Acta Universitatis Sapientiae

Informatica

Volume 8, Number 2, 2016

Sapientia Hungarian University of Transylvania
Scientia Publishing House

Contents

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* $(x_1, \ldots, x_n) \Rightarrow (y_1, \ldots, y_n)$ *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.*

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

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

We will show however in Section 4 that the above results regarding returning CF-PCGS [2, 3, 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, $y_{i_j} = S_{i_j}$, $1 \le j \le t$).

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 [2, 3], 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.

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

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:

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

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

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

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:

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

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

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

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 $G^y_{a_{1_x}}$ 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 $G^{y}_{a_1 x}$ 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 $G^{c_1}_{1_x}$, ..., $G^{c_2}_{4_x}$ 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 $P_{a_1 Orig}$. 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 $G_{a_1 x}$ 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 $P_{1_{OrigS_1}}^{c_1}$ 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.

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

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

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

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

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

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).

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

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

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

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

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

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

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

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

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

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

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.

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

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.

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

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).

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

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

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

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

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

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

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

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

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

$P^{c_1}_{1_{OrigS_1}}$ 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.

$$P^{c_1}_{1_{OrigS_1}} = \underline{\{S_1 \rightarrow Q^{c_1}_{GMS_1}, S_1 \rightarrow Q^{c_1}_{4S_{1original}}, C \rightarrow Q^{c_1}_{GMS_1}\}} \cup$$
$$\{[x, q, c_1, c_2, e_1, e_2] \rightarrow [e_1]', [+1]' \rightarrow AAC, [0]' \rightarrow AC,$$
$$[-1]' \rightarrow C | x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\}\} \cup$$
$$\{[I] \rightarrow [I]', [I]' \rightarrow AC\}$$

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).

$$P^{c_1}_{1_{S_1H_2(S_4)}} = \underline{\{S_1 \rightarrow Q^{c_1}_{GMS_1H_2(S_4)}, S_1 \rightarrow Q^{c_1}_{4S_1H_2(S_4)}, C \rightarrow Q_{GMS_1H_2(S_4)},}$$
$$\underline{C \rightarrow W\}} \cup$$
$$\{[x, q, c_1, c_2, e_1, e_2] \rightarrow [e_1]', [+1]' \rightarrow AAC, [0]' \rightarrow AC,$$
$$[-1]' \rightarrow C | x \in \Sigma, q \in E, c_1, c_2 \in \{Z, B\}, e_1, e_2 \in \{-1, 0, +1\} \cup$$
$$\{[I] \rightarrow [I]', [I]' \rightarrow AC\}$$

$$P^{c_1}_{1_{S_1H_3(S_4)}} = \underline{\{S_1 \rightarrow Q^{c_1}_{GMS_1H_3(S_4)}, S_1 \rightarrow Q^{c_1}_{4S_1H_3(S_4)}, C \rightarrow Q_{GMS_1H_3(S_4)},}$$

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

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 $P^{c_2}_{2_{Orig S_2}}$ and $P^{c_2}_{3_{Orig S_3}}$ 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.

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

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

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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

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

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

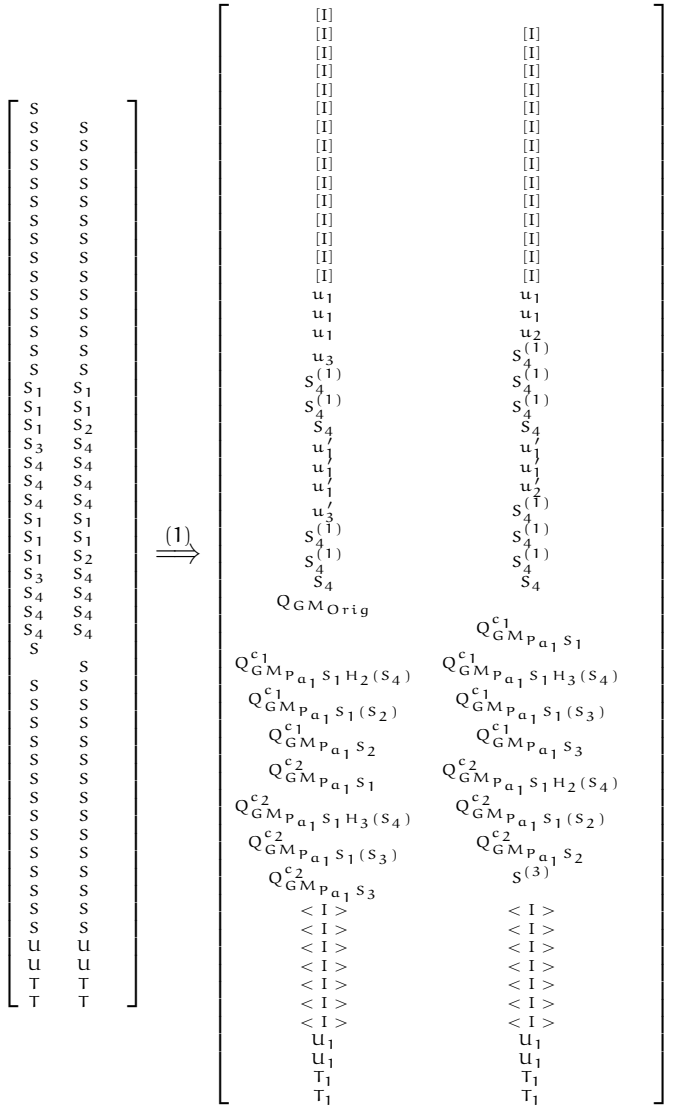

Figure 2: PCGS simulation of a 2-counter machine: Step 1 (nondeterministic).

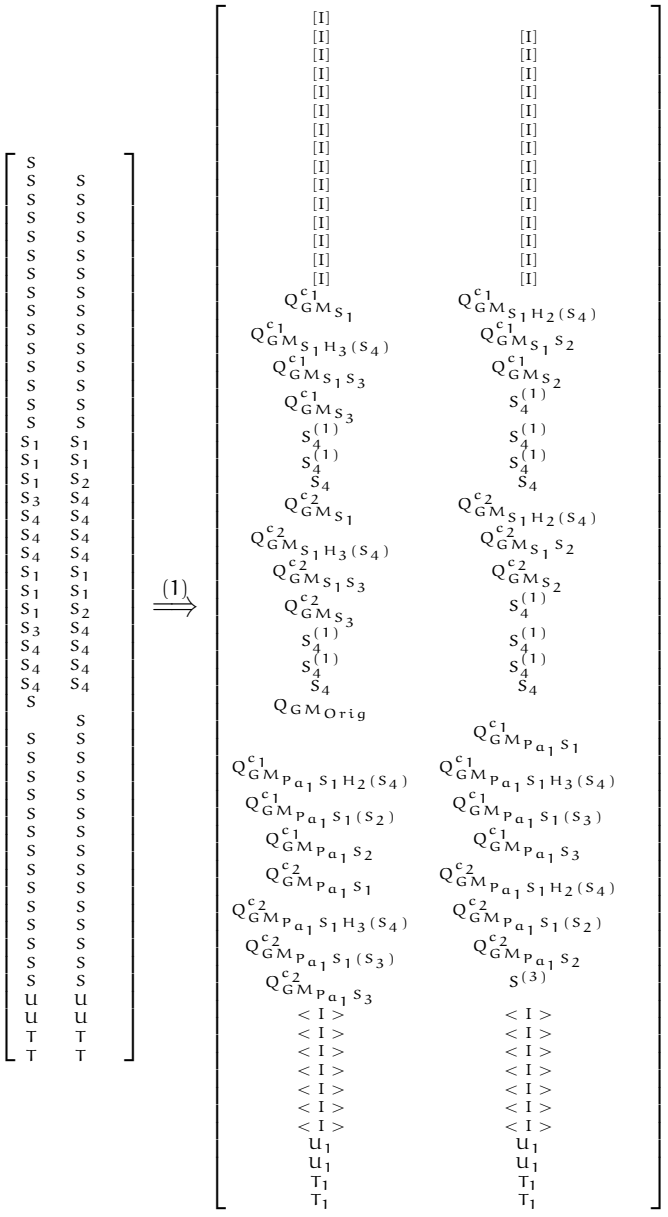

Figure 3: PCGS simulation of a 2-counter machine: Step 1.

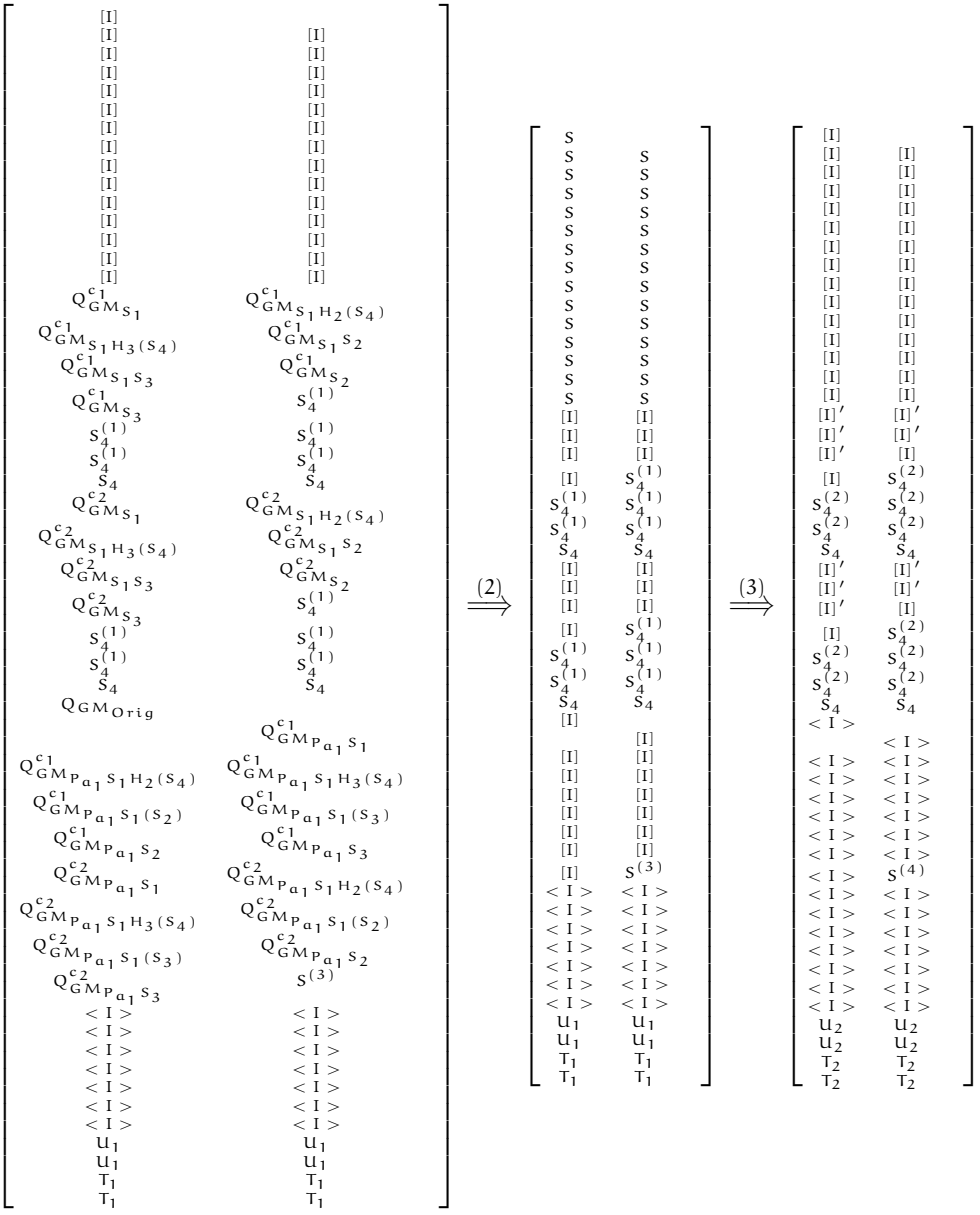

Figure 4: PCGS simulation of a 2-counter machine: Steps 2 and 3.

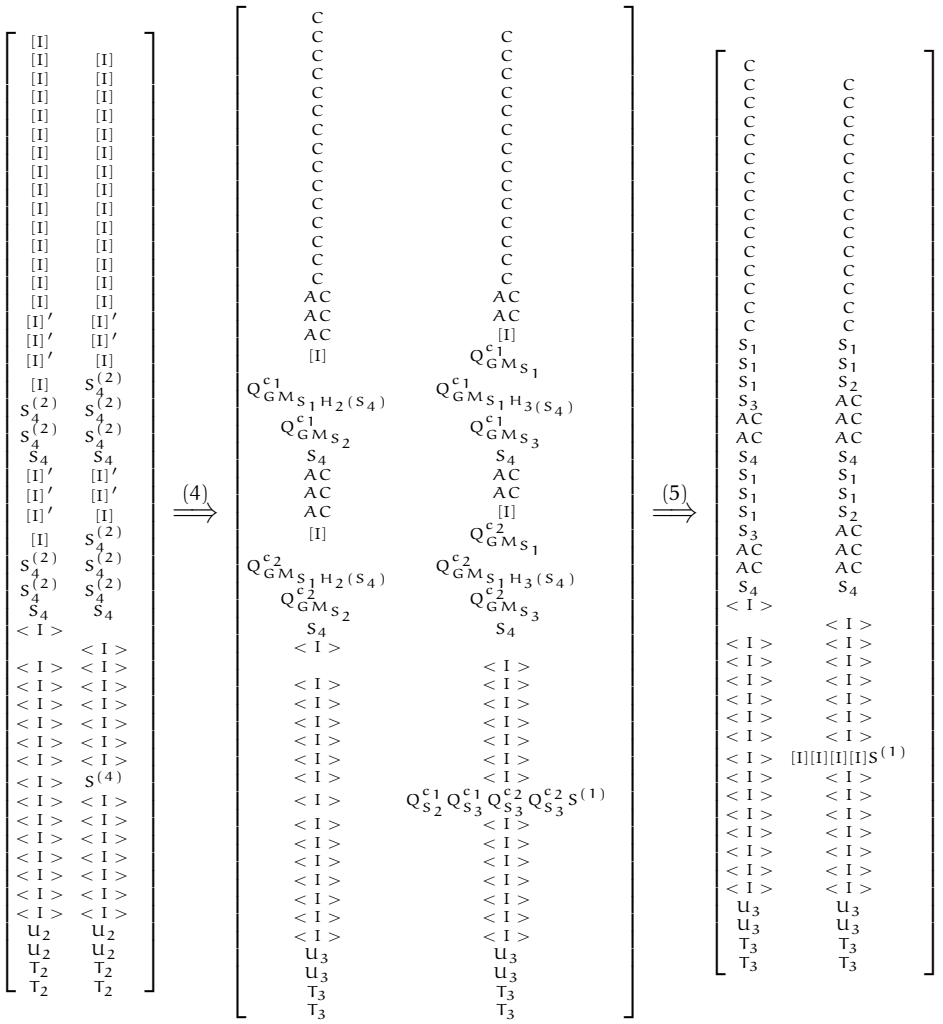

Figure 5: PCGS simulation of a 2-counter machine: Steps 4 and 5.

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

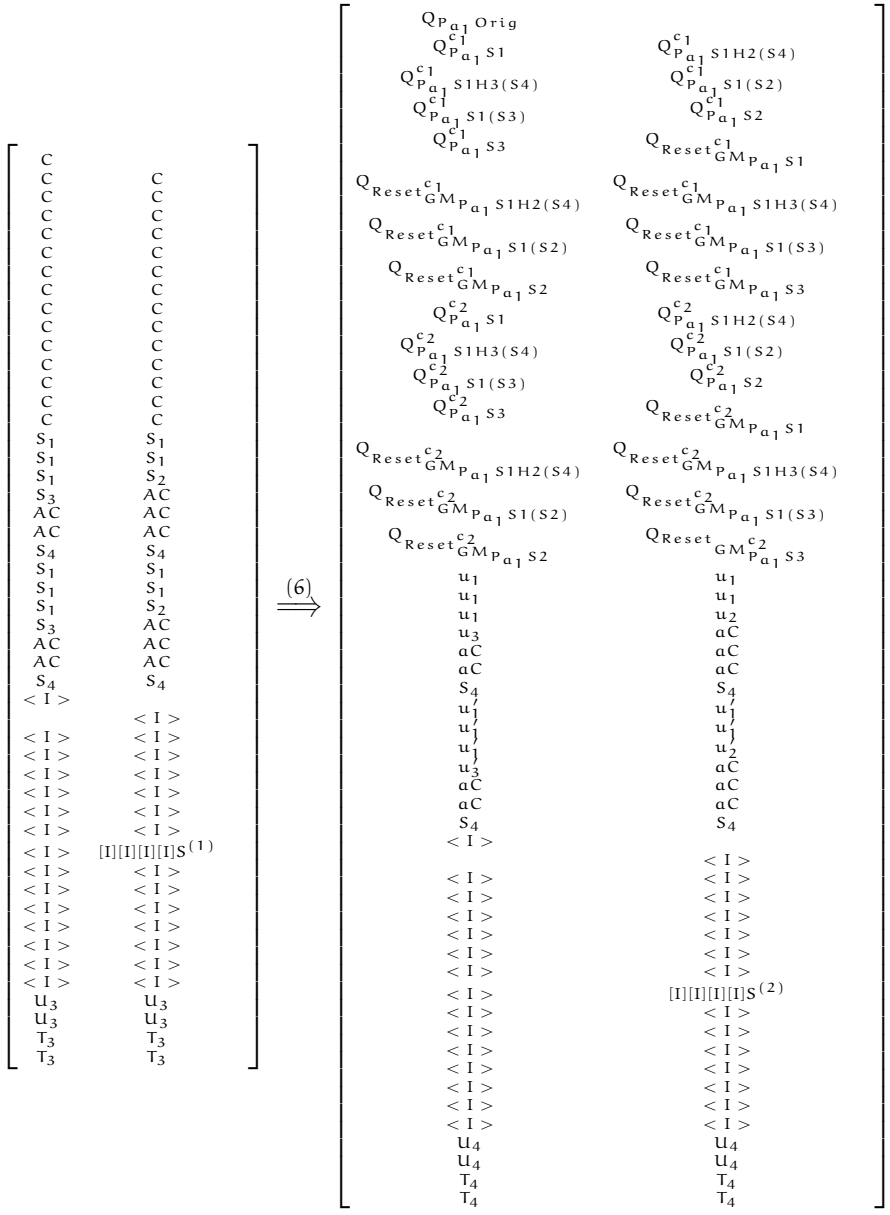

Figure 6: PCGS simulation of a 2-counter machine: Step 6.

