Technical Report 2014-003
Parallel Communicating Grammar Systems with Context-Free Components Are Turing Complete for Any Communication Model

Mary Sarah Ruth Wilkin and Stefan D. Bruda
Department of Computer Science
Bishop's University
2600 College St
Sherbrooke, Quebec J1M 1Z7, Canada
swilkin@cs.ubishops.ca, stefan@bruda.ca
21 November 2014

Abstract

Parallel Communicating Grammar Systems (PCGS) were introduced as a language-theoretic treatment of concurrent systems. A PCGS extends the concept of a grammar to a structure that consists of several grammars working in parallel, communicating with each other, and so contributing to the generation of strings. PCGS are generally more powerful than a single grammar of the same type; PCGS with context-free components (CF-PCGS) in particular were shown to be Turing complete. However, this result only holds when a specific type of communication (which we call broadcast communication, as opposed to one-step communication) is used. We expand the original construction that showed Turing completeness so that broadcast communication is eliminated at the expense of introducing a significant number of additional, helper component grammars. We thus show that CF-PCGS with one-step communication are also Turing complete. We introduce in the process several techniques that may be usable in other constructions and may be capable of removing broadcast communication in general.

Keywords: formal languages, theory of computation, formal grammar, parallel communicating grammar system, Turing completeness

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

1 Introduction

Parallel Communicating Grammar Systems (PCGS for short) have been introduced as a language-theoretic treatment of concurrent (or more general, multi-agent) systems [19]. A PCGS extends the concept of a grammar to a structure that consists of several grammars working in parallel and contributing to the generation of strings.

In a PCGS one grammar component is considered the master of the system and the other component grammars are called helpers or slaves; they all participate in the derivation but may or may not have a direct impact on the generation of the final string produced by the system. The master grammar controls the derivation which is considered complete as soon as it produces a string of terminals regardless of the state of the strings in the other components (hence the name helper or slave component). In order for the helper components to contribute to the derivation, communication (or query) steps are required. In essence a communication step allows the different components in the system to share strings with one another: A grammar proceeds with a communication step by introducing in its string a request for a string from another grammar. Once a communication step has been introduced, all rewriting steps are put on hold until the communication is complete, meaning they are put on hold until the requesting grammar(s) receive the string from the queried component(s). Grammars communicate in one of two ways: returning or non-returning. In a returning system, once a communication request has been completed the queried component returns to its original axiom and continues the derivation from there; conversely if a system is non-returning the component string remains intact and the derivation continues to rewrite that string [3, 21].

Our main area of interest in this paper is the generative capacity of PCGS. It has been shown that a PCGS with components of a certain type are generally more powerful than single grammars of the same type; we will summarize some results in this respect in Section 3. There have also been other attempts to associate the generative power of PCGS with additional representations, including parse trees [1] and coverability trees [17, 22].

We focus on PCGS with context-free components (CF-PCGS for short). Significant findings in this area include a proof that non-returning PCGS with context free components can generate all recursively enumerable languages [14]. Combined with the fact that non-returning systems can be simulated by returning systems [8] based on an earlier result [15], this result establishes that returning PCGS with context-free components are also computationally complete. An alternative investigation into the same matter consists in the development of a returning PCGS with context-free components that simulates an arbitrary 2-counter Turing machine (yet another complete model [9]), thus proving that this kind of PCGS are Turing complete [5]. On close examination of the derivations of this PCGS simulating a 2-counter machine [5] we noticed that the returning communication steps used are of a particular kind [20]. In this PCGS multiple components query the same component at the same time, and they all receive the same string from the queried component; only then does the queried component returns to its axiom. Throughout the document we will refer to this style of communication as *broadcast communication*. Later work used a different definition, stating that the queried component returns to its axiom immediately after it is communicated [3]; we will refer to this type of communication as *one-step communication*. According to this definition one querying component would receive the requested string and all the other components querying the same component would receive the axiom. One consequence is that the CF-PCGS simulation of a 2-counter Turing machine [5] will not hold with one-step communication, for indeed the proposed system will block after the first communication step.

In this paper we wonder whether the 2-counter Turing machine simulation can be modified so that it works with one-step communication. The answer turns out to be affirmative. We present in Section 4 a PCGS that observes the one-step communication definition and at the same time simulates a 2-counter Turing machine in a similar manner with the original construction [5]. The construction turns out to be substantially more complex. We eliminate broadcast communication using extra components (so that the original broadcast communication is replaced with queries to individual components), which increases the overall number of components substantially. The number of components however remains bounded. We thus conclude that CF-PCGS are indeed Turing complete regardless of the type of communication used.

2 Preliminaries

The symbol ε will be used to denote the empty string, and only the empty string; ω stands for $|\mathbb{N}|$. Given a string σ and a set A we denote the length of σ by $|\sigma|$, while $|\sigma|_A$ stands the length of the string σ after all the symbols not in A have been erased from it. We often write $|\sigma|_a$ instead of $|\sigma|_{\{a\}}$ for singleton sets $A = \{a\}$. The word "iff" stands as usual for "if and only if".

A grammar [13] is a quadruple $G = (\Sigma, N, S, R)$. Σ is a finite nonempty set; the elements of this set are referred to as terminals. N is a finite nonempty set disjoint from Σ; the elements of this set are referred to as nonterminals. $S \in N$ is a designated nonterminal referred to as the start symbol or axiom. R is a finite set of rewriting rules, of the form $\alpha \to \beta$ where $\alpha \in (\Sigma \cup N)^* N (\Sigma \cup N)^*$ and $\beta \in (\Sigma \cup N)^*$ (α

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

and β are strings of terminals and nonterminals but α has at least one nonterminal). Given a grammar G, the $\Rightarrow_G$ (yields in one step) binary operator on strings from the alphabet $W = (\Sigma \cup N)^*$ is defined as follows: $T_1 A T_2 \Rightarrow_G T_1 u T_2$ if and only if $A \to u \in R$ and $T_1, T_2 \in (\Sigma \cup N)^*$. We often omit the subscript from the yields in one step operator when there is no ambiguity. The language generated by a grammar $G = (\Sigma, N, S, R)$ is $\mathscr{L}(G) = \{w \in \Sigma^* : S \Rightarrow_G^* w\}$, where $\Rightarrow_G^*$ denotes as usual the reflexive and transitive closure of $\Rightarrow_G$.

Languages generated by an (unrestricted) grammar G are referred to as recursively enumerable (RE). G is called a context-sensitive grammar if each rewriting rule $\alpha \to \beta$ in R satisfies $|\alpha| \le |\beta|$; the languages generated by these grammars are referred to as context sensitive (CS). G is context free if every rewriting rule $\alpha \to \beta$ in R satisfies $|\alpha| = 1$ (meaning that α is a single nonterminal); these grammars generate context-free languages (CF). A special type of context-free grammars are linear grammars, generating linear languages (LIN), where no rewriting rule is allowed to have more that one non-terminal symbol on its right hand side. Finally, grammars are regular and generate regular languages (REG) if their rewriting rules have one of the following forms: $A \to cB$, $A \to c$, $A \to \varepsilon$, or $A \to B$ where A,B are nonterminals and c is a terminal [10, 12].

A Parallel Communicating Grammar System (or PCGS) provides a theoretical prototype that combines the concepts of grammars with parallelism and communication. This allows for the examination of the properties of parallel systems. The structure of a PCGS is similar to a basic grammar in the sense that all components of a PCGS have the characteristics that allow them to be classified in the Chomsky hierarchy. The major difference between a grammar and a PCGS is that a PCGS features more than one component grammar, and the component grammars of a PCGS work together to generate the resulting language instead of generating languages on their own [3, 21].

Definition 1. PARALLEL COMMUNICATING GRAMMAR SYSTEM [3]: *Let $n \ge 1$ be a natural number. A PCGS of degree n is an $(n+3)$ tuple $\Gamma = (N, K, T, G_1, \ldots, Gn)$ where N is a nonterminal alphabet, T is a terminal alphabet, and K is the set of query symbols, $K = \{Q_1, Q_2, \ldots, Q_n\}$. The sets N, T, K are mutually disjoint; let $V_\Gamma = N \cup K \cup T$. $G_i = (N \cup K, T, R_i, S_i), 1 \le i \le n$ are Chomsky grammars. The grammars $G_i, 1 \le i \le n$, represent the components of the system. The indices $1, \ldots, n$ of the symbols in K point to $G_1, \ldots, G_n$, respectively.*

A derivation in a PCGS consists of a series of communication and rewriting steps. A rewriting step is not possible if communication is requested (which happens whenever a query symbol appears in one of the components of a configuration).

Definition 2. DERIVATION IN A PCGS [3]: *Let $\Gamma = (N, K, T, G_1, \cdots, Gn)$ be a PCGS as above, and $(x_i, x_2, \ldots, x_n)$ and $(y_i, y_2, \ldots, y_n)$ be two n-tuples with $x_i, y_i \in V_\Gamma^*$, $1 \le i \le n$. We write $(x_i, \ldots, x_n) \Rightarrow (y_i, \ldots, y_n)$ iff one of the following two cases holds:*

1. *$|x_i|_K = 0, 1 \le i \le n$, and for each $i, 1 \le i \le n$, we have $x_i \Rightarrow_{G_i} y_i$ (in the grammar G_i), or $x_i \in T^*$ and $x_i = y_i$.*

2. *$|x_i|_K > 0$ for some $1 \le i \le n$; let $x_i = z_1 Q_{i_1} z_2 Q_{i_2} \ldots z_t Q_{i_t} z_{t+1}$, with $t \ge 1$ and $z_j \in (N \cup \Sigma)^*$, $1 \le j \le t+1$. Then $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$ [and $y_{i_j} = S_{i_j}$, $1 \le j \le t$] whenever $|x_{i_j}|_K = 0$, $1 \le j \le t$. If on the other hand $|x_{i_j}|_K \ne 0$ for some $1 \le j \le t$, then $y_i = x_i$. For all $1 \le k \le n$, $y_k = x_k$ whenever y_k was not specified above.*

The presence of "[and $y_{i_j} = S_{i_j}, 1 \le j \le t$" in the definition makes the PCGS returning. *The PCGS is* non-returning *if the phrase is eliminated.*

We use $\Rightarrow$ for both component-wise and communication steps, but we also use (sparingly) $\stackrel{\Lambda}{\Rightarrow}$ for communication steps whenever we want to emphasize that a communication takes place. A sequence

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

of interleaved rewriting and communication steps will be denoted by $\Rightarrow^*$, the reflexive and transitive closure of $\Rightarrow$.

In other words, an n-tuple $(x_1,\ldots,x_n)$ yields $(y_1,\ldots,y_n)$ if:

1. If there is no query symbol in $x_1,\ldots,x_n$, then we have a component-wise derivation ($x_i \Rightarrow_{G_i} y_i$, $1 \le i \le n$, which means that one rule is used per component G_i), unless x_i is all terminals ($x_i \in T^*$) in which case it remains unchanged ($y_i = x_i$).

2. If we have query symbols then a communication step is required. When this occurs each query symbol Q_j in x_i is replaced by x_j, if and only if x_j does not contain query symbols. In other words, a communication step involves the query symbol Q_j being replaced by the string x_j; the result of this replacement is referred to as Q_j being *satisfied* (by x_j). Once the communication step is complete the grammar G_j continues processing from its axiom, unless the system is non-returning. Communication steps always have priority over rewriting steps; if not all query symbols are satisfied during a communication step, they will be satisfied during the next communication step (as long as the replacement strings do not contain query symbols).

The derivation in a PCGS can be blocked in two ways [3, 21, 16, 18]:

1. if component x_i of the current n-tuple $(x_1,\ldots,x_n)$ does not contain a nonterminal that can be rewritten in G_i, or

2. if a circular query appears; in other words if G_{i_1} queries Q_{i_2}, G_{i_2} queries Q_{i_3}, and so on until $G_{i_{k-1}}$ queries Q_{i_k} and G_{i_k} queries Q_{i_1}, then a derivation will not be possible since the communication step always has priority, but no communication is possible because only strings without query symbols can be communicated.

Definition 3. LANGUAGES GENERATED BY PCGS [3]: *The language generated by a PCGS Γ is $\mathscr{L}(\Gamma) = \{w \in T^* : (S_1,S_2,...,S_n) \Rightarrow^* (w,\sigma_2,...,\sigma_n), \sigma_i \in V_\Gamma^*, 2 \le i \le n\}$.*

The derivation starts from the tuple of axioms $(S_1,S_2,...,S_n)$. A number of rewriting and/or communication steps are performed until G_1 produces a terminal string (we do not restrict the form of, or indeed care about the rest of the components of the final configuration).

As with any model certain behaviors have been defined in semantic terms to simplify their description. These terms will be used frequently in what follows.

Definition 4. PCGS SEMANTICS [21]: *A PCGS Γ is called centralized if there is a restriction that only the first component grammar G_1 can control the communication, meaning that only G_1 can introduce query symbols. If on the other hand any component grammar G_i can coordinate communications steps, meaning any component grammar can introduce communication symbols, then the system is non-centralized.*

A returning system refers to the component grammars returning to their respective axiom after a communication step. If on the other hand the component grammars do not return to their respective axioms but continue to process the current string after communicating then the PCGS is considered to be a non-returning system.

A system can be synchronized whenever a component grammar uses exactly one rewriting rule per derivation step (unless the component grammar is holding a terminal string, case in which it is allowed to wait). If a system is non-synchronized then in any step that is not a communication step the component may chose to rewrite or wait.

The family of languages generated by a non-centralized, returning PCGS with n components of type X (where X is an element of the Chomsky hierarchy) will be denoted by $PC_n(X)$. The language families

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

generated by centralized PCGS will be represented by $CPC_n(X)$. The fact that the PCGS is non-returning will be indicated by the addition of an N, thus obtaining the classes $NPC_n(X)$ and $NCPC_n(X)$. Let M be a class of PCGS, $M \in (PC, CPC, NPC, NCPC)$; then we define:

$$M(X) = M_*(X) = \bigcup_{n \geq 1} M_n(X)$$

3 Previous Work

We start by summarizing the existing results regarding the generative capacity of the most commonly studied PCGS. One will notice that not all structural variations have been studied in this respect. Most of the existing results are about centralized systems, and even then not all of the centralized variants have been studied thoroughly. As mentioned previously PCGS are more powerful than grammars of the same type.

CS and RE are the two most powerful PCGS and grammar types. Surprisingly their behavior is quite similar, as shown below. We start with the immediate finding that a RE grammar is just as powerful as a PCGS with RE components. Due to this the PCGS of this type with n components are not very interesting since they are just as powerful as a PCGS with one component. In other words a PCGS with unrestricted components are Turing equivalent and are just as powerful as RE grammars: $\mathrm{RE} = Y_n(\mathrm{RE}) = Y_*(\mathrm{RE})$, $n \geq 1$, for all $Y \in \{PC, CPC, NPC, NCPC\}$ [3].

The same holds to some degree for PCGS with context-sensitive components versus context-sensitive languages: $CS = Y_n(CS) = Y_*(CS)$, $n \geq 1$, for $Y \in \{CPC, NCPC\}$ [3]. Note that this result describes the centralized case. We would expect that the non-centralized case to be more powerful, so presumably this result does not hold in the non-centralized case. One should note that PCGS with CS components are computationally expensive, which limits their usefulness. As is the case with normal grammars, the most useful classes are the simple ones. The results in the area of PCGS with regular or context-free components are therefore much more interesting.

The following result shows that the class of languages generated by a centralized returning PCGS with regular components is a subset of the class of languages generated by a non-centralized, returning PCGS with regular components. This indicates that the generative power of a PCGS is greater than of a single grammar component, and that the more communication facilities we have the more powerful the resulting system is: $CPC_n(\mathrm{REG}) \subsetneq PC_n(\mathrm{REG})$, $n > 1$ [21].

A similar result was found for PCGS with context free components; however in this case increased communication may not make the system more powerful: $CPC_*(\mathrm{CF}) \subseteq PC_*(\mathrm{CF})$ [7].

We note in general that the centralized variant is a particular case of a non-centralized PCGS. Indeed, that centralized qualifier restricts the initiation of the communication to the first grammar in the system. As a consequence the class of languages generated by a centralized PCGS of any type can be generated by a non-centralized PCGS of the same type: $CPC_n(X) \subseteq PC_n(X)$ for any $n \geq 1$. The generative power of a PCGS is greater that of a single grammar component because communication, and once this parameter is restricted, the generative power is also restricted.

The following two results further demonstrate that there are limitations to the generative power of PCGS. When we have only two regular components the languages generated by centralized PCGS are all context free. Even the non-centralized variant is limited to generating context-free languages: $CPC_2(\mathrm{REG}) \subsetneq \mathrm{CF}$ and $PC_2(\mathrm{REG}) \subseteq \mathrm{CF}$ [3].

Another way to increase the generative power of a system is to increase the number of components in the system. We have shown that this does not change the generative capacity in the RE and (to some degree) CS case. However if we examine classes that are lower in the hierarchy we notice that an increase in the number of components generally increases the generative capacity of the system [3].

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

1. There exists a language generated by PCGS with 2 or more REG components that cannot be generated by a linear grammar: $Y_n(\text{REG}) \setminus \text{LIN} \neq \emptyset$ for $n \geq 2$, $Y \in \{PC, CPC, NPC, NCPC\}$.

2. There exists a language generated by a PCGS with 3 or more REG components that cannot be generated by a context free grammar: $Y_n(\text{REG}) \setminus \text{CF} \neq \emptyset$ for $n \geq 3$ (and $n \geq 2$ for non-returning PCGS), $Y \in \{PC, CPC, NPC, NCPC\}$.

3. There exists a language generated by a PCGS with 2 or more linear components that cannot be generated by a context free grammar: $Y_n(\text{LIN}) \setminus \text{CF} \neq \emptyset$, $n \geq 2$, $Y \in \{PC, CPC, NPC, NCPC\}$.

4. There exists a language generated by a non-returning PCGS with 2 or more regular components that cannot be generated by a context free grammar: $Y_n(\text{REG}) \setminus \text{CF} \neq \emptyset$, $n \geq 2$, $Y \in \{NPC, NCPC\}$.

