DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **8,** 2 (2016) 113–170

DOI: 10.1515/ausi-2016-0007

Parallel communicating grammar systems with context-free components are Turing complete for any communication model

Mary Sarah Ruth WILKIN
Department of Computer Science
Bishop's University
Sherbrooke, Quebec J1M 1Z7, Canada
email: swilkin@cs.ubishops.ca

Stefan D. BRUDA
Department of Computer Science
Bishop's University
Sherbrooke, Quebec J1M 1Z7, Canada
email: stefan@bruda.ca

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 usually 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.

Computing Classification System 1998: F.1.1, F.4.2, F.4.3
Mathematics Subject Classification 2010: 68Q45, 68Q10, 68Q42, 68Q15
Key words and phrases: formal languages, theory of computation, formal grammar, parallel communicating grammar system, Turing completeness

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 the master of the system and the other components 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 [6, 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 CF-PCGS 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 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 machine (yet another complete model [9]), thus proving that this kind of PCGS are Turing complete [4]. On close examination of the derivations of this PCGS simulating a 2-counter machine [4] we noticed that the 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; only then does the queried component returns to its axiom. Throughout the document we will refer to this style of communication as *broadcast communication* (also called immediate communication [24], though we prefer the term broadcast as being more intuitive). Later work uses a different definition, where the queried component returns to its axiom immediately after it is communicated [6]; we will refer to this type of communication as *one-step communication*. It follows that 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 machine [4] 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 machine simulation can be modified so that it works with one-step communication. The answer turns out to be affirmative. We present in Section 5.2 a PCGS that observes the one-step communication definition and at the same time simulates a 2-counter machine in a similar manner with the original construction [4]. 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.

This work was first published in a preliminary form as a technical report [25].

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 $A \to u$ where $A \in (\Sigma \cup N)^* N (\Sigma \cup N)^*$ and $u \in (\Sigma \cup N)^*$ (A and u are strings of terminals and nonterminals but A 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: $T_1 A T_2 \Rightarrow_G T_1 u T_2$ iff $A \to u \in R$ 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 $\mathcal{L}(G) = \{w \in \Sigma^* : S \Rightarrow_G^* w\}$, where $\Rightarrow_G^*$ denotes as usual the reflexive and transitive closure of $\Rightarrow_G$.

Unrestricted grammars generate recursively enumerable languages (RE). Context-sensitive grammars (where each rule $A \to u$ satisfies $|A| \leq |u|$) generate context-sensitive languages (CS). Context-free languages (CF) are generated by context-free grammars, where each rule $A \to u$ satisfies $|A| = 1$. Linear grammars (for linear languages LIN) are context-free grammars in which no rewriting rule is allowed to have more that one non-terminal in its right hand side. A regular grammar has only rules of the form $A \to cB$, $A \to c$, $A \to \varepsilon$, or $A \to B$ where A,B are nonterminals and c is a terminal, and generates regular languages (REG) [10, 12].

A Parallel Communicating Grammar System (or PCGS) consists of a number of grammars that work together and communicate with each other.

Definition 1 Parallel Communicating Grammar System [6]: *A PCGS of degree n, $n \geq 1$ is an $(n+3)$ tuple $\Gamma = (N, K, T, G_1, \ldots, G_n)$ 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 $V_\Gamma = N \cup K \cup T$. $G_i = (N \cup K, T, R_i, S_i), 1 \leq i \leq n$ are Chomsky grammars. The grammars $G_i, 1 \leq i \leq n$, 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 [6]: *Let $\Gamma = (N, K, T, G_1, \cdots, G_n)$ be a PCGS, and $(x_i, x_2, \ldots, x_n)$ and $(y_i, y_2, \ldots, y_n)$ be two n-tuples with $x_i, y_i \in V_\Gamma^*$,*

$1 \leq i \leq n$. *We write* $(x_i, \ldots, x_n) \Rightarrow (y_i, \ldots, y_n)$ *iff one of the following two cases holds:*

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

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

As usual $\Rightarrow^*$ *is 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 ($x_i \Rightarrow_{G_i} y_i$, $1 \leq i \leq n$, 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, iff 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 grammars G_j whose strings were communicated to x_i continue 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).

We use $\Rightarrow$ for both component-wise and communication steps, but we also use (sparingly) $\overset{\Lambda}{\Rightarrow}$ for communication steps whenever we want to emphasize that a communication takes place. A sequence of interleaved rewriting and communication steps will be denoted by $\Rightarrow^*$, the reflexive and transitive closure of $\Rightarrow$.

The derivation in a PCGS can be blocked in two ways [6, 16, 18, 21]:

1. if some component x_i of the current n-tuple $(x_1, \ldots, x_n)$ contains nonterminals but does not contain any 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 [6]: *The language generated by a PCGS* Γ *is* $\mathcal{L}(\Gamma) = \{w \in T^* : (S_1, S_2, ..., S_n) \Rightarrow^* (w, \sigma_2, ..., \sigma_n), \sigma_i \in V_\Gamma^*, 2 \leq i \leq n\}$.

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).

A PCGS is called centralized if only G_1 can introduce query symbols, otherwise it is called non-centralized. 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 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

Numerous results regarding the generative capacity of various kinds of PCGS exist. We focus in this paper on synchronized PCGS with context-free components.

First of all, in it immediate that the centralized variant is a particular case of a non-centralized PCGS and so and so centralized PCGS are weaker than the non-centralized ones. In particular we have $\mathrm{CPC}_*(\mathrm{CF}) \subseteq \mathrm{PC}_*(\mathrm{CF})$ [7]. It is not known whether the inclusion is strict or not.

A way to increase the generative power of a system is to increase the number of components in the system. Increasing the number of components to anything larger than one in the RE and to some degree CS case (the CS result holds for centralized systems only) does not increase the generative power, but when the component grammars are weaker this is no longer the case. Indeed, both the hierarchies $\mathrm{CPC}_n(\mathrm{REG})$ and $\mathrm{CPC}_n(\mathrm{LIN})$, $n \geq 1$ are infinite [6].

Unsurprisingly, the context-free case is somewhere in the middle, in the sense that the hierarchies $\mathrm{PC}_n(\mathrm{CF})$ and $\mathrm{NPC}_n(\mathrm{CF})$, $n \geq 1$ do collapse, though not at $n = 1$. Indeed, non-centralized CF-PCGS with 11 components can apparently generate the whole class of RE languages [4]:

$$\mathrm{RE} = \mathrm{PC}_{11}(\mathrm{CF}) = \mathrm{PC}_*(\mathrm{CF}). \tag{1}$$

We will discuss (and modify) this construction later, so we provide in Figure 1 the rewriting rules for the 11-component context-free PCGS that established the result shown in Equation (1) [4].

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 $\mathcal{L}(P) = L$. [2]:

$$\mathrm{RE} = \mathrm{PC}_5(\mathrm{CF}) = \mathrm{PC}_*(\mathrm{CF}). \tag{2}$$

Other papers have examined the generative capacity of CF-PCGS base don size complexity. It has been shown that every recursively enumerable language can be generated by a returning CF-PCGS, where the number of nonterminals in the system is less than or equal to a natural number k [3]. 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 machine [5].

$$
\begin{aligned}
P_m &= \{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] | (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, \\
&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\}, \\
P_1^{c_1} &= \{S_1 \to Q_m, S_1 \to Q_4^{c_1}, C \to Q_m\} \cup \\
&\{[x, q, c_1, c_2, e_1, e_2] \to [e_1]', [+1]' \to AAC, [0]' \to AC, [-1]' \to C | \\
&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\}, \\
P_2^{c_1} &= \{S_2 \to Q_m, S_2 \to Q_4^{c_1}, C \to Q_m, A \to A\} \cup \\
&\{[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, \\
&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 \\
&\{[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] \\
&[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 \\
&\{[x, q, c_1, c_2, e_1, e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C | \\
&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\} \\
P_2^{c_2} &= \{S_2 \to Q_m, S_2 \to Q_4^{c_2}, C \to Q_m, A \to A\} \cup \\
&\{[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] | x \in \Sigma, q \in E, \\
&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 \\
&\{[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] | 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 >, \\
&< 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, \\
&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)}, \\
&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 machine.

We will show however in Section 4 that the above results regarding returning CF-PCGS [3, 2, 4] use *broadcast communication* which modifies the power of a system when compared to *one-step communication* (which is implied by the original definition). We will also show (Section 5.2) that the hierarchy $PC_*(CF)$ 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 [5, 14]. Given that non-returning systems can be simulated by returning systems with the help of assistance grammars holding intermediate strings [8], these results [5, 14] also apply to returning systems (though the number of components necessary for this to happen does not remain the same).

4 One-step versus broadcast communication

Broadcast and one-step communication were introduced informally in Section 1. The original definition of PCGS derivations (Definition 2) implies the one-step communication model. Indeed, we note that Item 2 of the definition specifies that some (one) component $x_i = z_1 Q_{i_1} z_2 Q_{i_2} \dots z_t Q_{i_t} z_{t+1}$ is chosen and rewritten to $y_i = z_1 x_{i_1} z_2 x_{i_2} \dots z_t x_{i_t} z_{t+1}$, and then the components $i_1, \dots, i_t$ are reduced to their respective axioms (namely, $y_{i_j} = S_{i_j}$, $1 \leq j \leq t$).

Under the one-step communication model a configuration such as (Q_3, Q_3, σ) must perform a communication step in which the third component is communicated to either the first or the second component (chosen nondeterministically). The possible derivations are thus $(Q_3, Q_3, \sigma) \overset{\Lambda}{\Rightarrow} (\sigma, Q_3, S_3)$ or $(Q_3, Q_3, \sigma) \overset{\Lambda}{\Rightarrow} (Q_3, \sigma, S_3)$. In the next communication step the axiom will be communicated to the remaining component instead of the original string σ. In all, we end up nondeterministically with either (σ, S_3, S_3) or (S_3, σ, S_3).

By contrast, in the broadcast communication model the reduction to axiom happens only after all the queries in all the components have been satisfied. A configuration such as (Q_3, Q_3, σ) will have both query symbols satisfied before the third component is reduced to the axiom, so that the following is the only possible derivation starting from this configuration: $(Q_3, Q_3, \sigma) \overset{\Lambda}{\Rightarrow} (\sigma, \sigma, S_3)$.

The following definition introduces the broadcast communication model formally. The definition is an adaptation of the one provided elsewhere [24]; we note that broadcast communication was called immediate communication earlier, though we believe that our terminology conveys the phenomenon better.

Definition 4 Derivation with broadcast communication in a PCGS: *Let* $\Gamma = (N, K, T, G_1, \cdots, G_n)$ *be a returning PCGS, and* $(x_i, x_2, \ldots, x_n)$ *and* $(y_i, y_2, \ldots, y_n)$ *be two n-tuples with* $x_i, y_i \in V_\Gamma^*$, $1 \leq i \leq n$. *We write* $(x_i, \ldots, x_n) \Rightarrow (y_i, \ldots, y_n)$ *iff one of the following three cases holds:*

1. $|x_i|_K = 0, 1 \leq i \leq n$. *Then for each* $i, 1 \leq i \leq n$, *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. *The following set* I *is not empty: with* $J = \emptyset$, I *contains exactly all the indices* i *such that:*
   (a) $x_i = z_1 Q_{i_1} z_2 Q_{i_2} \ldots z_t Q_{i_t} z_{t+1}$ *for some* $t \geq 1$,
   (b) $|x_{i_j}|_K = 0$, $1 \leq j \leq t$,
   (c) *either* $z_j \in (N \cup \Sigma)^*$, *or* $z_j \in (N \cup \Sigma \cup K)^*$ *and for any query symbol* Q_m *appearing in* z_j *we have* $|x_m|_K \neq 0$, *and*
   (d) *let* J *be replaced by* $J \cup \{i_j : 1 \leq j \leq t\}$.

   Then for all $i \in I$ *we have* $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$, *and afterward for all* $j \in J$ *we have* $y_j = S_j$. *For all* $1 \leq k \leq n$, $y_k = x_k$ *whenever* y_k *was not specified above.*

The definition specifies that all the queries that can be satisfied in the current communication step will be satisfied before any reduction to the axiom happens. Note in passing that there might be query symbols that cannot be satisfied because the requested strings contain query symbols themselves; if so, then those queries will not be satisfied in the current step but will be left instead for subsequent communication steps.

Evidently, the communication model (broadcast or one step) 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:

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

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.

By contrast, the above derivation but this time using one-step communication would go like this:

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

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 generated the string $abbb$, whereas the derivation that followed the one-step communication model 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.

In this particular case, we can modify the original system Γ so that it works under the one-step communication model and yet generates the same language as the broadcast communication operation of Γ. The key will be to ensure that communication steps are monogamous, meaning that there are no two components that query a third component at the same time. We accomplish this in this particular case by duplicating the second component, so that the other components have their individual component to query. We end up with the PCGS Γ' with the following sets of rewriting rules for the master and now three slave components (indeed, notice the addition of one component with axiom $S_{1_{copy}}$):

$$\{S \to aS, S \to Q_2, S \to Q_3, S_1 \to b, S_2 \to c, S \to \varepsilon\}$$
$$\{S_1 \to bS_1, S_1 \to Q_3, S_2 \to c\}$$
$$\{S_{1_{copy}} \to bS_{1_{copy}}, S_{1_{copy}} \to Q_3, S_2 \to c\}$$
$$\{S_2 \to cS_2, S_2 \to Q_2, S_{1_{copy}} \to b\}$$

The following is an example of a possible derivation in Γ' that emulates the rewriting steps used in the above broadcast derivation for Γ:

$$(S, S_1, S_{1_{copy}}, S_2) \Rightarrow (aS, bS_1, bS_{1_{copy}}cS_2,) \Rightarrow (aQ_2, bbS_1, bbS_{1_{copy}}, cQ_{S_{1_{copy}}}) \\ \overset{\Lambda}{\Rightarrow} (abbS_1, S_1, S_{1_{copy}}cbbS_{1_{copy}}) \Rightarrow (abbb, bS_1, bS_{1_{copy}}, cbbb)$$

The resulting string `abbb` matches that of the string generated above using broadcast communication.

