lemma [5])**.** Let $\mathcal{R}$ be a finite set of rules over N, and $H \in \mathcal{H}_\mathcal{C}$, $X \in N$ 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 $\sigma : \text{lab}_R^{-1}(N) \to \mathcal{H}_\mathcal{C}$ with $H = R[\sigma]$ such that: $\forall e \in \text{lab}_R^{-1}(N), \text{lab}_R(e)^\bullet \Rightarrow^{k(e)}$; $\sum_{e \in \text{lab}_R^{-1}(N)} k(e) = k$.

Theorem 2.2 (Linear-Growth Theorem [5])**.** 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 that takes a hypergraph, and if it is a string graph, computes the string it represents, is denoted by $\text{STR} : \mathcal{H}_\mathcal{C} \rightharpoonup \Sigma^*$. Clearly this function is total on $\mathcal{S}_\mathcal{C}$, and undefined elsewhere. A

language $L \subseteq \mathcal{H}_\mathcal{C}$ is said to be a string graph language if $L \subseteq \mathcal{S}_\mathcal{C}$. Given a HR grammar $\mathcal{G}$ that generates a string graph language, then we write $\mathrm{STR}(\mathrm{L}(\mathcal{G}))$ for the actual string language it generates. Without loss of generality, we can assume all terminal labels are of type 2. A string language $L \subseteq A^*$ is called a hyperedge replacement string language of order k (k-HRS language) if there is a signature $\mathcal{C} = (\Sigma, \mathrm{type})$ and a grammar k-HR grammar $\mathcal{G}$ over $\mathcal{C}$ such that: $A \subseteq \Sigma$ and $\forall x \in A, \mathrm{type}(x) = 2$; $\mathcal{G}$ generates a string graph language; $\mathrm{STR}(\mathrm{L}(\mathcal{G})) = L \setminus \{\varepsilon\}$. The class of HRS languages is the union of all k-HRS languages for $k \in \mathbb{N}$. As we remarked in the introduction, these are exactly the multiple context-free languages:

Theorem 2.3 (String Generative Power [6, 20])**.** For all $k \geq 1$, $\mathcal{HRS}_{2k} = \mathcal{HRS}_{2k+1} = \mathcal{MCF}_k$.

3 New Results

We start by introducing a parallel version of direct derivations and the notion of a table, borrowed from so-called T0L systems. We then define parallel hyperedge replacement grammars.

Definition 3.1 (Parallel Direct Derivation)**.** Let $H \in \mathcal{H}_\mathcal{C}$ with $E_H = \{e_1, \ldots e_n\}$, and $\mathcal{R}$ be a set of rules. If for each $e_i \in E_H$, there is an $R_i \in \mathcal{H}_\mathcal{C}$ such that $(\mathrm{lab}_H(e_i), R_i) \in \mathcal{R}$, then we say that H parallelly directly derives $H' \cong H[e_1/R_1, \ldots, e_n/R_n]$ (using $\mathcal{R}$), and write $H \Rrightarrow_\mathcal{R} H'$.

Definition 3.2 (Parallel Derivation)**.** Let $H \in \mathcal{H}_\mathcal{C}$, $\mathcal{S} = \{\mathcal{R}_i \mid i \in I\}$ be a finite set of rule sets indexed by I, and $\mathcal{M}$ an FSA over I. Then H ($\mathcal{M}$-)parallelly derives H' (using $\mathcal{S}$) if there is a derivation sequence:

$$\Delta : H \Rrightarrow_{\mathcal{R}_{i_1}} H_1 \Rrightarrow_{\mathcal{R}_{i_2}} \cdots \Rrightarrow_{\mathcal{R}_{i_k}} H_k \cong H'$$

for some $k \in \mathbb{N}$ such that $i_1 i_2 \cdots i_k \in \mathrm{L}(\mathcal{M})$. We write $H \Rrightarrow_\mathcal{S}^\mathcal{M} H'$, $H \Rrightarrow_\mathcal{S}^{i_1 i_2 \cdots i_k} H'$, or $H \Rrightarrow_\mathcal{S}^k H'$.