Obviously an increase in the power of the components will generally increase the power of a PCGS. This holds strictly in the centralized case for REG versus LIN versus CF components: $CPC_n(\text{REG}) \subsetneq CPC_n(\text{LIN}) \subsetneq CPC_n(\text{CF})$, $n \geq 1$, [3]. Presumably the same relationship would hold for the non-centralized case, but this has not been investigated.

We already mentioned the number of components as an important factor in the generative power of PCGS. It therefore makes sense to consider the hierarchies generated by this factor. Some of these hierarchies are in fact infinite, namely $CPC_n(\text{REG})$ and $CPC_n(\text{LIN})$, $n \geq 1$ [3].

Some hierarchies however collapse. We have already mentioned that $CPC_n(CS)$ and $NCPC_n(CS)$, $n \geq 1$, do not give infinite hierarchies, for all of these classes coincide with CS. Lower classes also produce collapsing hierarchies; for instance non-centralized CF-PCGS with 11 components can apparently generate the whole class of RE languages [5]:

$$\text{RE} = PC_{11}\text{CF} = PC_*\text{CF}. \tag{1}$$

A later paper found that a CF-PCGS with only 5 components can generate the entire class of RE languages by creating a PCGS that has two components that track the number of non-terminals and use the fact that for each RE language L there exists and Extended Post Correspondence problem P [11] such that $\mathscr{L}(P) = L$. [4]:

$$\text{RE} = PC_5\text{CF} = PC_*\text{CF}. \tag{2}$$

There have also been other papers that have examined the size complexity of returning and non returning CF systems even further. It has been shown that every recursively enumerable language can be generated by a context fee returning PCGS, where the number of nonterminals in the system is less than or equal to a natural number k [2]. It has also been shown that non-returning CF-PCGS can generate the set of recursively enumerable languages with 6 context free components by simulating a 2-counter Turning machine [6].

We will show however in Section 3.1 that the above results [5, 4, 2] use *broadcast communication* which modifies the power of a system when compared to *one-step communication*. We will also show (Section 4) that the hierarchy $PC_*\text{CF}$ does collapse irrespective of the communication model being used (though not necessarily at $n = 11$ or $n = 5$).

Turing completeness was also shown for non-returning systems [14, 6]. In particular, if $k \geq 2$ and $L \subseteq \{a_1, \ldots, a_k\}^+$ is a recursively enumerable language, then there exists a non-returning CF-PCGS without ε-rules (meaning without rules of the form $A \to \varepsilon$) that generates L [6]. If we consider that non-returning systems can be simulated by returning systems via the help of assistance grammars holding intermediate strings [8], these results [14, 6] also apply to returning systems (though the number of components necessary for this to happen does not remain the same).

3.1 Broadcast Communication and the Turing Completeness of CF-PCGS

Recall that two different types of communication for returning PCGS were introduced in Section 1: broadcast and one-step communication. In broadcast communication the queried component retains its string until all components requesting that string have received copies of it. Once this process is complete the queried component returns to the axiom. This is different from a one-step returning system where the queried component returns to the axiom immediately after being queried, regardless of the number of components that are requesting a copy of its string.

Evidently, the type of communication step used in returning system has a direct impact on the generative power of a PCGS. Consider for example a PCGS Γ with the following sets of rewriting rules for the master and the two slave components, respectively:

$$\{S \to aS, S \to Q_2, S \to Q_3, S_1 \to b, S_2 \to c, S \to \varepsilon\}$$
$$\{S_1 \to bS_1, S_1 \to Q_3, S_2 \to c\}$$
$$\{S_2 \to cS_2, S_2 \to Q_2, S_1 \to b\}$$

The following is an example of a possible derivation with broadcast communication in Γ:

$$(S, S_1, S_2) \Rightarrow (aS, bS_1, cS_2,) \Rightarrow (aQ_2, bbS_1, cQ_2) \overset{\Lambda}{\Rightarrow} (abbS_1, S_1, cbbS_1) \Rightarrow (abbb, bS_1, cbbb),$$

(recall that the superscript Λ denotes a communication step). We note that in this example the second component is queried by both the other two components. Both querying components receive copies of the same string and then the second component returns to its axiom.

Here is another example of a possible derivation of Γ but this time using one-step communication:

$$(S, S_1, S_2) \Rightarrow (aS, bS_1, cS_2,) \Rightarrow (aQ_2, bbS_1, cQ_2) \overset{\Lambda}{\Rightarrow} (aS_1, S_1, cbbS_1) \Rightarrow (ab, bS_1, cbbb)$$

In this last case the third component was nondeterministically chosen to be the initial component to receive a string from the second component (bbS_1). Once communicated, the string of the second component was reset to the respective axiom, which was then communicated to the first component (which thus receives S_1).

The derivation that used broadcast communication steps generated the string *abbb*, whereas the derivation that followed the rules of a returning system generated *ab*. The different strings were obtained despite the use of the same rewriting rules, and same rewriting steps. It is therefore clear that the use of different styles of communication has a direct impact on the strings generated by a PCGS that is, the languages produced by the system.

This difference in communication steps is what causes us to call into question the result shown in Equation (1) [5]. Indeed, the proof that led to this result hinges on the use of broadcast communication steps. This approach to communication was also used in other related papers [4, 2], though we will focus on what was chronologically the first result in this family [5]. The sets of rewriting rules of the PCGS used in the proof of this result [5] are shown in Figure 1.

A derivation in this system begins with the initial configuration described below, then takes its first step which results in a nondeterministic choice.

$$\begin{array}{ll}(S, S_1, S_2, S_3, S_4, S_1, S_2, S_3, S_4, S, S) & \Rightarrow \\ ([I], u_1, u_2, u_3, S_4^{(1)}, u_1', u_2', u_3', S_4, Q_m, S^{(3)}) & \end{array}$$

As explained in the original paper u_1, u_2, u_3 are either Q_m or $Q_4^{c_1}$ and u_1', u_2', u_3' are either Q_m or $Q_4^{c_2}$. At this stage if any of the symbols are $Q_4^{c_1}$ or $Q_4^{c_2}$ the system will block, so the only successful rewriting step is:

$$\begin{array}{ll}(S, S_1, S_2, S_3, S_4, S_1, S_2, S_3, S_4, S, S) & \Rightarrow \\ ([I], Q_m, Q_m, Q_m, S_4^{(1)}, Q_m, Q_m, Q_m, S_4^{(1)}, Q_m, S^{(3)}) & \end{array}$$

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\begin{aligned}
P_{GM_{Original}} &= \{S \to [I], [I] \to C, C \to Q_{a_1}\} \cup \\
&\quad \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
&\quad \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
&\quad \{< x,q,c_1',c_2',e_1',e_2' > \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
&\quad x \in \Sigma, c_1',c_2' \in \{Z,B\}, e_1',e_2' \in \{-1,0,+1\}\} \cup \\
&\quad \{< x,q,c_1',c_2',e_1',e_2' > \to x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c_1',c_2',e_1',e_2' > \to x| \\
&\quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c_1',c_2' \in \{Z,B\}, \\
&\quad e_1',e_2' \in \{-1,0,+1\}, x,y \in \Sigma\}, \\
P_1^{c_1} &= \{S_1 \to Q_m, S_1 \to Q_4^{c_1}, C \to Q_m\} \cup \\
&\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_1]', [+1]' \to AAC, [0]' \to AC, [-1]' \to C| \\
&\quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\} \cup \\
&\quad \{[I] \to [I]', [I]' \to AC\}, \\
P_2^{c_1} &= \{S_2 \to Q_m, S_2 \to Q_4^{c_1}, C \to Q_m, A \to A\} \cup \\
&\quad \{[x,q,Z,c_2,e_1,e_2] \to [x,q,Z,c_2,e_1,e_2], [I] \to [I] | x \in \Sigma, q \in E, \\
&\quad c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\} \\
P_3^{c_1} &= \{S_3 \to Q_m, S_3 \to Q_4^{c_1}, C \to Q_m\} \cup \\
&\quad \{[x,q,Z,c_2,e_1,e_2] \to a, [x,q,B,c_2,e_1,e_2] \to [x,q,B,c_2,e_1,e_2] \\
&\quad [I] \to [I] | x \in \Sigma, q \in E, c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\} \\
P_4^{c_1} &= \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}, S_4^{(2)} \to Q_1^{c_1}, A \to a\} \\
P_1^{c_2} &= \{S_1 \to Q_m, S_1 \to Q_4^{c_2}, C \to Q_m\} \cup \\
&\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C| \\
&\quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\} \cup \\
&\quad \{[I] \to [I]', [I]' \to AC\} \\
P_2^{c_2} &= \{S_2 \to Q_m, S_2 \to Q_4^{c_2}, C \to Q_m, A \to A\} \cup \\
&\quad \{[x,q,c_1,Z,e_1,e_2] \to a, [x,q,c_1,B,e_1,e_2] \to [x,q,c_1,B,e_1,e_2], \\
&\quad [I] \to [I] | x \in \Sigma, q \in E, \\
&\quad c1 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\} \\
P_3^{c_2} &= \{S_3 \to Q_m, S_3 \to Q_4^{C_2}, C \to Q_m\} \cup \\
&\quad \{[x,q,c_1,Z,e_1,e_2] \to a, [x,q,c_1,B,e_1,e_2] \to [x,q,c_1,B,e_1,e_2] \\
&\quad [I] \to [I] | x \in \Sigma, q \in E, c1 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\} \\
P_4^{c_2} &= \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}, S_4^{(2)} \to Q_1^{c_2}, A \to a\} \\
P_{a_1} &= \{S \to Q_m, [I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
&\quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, , I > \to < I >, | x \in \Sigma, \\
&\quad q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\} \\
P_{a_2} &= \{S \to S^3, S^{(1)} \to S^{(2)}, S^{(2)} \to S^{(3)}, , S^{(3)} \to S^{(4)}, \\
&\quad S^{(4)} \to Q_2^{c_1} Q_3^{c_1} Q_2^{c_2} Q_3^{c_2} S^{(1)}\}.
\end{aligned}
$$

Figure 1: A CF-PCGS with broadcast communication that simulates a 2-counter Turing machine [5].

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

We will now proceed with the broadcast communication step. Notice that all occurrences of the symbol Q_m are replaced with the symbol $[I]$, and all of the components that receive $[I]$ have a corresponding rewriting rule for it:

$$([I], Q_m, Q_m, Q_m, S_4^{(1)}, Q_m, Q_m, Q_m, S_4^{(1)}, Q_m, S^{(3)}) \quad \Rightarrow \\ (S, [I], [I], [I], S_4^{(1)}, [I], [I], [I], S_4^{(1)}, [I], S^{(3)})$$

Should we have used one-step communication the behavior of the system would have been quite different. The initial Q_m symbol (chosen nondeterministically), would be replaced with the symbol $[I]$ from the master grammar, and all the other components that communicate with the master would receive axiom S since the master will return to the axiom before any of the other components had a chance to query it.

$$([I], Q_m, Q_m, Q_m, S_4^{(1)}, Q_m, Q_m, Q_m, S_4^{(1)}, Q_m, S^{(3)}) \quad \Rightarrow \\ (S, [I], S, S, S_4^{(1)}, S, S, S, S_4^{(1)}, S, S^{(3)})$$

We see again a notable difference in the different communication models. Indeed, if broadcast communication steps are not used then the derivation blocks since the returning communication step yields a configuration where all but one of the components $P_1^{c_1}$, $P_2^{c_1}$, $P_3^{c_1}$, $P_4^{c_1}$, $P_1^{c_2}$, $P_2^{c_2}$, $P_3^{c_2}$, and $P_4^{c_2}$ get a copy of the master grammar axiom S, yet none of them have a rewriting rule for S. Since we also know that if any of the components rewrite to $Q_4^{c_1}$ or $Q_4^{c_2}$ the system will block, it becomes clear that broadcast communication steps are essential for the original proof [5] to hold.

This being said, we will discuss in Section 4 how a form of this result does hold even in the absence of broadcast communication.

4 CF-PCGS Are Really Turing Complete

We are now ready to show that PCGS with context-free components are Turing complete even when broadcast communication is replaced with one-step communication. As discussed earlier (Section 3.1), broadcast communication steps are critical in the constructions used in earlier proofs of this result [5, 4, 2]. If we attempt to use the same construction with one-step returning communication the derivation will block. Nonetheless we are able to modify the original construction and eliminate the need for broadcast communication. The resulting system is considerably more complex and so our result is slightly weaker, but it shows that the result holds regardless of the communication model used.

Overall we have the following:

Theorem 1. $\text{RE} = \mathscr{L}(PC_{95}\text{CF}) = \mathscr{L}(PC_*\text{CF})$.

The remainder of this section is dedicated to the proof of Theorem 1. Specifically, we show the inclusion $\text{RE} \subseteq \mathscr{L}(PC_{95}\text{CF})$. Customary proof techniques demonstrate that $\mathscr{L}(PC_*\text{CF}) \subseteq \text{RE}$ and consequently $\mathscr{L}(PC_{95}\text{CF}) \subseteq \mathscr{L}(PC_*\text{CF}) \subseteq \text{RE}$. We describe first the PCGS simulating the Turing machine (Section 4.1) and we then describe how the simulation is carried out (Section 4.2).

The proof is comparable to the one developed earlier [5], in that we use a CF-PCGS to simulate an arbitrary 2-counter Turing machine. We use all of the components used originally in their construction, but with modified labels. However, we follow the definition of one-step communication, so we have to ensure that the components can work together under one-step communication without stumbling over each other. In order to do this we add many copycat components, giving them new labels and slightly different rewriting rules than the original component grammars; their job is to create and hold intermediate strings throughout the derivation. For the most part the intermediate strings that these components

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

hold are replicas of the original component strings, which allows every component grammar to communicate with its own respective copycats, and so receive the same string as in the original construction even in the absence of broadcast communication. We also add components to the system whose job is to fix synchronization issues by resetting their matching helpers at specific points in the derivation. Finally in order to avoid the generation of undesired strings we use blocking to our advantage by ensuring that any inadvertent communication that does not contribute to a successful simulation will introduce nonterminals that will subsequently cause that derivation to block.

4.1 A PCGS that Simulates a 2-Counter Turing Machine

Let $M = (\Sigma \cup \{Z,B\}, E, R)$ be a 2-counter Turing machine [9] that accepts some language L. M has a tape alphabet $\Sigma \cup \{Z,B\}$, a set of internal states E with q_0, $q_F \in E$ and a set of transition rules R. The 2-counter machine has a read only input tape and two counters that are semi-infinite storage tapes. The alphabet of the storage tapes contains two symbols Z and B, while the input tape has the alphabet $\Sigma \cup \{B\}$. The transition relation is defined as follows: if $(x,q,c_1,c_2,q',e_1,e_2,g) \in R$ then $x \in \Sigma \cup \{B\}$, $q,q' \in E$, $c_1,c_2 \in \{Z,B\}$, $e_1,e_2 \in \{-1,0,+1\}$, and $g \in \{0,+1\}$. The starting and final states of M are denoted by q_0 and q_F, respectively.

Intuitively, a 2-counter Turing machine has an input tape which is read only and unidirectional, as well as two read-write counter tapes. The counter tapes (just counters henceforth) are initialized with zero by placing the symbol Z on their leftmost cell, while the rest of the cells contain the symbol B. A counter stores an integer i by having the head of the respective tape moved i positions to the right of the cell containing the Z symbol. A counter can be incremented or decremented by moving the head to the right or to the left, respectively; it is an error condition to move the head to the left of a cell containing Z (that is, decrement a counter which holds a zero value). One can only test whether the counter holds a zero value or not by inspecting the symbol currently under the head (with is Z for a zero and B otherwise).

A transition of the 2-counter machine $(x,q,c_1,c_2,q',e_1,e_2,g) \in R$ is then enabled by the current state q, the symbol currently scanned on the input tape x, and the current value of the two counters c_1 and c_2 (which can be either Z for zero or B for everything else). The effect of such a transition is that the state of the machine is changed to q'; the counter $k \in \{1,2\}$ is decremented, unchanged, or incremented whenever the value of e_k is -1, 0, or $+1$, respectively; and the input head is advanced if $g = +1$ and stays put if $g = 0$. When the input head scans the last non-blank symbol on the input tape and the machine M is in the accepting state q_F then the input string is accepted by the machine. $\mathscr{L}(M)$ be the language of exactly all the input strings accepted by M.

We will now construct the following grammar system with 95 components that generates L by simulating the 2-counter Turing machine that accepts it:

$$\begin{aligned}\Gamma \quad = \quad & (N,K,\Sigma \cup \{a\}, G_{m_{original}}, \dots G_{m_{29}}, G^{C_1}_{P_1}, \dots, G^{C_1}_{P_{1_5}}, G^{c_1}_{P_2}, G^{c_1}_{P_3}, G^{C_1}_{P_1}, \dots \\ & G^{C_1}_{P_{1_5}}, G^{c_2}_{P_2}, G^{c_2}_{P_3}, G_{Pa_1} \dots, G_{Pa_{1_{15}}} G_{a_2}, G^{14}_{resetGM_{Pa1}}, G^{4}_{resetP_1} \dots G^{4}_{resetP_4})\end{aligned}$$

where

$$\begin{aligned}N \quad = \quad & \{[x,q,c_1,c_2,e_1,e_2],[e_1]',[e_2]',[I],[I]',<I>,<x,q,c_1,c_2,e_1,e_2> \mid \\ & x \in \Sigma, q \in E, C_1, c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\} \cup \\ & \{S,S_1,S_2,S_3,S_4,S_4^{(1)},S_4^{(2)},S^{(1)},S^{(2)},S^{(3)},S^{(4)}\} \cup \{A,C\}\end{aligned}$$

and the rewriting rules sets are defined later. Note that all of the component definitions from the original system have the word *original* in their label in order to differentiate them from the helper grammars that were added in order to accommodate the requirements of a on step-communication (returning) system.

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

In order for our construction to hold it is enough for the grammars that represent the original components to terminate the derivation with the same strings as in the original 11-component derivation. The components defined as "original" will work with the Turing machine M simulating the steps of M in their derivation. The system will change its configuration in sync with the state of M and according to the value of the string derived so far in the master component (which will correspond at the end of the derivation with an input accepted by M).