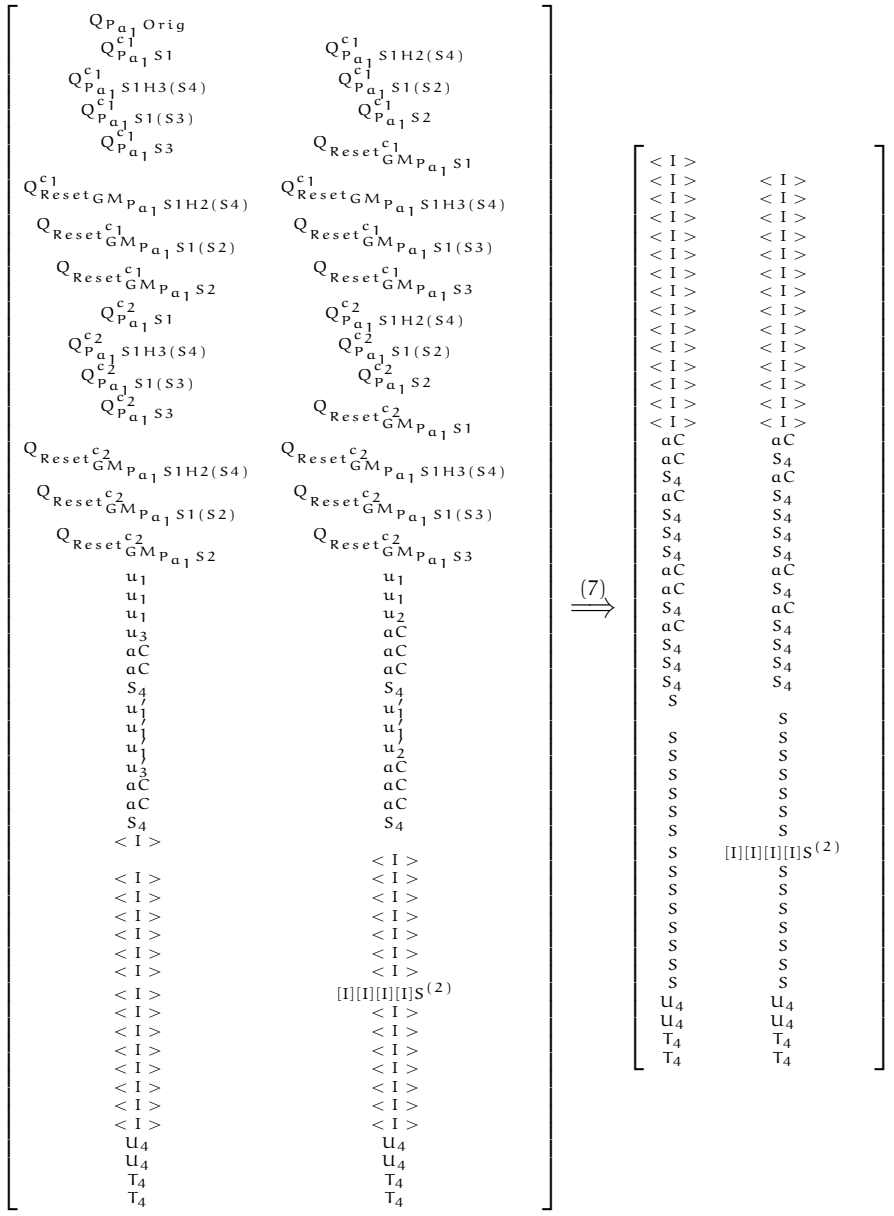

Figure 7: PCGS simulation of a 2-counter machine: Step 7.

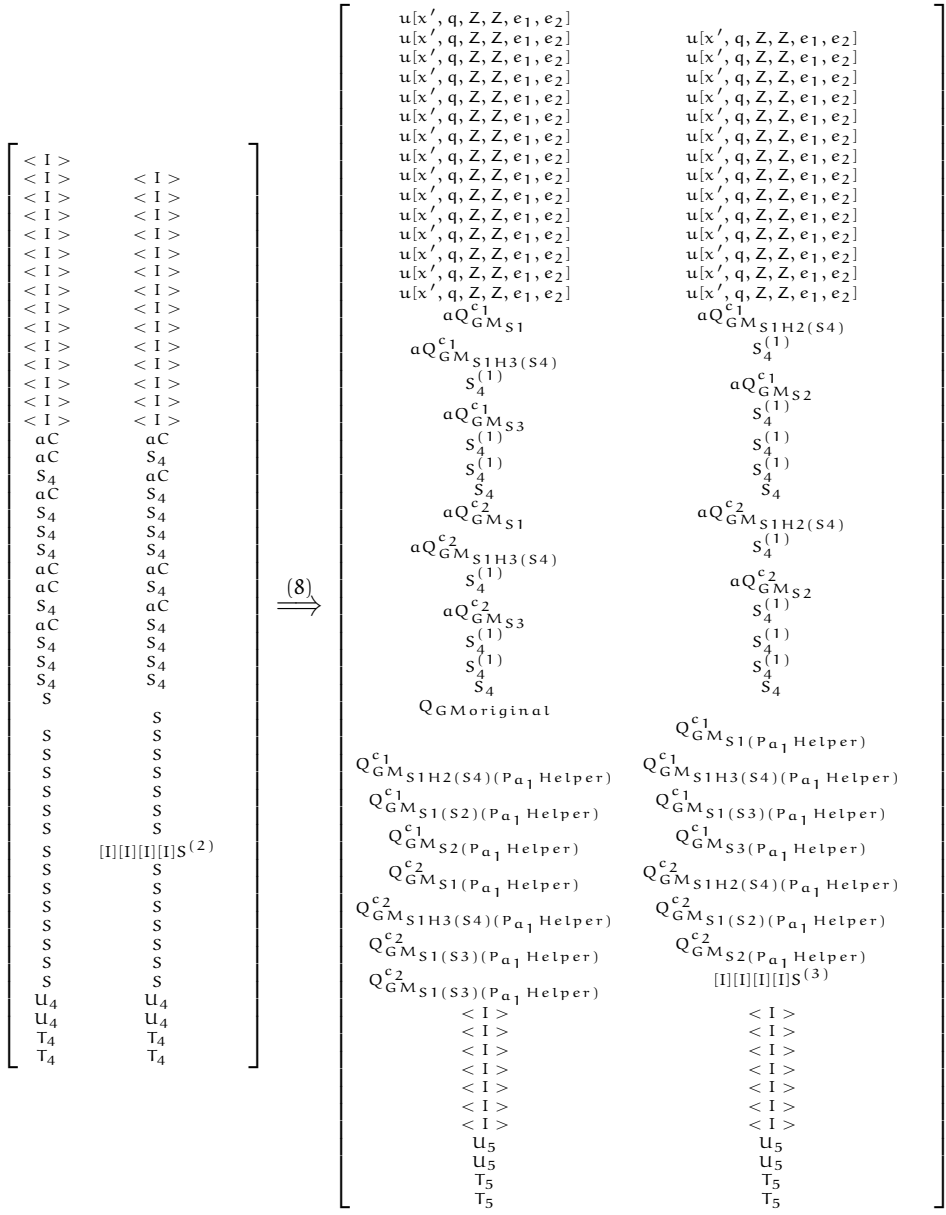

Figure 8: PCGS simulation of a 2-counter machine: Step 8.

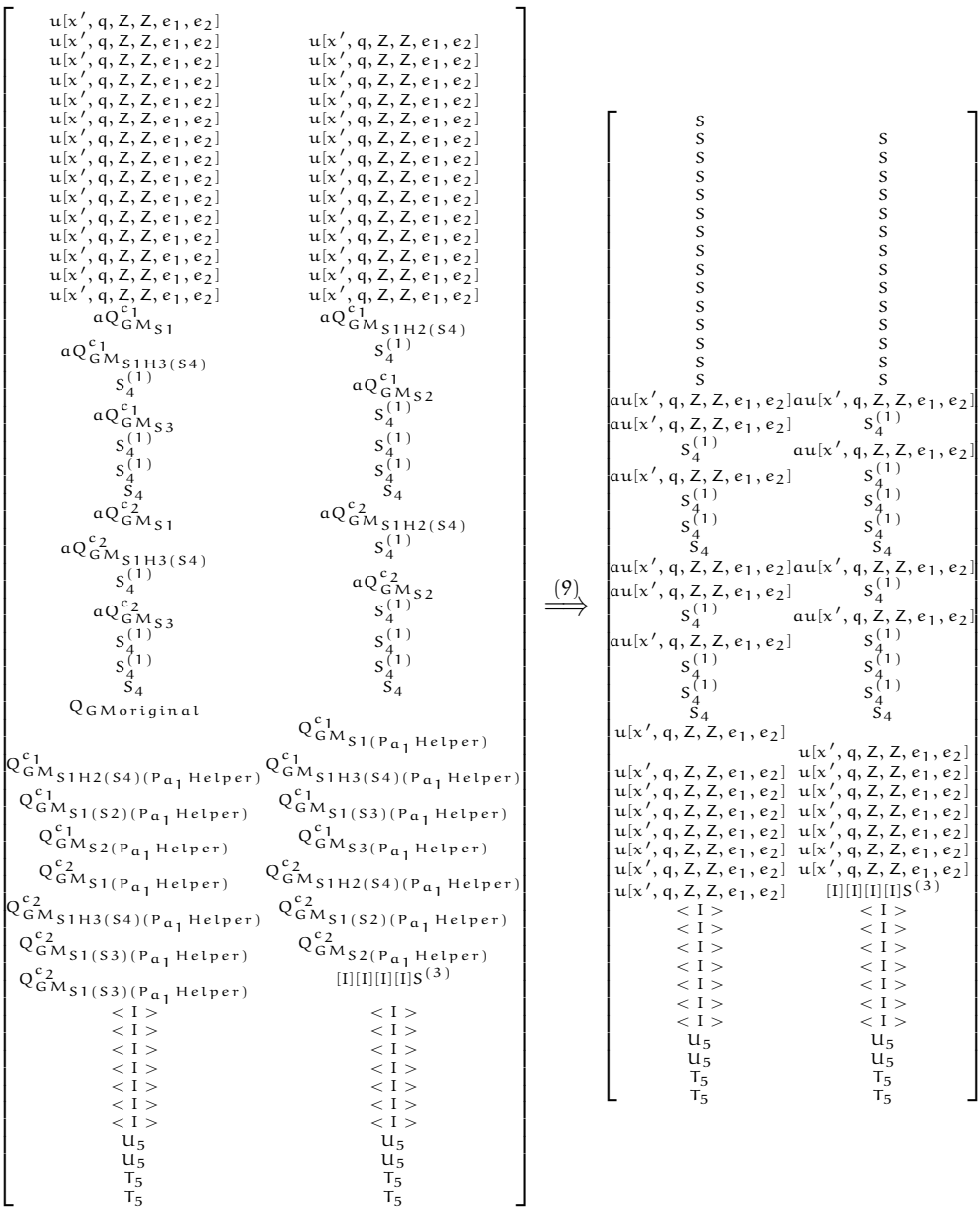

Figure 9: PCGS simulation of a 2-counter machine: Step 9.

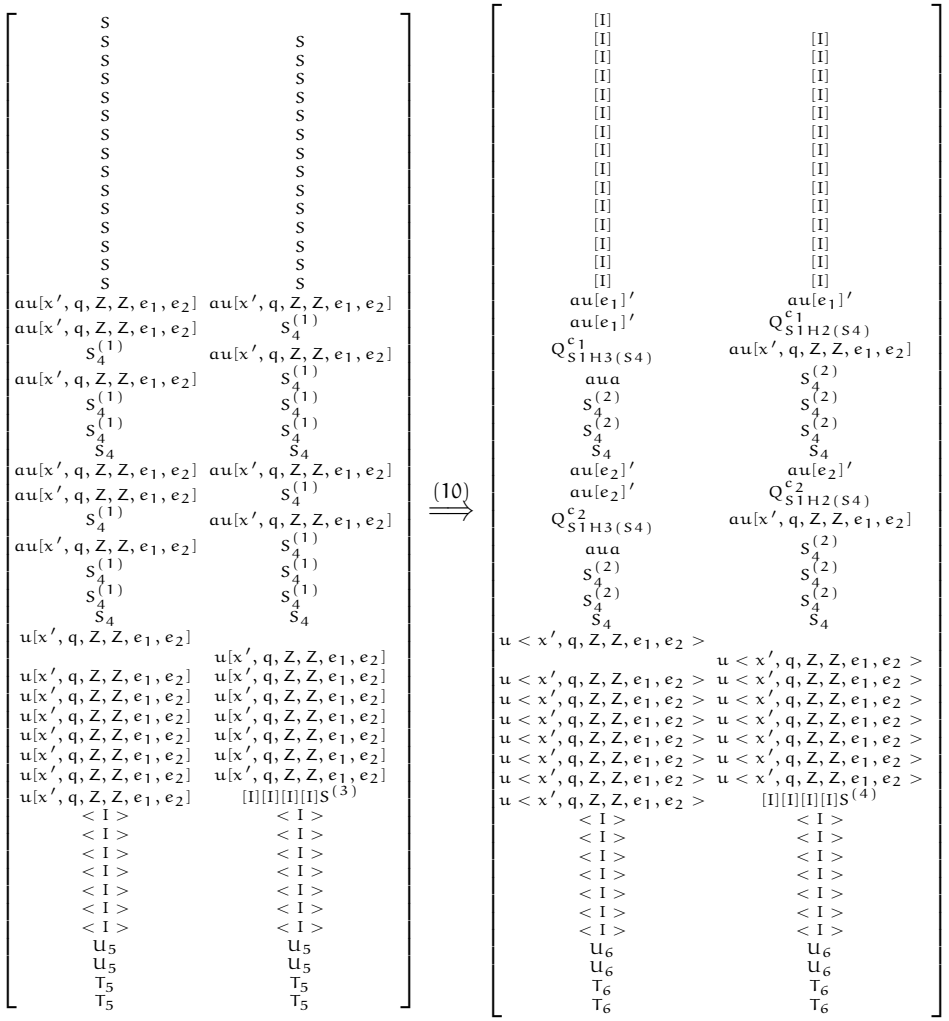

Figure 10: PCGS simulation of a 2-counter machine: Step 10.

$$
\begin{bmatrix}
[I] & \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
[I] & [I] \\
au[e_1]' & S_1 \\
S_1 & au[e_1]' \\
au[e_1]' & au[x', q, Z, Z, e_1, e_2] \\
aua & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4 & S_4 \\
au[e_2]' & S_1 \\
S_1 & au[e_2]' \\
au[e_2]' & au[x', q, Z, Z, e_1, e_2] \\
aua & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4^{(2)} & S_4^{(2)} \\
S_4 & S_4 \\
u < x', q, Z, Z, e_1, e_2 > & \\
 & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & [I][I][I][I]S^{(4)} \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
U_6 & U_6 \\
U_6 & U_6 \\
T_6 & T_6 \\
T_6 & T_6
\end{bmatrix}
\xRightarrow{(11)}
\begin{bmatrix}
C & \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
C & C \\
\alpha C & Q^{c_1}_{GM_{S1H2(S4)}} \\
Q^{c_1}_{GM_{S1H3(S4)}} & \alpha C \\
\alpha C & au[x', q, Z, Z, e_1, e_2] \\
aua & Q^{c_1}_{S1} \\
S_4(2) & S_4(2) \\
Q^{c_1}_{S1(S2)} & Q^{c_1}_{S1(S3)} \\
S_4 & S_4 \\
\beta C & Q^{c_2}_{GM_{S1H2(S4)}} \\
Q^{c_2}_{GM_{S1H3(S4)}} & \beta C \\
\beta C & au[x', q, Z, Z, e_1, e_2] \\
aua & Q^{c_2}_{S1} \\
S_4(2) & S_4(2) \\
Q^{c_2}_{S1(S2)} & Q_{S1(S3)}c_2 \\
S_4 & S_4 \\
u < x', q, Z, Z, e_1, e_2 > & \\
 & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & u < x', q, Z, Z, e_1, e_2 > \\
u < x', q, Z, Z, e_1, e_2 > & \gamma Q^{c_1}_{S2} Q^{c_1}_{S3} Q^{c_2}_{S2} Q^{c_2}_{S3} S(1) \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
< I > & < I > \\
U_7 & U_7 \\
U_7 & U_7 \\
T_7 & T_7 \\
T_7 & T_7
\end{bmatrix}
$$

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

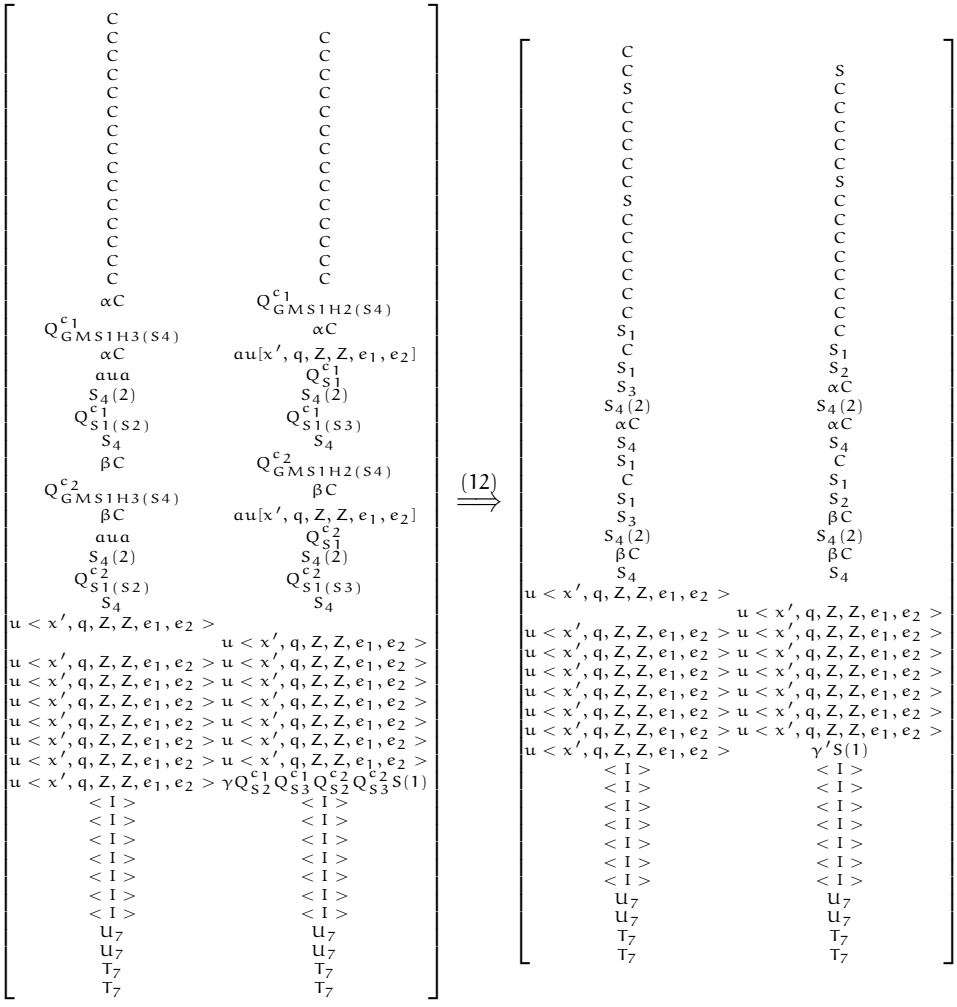

Figure 12: PCGS simulation of a 2-counter machine: Step 12.