The above technique of providing "copycat" components is not accidental and is not particular to this example. Indeed, we will use the same technique on a larger scale (and in combination with other modifications) later.

5 Turing completeness of CF-PCGS

We are now ready to discuss the Turing completeness of CF-PCGS. We first note that the previous results on the matter use broadcast communication, which is in contradiction to the original definition [6]. However, we then show that CF-PCGS are still Turing complete under the one-step communication model.

5.1 Broadcast communication and the Turing completeness of CF-PCGS

The existence of two communication models is what causes us to call into question the result shown in Equation (1) [4]. Indeed, the proof that led to this result hinges on the use of broadcast communication, in contrast with the original PCGS definition. This approach to communication was also used in other related papers [3, 2], though we will focus on what was chronologically the first result in this family [4].

A derivation in the system [4] that shows Turing completeness for CF-PCGS (shown in Figure 1) begins with the initial configuration and then takes its first step which results in a nondeterministic choice.

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

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:

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

We have now a communication step, which proceeds as follows [4]:

$$\begin{aligned}&([I], Q_m, Q_m, Q_m, S_4^{(1)}, Q_m, Q_m, Q_m, S_4^{(1)}, Q_m, S^{(3)}) &&\Rightarrow\\ &(S, [I], [I], [I], S_4^{(1)}, [I], [I], [I], S_4^{(1)}, [I], S^{(3)})\end{aligned}$$

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. Should we have used one-step communication the behavior of the system would have been quite different. Some 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 the axiom S since the master will return to the axiom before any of the other components had a chance to query it.

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

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 [4] to hold.

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

5.2 CF-PCGS with one-step communication are Turing complete

We are now to modify the original construction [4] and so 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 5 $\mathrm{RE} = \mathcal{L}(\mathrm{PC}_{95}(\mathrm{CF})) = \mathcal{L}(\mathrm{PC}_*(\mathrm{CF}))$.

The remainder of this section is dedicated to the proof of Theorem 5. Specifically, we show the inclusion $\mathrm{RE} \subseteq \mathcal{L}(\mathrm{PC}_{95}(\mathrm{CF}))$. Customary proof techniques demonstrate that $\mathcal{L}(\mathrm{PC}_*(\mathrm{CF})) \subseteq \mathrm{RE}$ and so $\mathcal{L}(\mathrm{PC}_{95}(\mathrm{CF})) \subseteq \mathcal{L}(\mathrm{PC}_*(\mathrm{CF})) \subseteq \mathrm{RE}$. We describe first informally a PCGS simulating an arbitrary 2-counter machine (Section 5.2.1), then we present that PCGS in detail (Section 5.2.2), and then we then describe how the simulation is carried out (Section 5.2.3).

Let $M = (\Sigma \cup \{Z, B\}, E, R)$ be a 2-counter 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 machine has a read only and unidirectional input tape as well as two read-write counter tapes (or just counters). The counters are initialized with zero by placing the symbol Z on their leftmost cell, while the rest of the cells contain B. A counter can be incremented or decremented by moving the head to the right or to the left, respectively; it thus stores an integer i by having the head moved i positions to the right of the cell containing Z. It is an error condition to move the head to the left of Z. One can test whether the counter holds a zero value or not by inspecting the symbol currently under the head (which 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 status of the two counters (c_1 and c_2, which can be either Z or B). 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 $g = 0$. The input string is accepted by the machine iff the input head scans one cell to the right of the last non-blank symbol on the input tape and the machine is in an accepting state. $\mathcal{L}(M)$ be the language of exactly all the input strings accepted by M.

5.2.1 CF-PCGS simulation of a 2-counter machine: overall structure

We demonstrated in Section 4 (through the modification of the PCGS Γ to obtain Γ') the "copycat" technique of duplicating a components to ensure that all communication steps are monogamous. In a nutshell, we will apply this technique on the CF-PCGS developed earlier [4].

We still provide a CF-PCGS that simulates an arbitrary 2-counter machine. We use all of the components used originally, but we ensure that every grammar that requests a string from the same component in the original system

can retrieve a similar string from it own exclusive copycat component. In other words, our system includes copycat components (or helpers) which ensure that all the components can work under one-step communication without stumbling over each other. For the most part the intermediate strings that the copycat components 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 under one-step communication.

We also need to add components to the system whose job is to fix synchronization issues by resetting their matching helpers at specific points in the derivation. This ensures that the one-step communication version of the system remains in harmony with the broadcast communication system. 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.

Concretely, we now construct the following grammar system with 95 components:

$$
\begin{aligned}
\Gamma \;=\; & (N, K, \Sigma \cup \{a\},\\
& G_{GM_{Orig}}, G^{c_1}_{GM_{S1}}, G^{c_1}_{GM_{S1H2(S4)}}, G^{c_1}_{GM_{S1H3(S4)}}, G^{c_1}_{GM_{S1(S2)}}, G^{c_1}_{GM_{S1(S3)}},\\
& G^{c_1}_{GM_{S2}}, G^{c_1}_{GM_{S3}}, G^{c_1}_{GM_{PA1S1}}, G^{c_1}_{GM_{PA1S1H2}}, G^{c_1}_{GM_{PA1S1H3}}, G^{c_1}_{GM_{PA1S1(S2)}},\\
& G^{c_1}_{GM_{PA1S1(S3)}}, G^{c_1}_{GM_{PA1S2}}, G^{c_1}_{GM_{PA1S3}}, G^{c_2}_{GM_{S1}}, G^{c_2}_{GM_{S1H2(S4)}}, G^{c_2}_{GM_{S1H3(S4)}},\\
& G^{c_2}_{GM_{S1(S2)}}, G^{c_2}_{GM_{S1(S3)}}, G^{c_2}_{GM_{S2}}, G^{c_2}_{GM_{S3}}, G^{c_2}_{GM_{PA1S1}}, G^{c_2}_{GM_{PA1S1H2}}, G^{c_2}_{GM_{PA1S1H3}},\\
& G^{c_2}_{GM_{PA1S1S2}}, G^{c_2}_{GM_{PA1S1S3}}, G^{c_2}_{GM_{PA1S2}}, G^{c_2}_{GM_{PA1S3}}, G^{c_1}_{1_{OrigS_1}}, G^{c_1}_{1_{S_1H_2(S_4)}},\\
& G^{c_1}_{1_{S_1H_3(S_4)}}, G^{c_1}_{1_{S_1(S_2)}}, G^{c_1}_{1_{S_1(S_3)}}, G^{c_1}_{2_{OrigS_2}}, G^{c_1}_{3_{OrigS3}}, G^{c_1}_{4_{OrigS_4}},\\
& G^{c_1}_{4_{S_1H_2(S_4)}}, G^{c_1}_{4_{S_1H_3(S_4)}}, G^{c_1}_{4_{S_2}}, G^{c_1}_{4_{S_3}}, G^{c_1}_{4SpecHelp1_{S1S2}}, G^{c_1}_{4SpecHelp2_{S1S3}},\\
& G^{c_2}_{1_{OrigS_1}}, G^{c_2}_{1_{S_1H_2(S_4)}}, G^{c_2}_{1_{S_1H_3(S_4)}}, G^{c_2}_{1_{S_1(S_2)}}, G^{c_2}_{1_{S_1(S_3)}}, G^{c_2}_{2_{OrigS_2}}, G^{c_2}_{3_{OrigS3}},\\
& G^{c_2}_{4_{OrigS4}}, G^{c_2}_{4_{S_1H_2(S_4)}}, G^{c_2}_{4_{S_1H_3(S_4)}}, G^{c_2}_{4_{S_2}}, G^{c_2}_{4_{S_3}}, G^{c_2}_{4SpecHelp1_{S1S2}}, G^{c_2}_{4SpecHelp2_{S1S3}},\\
& G_{a_1Orig}, G^{c_1}_{a_1GMS_1}, G^{c_1}_{a_1GMS_1H_2(S_4)}, G^{c_1}_{a_1GMS_1H_3(S_4)}, G^{c_1}_{a_1GMS_1(S_2)},\\
& G^{c_1}_{a_1GMS_1(S_3)}, G^{c_1}_{a_1GMS_2}, G^{c_1}_{a_1GMS_3}, G^{c_2}_{a_1GMS_1}, G^{c_2}_{a_1GMS_1H_2(S_4)}, G^{c_2}_{a_1GMS_1H_3(S_4)},\\
& G^{c_2}_{a_1GMS_1(S_2)}, G^{c_2}_{a_1GMS_1(S_3)}, G^{c_2}_{a_1GMS_2}, G^{c_2}_{a_1GMS_3}, G_{a_2Orig}, R_{GM^{c_1}_{Pa1_{S1}}},\\
& R_{GM^{c_1}_{Pa1_{S1H2(S4)}}}, R_{GM^{c_1}_{Pa1_{S1H3(S4)}}}, R_{GM^{c_1}_{Pa1_{S1(S2)}}}, R_{GM^{c_1}_{Pa1_{S1(S3)}}}, R_{GM^{c_1}_{Pa1_{S2}}},
\end{aligned}
$$

$$R_{GM^{c_1}_{Pa1_{S3}}}, R_{GM^{c_2}_{Pa1_{S1}}}, R_{GM^{c_2}_{Pa1_{S1H2(S4)}}}, R_{GM^{c_2}_{Pa1_{S1H3(S4)}}}, R_{GM^{c_2}_{Pa1_{S1(S2)}}},$$
$$R_{GM^{c_2}_{Pa1_{S1(S3)}}}, R_{GM^{c_2}_{Pa1_{S2}}}, R_{GM^{c_2}_{Pa1_{S3}}}, R_{P^{c_1}_{1_{S_1H_2(S_4)}}}, R_{P^{c_1}_{1_{S_1H_3(S_4)}}}, R_{P^{c_2}_{1_{S_1H_2(S_4)}}},$$
$$R_{P^{c_2}_{1_{S_1H_3(S_4)}}}, R_{P^{c_1}_{4_{S_1H_2(S_4)}}}, R_{P^{c_1}_{4_{S_1H_3(S_4)}}}, R_{P^{c_2}_{4_{S_1H_2(S_4)}}}, R_{P^{c_2}_{4_{S_1H_3(S_4)}}})$$

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

The rewriting rules will be formally defined later. As already mentioned, this system is a simulation of the construction using broadcast communication and described in Figure 1. All of the component definitions from the original system have the marker *Orig* in their label in order to differentiate them from the helper grammars that were added in order to accommodate the requirements of a one step-communication system. In what follows we use x and y as wildcards, which can be replaced by any string. For example the grammars $G^{c_1}_{1_{Orig S_1}}$, $G^{c_1}_{1_{S_1H_2(S_4)}}$, $G^{c_1}_{1_{S_1H_3(S_4)}}$, $G^{c_1}_{1_{S_1}(S_2)}$, and $G^{c_1}_{1_{S_1}(S_3)}$ will be referred to collectively as $G^{c_1}_{1_x}$.

The above system Γ uses the following labeling: The grammars G^y_x with the set of rewriting rules P^y_x are the "major" components, as opposed to the grammars R_x (with $Reset_x$ as the set of rewriting rules) which are reset grammars (used to reset to axiom various components throughout the derivation). When we refer to a component grammar we will use interchangeably its name or the name of its set of rewriting rules. $G_{GM_{Orig}}$ is the master grammar of the system, while G_{GM_x} are copycat grammars that replicate the steps of the master. $G^{c_1}_x$ and $G^{c_2}_x$ indicates that the grammar works with counter c_1 or c_2, respectively. $G^y_{1_x}$ indicate that the grammar is either the original or a replica of grammar P_1 in the original system. $G_{2_{Orig S_2}}$ and $G_{3_{Orig S_3}}$ indicate that the grammar is the original P_2 and P_3, respectively. $G^y_{4_x}$ and $G^y_{a_{1x}}$ indicate that the grammar is either the original or a replica of grammar P_4 and P_{a_1} in the original system, respectively, and so on.

It is no accident that the sub- and superscripts described above suggest a grouping of most of the 95 components in classes that correspond to the original components. Creating copycat grammars is the most basic tool used

in our construction, so it is inevitable to have several grammars playing a similar role and being all roughly equivalent to one original component.

The new master $G_{GM_{Orig}}$ contains the same rewriting rules and communications steps as it had in the original construction [4]. The primary role of the master is to maintain its relationship with the $G^{y}_{a_1x}$ component grammars. The other components $G^{y}_{GM_x}$ are copycat grammars designed to copy the functionality of the master; they have been added to the system to handle queries from $G^{c_1}_{1_x}$, $G^{c_1}_{2_x}$, $G^{c_1}_{3_x}$, $G^{c_1}_{4_x}$, $G^{c_2}_{1_x}$, $G^{c_2}_{2_x}$, $G^{c_2}_{3_x}$, and $G^{c_2}_{4_x}$ (all described later). In essence we ensure that every component grammar $G^{c_1}_{1_x}, \ldots, G^{c_2}_{4_x}$ that can query the master grammar in the original broadcast construction has a matching helper grammar that will exclusively handle their communication requests.

$G^{c_1}_{1_{OrigS_1}}$ contains the same rewriting rules and communication steps as the component $P^{c_1}_1$ in the original system [4], though 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. There are 4 new helper grammars to ensure that $G^{c_1}_2$, $G^{c_1}_3$, and $G^{c_1}_4$ have their own unique component grammars to communicate with.

Component $G^{c_1}_{4_{OrigS4}}$ (equivalent to the original $P^{c_1}_4$) needs extra helper grammars to ensure that components defined in other sections have their own unique $G^{c_1}_{4_x}$ component to query. Similarly, $G^{c_2}_{1_{OrigS_1}}$ is similar with the original $P^{c_2}_1$ and needs 4 new helper grammars; $G^{c_2}_{4_{OrigS4}}$ is equivalent to $P^{c_2}_4$ and requires 6 additional helpers.

The original P_{a_1} grammar remains as it was in the original system and is now named P_{a_1Orig}. In order for component grammars $G^{c_1}_{1_x}$, $G^{c_1}_{2_x}$, $G^{c_1}_{3_x}$, $G^{c_1}_{4_x}$, $G^{c_2}_{1_x}$, $G^{c_2}_{2_x}$, $G^{c_2}_{3_x}$, and $G^{c_2}_{4_x}$ to derive correctly 14 additional G_{a_1x} helpers have been added to the system. Their names and labels reflect the components they will work with during a derivation.