We now describe the rewriting rules of the component grammars. We use the symbols Q_l as usual to identify communication requests, but for clarity the label l will no longer be purely numerical. Most components are modifications of components in the original 11-component construction, so we group the newly introduced rules in sets labelled $\mathfrak{N}$. In most cases new rules have label(s) modified to match the components they are designed to work with; in some cases the rewriting rule themselves are changed. Those components that do not have an equivalent in the original construction have all their rules in the set $\mathfrak{N}$.

The new master contains the same rewriting rules and communications steps as it had in the original construction [5]. The primary role of the master is to maintain its relationship with the P_{a_1} component grammar. The other component definitions that follow the new master are *helper grammars* designed to copy the functionality of the master; they have been added to the system to handle queries from $P_1^{c_1}$, $P_2^{c_1}$, $P_3^{c_1}$, $P_4^{c_1}$, $P_1^{c_2}$, $P_2^{c_2}$, $P_3^{c_2}$, and $P_4^{c_2}$ (these components will all be described in detail later). In essence we ensure that every component grammar $P_1^{c_1}$, $P_2^{c_1}$, $P_3^{c_1}$, $P_4^{c_1}$, $P_1^{c_2}$, $P_2^{c_2}$, $P_3^{c_2}$, or $P_4^{c_2}$ that can query the master grammar in the original broadcast construction has a matching helper grammar that can handle their communication requests.

$$
\begin{aligned}
P_{GM_{Original}} \quad = \quad & \{S \to [I], [I] \to C, C \to Q_{a_1}\} \cup \\
& \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c_1',c_2',e_1',e_2' > \to [x,q',c_1,c_2,e_1,e_2] | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,0) \in R, x \in \Sigma, c_1',c_2' \in \{Z,B\}, \\
& \quad e_1',e_2' \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c_1',c_2',e_1',e_2' > \to x[y,q',c_1,c_2,e_1,e_2], \\
& \quad < x,q_F,c_1',c_2',e_1',e_2' > \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c_1',c_2' \in \{Z,B\}, e_1',e_2' \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}
$$

The following 5 helper grammars simulate rules from the new master but each component is designed to work with different components in $P_1^{c_1}$, including the $P_{1_{originalS_1}}^{c_1}$ grammar and its four newly defined helpers. The components below work with the $P_1^{c_1}$ grammars as the single grammar version would have in the original construction but the labels of the query symbols have been modified to reflect the labels of their matching component grammar.

$$
\begin{aligned}
P_{GM_{S_1}}^{c_1} \quad = \quad & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q_{a_1 P_{a_1} S_1}^{c_1}\} \cup \\
& \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c_1',c_2',e_1',e_2' > \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c_1',c_2' \in \{Z,B\}, e_1',e_2' \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c_1',c_2',e_1',e_2' > \to x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c_1',c_2',e_1',e_2' > \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c_1',c_2' \in \{Z,B\}, e_1',e_2' \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}
$$

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

The following two grammars have new communication steps $S \to Q^{C_1}_{a_1 P_{a_1} S_1 H_2(S_4)}$ and $S \to Q^{C_1}_{a_1 P_{a_1} S_1 H_3(S_4)}$, respectively. In a successful derivation these components will rewrite to this communication request in Step 13 of the derivation. If this rewriting rule is used in any other step the derivation will block; more precisely if this rule is nondeterministically chosen in Step 1 it results in a circular query and the derivation will block immediately. If it is used in Step 3 it will receive the string $<I>$ which will rewrite to $[x,q,Z,Z,e_1,e_2]$ or $x[y,q,Z,Z,e_1,e_2]$. We however have no rewriting rule for either of these strings and so we will block. Finally, if these rules are used in Step 9 the components will receive the string $u[x',q,Z,Z,e_1,e_2]$, for which no rewriting rules exist so once more the system will block.

$$\begin{aligned}
P^{c_1}_{GM_{S1H2(S4)}} = {} & \{S \to [I], [I] \to C\} \cup \\
& \mathfrak{N} = \{C \to Q^{c_1}_{a_1 P_{a_1} S_1 H_2(S_4)}, S \to Q^{c_1}_{a_1 P_{a_1} S_1 H_2(S_4)}\} \cup \\
& \{<I> \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{<I> \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{<x,q,c'_1,c'_2,e'_1,e'_2> \to [x,q',c_1,c_2,e_1,e_2] | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{<x,q,c'_1,c'_2,e'_1,e'_2> \to x[y,q',c_1,c_2,e_1,e_2], <x,q_F,c'_1,c'_2,e'_1,e'_2> \to x | \\
& (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}$$

$$\begin{aligned}
P^{c_1}_{GM_{S1H3(S4)}} = {} & \{S \to [I], [I] \to C\} \cup \\
& \mathfrak{N} = \{C \to Q^{c_1}_{a_1 P_{a_1} S_1 H_3(S_4)}, S \to Q^{c_1}_{a_1 P_{a_1} S_1 H_3(S_4)}\} \cup \\
& \{<I> \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{<I> \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{<x,q,c'_1,c'_2,e'_1,e'_2> \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{<x,q,c'_1,c'_2,e'_1,e'_2> \to x[y,q',c_1,c_2,e_1,e_2], <x,q_F,c'_1,c'_2,e'_1,e'_2> \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}$$

$$\begin{aligned}
P^{c_1}_{GM_{S1(S2)}} = {} & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q^{c_1}_{a_1 P_{a_1} S_1(S_2)}\} \cup \\
& \{<I> \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{<I> \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{<x,q,c'_1,c'_2,e'_1,e'_2> \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{<x,q,c'_1,c'_2,e'_1,e'_2> \to x[y,q',c_1,c_2,e_1,e_2], <x,q_F,c'_1,c'_2,e'_1,e'_2> \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}$$

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\begin{aligned}
P^{c_1}_{GM_{S1(S3)}} \quad = \quad & \{S \rightarrow [I], [I] \rightarrow C\} \cup \mathfrak{N} = \{C \rightarrow Q^{c_1}_{a_1 P_{a_1} S_1(S_3)}\} \cup \\
& \{< I > \rightarrow [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \rightarrow x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \rightarrow [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \rightarrow x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \rightarrow x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}
$$

We only need one $P_2^{c_1}$ component. The grammar below will simulate rules from the master grammar and will work indirectly with $P^{c_1}_{2_{OriginalS2}}$ holding intermediate strings. The labels in the communication rules have been modified to ensure that the correct component grammars are queried during a derivation.

$$
\begin{aligned}
P^{c_1}_{GM_{S2}} \quad = \quad & \{S \rightarrow [I], [I] \rightarrow C\} \cup \mathfrak{N} = \{C \rightarrow Q^{c_1}_{a_1 P_{a_1} S_2}\} \cup \\
& \{< I > \rightarrow [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \rightarrow x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \rightarrow [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \rightarrow x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \rightarrow x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}
$$

Similar to the $P_2^{c_1}$ we only need one $P_3^{c_1}$. The helper below contains modified rules from the new master grammar. This grammar will work indirectly with $P^{c_1}_{3_{OriginalS3}}$, holding intermediate strings. The labels in the communication steps reflect the labeling of component grammar it will work with during a derivation.

$$
\begin{aligned}
P^{c_1}_{GM_{S3}} \quad = \quad & \{S \rightarrow [I], [I] \rightarrow C\} \cup \mathfrak{N} = \{C \rightarrow Q^{c_1}_{a_1 P_{a_1} S_3}\} \cup \\
& \{< I > \rightarrow [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \rightarrow x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \rightarrow [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \rightarrow x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \rightarrow x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}
$$

The following 7 helper grammars imitate P_{a_1}. The first 5 work with $P^{c_1}_{1_{original}}$ and four of its helpers, while the remaining 2 work with $P^{c_1}_{2_{original}}$ and $P^{c_1}_{3_{original}}$ holding intermediate strings during derivations. A new rule has been added to these components; this rule allows the grammars to reset themselves by querying

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

their new helper component defined later in the “reset” section.

$$
\begin{aligned}
P^{c_1}_{GM_{PA1S1}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \\
& \mathfrak{N} = \{C \to Q_{Reset_{GM_{Pa1^{c_1}_{S1}}}}\} \cup \\
& \{< I > \to [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \to x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_1}_{GM_{PA1S1H2}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \\
& \mathfrak{N} = \{C \to Q_{Reset_{GM^{c_1}_{Pa1_{S1H2(S4)}}}}\} \cup \\
& \{< I > \to [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \to x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_1}_{GM_{PA1S1H3}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \\
& \mathfrak{N} = \{C \to Q_{Reset_{GM^{c_1}_{Pa1_{S1H3(S4)}}}}\} \cup \\
& \{< I > \to [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \to x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_1}_{GM_{PA1S1(S2)}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \\
& \mathfrak{N} = \{C \to Q_{Reset_{GM^{c_1}_{Pa1_{S1(S2)}}}}\} \cup \\
& \{< I > \to [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \to x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}
$$

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$\begin{aligned}
P^{c_1}_{GM_{PA1S1(S3)}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q_{Reset_{GM^{c_1}_{Pa1_{S1(S3)}}}}\} \cup \\
& \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, \quad e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}$$

$$\begin{aligned}
P^{c_1}_{GM_{PA1S2}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q_{Reset_{GM^{c_1}_{Pa1_{S2}}}}\} \cup \\
& \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}$$

$$\begin{aligned}
P^{c_1}_{GM_{PA1S3}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q_{Reset_{GM^{c_1}_{Pa1_{S3}}}}\} \cup \\
& \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}$$

The following 5 helper grammars simulate rules from the new master. Each component defined below is designed to work with a different component in the $P^{c_2}_1$ family, including the $P^{c_2}_{1_{OriginalS_1}}$ and its 4 helpers. The first one works indirectly with $P^{c_2}_{1_{original}}$ as it does in the original construction but communication step labels have been modified to ensure that each component queries the right grammar.

$$\begin{aligned}
P^{c_2}_{GM_{S1}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q^{c_2}_{a_1 P_{a_1} S_1}\} \cup \\
& \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}$$

Note that the following two grammars have a new communication step $S \to Q^{c_2}_{a_1 P_{a_1} S_1 H_2(S_4)}$ and $S \to Q^{c_2}_{a_1 P_{a_1} S_1 H_3(S_4)}$ respectively. In a successful derivation this communication step will be used in Step 13 of the derivation. If this rule is introduced in any other step the system will block. More specifically if this

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

rule is used in Step 1 it results in a circular query and blocks; if it is used in Step 3 it will receive the string $< I >$ which will rewrite to $[x,q,Z,Z,e_1,e_2]$ or $x[y,q,Z,Z,e_1,e_2]$ for which no rewriting rule exists; finally if it is used in Step 9 the $P^{c_2}_{GM_{S1H2(S4)}}$ or $P^{c_2}_{GM_{S1H3(S4)}}$ component will receive the string $u[x',q,Z,Z,e_1,e_2]$, for which it has no rewriting rule.

$$
\begin{aligned}
P^{c_2}_{GM_{S1H2(S4)}} = \; & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q^{c_2}_{a_1 P_{a_1} S_1 H_2(S_4)}, S \to Q^{c_2}_{a_1 P_{a_1} S_1 H_2(S_4)}\} \cup \\
& \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_2}_{GM_{S1H3(S4)}} = \; & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q^{c_2}_{a_1 P_{a_1} S_1 H_3(S_4)}, S \to Q^{c_2}_{a_1 P_{a_1} S_1 H_3(S_4)}\} \cup \\
& \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_2}_{GM_{S1(S2)}} = \; & \{S \to [I], [I] \to C\} \cup \\
& \mathfrak{N} = \{C \to Q^{c_2}_{a_1 P_{a_1} S_1(S_2)}\} \cup \\
& \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_2}_{GM_{S1(S3)}} = \; & \{S \to [I], [I] \to C\} \cup \\
& \mathfrak{N} = \{C \to Q^{c_2}_{a_1 P_{a_1} S_1(S_2)}\} \cup \\
& \{< I > \to [x,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,e_1,e_2,0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y,q,Z,Z,e_1,e_2] | (x,q_0,Z,Z,q,e_1,e_2,+1) \in R, x,y \in \Sigma\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to [x,q',c_1,c_2,e_1,e_2] | (x,q,c_1,c_2,q',e_1,e_2,0) \in R, \\
& \quad x \in \Sigma, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}\} \cup \\
& \{< x,q,c'_1,c'_2,e'_1,e'_2 > \to x[y,q',c_1,c_2,e_1,e_2], < x,q_F,c'_1,c'_2,e'_1,e'_2 > \to x | \\
& \quad (x,q,c_1,c_2,q',e_1,e_2,+1) \in R, c'_1,c'_2 \in \{Z,B\}, e'_1,e'_2 \in \{-1,0,+1\}, x,y \in \Sigma\}
\end{aligned}
$$

There is only one $P_2^{c_2}$ as in the original system and the below master helper works indirectly with it. The query labels are modified to ensure that the correct component grammars are queried during the

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

derivation.

$$\begin{aligned}
P^{c_2}_{GM_{S2}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q^{c_2}_{a_1 P_{a_1} S_2}\} \cup \\
& \{< I > \to [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \to x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}$$

Similarly, there is only one $P^{c_2}_3$, as in the original system and the below master helper will work indirectly with it. The labels of the query symbols have been modified in order to ensure that the correct component grammars are queried during the derivation.

$$\begin{aligned}
P^{c_2}_{GM_{S3}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q^{c_2}_{a_1 P_{a_1} S_3}\} \cup \\
& \{< I > \to [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \to x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, \quad e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}$$

The following 7 grammars work with the $P^{c_2}_{a_1}$ components; the first 5 work with the $P^{c_2}_1$ helper grammars, and the other 2 work with $P^{c_2}_{2_{OriginalS_2}}$ and $P^{c_2}_{3_{OriginalS_3}}$ holding intermediate strings to ensure successful derivations. A new rule has been added to these grammar components which allows them to reset themselves by querying their matching reset component (defined later).

$$\begin{aligned}
P^{c_2}_{GM_{PA1S1}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q_{Reset_{GM^{c_2}_{Pa1_{S1}}}}\} \cup \\
& \{< I > \to [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \to x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, \quad e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}$$

$$\begin{aligned}
P^{c_2}_{GM_{PA1S1H2}} \quad = \quad & \{S \to [I], [I] \to C\} \cup \mathfrak{N} = \{C \to Q_{Reset_{GM^{c_2}_{Pa1_{S1H2(S4)}}}}\} \cup \\
& \{< I > \to [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \to x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \to x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, \quad e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}$$

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\begin{aligned}
P^{c_2}_{GM_{PA1S1H3}} \;=\; & \{S \rightarrow [I], [I] \rightarrow C\} \cup \mathfrak{N} = \{C \rightarrow Q_{Reset_{GM^{c_2}_{Pa1_{S1H3(S4)}}}}\} \cup \\
& \{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \rightarrow x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \rightarrow x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_2}_{GM_{PA1S1S2}} \;=\; & \{S \rightarrow [I], [I] \rightarrow C\} \cup \mathfrak{N} = \{C \rightarrow Q_{Reset_{GM^{c_2}_{Pa1_{S1(S2)}}}}\} \cup \\
& \{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \rightarrow x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \rightarrow x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_2}_{GM_{PA1S1S3}} \;=\; & \{S \rightarrow [I], [I] \rightarrow C\} \cup \mathfrak{N} = \{C \rightarrow Q_{Reset_{GM^{c_2}_{Pa1_{S1(S3)}}}}\} \cup \\
& \{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \rightarrow x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \rightarrow x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_2}_{GM_{PA1S2}} \;=\; & \{S \rightarrow [I], [I] \rightarrow C\} \cup \mathfrak{N} = \{C \rightarrow Q_{Reset_{GM^{c_2}_{Pa1_{S2}}}}\} \cup \\
& \{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \rightarrow x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \rightarrow x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, \quad e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_2}_{GM_{PA1S3}} \;=\; & \{S \rightarrow [I], [I] \rightarrow C\} \cup \mathfrak{N} = \{C \rightarrow Q_{Reset_{GM^{c_2}_{Pa1_{S3}}}}\} \cup \\
& \{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
& \quad x \in \Sigma, c'_1, c'_2 \in \{Z, B\}, e'_1, e'_2 \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c'_1, c'_2, e'_1, e'_2 > \rightarrow x[y, q', c_1, c_2, e_1, e_2], < x, q_F, c'_1, c'_2, e'_1, e'_2 > \rightarrow x | \\
& \quad (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, c'_1, c'_2 \in \{Z, B\}, \quad e'_1, e'_2 \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}
$$

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$P^{c_1}_{1_{originalS_1}}$ contains the same rewriting rules and communication steps as the component $P^{c_1}_1$ in the original system [5]. Some labels in the rewriting rules have been modified to ensure that the components query their corresponding helper grammars in the other sections of the system. Note that the $P^{c_1}_1$ component has 4 new helper grammars in this construction; these helper grammars are required to ensure that $P^{c_1}_2$, $P^{c_1}_3$, and $P^{c_1}_4$ have their own unique component grammars to communicate with.

$$\begin{aligned} P^{c_1}_{1_{originalS_1}} &= \mathfrak{R} = \{S_1 \to Q^{c_1}_{GM_{S_1}}, S_1 \to Q^{c_1}_{4_{S_{1original}}}, C \to Q^{c_1}_{GM_{S_1}}\} \cup \\ &\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_1]', [+1]' \to AAC, [0]' \to AC, [-1]' \to C | \\ &\quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \cup \{[I] \to [I]', [I]' \to AC\} \end{aligned}$$

The following two $P^{c_1}_1$ helper grammars work with their respective helper grammars as defined in their rewriting rules; their definition contains a rule $C \to W$, which will be used in Step 13 during successful derivations. If this rule is used at any other step the system will block (just like in the similar situations discussed earlier).