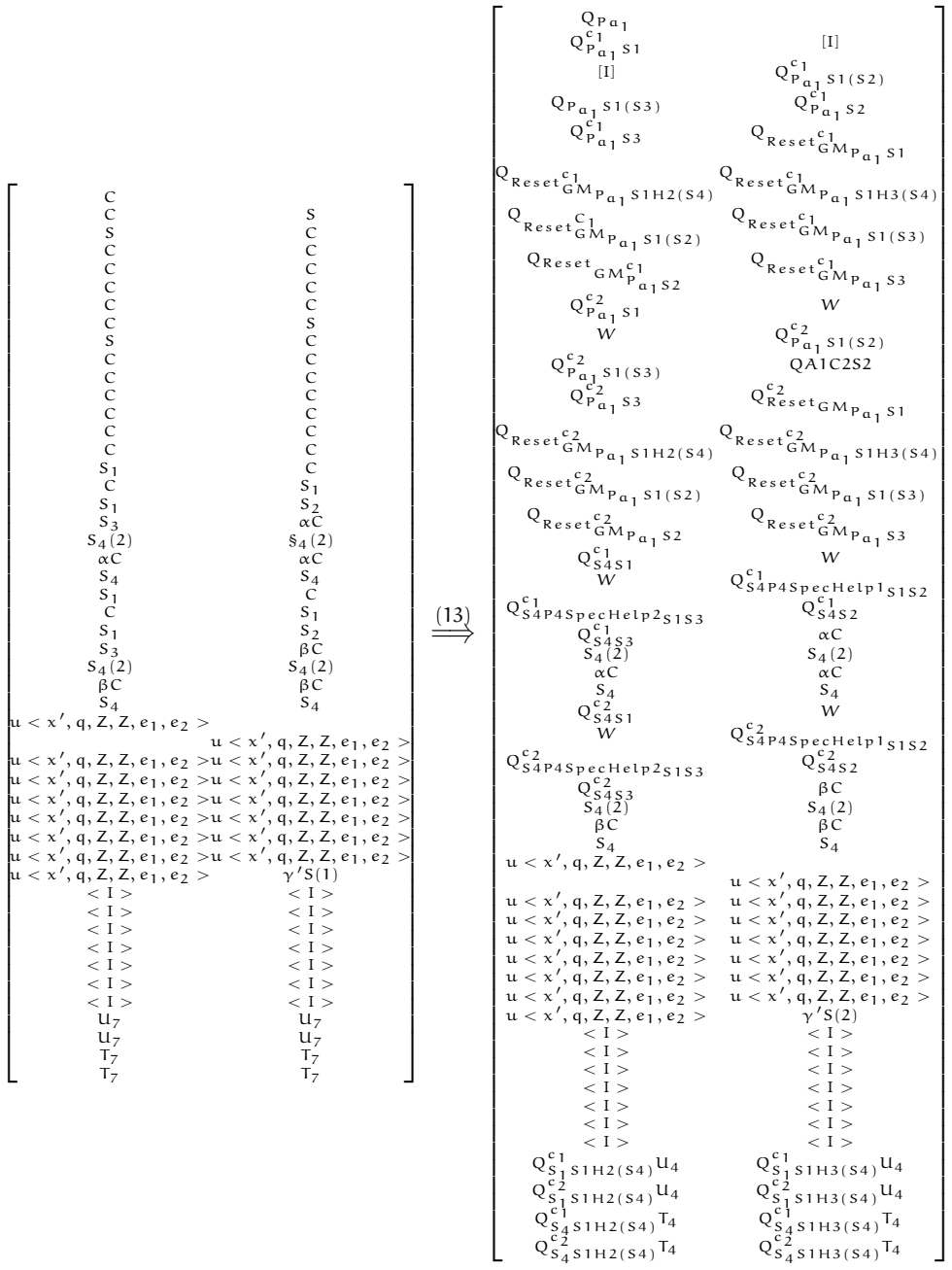

Figure 13: PCGS simulation of a 2-counter machine: Step 13.

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 $S_2 \to S_2$ 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 $< x', q_F, Z, Z, e_1, e_2 > \to x'$ 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: $< x', q, Z, Z, e_1, e_2 > \to x[y, q', c_1', c_2', e_1', e_2']$. If the scanned symbol does not change the input head will not move, and G_m can then use the following rule: $< x', q, Z, Z, e_1, e_2 > \to [x', q', c_1', c_2', e_1', e_2']$. 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 $S \to \varepsilon$ 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

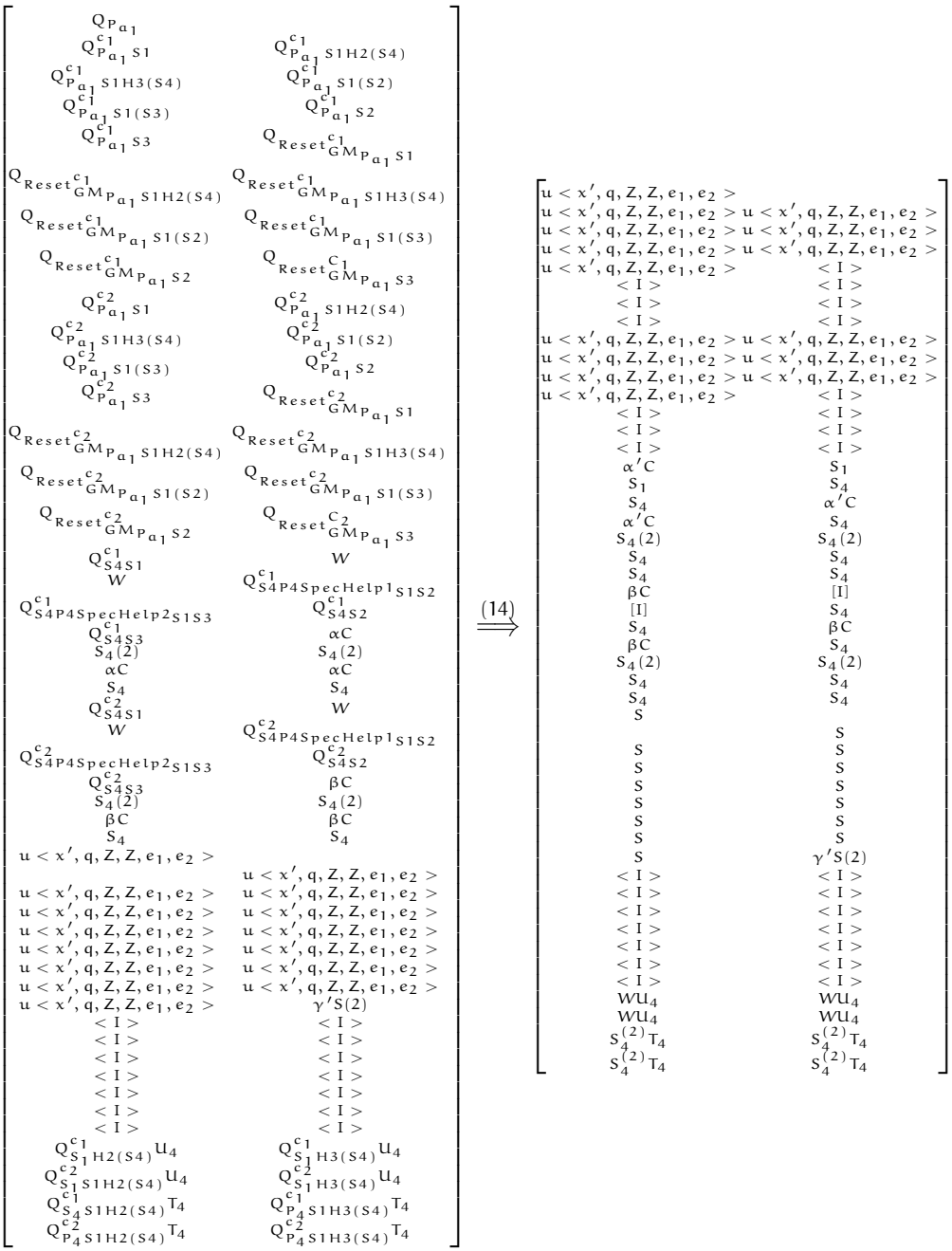

Figure 14: PCGS simulation of a 2-counter machine: Step 14.

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 [2, 3, 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

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **8,** 2 (2016) 171–185

DOI: 10.1515/ausi-2016-0008

Jet browser model accelerated by GPUs

Richárd FORSTER
Eötvös Loránd University
Faculty of Informatics
email: forceuse@inf.elte.hu

Ágnes FÜLÖP
Eötvös Loránd University
Faculty of Informatics
email: fulop@caesar.elte.hu

Abstract.

In the last centuries the experimental particle physics began to develop thank to growing capacity of computers among others. It is allowed to know the structure of the matter to level of quark gluon. Plasma in the strong interaction. Experimental evidences supported the theory to measure the predicted results. Since its inception the researchers are interested in the track reconstruction. We studied the jet browser model, which was developed for 4π calorimeter. This method works on the measurement data set, which contain the components of interaction points in the detector space and it allows to examine the trajectory reconstruction of the final state particles. We keep the total energy in constant values and it satisfies the Gauss law. Using GPUs the evaluation of the model can be drastically accelerated, as we were able to achieve up to 223 fold speedup compared to a CPU based parallel implementation.

1 Introduction

The huge measurement data is generated in the experiment of high energy particle physics e.g. in the ATLAS experiment (CERN) $\sim 40 \times 10^6$ events per second are detected which requires 64 TB/sec. Every year at about one

Computing Classification System 1998: I.1.4
Mathematics Subject Classification 2010: 58A20
Key words and phrases: jet, trajectory reconstruction algorithm, database of experimenthal particle physics parallel computing, GPU, CUDA

milliard events are measured, this represents three milliard simulation of the event each year and the number of detected particles is growing exponentially. The evaluation requires large computer capacity. Many fundamental questions raise in this research. One of these is the trajectory reconstruction. This process contains more levels. The first is a clearing the pure measurement data. Next process contains the path reconstruction. This method includes two types of elaboration. One of them is an online process to apply the level of assembly program. In the next step the valuable stored data is studied by high level computer program in batch mode.

In our article we work on the first level to use directly measured data set. A jet browser algorithm was published in [1] to study the particle orbit in 4π calorimeter.

The GPUs are providing an easy to program parallel model [10] that makes us capable to achieve higher precision in our computations, while also reaching out for bigger datasets [5]. The ever evolving and increasingly efficient architecture of the GPUs makes it also a viable option to run the applications even on a laptop these days without relying on supercomputers and very special not consumer level hardwares. Thus we implement a CUDA based parallel implementation of the Jet browser to broaden the limit of the original algorithm.

2 Jet physics

The jet physics [6] plays important role in the high energy physics. We present the process from the collision of protons to observable particles by detector (Figure 1).

It contains three different parts. The first range is the *parton* session, which contains the quark gluon plasma with strong interaction on the distance 10^{-15}m. In the theoretical model the jet are produced in hard scattering processes generating high transverse momentum quarks or gluon. The next part of the process is *particle*, where the constituents are formed from the quarks and gluon. This is called *hadronisation* procedure. In the last phase we detect the final state objects, which are the components of the electromagnetic and hadronic showers. The theoretical studying applies the Monte Carlo simulations. At the *parton* and *particle* process PYTHIA [8] software package is used to calculate and at *detector* phase the final state particle is simulated by GEANT [11].

We mention that, there are more kind definitions of the jet, because the laws of physics are different between short-range and the finale hadronisation.

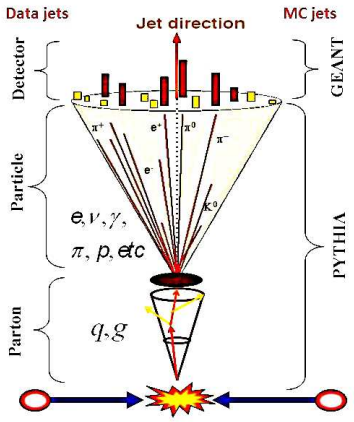

Figure 1: The structure of jet

It is important, that the definition is adequate with both theoretically and experimentally.

3 Jet algorithm

The experimental data which is contains of large number of particles to measure their four-momentum. The particle's energy, impulses p_T and position retrieve from data set. Two main jet definitions [2] are being used. One of them is Cone Jet and the other is the k_T Jet algorithm. In the cone jet algorithm we apply that the jet stays in circular range in the plain of the detector to describe by angles to search the stable energy state. The k_T algorithm can be used, where the values and direction of particle's momentum have the same order of magnitude, therefore the finale state showers will be collinear. These definitions fulfill both theoretical and experimental model also. Several advanced procedures [4, 7, 3] has been developed in this research field.

4 Parallel graph based trajectory reconstruction

In [1] a graph based trajectory reconstructing method was introduced. As all reconstruction algorithms, this is also very computation intensive, where even if the input dataset isn't very large, the combination of those inputs can generate a lot of work. To effectively speed up the process now we introduce a

CUDA based GPU implementation for the same problem, as the GPUs proved to be a valuable device in parallel computations [5].

4.1 Jet browser model

The jet analysis and the jet reconstruction method are applied to study the structure of jet in high energy physics. A shower is a narrow cone of hadrons and other particles which are produced in *detector* phase. These constituents are measured by detector to determine the trajectories and the type of constituents [1].

In 2009 year Gy. Vesztergombi has presented a new idea of a 4π detector [9]. A few particle trajectories (e^-, e^+, γ) can be reconstructed by a model, which was published [1]. The component of shower take part in the electromagnetic and strong interaction, decaying in two or more particles, but we neglect the strong interaction, because the cross section is very small. We can measure the point of the elements (e^-, e^+), which consists of components of the three dimensional Euclidean space. The jet reconstruction model involves the energy and charge conservation. It has been proven that the orbits correspond to weighted directed tree graph $G(\Psi, E, V, w)$.

This method consists of three steps: We find all of neighbor detected points and fit a straight line to them. Next we merge these small pieces for a long orbit. At the end we determine the common points of the trajectories.

Let us denote the set of measured points by V_p in the Euclidean space. The three-points-straight is accepted, if the fitting error is smaller then the value of ε_Y, ε_Z. These quantity depend on the experimental and theoretical considerations. The set S_{stp} contains the three-point-straight. This can be constructed by recursive using the set V_p with finite steps.

In the second part of the model we merge the short peaces for a long trajectories. In this case we have taking into account the curvature of the orbits and the distance between two different straights. The insertion was successfully, if the peaces were very close to each other. The time sequence of the measured points has strong consecutively, therefore the postfix and prefix map can be defined unique on this set.

During the hadronisation we needed to find the decay points to develop an algorithm, which can solve the original problem of this article.

Three types of decay point were introduced *Children case*, *Parent-child cases* and *Undetected parent*. We study, when two or more orbits create from the same points, it means the *Children case*. The *Parent-child case* continues the building *ChildStraights*, then we study the energy dominant particle case to

construct a more complicated tree graph. At the end we need take into account that situation, when a particle is not measured. In the experimental particle physics there are a few particle which we can not detected by this type of the calorimeter (i.e. γ). Then we apply that the energy is conservation during this process.

The experiment consists of a beam (direction of the shoot is parallel with Z axis). It interact with target due to result electromagnetic shower

In [1] a graph based trajectory reconstructing method was introduced. As all reconstruction algorithms, this is also very computation intensive, where even if the input dataset isn't very large, the combination of those inputs can generate a lot of work. As these ideas were proposed a couple of years ago, the originally used architecture for the computation may not be so efficient now. Back then a grid cluster was used to run the algorithm, but now the GPUs have outgrown the performance of smaller CPU clusters with their much more efficient design and seriously parallel architecture. Hence our current implementation involves CPUs, running the algorithm in parallel and GPUs, doing the computation in massively distributed manner, as they can effectively speed up the process as we already shown it in [5].

4.2 Implementation

The machine used for implementation has a GeForce GTX 980M with "computing capability" 5.2 [10] and an Intel Core i7-4710HQ CPU (Table 1).

4.2.1 CUDA memory hierarchy

The threads running on a CUDA capable GPU can access data from multiple memory spaces (Figure 2) [10]. Each thread has its own private local memory. The blocks containing the threads has access to a shared memory that is visible from all the contained threads. During the execution the whole set of threads launched in the grid have access to the same global memory. Additionally there are two read-only memory spaces, that can also be accessed by all the threads. These are the *constant* and *texture* memories. The global, constant, and texture memories are optimized for different usage scenarios.