Definition 3.3 (Table)**.** A table T is a finite set of rules over Σ such that for each $L \in \Sigma$, there is at least one $R \in \mathcal{H}_\mathcal{C}$ such that $(L, R) \in T$.

Compared to hyperedge replacement grammars, rather than having a set of non-terminals, and only replacing non-terinally labelled hyperedges, we allow all hyperedges to be replaced, and have a special set of terminal symbols to allow us to define when it is that a hypergraph is terminally labelled. In fact, the definition of a table ensures that progress can always be made, and each derivation step necessarily replaces all the hyperedges, though not necessarily with something different.

Definition 3.4 (PHR Grammar)**.** A parallel hyperedge replacement grammar of order k (k-PHR grammar) is a system $\mathcal{G} = (\mathcal{C}, A, S, \mathcal{T}, \mathcal{M})$ where $\mathcal{C} = (\Sigma, \mathrm{type})$ is a signature; $A \subseteq \Sigma$ is the set of terminal labels; $S \in \Sigma \setminus A$ is the start symbol; $\mathcal{T} = \{T_i \mid i \in I\}$ is a finite set of tables indexed by I; $\mathcal{M} = (Q, I, \delta, i, F)$ is an FSA over I; with $\max(\{\mathrm{type}(R) \mid (L, R) \in \bigcup_{T_i \in \mathcal{T}} T_i\}) \leq k$. We call $\Sigma \setminus A$ the non-terminal labels. The language generated by $\mathcal{G}$ is $\mathrm{L}(\mathcal{G}) = \{H \in \mathcal{H}_\mathcal{C} \mid S^\bullet \Rrightarrow_\mathcal{T}^\mathcal{M} H \text{ with } \mathrm{lab}_H^{-1}(A) = E_H\} \subseteq \mathcal{H}_\mathcal{C}$. $L \subseteq \mathcal{H}_\mathcal{C}$ is called a parallel hyperedge replacement language of order k (k-PHR language) if there is a k-PHR grammar $\mathcal{G}$ such that $\mathrm{L}(\mathcal{G}) = L$. The class of PHR languages is the union of all k-PHR languages for $k \in \mathbb{N}$.

Proposition 3.5. The PHR languages are closed under hypergraph isomorphism and are homogeneous.

Proof. By induction on the underlying sequential derivation length, where each parallel derivation can be broken down into a sequence of direct derivations. □

The inclusion of rational control, originally considered for ET0L [3], is a convenience to aid with the specification of human understandable grammars and the process of combining grammars. On the surface, one might worry this adds more generative power, making such systems more powerful than that given by Habel [9], but actually the inclusion of rational control does not increase generational power.

Definition 3.6 (PHR Grammar Without Control)**.** A k-PHR grammar without control is a tuple $\mathcal{G} = (\mathcal{C}, A, S, \mathcal{T})$ such that $(\mathcal{C}, A, S, \mathcal{T}, \mathcal{M})$ is a k-PHR grammar where $\mathcal{M}$ is an FSA which accepts everything. Its generated language is defined in the obvious way.

Lemma 3.7 (Control Removal)**.** Given a k-PHR grammar $\mathcal{G}$, one can effectively construct a k-PHR grammar $\mathcal{G}'$ without control such that $\mathrm{L}(\mathcal{G}) = \mathrm{L}(\mathcal{G}')$.