Finally, grammars $G^{c_1}_{2_x}$, $G^{c_1}_{3_x}$, $G^{c_2}_{2_x}$, $P^{c_2}_{3_x}$, and $P_{a_{2x}}$ are similar to the original $P^{c_1}_2$, $P^{c_1}_3$, $P^{c_2}_2$, $P^{c_2}_3$, and P_{a_2}, respectively without any additional helper grammars (but with the usual label changes) and serve the same role as in the original construction.

This all being said and done, the derivation in the new system is obviously more complex than in the original. Several undesired side derivations become possible and need to be eliminated. One mechanism used for this purpose is the reset grammars R_x which are used to reset several major components at strategic moments. Which reset grammar handles which major component is given by the subscript x. Their use is illustrated in Section 5.2.3. Another mechanism for eliminating undesired derivations is the existence of several

additional rules that did not exist in the original system and that cause various derivations to block. These extra rules are described in Section 5.2.2.

5.2.2 CF-PCGS simulation of a 2-counter machine: rewriting rules

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, as outlined in the previous section, but we also add new rules to some components. To emphasize these additions we group the newly introduced rules into separate sets that are underlined. In most cases the 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 an underlined set.

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

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 $P_{1_{Orig S_1}}^{c_1}$ 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.

$$
\begin{aligned}
P_{GM_{S1}}^{c_1} \;=\; & \{S \to [I], [I] \to C\} \cup \underline{\{C \to Q_{a_1 P_{a_1} S_1}^{c_1}\}} \cup \\
& \{< I > \to [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup
\end{aligned}
$$

$$\{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R,$$
$$x, y \in \Sigma\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] |$$
$$(x, q, c_1, c_2, q', e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\},$$
$$e_1', e_2' \in \{-1, 0, +1\}\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$$
$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q',$$
$$e_1, e_2, +1) \in R, c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

The following two grammars have new communication steps $S \rightarrow Q^{C_1}_{a_1 P_{a_1} S_1 H_2(S_4)}$ and $S \rightarrow Q^{C_1}_{a_1 P_{a_1} S_1 H_3(S_4)}$, 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.

$$P^{c_1}_{GM_{S1H2(S4)}} = \{S \rightarrow [I], [I] \rightarrow C\} \cup$$
$$\underline{\{C \rightarrow Q^{c_1}_{a_1 P_{a_1} S_1 H_2(S_4)}, S \rightarrow Q^{c_1}_{a_1 P_{a_1} S_1 H_2(S_4)}\} \cup}$$
$$\{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup$$
$$\{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R,$$
$$x, y \in \Sigma\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] | x \in \Sigma, c_1', c_2' \in$$
$$\{Z, B\}, (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, e_1', e_2' \in \{-1, 0, +1\}\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$$
$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

$$P^{c_1}_{GM_{S1H3(S4)}} = \{S \rightarrow [I], [I] \rightarrow C\} \cup$$
$$\underline{\{C \rightarrow Q^{c_1}_{a_1 P_{a_1} S_1 H_3(S_4)}, S \rightarrow Q^{c_1}_{a_1 P_{a_1} S_1 H_3(S_4)}\} \cup}$$

$$
\begin{aligned}
&\{< I >\rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
&\{< I >\rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, \\
&\quad x, y \in \Sigma\} \cup \\
&\quad \{< x, q, c_1', c_2', e_1', e_2' >\rightarrow [x, q', c_1, c_2, e_1, e_2] | \\
&\quad (x, q, c_1, c_2, q', e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\}, \\
&\quad e_1', e_2' \in \{-1, 0, +1\}\} \cup \\
&\{< x, q, c_1', c_2', e_1', e_2' >\rightarrow x[y, q', c_1, c_2, e_1, e_2], \\
&\quad < x, q_F, c_1', c_2', e_1', e_2' >\rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, \\
&\quad c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}
$$

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

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

We only need one $P^{c_1}_2$ component and one $P^{c_1}_3$ component. These will simulate rules from the master grammar and will work directly with $P^{c_1}_{2_{OrigS2}}$ and

$P^{c_1}_{3_{OrigS3}}$, respectively. The labels in the communication rules have been modified to ensure that the correct component grammars are queried.

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

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

The following 7 helper grammars imitate P_{a_1}. The first 5 work with $P^{c_1}_{1_{Orig}}$ and four of its helpers, while the remaining 2 work with $P^{c_1}_{2_{Orig}}$ and $P^{c_1}_{3_{Orig}}$. The new rule allows the grammars to reset their string by querying new helper components (defined later).

$$
\begin{aligned}
P^{c_1}_{GM_{PA1S1}} \;=\; & \{S \to [I], [I] \to C\} \cup \underline{\{C \to Q_{Reset_{GM^{c_1}_{Pa1S1}}}\}} \cup \\
& \{< I > \to [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \to x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \\
& \quad \Sigma\} \cup \quad \{< x, q, c_1', c_2', e_1', e_2' > \to [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, \\
& \quad q', e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}\} \cup
\end{aligned}
$$

$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$$
$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

$$P^{c_1}_{GM_{PA1S1H2}} = \{S \rightarrow [I], [I] \rightarrow C\} \cup \{C \rightarrow Q_{\overline{Reset_{GM^{c_1}_{Pa1}}}_{S1H2(S4)}}}\} \cup$$
$$\{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup$$
$$\{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x,$$
$$y \in \Sigma\} \cup \quad \{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] |$$
$$(x, q, c_1, c_2, q', e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\},$$
$$e_1', e_2' \in \{-1, 0, +1\}\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$$
$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

$$P^{c_1}_{GM_{PA1S1H3}} = \{S \rightarrow [I], [I] \rightarrow C\} \cup \{C \rightarrow Q_{\overline{Reset_{GM^{c_1}_{Pa1}}}_{S1H3(S4)}}}\} \cup$$
$$\{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup$$
$$\{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R,$$
$$x, y \in \Sigma\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] |$$
$$(x, q, c_1, c_2, q', e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\},$$
$$e_1', e_2' \in \{-1, 0, +1\}\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$$
$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

$$P^{c_1}_{GM_{PA1S1(S2)}} = \{S \rightarrow [I], [I] \rightarrow C\} \cup \{C \rightarrow Q_{\overline{Reset_{GM^{c_1}_{Pa1}}}_{S1(S2)}}}\} \cup$$
$$\{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup$$
$$\{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R,$$
$$x, y \in \Sigma\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q',$$

$e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}\} \cup$
$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$
$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$
$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$

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

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

$$P^{c_1}_{GM_{PA1S3}} = \{S \rightarrow [I], [I] \rightarrow C\} \cup \{C \rightarrow Q_{Reset_{GM^{c_1}_{Pa1_{S3}}}}\} \cup$$
$\{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup$
$\{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in$
$\Sigma\} \cup \quad \{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2,$
$q', e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}\} \cup$
$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$

$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

The following 5 helpers simulate rules from the new master. Each grammar below is designed to work with a different component in the $P_1^{c_2}$ family, including $P_{1_{OrigS_1}}^{c_2}$ and its 4 helpers. The first works directly with $P_{1_{Orig}}^{c_2}$ as it did originally, but communication labels have been modified to ensure that each component queries the right grammar.

$$\begin{aligned}
P_{GM_{S1}}^{c_2} = {} & \{S \rightarrow [I], [I] \rightarrow C\} \cup \underline{\{C \rightarrow Q_{a_1 P_{a_1} S_1}^{c_2}\}} \cup \\
& \{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, x, y \in \Sigma\} \cup \\
& \{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', e_1, e_2, 0) \\
& \quad \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2], \\
& \quad < x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R, \\
& \quad c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}
\end{aligned}$$

Note that the following two grammars have a new communication step $S \rightarrow Q_{a_1 P_{a_1} S_1 H_2(S_4)}^{c_2}$ and $S \rightarrow Q_{a_1 P_{a_1} S_1 H_3(S_4)}^{c_2}$ 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 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 $P_{GM_{S1H2(S4)}}^{c_2}$ or $P_{GM_{S1H3(S4)}}^{c_2}$ component will receive the string $u[x', q, Z, Z, e_1, e_2]$, for which it has no rewriting rule.

$$\begin{aligned}
P_{GM_{S1H2(S4)}}^{c_2} = {} & \{S \rightarrow [I], [I] \rightarrow C\} \cup \\
& \underline{\{C \rightarrow Q_{a_1 P_{a_1} S_1 H_2(S_4)}^{c_2}, S \rightarrow Q_{a_1 P_{a_1} S_1 H_2(S_4)}^{c_2}\}} \cup \\
& \{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup \\
& \{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R, \\
& \quad x, y \in \Sigma\} \cup \\
& \{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q', \\
& \quad e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}\} \cup \\
& \{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],
\end{aligned}$$

$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

$$P^{c_2}_{GM_{S1H3(S4)}} = \{S \rightarrow [I], [I] \rightarrow C\} \cup$$
$$\underline{\{C \rightarrow Q^{c_2}_{a_1 P_{a_1} S_1 H_3(S_4)}, S \rightarrow Q^{c_2}_{a_1 P_{a_1} S_1 H_3(S_4)}\}} \cup$$
$$\{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup$$
$$\{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R,$$
$$x, y \in \Sigma\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q',$$
$$e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$$
$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

$$P^{c_2}_{GM_{S1(S2)}} = \{S \rightarrow [I], [I] \rightarrow C\} \cup \underline{\{C \rightarrow Q^{c_2}_{a_1 P_{a_1} S_1(S_2)}\}} \cup$$
$$\{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup$$
$$\{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R,$$
$$x, y \in \Sigma\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q',$$
$$e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$$
$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

$$P^{c_2}_{GM_{S1(S3)}} = \{S \rightarrow [I], [I] \rightarrow C\} \cup \underline{\{C \rightarrow Q^{c_2}_{a_1 P_{a_1} S_1(S_2)}\}} \cup$$
$$\{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup$$
$$\{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R,$$
$$x, y \in \Sigma\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q',$$
$$e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}\} \cup$$

$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$$
$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

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

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

Similarly, there is only one $P_3^{c_2}$, as in the original system and the below master helper will work directly 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.

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

The following 7 grammars work with the $P_{a_1}^{c_2}$ components; the first 5 work with the $P_1^{c_2}$ helper grammars, and the other 2 work with $P_{2_{OrigS_2}}^{c_2}$ and $P_{3_{OrigS_3}}^{c_2}$ 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).

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

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

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

$$< x, q_F, c_1', c_2', e_1', e_2' > \rightarrow x | (x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

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

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

$$P^{c_2}_{GM_{PA1S2}} = \{S \rightarrow [I], [I] \rightarrow C\} \cup \{C \rightarrow Q_{Reset_{GM^{c_2}_{Pa1_{S2}}}}\} \cup$$
$$\{< I > \rightarrow [x, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, e_1, e_2, 0) \in R, x \in \Sigma\} \cup$$
$$\{< I > \rightarrow x[y, q, Z, Z, e_1, e_2] | (x, q_0, Z, Z, q, e_1, e_2, +1) \in R,$$
$$x, y \in \Sigma\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow [x, q', c_1, c_2, e_1, e_2] | (x, q, c_1, c_2, q',$$
$$e_1, e_2, 0) \in R, x \in \Sigma, c_1', c_2' \in \{Z, B\}, e_1', e_2' \in \{-1, 0, +1\}\} \cup$$
$$\{< x, q, c_1', c_2', e_1', e_2' > \rightarrow x[y, q', c_1, c_2, e_1, e_2],$$

$$< x, q_F, c_1', c_2', e_1', e_2' >\rightarrow x|(x, q, c_1, c_2, q', e_1, e_2, +1) \in R,$$
$$c_1', c_2' \in \{Z, B\}, \quad e_1', e_2' \in \{-1, 0, +1\}, x, y \in \Sigma\}$$

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

$P^{c_1}_{1_{OrigS_1}}$ contains the same rewriting rules and communication steps as the component $P^{c_1}_1$ in the original system [4] with suitable modifications for labels.

$$\begin{aligned} P^{c_1}_{1_{OrigS_1}} = {} & \{S_1 \rightarrow Q^{c_1}_{GM_{S_1}}, S_1 \rightarrow Q^{c_1}_{4_{S_{1original}}}, C \rightarrow Q^{c_1}_{GM_{S_1}}\} \cup \\ & \{[x, q, c_1, c_2, e_1, e_2] \rightarrow [e_1]', [+1]' \rightarrow AAC, [0]' \rightarrow AC, \\ & \quad [-1]' \rightarrow C|x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \cup \\ & \{[I] \rightarrow [I]', [I]' \rightarrow AC\} \end{aligned}$$

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 $C \rightarrow W$, 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).