$$\begin{aligned} P^{c_1}_{1_{S_1H_2(S_4)}} &= \mathfrak{R} = \{S_1 \to Q^{c_1}_{GM_{S_1H_2(S_4)}}, S_1 \to Q^{c_1}_{4S_1H_2(S_4)}, C \to Q_{GM_{S_1H_2(S_4)}}, C \to W\} \cup \\ &\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_1]', [+1]' \to AAC, [0]' \to AC, [-1]' \to C | \\ &\quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \cup \{[I] \to [I]', [I]' \to AC\} \end{aligned}$$

$$\begin{aligned} P^{c_1}_{1_{S_1H_3(S_4)}} &= \mathfrak{R} = \{S_1 \to Q^{c_1}_{GM_{S_1H_3(S_4)}}, S_1 \to Q^{c_1}_{4S_1H_3(S_4)}, C \to Q_{GM_{S_1H_3(S_4)}}, C \to W\} \cup \\ &\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_1]', [+1]' \to AAC, [0]' \to AC, [-1]' \to C | \\ &\quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \cup \{[I] \to [I]', [I]' \to AC\} \end{aligned}$$

The following two $P^{c_1}_1$ helper grammars will ensure the proper derivation of $P^{c_1}_{2_{OriginalS_2}}$ and $P^{c_1}_{3_{OriginalS_3}}$. They work by communicating with their corresponding helper grammars and their designated special helper in the $P^{c_1}_4$ section.

$$\begin{aligned} P^{c_1}_{1_{S_1}(S_2)} &= \mathfrak{R} = \{S_1 \to Q^{c_1}_{GM_{S_1(S_2)}}, S_1 \to Q^{c_1}_{4SpecialHelper1_{S1S2}}, C \to Q_{GM_{S_1(S_2)}}, \\ &\quad S_4 \to S_4^{(1)}, S_4^{(1)} \to Q_{P^{c_1}_{1_{S_1H_2(S_4)}}}\} \cup \\ &\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_1]', [+1]' \to AAC, [0]' \to AC, [-1]' \to C | \\ &\quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \cup \{[I] \to [I]', [I]' \to AC\} \end{aligned}$$

$$\begin{aligned} P^{c_1}_{1_{S_1}(S_3)} &= \mathfrak{R} = \{S_1 \to Q^{c_1}_{GM_{S_1(S_3)}}, S_1 \to Q^{c_1}_{4SpecialHelper1_{S1S3}}, C \to Q^{c_1}_{GM_{S_1(S_3)}}, \\ &\quad S_4 \to S_4^{(1)}, S_4^{(1)} \to Q_{P^{c_1}_{1_{S_1H_3(S_4)}}}\} \cup \\ &\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_1]', [+1]' \to AAC, [0]' \to AC, [-1]' \to C | \\ &\quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \cup \{[I] \to [I]', [I]' \to AC\} \end{aligned}$$

Component grammar $P^{c_1}_2$ remains similar to the original system without any additional helper grammars. It has been renamed and labels have been modified to ensure that it works with its matching helper components.

$$\begin{aligned} P^{c_1}_{2_{OriginalS_2}} &= \mathfrak{R} = \{S_2 \to Q^{c_1}_{GMS_2}, S_2 \to Q^{c_1}_{4S_2}, C \to Q^{c_1}_{GMS_2}, A \to A\} \cup \\ &\quad \{[x,q,Z,c_2,e_1,e_2] \to [x,q,Z,c_2,e_1,e_2], [I] \to [I] | x \in \Sigma, q \in E, \\ &\quad c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \end{aligned}$$

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

Component grammar $P_3^{c_1}$ is again similar to the original definition and it does not need any helper grammars in this construction. Its name has been modified to identify that it was part of the original construction and the labeling in the communication steps has been modified to ensure the correct helper components are queried.

$$\begin{aligned} P^{c_1}_{3_{OriginalS3}} &= \mathfrak{N} = \{S_3 \to Q^{c_1}_{GMS_3}, S_3 \to Q^{c_1}_{4S_3}, C \to Q^{c_1}_{GMS_3}\} \cup \\ &\quad \{[x,q,Z,c_2,e_1,e_2] \to a, [x,q,B,c_2,e_1,e_2] \to [x,q,B,c_2,e_1,e_2] \\ &\quad [I] \to [I] | x \in \Sigma, q \in E, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \end{aligned}$$

Component $P^{c_1}_{4_{OriginalS4}}$, needs extra helper grammars to ensure that components defined in other sections have their own unique $P_4^{c_1}$ component to query. The rules in the original grammar are for the most part unchanged, the only difference is the labeling.

$$P^{c_1}_{4_{OriginalS_4}} = \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \mathfrak{N} = \{S_4^{(2)} \to Q^{c_1}_{P_1S_1}\} \cup \{A \to a\}$$

A new nondeterministic step has been added to the following two helpers in the P_4 section, specifically: $S_4^{(2)} \to S_4^{(2)}$. This rule was added to avoid a circular query in Step 12 of the derivation. This being said this rule could be used whenever the non terminal $S_4^{(2)}$ appears, but if it is used in any other step there is a chance that the matching P_1 component queries it and receives $S_4^{(2)}$, but since P_1 does not contain a rewriting rule for $S_4^{(2)}$ the derivation would block. The only successful use of this rewriting rule is in Step 12.

$$\begin{aligned} P^{c_1}_{4_{S_1H_2(S_4)}} &= \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \mathfrak{N} = \{S_4^{(2)} \to Q^{c_1}_{P_1S_1H_2(S_4)}, S_4^{(2)} \to S_4^{(2)}\} \cup \{A \to a\} \\ P^{c_1}_{4_{S_1H_3(S_4)}} &= \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \mathfrak{N} = \{S_4^{(2)} \to Q^{c_1}_{P_1S_1H_3(S_4)}, S_4^{(2)} \to S_4^{(2)}\} \cup \{A \to a\} \\ P^{c_1}_{4_{S_2}} &= \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \\ &\quad \mathfrak{N} = \{S_4^{(2)} \to Q^{c_1}_{P_1S_2}\} \cup \{A \to a\} \\ P^{c_1}_{4_{S_3}} &= \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \\ &\quad \mathfrak{N} = \{S_4^{(2)} \to Q^{c_1}_{P_1S_3}\} \cup \{A \to a\} \end{aligned}$$

$$\begin{aligned} P^{c_1}_{4SpecialHelper1_{S1S2}} &= \mathfrak{N} = \{S_4 \to S_4\} \\ P^{c_1}_{4SpecialHelper2_{S1S3}} &= \mathfrak{N} = \{S_4 \to S_4\} \end{aligned}$$

$P^{c_2}_{1_{OriginalS_1}}$ contains similar rules as in $P_1^{c_2}$ except it has new labels. It also need 4 new helper grammars.

$$\begin{aligned} P^{c_2}_{1_{OriginalS_1}} &= \mathfrak{N} = \{S_1 \to Q^{c_2}_{GMS_1}, S_1 \to Q^{c_2}_{P_4S_1}, C \to Q^{c_2}_{GMS_1}\} \cup \\ &\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C | \\ &\quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \cup \{[I] \to [I]', [I]' \to AC\} \end{aligned}$$

The following two $P_1^{c_2}$ have a new rule added to them that will be used in Step 13 of the derivation: $C \to W$. If this rule is used at any other step the system will block for the same reason as above.

$$\begin{aligned} P^{c_2}_{1_{S_1H_2(S_4)}} &= \mathfrak{N} = \{S_1 \to Q^{c_2}_{GM_{S_1H_2(S_4)}}, S_1 \to Q^{c_2}_{P_4S_1H_2(S_4)}, C \to Q^{c_2}_{GM_{S_1H_2(S_4)}}, C \to W\} \cup \\ &\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C | \\ &\quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \cup \{[I] \to [I]', [I]' \to AC\} \end{aligned}$$

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$\begin{aligned}P^{c_2}_{1_{S_1H_3(S_4)}} &= \mathfrak{N} = \{S_1 \to Q^{c_2}_{GM_{S_1H_3(S_4)}}, S_1 \to Q^{c_2}_{P_4S_1H_3(S_4)}, C \to Q^{c_2}_{GM_{S_1H_3(S_4)}}, C \to W\} \cup \\ &\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C | \\ &\quad\quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \cup \{[I] \to [I]', [I]' \to AC\}\end{aligned}$$

The following two $P^{c_2}_1$ helper grammars are components that will help ensure the proper derivation of $P^{c_2}_{2_{OriginalS_2}}$ and $P^{c_2}_{3_{OriginalS_3}}$ by holding intermediate strings throughout the derivation.

$$\begin{aligned}P^{c_2}_{1_{S_1(S_2)}} &= \mathfrak{N} = \{S_1 \to Q^{c_2}_{GM_{S_1(S_2)}}, S_1 \to Q^{c_2}_{4SpecialHelper1_{S1S2}}, C \to Q^{c_2}_{GM_{S_1(S_2)}}, \\ &\quad S_4 \to S_4^{(1)}, S_4^{(1)} \to Q_{P^{c_2}_{1_{S_1H_2(S_4)}}}\} \cup \\ &\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C | \\ &\quad\quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \cup \{[I] \to [I]', [I]' \to AC\}\end{aligned}$$

$$\begin{aligned}P^{c_2}_{1_{S_1(S_3)}} &= \mathfrak{N} = \{S_1 \to Q^{c_2}_{GM_{S_1(S_3)}}, S_1 \to Q^{c_2}_{4SpecialHelper1_{S1S3}}, C \to Q^{c_2}_{GM_{S_1(S_3)}}, \\ &\quad S_4 \to S_4^{(1)}, S_4^{(1)} \to Q_{P^{c_2}_{1_{S_1H_3(S_4)}}}\} \cup \\ &\quad \{[x,q,c_1,c_2,e_1,e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C | \\ &\quad\quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\} \cup \{[I] \to [I]', [I]' \to AC\}\end{aligned}$$

Component grammar $P^{c_2}_2$ is the same as in the original system, except that it has been renamed and the communication rewriting rules have been modified to match the correct helper components.

$$\begin{aligned}P^{c_2}_{2_{OriginalS_2}} &= \mathfrak{N} = \{S_2 \to Q^{c_2}_{GMS_2}, S_2 \to Q^{c_2}_{P_4S_2}, C \to Q^{c_2}_{GMS_2}\} \cup \{A \to A\} \cup \\ &\quad \{[x,q,c_1,Z,e_1,e_2] \to a, [x,q,c_1,B,e_1,e_2] \to [x,q,c_1,B,e_1,e_2], [I] \to [I] | \\ &\quad\quad x \in \Sigma, q \in E, c1 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\}\end{aligned}$$

Component grammar $P^{c_2}_3$ contains similar rules with the original construction. Similarly to $P^{c_2}_{2_{OriginalS_2}}$ it does not require any helper grammars. Its name has been modified to reflect that it was part of the original construction and its communication rules have been modified to reflect the labeling of the proper helper components.

$$\begin{aligned}P^{c_2}_{3_{OriginalS3}} &= \mathfrak{N} = \{S_3 \to Q^{c_2}_{GMS_3}, S_3 \to Q^{c_2}_{P_4S_2}, C \to Q^{c_2}_{GMS_3}\} \cup \\ &\quad \{[x,q,c_1,Z,e_1,e_2] \to a, [x,q,c_1,B,e_1,e_2] \to [x,q,c_1,B,e_1,e_2] \\ &\quad [I] \to [I] | x \in \Sigma, q \in E, c1 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\}\end{aligned}$$

Component $P^{c_2}_{4_{OriginalS4}}$, requires 6 additional components to ensure a successful derivation. The name of the grammar has been modified and the rules in the grammar have had their labeling updated to match the respective helper grammars.

$$P^{c_2}_{4_{OriginalS4}} = \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \mathfrak{N} = \{S_4^{(2)} \to Q^{c_2}_{P_1S_1}\} \cup \{A \to a\}$$

A new nondeterministic step has been added to the following two helpers for the original P_4 component. The rule $S_4^{(2)} \to S_4^{(2)}$ was added specifically to avoid a circular query in Step 12 of the derivation, but this rule could be used whenever the non terminal $S_4^{(2)}$ appears. If it is used in any other step there is a chance that the matching P_1 component requests its string and receives $S_4^{(2)}$. Thankfully the matching

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

P_1 component does not have a corresponding rewriting rule and thus the derivation will block. In a successful derivation this rule will thus be used only in Step 12.

$$\begin{aligned}
P^{c_2}_{4_{S_1H_2(S_4)}} &= \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \mathfrak{N} = \{S_4^{(2)} \to Q^{c_2}_{P_1S_1H_2(S_4)}, S_4^{(2)} \to S_4^{(2)}\} \cup \{A \to a\} \\
P^{c_2}_{4_{S_1H_3(S_4)}} &= \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \mathfrak{N} = \{S_4^{(2)} \to Q^{c_2}_{P_1S_1H_3(S_4)}, S_4^{(2)} \to S_4^{(2)}\} \cup \{A \to a\} \\
P^{c_2}_{4_{S_2}} &= \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \mathfrak{N} = \{S_4^{(2)} \to Q^{c_2}_{P_1S_2}\} \cup \{A \to a\} \\
P^{c_2}_{4_{S_3}} &= \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \mathfrak{N} = \{S_4^{(2)} \to Q^{c_2}_{P_1S_3}\} \cup \{A \to a\}
\end{aligned}$$

$$\begin{aligned}
P^{c_2}_{4SpecialHelper1_{S1S2}} &= \mathfrak{N} = \{S_4 \to S_4\} \\
P^{c_2}_{4SpecialHelper2_{S1S3}} &= \mathfrak{N} = \{S_4 \to S_4\}
\end{aligned}$$

The original P_{a_1} grammar remains as it was in the original system. In order for component grammars in sections $P^{c_1}_1$, $P^{c_1}_2$,$P^{c_1}_3$,$P^{c_1}_4$, $P^{c_2}_1$, $P^{c_2}_2$,$P^{c_2}_3$, and $P^{c_2}_4$ to derive correctly 14 additional P_{a_1} helpers have been added to the system. Their names and labels reflect the components they will work with during a derivation.

$$\begin{aligned}
P_{a_1Original} = \;& \mathfrak{N} = \{S \to Q_{GM_{original}}\} \cup \\
& \{[I] \to < I >, \quad [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_1}_{a_1GMS_1} = \;& \mathfrak{N} = \{S \to Q^{c_1}_{GMPA1S_1}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_1}_{a_1GMS_1H_2(S_4)} = \;& \mathfrak{N} = \{S \to Q^{c_1}_{GMPA1S_1H_2(S_4)}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_1}_{a_1GMS_1H_3(S_4)} = \;& \mathfrak{N} = \{S \to Q^{c_1}_{GMPA1S_1H_3(S_4)}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_1}_{a_1GMS_1(S_2)} = \;& \mathfrak{N} = \{S \to Q^{c_1}_{GMS_1(S_2)}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$\begin{aligned}
P^{c_1}_{a_1GMS_1(S_3)} \;=\;& \mathfrak{N} = \{S \to Q^{c_1}_{GMS_1(S_3)}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_1}_{a_1GMS_2} \;=\;& \mathfrak{N} = \{S \to Q^{c_1}_{GMPA1S_2}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_1}_{a_1GMS_3} \;=\;& \mathfrak{N} = \{S \to Q^{c_1}_{GMPA1S_3}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_2}_{a_1GMS_1} \;=\;& \mathfrak{N} = \{S \to Q^{c_2}_{GMPA1S_1}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_2}_{a_1GMS_1H_2(S_4)} \;=\;& \mathfrak{N} = \{S \to Q^{c_2}_{GMPA1S_1H_2(S_4)}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_2}_{a_1GMS_1H_3(S_4)} \;=\;& \mathfrak{N} = \{S \to Q^{c_2}_{GMPA1S_1H_3(S_4)}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_2}_{a_1GMS_1(S_2)} \;=\;& \mathfrak{N} = \{S \to Q^{c_2}_{GMS_1(S_2)}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

$$\begin{aligned}
P^{c_2}_{a_1GMS_1(S_3)} \;=\;& \mathfrak{N} = \{S \to Q^{c_2}_{GMS_1(S_3)}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1,c_2 \in \{Z,B\}, e_1,e_2 \in \{-1,0,+1\}\}
\end{aligned}$$

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\begin{aligned}
P^{c_2}_{a_1 GMS_2} \quad = \quad & \mathfrak{N} = \{S \to Q^{c_2}_{GMPA1S_2}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_2}_{a_1 GMS_3} \quad = \quad & \mathfrak{N} = \{S \to Q^{c_2}_{GMPA1S_3}, C \to C\} \cup \\
& \{[I] \to < I >, [x,q,c_1,c_2,e_1,e_2] \to < x,q,c_1,c_2,e_1,e_2 >, \\
& \quad < x,q,c_1,c_2,e_1,e_2 > \to < x,q,c_1,c_2,e_1,e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z,B\}, e_1, e_2 \in \{-1,0,+1\}\}
\end{aligned}
$$

The original component grammar P_{a_2} remains unchanged and works as it did in the original system, but we will refer to it as $P_{a_{2Original}}$ in order to remain consistent with the naming of the other original components in the system. The communication rule has also been modified to reflect the new names of the component grammars.

$$
\begin{aligned}
P_{a_{2Original}} \quad = \quad & \{S \to S^3, S^{(1)} \to S^{(2)}, S^{(2)} \to S^{(3)}, S^{(3)} \to S^{(4)}\} \cup \\
& \mathfrak{N} = \{S^{(4)} \to Q^{c_1}_{P_{2_{originalS_2}}} Q^{c_1}_{P_{3_{originalS_3}}} Q^{c_2}_{P_{2_{OriginalS_2}}} Q^{c_2}_{P_{3_{originalS_3}}} S^{(1)}\}.
\end{aligned}
$$

Now we define the grammars that are used to reset the P_{a1} helpers. They will send the non-terminal $< I >$ to their matching component grammar, which will allow their derivation to restart. These components and their rewriting rules are not part of the original system.