	GeForce GTX 980M
Technical Specifications	Compute Capabiliting 5.2
Transistors (Million)	5200
Memory (GB)	4
Memory Bandwidth (GB/s)	160
GFLOPs	3189
TDP (watts)	125
	i7-4710HQ
Transistors (Million)	1400
Connected memory (GB)	24
Memory Bandwidth (GB/s)	25.6
GFLOPs	422
TDP (watts)	47

Table 1: GeForce GTX 980M and Core i7-4170HQ technical specifications

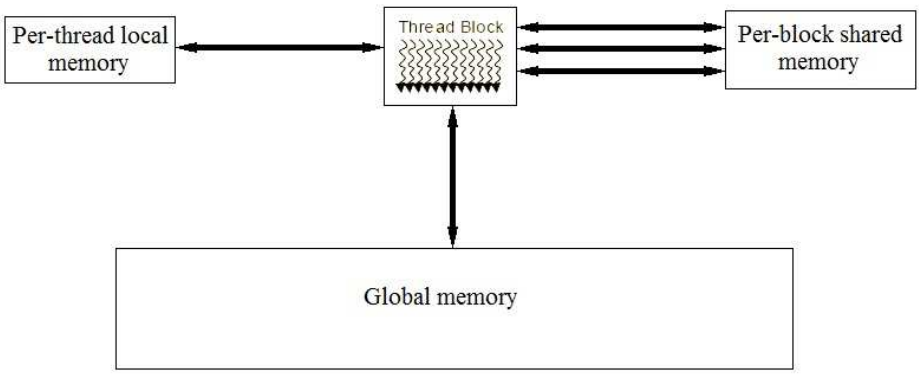

Figure 2: CUDA capable GPU's memory hierarchy

4.2.2 Design choices

From the memory hierarchy (Section 4.2.1) our solution uses the global, shared and local memory. For initialization we move the array storing the original detected points to the GPU's global memory. We process on this input to generate the required triplets, that will build up the trajectories. As for the triplets we have to match the points of three consecutive detector layers, we can map this to a 3 dimensional block to process on the GPU. As the maximum length in the Z coordinate can be only 64, we decided to set it to it's maximum value and align the rest accordingly. As one block can have 1024 threads maximum, we make our blocks to be $(x = 4, y = 4, z = 64)$ in size. Because we would like to check all points from one layer to all the others in the next and also on the third one, this process will generate a huge number of read operations (1024 comparisons by each block) on the device's global memory. While caching is available [10] on the GPU used for implementation, it is still time consuming to fetch all the data from the device memory. Hence at the beginning of the computation we further push the data from the global memory into shared memory, drastically decreasing the required time to proceed, considering it can be 100 times faster [10] compared to the global memory. This is because the global memory is on the card, while the shared memory and the registers are on the chip. As a result of this, the triplets will be the generated online and will be stored on the device. An example of block assignment is shown on Figure 3.

The triplets stored in device memory contains a pointer to the possible next part, the child element and there is also a pointer to the previous chunk, the parent element, basically making a linked list at the end. The process on the triplets is similar to the point matching done before in terms of block settings. In this case we only need a two dimensional block as we only need to check the triplets with each other, so we set the size of the block in the number threads to be $(x = 32, y = 32)$.

4.2.3 Algorithm

Following the stated principles in 4.2.1 and 4.2.2 the host side of the final algorithm for the triplet generation is in Figure 4.

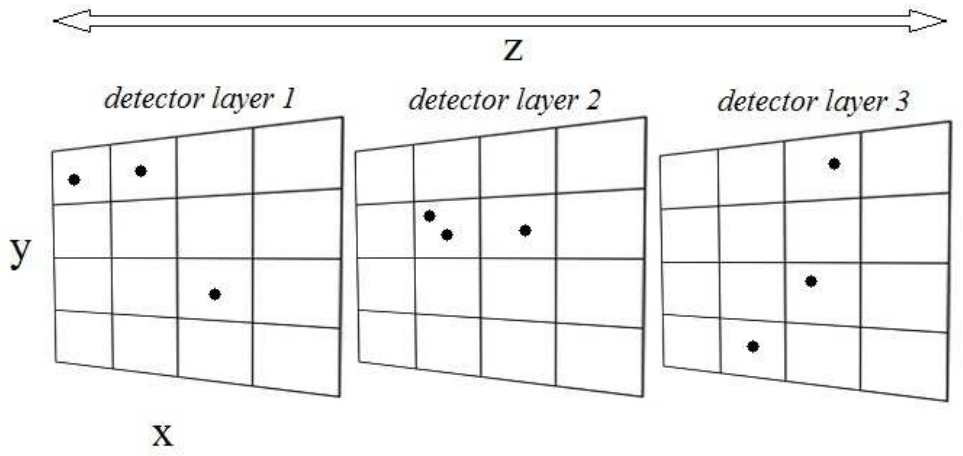

Figure 3: The detector layers as they are represented in a three dimensional CUDA block.

```
1:  procedure makeTriplets(P, L, NP)
2:      CudaMalloc(T)
3:      DT, DP, DNP ← CudaMemcpyToDevice(T, P, NP)
4:      start ← 0
5:      for each i ∈ 0..L − 2 do
6:          x, y, z ← NP[i]/4 + 1, NP[i + 1]/4 + 1, NP[i + 2]/64 + 1
7:          threads(4, 4, 64)
8:          blocks(x, y, z)
9:          makeTripletsKernel(DT, DP, DNP, i, start)
10:         start ← start + NP[i]
11:     end for
12: end procedure
```

Figure 4: The host part of the triplet generation, inputs are the points, the number of layers and the number of points per layers.

The implemented kernel function, which needs to be called from the host side to initiate the computations on the GPU is detailed in Figure 5. As this is the most time consuming kernel, it incorporates the shared memory to compute the triplets as fast as possible.

```
1: procedure makeTripletsKernel(DT, DP, DNP, i, start)
2:     j ← blockIdx.x ∗ blockDim.x + threadIdx.x
3:     k ← blockIdx.y ∗ blockDim.y + threadIdx.y
4:     l ← blockIdx.z ∗ blockDim.z + threadIdx.z
5:     SP ← Memcpy DP to shared memory
6:     if Line found on (SP[j], SP[k], SP[l]) then
7:         T[start + j] ← (SP[j], SP[k], SP[l])
8:     end if
9: end procedure
```

Figure 5: Device kernel to generate triplets, inputs are the array for the triplets, the detected points, the number of points per detector layer, the index of the first layer being tested and the index, where the triplets are starting in the array for the given layer.

The host side of the algorithm used to generated the lines in the trajectories can be found in Figure 6.

```
1: procedure makeTriplets(T, NT)
2:     x, y, z ← NT/32 + 1, x, 1
3:     threads(32, 32, 1)
4:     blocks(x, y, z)
5:     makeLinesKernel(T)
6: end procedure
```

Figure 6: The host side algorithm for the line generation, inputs are the triplets in the device memory and the number of them.

The device function required for the line generation on GPU is in Figure 7.

```
1: procedure makeLinesKernel(T)
2:     i ← blockIdx.x * blockDim.x + threadIdx.x
3:     j ← blockIdx.y * blockDim.y + threadIdx.y
4:     if T[j].next == NULL then
5:         if could fit line on (T[j], T[i]) then
6:             T[j].next ← T[i]
7:             return
8:         end if
9:     end if
10: end procedure
```

Figure 7: The device function to generate the lines, input is the array allocated in the device memory for the triplets.

4.3 Results

Because the problem at hand requires the comparison of all the possible combinations of the points in three consecutive layers and later the triplets are checked in a similar fashion, there is plenty of room for parallelization. Also the memory bound properties of the algorithm makes good use of the shared memory on the GPU. As we will see, the GPUs memory hierarchy (Section 4.2.1) makes them more suitable for these kind of applications. First, we take a look at the triplet generation using a CPU based parallel implementation and the GPU specific solution, comparing how they fair to each other. Then we do the same for the line generation.

The system used for development and testing is described in Table 2.

CPU	GPU	OS	Compiler	CUDA version
Intel Core i7 4710HQ	GeForce GTX 980M	Windows 10 Pro	Visual C++ 2013	7.0

Table 2: The test system

To evaluate the performance of our algorithms running on both CPU and GPU, we generated a dataset by simulating 2000, 4000, 8000, 16000 events using Geant4, running it with 100 MeV as the energy. The number of detected points were 19500, 38855, 77474, 154905 respectively (Figure 8). The simulated detector had 9 layers, giving us 3 batch of layers containing points to check for triplets.

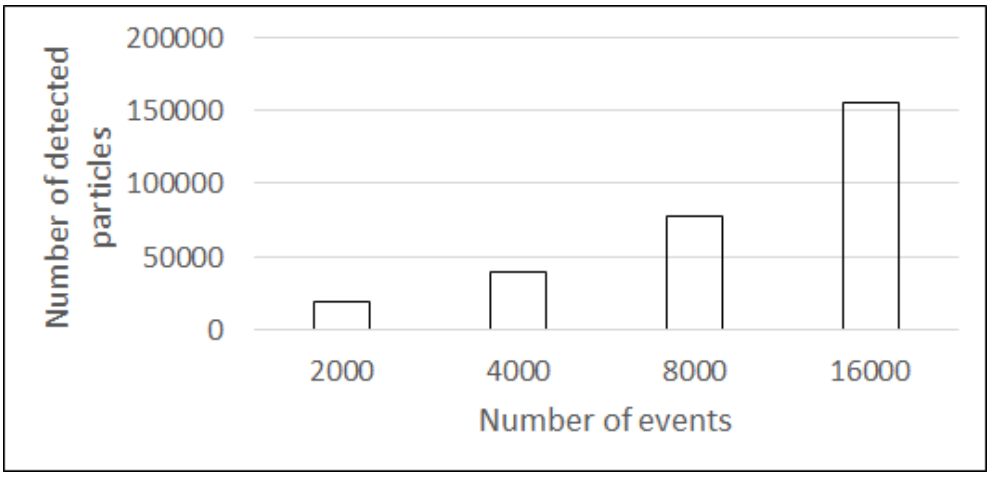

Figure 8: Number of detected points on the different number of events

Evaluating the same dataset on a version of the implementation, that does not use any of the shared memory on the GPU, the runtime is 5 times faster compared to the parallel CPU implementation. By changing the algorithm just slightly with moving the data to shared memory, we gain 27 times faster performance compared to the previous GPU results, which means compared to the CPU implementation the computation is up to 168 times faster in triplet generation and 223 times faster in line generation.

The runtime (Figure 9) on CPU was $\mathtt{timeT}_{\mathrm{cpu}} = 226.627\mathrm{s}$, the same computation took $\mathtt{timeT}_{\mathrm{gpu}} = 44.083\mathrm{s}$ on the GPU, while using the shared memory it was just $\mathtt{timeT}_{\mathrm{gpu_{sh}}} = 1.607\mathrm{s}$.

In the following we will keep using the GPU implementation using the shared memory. In Figure 10 we can see as we increase the number of iterations the difference between the CPU's and GPU's runtime is increasing, while the CPU can take hours in some cases the GPU runs only for minutes.

As we already saw the results for 2000 events, here we describe the numbers for the other ones. As such on 4000 events the runtime on CPU was $\mathtt{timeT}_{\mathrm{cpu}_{4000}} = 1817.95\mathrm{s}$, the same computation took $\mathtt{timeT}_{\mathrm{gpu}_{\mathrm{sh}_{4000}}} = 10.387\mathrm{s}$ on the GPU. On 8000 events the times were the following: on CPU we finished in $\mathtt{timeT}_{\mathrm{cpu}_{8000}} = 13647.3\mathrm{s}$, while on the GPU in $\mathtt{timeT}_{\mathrm{gpu}_{\mathrm{sh}_{8000}}} = 80.998\mathrm{s}$. For 16000 events on the CPU we could not finish in a reasonable time. The computation was running for more than 15 hours and it still couldn't finish. On the other hand the GPU could give back results in $\mathtt{timeT}_{\mathrm{gpu}_{\mathrm{sh}_{16000}}} = 644.446\mathrm{s}$.

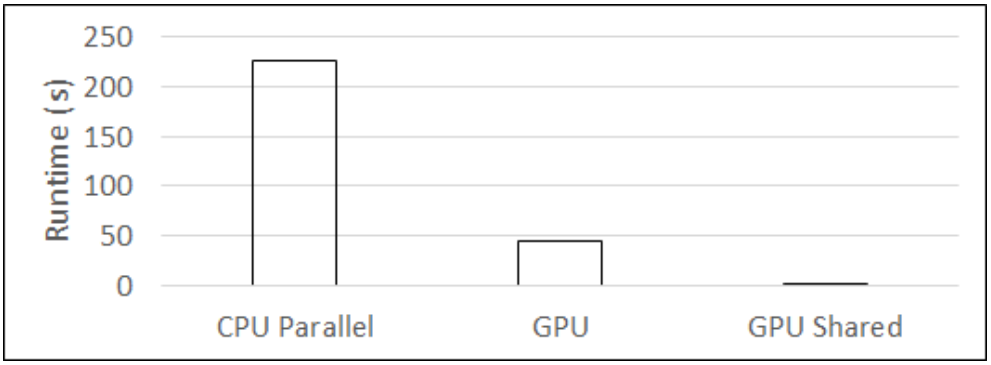

Figure 9: Triplet generation time on 2000 events

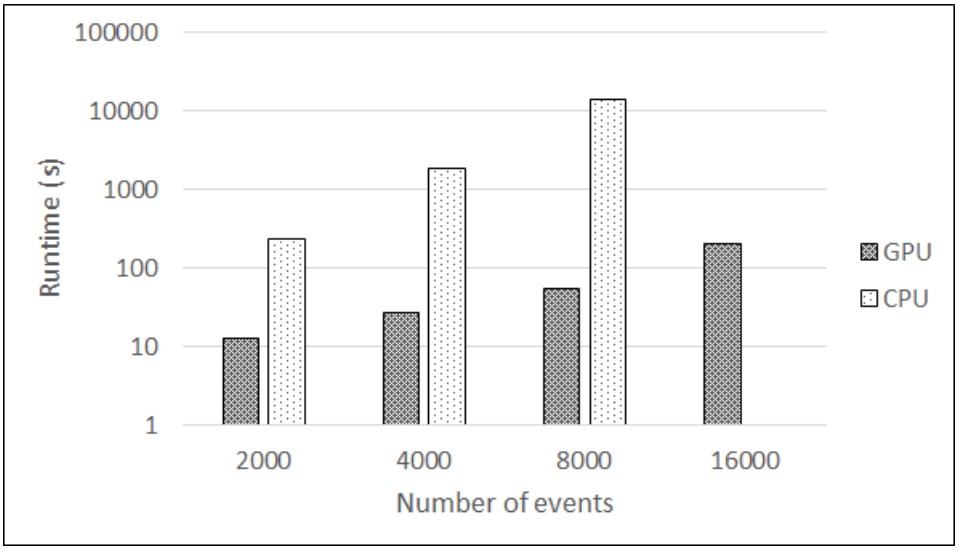

Figure 10: Triplet generation time on CPU and GPU

In Figure 11 we can see that the performance difference is also very clear in the generation of the lines. In this case while the CPU takes minutes to finish, the GPU can be done in seconds. Also here we have a runtime value for CPU under 16000 events, thanks to the lower number of combinations, that needs to be computed.

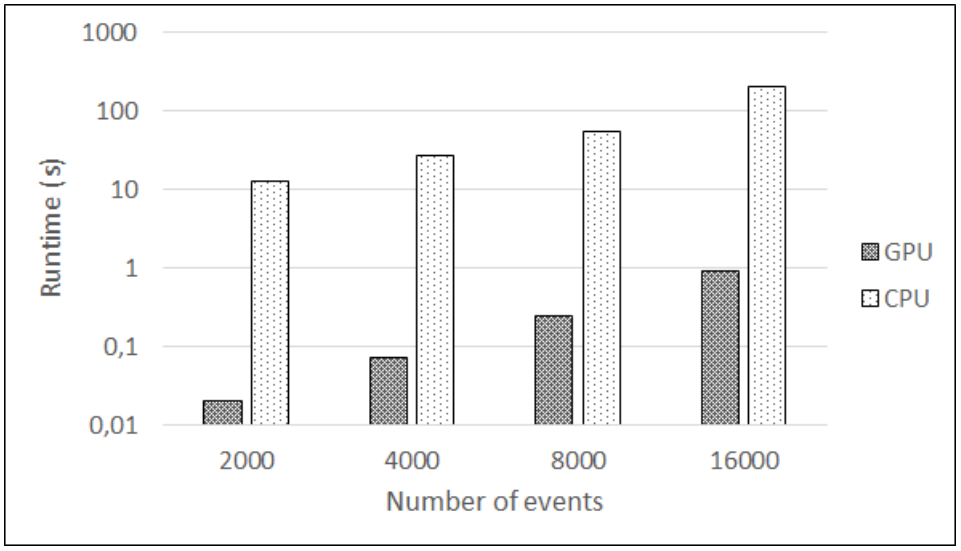

Figure 11: Line generation time on CPU and GPU

Taking a closer look on the Figure: on 2000 events, the runtime on the CPU is $\mathtt{timeL}_{\mathtt{cpu}_{2000}} = 12.685\mathtt{s}$ and on the GPU $\mathtt{timeL}_{\mathtt{gpu}_{2000}} = 0.02\mathtt{s}$. While on 4000 events the runtime on CPU was $\mathtt{timeL}_{\mathtt{cpu}_{4000}} = 26.027\mathtt{s}$, the same computation took $\mathtt{timeL}_{\mathtt{gpu}_{4000}} = 0.071\mathtt{s}$ on the GPU. On 8000 events the times were the following: on CPU we finished in $\mathtt{timeL}_{\mathtt{cpu}_{8000}} = 53.566\mathtt{s}$, while on the GPU in $\mathtt{timeL}_{\mathtt{gpu}_{8000}} = 0.247\mathtt{s}$. For 16000 events on the CPU we got $\mathtt{timeL}_{\mathtt{cpu}_{4000}} = 204.694\mathtt{s}$, while on the other hand the GPU could give back results in $\mathtt{timeL}_{\mathtt{gpu}_{16000}} = 0.916\mathtt{s}$.