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

$$P^{c_1}_{1_{S_1H_3(S_4)}} = \{S_1 \rightarrow Q^{c_1}_{GM_{S_1H_3(S_4)}}, S_1 \rightarrow Q^{c_1}_{4S_1H_3(S_4)}, C \rightarrow Q_{GM_{S_1H_3(S_4)}},$$

$$\underline{C \to W\} \cup}$$
$$\{[x, q, c_1, c_2, e_1, e_2] \to [e_1]', [+1]' \to AAC, [0]' \to AC,$$
$$[-1]' \to C | x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\} \cup$$
$$\{[I] \to [I]', [I]' \to AC\}$$

The following two $P_1^{c_1}$ helpers will ensure the proper derivation of $P_{2_{OrigS_2}}^{c_1}$ and $P_{3_{OrigS_3}}^{c_1}$. They work by communicating with their corresponding helper grammars and their designated special helper in the $P_4^{c_1}$ section.

$$P_{1_{S_1}(S_2)}^{c_1} = \underline{\{S_1 \to Q_{GM_{S_1(S_2)}}^{c_1}, S_1 \to Q_{4SpecHelp1_{S1S2}}^{c_1}, C \to Q_{GM_{S_1(S_2)}},}$$
$$\underline{S_4 \to S_4^{(1)}, S_4^{(1)} \to Q_{P_{1_{S_1H_2(S_4)}}^{c_1}}\} \cup}$$
$$\{[x, q, c_1, c_2, e_1, e_2] \to [e_1]', [+1]' \to AAC, [0]' \to AC, [-1]' \to C |$$
$$x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \cup$$
$$\{[I] \to [I]', [I]' \to AC\}$$

$$P_{1_{S_1}(S_3)}^{c_1} = \underline{\{S_1 \to Q_{GM_{S_1(S_3)}}^{c_1}, S_1 \to Q_{4SpecHelp2_{S1S3}}^{c_1}, C \to Q_{GM_{S_1(S_3)}}^{c_1},}$$
$$\underline{S_4 \to S_4^{(1)}, S_4^{(1)} \to Q_{P_{1_{S_1H_3(S_4)}}^{c_1}}\} \cup}$$
$$\{[x, q, c_1, c_2, e_1, e_2] \to [e_1]', [+1]' \to AAC, [0]' \to AC, [-1]' \to C |$$
$$x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \cup$$
$$\{[I] \to [I]', [I]' \to AC\}$$

Component grammar $P_2^{c_1}$ has been renamed and labels have been modified to ensure that it works with its matching helper components, but is otherwise unchanged from the original definition.

$$P_{2_{OrigS_2}}^{c_1} = \underline{\{S_2 \to Q_{GMS_2}^{c_1}, S_2 \to Q_{4S_2}^{c_1}, C \to Q_{GMS_2}^{c_1}, A \to A\} \cup}$$
$$\{[x, q, Z, c_2, e_1, e_2] \to [x, q, Z, c_2, e_1, e_2], [I] \to [I] | x \in \Sigma, q \in E,$$
$$c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\}$$

Component grammar $P_3^{c_1}$ is again similar to the original definition (with suitable label changes) and it does not need any helper grammars.

$$P_{3_{OrigS3}}^{c_1} = \underline{\{S_3 \to Q_{GMS_3}^{c_1}, S_3 \to Q_{4S_3}^{c_1}, C \to Q_{GMS_3}^{c_1}\} \cup}$$

$$\{[x, q, Z, c_2, e_1, e_2] \to a, [x, q, B, c_2, e_1, e_2] \to [x, q, B, c_2, e_1, e_2]$$
$$[I] \to [I] | x \in \Sigma, q \in E, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\}$$

Component $P^{c_1}_{4_{OrigS4}}$ has the same rules as in the original system save for the usual re-labeling.

$$P^{c_1}_{4_{OrigS_4}} = \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \underline{\{S_4^{(2)} \to Q^{c_1}_{P_1 S_1}\}} \cup \{A \to a\}$$

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.

$$P^{c_1}_{4_{S_1 H_2(S_4)}} = \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \underline{\{S_4^{(2)} \to Q^{c_1}_{P_1 S_1 H_2(S_4)}, S_4^{(2)} \to S_4^{(2)}\}} \cup \{A \to a\}$$

$$P^{c_1}_{4_{S_1 H_3(S_4)}} = \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \underline{\{S_4^{(2)} \to Q^{c_1}_{P_1 S_1 H_3(S_4)}, S_4^{(2)} \to S_4^{(2)}\}} \cup \{A \to a\}$$

$$P^{c_1}_{4_{S_2}} = \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \underline{\{S_4^{(2)} \to Q^{c_1}_{P_1 S_2}\}} \cup \{A \to a\}$$

$$P^{c_1}_{4_{S_3}} = \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \underline{\{S_4^{(2)} \to Q^{c_1}_{P_1 S_3}\}} \cup \{A \to a\}$$

$$P^{c_1}_{4SpecHelp1_{S1S2}} = P^{c_1}_{4SpecHelp2_{S1S3}} = \underline{\{S_4 \to S_4\}}$$

$P^{c_2}_{1_{OrigS_1}}$ is similar to the original $P_1^{c_2}$. It also need 4 new helper grammars.

$$P^{c_2}_{1_{OrigS_1}} = \underline{\{S_1 \to Q^{c_2}_{GMS_1}, S_1 \to Q^{c_2}_{P_4 S_1}, C \to Q^{c_2}_{GMS_1}\}} \cup$$
$$\{[x, q, c_1, c_2, e_1, e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C |$$
$$x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \cup$$
$$\{[I] \to [I]', [I]' \to AC\}$$

The following two $P_1^{c_2}$ helpers 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.

$$P^{c_2}_{1_{S_1 H_2(S_4)}} = \underline{\{S_1 \to Q^{c_2}_{GM_{S_1 H_2(S_4)}}, S_1 \to Q^{c_2}_{P_4 S_1 H_2(S_4)}, C \to Q^{c_2}_{GM_{S_1 H_2(S_4)}},}$$

$$\underline{C \to W\}} \cup$$
$$\{[x, q, c_1, c_2, e_1, e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C |$$
$$x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \cup \{[I] \to [I]',$$
$$[I]' \to AC\}$$

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

The following two $P^{c_2}_1$ helper grammars are components that will help ensure the proper derivation of $P^{c_2}_{2_{OrigS_2}}$ and $P^{c_2}_{3_{OrigS_3}}$ by holding intermediate strings throughout the derivation.

$$P^{c_2}_{1_{S_1(S_2)}} = \underline{\{S_1 \to Q^{c_2}_{GM_{S_1(S_2)}}, S_1 \to Q^{c_2}_{4SpecHelp1_{S1S2}}, C \to Q^{c_2}_{GM_{S_1(S_2)}},}$$
$$\underline{S_4 \to S_4^{(1)}, S_4^{(1)} \to Q_{P^{c_2}_{1_{S_1H_2(S_4)}}}\}} \cup$$
$$\{[x, q, c_1, c_2, e_1, e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C |$$
$$x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \cup$$
$$\{[I] \to [I]', [I]' \to AC\}$$

$$P^{c_2}_{1_{S_1(S_3)}} = \underline{\{S_1 \to Q^{c_2}_{GM_{S_1(S_3)}}, S_1 \to Q^{c_2}_{4SpecHelp2_{S1S3}}, C \to Q^{c_2}_{GM_{S_1(S_3)}},}$$
$$\underline{S_4 \to S_4^{(1)}, S_4^{(1)} \to Q_{P^{c_2}_{1_{S_1H_3(S_4)}}}\}} \cup$$
$$\{[x, q, c_1, c_2, e_1, e_2] \to [e_2]', [+1]' \to AAC, [0] \to AC, [-1] \to C |$$
$$x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \cup$$
$$\{[I] \to [I]', [I]' \to AC\}$$

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.

$$P^{c_2}_{2_{OrigS_2}} = \underline{\{S_2 \to Q^{c_2}_{GMS_2}, S_2 \to Q^{c_2}_{P_4S_2}, C \to Q^{c_2}_{GMS_2}\}} \cup \{A \to A\} \cup$$
$$\{[x, q, c_1, Z, e_1, e_2] \to a, [x, q, c_1, B, e_1, e_2] \to [x, q, c_1, B, e_1, e_2],$$

$$[I] \to [I] | \quad x \in \Sigma, q \in E, c1 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\}$$

Component grammar $P_3^{c_2}$ contains similar rules with the original construction. Similarly to $P_{2_{OrigS_2}}^{c_2}$ it does not require any helper grammars, but the labels have been changed as before.

$$\begin{aligned} P_{3_{OrigS3}}^{c_2} \quad = \quad & \underline{\{S_3 \to Q_{GMS_3}^{c_2}, S_3 \to Q_{P_4S_2}^{c_2}, C \to Q_{GMS_3}^{c_2}\}} \cup \\ & \{[x, q, c_1, Z, e_1, e_2] \to a, [x, q, c_1, B, e_1, e_2] \to [x, q, c_1, B, e_1, e_2] \\ & [I] \to [I] | x \in \Sigma, q \in E, c1 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \end{aligned}$$

Component $P_{4_{OrigS4}}^{c_2}$, 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.

$$P_{4_{OrigS4}}^{c_2} \quad = \quad \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \underline{\{S_4^{(2)} \to Q_{P_1S_1}^{c_2}\}} \cup \{A \to a\}$$

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 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.

$$\begin{aligned} P_{4_{S_1H_2(S_4)}}^{c_2} \quad = \quad & \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \\ & \underline{\{S_4^{(2)} \to Q_{P_1S_1H_2(S_4)}^{c_2}, S_4^{(2)} \to S_4^{(2)}\}} \cup \{A \to a\} \\ P_{4_{S_1H_3(S_4)}}^{c_2} \quad = \quad & \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \\ & \underline{\{S_4^{(2)} \to Q_{P_1S_1H_3(S_4)}^{c_2}, S_4^{(2)} \to S_4^{(2)}\}} \cup \{A \to a\} \\ P_{4_{S_2}}^{c_2} \quad = \quad & \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \underline{\{S_4^{(2)} \to Q_{P_1S_2}^{c_2}\}} \cup \{A \to a\} \\ P_{4_{S_3}}^{c_2} \quad = \quad & \{S_4 \to S_4^{(1)}, S_4^{(1)} \to S_4^{(2)}\} \cup \underline{\{S_4^{(2)} \to Q_{P_1S_3}^{c_2}\}} \cup \{A \to a\} \end{aligned}$$

$$P_{4SpecHelp1_{S1S2}}^{c_2} = P_{4SpecHelp2_{S1S3}}^{c_2} \quad = \quad \underline{\{S_4 \to S_4\}}$$

The original P_{a_1} grammar remains as it was in the original system and needs 14 additional helpers.

$$
\begin{aligned}
P_{a_1 Orig} \;=\; & \underline{\{S \to Q_{GM_{Orig}}\}} \cup \\
& \{[I] \to < I >,\quad [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\
& \quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_1}_{a_1 GMS_1} \;=\; & \underline{\{S \to Q^{c_1}_{GMPA1S_1}, C \to C\}} \cup \\
& \{[I] \to < I >, [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\
& \quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to < I > \mid \\
& \quad x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_1}_{a_1 GMS_1 H_2(S_4)} \;=\; & \underline{\{S \to Q^{c_1}_{GMPA1S_1 H_2(S_4)}, C \to C\}} \cup \\
& \{[I] \to < I >, [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\
& \quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to \\
& \quad < I > \mid x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_1}_{a_1 GMS_1 H_3(S_4)} \;=\; & \underline{\{S \to Q^{c_1}_{GMPA1S_1 H_3(S_4)}, C \to C\}} \cup \\
& \{[I] \to < I >, [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\
& \quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to \\
& \quad < I > \mid x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_1}_{a_1 GMS_1 (S_2)} \;=\; & \underline{\{S \to Q^{c_1}_{GMS_1 (S_2)}, C \to C\}} \cup \\
& \{[I] \to < I >, [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\
& \quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to \\
& \quad < I > \mid x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\}
\end{aligned}
$$

$$
\begin{aligned}
P^{c_1}_{a_1 GMS_1 (S_3)} \;=\; & \underline{\{S \to Q^{c_1}_{GMS_1 (S_3)}, C \to C\}} \cup \\
& \{[I] \to < I >, [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\
& \quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to
\end{aligned}
$$

$$< I > | x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\}$$

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

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

$$\begin{aligned} P^{c_2}_{a_1 GMS_1} &= \underline{\{S \to Q^{c_2}_{GMPA1S_1}, C \to C\}} \cup \\ &\{[I] \to < I >, [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\ &\quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to \\ &\quad < I > | x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \end{aligned}$$

$$\begin{aligned} P^{c_2}_{a_1 GMS_1 H_2(S_4)} &= \underline{\{S \to Q^{c_2}_{GMPA1S_1 H_2(S_4)}, C \to C\}} \cup \\ &\{[I] \to < I >, [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\ &\quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to \\ &\quad < I > | x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \end{aligned}$$

$$\begin{aligned} P^{c_2}_{a_1 GMS_1 H_3(S_4)} &= \underline{\{S \to Q^{c_2}_{GMPA1S_1 H_3(S_4)}, C \to C\}} \cup \\ &\{[I] \to < I >, [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\ &\quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to \\ &\quad < I > | x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \end{aligned}$$

$$\begin{aligned} P^{c_2}_{a_1 GMS_1(S_2)} &= \underline{\{S \to Q^{c_2}_{GMS_1(S_2)}, C \to C\}} \cup \\ &\{[I] \to < I >, [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\ &\quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to \end{aligned}$$

$$< I > | x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\}$$

$$\begin{aligned} P^{c_2}_{a_1 GMS_1(S_3)} &= \underline{\{S \to Q^{c_2}_{GMS_1(S_3)}, C \to C\}} \cup \\ &\{[I] \to < I >, [x, q, c_1, c_2, e_1, e_2] \to < x, q, c_1, c_2, e_1, e_2 >, \\ &\quad < x, q, c_1, c_2, e_1, e_2 > \to < x, q, c_1, c_2, e_1, e_2 >, < I > \to \\ &\quad < I > | x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \end{aligned}$$

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

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

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 $P_{a_{2Orig}}$ 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.

$$\begin{aligned} P_{a_{2Orig}} &= \{S \to S^3, S^{(1)} \to S^{(2)}, S^{(2)} \to S^{(3)}, S^{(3)} \to S^{(4)}\} \cup \\ &\quad \underline{\{S^{(4)} \to Q^{c_1}_{P_{2_{OrigS_2}}} Q^{c_1}_{P_{3_{OrigS_3}}} Q^{c_2}_{P_{2_{OrigS_2}}} Q^{c_2}_{P_{3_{OrigS_3}}} S^{(1)}\}}. \end{aligned}$$

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.