$$
\begin{aligned}
Reset_{GM^{c_1}_{Pa1_{S1}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_1}_{Pa1_{S1H2(S4)}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_1}_{Pa1_{S1H3(S4)}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_1}_{Pa1_{S1(S2)}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_1}_{Pa1_{S1(S3)}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_1}_{Pa1_{S2}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_1}_{Pa1_{S3}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_2}_{Pa1_{S1}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_2}_{Pa1_{S1H2(S4)}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_2}_{Pa1_{S1H3(S4)}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_2}_{Pa1_{S1(S2)}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_2}_{Pa1_{S1(S3)}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_2}_{Pa1_{S2}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\} \\
Reset_{GM^{c_2}_{Pa1_{S3}}} &= \mathfrak{N} = \{S \to < I >, < I > \to < I >\}
\end{aligned}
$$

The components below will be used to reset $P^{c_1}_{1_{S_1 H_2(S_4)}}$, $P^{c_1}_{1_{S_1 H_3(S_4)}}$, $P^{c_2}_{1_{S_1 H_2(S_4)}}$, and $P^{c_2}_{1_{S_1 H_3(S_4)}}$ in Step 13 of the derivation. This reset allows queried components to be reset to their axioms which in turn allows the

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

derivation to restart. These components were not part of the original system definition.

$$
\begin{aligned}
Reset_{P^{c_1}_{1_{S_1H_2(S_4)}}} &= \mathfrak{N} = \{\, U \to U_1, U_1 \to U_2, U_2 \to U_3, \\
&\quad U_3 \to U_4, U_4 \to U_5, U_6 \to U_7, U_7 \to Q_{P^{c_1}_{1_{S_1H_2(S_4)}}} U_4\} \\
Reset_{P^{c_1}_{1_{S_1H_3(S_4)}}} &= \mathfrak{N} = \{U \to U_1, U_1 \to U_2, U_2 \to U_3, \\
&\quad U_3 \to U_4, U_4 \to U_5, U_6 \to U_7, U_7 \to Q_{P^{c_1}_{1_{S_1H_3(S_4)}}} U_4\} \\
Reset_{P^{c_2}_{1_{S_1H_2(S_4)}}} &= \mathfrak{N} = \{U \to U_1, U_1 \to U_2, U_2 \to U_3, \\
&\quad U_3 \to U_4, U_4 \to U_5, U_6 \to U_7, U_7 \to Q_{P^{c_1}_{1_{S_1H_2(S_4)}}} U_4\} \\
Reset_{P^{c_2}_{1_{S_1H_3(S_4)}}} &= \mathfrak{N} = \{U \to U_1, U_1 \to U_2, U_2 \to U_3, \\
&\quad U_3 \to U_4, U_4 \to U_5, U_6 \to U_7, U_7 \to Q_{P^{c_1}_{1_{S_1H_3(S_4)}}} U_4\}
\end{aligned}
$$

The following, new grammar components will be used to reset $P^{c_1}_{4_{S_1H_2(S_4)}}$, $P^{c_1}_{4_{S_1H_3(S_4)}}$, $P^{c_2}_{4_{S_1H_2(S_4)}}$, and $P^{c_2}_{4_{S_1H_3(S_4)}}$ in Step 14 of a successful derivation. The reset components allows the system to restart the derivation process.

$$
\begin{aligned}
Reset_{P^{c_1}_{4_{S_1H_2(S_4)}}} &= \mathfrak{N} = \{T \to T_1, T_1 \to T_2, T_2 \to T_3, \\
&\quad T_3 \to T_4, T_4 \to T_5, T_6 \to T_7, T_7 \to Q_{P^{c_1}_{4_{S_1H_2(S_4)}}} T_4\} \\
Reset_{P^{c_1}_{4_{S_1H_3(S_4)}}} &= \mathfrak{N} = \{T \to T_1, T_1 \to T_2, T_2 \to T_3, \\
&\quad T_3 \to T_4, T_4 \to T_5, T_6 \to T_7, T_7 \to Q_{P^{c_1}_{4_{S_1H_3(S_4)}}} T_4\} \\
Reset_{P^{c_2}_{4_{S_1H_2(S_4)}}} &= \mathfrak{N} = \{T \to T_1, T_1 \to T_2, T_2 \to T_3, \\
&\quad T_3 \to T_4, T_4 \to T_5, T_6 \to T_7, T_7 \to Q_{P^{c_2}_{4_{S_1H_2(S_4)}}} T_4\} \\
Reset_{P^{c_2}_{4_{S_1H_3(S_4)}}} &= \mathfrak{N} = \{T \to T_1, T_1 \to T_2, T_2 \to T_3, \\
&\quad T_3 \to T_4, T_4 \to T_5, T_6 \to T_7, T_7 \to Q_{P^{c_2}_{4_{S_1H_3(S_4)}}} T_4\}
\end{aligned}
$$

4.2 The Simulation of the 2-Counter Turing Machine

As in any PCGS the master grammar controls the derivation. The string $[x,q,c_1,c_2,e_1,e_2]$ present in the master component, where $x \in \Sigma$, $q \in E$, $c_1, c_2 \in \{Z,B\}$, $e_1, e_2 \in \{-1,0,+1\}$ means that the 2-counter machine M is in state q, the input head proceeds to scan x onto the input tape and c_1, c_2 on the two storage (counter) tapes, respectively, and then the heads of the storage tapes are moved according to values in e_1, and e_2. The number of A symbols in the strings of the c_1, c_2 component grammars keep track of the value of the counters of M, meaning that these numbers should always match the value stored in the counters of M or else the system will block.

We used the "original" grammar system components $P_i^{c_1}$, $P_i^{c_2}$, $1 \geq i \geq 4$ to simulate the changes in the counters, as done in the original system [5]. All of the other component grammars included in our construction enable the original components to work correctly using one-step communication throughout the derivation.

The PCGS Γ first introduces $[I]$ in the master grammar, then a number of rewriting steps occur in a sequence that initializes Γ by setting the counters to 0. Once these steps are completed Γ can then simulate the first transition of M by rewriting $[I]$ to $u[x', q, Z, Z, e_1, e_2]$ where $(x, q_0, Z, Z, q, e_1, e_2, g)$ is a rule of M. Here $u = x$ if $g = +1$ and $u = \varepsilon$, $x' = x$ if $g = 0$. In the case that the input head moves ($g = +1$), the master grammar generates x followed by $[x', q, Z, Z, e_1, e_2]$ which shows that M is now scanning a new symbol. If the input head does not move, the master grammar does not generate any terminals and the string $[x', q, Z, Z, e_1, e_2]$ indicates that M is still scanning the same symbol. At this point $P_2^{c_1}$, $P_3^{c_1}$, $P_2^{c_2}$, and $P_3^{c_2}$ verify the values stored in the counters of M, and modify the values according to e_1 and e_2. Γ can then determine if it can enter state q by verifying and updating the counters before moving forward. In order to simulate the next step the master grammar rewrites $[x, q, c_1, c_2, e_1, e_2]$ to $[x', q', c_1', c_2', e_1', e_2']$, $u \in \{x, \varepsilon\}$, if M has a rule $(x, q, c_1', c_2', q', e_1', e_2', g)$. Here $u = x$ if $g = +1$, and $u = \varepsilon$, $x' = x$ if $g = 0$. Γ then validates if c_1', and c_2' have been scanned on the counter tapes and then updates these tapes to reflect the values in e_1',and e_2'. If the input head moved ($g = +1$), the symbol x is added to the string of the master component, and so on.

We now present the process outlined above in more details. For the remainder of this section we use the layout shown in Figure 2 to present the configurations of Γ. The component strings are identified in the figure by the names of the components in Γ; these names will be replaced by the actual strings. As mentioned earlier the 11 original grammars have the word 'original' in their names.

We number the steps of the derivation so that we can refer to them in a convenient manner. Such a numbering is shown parenthetically on top of the $\Longrightarrow$ operator.

The initial configuration of Γ (having the respective axiom in each component) is rewritten as follows. There are nondeterministic rewriting choices in several components as shown in Figure 3. Here u_1, u_2, u_3, represent the original $P_1^{c_1}$, $P_2^{c_1}$, $P_3^{c_1}$ components and their copycat grammars; they can either rewrite to query components that simulate the rules in the master grammar or they can rewrite to query a helper component in the $P_4^{c_1}$ section. u_1', u_2', u_3', represent the original $P_1^{c_2}$, $P_2^{c_2}$, $P_3^{c_2}$ components and their modified copy grammars; they can either rewrite to query helper grammars that contain rules similar to the master grammar or they can rewrite to query helpers in the $P_4^{c_2}$ group. In this case if any of the components rewrite to query the $P_4^{c_1}$ or $P_4^{c_2}$ helpers the system will block because none of the components requesting strings from $P_4^{c_1}$or $P_4^{c_2}$ have a rewriting rule for S_4. Therefore, the only first step that will lead to a successful derivation is the one shown in Figure 4. We then continue as shown in Figures 5 and 6.

Now we have yet another nondeterministic rewriting choice in several components, as depicted in Figure 7. Here u_1, u_2, u_3, represent the original and helper components for $P_1^{c_1}$, $P_2^{c_1}$, $P_3^{c_1}$; they can rewrite and query their collaborating grammars that mimic either the rules in the master or $P_4^{c_1}$ components. u_1', u_2', u_3', represent the original and helper components for $P_1^{c_2}$, $P_2^{c_2}$, $P_3^{c_2}$; they can rewrite and query their matching component that simulate the master or $P_4^{c_2}$ rules. The master grammar and all of the helper components have only one rewriting choice, to query their corresponding P_{a_1} component, or to rewrite to the non-terminal C. $P_1^{c_1}$, $P_2^{c_1}$, $P_3^{c_1}$, $P_1^{c_2}$, $P_2^{c_2}$, and $P_3^{c_2}$, could have rewritten to query their corresponding component grammars in the master grammar helpers or could have rewritten to query $P_4^{c_1}$or $P_4^{c_2}$. The former choice would result in a blocked derivation due to the introduction of circular queries. This is the first step that makes use of the reset queries in the section of grammars that copies the rules of the master. The only possible step that will lead to a successful derivation is the one in Figure 8.

It is at this point that Γ can start to simulate the 2-counter Machine M. The configuration described above represents the initial state of M with 0 stored in both counters. If M has a rule $(x, q_0, Z, Z, q, e_1, e_2, g)$, and so can enter the state q by reading input x and the counter symbols are both Z, then the master grammar can chose to introduce the string $u[x', q, Z, Z, e_1, e_2]$. If the input head of M changes to $g = +1$, then $u = x$ and a new symbol x' gets scanned onto the input tape, but if the input head does not move ($g = 0$), then $u = \varepsilon$, $x' = x$, and the symbol x is scanned on the input tape. We thus continue the derivation as shown in Figures 9 and 10.

The original $P_1^{c_1}$, $P_4^{c_1}$, $P_1^{c_2}$, and $P_4^{c_2}$, components modify the number of A symbols in their respective strings according to e_1 and e_2. $P_1^{c_1}$ and $P_1^{c_2}$ introduce AAC, AC, C whenever e_1 and e_2 are, $+1$, 0, or -1,

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\begin{bmatrix}
P_{GM_{Original}} & \\
P^{c_1}_{GM_{S1}} & P^{c_1}_{GM_{S1H2(S4)}} \\
P^{c_1}_{GM_{S1H3(S4)}} & P^{c_1}_{GM_{S1(S2)}} \\
P^{c_1}_{GM_{S1(S3)}} & P^{c_1}_{GM_{S2}} \\
P^{c_1}_{GM_{S3}} & P^{c_1}_{GM_{PA1S1}} \\
P^{c_1}_{GM_{PA1S1H2}} & P^{c_1}_{GM_{PA1S1H2}} \\
P^{c_1}_{GM_{PA1S1(S2)}} & P^{c_1}_{GM_{PA1S1(S3)}} \\
P^{c_1}_{GM_{PA1S2}} & P^{c_1}_{GM_{PA1S3}} \\
P^{c_2}_{GM_{S1}} & P^{c_2}_{GM_{S1H2(S4)}} \\
P^{c_2}_{GM_{S1H3(S4)}} & P^{c_2}_{GM_{S1(S2)}} \\
P^{c_2}_{GM_{S1(S3)}} & P^{c_2}_{GM_{S2}} \\
P^{c_2}_{GM_{S3}} & P^{c_2}_{GM_{PA1S1}} \\
P^{c_2}_{GM_{PA1S1H2}} & P^{c_2}_{GM_{PA1S1H3}} \\
P^{c_2}_{GM_{PA1S1S2}} & P^{c_2}_{GM_{S1(S3)}} \\
P^{c_2}_{GM_{PA1S2}} & P^{c_2}_{GM_{PA1S3}} \\
P^{c_1}_{1_{OriginalS_1}} & P^{c_1}_{1_{S_1H_2(S_4)}} \\
P^{c_1}_{1_{S_1H_3(S_4)}} & P^{c_1}_{1_{S_1}(S_2)} \\
P^{c_1}_{1_{S_1}(S_3)} & P^{c_1}_{2_{OriginalS_2}} \\
P^{c_1}_{3_{OriginalS3}} & P^{c_1}_{4_{OriginalS_4}} \\
P^{c_1}_{4_{S_1H_2(S_4)}} & P^{c_1}_{4_{S_1H_3(S_4)}} \\
P^{c_1}_{4_{S_2}} & P^{c_1}_{4_{S_3}} \\
P^{c_1}_{4SpecialHelper1_{S1S2}} & P^{c_1}_{4SpecialHelper2_{S1S3}} \\
P^{c_2}_{1_{OriginalS_1}} & P^{c_2}_{1_{S_1H_2(S_4)}} \\
P^{c_2}_{1_{S_1H_3(S_4)}} & P^{c_2}_{1_{S_1}(S_2)} \\
P^{c_2}_{1_{S_1}(S_3)} & P^{c_2}_{2_{OriginalS_2}} \\
P^{c_2}_{3_{OriginalS3}} & P^{c_2}_{4_{OriginalS4}} \\
P^{c_2}_{4_{S_1H_2(S_4)}} & P^{c_2}_{4_{S_1H_3(S_4)}} \\
P^{c_2}_{4_{S_2}} & P^{c_2}_{4_{S_3}} \\
P^{c_2}_{4SpecialHelper1_{S1S2}} & P^{c_2}_{4SpecialHelper2_{S1S3}} \\
P_{a_1 Original} & \\
 & P^{c_1}_{a_1 GMS_1H_2(S_4)} \\
P^{c_1}_{a_1 GMS_1H_2(S_4)} & P^{c_1}_{a_1 GMS_1H_3(S_4)} \\
P^{c_1}_{a_1 GMS_1(S_2)} & P^{c_1}_{a_1 GMS_1(S_3)} \\
P^{c_1}_{a_1 GMS_2} & P^{c_1}_{a_1 GMS_3} \\
P^{c_2}_{a_1 GMS_1} & P^{c_2}_{a_1 GMS_1H_2(S_4)} \\
P^{c_2}_{a_1 GMS_1H_3(S_4)} & P^{c_2}_{a_1 GMS_1(S_2)} \\
P^{c_2}_{a_1 GMS_1(S_3)} & P^{c_2}_{a_1 GMS_2} \\
P^{c_2}_{a_1 GMS_3} & P_{a_{2Original}} \\
Reset_{GM^{c_1}_{P_{a_1}\,S1}} & Reset_{GM^{c_1}_{P_{a_1}\,S1H2(S4)}} \\
Reset_{GM^{c_1}_{P_{a_1}\,S1H3(S4)}} & Reset_{GM^{c_1}_{P_{a_1}\,S1(S2)}} \\
Reset_{GM^{c_1}_{P_{a_1}\,S1(S3)}} & Reset_{GM^{c_1}_{P_{a_1}\,S2}} \\
Reset_{GM^{c_1}_{P_{a_1}\,S3}} & Reset_{GM^{c_2}_{P_{a_1}\,S1}} \\
Reset_{GM_{P^{c_2}_{a_1 S1H2(S4)}}} & Reset_{GM^{c_2}_{P_{a_1}\,S1H3(S4)}} \\
Reset_{GM^{c_2}_{P_{a_1}\,S1(S2)}} & Reset_{GM^{c_2}_{P_{a_1}\,S1(S3)}} \\
Reset_{GM_{P_{a_1}\,S2^{c_2}}} & Reset_{GM_{P^{c_2}_{a_1 S3}}} \\
Reset_{P^{c_1}_{1_{S_1H_2(S_4)}}} & Reset_{P^{c_1}_{1_{S_1H_3(S_4)}}} \\
Reset_{P^{c_2}_{1_{S_1H_2(S_4)}}} & Reset_{P^{c_2}_{1_{S_1H_3(S_4)}}} \\
Reset_{P^{c_1}_{4_{S_1H_2(S_4)}}} & Reset_{P^{c_1}_{4_{S_1H_3(S_4)}}} \\
Reset_{P^{c_2}_{4_{S_1H_2(S_4)}}} & Reset_{P^{c_2}_{4_{S_1H_3(S_4)}}}
\end{bmatrix}
$$

Figure 2: Compact representation for configurations in our CF-PCGS that simulates a 2-counter machine.