Proof. Let $\mathcal{G} = (\mathcal{C}, A, S, \mathcal{T}, \mathcal{M})$ and w.l.o.g. suppose that $\mathcal{C} = (\Sigma, \mathrm{type})$, $\mathcal{T} = \{T_1, \ldots T_n\}$, and $\mathcal{M} = (Q, \underline{n}, \delta, i, F)$ be deterministic and full with trap state t (see, for example [11]). We make $|Q|$ disjoint copies of Σ, Σ_q for each $q \in Q$. Let $\mathrm{lift}_q : \Sigma \to \Sigma_q$ be defined in the obvious way, for each $q \in Q$, and let $\bar{\Sigma} = \bigcup_{q \in Q} \Sigma_q$. We now construct $\mathcal{G}' = (\mathcal{C}', A, S', \mathcal{T}')$:

1. $\mathcal{C}' = (\Sigma', \mathrm{type}')$ with $\Sigma' = \{S'\} \cup \bar{\Sigma} \cup \Sigma$, $\mathrm{type}'|_{\{S'\}} = (S' \mapsto \mathrm{type}(S))$, $\forall q \in Q, \mathrm{type}'|_{\Sigma_q} = \mathrm{type} \circ \mathrm{lift}_q^{-1}$, and $\mathrm{type}'|_{\Sigma} = \mathrm{type}$;
2. $S' = \mathrm{lift}_i(S)$;
3. $\mathcal{T}' = \{T'_i \mid i \in (\underline{n} \times Q) \cup \{\mathrm{END}\}\}$ where:
   (a) $\forall i \in \underline{n}, q \in Q, T'_{(i,q)} = \mathcal{R} \oplus \{(\mathrm{lift}_q(L), \mathrm{lift}_{\delta(q,i)}(R)) \mid (L,R) \in T_i\}$
   (b) $T'_{\mathrm{END}} = \mathcal{R} \oplus (\{(\mathrm{lift}_f(X), X) \mid f \in F, X \in \Sigma\} \cup \{(\mathrm{lift}_q(X), \mathrm{lift}_t(X)) \mid q \in Q \setminus F, X \in \Sigma\})$;
   where $\mathcal{R} = \{(X, X^\bullet) \mid X \in \Sigma'\}$ and $\oplus$ denotes relational override.

First, we show $\mathrm{L}(\mathcal{G}) \subseteq \mathrm{L}(\mathcal{G}')$. By definition, $G \in \mathrm{L}(\mathcal{G})$ if and only if it is terminally labelled and there is a derivation $\Delta : S^\bullet \Rrightarrow^{i_1}_{\mathcal{T}} G_1 \Rrightarrow^{i_2}_{\mathcal{T}} \cdots \Rrightarrow^{i_l}_{\mathcal{T}} G$ for some $l \in \mathbb{N}$ such that $i_1 i_2 \cdots i_l \in \mathrm{L}(\mathcal{M})$. Notice how the FSA $\mathcal{M}$ starts in the initial state i and then after reading i_1 advances to some state, let's call it q_{i_1}, and then after reading i_2..., and then after reading i_l advances to some state, let's call it q_{i_l}. Now, it must be the case that $q_{i_l} \in F$. Our construction of $\mathcal{G}'$ means we can simulate the derivation $\Delta' : S'^\bullet \Rrightarrow^{(i_1,q_{i_1})}_{\mathcal{T}'} \mathrm{lift}_{q_{i_1}}(G_1) \Rrightarrow^{(i_2,q_{i_2})}_{\mathcal{T}'} \cdots \Rrightarrow^{(i_l,q_{i_l})}_{\mathcal{T}'} \mathrm{lift}_{q_{i_l}}(G) \Rrightarrow^{\mathrm{END}}_{\mathcal{T}'} G$, as required.

Finally, to see $\mathrm{L}(\mathcal{G}') \subseteq \mathrm{L}(\mathcal{G})$, we argue that every derivation in $\mathcal{G}'$ from $S'^\bullet$ to a terminally labelled hypergraph can be transformed into a derivation of the above form, and then we can simply reverse our simulation to give us the result.