$$\begin{aligned} Reset_{GM^{c_1}_{Pa1_{S1}}} = Reset_{GM^{c_1}_{Pa1_{S1H2(S4)}}} = Reset_{GM^{c_1}_{Pa1_{S1H3(S4)}}} = \\ Reset_{GM^{c_1}_{Pa1_{S1(S2)}}} = Reset_{GM^{c_1}_{Pa1_{S1(S3)}}} = Reset_{GM^{c_1}_{Pa1_{S2}}} = \end{aligned}$$

$$\mathrm{Reset}_{GM^{c_1}_{Pa1_{S3}}} = \mathrm{Reset}_{GM^{c_2}_{Pa1_{S1}}} = \mathrm{Reset}_{GM^{c_2}_{Pa1_{S1H2(S4)}}} =$$
$$\mathrm{Reset}_{GM^{c_2}_{Pa1_{S1H3(S4)}}} = \mathrm{Reset}_{GM^{c_2}_{Pa1_{S1(S2)}}} = \mathrm{Reset}_{GM^{c_2}_{Pa1_{S1(S3)}}} =$$
$$\mathrm{Reset}_{GM^{c_2}_{Pa1_{S2}}} = \mathrm{Reset}_{GM^{c_2}_{Pa1_{S3}}} = \underline{\{S \rightarrow < I >, < I > \rightarrow < I >\}}$$

The components below will be used to reset $P^{c_1}_{1_{S_1H_2(S_4)}}$, $P^{c_1}_{1_{S_1H_3(S_4)}}$, $P^{c_2}_{1_{S_1H_2(S_4)}}$, and $P^{c_2}_{1_{S_1H_3(S_4)}}$ in Step 13 of the derivation. This reset allows queried components to be reset to their axioms which in turn allows the derivation to restart. These components were not part of the original system definition.

$$U_s = \{\, U \rightarrow U_1, U_1 \rightarrow U_2, U_2 \rightarrow U_3, U_3 \rightarrow U_4, U_4 \rightarrow U_5, U_6 \rightarrow U_7 \,\}$$

$$\begin{aligned}
\mathrm{Reset}_{P^{c_1}_{1_{S_1H_2(S_4)}}} &= \underline{U_s \cup \{U_7 \rightarrow Q_{P^{c_1}_{1_{S_1H_2(S_4)}}} U_4\}} \\
\mathrm{Reset}_{P^{c_1}_{1_{S_1H_3(S_4)}}} &= \underline{U_s \cup \{U_7 \rightarrow Q_{P^{c_1}_{1_{S_1H_3(S_4)}}} U_4\}} \\
\mathrm{Reset}_{P^{c_2}_{1_{S_1H_2(S_4)}}} &= \underline{U_s \cup \{U_7 \rightarrow Q_{P^{c_1}_{1_{S_1H_2(S_4)}}} U_4\}} \\
\mathrm{Reset}_{P^{c_2}_{1_{S_1H_3(S_4)}}} &= \underline{U_s \cup \{U_7 \rightarrow Q_{P^{c_1}_{1_{S_1H_3(S_4)}}} U_4\}}
\end{aligned}$$

The following, new grammars 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.

$$\begin{aligned}
T_s &= \{T \rightarrow T_1, T_1 \rightarrow T_2, T_2 \rightarrow T_3, T_3 \rightarrow T_4, T_4 \rightarrow T_5, T_6 \rightarrow T_7\} \\
\mathrm{Reset}_{P^{c_1}_{4_{S_1H_2(S_4)}}} &= \underline{T_s \cup \{T_7 \rightarrow Q_{P^{c_1}_{4_{S_1H_2(S_4)}}} T_4\}} \\
\mathrm{Reset}_{P^{c_1}_{4_{S_1H_3(S_4)}}} &= \underline{T_s \cup \{T_7 \rightarrow Q_{P^{c_1}_{4_{S_1H_3(S_4)}}} T_4\}} \\
\mathrm{Reset}_{P^{c_2}_{4_{S_1H_2(S_4)}}} &= \underline{T_s \cup \{T_7 \rightarrow Q_{P^{c_2}_{4_{S_1H_2(S_4)}}} T_4\}} \\
\mathrm{Reset}_{P^{c_2}_{4_{S_1H_3(S_4)}}} &= \underline{T_s \cup \{T_7 \rightarrow Q_{P^{c_2}_{4_{S_1H_3(S_4)}}} T_4\}}
\end{aligned}$$

5.2.3 The CF-PCGS simulation of the 2-counter machine

In order for our construction to be valid 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 2-counter machine M simulating the steps of M in their derivation. The system will change its configuration according to the state of M and 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). Throughout the derivation strings of terminals will appear in some components but will have no further role in the derivation; such occurrences have been silently replaced with generic symbols not appearing in the description of the grammars or 2-counter machines (mostly α and β).

The master grammar will control 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 $P_i^{c_1}$, $P_i^{c_2}$, $1 \leq i \leq 4$ to simulate the changes in the counters, as done in the original system [4]. All of the other component grammars included in our construction enable the original components to work correctly using one-step communication throughout the derivation. This design ensures an exclusive communication relationship between the the helper components that generate the same strings as the grammar components they are copying which ensures the correct string gets communicated to their corresponding original component at the right step. This ensure that the 2-counter machine works as it did in the original construction [4] because all of the strings generated in the original components are the same.

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 a two-column layout to represent the configurations of Γ. As mentioned earlier the 11 original grammars have the word "Orig" 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 2. 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 3. We then continue as shown in Figures 4 and 5.

Now we have yet another nondeterministic rewriting choice in several components; please refer to Figure 6. 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

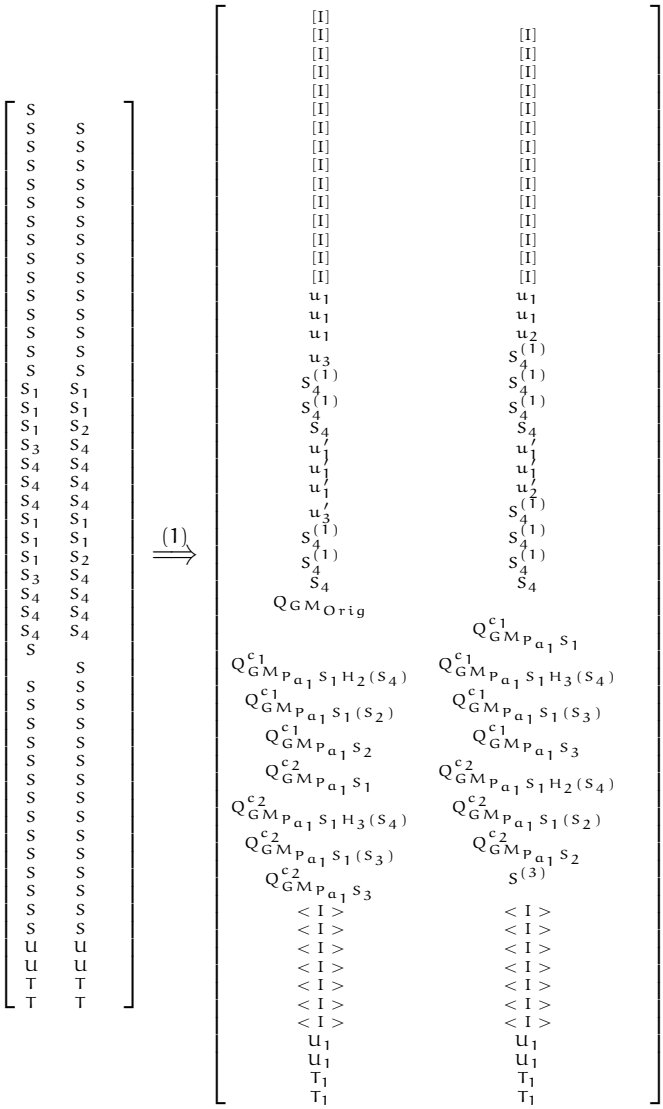

Figure 2: PCGS simulation of a 2-counter machine: Step 1 (nondeterministic).

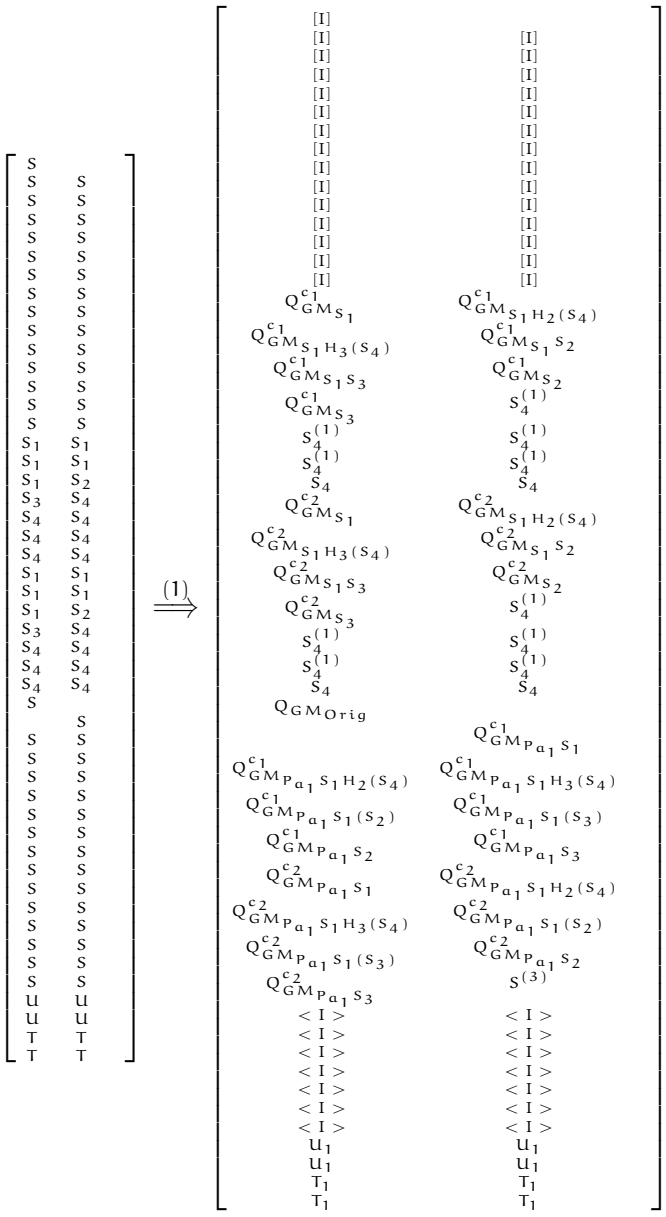

Figure 3: PCGS simulation of a 2-counter machine: Step 1.

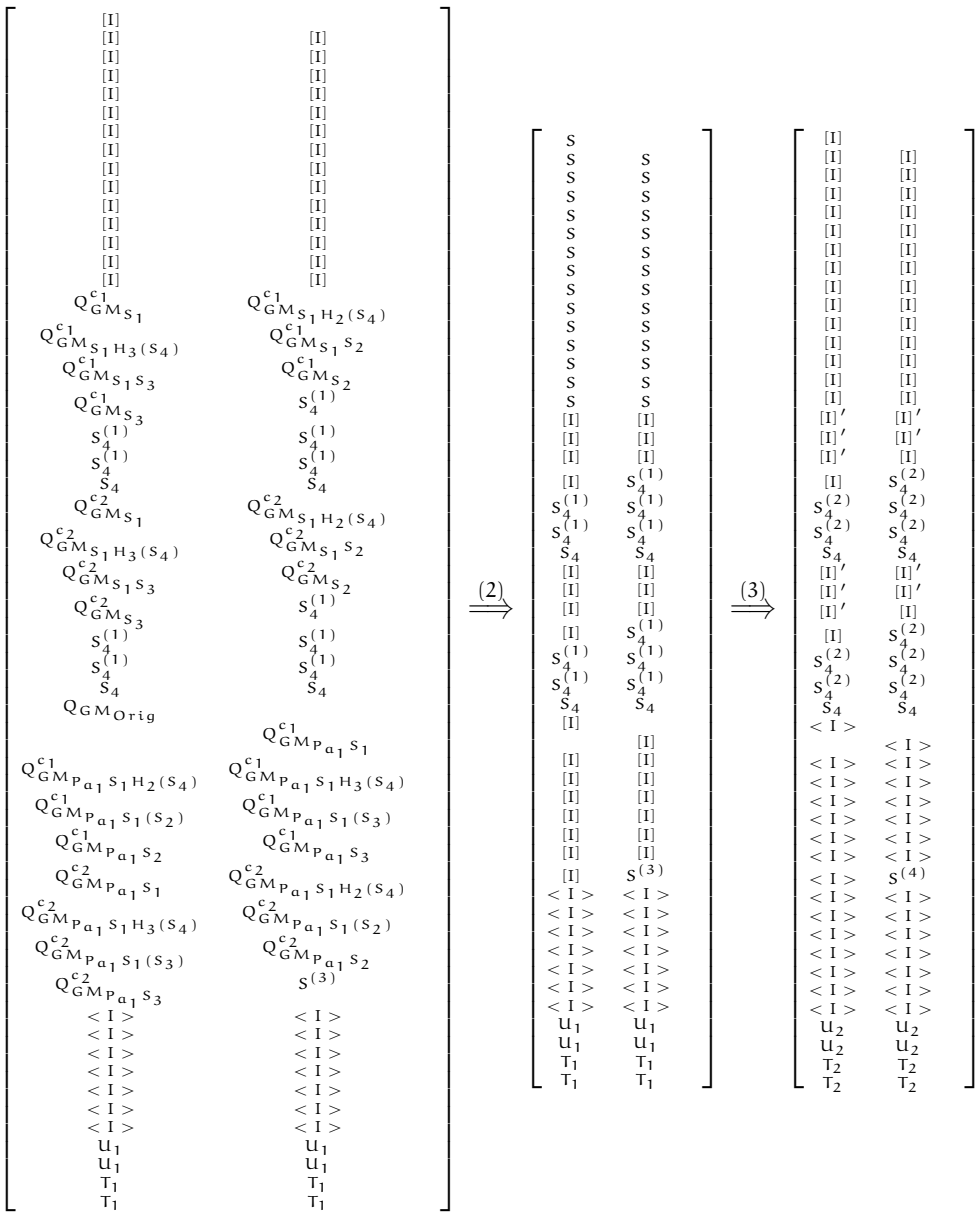

Figure 4: PCGS simulation of a 2-counter machine: Steps 2 and 3.