One reconstructed event can be seen on Figure 12.

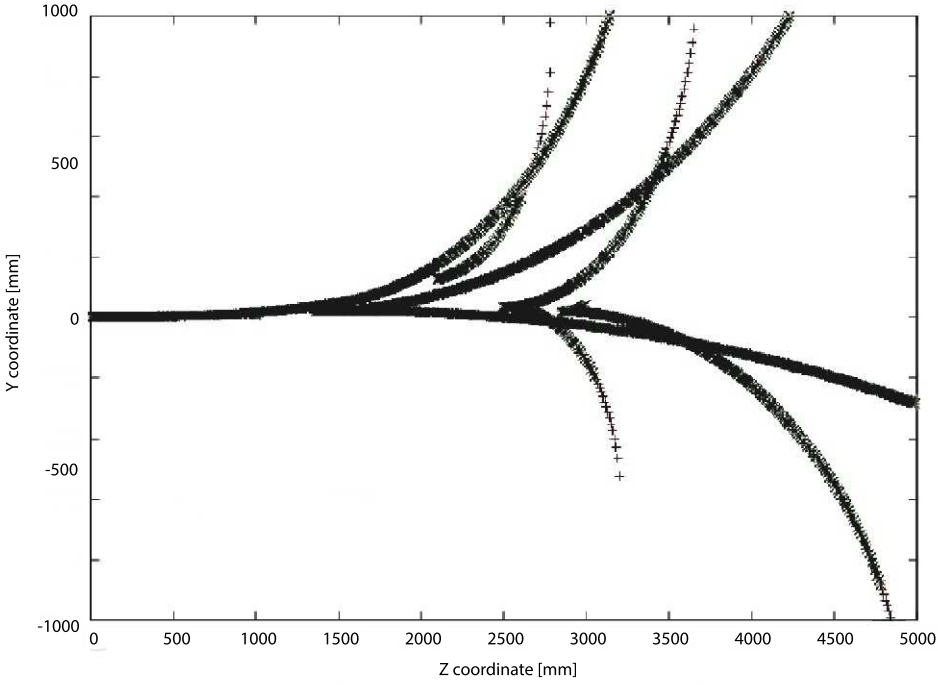

Figure 12: Reconstructed event on the Geant event. Black crosses with the noisy tail are the calculated points, while the full black parts are the Geant generated points.

5 Summary

As the GPUs are evolving, introducing new, more efficient architectures, it becomes easier to modify the existing applications and algorithms to be parallel. In this paper we were able to achive a 168 fold speed up compared to the CPU version, while computing the triplets of the trajectories. When calculating the full lines of the trajectories the system shows a 223 fold speed up in favor of the GPU.

In all cases the performance was definitely better on the GPU. On triplets $\mathtt{timeT}_{\mathtt{gpu}_{\mathtt{sh}_i}} << \mathtt{timeT}_{\mathtt{cpu}_i}$ and also for the lines $\mathtt{timeL}_{\mathtt{gpu}_i} << \mathtt{timeT}_{\mathtt{cpu}_i}, i \in [2000, 4000, 8000, 16000]$.

The performance of the GPUs make it possible to reconstruct a high volume of trajectories in parallel, finishing it in just a fraction on the runtime of the CPU.

References

[1] A. Agocs, Á. Fülöp, Jet reconstruction of individual orbits at many particles problems, *The 8th Joint Conference on Mathematics and Computer Science:MACS 2010*, Novadat Company, Komárno, Szlovákia 2011, pp. 123-138. ⇒172, 173, 174, 175

[2] R. Atkin, Review of jet reconstruction algorithms, *Journ. of Physics. Conf. Ser.* **645** (2015) 012008. ⇒173

[3] S. Catani, Yu.L. Dokshitzer, M. Olsson, G. Turnock and B.R. Webber, New clustering algorithm for multijet cross sections in e^+e^- annihilation, *Phys. Lett.*, **B269**, 3-4 (1991) 432-438. ⇒173

[4] S. D. Ellis, D. E. Soper, Successive combination jet algorithm for hadron collisions, *Phys. Rev. D* **48**, 7 (1993) 3160. ⇒173

[5] R. Forster, Á. Fülöp, Yang-Mills lattice on CUDA, *Acta Univ. Sapientiae, Inf.*, **5**, 2 (2013) 184–211. ⇒172, 174, 175

[6] M. E. Peskin, D. V. Schroeder, *Quantum Field Theory*, Westview Press, 1995. ⇒172

[7] S. Salur, Full Jet Reconstruction in Heavy Ion Collisions, *Nuclear Physics A* **830**, 1-4 (2009) 139c-146c. ⇒173

[8] T. Sjöstrand, S. Mrenna, P. Skands, A brief introduction to PYTHIA 8.1, *Computer Physics Communications* **178**, 11 (2008) 852-867. ⇒172

[9] Gy. Vesztergombi, Reflections about EXChALIBUR, the Exclusive 4π Detector, *Conf. "New Opportunities in the Physics Landscape at CERN"*, 2009. ⇒174

[10] CUDA C Programming Guide, NVIDIA Corp., 2016. ⇒172, 175, 177

[11] GEANT Detector Description and Simulation Tool, CERN Program Library Long Writeup, Geneva, 1993. ⇒172

Received: October 7, 2016 • Revised: November 9, 2016

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **8,** 2 (2016) 186–215

DOI: 10.1515/ausi-2016-0009

Monocular indoor localization techniques for smartphones

Gergely HOLLÓSI
Budapest University of Technology and Economics
email: hollosi@tmit.bme.hu

Csaba LUKOVSZKI
Budapest University of Technology and Economics
email: lukovszki@tmit.bme.hu

István MOLDOVÁN
Budapest University of Technology and Economics
email: moldovan@tmit.bme.hu

Sándor PLÓSZ
Budapest University of Technology and Economics
email: plosz@tmit.bme.hu

Frigyes HARASZTOS
Flaxcom Holding co. ltd.
email: harasztos.frigyes@flaxcom.hu

Abstract. In the last decade huge research work has been put to the indoor visual localization of personal smartphones. Considering the available sensor capabilities monocular odometry provides promising solution, even reflecting requirements of augmented reality applications. This paper is aimed to give an overview of state-of-the-art results regarding monocular visual localization. For this purpose essential basics of computer vision are presented and the most promising solutions are reviewed.

Computing Classification System 1998: F.1.1, G.1.3, I.2.10, I.4.1, I.4.8, I.4.9, K.8.1
Mathematics Subject Classification 2010: 68-02, 68U10, 68T45
Key words and phrases: computer vision, visual odometry, SLAM, indoor localization

1 Introduction

Due to the increasing capabilities and penetration, more and more applications are available on smart-phones and assist our everyday activities. In the last decade huge research work was put to the indoor location-based applications, from which the augmented reality based applications demand the highest requirements mostly expressed in accuracy, real-time and seamless localization. Based on the sensors that are available in recent smartphones and their computational and storage capabilities, a real-time implementation of monocular visual relative pose estimation seems to be a key to achieve the overall goal.

Besides, this topic presents great research interest, and high effort has been put on providing scalable and accurate solutions to satisfy the real-time requirements. Traditionally, the problem of visual pose estimation is discussed as Structure from Motion (SFM) [34] [17] problem, where the main goal is the off-line reconstruction of a 3D structure from pictures taken from different viewpoints. During the reconstruction process the viewpoints of the camera are also calculated, but problem formulation does not focus on relative pose estimation. The family of SLAM (Simultaneous Localization and Mapping) algorithms focuses on the environment modelling (map building) and relative camera pose estimation simultaneously [9]. To overcome the real time and accuracy requirements these solutions induced the PTAM (Parallel Tracking and Mapping) algorithm [22]. In the meantime, the problem also targeted by another application field, the odometry. The original requirement of the monocular Visual Odometry (VO) [50] [14] was to accurately determine the relative pose of a rover.

In this paper authors attempt to give a theoretical overview of the monocular odometry problem and its solutions. Also, some of the implementations are emphasized that seem to able to cope with the strict requirements even in mobile environments.

During the discussion, authors focus on capabilities of recent smartphones. Common smartphones are equipped with a thin-lens perspective camera, which can be modelled with an ideal pin-hole model [20], and they are also equipped with IMU (Inertial Measurement Unit) integrating gyroscope and accelerometer. Reasonable capacity for storage and processing. Regarding the motion of the device the following discussion suggests 6DOF (degree-of-freedom).

2 Theoretical background

Monocular visual odometry tries to determine pose and location of a device mostly using visual perception aided by couple of auxiliary sensors (e.g. gyro-

scope or acceleration sensor). The common implementation of visual perception is a monocular camera which provides continuous stream of frames at a variable or uniform time instants.

2.1 Projection model

The camera has a couple of internal parameters which are typically fixed and known a priori (e.g. by calibration). The most important characteristic of the camera is the projection model which projects three dimensional world points onto the image:

$$\mathbf{u} = \pi(\mathbf{p}_{\mathcal{C}}) \tag{1}$$

where $\mathbf{p}_{\mathcal{C}} = \left[x_{\mathcal{C}}, y_{\mathcal{C}}, z_{\mathcal{C}}\right]$ is a three dimensional world point in the reference frame of the camera, $\mathbf{u} = \left[x, y\right]$ is the projected point and $\pi()$ is the projection model. It is essential to mention that in case of monocular systems the $\pi()$ projection model is invertible only when the depth $d_{\mathbf{u}}$ of the model point is known:

$$\mathbf{p}_{\mathcal{C}} = \pi^{-1}(\mathbf{u}, d_{\mathbf{u}}) \tag{2}$$

We can see that monocular systems have the big drawback of loosing the depth information while recording frames.

In practice projection model is considered to be linear in homogeneous space , i.e. it can be represented by a matrix product (commonly referred to the pinhole camera model). Let $\mathbf{X}_{\mathcal{C}} = \left[X, Y, Z, 1\right]^T$ be the homogeneous coordinates of a three dimensional point in the reference frame of the camera. In this case the projection model can be expressed with a K intrinsic camera matrix:

$$\mathbf{x} = K(f)\left[I_{3\times3}|0_{3\times1}\right]\mathbf{X}_{\mathcal{C}} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\mathbf{X}_{\mathcal{C}} \tag{3}$$

where f is the focal length of the camera and $\mathbf{x} = \left[\lambda x, \lambda y, \lambda\right]^T$ are the homogeneous coordinates of the two dimensional projection. It is easy to see that the projection model is not invertible.

To represent camera movement in world frame we assign a T_k rigid-body transformation to each frame I_k at k time instants which contains orientation ($\mathbf{R}_k$) and location ($\mathbf{C}_k$) of the camera. The transformation can be expressed as a 4×4 matrix as

$$T_k = \begin{bmatrix} \mathbf{R}_k & \mathbf{C}_k \\ 0 & 1 \end{bmatrix} \tag{4}$$

A fixed world point $\mathbf{X} = \begin{bmatrix} X, Y, Z, 1 \end{bmatrix}$ can be projected at the k-th image frame as

$$\mathbf{x}_k = K(f) \begin{bmatrix} I|0 \end{bmatrix} T_k^{-1} \mathbf{X} = K(f) \begin{bmatrix} R_k^{-1} | - R_k^{-1} \mathbf{C}_k \end{bmatrix} \mathbf{X} = K(f) P_k^e \mathbf{X} \tag{5}$$

where P_k^e is commonly called as the extrinsic matrix describing the world-to-camera transformation. Eq. (5) is the most basic and substantial constraint in the monocular visual odometry systems.

The goal of the monocular visual odometry algorithms is to determine the P_k^e extrinsic camera matrices or the T_k rigid-body transformation of the cameras mainly based on (but not exclusively) the visual information encoded in frames.

2.2 Projection distortion

An accurate algorithm must take into consideration that the projection model of the classical pinhole camera is only an approximation. Real cameras always have some non-linear distortion which is basically modelled as *radial* distortion, however, other distortion models also exist (i.e. tangential distortion)[4]. Radial distortion depends on the radial distance from the radial distortion centre (typically the principal point) and it is represented as an arbitrary function:

$$\hat{x} = x_c + L(r)(x - x_c) \qquad \hat{y} = y_c + L(r)(y - y_c) \tag{6}$$

where $r^2 = (x - x_c)^2 + (y - y_c)^2$ is the radial distance and x_c, y_c are the radial centres (commonly considered as zero). In practice, $L(r)$ is represented as a Taylor-series

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \cdots \tag{7}$$

where κ_i are the radial distortion coefficients. In practice only the lower coefficients $(\kappa_1, \kappa_2, \kappa_3)$ are used.

2.3 Visual information retrieval

Visual odometry solutions are based on visual information encoded in the sequence of image frames. We can distinguish two widespread methods: intensity based *direct* methods and *feature* based methods.

2.3.1 Direct methods

In general, direct methods uses the $I_k(\mathbf{u})$ intensity map of the image, which represents the brightness of the image pixel coordinate or – rarely – the RGB

vector. The intensity map can be either quantized (i.e. pixel accuracy) or continuous (i.e. sub-pixel accuracy), however, the latter requires some kind of filtering or interpolating algorithm that in some cases can cause information loss.

2.3.2 Feature detection

Feature based methods works on point projections using *feature detection* and *feature extraction* algorithms that are able to detect and match the same points on different images without preliminary geometric knowledge. This way, visual odometry solutions are simplified to use only projections of real 3D landmarks. The efficiency of these algorithms can be measured by their invariance and speed. Invariance means that the detector can detect features which can be successfully matched even if the feature is rotated, scaled or suffered other transformations (e.g. affine transformation). There are a couple of such algorithms overviewed in [5], from which the most widely used are the Harris detector [18], the Scale-invariant feature transform (SIFT) that bases on Laplacian of Gaussian filters [36], the Maximally stable extremal regions (MSER) [37], the Features from accelerated segment test (FAST) and Oriented FAST and Rotated BRIEF (ORB) [48]. Considering the overall requirements SIFT is the most promising, however, due to its high complexity, strict constraints should be taken into account during its application in mobile environments.

3 Feature based solutions

Feature based solutions have the attribute to detect features on the frames first then match them to the previous frame resulting in projection *tracks* over a couple of sequential frames. These tracks can then be used to compute the geometry of the scene and to recover the camera translations and orientations. This method utilizes only point geometry models and correspondences, this way the well established framework of multiple view geometry can be applied [20].

3.1 Theory

The most important term here is pose estimation, which is the process of estimating the extrinsic (and sometimes the intrinsic) matrix from point correspondences. Depending on the point pairs we can establish two types of pose estimation: in case of 3D-2D point pairs (i.e. the world points and their pro-

jections) it is called *absolute pose estimation* and in case of 2D-2D point pairs (i.e. the projection pairs on two images) we call it *relative pose estimation.*

3.1.1 PnP problem

The absolute pose estimation problem is generally called Perspective-n-Point (PnP) problem, which has a couple of methods presented. The classical method for $n > 6$ point pairs is the DLT (Direct Linear Transform) method, but it is known to be unstable and requires the camera calibration [1]. For 5 or 4 points the [55] uses a polynomial technique which allows it to work well even in case of coplanar points. The EPnP solution is accurate for an arbitrary $n \geq 4$ point pairs and can handle planar and non-planar cases [26]. The P3P leads to to finite number of solutions using only three point pairs as the smallest subset of points pairs [24]. The P3P solution has the advantage of using only three points in a RANSAC framework to eliminate outlier point pairs decreasing the required number of iterations.

3.1.2 Random Sample Consensus

Since feature matching is prone to result false matches, a method is required to overcome this issue. It is common in image processing to use the minimal sample set to recover model parameters and classify samples to inliers and outliers. The most noted algorithm is the Random Sample Consensus (RANSAC) method, which is widely used in further solutions [12].

3.1.3 Relative pose estimation

The basic terms in relative pose estimation are the *fundamental matrix* and the *essential matrix*, both can be computed from 3D feature projection pairs. The fundamental matrix is a 3×3 matrix ($\mathbf{F}$) satisfying $\mathbf{x}'^{\mathsf{T}}\mathbf{F}\mathbf{x} = 0$, where projections ($\mathbf{x}$ and $\mathbf{x}'$) are of the same world point in two different images. The essential matrix ($\mathbf{E}$) uses normalized image coordinates, so it can be computed from the intrinsic camera matrix ($\mathbf{K}$) and the fundamental matrix as $\mathbf{E} = \mathbf{K}'^{\mathsf{T}}\mathbf{F}\mathbf{K}$. The essential matrix is applicable to recover the pose of the cameras by decomposition [20]. A lot of methods are known to determine the relative pose of the cameras: the 8 point algorithm [19], the 7 point algorithm [20], 6 point algorithm [45] and 5 point algorithms [27][42]. It is essential to mark that these algorithms differ in handling degenerate configurations (i.e. coplanar objects or cylinder containing the projection centres) and are unable to recover the scale of the set-up.

3.1.4 Bundle adjustment

The fundamental algorithms, like the relative and absolute pose estimation and triangulation, find the right solution only in case of noiseless measurements. Otherwise, they minimize the algebraic error which has no physical meaning. It can be proven that the maximum likelihood (ML) solution of these problems are the minimization of *re-projection error*. If we have N cameras and M points in space, then we can assign a $\theta_n(K_n, T_n, \pi_n)$ projection model to each camera that contains the projection (π_n), distortions (K_n) and rigid body transformation (T_n) of the camera (i.e. intrinsic and extrinsic behaviour). For a $\mathbf{p}_m$ point in space the projection for the camera n yields to $\mathbf{u}_{m,n} = \theta_n(\mathbf{p}_m)$. If pixel measurements are $\overline{\mathbf{u}}_{n,m}$, then the optimization of re-projection error equals the expression

$$\arg\min_{\theta_n, \mathbf{p}_m} \sum_{n,m} |\overline{\mathbf{u}}_{n,m} - \theta_n(\mathbf{p}_m)|^2 \tag{8}$$

meaning minimization of the euclidean-distance between the measurements and the re-projected points.