So, suppose we have a derivation in $\mathcal{G}'$ from $S'^\bullet$ of length l. If step $i \in \underline{l}$ is an application of $T'_{(i,q)}$ that replaces all edges by themselves, then that step can be deleted. If it is an application of T'_{END} then it follows that all subsequent steps do nothing to the resultant graph. So, we can assume w.l.o.g. that any such derivation is a (possibly empty) sequence of direct derivations using a table $T'_{(i,q)}$ possibly followed by a direct derivation using the table T'_{END}. Now, any derivation sequence that does not end with an application of T'_{END} necessarily must have derived a non-terminally labelled hypergraph, so we can discount such derivation sequences. Next, due to the construction of our tables, any of the prescribed derivation sequences we are left with are exactly of the form from the forward direction of the proof (formally, by induction on derivation length), so have a corresponding derivation in $\mathcal{G}$ from $S^\bullet$. □

Our next theorem confirms that PHR languages strictly contain the HR languages, as expected. We note that our statement is actually stronger than Theorem 3.3 of Habel's book [9], which sketches a proof for only $k \geq 2$. First, we provide an intermediate result (which is used both now, and later on):

Proposition 3.8. $L = \{a^{2^n} \mid n \in \mathbb{N}\}$ is an ET0L language but not MCF. Moreover, it is not semilinear.

Proof. It is easy to see that $\mathcal{G} = (\{a\}, \{a\}, a, \{\{(a, aa)\}\})$ is an ET0L grammar with $\mathrm{L}(\mathcal{G}) = L$. Recall that a language is semilinear if and only if it is letter-equivalent to a regular language [16]. 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. □

Theorem 3.9 (PHR Generalises HR)**.** For $k \geq 0$, $\mathcal{HR}_k \subsetneq \mathcal{PHR}_k$.

Proof. Suppose $\mathcal{C} = (\Sigma, \text{type})$ and $\mathcal{G} = (\mathcal{C}, N, S, \mathcal{R})$ is a k-HR grammar with $\mathcal{R} = \{r_1, \ldots, r_n\}$. Then we construct a k-PHR grammar $\mathcal{G}' = (\mathcal{C}, A, S, \mathcal{T})$ with $A = \Sigma \setminus N$ and $\mathcal{T} = \{T_1\}$ where $T_1 = \mathcal{R} \cup \{(X, X^\bullet) \mid X \in \Sigma\}$. 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, we have that $\mathrm{L}(\mathcal{G}) = \mathrm{L}(\mathcal{G}')$.

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 (Proposition 3.8). We will show there is a 0-PHR language that is not k-HR for any $k \in \mathbb{N}$. Let $\mathcal{G} = (\mathcal{C}, A, S, \mathcal{T})$ be the 0-PHR grammar with $\mathcal{C} = (\{\square\}, \{(\square, 0)\})$, $A = \{\square\}$, $S = \square$, and $\mathcal{T} = \{T_1\}$ where $T_1 = \{(\square, \square^\bullet \sqcup \square^\bullet)\}$ ($\sqcup$ denotes disjoint union of hypergraphs). Clearly $\mathrm{L}(\mathcal{G})$ is the language of hypergraphs over $\mathcal{C}$ with 2^n hyperedges. By Theorem 2.2, $\mathrm{L}(\mathcal{G})$ is not a HR language, as required. □

Corollary 3.10. PHR languages need not have only linear growth, in the sense of Theorem 2.2.

We now turn our attention to string languages. We believe our definition of parallel hyperedge replacement string languages is a genuinely new class of languages containing both the multiple context-free languages and the ET0L languages. It is not simply equal to the (parallel) multiple context-free languages because these are known to be incomparable with ET0L [14]. Recall that the hyperedge replacement string languages are exactly the multiple context-free languages. We will confirm that parallel hyperedge replacement string languages contain all of these and also all of the ET0L languages.

Definition 3.11 (Generated String Language)**.** Given a PHR grammar $\mathcal{G}$ that generates a string graph language, then we write $\mathrm{STR}(\mathrm{L}(\mathcal{G}))$ for the actual string language it generates.

Proposition 3.12. Given a PHR grammar $\mathcal{G} = (\mathcal{C}, A, S, \mathcal{T})$ that generates a string graph language. w.l.o.g. we can assume all terminals are of type 2. Moreover, if the generated language is non-empty, then the start label S must be of type 2. In any case we can assume this w.l.o.g..