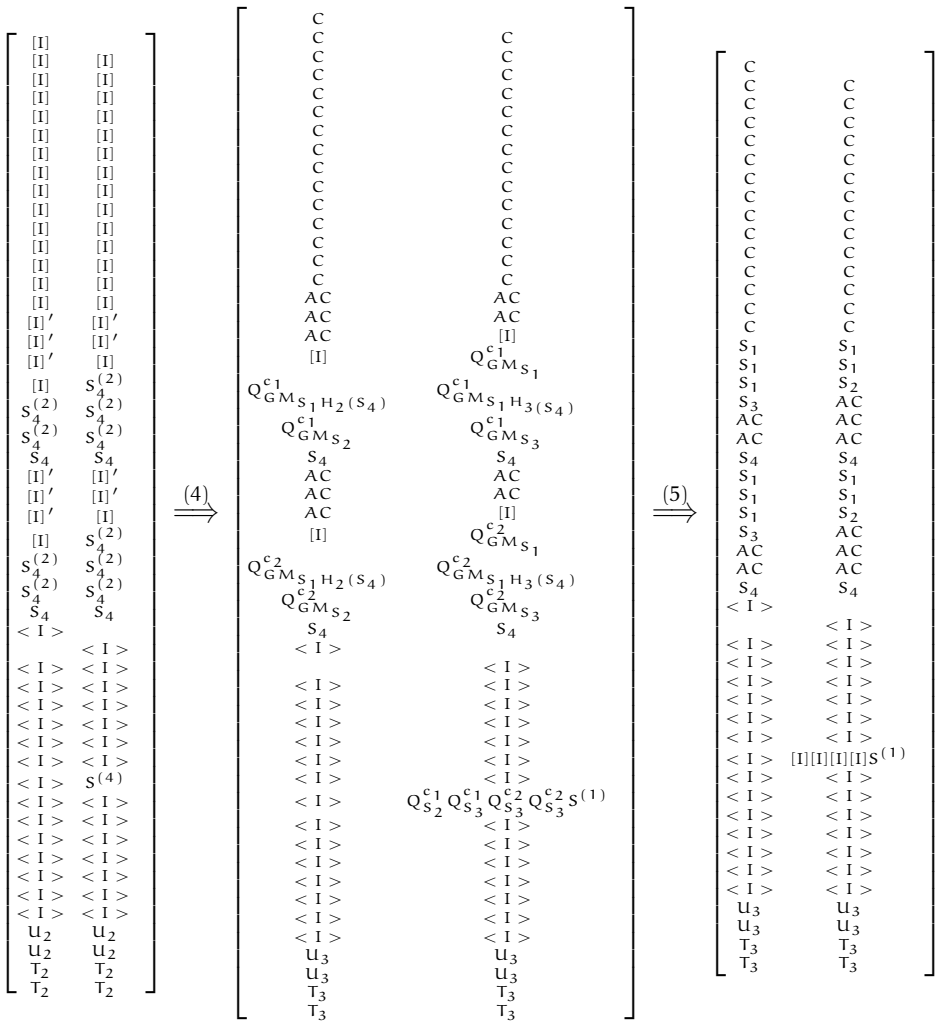

Figure 5: PCGS simulation of a 2-counter machine: Steps 4 and 5.

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 step makes use of reset queries; this ensures that all copy cat grammars that are mimicking the functionality of the master grammar remain synchronized with one another. Again, it is critical that all helper components that are copying the tasks of the original system generate the same string at the same time. The only possible step that will lead to a successful derivation is the one in Figure 6.

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 7 and 8.

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, 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 10) we use the new rule $S_1 \rightarrow Q_{4SpecHelp1}$ so its role in $P_{1_{S_1(S_2)}}^{c_1}$, $P_{1_{S_1(S_3)}}^{c_1}$ $P_{1_{S_1(S_2)}}^{c_2}$, and $P_{1_{S_1(S_3)}}^{c_2}$ components becomes apparent. This step ensures that $P_{2S_{2original}}^{c_1}$, $P_{2S_{3original}}^{c_1}$, $P_{2S_{2original}}^{c_2}$, $P_{2S_{3original}}^{c_3}$ receive the correct strings in Step 14.

Similar to the first step in the derivation in Step 13 the P_1, P_2, and P_3

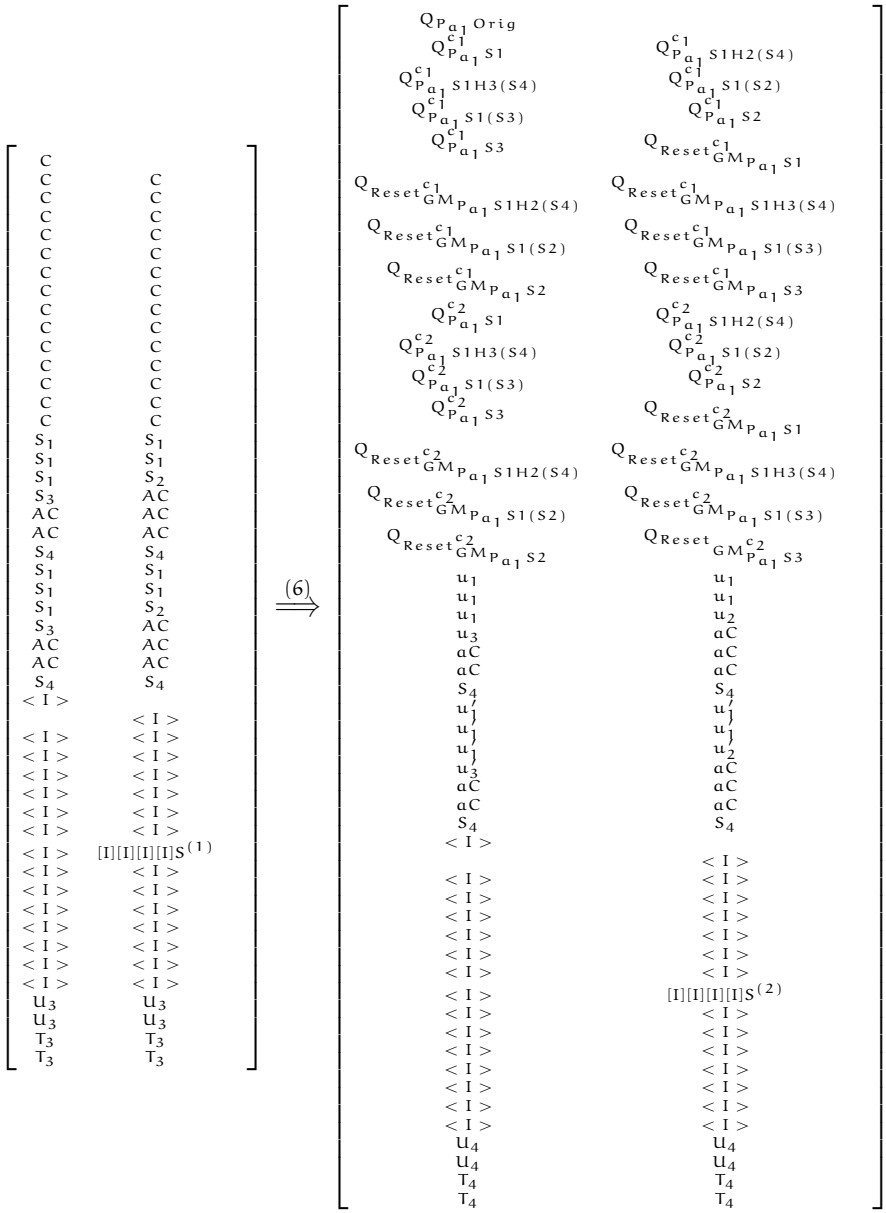

Figure 6: PCGS simulation of a 2-counter machine: Step 6.

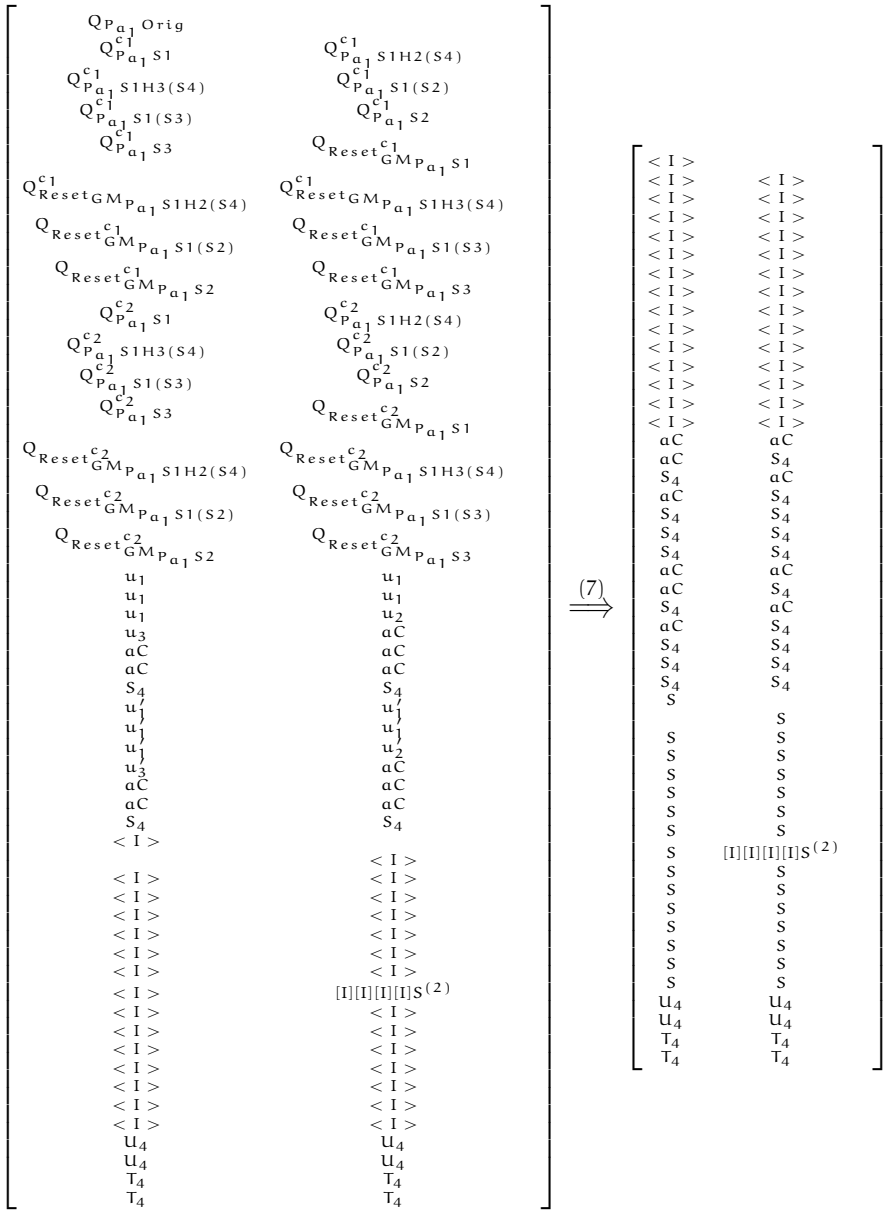

Figure 7: PCGS simulation of a 2-counter machine: Step 7.

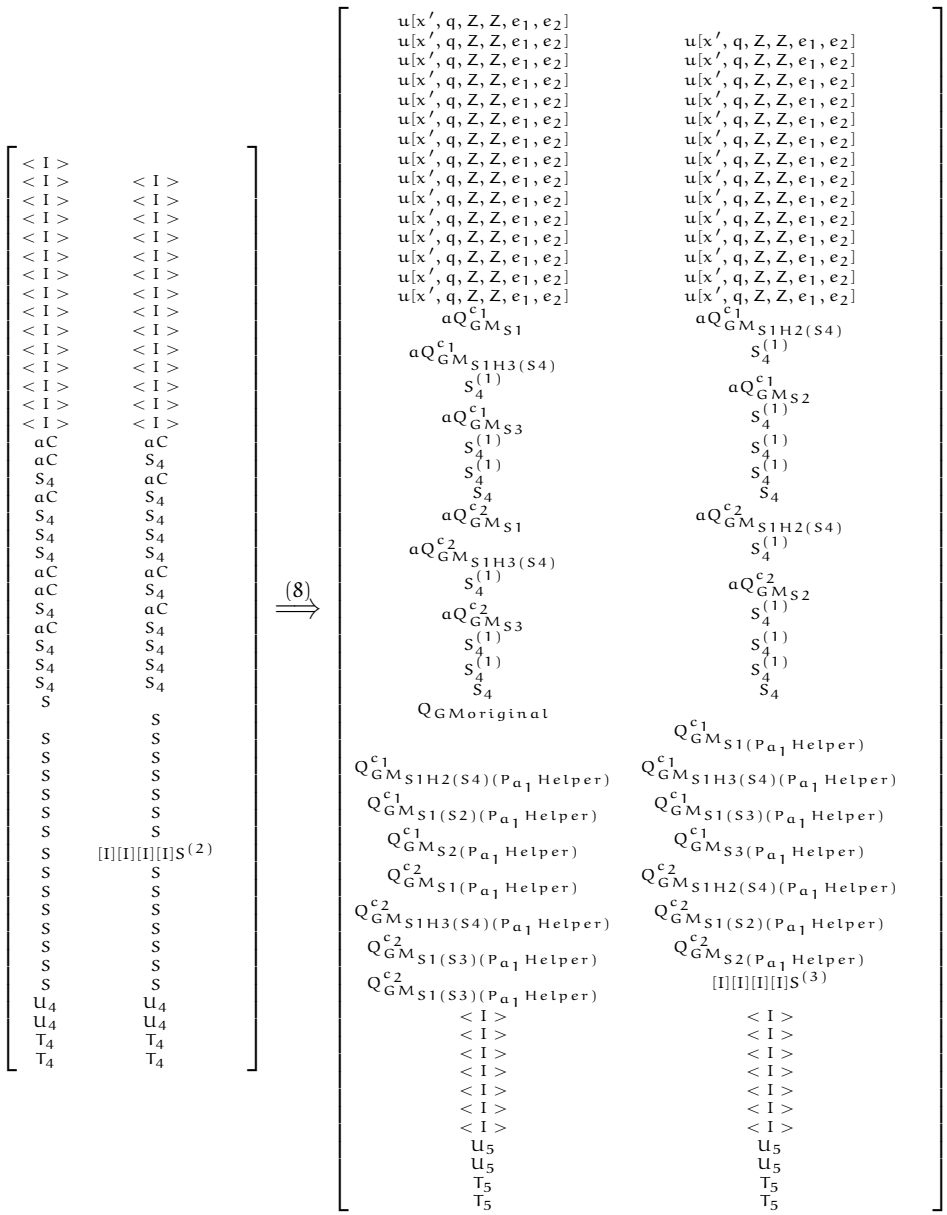

Figure 8: PCGS simulation of a 2-counter machine: Step 8.