As it is obvious from Eq. (8), the re-projection error is not linear so we need an iterative Newton-like solution to solve the minimization problem. The process of solving Eq. (8) with Levenberg-Marquardt iteration is specially called *bundle adjustment* [56]. Bundle adjustment is widely used in SLAM, SFM and odometry problems to refine a coarse solution or co-optimize the map of landmarks and camera poses calculated before.

It is worth to mention that the special form of the projection equation yields to a sparse matrix, which can be utilized to speed up the bundle adjustment and relax the memory and processing requirements. This method is called *sparse bundle adjustment* [35][20].

3.2 Implementations

The solutions and implementations use the algorithms shown above but combine them in quite different ways.

3.2.1 PTAM

SLAM methods have the controversial problem of running at real-time speed while building an accurate map by a slow non-linear optimization process (i.e. bundle adjustment). Parallel Tracking and Mapping (PTAM) solves this problem by running two threads: one for the real-time tracking and one for the

map building [22]. PTAM was designed to work in small-scale, e.g. to provide desk-scale augmented reality. PTAM has several extensions implemented, like new initializer based on homography or a re-localiser [23].

PTAM detects FAST features on a scale pyramid to provide scale invariance and uses these feature points to recover the geometry. PTAM applies the 5-point algorithm to recover the initial camera relative pose (i.e. the fundamental matrix) and to construct the initial map. Hence, the process of PTAM odometry can be briefly described as follows:

- Tracking runs on its own thread and starts by detecting FAST features. A motion model is used to estimate the camera a-priori pose followed by projecting map points onto the image to detect feature matches and finally camera pose is refined from all the matches.

- The mapping thread selects key-frames at regular intervals based on a couple of conditions, then the thread triangulates new points and registers new projections. To refine the map, PTAM applies local and global bundle adjustments periodically.

PTAM solution is capable to track the camera pose accurately and in real-time thanks to the decoupled tracking and mapping processes, but its performance is limited by the number of landmarks registered in the map. This way PTAM is suitable only for small workspaces. One of the drawbacks of PTAM is the simple initialization process of the 5-point algorithm which is sensitive to planar degeneracy. It is worth to mention that PTAM does not employ any methods to recover the accumulated odometry error (i.e. loop closing).

3.2.2 ORB-SLAM

ORB-SLAM realizes a rather complex visual odometry solution, however, it is based basically on feature detection and point geometry [40]. As its name suggests it uses ORB features to gather image information and provides odometry and 3D reconstruction simultaneously. Besides, ORB-SLAM provides relocalization and loop closing capabilities in order to make the process more accurate.

ORB-SLAM works pretty much like PTAM by running three threads parallel to provide real-time odometry. The *tracking* thread is responsible for real-time motion estimation by detecting ORB features and camera pose recovery. The *local mapping* thread calculates the 3D reconstruction of the map in the background for every key-frame chosen by the tracking thread. The

loop closing thread is watching for map points to reoccur using bag of words model, and when it finds one the loop closing corrects the loop by similarity transformation.

ORB-SLAM applies ORB feature detection as it provides rotation and scale invariance. It is fast enough to maintain real-time performance, while it is suitable for both large-scale (i.e. distant frames) and small-scale (i.e. subsequent frames) matching. The great innovation in ORB SLAM is that it uses ORB for every part of the process: tracking, mapping and loop closing are executed on ORB features. ORB-SLAM system provides visual odometry as follows:

1. The ORB-SLAM starts with an automatic initialization method to retrieve the initial pose and map by extracting the ORB features, matching them and computing corresponding fundamental matrix and homography (i.e. the two dimensional projective transformation) in the same time. It computes a score to both the homography and the fundamental matrix as:

$$S_M = \sum_i \left(\rho_M\left(d_{cr}^2(\mathbf{x}_c^i, \mathbf{x}_r^i, M)\right) + \rho_M(d_{rc}^2(\mathbf{x}_c^i, \mathbf{x}_r^i, M))\right)$$
$$\rho_M(d^2) = \begin{cases} \Gamma - d^2 & \text{if} \quad d^2 < T_M \\ 0 & \text{if} \quad d^2 \geq T_M \end{cases} \tag{9}$$

   where M is the model (H for homography and F for fundamental matrix), d_{cr}^2 and d_{rc}^2 are the symmetric transfer errors, T_M is the outlier rejection threshold based on the χ^2 test at 95% ($T_H = 5.99$, $T_F = 3.84$, assuming a standard deviation of 1 pixel in the measurement error). Γ is a score compensating constant. ORB-SLAM recover initial pose and map from homography if

$$\frac{S_H}{S_H + S_F} > 0.45 \tag{10}$$

   Otherwise, it uses the fundamental matrix. After recovering pose and map, it starts a non-linear optimization (bundle adjustment) to refine the initial model.

2. After map initialization tracking tries to match ORB features of the current frame to the ORB features of the previous frame through a guided search employing a constant velocity model. The pose is then refined by non-linear optimization. After pose estimation ORB-SLAM tries to re-project the map onto the frame recovering more feature matches. The

last step is the key-frame decision which judges that the current frame should be passed to the local mapping thread. This step utilizes a couple of complex conditions.

3. Parallel to tracking, every key-frame is processed to provide a consistent map that is able to refine the tracking process and provides input to loop closing. Briefly, local mapping triangulates new point candidates having passed a restrictive map point culling test and uses local bundle adjustment to minimize re-projection error. To maintain compact reconstruction ORB-SLAM removes redundant key-frames

4. Loop closing happens parallel to tracking and mapping and uses bag of words representation and co-visibility information to detect loop candidates [43]. In case of loop detection it computes the similarity transformation accumulated while tracking to distributes the error along the whole path.

ORB-SLAM has been proven to be a robust and accurate solution even in large-scale areas and can successfully track ad-hoc movements while providing stable map initialization in case of a lost track. ORB-SLAM requires at least 20 frames per second to work well, which can hardly be satisfied using ORB feature detection on embedded devices like smartphones without exploiting massive GPU calculations.

4 Direct solutions

The principle behind direct solutions states that using the image intensities results in better odometry accuracy because it exploits all the information embedded in the frames while feature based solutions discard image information over feature points. The most important term of direct solutions is the *photo-consistency* discussed in the next section.

4.1 Photo-consistency theory

From a mathematical perspective, photo-consistency means that given two images I_1 and I_2, an observed point $\mathbf{p}$ by the two cameras yields to the same brightness in both images [21]:

$$I_1(\mathbf{u}) = I_2(\tau(\xi, \mathbf{u})) \tag{11}$$

where $\mathbf{u}$ is the projection of $\mathbf{p}$, $\tau(\cdot)$ is the *warping* function, which depends on $\pi(\cdot)$ (see Eq. (1)). The warping function maps a pixel coordinate from the first image to the second one given the camera motion ξ. Here, the motion ξ can be represented in any minimal representation (e.g. twist coordinates). Given the residual function for any $\mathbf{u}$ point in the Ω image domain

$$r(\xi, \mathbf{u}) = I_2(\tau(\xi, \mathbf{u})) - I_1(\mathbf{u}) \tag{12}$$

which depends on ξ and assuming independent pixel noise, the Maximum Likelihood (ML) solution is a classical minimization problem:

$$\xi_{ML} = \arg\min_{\xi} \int_{\Omega} r^2(\xi, \mathbf{u}) \, d\mathbf{u} \tag{13}$$

The problem is obviously non-linear, so the common solution is to run iterative minimization algorithms like Newton-Gauss method over a discretised image. To speed up the integration process, the integration can be run over a couple of selected patches instead of every pixels in the images.

4.2 DTAM

Dense Tracking and Mapping (DTAM) uses the photo-consistency theory in a special way to provide dense maps and real-time visual odometry [41]. The main idea behind dense mapping is to sum the photometric error along a ray from the camera centre and find the d distance that minimizes the sum, thus finding the depth parameter for that pixel. The summing is made along a couple of short baseline frames $m \in I(r)$ for a r reference frame:

$$C_r(\mathbf{u}, d) = \frac{1}{|I(r)|} \sum_{m \in I(r)} \|r_r(I_m, \mathbf{u}, d)\|_1 \tag{14}$$

where $\|\cdot\|_1$ is the L1 norm and the photometric error is

$$r_r(I_m, \mathbf{u}, d) = I_m(\tau(d, \mathbf{u}_i)) - I_r(\mathbf{u}_i) \tag{15}$$

Note that the only change in the equation is the parameter d. DTAM showed that minimizing the cost yields to a correct estimation of pixel depth which can be used to build dense maps.

The tracking part of the DTAM solution provides 6DOF estimation and basically happens the same way as shown in Eq. (13) with a couple of extensions to provide robust tracking with occlusion detection.

DTAM is robust and accurate visual odometry solution with excellent mapping capabilities. It is not only capable of handling occlusions but can track the movements even in case of total lost in focus and keep on tracking even for fast and random movements. The only drawback of the solution is real-time performance requires huge computing capacity and massive GPU utilization.

4.3 LSD-SLAM

Large-Scale Direct Monocular SLAM (LSD-SLAM) uses direct methods combined with a probabilistic approach to track camera movements and build dense map real-time [10]. LSD-SLAM has scale-aware image alignment algorithm which estimate directly the similarity transformation between two key-frames to provide scale consistent maps and odometry.

The main process of the LSD-SLAM is as follows: at every new frame it tries to estimate the movement relative to the current key-frame, then it decides whether the actual key-frame should be replaced by the new frame. In case of replacing, it initializes a new depth map, otherwise it propagates the depth map of the current key-frame. At every key-frame replacement LSD-SLAM runs map optimization, which is essential to create accurate dense maps.

LSD-SLAM uses image patches to recover pose around pixels with large intensity gradients. The tracking process is composed of two steps: estimation of rigid body transformation and depth map propagation. The former one is a weighted optimization of the variance-normalized photometric error

$$E_p(\xi_j) = \sum_{p \in \omega_{D_i}} \left\| \frac{r_p^2(\mathbf{u}, \xi_j)}{\sigma^2_{r_p(\mathbf{u},\xi_j)}} \right\|_\delta \tag{16}$$

for an existing key-frame and the new frame I_j. In the equation $r_p(\cdot)$ is the photometric error, σ_{r_p} is the variance of the photometric error and $\|\cdot\|_\delta$ means the Huber-norm. Apart from normalization by variance, this is a classical photometric error based odometry solution as in Eq. (13).

The biggest difference to other direct solutions is that the depth information for a key-frame is calculated in a probabilistic way, i.e. it is refined as new frames received. An inverse depth map and a depth map variance map is assigned to every key-frame selected by the LSD-SLAM process. The depth map is initialized with the depth map of the previous key-frame or with a random depth map if no key-frame exists. For each new frame the depth map is propagated as in [11], namely if the inverse depth for a pixel was d_0 then

for the new frame it is approximated as

$$\begin{aligned} d_1 &= (d_0^{-1} - t_z)^{-}1 \\ \sigma_{d_1}^2 &= \left(\frac{d_1}{d_0}\right)^4 \sigma_{d_0}^2 + \sigma_p^2 \end{aligned} \tag{17}$$

where σ_p is the prediction uncertainty and t_z is the camera translation along the optical axis.

LSD-SLAM also contains solution for the problem of scale-drift over long trajectories, which is the major source of error in the family of SLAM solutions. LSD-SLAM thus aligns two differently scaled key-frames by incorporating the depth residual into the error function shown above. This method penalizes deviations in inverse depth between key-frames and helps to estimate the scaled transformation between them.

4.4 SVO

Fast Semi-Direct Monocular Odometry (SVO) is a great example of hybrid solutions for visual odometry using direct and featured based algorithms as well [13]. SVO combines the probabilistic approach of depth map with the computationally attractive feature based concept as the name suggests providing real-time odometry and sparse mapping.

The basic process of SVO is tracking and mapping that are implemented on parallel threads, i.e. calculating the movement trajectory at each frame real-time and select key-frames that can be used for mapping on the mapping thread. As the mapping thread uses features, bundle adjustment can be used to minimize re-projection error and construct accurate maps.

The tracking thread projects 3D points of the map onto the new frame and uses the vicinity of the projected points in the image to estimate the motion relative the previous frame by photometric error optimization. The pose is refined by aligning the frame to the whole map (using Lucas-Kanade algorithm [3]) then by local bundle adjustment to apply the epipolar constraints.

SVO is unique in the way that no depth map is computed but for each feature point on a key-frame a depth-filter is assigned that estimates the feature depth in a probabilistic way. First, the mapping thread decides whether a new frame a key-frame or not. Feature extraction is executed on new key-frames and to each feature a freshly initialized depth-filter is assigned. On inter-frames (i.e. not key-frames) the feature depth-filters are updated until they converge to the estimated value and the variance is small enough. Converged depth filter are converted to map points by triangulation.

Thanks to the feature based mapping process SVO has proven to be faster than other direct solutions, however, the result is a sparse map rather than a dense one. The depth filters are capable of detecting outlier measurement and the map is always consistent and correct because triangulation happens only when the filters converged. As the SVO uses couple of small patches around features to estimate motion, it is capable of running real-time as well.

5 Filter-based solutions

In real applications relative pose estimation should be seamless, which cannot be guaranteed by solutions based only on image processing. To overcome this requirement motion models are applied to estimate the camera state between visual pose estimations. The first reliable solution is MonoSLAM [8], which introduces an extrapolated motion model. MonoSLAM is thus applicable for smooth camera motion, but can cover only desk-scale local environment.

The most reasonable choice for motion estimation is to combine measurements of IMU with projections of real 3D landmarks of the local environment. The filter based family of visual odometry algorithms fuses inertial IMU measurements with visual feature observations. In these methods, the current camera pose and positions of visual landmarks are jointly estimated, sharing the same basic principles with camera-only localizations. These combined techniques can be categorized as *loosely coupled* and *tightly coupled* systems. In loosely coupled systems [46] [54] [57] inertial and camera measurements are processed separately before being fused as a single relative pose estimate, while tightly coupled systems process all the information together [44] [25]. However, loosely coupled systems limits computational complexity, in the following we are focused on tightly coupled techniques due to its ability to reach higher consistency between camera poses and map of landmarks.

5.1 Theory

The original relative pose estimation problem is hard due to its nature. The algorithms use a map containing visual information to localize, while relative pose is necessary to construct and update the visual map. The problem becomes even harder to solve if we consider the noise of the sensors. Various probabilistic methods are used to deal with the uncertainty introduced by measurement noise, Extended Kalman Filter (EKF), Particle Filter (PF), which are all based on Bayesian technique for random value estimation of the system state parameters, including camera locations and orienta-

tions at discrete time instants ($\mathbf{x}_k$) based on observations of the landmarks ($\mathbf{z}_k = \{\mathbf{z}_{ik}\}$) from a given camera viewpoint, in other words the map points ($\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n\} = \mathbf{m}_{1:n}$), while the camera location is controlled independently of the system state by control parameters ($\mathbf{u}_k$). The problem of relative pose estimation is given then in the probabilistic form as follows [9].

$$\mathsf{P}(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0) \tag{18}$$

The calculation of position probability distribution is done iteratively starting from $\mathsf{P}(\mathbf{x}_{k-1}, \mathbf{m} | \mathbf{z}_{1:k-1}, \mathbf{u}_{1:k-1}, \mathbf{x}_0)$ with input of the actual control $\mathbf{u}_k$ and measurement $\mathbf{z}_k$ using Bayesian Theorem. The computation from one side requires the *state transition* or *motion model* for the camera that describes the new state regarding the control input.

$$\mathsf{P}(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k) \tag{19}$$

Secondly, the *observation model* describes the probability of making and observation $\mathbf{z}_k$ when a camera and landmark locations are known.

$$\mathsf{P}(\mathbf{z}_k | \mathbf{x}_k, \mathbf{m}) \tag{20}$$

The iteration is then implemented in a standard two-step recursive process. The first step is the *time update* that *propagates* state in time according to the control.

$$\mathsf{P}(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_{0:k-1}, \mathbf{u}_{0:k}, \mathbf{x}_0) = \int \mathsf{P}(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k) \cdot \mathsf{P}(\mathbf{x}_{k-1}, \mathbf{m} | \mathbf{z}_{0:k-1}, \mathbf{u}_{0:k-1}, \mathbf{x}_0) \, \mathrm{d}\mathbf{x}_{k-1} \tag{21}$$

The second step conveys the *measurement* or *update* when based on the actual state-dependent measurements *correction* is done on the actual state.

$$\mathsf{P}(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0) = \frac{\mathsf{P}(\mathbf{z}_k | \mathbf{x}_k, \mathbf{m}) \mathsf{P}(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_{0:k-1}, \mathbf{u}_{0:k}, \mathbf{x}_0)}{\mathsf{P}(\mathbf{z}_k | \mathbf{z}_{0:k-1}, \mathbf{u}_{0:k})} \tag{22}$$

The above principle is implemented in various ways assuming different terms on the model and random value distributions.

5.2 The IMU model

In practice, gyroscope and accelerometer measurements can be used to estimate actual relative pose based on the kinematic model. This is done during

the timely filter state propagation. All these measurements are stressed with local measurement noise, distortion and biases. The accelerometer measures actual acceleration ($\mathbf{a}_{m,\mathcal{I}} \in \mathbb{R}^3$) in the IMU orientation frame ($\mathcal{I}$) and its model can be formulated as follows.

$$\mathbf{a}_{m,\mathcal{I}}(t) = \mathbf{T}_a \mathbf{R}_{\mathcal{I}\mathcal{G}}(t)(\mathbf{a}_{\mathcal{G}}(t) - \mathbf{g}) + \mathbf{a}_b(t) + \mathbf{a}_n(t) \tag{23}$$

where $\mathbf{a}_{\mathcal{G}}$ is the real acceleration in global orientation frame, $\mathbf{g}$ is gravitational acceleration. $\mathbf{R}_{\mathcal{I}\mathcal{G}}$ represents rotational transformation between IMU frame ($\mathcal{I}$) and global frame ($\mathcal{G}$), while $\mathbf{T}_a$ shape matrix comprises gyroscope axis misalignments and scale errors of bases. The measurement noise $\mathbf{a}_n$ is modelled as a zero mean Gaussian random variable, $\mathbf{a}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{N}_a)$, and the bias $\mathbf{a}_b$ changes over the time and is modelled as a random walk process driven by its own noise vector $\mathbf{a}_{wn} \sim \mathcal{N}(\mathbf{0}, \mathbf{N}_{wa})$.