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\left[\begin{array}{cc}
S & \\
S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\
S_1 & S_1 \\ S_1 & S_1 \\ S_1 & S_2 \\ S_3 & S_4 \\ S_4 & S_4 \\ S_4 & S_4 \\ S_4 & S_4 \\
S_1 & S_1 \\ S_1 & S_1 \\ S_1 & S_2 \\ S_3 & S_4 \\ S_4 & S_4 \\ S_4 & S_4 \\ S_4 & S_4 \\
S & \\
 & S \\
S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\ S & S \\
U & U \\ U & U \\ T & T \\ T & T
\end{array}\right]
\xRightarrow{(1)}
\left[\begin{array}{cc}
[I] & \\
[I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\ [I] & [I] \\
u_1 & u_1 \\ u_1 & u_1 \\ u_1 & u_2 \\
u_3 & S_4^{(1)} \\ S_4^{(1)} & S_4^{(1)} \\ S_4^{(1)} & S_4^{(1)} \\ S_4 & S_4 \\
u_1' & u_1' \\ u_1' & u_1' \\ u_1' & u_2' \\
u_3' & S_4^{(1)} \\ S_4^{(1)} & S_4^{(1)} \\ S_4^{(1)} & S_4^{(1)} \\ S_4 & S_4 \\
Q_{GM_{original}} & \\
 & Q^{c_1}_{GM_{P_{a_1}S_1}} \\
Q^{c_1}_{GM_{P_{a_1}S_1H_2(S_4)}} & Q^{c_1}_{GM_{P_{a_1}S_1H_3(S_4)}} \\
Q^{c_1}_{GM_{P_{a_1}S_1(S_2)}} & Q^{c_1}_{GM_{P_{a_1}S_1(S_3)}} \\
Q^{c_1}_{GM_{P_{a_1}S_2}} & Q^{c_1}_{GM_{P_{a_1}S_3}} \\
Q^{c_2}_{GM_{P_{a_1}S_1}} & Q^{c_2}_{GM_{P_{a_1}S_1H_2(S_4)}} \\
Q^{c_2}_{GM_{P_{a_1}S_1H_3(S_4)}} & Q^{c_2}_{GM_{P_{a_1}S_1(S_2)}} \\
Q^{c_2}_{GM_{P_{a_1}S_1(S_3)}} & Q^{c_2}_{GM_{P_{a_1}S_2}} \\
Q^{c_2}_{GM_{P_{a_1}S_3}} & S^{(3)} \\
<I> & <I> \\ <I> & <I> \\ <I> & <I> \\ <I> & <I> \\ <I> & <I> \\ <I> & <I> \\ <I> & <I> \\
U_1 & U_1 \\ U_1 & U_1 \\ T_1 & T_1 \\ T_1 & T_1
\end{array}\right]
$$

Figure 3: PCGS simulation of a 2-counter Turing machine: Step 1 (nondeterministic).

respectively, while $P_4^{c_1}$ and $P_4^{c_2}$ remove an A. The system thus adjusts the counters and if they decrement below 0 the derivation blocks.

The original grammars $P_2^{c_1}$, $P_3^{c_1}$, $P_2^{c_2}$, and $P_3^{c_2}$ verify the number of A symbols in their respective strings to see if they agree with c_1, c_2. Γ now starts to validate the value stored in the first counter (the second counter will be verified in exactly the same way). If $c_1 = Z$, then we have the following string $\alpha[x', q, Z, c_2, e_1, e_2]$ in $P_2^{c_1}$, $P_3^{c_1}$, which means the number of A symbols in α is 0. If this is not true the system blocks because in the next step $P_3^{c_1}$ would rewrite $[x', q, Z, c_2, e_1, e_2]$ to a (a terminal symbol), and it does not have a rewriting rule for A. If $c_1 = B$ then we have the following string $\alpha[x', q, B, c_2, e_1, e_2]$, where the there is at least one A in the string α. If there is no A then the system will block because $P_2^{c_2}$ does not have an applicable rewriting rule for any other non-terminal.

In the following step (Figure 12) we use the new rewriting rule $S_1 \rightarrow Q_{4SpecialHelper1}$ so its role in $P^{c_1}_{1_{S_1(S_2)}}$, $P^{c_1}_{1_{S_1(S_3)}}$ $P^{c_2}_{1_{S_1(S_2)}}$, and $P^{c_2}_{1_{S_1(S_3)}}$, components becomes apparent. This step ensures that $P^{c_1}_{2S_{2original}}$, $P^{c_1}_{2S_{3original}}$, $P^{c_2}_{2S_{2original}}$, $P^{c_3}_{2S_{3original}}$ receive the correct strings in Step 14.

The following step (Figure 13) is a communication step. It allows two of the $P_1^{c_1}$ and $P_1^{c_2}$ helper

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\begin{bmatrix}
S & \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S_1 & S_1 \\
S_1 & S_1 \\
S_1 & S_2 \\
S_3 & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
S_1 & S_1 \\
S_1 & S_1 \\
S_1 & S_2 \\
S_3 & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
S & \\
 & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
U & U \\
U & U \\
T & T \\
T & T
\end{bmatrix}
\stackrel{(1)}{\Longrightarrow}
\begin{bmatrix}
[I] & \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
Q^{c_1}_{GM_{S_1}} & Q^{c_1}_{GM_{S_1H_2(S_4)}} \\
Q^{c_1}_{GM_{S_1H_3(S_4)}} & Q^{c_1}_{GM_{S_1}S_2} \\
Q^{c_1}_{GM_{S_1S_3}} & Q^{c_1}_{GM_{S_2}} \\
Q^{c_1}_{GM_{S_3}} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4 & S_4 \\
Q^{c_2}_{GM_{S_1}} & Q^{c_2}_{GM_{S_1H_2(S_4)}} \\
Q^{c_2}_{GM_{S_1H_3(S_4)}} & Q^{c_2}_{GM_{S_1}S_2} \\
Q^{c_2}_{GM_{S_1S_3}} & Q^{c_2}_{GM_{S_2}} \\
Q^{c_2}_{GM_{S_3}} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4 & S_4 \\
Q_{GM_{original}} & \\
 & Q^{c_1}_{GM_{P_{a_1}}S_1} \\
Q^{c_1}_{GM_{P_{a_1}}S_1H_2(S_4)} & Q^{c_1}_{GM_{P_{a_1}}S_1H_3(S_4)} \\
Q^{c_1}_{GM_{P_{a_1}}S_1(S_2)} & Q^{c_1}_{GM_{P_{a_1}}S_1(S_3)} \\
Q^{c_1}_{GM_{P_{a_1}}S_2} & Q^{c_1}_{GM_{P_{a_1}}S_3} \\
Q^{c_2}_{GM_{P_{a_1}}S_1} & Q^{c_2}_{GM_{P_{a_1}}S_1H_2(S_4)} \\
Q^{c_2}_{GM_{P_{a_1}}S_1H_3(S_4)} & Q^{c_2}_{GM_{P_{a_1}}S_1(S_2)} \\
Q^{c_2}_{GM_{P_{a_1}}S_1(S_3)} & Q^{c_2}_{GM_{P_{a_1}}S_2} \\
Q^{c_2}_{GM_{P_{a_1}}S_3} & S^{(3)} \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
U_1 & U_1 \\
U_1 & U_1 \\
T_1 & T_1 \\
T_1 & T_1
\end{bmatrix}
$$

Figure 4: PCGS simulation of a 2-counter Turing machine: Step 1.

grammars that are holding intermediate strings to communicate with the components that will be used for the derivation of the original $P_2^{c_1}$, $P_3^{c_1}$,$P_2^{c_2}$, and $P_3^{c_2}$ components. In the above step two of the $P_4^{c_1}$, and two of the $P_4^{c_2}$ helpers use the new rewriting rule $S_2 \to S_2$ in order to avoid the introduction of a circular query. We continue as in Figures 14 and 15.

Similar to the first step in the derivation in Step 13 the P_1, P_2, and P_3 original and helper components have a nondeterministic choice. They could rewrite to either the original, or helper forms of Q_m, or $Q_4^{c_1}$ and $Q_4^{c_2}$. If any of these symbols is not Q_m, then the system will block after the communication step. The reset grammars now rewrite to request strings from there matching helper grammars that simulate rules in the master grammar. During the next step the query will reset the components that have $GM_{P_{a1}}$ in their labels (see Figure 16).

If αC and βC contain the same number of A symbols as stored in the counters of M, and if M is in the accepting state ($q = q_F$), then the system can either rewrite to a terminal string by using the rule

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

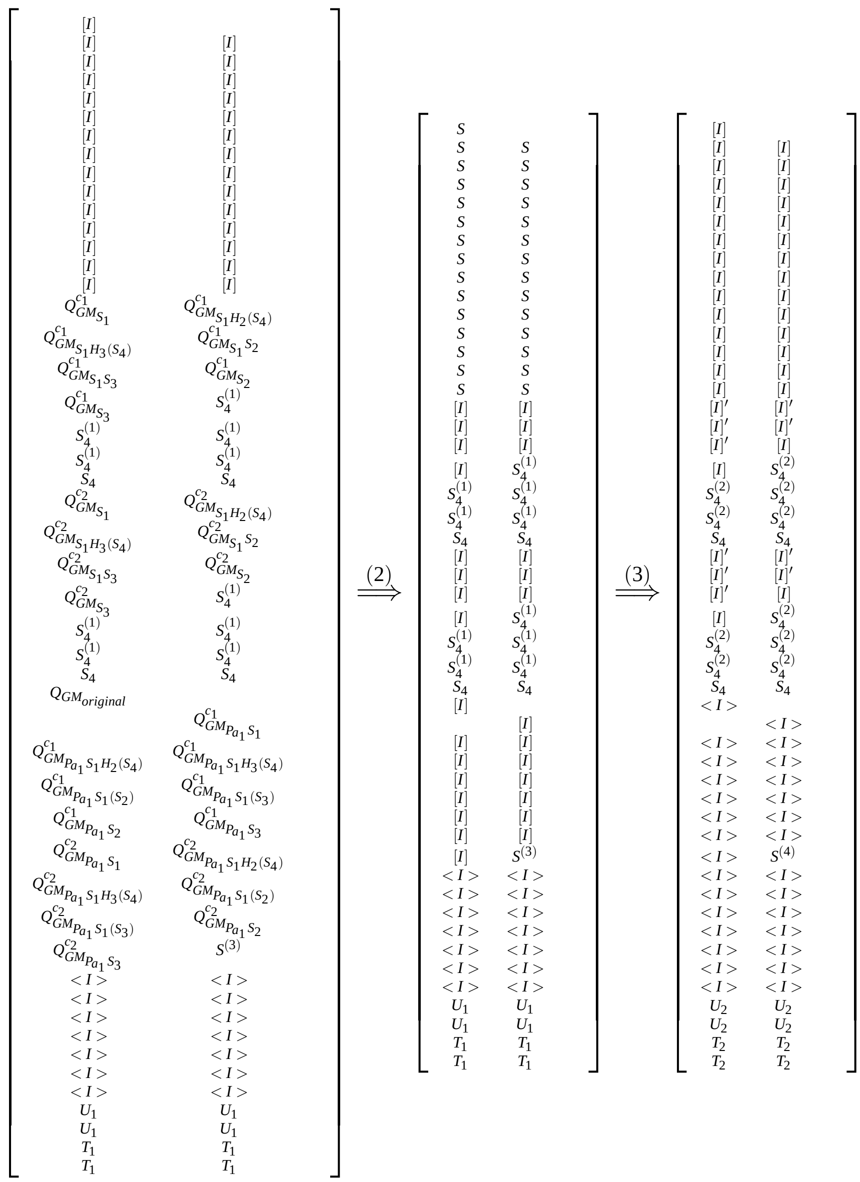

Figure 5: PCGS simulation of a 2-counter Turing machine: Steps 2 and 3.

$< x', q_F, Z, Z, e_1, e_2 > \rightarrow x'$ in G_m, or continue; otherwise the system has no chance but to continue the derivation. If the system continues the derivation then the input head of M will move to the right, and the symbol x' will be left behind. Then x' will become part of the string generated by Γ by using the rule: $< x', q, Z, Z, e_1, e_2 > \rightarrow x[y, q', c_1', c_2', e_1', e_2']$. If the scanned symbol does not change the input head will not move, and G_m can then use the following rule: $< x', q, Z, Z, e_1, e_2 > \rightarrow [x', q', c_1', c_2', e_1', e_2']$. The tuple (x, i, j) will represent the current state of the storage tapes of M, where i and j are integers that correspond to the number of A in the counters; these numbers will continue to increment and decrement according to the values of e_1 and e_2. The system will continue to loop and compare the number of A symbols in its counters to those in the grammar system indefinitely or can chose to stop (when permitted) as described above. We conclude that every successful computation of M has a matching successful derivation in Γ, and vice versa.

Note finally that this construction will not accept the empty string even if this string is in $\mathscr{L}(M)$. In

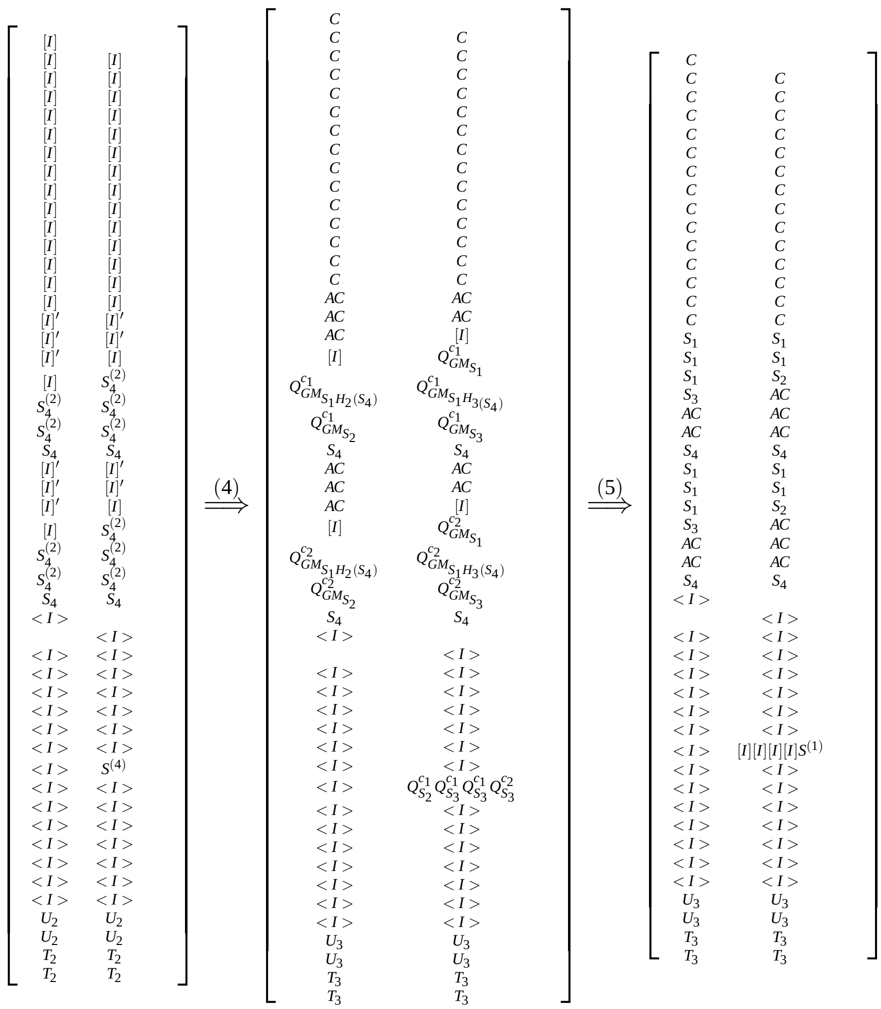

Figure 6: PCGS simulation of a 2-counter Turing machine: Steps 4 and 5.

such a case Γ can be modified to accept the empty string simply by adding the rule $S \to \varepsilon$ to its master grammar.

5 Conclusion

PCGS offer an inherently concurrent model for describing formal languages. It is precisely because of this inherent parallelism that one of our longer term interest is to exploit this model in general (and CF-PCGS in particular) in formal methods. Before this can even begin however several formal language questions need to be addressed. One of them is the generative power of CF-PCGS

Recall that the result regarding the expressiveness of synchronized CF-PCGS makes them Turing complete [5]. We noted however that the Turing complete proof used broadcast communication and not one-step communication and we were secretly hoping that the second result is the correct one (since this would give CF-PCGS a better chance to be useful in formal methods). This turned out in the end not to be the case. Indeed, we showed that the Turing completeness result is correct regardless of the communication style used, though the simulation that uses one-step communication is substantially more complex than was originally thought.

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\begin{bmatrix}
C & \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
S_1 & S_1 \\
S_1 & S_1 \\
S_1 & S_2 \\
S_3 & AC \\
AC & AC \\
AC & AC \\
S_4 & S_4 \\
S_1 & S_1 \\
S_1 & S_1 \\
S_1 & S_2 \\
S_3 & AC \\
AC & AC \\
AC & AC \\
S_4 & S_4 \\
<I> & \\
 & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & [I][I][I][I]S^{(1)} \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
U_3 & U_3 \\
U_3 & U_3 \\
T_3 & T_3 \\
T_3 & T_3
\end{bmatrix}
\xRightarrow{(6)}
\begin{bmatrix}
Q_{Pa_1\,Original} & \\
Q^{c_1}_{Pa_1\,S1} & Q^{c_1}_{Pa_1\,S1H2(S4)} \\
Q^{c_1}_{Pa_1\,S1H3(S4)} & Q^{c_1}_{Pa_1\,S1(S2)} \\
Q^{c_1}_{Pa_1\,S1(S3)} & Q^{c_1}_{Pa_1\,S2} \\
Q^{c_1}_{Pa_1\,S3} & Q_{Reset^{c_1}_{GM_{Pa_1}\,S1}} \\
Q_{Reset^{c_1}_{GM_{Pa_1}\,S1H2(S4)}} & Q_{Reset^{c_1}_{GM_{Pa_1}\,S1H3(S4)}} \\
Q_{Reset^{c_1}_{GM_{Pa_1}\,S1(S2)}} & Q_{Reset^{c_1}_{GM_{Pa_1}\,S1(S3)}} \\
Q_{Reset^{c_1}_{GM_{Pa_1}\,S2}} & Q_{Reset^{c_1}_{GM_{Pa_1}\,S3}} \\
Q^{c_2}_{Pa_1\,S1} & Q^{c_2}_{Pa_1\,S1H2(S4)} \\
Q^{c_2}_{Pa_1\,S1H3(S4)} & Q^{c_2}_{Pa_1\,S1(S2)} \\
Q^{c_2}_{Pa_1\,S1(S3)} & Q^{c_2}_{Pa_1\,S2} \\
Q^{c_2}_{Pa_1\,S3} & Q_{Reset^{c_2}_{GM_{Pa_1}\,S1}} \\
Q_{Reset^{c_2}_{GM_{Pa_1}\,S1H2(S4)}} & Q_{Reset^{c_2}_{GM_{Pa_1}\,S1H3(S4)}} \\
Q_{Reset^{c_2}_{GM_{Pa_1}\,S1(S2)}} & Q_{Reset^{c_2}_{GM_{Pa_1}\,S1(S3)}} \\
Q_{Reset^{c_2}_{GM_{Pa_1}\,S2}} & Q_{Reset}\,{}_{GM^{c_2}_{Pa_1}\,S3} \\
u_1 & u_1 \\
u_1 & u_1 \\
u_1 & u_2 \\
u_3 & aC \\
aC & aC \\
aC & aC \\
S_4 & S_4 \\
u'_1 & u'_1 \\
u'_1 & u'_1 \\
u'_1 & u'_2 \\
u'_3 & aC \\
aC & aC \\
aC & aC \\
S_4 & S_4 \\
<I> & \\
 & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & [I][I][I][I]S^{(2)} \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
U_4 & U_4 \\
U_4 & U_4 \\
T_4 & T_4 \\
T_4 & T_4
\end{bmatrix}
$$

Figure 7: PCGS simulation of a 2-counter Turing machine: Step 6 (nondeterministic).