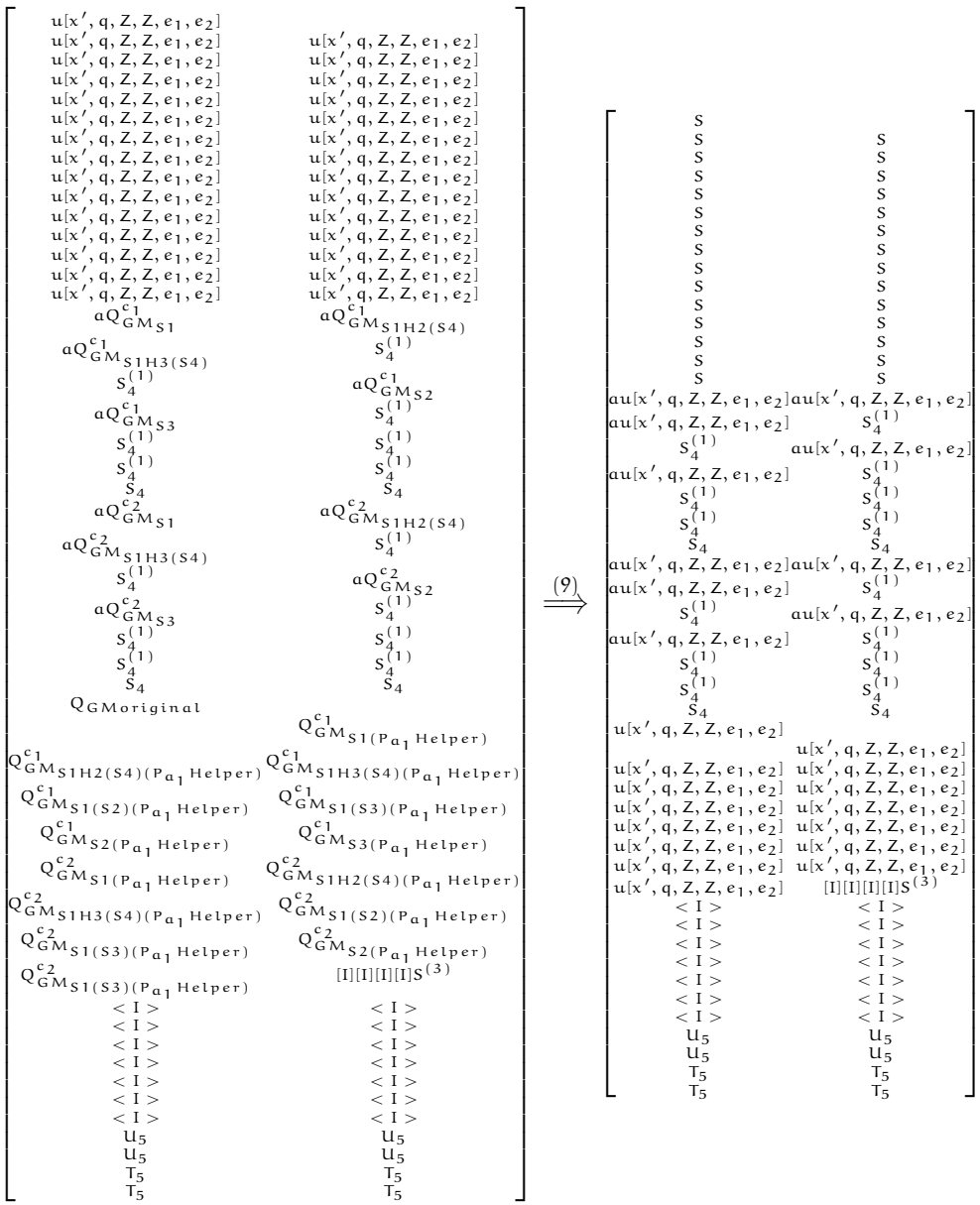

Figure 9: PCGS simulation of a 2-counter machine: Step 9.

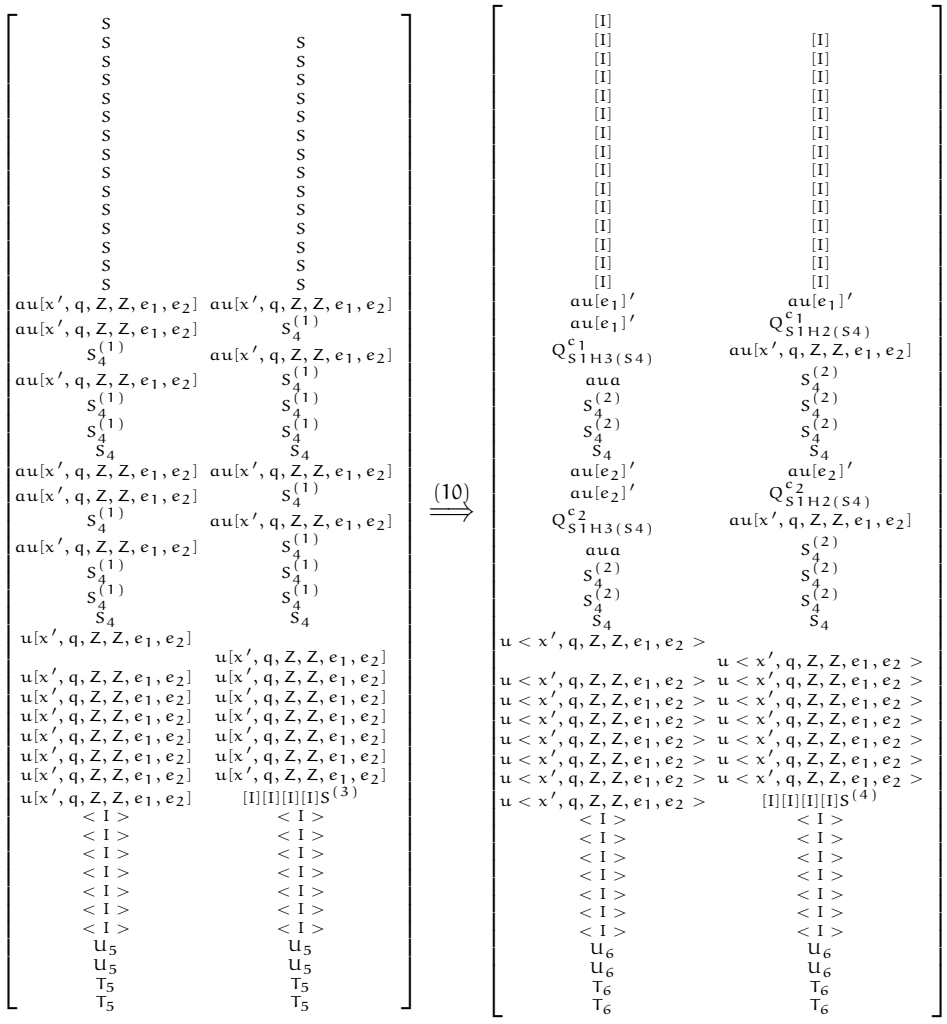

Figure 10: PCGS simulation of a 2-counter machine: Step 10.

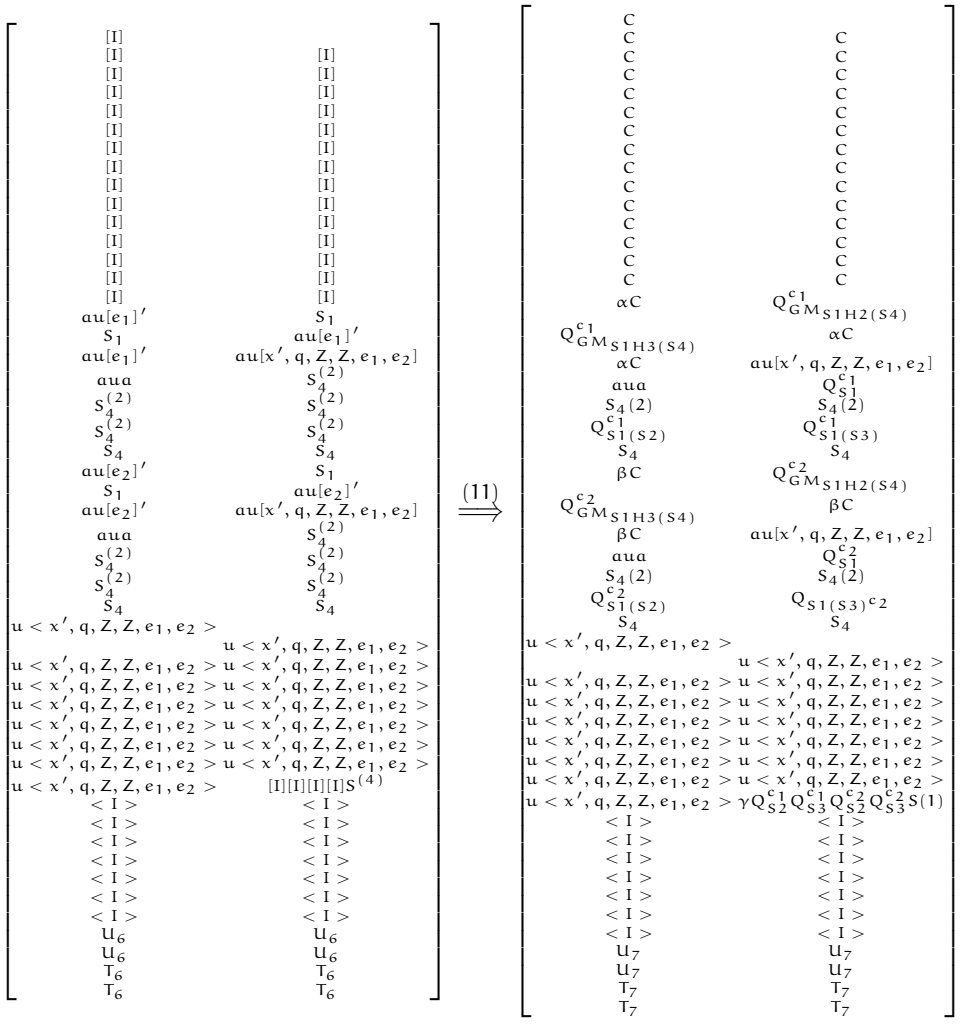

Figure 11: PCGS simulation of a 2-counter machine: Step 11.

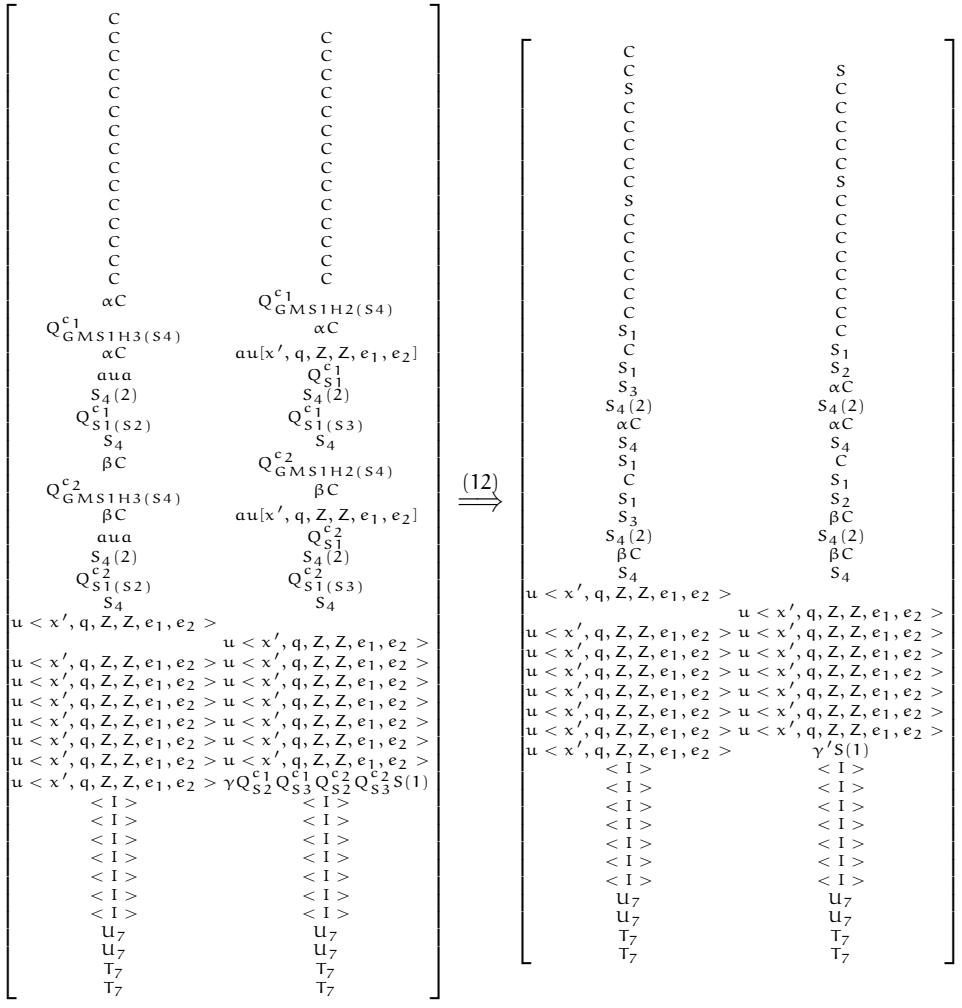

Figure 12: PCGS simulation of a 2-counter machine: Step 12.