Proof. Any symbol labelling an edge of a string graph necessarily has type 2. Any terminally labelled string graph has only terminal symbols labelling its edges. Thus only the type 2 terminal symbols are used to label the terminally labelled string graphs. We can safely delete the other symbols from the terminal set A. For the next part, using the fact that the generated language is non-empty, there is a derivation sequence from $S^\bullet$ to some string graph G. Since G is of type 2, it follows by induction on derivation length (as in the proof of Proposition 3.5) that $S^\bullet$ is of type 2, thus S is of type 2. Finally, to deal with the w.l.o.g. statement, if the generated language is empty, then we can just have no tables and choose S to be a type 2 symbol. □

Definition 3.13 (PHR String Language)**.** A string language $L \subseteq A^*$ is called a parallel hyperedge replacement string language of order k (k-PHRS language) if there is a signature $\mathcal{C} = (\Sigma, \text{type})$ and a grammar k-PHR grammar $\mathcal{G}$ over $\mathcal{C}$ such that: $A \subseteq \Sigma$ and $\forall x \in A, \text{type}(x) = 2$; $\mathcal{G}$ generates a string graph language; $\mathrm{STR}(\mathrm{L}(\mathcal{G})) = L \setminus \{\varepsilon\}$. The class of PHRS languages is the union of all k-PHR languages for $k \in \mathbb{N}$.

Proposition 3.14. $\mathcal{HRS}_0 = \mathcal{HRS}_1 = \mathcal{PHRS}_0 = \mathcal{PHRS}_1 = \{\emptyset, \{\varepsilon\}\}$.

Proposition 3.15. Given a k-PHR grammar $\mathcal{G}$, one can effectively construct a k-PHR grammar $\mathcal{G}'$ such that: all terminals and the start label S are of type 2; and all hyperedges in rules are proper; $\mathrm{L}(\mathcal{G}) = \mathrm{L}(\mathcal{G}')$.

Definition 3.16 (Useless Symbols)**.** Given a PHR grammar $\mathcal{G}$ over $\mathcal{C} = (\Sigma, \text{type})$. Call a symbol $X \in \Sigma$ useful if it occurs as a label of some hypergraph in a derivation from the start hypergraph to any terminally labelled hypergraph.

The following characterisation follows from Theorem 2.1 with not much work:

Proposition 3.17. Given a PHR grammar $\mathcal{G} = ((\Sigma, \text{type}), A, S, \mathcal{T})$. A symbol $X \in \Sigma$ is useful if and only if it is both generating ($\exists n \in \mathbb{N}, \exists G \in \mathcal{H}_{\mathcal{C}}, X^\bullet \Rrightarrow^n G$ and $\text{lab}_G(E_G) \subseteq A$) and reachable ($\exists n \in \mathbb{N}, \exists G \in \mathcal{H}_{\mathcal{C}}, S^\bullet \Rrightarrow^n G$ and $X \in \text{lab}_G(E_G)$).

Moreover, we can iteratively compute the useless symbols in the usual way for context-free grammars (see, for example, Section 7.1 of [11]). Note also how we used the form of grammars without control in that result - it is much more fiddly to define if have to handle control within the derivations.

Proposition 3.18. Given a k-PHR grammar $\mathcal{G}$, one can effectively construct a k-PHR grammar $\mathcal{G}'$ such that there are no useless symbols and $\text{L}(\mathcal{G}) = \text{L}(\mathcal{G}')$.

Next, we have a result about ET0L grammars which we will also need to show Lemma 3.20:

Proposition 3.19. Given an ET0L grammar $\mathcal{G}$, one can effectively construct an ET0L grammar $\mathcal{G}'$) such that all rules necessarily have non-empty RHSs and $\text{L}(\mathcal{G}) \setminus \{\varepsilon\} = \text{L}(\mathcal{G}')$.