We first examined one system designed earlier (using broadcast communication) to show Turing completeness [5]. We explained that such an interpretation of communication steps modifies the power of the PCGS and hence this simulation does not work if one-step communication is used (Section 3.1). We then proceeded to design a system that uses a similar approach, except that we created an arrangement that would allow one component to be queried by one and only one grammar during each communication step, thus eliminating the need for broadcast communication. In order to do this we created a number of helper components that act as support systems for the original component grammars; the role of the helpers was to create and hold intermediate strings until they were requested from their corresponding original grammar. In order to get the construction to work we used a number of different strategies, as follows:

1. A number of copycat components were created. They contain rules similar to the original components. These components derive the same strings during the same steps as the original components,

$$
\left[\begin{array}{cc}
C & \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
S_1 & S_1 \\
S_1 & S_1 \\
S_1 & S_2 \\
S_3 & AC \\
AC & AC \\
AC & AC \\
S_4 & S_4 \\
S_1 & S_1 \\
S_1 & S_1 \\
S_1 & S_2 \\
S_3 & AC \\
AC & AC \\
AC & AC \\
S_4 & S_4 \\
<I> & \\
 & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & [I][I][I][I]S^{(1)} \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
U_3 & U_3 \\
U_3 & U_3 \\
T_3 & T_3 \\
T_3 & T_3
\end{array}\right]
\overset{(6)}{\Longrightarrow}
\left[\begin{array}{cc}
Q_{Pa_1\,Original} & \\
Q^{c_1}_{Pa_1\,S1} & Q^{c_1}_{Pa_1\,S1H2(S4)} \\
Q^{c_1}_{Pa_1\,S1H3(S4)} & Q^{c_1}_{Pa_1\,S1(S2)} \\
Q^{c_1}_{Pa_1\,S1(S3)} & Q^{c_1}_{Pa_1\,S2} \\
Q^{c_1}_{Pa_1\,S3} & Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1} \\
Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1H2(S4)} & Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1H3(S4)} \\
Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1(S2)} & Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1(S3)} \\
Q_{Reset^{c_1}_{GM_{Pa_1}}\,S2} & Q_{Reset^{c_1}_{GM_{Pa_1}}\,S3} \\
Q^{c_2}_{Pa_1\,S1} & Q^{c_2}_{Pa_1\,S1H2(S4)} \\
Q^{c_2}_{Pa_1\,S1H3(S4)} & Q^{c_2}_{Pa_1\,S1(S2)} \\
Q^{c_2}_{Pa_1\,S1(S3)} & Q^{c_2}_{Pa_1\,S2} \\
Q^{c_2}_{Pa_1\,S3} & Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1} \\
Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1H2(S4)} & Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1H3(S4)} \\
Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1(S2)} & Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1(S3)} \\
Q_{Reset^{c_2}_{GM_{Pa_1}}\,S2} & Q_{Reset}\,{}_{GM^{c_2}_{Pa_1}\,S3} \\
u_1 & u_1 \\
u_1 & u_1 \\
u_1 & u_2 \\
u_3 & aC \\
aC & aC \\
aC & aC \\
S_4 & S_4 \\
u'_1 & u'_1 \\
u'_1 & u'_1 \\
u'_1 & u'_2 \\
u'_3 & aC \\
aC & aC \\
aC & aC \\
S_4 & S_4 \\
<I> & \\
 & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & [I][I][I][I]S^{(2)} \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
U_4 & U_4 \\
U_4 & U_4 \\
T_4 & T_4 \\
T_4 & T_4
\end{array}\right]
$$

Figure 8: PCGS simulation of a 2-counter Turing machine: Step 6.

which allows for each of the original grammars to request the same string at the same time without the need to query the same component.

2. We introduced reset components, whose purpose is to reset some of the copycat grammars at precise steps in the derivation in order to fix synchronization issues.

3. We used waiting rules to ensure that communication steps would only be triggered at certain points in the derivation.

4. We used selective rewriting rules in conjunction with blocking, thus allows certain rewriting rules to be successful only at specific steps and ensures that no undesired strings are created.

Using these techniques we were able to construct a CF-PCGS capable of simulating an arbitrary 2-counter Turing machine, and so show that CF-PCGS are indeed Turing complete using either style

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\begin{bmatrix}
Q_{Pa_1\,Original} & \\
Q^{c_1}_{Pa_1\,S1} & Q^{c_1}_{Pa_1\,S1H2(S4)} \\
Q^{c_1}_{Pa_1\,S1H3(S4)} & Q^{c_1}_{Pa_1\,S1(S2)} \\
Q^{c_1}_{Pa_1\,S1(S3)} & Q^{c_1}_{Pa_1\,S2} \\
Q^{c_1}_{Pa_1\,S3} & Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1} \\
Q^{c_1}_{Reset_{GM_{Pa_1}}\,S1H2(S4)} & Q^{c_1}_{Reset_{GM_{Pa_1}}\,S1H3(S4)} \\
Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1(S2)} & Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1(S3)} \\
Q_{Reset^{c_1}_{GM_{Pa_1}}\,S2} & Q_{Reset^{c_1}_{GM_{Pa_1}}\,S3} \\
Q^{c_2}_{Pa_1\,S1} & Q^{c_2}_{Pa_1\,S1H2(S4)} \\
Q^{c_2}_{Pa_1\,S1H3(S4)} & Q^{c_2}_{Pa_1\,S1(S2)} \\
Q^{c_2}_{Pa_1\,S1(S3)} & Q^{c_2}_{Pa_1\,S2} \\
Q^{c_2}_{Pa_1\,S3} & Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1} \\
Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1H2(S4)} & Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1H3(S4)} \\
Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1(S2)} & Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1(S3)} \\
Q_{Reset^{c_2}_{GM_{Pa_1}}\,S2} & Q_{Reset^{c_2}_{GM_{Pa_1}}\,S3} \\
u_1 & u_1 \\
u_1 & u_1 \\
u_1 & u_2 \\
u_3 & aC \\
aC & aC \\
aC & aC \\
S_4 & S_4 \\
u_1' & u_1' \\
u_1' & u_1' \\
u_1' & u_2' \\
u_3' & aC \\
aC & aC \\
aC & aC \\
S_4 & S_4 \\
<I> & \\
 & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & [I][I][I][I]S^{(2)} \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
U_4 & U_4 \\
U_4 & U_4 \\
T_4 & T_4 \\
T_4 & T_4
\end{bmatrix}
\overset{(7)}{\Longrightarrow}
\begin{bmatrix}
<I> & \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
aC & aC \\
aC & S_4 \\
S_4 & aC \\
aC & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
aC & aC \\
aC & S_4 \\
S_4 & aC \\
aC & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
S & \\
 & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & [I][I][I][I]S^{(2)} \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
U_4 & U_4 \\
U_4 & U_4 \\
T_4 & T_4 \\
T_4 & T_4
\end{bmatrix}
$$

Figure 9: PCGS simulation of a 2-counter Turing machine: Step 7.

of communication (Theorem 1). Admittedly our construction is not as compact or elegant as the ones used in similar proofs [5, 4, 2], but it has the advantage of being correct according to the one-step communication model.

True, the result established in this paper is already known. Indeed, one other path of showing Turing completeness of returning CF-PCGS exists: one can take one of the constructions that show completeness of non-returning CF-PCGS [14, 6] and then convert such a construction into a returning CF-PCGS (a single construction for this conversion is known [8]).

Even so, our result has several advantages. For one thing we are doing it more efficiently. Note first that the conversion from non-returning to returning CF-PCGS [8] increases the number of components from n to $4n^2-3n+1$ [23]. One of the results showing Turing completeness of non-returning CF-PCGS [14] uses a construction with an arbitrary number of components, so that it proves that $\mathrm{RE} = \mathscr{L}(PC_*\mathrm{CF})$ instead of our $\mathrm{RE} = \mathscr{L}(PC_{95}\mathrm{CF})$. The other proof of Turing completeness for non-returning CF-PCGS

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\begin{bmatrix}
\langle I \rangle & \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
aC & aC \\
aC & S_4 \\
S_4 & aC \\
aC & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
aC & aC \\
aC & S_4 \\
S_4 & aC \\
aC & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
S_4 & S_4 \\
S & \\
 & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & [I][I][I][I]S^{(2)} \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
U_4 & U_4 \\
U_4 & U_4 \\
T_4 & T_4 \\
T_4 & T_4
\end{bmatrix}
\overset{(8)}{\Longrightarrow}
\begin{bmatrix}
u[x',q,Z,Z,e_1,e_2] & \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
aQ^{c_1}_{GM_{S1}} & aQ^{c_1}_{GM_{S1H2(S4)}} \\
aQ^{c_1}_{GM_{S1H3(S4)}} & S_4^{(1)} \\
S_4^{(1)} & aQ^{c_1}_{GM_{S2}} \\
aQ^{c_1}_{GM_{S3}} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4 & S_4 \\
aQ^{c_2}_{GM_{S1}} & aQ^{c_2}_{GM_{S1H2(S4)}} \\
aQ^{c_2}_{GM_{S1H3(S4)}} & S_4^{(1)} \\
S_4^{(1)} & aQ^{c_2}_{GM_{S2}} \\
aQ^{c_2}_{GM_{S3}} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4 & S_4 \\
Q_{GMoriginal} & Q^{c_1}_{GM_{S1(Pa_1 Helper)}} \\
Q^{c_1}_{GM_{S1H2(S4)(Pa_1 Helper)}} & Q^{c_1}_{GM_{S1H3(S4)(Pa_1 Helper)}} \\
Q^{c_1}_{GM_{S1(S2)(Pa_1 Helper)}} & Q^{c_1}_{GM_{S1(S3)(Pa_1 Helper)}} \\
Q^{c_1}_{GM_{S2(Pa_1 Helper)}} & Q^{c_1}_{GM_{S3(Pa_1 Helper)}} \\
Q^{c_2}_{GM_{S1(Pa_1 Helper)}} & Q^{c_2}_{GM_{S1H2(S4)(Pa_1 Helper)}} \\
Q^{c_2}_{GM_{S1H3(S4)(Pa_1 Helper)}} & Q^{c_2}_{GM_{S1(S2)(Pa_1 Helper)}} \\
Q^{c_2}_{GM_{S1(S3)(Pa_1 Helper)}} & Q^{c_2}_{GM_{S2(Pa_1 Helper)}} \\
Q^{c_2}_{GM_{S1(S3)(Pa_1 Helper)}} & [I][I][I][I]S^{(3)} \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
\langle I \rangle & \langle I \rangle \\
U_5 & U_5 \\
U_5 & U_5 \\
T_5 & T_5 \\
T_5 & T_5
\end{bmatrix}
$$

Figure 10: PCGS simulation of a 2-counter Turing machine: Step 8.

[6] provides a PCGS with 6 components, which is equivalent to $4 \times 6^2 - (3 * 6) + 1 = 127$ components for the returning case, so this shows $\mathrm{RE} = \mathscr{L}(PC_{127}\mathrm{CF})$ versus our $\mathrm{RE} = \mathscr{L}(PC_{95}\mathrm{CF})$. In both cases our result is tighter.

It is apparent that broadcast communication allows for a more compact CF-PCGS for certain languages. Indeed, one could compare our 2-counter Turing machine simulation (featuring as many as 95 components) with the broadcast communication-enabled simulation [5] (having only 11 components). A further study on simulating non-returning CF-PCGS using the returning variant [23] also determined that the use of broadcast communication (called this time "homogenous queries") results in a PCGS with fewer components (though this time the number of components remain of the same order of magnitude in the general case). We now effectively showed that this (reducing the number of components) is the sole advantage of broadcast communication, which does not otherwise increase the power of CF-PCGS. It would also be interesting to see whether our construction can be made even more concise, which we

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\left[\begin{array}{cc}
u[x',q,Z,Z,e_1,e_2] & \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
aQ^{c1}_{GM_{S1}} & aQ^{c1}_{GM_{S1H2(S4)}} \\
aQ^{c1}_{GM_{S1H3(S4)}} & S_4^{(1)} \\
S_4^{(1)} & aQ^{c1}_{GM_{S2}} \\
aQ^{c1}_{GM_{S3}} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4 & S_4 \\
aQ^{c2}_{GM_{S1}} & aQ^{c2}_{GM_{S1H2(S4)}} \\
aQ^{c2}_{GM_{S1H3(S4)}} & S_4^{(1)} \\
S_4^{(1)} & aQ^{c2}_{GM_{S2}} \\
aQ^{c2}_{GM_{S3}} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4 & S_4 \\
Q_{GMoriginal} & \\
 & Q^{c1}_{GM_{S1(Pa_1 Helper)}} \\
Q^{c1}_{GM_{S1H2(S4)(Pa_1 Helper)}} & Q^{c1}_{GM_{S1H3(S4)(Pa_1 Helper)}} \\
Q^{c1}_{GM_{S1(S2)(Pa_1 Helper)}} & Q^{c1}_{GM_{S1(S3)(Pa_1 Helper)}} \\
Q^{c1}_{GM_{S2(Pa_1 Helper)}} & Q^{c1}_{GM_{S3(Pa_1 Helper)}} \\
Q^{c2}_{GM_{S1(Pa_1 Helper)}} & Q^{c2}_{GM_{S1H2(S4)(Pa_1 Helper)}} \\
Q^{c2}_{GM_{S1H3(S4)(Pa_1 Helper)}} & Q^{c2}_{GM_{S1(S2)(Pa_1 Helper)}} \\
Q^{c2}_{GM_{S1(S3)(Pa_1 Helper)}} & Q^{c2}_{GM_{S2(Pa_1 Helper)}} \\
Q^{c2}_{GM_{S1(S3)(Pa_1 Helper)}} & [I][I][I][I]S^{(3)} \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
U_5 & U_5 \\
U_5 & U_5 \\
T_5 & T_5 \\
T_5 & T_5
\end{array}\right]
\overset{(9)}{\Longrightarrow}
\left[\begin{array}{cc}
S & \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
au[x',q,Z,Z,e_1,e_2] & au[x',q,Z,Z,e_1,e_2] \\
au[x',q,Z,Z,e_1,e_2] & S_4^{(1)} \\
S_4^{(1)} & au[x',q,Z,Z,e_1,e_2] \\
au[x',q,Z,Z,e_1,e_2] & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4 & S_4 \\
au[x',q,Z,Z,e_1,e_2] & au[x',q,Z,Z,e_1,e_2] \\
au[x',q,Z,Z,e_1,e_2] & S_4^{(1)} \\
S_4^{(1)} & au[x',q,Z,Z,e_1,e_2] \\
au[x',q,Z,Z,e_1,e_2] & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4 & S_4 \\
u[x',q,Z,Z,e_1,e_2] & \\
 & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & [I][I][I][I]S^{(3)} \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
U_5 & U_5 \\
U_5 & U_5 \\
T_5 & T_5 \\
T_5 & T_5
\end{array}\right]
$$

Figure 11: PCGS simulation of a 2-counter Turing machine: Step 9.

believe to be the case. Indeed, applying the techniques from this paper to another proof using broadcast communication [4] (and resulting in a system with only 5 components) is very likely to result in a smaller PCGS. We believe that our construction is general and so can be applied in this way with relative ease.

Indeed, the discussion above suggests that the techniques used in our approach are applicable not only to our construction but in a more general environment. That is, they appear to be useful for eliminating broadcast communication in general. Whether this is indeed the case and if so in what circumstances is an interesting open question.

On a practical side we note that CF-PCGS being Turing complete makes them too complex for formal methods (since nobody in their right mind will model a system using a formalism that is just as complex). We also note that most actual systems do not run their concurrent threads of execution in a fully synchronized manner. Therefore strong synchronization as implemented by synchronized CF-PCGS is unneeded.

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\left[\begin{array}{cc}
S & \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
au[x',q,Z,Z,e_1,e_2] & au[x',q,Z,Z,e_1,e_2] \\
au[x',q,Z,Z,e_1,e_2] & S_4^{(1)} \\
S_4^{(1)} & au[x',q,Z,Z,e_1,e_2] \\
au[x',q,Z,Z,e_1,e_2] & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4 & S_4 \\
au[x',q,Z,Z,e_1,e_2] & au[x',q,Z,Z,e_1,e_2] \\
au[x',q,Z,Z,e_1,e_2] & S_4^{(1)} \\
S_4^{(1)} & au[x',q,Z,Z,e_1,e_2] \\
au[x',q,Z,Z,e_1,e_2] & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4^{(1)} & S_4^{(1)} \\
S_4 & S_4 \\
u[x',q,Z,Z,e_1,e_2] & \\
 & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & u[x',q,Z,Z,e_1,e_2] \\
u[x',q,Z,Z,e_1,e_2] & [I][I][I][I]S^{(3)} \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
U_5 & U_5 \\
U_5 & U_5 \\
T_5 & T_5 \\
T_5 & T_5
\end{array}\right]
\overset{(10)}{\Longrightarrow}
\left[\begin{array}{cc}
[I] & \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
au[e_1]' & au[e_1]' \\
au[e_1]' & Q^{c_1}_{S1H2(S4)} \\
Q^{c_1}_{S1H3(S4)} & au[x',q,Z,Z,e_1,e_2] \\
aua & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4 & S_4 \\
au[e_2]' & au[e_2]' \\
au[e_2]' & Q^{c_2}_{S1H2(S4)} \\
Q^{c_2}_{S1H3(S4)} & au[x',q,Z,Z,e_1,e_2] \\
aua & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4 & S_4 \\
u<x',q,Z,Z,e_1,e_2> & \\
 & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & [I][I][I][I]S^{(4)} \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
U_6 & U_6 \\
U_6 & U_6 \\
T_6 & T_6 \\
T_6 & T_6
\end{array}\right]
$$

Figure 12: PCGS simulation of a 2-counter Turing machine: Step 10.