Regarding gyroscope, it measures rotational velocity ($\boldsymbol{\omega}_{m,\mathcal{I}} \in \mathbb{R}^3$) in IMU orientation frame, its realistic model can be figured out as below.

$$\boldsymbol{\omega}_{m,\mathcal{I}}(t) = \mathbf{T}_g \boldsymbol{\omega}_{\mathcal{I}}(t) + \mathbf{T}_s \mathbf{a}_{\mathcal{I}}(t) + \boldsymbol{\omega}_b(t) + \boldsymbol{\omega}_n(t) \tag{24}$$

where $\boldsymbol{\omega}_{\mathcal{I}}$ is the real rotational velocity in IMU orientation frame, $\mathbf{T}_g$ is the shape matrix, while $\mathbf{T}_s \mathbf{a}_{\mathcal{I}}$ represents influence of acceleration upon rotational velocity.

In practice, due to their insignificant effects scale, misalignment and acceleration influence is considered idealistic ($\mathbf{T}_a = \mathbf{T}_g = \mathbf{I}, \mathbf{T}_s = \mathbf{0}$). Most of the real implementations assume biases and noises during modelling.

5.3 Extended Kalman Filter (EKF)

The Bayesian technique can be solved by EKF, which is applicable for non-linear systems, while noise is considered as Gaussian. In EKF, the motion or state transition model Eq. (19) is formalized by the following equation.

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{25}$$

where $\mathbf{f}$ function models vehicle kinematics in function of actual state $\mathbf{x}_{k-1}$ and actual control input $\mathbf{u}_k$ and $\mathbf{w}_k$ is an additive zero mean Gaussian noise with covariance $\mathbf{Q}_k$ ($\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$).

On the other side, EKF implements the generic observation model Eq. (20) by the following equation.

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{26}$$

where $\mathbf{h}$ function describes relation between actual state $\mathbf{x}_k$ and state-dependent measurements $\mathbf{z}_k$. The $\mathbf{v}_k$ is again an additive zero mean Gaussian error of observation with covariance $\mathbf{R}$ ($\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$).

The system state vector of filter-based visual odometry solutions can be divided into the part related to the motion estimation ($\mathbf{x}_{base}$) and the auxiliary section related to the observation model related to the certain solution ($\mathbf{x}_{aux}$). The base part $\mathbf{x}_{base}$ comprises parameters necessary to describe kinetic and dynamic state it also contains parameters necessary for modelling gyroscope and accelerometer. The auxiliary part $\mathbf{x}_{aux}$, in visual based systems, is necessary to describe the visual observation model. In real implementations it can contain real 3D positions or projected positions of map landmarks ($\mathbf{m}$) or even consecutive camera positions and orientations.

$$\mathbf{x}_k = [\mathbf{x}_{base,k}, \mathbf{x}_{aux,k}] \tag{27}$$

The related state covariance matrix ($\mathbf{P}$) is also can be divided into parts related to the motion model ($\mathbf{P}_{base}$), to the observation model ($\mathbf{P}_{aux}$), and describes the relation between these parameters ($\mathbf{P}_{base,aux}$).

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{P}_{base,k} & \mathbf{P}_{base,aux,k} \\ \mathbf{P}^T_{base,aux,k} & \mathbf{P}_{aux,k} \end{bmatrix} \tag{28}$$

During time update the state vector estimate and related covariance matrix are updated according to the following equations.

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}^T_k + \mathbf{Q}_k \end{aligned} \tag{29}$$

where the $\mathbf{F}$ is the Jacobian of $\mathbf{f}$ function and evaluated around at the state estimate $\hat{\mathbf{x}}_k$ and actual control input $\mathbf{u}_k$, $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_k, \mathbf{u}_k)$.

Based on the visual observations, correction is formulated according to following equations that describe the residual, the Kalman gain, respectively.

$$\begin{aligned} \mathbf{r}_k &= \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}^T_k (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}^T_k + \mathbf{R}_k)^{-1} \end{aligned} \tag{30}$$

According to the residual and the Kalman gain estimated state and covariance matrix updates are defined as the followings.

$$\begin{aligned} \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{r}_k \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \end{aligned} \tag{31}$$

Although, due to its flexibility and moderate complexity, EKF is the most widely used Bayesian filter, during its application some of the drawbacks have to be considered. Since EKF linearises the actual non-linear characteristics, the choice of point of linearisation affects its stability, while special care has to be taken to the estimation of noise variances.

Considering 6DOF kinematic properties of the smart-phone, application requires from the filter state to store actual orientation, position, velocity and gyroscope and accelerometer bias parameters, at least. According to this consideration, kinematic part of the filter state is defined by the following vector.

$$\mathbf{x} = [\mathbf{q}_{\mathcal{GI}}, \mathbf{p}_{\mathcal{I},\mathcal{G}}, \mathbf{v}_{\mathcal{I},\mathcal{G}}, \boldsymbol{\omega}_b, \mathbf{a}_b]^T \tag{32}$$

During state propagation using the gyroscope-accelerometer measurement pair, the nominal values of kinetic part of the state should follow the kinetic equations below.

$$\begin{aligned} \dot{\mathbf{q}}_{\mathcal{GI}} &= \frac{1}{2}\mathbf{q}_{\mathcal{GI}} \otimes (\boldsymbol{\omega}_m - \boldsymbol{\omega}_b), \quad \dot{\mathbf{p}}_{\mathcal{I},\mathcal{G}} = \mathbf{v}_{\mathcal{I},\mathcal{G}}, \\ \dot{\mathbf{v}}_{\mathcal{I},\mathcal{G}} &= \mathbf{R}_{\mathcal{GI}}(\mathbf{a}_m - \mathbf{a}_b) + \mathbf{g}, \quad \dot{\boldsymbol{\omega}}_b = \mathbf{0}, \quad \dot{\mathbf{a}}_b = \mathbf{0} \end{aligned} \tag{33}$$

5.4 Particle filter

The Bayesian propagation and measurement equations (see Eq. (22) and Eq. (21)) cannot be solved in closed form for the SLAM problem. For Gaussian-distribution the solution can be approximated with various Kalman-filters but the exact solution for strongly non-linear models can only be found by numerical integration.

Given a $\mathbf{g}(\mathbf{x}) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ function, the expectation over a posterior distribution:

$$E[\mathbf{g}(\mathbf{x})|\mathbf{z}_{1:k}] = \int \mathbf{g}(\mathbf{x}) P(\mathbf{x}|\mathbf{z}_{1:k}) \, d\mathbf{x} \tag{34}$$

can be approximated by drawing N independent random samples $\mathbf{x}^{(i)}$ form the $p(\mathbf{x}|\mathbf{z}_{1:k})$ distribution:

$$E[\mathbf{g}(\mathbf{x})|\mathbf{z}_{1:k}] \approx \frac{1}{N}\sum_{i=1}^{N} \mathbf{g}(\mathbf{x}^{(i)}) \tag{35}$$

This type of numerical calculation of integrals is called Monte Carlo method [49]. However, in case of the Bayesian models it is not possible to draw samples from $P(\mathbf{x}|\mathbf{z}_{1:k})$, so we need to use importance sampling in order to approximate the distribution.

5.4.1 Importance sampling

To overcome the issue of not having the $P(\mathbf{x}|\mathbf{z}_{1:k})$ distribution, we can construct an importance distribution $\Pi(\mathbf{x}|\mathbf{z}_{1:k})$ from which it is easy to draw samples [33]. It is straightforward that

$$\int \mathbf{g}(\mathbf{x})P(\mathbf{x}|\mathbf{z}_{1:k})\ d\mathbf{x} = \int \left[\mathbf{g}(\mathbf{x})\frac{P(\mathbf{x}|\mathbf{z}_{1:k})}{\Pi(\mathbf{x}|\mathbf{z}_{1:k})}\right]\Pi(\mathbf{x}|\mathbf{z}_{1:k})\ d\mathbf{x} \tag{36}$$

By using this form we can establish a Monte Carlo approximation of the expectation by drawing samples from the importance distribution as $\mathbf{x}^{(i)} \sim \Pi(\mathbf{x}|\mathbf{z}_{1:k})$ and calculating the sum

$$\begin{aligned} E[\mathbf{g}(\mathbf{x})|\mathbf{z}_{1:k}] &\approx \frac{1}{N}\sum_{i=1}^{N}\frac{P(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})}{\Pi(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})}\mathbf{g}(\mathbf{x}^{(i)}) \\ &= \sum_{i=1}^{N}\tilde{w}^{(i)}\mathbf{g}(\mathbf{x}^{(i)}) \end{aligned} \tag{37}$$

where $\tilde{w}^{(i)}$ is defined as

$$\tilde{w}^{(i)} = \frac{1}{N}\frac{P(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})}{\Pi(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})} \tag{38}$$

By using normalized weights and applying Bayes-rule we can replace the posterior distribution $P(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})$ with the prior and the likelihood function. The normalized weights $w^{(i)}$ can be written as

$$\begin{aligned} w^{*(i)} &= \frac{P(\mathbf{z}_{1:k}|\mathbf{x}^{(i)})P(\mathbf{x}^{(i)})}{\Pi(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})} \\ w^{(i)} &= \frac{w^{*(i)}}{\sum_{j=1}^{N} w^{*(j)}} \end{aligned} \tag{39}$$

Finally, the posterior probability density function can be formed by using the Dirac delta function $\delta(\cdot)$:

$$p(\mathbf{x}|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w^{(i)}\delta(\mathbf{x} - \mathbf{x}^{(i)}) \tag{40}$$

5.4.2 Sequential importance sampling

Specifically, for state space models shown in Eq. (19) and Eq. (20), the modified version of the importance sampling algorithm can be used to calculate the expectation and probability density efficiently. The sequential importance sampling algorithm uses a weighted set of particles $\{(w_k^{(i)}, \mathbf{x}_k^{(i)})\}$, which contains samples from an importance distribution and their weights for every k time instant. The distribution can be approximated with the particles as

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}) \tag{41}$$

The weights can be calculated at every time instant with the equation

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{P(\mathbf{z}_k|\mathbf{x}_k^{(i)}) P(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{\Pi(\mathbf{x}_k^{(i)}|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k})} \tag{42}$$

where $w_k^{(i)}$ shall be normalized to sum to unity [49].

Sequential importance sampling still has the problem of having too many particles with almost zero weights, thanks to the special properties of the distributions used in SLAM techniques. This situation is called degeneracy problem and has a great impact on using them in real applications. To avoid degeneracy problem a re-sampling method called *sequential importance re-sampling* is used [16]. The main idea behind re-sampling is to draw new samples from the discrete distribution defined by the weights and use them as new particles. This way particles with relevant weights get duplicated and particles with small weights get removed. Commonly, the sequential importance re-sampling algorithm is referred to as *particle filter*.

5.5 Solutions

5.5.1 EKF-SLAM

In EKF-SLAM algorithms filter state vector contains current IMU state $\mathbf{x}_{base}$ and the observed feature 3D positions ($\mathbf{p}_{f_i}$). Thus the filter state vector is defined as follows.

$$\mathbf{x}_k = [\mathbf{x}_{base,k}, \mathbf{p}_{f_{1,k}}^T ... \mathbf{p}_{f_{n,k}}^T]^T \tag{43}$$

The 3D feature can be parametrized traditionally using (x, y, z) coordinates, anchored homogeneous parametrization [52] or the inverse-depth parametrization [6]. However, the former one is straightforward the latter two increases the consistency and accuracy.

EKF-SLAM uses "standard" propagation method of states ($\mathbf{x}_k$) and covariance matrix ($\mathbf{P}_k$) based on the IMU inertial measurements as described above, while the *update process* is calculated on the observed image features. Assuming a calibrated perspective camera, observation of feature i on the actual image at time step k is described by the following equation that describes the actual observation.

$$\mathbf{z}_{i,k} = \mathbf{h}(\mathbf{x}_{IMU,k}, \mathbf{p}_{f_{i,k}}) = \frac{1}{z_{f_i,\mathcal{C}_k}} \begin{bmatrix} x_{f_i,\mathcal{C}_k} \\ y_{f_i,\mathcal{C}_k} \end{bmatrix} + \mathbf{n}_{i,k} \tag{44}$$

where $\mathbf{n}_{i,k}$ is the measurement noise, and $\mathbf{p}_{f_i,\mathcal{C}_k} = [x_{f_i,\mathcal{C}_k}, y_{f_i,\mathcal{C}_k}, z_{f_i,\mathcal{C}_k}]$ describes observed position of feature f_i in camera orientation frame $\mathcal{C}_k$, and this position is described by the following equation and the $\mathbf{p}_{\mathcal{I},\mathcal{C}}$ and $\mathbf{R}_{\mathcal{C}\mathcal{I}}$ are the fixed position and rotation transformations between the IMU ($\mathcal{I}$) and the camera ($\mathcal{C}$) frames.

$$\mathbf{p}_{f_i,\mathcal{C}_k} = \mathbf{R}_{\mathcal{C}\mathcal{I}}\mathbf{R}_{\mathcal{I}_k\mathcal{G}}(\mathbf{p}_{f_i,\mathcal{G}} - \mathbf{p}_{\mathcal{I}_k,\mathcal{G}}) + \mathbf{p}_{\mathcal{I},\mathcal{C}} \tag{45}$$

Assuming that the actual position of the IMU frame is $\mathbf{p}_{\mathcal{I}_k,\mathcal{G}}$, EKF-SLAM defines a residual as the difference between the real observation $\mathbf{z}_{i,k}$ of the feature i and the projection of the estimated feature position ($\hat{\mathbf{p}}_{f_i,\mathcal{C}_k}$), and linearise it around the actual state ($\hat{\mathbf{x}}_{IMU,k}$) as:

$$\mathbf{r}_{i,k} = \mathbf{z}_{i,k} - \mathbf{h}(\hat{\mathbf{x}}_{IMU,k}, \hat{\mathbf{p}}_{f_i,\mathcal{C}_k}) \simeq \mathbf{H}_{i,k}(\hat{\mathbf{x}}_k)\tilde{\mathbf{x}}_k + \mathbf{n}_{i,k} \tag{46}$$

The $\mathbf{H}_{i,k}(\hat{\mathbf{x}}_k)$ is the Jacobian matrix of $\mathbf{h}$ with respect to the actual filter state estimate ($\hat{\mathbf{x}}_k$).

When $\mathbf{r}_{i,k}$ and $\mathbf{H}_{i,k}$ are computed, the outlier detection is done using Mahalanobis gating. If it succeeds the test using residual and observation Jacobian, Kalman gain and state innovation are computed according to basic EKF rules (see Eq. (31)). For Mahalanobis gating we compute the following:

$$\gamma_i = \mathbf{r}_i^T(\mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^T + \sigma^2\mathbf{I})^{-1}\mathbf{r}_i \tag{47}$$

Then it is compared to the threshold given by the 96 percent of the χ^2 distribution of dimensions equal to the residual vector.

Observation update step requires that all landmarks and joint-covariance matrix must be updated every time an image is registered by the camera. Considering the complexity of the EKF-SLAM, it is straightforward that the computational complexity is dominated by cubic of actual number of landmarks, thus the complexity is $\mathcal{O}(n^2)$. In practice, a map can consists of thousands of features, thus the EKF-SLAM becomes computationally intractable for large areas.

To provide the first-aid to this problem Sola proposed a method, when state and covariance matrices are updated by the actual observed feature numbers, in this way making a step to cover the real-time requirements. [47]

5.5.2 MSCKF

The fundamental advantage of filter-based algorithms is that they account for the correlations exist between pose of the camera and 3D positions of the observed features. On the other hand, the main limitation is its high computational complexity.

The motivation of Multi-State Constraint Filter (MSCKF) is the introduction of consecutive camera poses into state instead of observable feature landmarks, as it is first introduced by Nister [43], however this method does not incorporate inertial measurements. Sliding window-based solutions also appear in other papers [7].

Assuming that, N of the camera poses are included in the EKF state vector at time step k the MSCK state vector has the following form.

$$\mathbf{x}_k = [\mathbf{x}_{base,k}, \mathbf{q}_{\mathcal{GC}_1}^T, \mathbf{p}_{\mathcal{C}_1,\mathcal{G}} \ldots \mathbf{q}_{\mathcal{GC}_N}^T, \mathbf{p}_{\mathcal{C}_N,\mathcal{G}}]^T \tag{48}$$

Since *time update* is common for EKF-based pose estimators, the difference is maintained during *measurement update* step, when new image is arrived and features are tracked among the last N camera poses. The update process is based on each single feature f_j that has been observed from the set of N_j camera poses $(\mathbf{q}_{\mathcal{GC}_i}^T, \mathbf{p}_{\mathcal{C}_i,\mathcal{G}})$ that has been also available in the state vector.