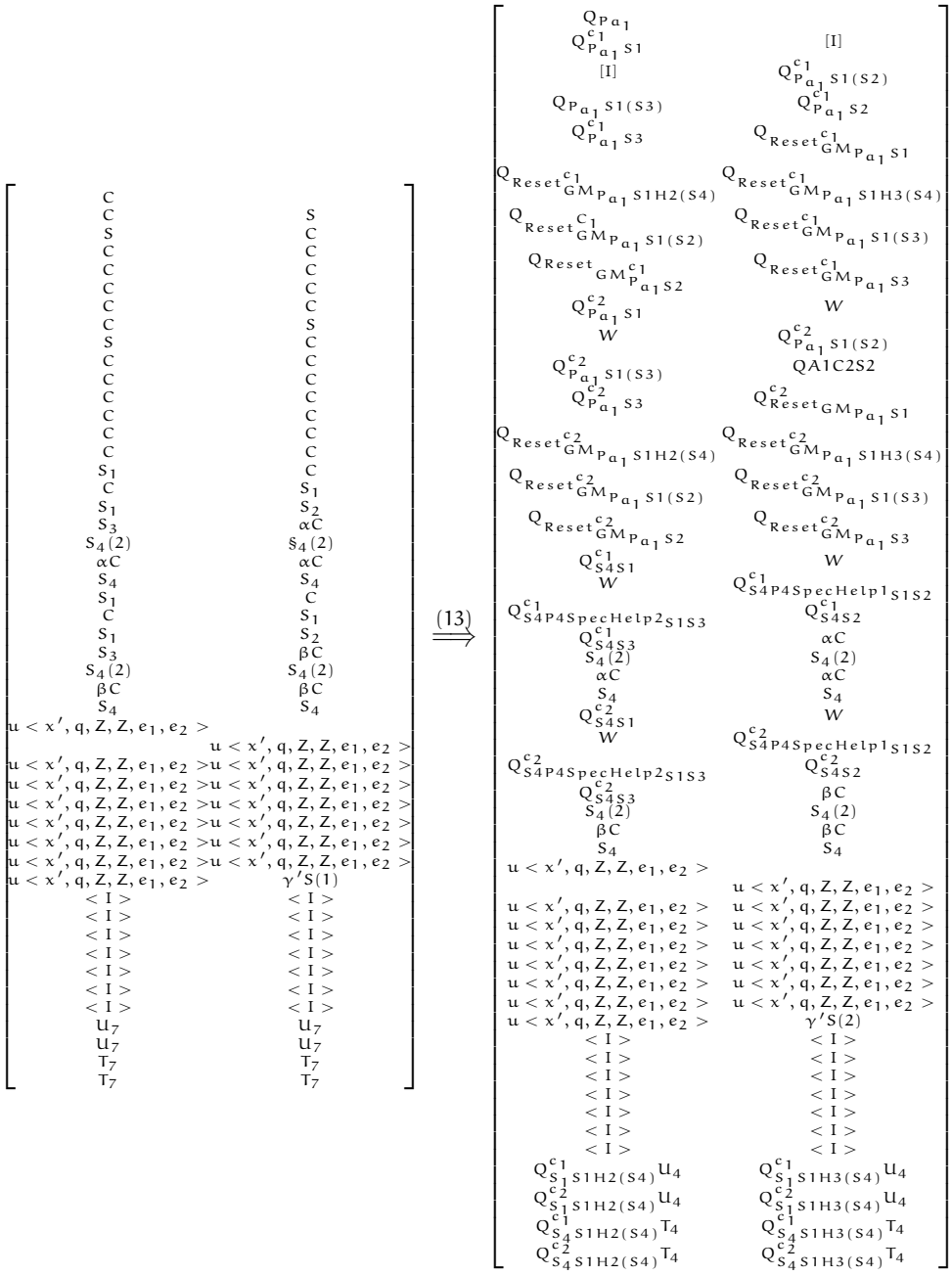

Figure 13: PCGS simulation of a 2-counter machine: Step 13.

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 14).

The following step (Figure 11) is a communication step. It allows two of the $P_1^{c_1}$ and $P_1^{c_2}$ helper 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 $S_2 \rightarrow S_2$ in order to avoid the introduction of a circular query. We continue as in Figures 12 and 13.

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 $< x', q_F, Z, Z, e_1, e_2 > \rightarrow x'$ 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 $\mathcal{L}(M)$. In such a case Γ can be modified to accept the empty string simply by adding the rule $S \rightarrow \varepsilon$ to its master grammar.

6 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

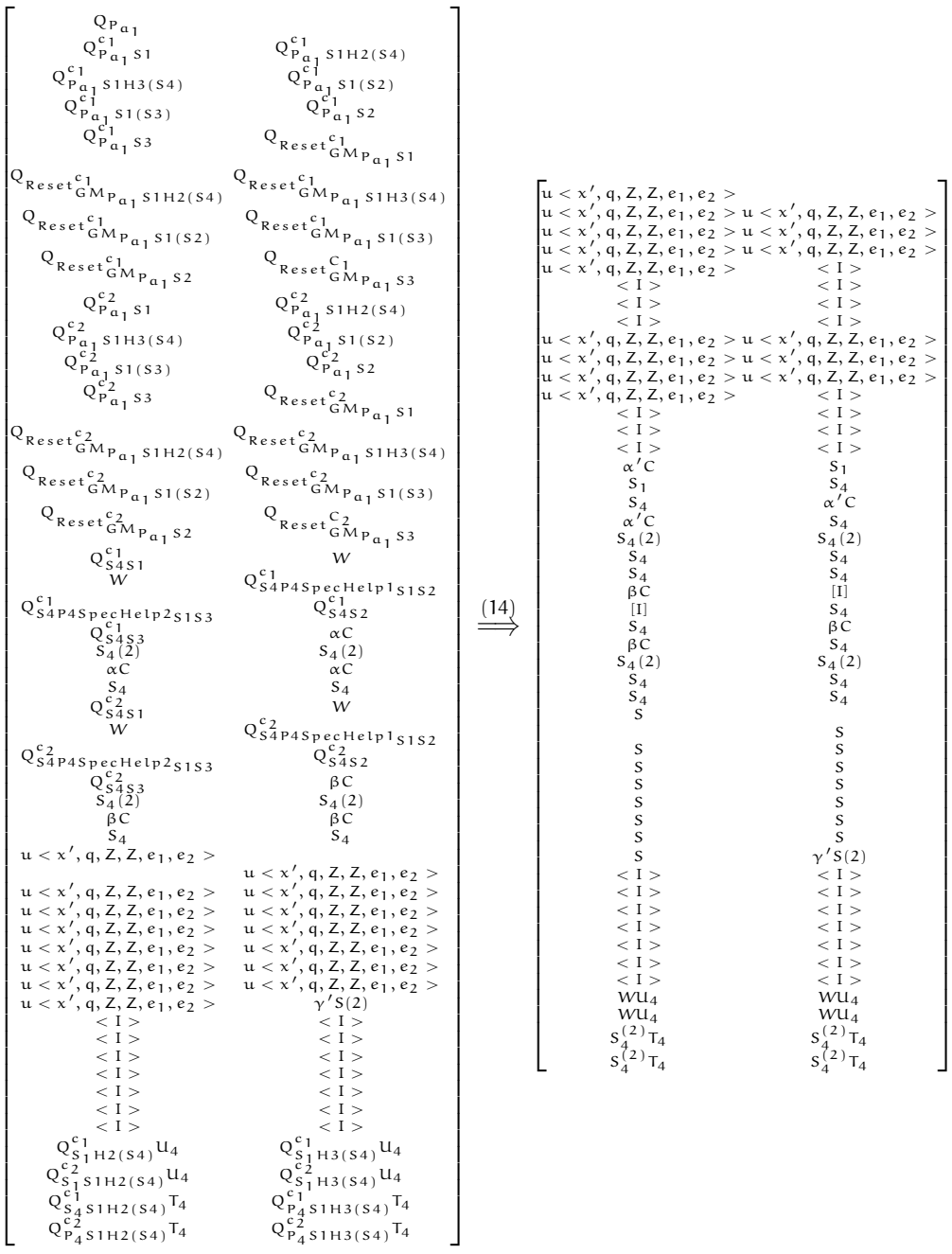

Figure 14: PCGS simulation of a 2-counter machine: Step 14.

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.

We first examined one system designed earlier (using broadcast communication) to show Turing completeness [4]. We explained that such an interpretation of communication modifies the power of the PCGS and hence this simulation does not work if one-step communication is used (Section 4). 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 helpers 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, 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 machine, and so show that CF-PCGS are indeed Turing complete using either style of communication (Theorem 5). Admittedly our construction is not as compact or elegant as the ones used in similar proofs [3, 2, 4], 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 [5, 14] 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 RE $= \mathcal{L}(\mathrm{PC}_*(\mathrm{CF}))$ instead of our RE $= \mathcal{L}(\mathrm{PC}_{95}(\mathrm{CF}))$. The other proof of Turing completeness for non-returning CF-PCGS [5] 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 RE $= \mathcal{L}(\mathrm{PC}_{127}(\mathrm{CF}))$ versus our RE $= \mathcal{L}(\mathrm{PC}_{95}(\mathrm{CF}))$. 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 machine simulation (featuring as many as 95 components) with the broadcast communication-enabled simulation [4] (with 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 be interesting to see whether our construction can be made even more concise, which we believe to be the case. Indeed, applying the techniques from this paper to another proof using broadcast communication [2] (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.

References

[1] S. D. Bruda, M. S. R. Wilkin, Parse trees for context-free parallel communicating grammar systems, *Proc. 13th International Conference on Automation and Information (ICAI 12)*, Iasi, Romania, June 2012, pp. 144–149. ⇒114

[2] E. Csuhaj-Varjú, G. Paun, G. Vaszil, PC grammar systems with five context-free components generate all recursively enumerable languages, *Theoretical Computer Science* **299** (2003) 785–794. ⇒119, 121, 124, 167, 168

[3] E. Csuhaj-Varjú, On size complexity of context-free returning parallel communicating grammar systems, in: *Where Mathematics, Computer Scients, Linguistics and Biology Meet*, (ed. C. Martin-Vide and V. Mitrana), Springer, 2001, pp. 37–49. ⇒119, 121, 124, 167

[4] E. Csuhaj-Varjú, G. Vaszil, On the computational completeness of context-free parallel communicating grammar systems, *Theoretical Computer Science*, **215** (1999) 349–358. ⇒115, 119, 121, 124, 125, 126, 129, 141, 150, 167, 168

[5] E. Csuhaj-Varjú, G. Vaszil, On the size complexity of non-returning context-free PC grammar systems, *Proc. 11th International Workshop on Descriptional Complexity of Formal Systems (DCFS 2009)*, Magdeburg, Germany, 2009, pp. 91–100. ⇒119, 121, 168

[6] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, G. Păun, *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, 1994. ⇒114, 115, 116, 118, 119, 124

[7] J. Dassow, G. Păun, G. Rozenberg, Grammar systems, in: *Handbook of Formal Languages – Volume 2. Linear Modeling: Background and Applications*, Springer, 1997, pp. 155–213. ⇒119

[8] S. Dumitrescu, Nonreturning PC grammar systems can be simulated by returning systems, *Theoretical Computer Science*, **165** (1996) 463–474. ⇒114, 121, 168

[9] P. C. Fischer, Turing machines with restricted memory access, *Information and Computation*, **9** (1966) 364–379. ⇒115, 126

[10] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Macmillan Higher Education, 1979. ⇒116

[11] V. Geffert, Context-free-like forms for the phrase-structure grammars, *Mathematical Foundations of Computer Science*, *Lecture Notes in Computer Science*, **324** (1988) 309–317. ⇒119

[12] G. Katsirelos, S. Maneth, N. Narodytska, T. Walsh, Restricted global grammar contraints, *Proc. Principles and Practice of Constraint Programming (CP 2009)*, *Lecture Notes in Computer Science*, **5732** (2009) 501–508. ⇒116

[13] H. R. Lewis, C. H. Papadimitriou, *Elements of the Theory of Computation*, Prentice Hall, 2nd edition, 1998. ⇒116

[14] N. Mandache, On the computational power of context-free PC grammar systems, *Theoretical Computer Science*, **237** (2000) 135–148. ⇒114, 121, 168

[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. ⇒114

[16] V. Mihalache, On the generative capacity of parallel communicating grammer systems with regular components, Technical report, Turku Centre for Computer Science, Turku, Finland, 1996. ⇒118

[17] V. Mihalache, On the expressiveness of coverability trees for PC grammar systems, in *Grammatical Models of Multi-Agent Systems (Topics in Computer Mathematics)*, Gordon and Breach, 1999. ⇒114

[18] D. Pardubska, M. Platek, Parallel communicating grammar systems and analysis by reduction by restarting automata, Technical report, Deptartment of Computer Science, Comenius University, Bratislava, Slovakia, 2008. ⇒118

[19] G. Păun, L. Sântean, Parallel communicating grammar systems: the regular case, *Analele Universitatii din Bucuresti, Seria Matematica-Informatica*, **2** (1989) 55–63. ⇒114

[20] G. Păun, L. Sântean, Further remarks on parallel communicating grammar systems, *International Journal of Computer Mathematics*, **34** (1990) 187–203. ⇒115

[21] L. Sântean, Parallel communicating grammar systems, *Bulletion of the EATCS (Formal Language Theory Column)*, **1**, 1990. ⇒114, 118

[22] F. L. Tiplea, C. Ene, C. M. Ionescu, O. Procopiuc, Some decision problems for parallel communicating grammar systems. *Theoretical Computer Science*, **134** (1994) 365–385. ⇒114

[23] G. Vaszil, On simulating non-returning PC grammar systems with returning systems, *Theoretical Computer Science*, **209** (1997) 319–329. ⇒168

[24] G. Vaszil, Various communications in PC grammar systems, *Acta Cybernetica*, **13** (1997) 173–196. ⇒115, 121

[25] M. S. R. Wilkin, S. D. Bruda, Parallel communicating grammar systems with context-free components are Turing complete for any communication model, Technical Report 2014-003, Department of Computer Science, Bishop's University, 2014. ⇒115

Received: October 10, 2016 • Revised: November 3, 2016