Both the arguments above (complexity and the nature of real-life synchronization) suggest that overall unsynchronized PCGS are more amenable to applications in formal methods, they being less powerful but still expressive enough to model complex, potentially recursive systems. They seem better suited for the particular task of system specification, as they are arguably closer to the way an actual concurrent system works.

Unfortunately unsynchronized PCGS have received little attention: They have been found to be weaker in terms of generative power compared to their synchronized counterparts, and then they have been effectively ignored. Substantial effort is therefore needed to study the language-theoretical properties of unsynchronized CF-PCGS before being able to use them in formal methods (or indeed anywhere else).

References

[1] S. D. BRUDA AND M. S. R. WILKIN, *Parse trees for context-free parallel communicating grammar systems*, in Proceedings of the 13th International Conference on Automation and Information (ICAI 12), Iasi, Romania, June 2012, pp. 144–149.

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\left[\begin{array}{cc}
[I] & \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
au[e_1]' & S_1 \\
S_1 & au[e_1]' \\
au[e_1]' & au[x',q,Z,Z,e_1,e_2] \\
aua & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4 & S_4 \\
au[e_2]' & S_1 \\
S_1 & au[e_2]' \\
au[e_2]' & au[x',q,Z,Z,e_1,e_2] \\
aua & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4 & S_4 \\
u<x',q,Z,Z,e_1,e_2> & \\
 & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & [I][I][I][I]S^{(4)} \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
U_6 & U_6 \\
U_6 & U_6 \\
T_6 & T_6 \\
T_6 & T_6
\end{array}\right]
\overset{(11)}{\Longrightarrow}
\left[\begin{array}{cc}
C & \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
\alpha C & Q^{c1}_{GM_{S1H2(S4)}} \\
Q^{c1}_{GM_{S1H3(S4)}} & \alpha C \\
\alpha C & au[x',q,Z,Z,e_1,e_2] \\
aua & Q^{c1}_{S1} \\
S_4(2) & S_4(2) \\
Q^{c1}_{S1(S2)} & Q^{c1}_{S1(S3)} \\
S_4 & S_4 \\
\beta C & Q^{c2}_{GM_{S1H2(S4)}} \\
Q^{c2}_{GM_{S1H3(S4)}} & \beta C \\
\beta C & au[x',q,Z,Z,e_1,e_2] \\
aua & Q^{c2}_{S1} \\
S_4(2) & S_4(2) \\
Q^{c2}_{S1(S2)} & Q_{S1(S3)}{}^{c2} \\
S_4 & S_4 \\
u<x',q,Z,Z,e_1,e_2> & \\
 & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & \gamma Q^{c1}_{S2}Q^{c1}_{S3}Q^{c2}_{S2}Q^{c2}_{S3}S(1) \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
U_7 & U_7 \\
U_7 & U_7 \\
T_7 & T_7 \\
T_7 & T_7
\end{array}\right]
$$

Figure 13: PCGS simulation of a 2-counter Turing machine: Step 11.

[2] E. CSUHAJ-VARJÚ, *On size complexity of context-free returning parallel communicating grammar systems*, in Where Mathematics, Computer Scients, Linguistics and Biology Meet, C. Martin-Vide and V. Mitrana, eds., Springer, 2001, pp. 37–49.

[3] E. CSUHAJ-VARJÚ, J. DASSOW, J. KELEMEN, AND G. PAUN, *Grammar Systems: a Grammatical Approach to Distribution and Cooperation*, Gordon and Breach Science Publishers S.A., 1994.

[4] E. CSUHAJ-VARJÚ, P. GHEORGHE, AND G. VASZIL, *PC grammar systems with five context-free components generate all recursively enumerable languages*, Theoretical Computer Science, 299 (2003), pp. 785–794.

[5] E. CSUHAJ-VARJÚ AND G. VASZIL, *On the computational completeness of context-free parallel communicating grammar systems*, Theoretical Computer Science, 215 (1999), pp. 349–358.

[6] E. CSUHAJ-VARJÚ AND G. VASZIL, *On the size complexity of non-returning context-free PC grammar systems*, in 11th International Workshop on Descriptional Complexity of Formal Systems (DCFS 2009), 2009, pp. 91–100.

[7] J. DASSOW, G. PAUN, AND G. ROZENBERG, *Grammar systems*, in Handbook of Formal Languages – Volume 2: Linear Modeling: Background and Applications, Springer, 1997, pp. 155–213.

[8] S. DUMITRESCU, *Nonreturning PC grammar systems can be simulated by returning systems*, Theoretical Computer Science, 165 (1996), pp. 463–474.

[9] P. C. FISCHER, *Turing machines with restricted memory access*, Information and Computation, 9 (1966),

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\left[\begin{array}{cc}
C & \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
\alpha C & Q^{c_1}_{GMS1H2(S4)} \\
Q^{c_1}_{GMS1H3(S4)} & \alpha C \\
\alpha C & au[x',q,Z,Z,e_1,e_2] \\
aua & Q^{c_1}_{S1} \\
S_4(2) & S_4(2) \\
Q^{c_1}_{S1(S2)} & Q^{c_1}_{S1(S3)} \\
S_4 & S_4 \\
\beta C & Q^{c_2}_{GMS1H2(S4)} \\
Q^{c_2}_{GMS1H3(S4)} & \beta C \\
\beta C & au[x',q,Z,Z,e_1,e_2] \\
aua & Q^{c_2}_{S1} \\
S_4(2) & S_4(2) \\
Q^{c_2}_{S1(S2)} & Q^{c_2}_{S1(S3)} \\
S_4 & S_4 \\
u < x',q,Z,Z,e_1,e_2 > & \\
 & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & \gamma Q^{c_1}_{S2} Q^{c_1}_{S3} Q^{c_2}_{S2} Q^{c_2}_{S3} S(1) \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
U_7 & U_7 \\
U_7 & U_7 \\
T_7 & T_7 \\
T_7 & T_7
\end{array}\right]
\overset{(12)}{\Longrightarrow}
\left[\begin{array}{cc}
C & \\
C & S \\
S & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & S \\
S & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
S_1 & C \\
C & S_1 \\
S_1 & S_2 \\
S_3 & \alpha C \\
S_4(2) & \S_4(2) \\
\alpha C & \alpha C \\
S_4 & S_4 \\
S_1 & C \\
C & S_1 \\
S_1 & S_2 \\
S_3 & \beta C \\
S_4(2) & S_4(2) \\
\beta C & \beta C \\
S_4 & S_4 \\
u < x',q,Z,Z,e_1,e_2 > & \\
 & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & u < x',q,Z,Z,e_1,e_2 > \\
u < x',q,Z,Z,e_1,e_2 > & \gamma' S(1) \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
U_7 & U_7 \\
U_7 & U_7 \\
T_7 & T_7 \\
T_7 & T_7
\end{array}\right]
$$

Figure 14: PCGS simulation of a 2-counter Turing machine: Step 12.

pp. 364–379.

[10] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability A Guide to the Theory of NP-Completeness*, Macmillan Higher Education, 1979.

[11] V. GEFFERT, *Context-free-like forms for the phrase-structure grammars*, in Mathematical Foundations of Computer Science, vol. 324 of Lecture Notes in Computer Science, Springer, 1988, pp. 309–317.

[12] G. KATSIRELOS, S. MANETH, N. NARODYTSKA, AND T. WALSH, *Restricted global grammar contraints*, in Principles and Practice of Constraint Programming (CP 2009), vol. 5732 of Lecture Notes in Computer Science, 2009, pp. 501–508.

[13] H. R. LEWIS AND C. H. PAPADIMITRIOU, *Elements of the Theory of Computation*, Prentice Hall, 2nd ed., 1998.

[14] N. MANDACHE, *On the computational power of context-free PCGS*, Theoretical Computer Science, 237 (2000), pp. 135–148.

[15] V. MIHALACHE, *On parallel communicating grammar systems with context-free components*, in Mathematical Linguistics and Related Topics, The Publishing House of the Romanian Academy of Science, 1994, pp. 258–270.

[16] V. MIHALACHE, *On the generative capacity of parallel communicating grammer systems with regular components*, tech. rep., Turku Centre for Computer Science, Turku, Finland, 1996.

[17] ——, *On the expressiveness of coverability trees for PC grammar systems*, in Grammatical Models of Multi-

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

$$
\begin{bmatrix}
C & \\
C & S \\
S & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & S \\
S & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
S_1 & C \\
C & S_1 \\
S_1 & S_2 \\
S_3 & \alpha C \\
S_4(2) & \S_4(2) \\
\alpha C & \alpha C \\
S_4 & S_4 \\
S_1 & C \\
C & S_1 \\
S_1 & S_2 \\
S_3 & \beta C \\
S_4(2) & S_4(2) \\
\beta C & \beta C \\
S_4 & S_4 \\
u < x', q, Z, Z, e_1, e_2 > & \\
 & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & \gamma' S(1) \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
U_7 & U_7 \\
U_7 & U_7 \\
T_7 & T_7 \\
T_7 & T_7
\end{bmatrix}
\overset{(13)}{\Longrightarrow}
\begin{bmatrix}
Q_{Pa_1} & \\
Q^{c_1}_{Pa_1 S1} & [I] \\
[I] & Q^{c_1}_{Pa_1 S1(S2)} \\
Q_{Pa_1 S1(S3)} & Q^{c_1}_{Pa_1 S2} \\
Q^{c_1}_{Pa_1 S3} & Q_{Reset^{c_1}_{GM_{Pa_1}} S1} \\
Q_{Reset^{c_1}_{GM_{Pa_1}} S1H2(S4)} & Q_{Reset^{c_1}_{GM_{Pa_1}} S1H3(S4)} \\
Q_{Reset^{C_1}_{GM_{Pa_1}} S1(S2)} & Q_{Reset^{c_1}_{GM_{Pa_1}} S1(S3)} \\
Q_{Reset_{GM^{c_1}_{Pa_1}} S2} & Q_{Reset^{c_1}_{GM_{Pa_1}} S3} \\
Q^{c_2}_{Pa_1 S1} & W \\
W & Q^{c_2}_{Pa_1 S1(S2)} \\
Q^{c_2}_{Pa_1 S1(S3)} & QA1C2S2 \\
Q^{c_2}_{Pa_1 S3} & Q^{c_2}_{Reset_{GM_{Pa_1}} S1} \\
Q_{Reset^{c_2}_{GM_{Pa_1}} S1H2(S4)} & Q_{Reset^{c_2}_{GM_{Pa_1}} S1H3(S4)} \\
Q_{Reset^{c_2}_{GM_{Pa_1}} S1(S2)} & Q_{Reset^{c_2}_{GM_{Pa_1}} S1(S3)} \\
Q_{Reset^{c_2}_{GM_{Pa_1}} S2} & Q_{Reset^{c_2}_{GM_{Pa_1}} S3} \\
Q^{c_1}_{S4S1} & W \\
W & Q^{c_1}_{S4P4SpecialHelper1_{S1S2}} \\
Q^{c_1}_{S4P4SpecialHelper2_{S1S3}} & Q^{c_1}_{S4S2} \\
Q^{c_1}_{S4S3} & \alpha C \\
S_4(2) & S_4(2) \\
\alpha C & \alpha C \\
S_4 & S_4 \\
Q^{c_2}_{S4S1} & W \\
W & Q^{c_2}_{S4P4SpecialHelper1_{S1S2}} \\
Q^{c_2}_{S4P4SpecialHelper2_{S1S3}} & Q^{c_2}_{S4S2} \\
Q^{c_2}_{S4S3} & \beta C \\
S_4(2) & S_4(2) \\
\beta C & \beta C \\
S_4 & S_4 \\
u < x', q, Z, Z, e_1, e_2 > & \\
 & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & \gamma' S(2) \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
Q^{c_1}_{S_1 S1H2(S4)} U_4 & Q^{c_1}_{S_1 S1H3(S4)} U_4 \\
Q^{c_2}_{S_1 S1H2(S4)} U_4 & Q^{c_2}_{S_1 S1H3(S4)} U_4 \\
Q^{c_1}_{S_4 S1H2(S4)} T_4 & Q^{c_1}_{S_4 S1H3(S4)} T_4 \\
Q^{c_2}_{S_4 S1H2(S4)} T_4 & Q^{c_2}_{S_4 S1H3(S4)} T_4
\end{bmatrix}
$$

Figure 15: PCGS simulation of a 2-counter Turing machine: Step 13.

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research

Agent Systems (Topics in Computer Mathematics), Gordon and Breach Science Publishers, 1999.

[18] D. PARDUBSKA AND M. PLATEK, *Parallel communicating grammar systems and analysis by reduction by restarting automata*, tech. rep., Deptartment of Computer Science, Comenius University, Bratislava, Slovakia, 2008.

[19] G. PAUN AND L. SANTEAN, *Parallel communicating grammar systems: the regular case*, Analele Universitatii din Bucuresti, Seria Matematica-Informatica, 2 (1989), pp. 55–63.

[20] G. PAUN AND L. SANTEAN, *Further remarks on parallel communicating grammar systems*, International Journal of Computer Mathematics, 34 (1990), pp. 187–203.

[21] L. SANTEAN, *Parallel communicating grammar systems*, Bulletion of the EATCS (Formal Language Theory Column), 1 (1990).

$$
\begin{bmatrix}
Q_{Pa_1} & \\
Q^{c_1}_{Pa_1\,S1} & Q^{c_1}_{Pa_1\,S1H2(S4)} \\
Q^{c_1}_{Pa_1\,S1H3(S4)} & Q^{c_1}_{Pa_1\,S1(S2)} \\
Q^{c_1}_{Pa_1\,S1(S3)} & Q^{c_1}_{Pa_1\,S2} \\
Q^{c_1}_{Pa_1\,S3} & Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1} \\
Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1H2(S4)} & Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1H3(S4)} \\
Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1(S2)} & Q_{Reset^{c_1}_{GM_{Pa_1}}\,S1(S3)} \\
Q_{Reset^{c_1}_{GM_{Pa_1}}\,S2} & Q_{Reset^{C_1}_{GM_{Pa_1}}\,S3} \\
Q^{c_2}_{Pa_1\,S1} & Q^{c_2}_{Pa_1\,S1H2(S4)} \\
Q^{c_2}_{Pa_1\,S1H3(S4)} & Q^{c_2}_{Pa_1\,S1(S2)} \\
Q^{c_2}_{Pa_1\,S1(S3)} & Q^{c_2}_{Pa_1\,S2} \\
Q^{c_2}_{Pa_1\,S3} & Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1} \\
Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1H2(S4)} & Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1H3(S4)} \\
Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1(S2)} & Q_{Reset^{c_2}_{GM_{Pa_1}}\,S1(S3)} \\
Q_{Reset^{c_2}_{GM_{Pa_1}}\,S2} & Q_{Reset^{C_2}_{GM_{Pa_1}}\,S3} \\
Q^{c_1}_{S4S1} & W \\
W & Q^{c_1}_{S4P4SpecialHelper1_{S1S2}} \\
Q^{c_1}_{S4P4SpecialHelper2_{S1S3}} & Q^{c_1}_{S4S2} \\
Q^{c_1}_{S4S3} & \alpha C \\
S_4(2) & S_4(2) \\
\alpha C & \alpha C \\
S_4 & S_4 \\
Q^{c_2}_{S4S1} & W \\
W & Q^{c_2}_{S4P4SpecialHelper1_{S1S2}} \\
Q^{c_2}_{S4P4SpecialHelper2_{S1S3}} & Q^{c_2}_{S4S2} \\
Q^{c_2}_{S4S3} & \beta C \\
S_4(2) & S_4(2) \\
\beta C & \beta C \\
S_4 & S_4 \\
u<x',q,Z,Z,e_1,e_2> & \\
 & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & \gamma' S(2) \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
Q^{c_1}_{S_1H2(S4)}U_4 & Q^{c_1}_{S_1H3(S4)}U_4 \\
Q^{c_2}_{S_1S1H2(S4)}U_4 & Q^{c_2}_{S_1H3(S4)}U_4 \\
Q^{c_1}_{S_4S1H2(S4)}T_4 & Q^{c_1}_{P_4S1H3(S4)}T_4 \\
Q^{c_2}_{P_4S1H2(S4)}T_4 & Q^{c_2}_{P_4S1H3(S4)}T_4
\end{bmatrix}
\xRightarrow{(14)}
\begin{bmatrix}
u<x',q,Z,Z,e_1,e_2> & \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & u<x',q,Z,Z,e_1,e_2> \\
u<x',q,Z,Z,e_1,e_2> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
\alpha' C & S_1 \\
S_1 & S_4 \\
S_4 & \alpha' C \\
\alpha' C & S_4 \\
S_4(2) & S_4(2) \\
S_4 & S_4 \\
S_4 & S_4 \\
\beta C & [I] \\
[I] & S_4 \\
S_4 & \beta C \\
\beta C & S_4 \\
S_4(2) & S_4(2) \\
S_4 & S_4 \\
S_4 & S_4 \\
S & \\
 & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & S \\
S & \gamma' S(2) \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
<I> & <I> \\
WU_4 & WU_4 \\
WU_4 & WU_4 \\
S_4^{(2)}T_4 & S_4^{(2)}T_4 \\
S_4^{(2)}T_4 & S_4^{(2)}T_4
\end{bmatrix}
$$

Figure 16: PCGS simulation of a 2-counter Turing machine: Step 14.

[22] F. L. TIPLEA, C. ENE, C. M. IONESCU, AND O. PROCOPIUC, *Some decision problems for parallel communicating grammar systems*, Theoretical Computer Science, 134 (1994), pp. 365–385.

[23] G. VASZIL, *On simulating non-returning PC grammar systems with returning systems*, Theoretical Computer Science, 209 (1997), pp. 319–329.

(c) Department of Computer Science, Bishop's University
http://cs.ubishops.ca/research