Proof. This is due to Theorem 1.6 of Part V of Rozenberg and Salomaa [17] where they show that every ET0L grammar has an equivalent EPT0L grammar, up to treatment of the empty string. □

Lemma 3.20 (PHRS Generalises ET0L)**.** $\mathcal{ETOL} = \mathcal{PHRS}_2$ and for $k \geq 4$, $\mathcal{ETOL} \subsetneq \mathcal{PHRS}_k$.

Proof. First we show $\mathcal{ETOL} \subseteq \mathcal{PHRS}_k$ for all $k \geq 2$. Suppose L is an ET0L language, then by Proposition 3.19 there exists an ET0L grammar $\mathcal{G} = (V, A, S, \{T_i \mid i \in I\})$ such that no rule contains the empty string and $L \setminus \{\varepsilon\} = \text{L}(\mathcal{G})$. It follows that every rule can be encoded as a hyperedge replacement rule over $\mathcal{C}' = (V, V \times \{2\})$ giving us a 2-PHR grammar $\mathcal{G}' = (\mathcal{C}', A, S, \{\{(L, R^\bullet) \mid (L, R) \in T_i\} \mid i \in I\})$ with $\text{L}(\mathcal{G} = \text{STR}(\text{L}(\mathcal{G}'))$.

Next, we show that $\mathcal{PHRS}_2 \subseteq \mathcal{ETOL}$. 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 $L \setminus \{\varepsilon\} = \text{STR}(\text{L}(\mathcal{G}))$. Due to Propositions 3.15 and 3.18, we can assume all the symbols are of type exactly 2, all hyperedges occuring in the RHSs of rules are proper, and that there are no useless symbols in the signature. By induction on derivation length, it is then obvious that any derivation that ends with a string graph must be such that all sub-derivations derive only string graphs. In particular, this means all rules have string graphs as their RHSs. Every such grammar can be converted into an ET0L grammar.

Finally, strictness follows from Theorems 2.3 and 3.9, and the proof of Theorem 8 of [14]. □

Corollary 3.21. There are 2-PHRS languages that are not semilinear.

Proof. Lemma 3.20 gives us a 2-PHRS language which is not semilinear by Proposition 3.8. □

Lemma 3.22 (PHRS Generalises MCF)**.** For $k \geq 2$, $\mathcal{HRS}_k \subsetneq \mathcal{PHRS}_k$.

Proof. This follows from Theorem 2.3, and Theorem 3.9 and its proof. We get strictness from Proposition 3.8 together with Lemma 3.20. □

Conjecture 3.23 (CS Generalises PHRS)**.** $\mathcal{PHRS} \subsetneq \mathcal{CS}$.

We have shown (not included in this paper, using Lemma 3.7) that we have ε-free substitution closure. If we can show full substitution closure and closure under rational intersection, then we have:

Conjecture 3.24 (Substitution-Closed Full AFL)**.** For $k \geq 2$, $\mathcal{PHRS}_k$ and $\mathcal{PHRS}$ are substitution-closed full abstract families of languages.

The following corollary and its contrapositive, to the above conjecture, are extremely important:

Corollary 3.25. For $k \geq 2$, $\mathcal{PHRS}_k$ and $\mathcal{PHRS}$ are closed under inverse homomorphisms. Moreover, if a group has a PHRS word problem for some given presentation, then all presentations necessarily do.

Our final conjecture is wide reaching due to its corollary (assuming Conjecture 3.24 is true):

Conjecture 3.26 (WP Double Torus)**.** The fundamental group of the double torus admits a PHRS word problem which is neither a MCF nor ET0L language.

Corollary 3.27. The word problem of any surface group is a PHRS language.

References

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

[2] Anatoly Anisimov (1971): *Group Languages*. *Kibernetika* 4, pp. 18–24.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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`.

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

[10] 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.

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

[12] Hans-Jörg Kreowski (1993): *Five facets of hyperedge replacement beyond context-freeness*. In Zoltán Ésik, editor: *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.

[13] 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.

[14] 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.

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

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

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

[18] 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.

[19] 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.

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