The estimated feature position $\hat{\mathbf{p}}_{f_j,\mathcal{G}}$ in the global frame is triangulated from these N camera poses using feature observations. During this process, usually, a least-square minimization used with inverse-depth parametrization [6]. Then, the residual is defined as the difference between re-projections of estimated feature $\hat{\mathbf{p}}_{f_j,\mathcal{G}}$ to the N_j cameras and the real feature observation is defined as $\mathbf{r}_j^{(j)}$.

$$\mathbf{r}_i^{(j)} = \mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)} \tag{49}$$

On the other hand, the residual can be approximated by linearising around the estimates of the camera poses and the feature positions, where $\mathbf{H}_{\mathbf{x}_i}$ and $\mathbf{H}_{f_j}^{(j)}$ are the Jacobians of the measurement $\mathbf{z}_i^{(j)}$ with respect to the state and the feature position, respectively. After stacking the residuals for each N_j measurements of the f_j features we get the following equation.

$$\mathbf{r}^{(j)} \simeq \mathbf{H}_{\mathbf{x}}\tilde{\mathbf{x}} + \mathbf{H}_f^{(j)}\tilde{\mathbf{p}}_{f_j,\mathcal{G}} \tag{50}$$

Since actual state estimate $\mathbf{x}$ is used for estimation of $\hat{\mathbf{p}}_{f_j,\mathcal{G}}$, the error of state $\tilde{\mathbf{x}}$ and of feature position $\tilde{\mathbf{p}}_{f_j,\mathcal{G}}$ are correlated. The solution to this problem is projecting $\mathbf{r}^{(j)}$ on the left null-space of the matrix $\mathbf{H}_f^{(j)}$. Define $\mathbf{A}^{(j)}$ as the unitary matrix whose columns form the basis of the left null-space of $\mathbf{H}_f$, we get:

$$\mathbf{r}_o^{(j)} \simeq \mathbf{A}^{(j)\mathsf{T}}\mathbf{H}_x^{(i)}\tilde{\mathbf{x}}^{(j)} + \mathbf{A}^{(j)\mathsf{T}}\mathbf{n}^{(j)} = \mathbf{H}_o^{(j)}\tilde{\mathbf{x}}^{(j)} + \mathbf{n}_o^{(j)} \tag{51}$$

By also stacking residuals into a single vector from observations from each f_j features, we obtain the following single form of equation.

$$\mathbf{r}_o = \mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{n}_o \tag{52}$$

To reduce the computational complexity during update QR decomposition is applied of $\mathbf{H}_x$ [31]. After determining the $\mathbf{T}_H$ upper triangular matrix and its corresponding unitary matrix whose columns form bases for the null-space of $\mathbf{H}_x$, $\mathbf{Q}_1$. The residual is then reformulated as the following.

$$\mathbf{r}_n = \mathbf{Q}_1^\mathsf{T}\mathbf{r}_o = \mathbf{T}_H\tilde{\mathbf{x}} + \mathbf{n}_n \tag{53}$$

Based on the above measures, the residual $\mathbf{r}_n$ and the measurement Jacobian $\mathbf{T}_H$ the basic EKF update is applied (see Eg. (31)).

The correct co-operation between image based relative observations and inertial measurements requires to exactly know the transformation between camera and IMU orientation frames. In most of the solutions this transformation assumed to be known exactly, EKF is appropriate also to estimate these parameters. The improvements in MSCKF 2.0 [31] introduces these parameters $(\mathbf{q}_{IC}, \mathbf{p}_{C,I})$ to the state parameters. Besides, in this paper global orientation errors are considered and improved linearisation and calculation of Jacobians are provided. These methods improve the observability and increase accuracy and stability while estimating relative orientation and positions.

MSCKF model is also augmented with estimation of rolling shutter camera properties [28] and temporal calibration [30], while algorithm is provided for on-line self-calibration [32].

Regarding the computational complexity, it is easy to realize that instead of EKF-SLAM, complexity fundamentally depends on the number of registered camera states not on the number of observed features. However, calculation of $\mathbf{T}_H$ depends on the number of features ($\sim d$) and the columns of the $\mathbf{Q}_1$ (r). Other crucial factor is determined by the computation of covariance matrix update. The cost of MSCKF update is then determined by $\max(\mathcal{O}(r^2d), \mathcal{O}(m^3))$, where m is the size of the state vector.

One can see that since MSCKF uses sliding window for camera states, tracked features can be observed only for a time limited to window size. To overcome this limitation the authors designed a hybrid MSCKF-EKF SLAM solution, where the optimality found on using MSCKF for short features and long feature track is inserted to the state as it is done in EKF SLAM [29].

5.5.3 FastSLAM

FastSLAM implements PF method, however, high dimensional state-space of the SLAM problem makes it computationally infeasible to apply particle filters on the Bayesian-equations directly. FastSLAM solves this problem by applying a factorization to the posterior distribution as follows. [38]:

$$\mathrm{p}(\mathbf{x}_{1:k}, \mathbf{m}|\mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0) = \mathrm{p}(\mathbf{x}_{1:k}|\mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0) \prod_k \mathrm{p}(\mathbf{m}_k|\mathbf{x}_{1:k}, \mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0) \quad (54)$$

Estimation thus can be done in two steps: first we estimate the posterior of path trajectories then – based on estimated trajectory – we estimate the locations of the K landmarks independently. The path estimation is done by a modified particle estimator using Monte Carlo method, while estimation of landmarks is achieved by Kalman-filters. Because landmarks are conditioned on path estimation, if M particle is used to estimate the trajectory, then KM two dimensional Kalman-filters are required to estimate the landmarks.

FastSLAM runs time linear in the number of landmarks, however, the implementation of FastSLAM uses a tree representation of particles to run in $\mathcal{O}(M \log K)$. This way, the re-sampling of particles can happen much faster than using native implementation.

FastSLAM can handle huge amounts of landmarks – as extensive simulation has shown – and is at least as accurate as EKF-SLAM. However, the biggest problem of FastSLAM is the inability to forget the past (i.e. the pose and measurement history) and this way the statistical accuracy is lost [2].

FastSLAM has a more efficient extension called FastSLAM 2.0, which uses another proposal distribution including current landmark observations and this way calculating importance weights differently [39].

6 Implementation aspects

It is essential for visual odometry and SLAM algorithms to run real-time. Recent smartphones are equipped with a considerable amount of resources,

like multiple cores of CPU and GPU. To face to the real-time requirements by utilizing parallel resources, a couple of algorithms decouple real-time and background tasks. The computational burden is still really high for embedded devices. Fortunately, these algorithms give way to a lot of parallelisation to speed up computations.

Feature extraction is also much faster if done parallel, e.g. SiftGPU reported to extract SIFT features at 27 FPS on a nVidia 8800GTX card [58]. The widespread OpenCV[1] has also GPU support for various algorithms using CUDA and OpenCL. Not only feature detection and extraction but bundle adjustment can be parallelised to be ca. 30 times faster than native implementation as the Multi-core Bundle Adjustment project shows [59].

7 Evaluation

Beside the solutions described in this work, a huge amount of implementations are available (see `https://openslam.org`). For the prudent comparison of the methods, algorithms and real implementations, widely known datasets are used. These datasets provide huge amount of video frames of different trajectories with ground truth, containing mainly grayscale and RGB images but often RGB-D and laser data is also accessible. The most widely used datasets are the KITTI dataset [15], the RGB-D dataset [53] and New College Data Set [51], from those the KITTI odometry dataset consists of 22 stereo sequences (which can also be used as a monocular data) and a comprehensive evaluation of different SLAM methods listing accuracy and speed.

Regarding the KITTI dataset a huge list about performance evaluation of available implementations is published at `http://www.cvlibs.net/datasets/kitti/eval_odometry.php`.

8 Conclusion

Huge variety of algorithms and solutions are currently available to tackle the strict requirements of the accurate and real time visual indoor positioning that augmented reality-based applications demand. These algorithms build on the results of research work on computer vision from the last decades, which went through big evolution, from SFM to the real-time SLAM approaches. However, to face to real-time requirements filter-based solutions tightly coupling inertial measurements with visual odometry are emerging. Through embedding inertial

[1] OpenCV can be found at `http://opencv.org`

measurements from IMU for motion estimation to the projective geometry principles, these approaches are promising for future implementations, however they suffer from the capability of long-lasting state parameter estimations.

Acknowledgements

This publication and related research was supported by the PIAC-13-1-2013-0226 (Organic Localization) project. The project is supported by the Hungarian Government, and financed by the Research and Technology Innovation Fund.

References

[1] Y. I. Abdel-Aziz, H. M. Karara, M. Hauck, Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry, *Photogrammetric Engineering and Remote Sensing* **81**, 2 (2015) 103–107. ⇒191

[2] T. Bailey, J. Nieto, E. Nebot, Consistency of the fastslam algorithm, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'06)*, Orlando, FL, USA, 2006, pp. 424–429. ⇒209

[3] S. Baker, I. Matthews, Lucas-kanade 20 years on: A unifying framework, *International Journal of Computer Vision (IJCV)* **56**, 3 (2004) 221–255. ⇒198

[4] S. S. Beauchemin, R. Bajcsy, Modelling and removing radial and tangential distortions in spherical lenses, *Proc. of International Workshop on Theoretical Foundations of Computer Vision (TFCV'00), Multi-Image Analysis, Lecture Notes in Computer Science*, Dagstuhl Castle, Germany, 2000, pp. 1–21. ⇒189

[5] J. Chao, A. Al-Nuaimi, G. Schroth, E. Steinbach, Performance comparison of various feature detector-descriptor combinations for content-based image retrieval with jpeg-encoded query images, *Proc. of International Workshop on Multimedia Signal Processing (MMSP'13)*, Pula, Sardinia, Italy, 2013, pp. 29–34. ⇒190

[6] J. Civera, A. J. Davison, J. M. M. Montiel, Inverse depth parametrization for monocular slam, *IEEE Transactions on Robotics* **24**, 5 (2008) 932–945. ⇒205, 207

[7] L. E. Clement, V. Peretroukhin, J. Lambert, J. Kelly, The battle for filter supremacy: A comparative study of the multi-state constraint kalman filter and the sliding window filter, *Proc. of Conference on Computer and Robot Vision (CRV'15)*, Halifax, Nova Scotia, 2015, pp. 23–30. ⇒207

[8] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, Monoslam: Real-time single camera slam, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* **29**, 6 (2007) 1052–1067. ⇒199

[9] H. Durrant-Whyte, T. Bailey, Simultaneous localization and mapping: part i, *IEEE Robotics & Automation Magazine* **13**, 2 (2006) 99–110. ⇒187, 200

[10] J. Engel, T. Schöps, D. Cremers, Lsd-slam: Large-scale direct monocular slam, *Proc. of 13th European Conference on Computer Vision (ECCV'14)*, volume 2, Springer International Publishing, Zurich, Switzerland, 2014, pp. 834–849. ⇒ 197

[11] J. Engel, J. Sturm, D. Cremers, Semi-dense visual odometry for a monocular camera, *Proc. of IEEE International Conference on Computer Vision (ICCV'13)*, Sydney, Australia, 2013, pp. 1449–1456. ⇒197

[12] M. A. Fischler, R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM* **24**, 6 (1981) 381–395. ⇒191

[13] C. Forster, M. Pizzoli, D. Scaramuzza, Svo: Fast semi-direct monocular visual odometry, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'14)*, Hong Kong, China, 2014, pp. 15–22. ⇒198

[14] F. Fraundorfer, D. Scaramuzza, Visual odometry, part ii: Matching, robustness, optimization, and applications, *IEEE Robotics & Automation Magazine* **19**, 2 (2012) 78–90. ⇒187

[15] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR'12)*, IEEE, Providence, RI, USA, 2012, pp. 3354–3361. ⇒ 210

[16] N. Gordon, Novel approach to nonlinear/non-gaussian bayesian state estimation, *IEE Proceedings F (Radar and Signal Processing)* **140** (1993) 107–113(6). ⇒205

[17] C. Harris, J. Pike, 3d positional integration from image sequences, *Proc. of the Alvey Vision Conference*, Manchester, UK, 1987, pp. 87–90. ⇒187

[18] C. Harris, M. Stephens, A combined corner and edge detector, *Proc. of the 4th Alvey Vision Conference*, Manchester, UK, 1988, pp. 147–151. ⇒190

[19] R. I. Hartley, In defense of the eight-point algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI'97)* **19**, 6 (1997) 580–593. ⇒191

[20] R. I. Hartley, A. Zisserman, *Multiple View Geometry in Computer Vision*, (2nd edition), Cambridge University Press, Cambridge, England, UK, 2004. ⇒187, 190, 191, 192

[21] C. Kerl, J. Sturm, D. Cremers, Robust odometry estimation for rgb-d cameras, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, 2013, pp. 3748–3754. ⇒195

[22] G. Klein, D. Murray, Parallel tracking and mapping for small ar workspaces, *Proc. of 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007)*, Nara, Japan, 2007, pp. 225–234. ⇒187, 193

[23] G. Klein, D. Murray, Improving the agility of keyframe-based SLAM, *Proc. of 10th European Conference on Computer Vision (ECCV'08)*, Marseille, France, 2008, pp. 802–815. ⇒193

[24] L. Kneip, D. Scaramuzza, R. Siegwart, A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation, *Proc. of IEEE Conference on Computer Vision and*

Pattern Recognition (CVPR'11), Colorado Springs, USA, 2011, pp. 2969–2976. ⇒191

[25] K. Konolige, M. Agrawal, J. Solà, Large-scale visual odometry for rough terrain, *Proc. of International Symposium on Robotics Research*, Hiroshima, Japan, 2007, pp. 201–212. ⇒199

[26] V. Lepetit, F.Moreno-Noguer, P.Fua, Epnp: An accurate o(n) solution to the pnp problem, *International Journal Computer Vision* **81**, 2 (2009) 155–166. ⇒191

[27] H. Li, R. Hartley, Five-point motion estimation made easy, *Proc. of 18th International Conference on Pattern Recognition (ICPR'06)*, volume 1, Hong Kong, China, 2006, pp. 630–633. ⇒191

[28] M. Li, B. H. Kim, A. I. Mourikis, Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, 2013, pp. 4712–4719. ⇒208

[29] M. Li, A. I. Mourikis, Optimization-based estimator design for vision-aided inertial navigation, *Proc. of the Robotics: Science and Systems Conference*, Sydney, NSW, Australia, 2012, pp. 504–509. ⇒209

[30] M. Li, A. I. Mourikis, 3-d motion estimation and online temporal calibration for camera-imu systems, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, 2013, pp. 5709–5716. ⇒208

[31] M. Li, A. I. Mourikis, High-precision, consistent ekf-based visual-inertial odometry, *International Journal of Robotics Research* **32**, 6 (2013) 690–711. ⇒208

[32] M. Li, H. Yu, X. Zheng, A. I. Mourikis, High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'14)*, Hong Kong, China, 2014, pp. 409–416. ⇒208

[33] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer Publishing Company, Incorporated, 2008. ⇒204

[34] H. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, *Nature* **293**, 10 (1981) 133–135. ⇒187

[35] M. A. Lourakis, A. Argyros, SBA: A software package for generic sparse bundle adjustment, *ACM Transactions on Mathematical Software (TOMS)* **36**, 1 (2009) 1–30. ⇒192

[36] D. G. Lowe, Object recognition from local scale-invariant features, *Proc. of IEEE International Conference on Computer Vision (ICCV'99)*, volume 2, Kerkyra, Greece, 1999, pp. 1150–1157. ⇒190

[37] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide-baseline stereo from maximally stable extremal regions, *Image and Vision Computing* **22**, 10 (2004) 761–767. ⇒190

[38] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, Fastslam: A factored solution to the simultaneous localization and mapping problem, *Proc. of National Conference on Artificial Intelligence*, American Association for Artificial Intelligence (AAAI), Edmonton, Alberta, Canada, 2002, pp. 593–598. ⇒209

[39] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, FastSLAM 2.0: An improved

particle filtering algorithm for simultaneous localization and mapping that provably converges, *Proc. of International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, 2003, pp. 1151–1156. ⇒209

[40] R. Mur-Artal, J. M. M. Montiel, J. D. Tardós, ORB-SLAM: a versatile and accurate monocular SLAM system, *IEEE Transactions on Robotics* **31**, 5 (2015) 1147–1163. ⇒193

[41] R. A. Newcombe, S. J. Lovegrove, A. J. Davison, Dtam: Dense tracking and mapping in real-time, *Proc. of International Conference on Computer Vision (ICCV'11)*, IEEE Computer Society, Washington, DC, USA, 2011, pp. 2320–2327. ⇒196

[42] D. Nistér, An efficient solution to the five-point relative pose problem, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI'04)* **26**, 6 (2004) 756–777. ⇒191

[43] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, IEEE Computer Society, Seattle, WA, USA, 2006, pp. 2161–2168. ⇒195, 207

[44] T. Oskiper, Z. Zhu, S. Samarasekera, R. Kumar, Visual odometry system using multiple stereo cameras and inertial measurement unit, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, Minneapolis, MN, USA, 2007, pp. 1–8. ⇒199

[45] O. Pizarro, R. Eustice, H. Singh, Relative pose estimation for instrumented, calibrated imaging platforms, *Proc. of Digital Image Computing Techniques and Applications*, Sydney, Australia, 2003, pp. 601–612. ⇒191

[46] S. I. Roumeliotis, A. E. Johnson, J. F. Montgomery, Augmenting inertial navigation with image-based motion estimation, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'02)*, volume 4, Washington, DC, USA, 2002, pp. 4326–4333. ⇒199

[47] C. Roussillon, A. Gonzalez, J. Solà, J.-M. Codol, N. Mansard, S. Lacroix, M. Devy, Rt-slam: A generic and real-time visual slam implementation, *Proc. of International Conference of Computer Vision Systems (ICVS'11), Lecture Notes in Computer Science* **6962** (2011) 31–40. ⇒207

[48] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: An efficient alternative to sift or surf, *Proc. of International Conference on Computer Vision (ICCV'11)*, Barcelona, Spain, 2011, pp. 2564–2571. ⇒190

[49] S. Särkkä, *Bayesian Filtering and Smoothing*, (1st edition), Cambridge University Press, New York, NY, USA, 2013. ⇒203, 205

[50] D. Scaramuzza, F. Fraundorfer, Visual odometry, part i: The first 30 years and fundamentals, *IEEE Robotics & Automation Magazine* **18**, 4 (2011) 80–92. ⇒187

[51] M. Smith, I. Baldwin, W. Churchill, R. Paul, P. Newman, The new college vision and laser data set, *The International Journal of Robotics Research* **28**, 5 (2009) 595–599. ⇒210

[52] J. Solà, Consistency of the monocular ekf-slam algorithm for three different land-

mark parametrizations, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'10)*, Anchorage, Alaska, 2010, pp. 3513–3518. ⇒205

[53] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers, A benchmark for the evaluation of rgb-d slam systems, *Proc. of International Conference on Intelligent Robot Systems (IROS'12)*, Vilamoura, Algarve, Portugal, 2012, pp. 573–580. ⇒210

[54] J. Tardif, M. G. M. Laverne, A. Kelly, M. Laverne, A new approach to vision-aided inertial navigation, *Proc. of International Conference on Intelligent Robots and Systems (IROS'10)*, Taipei, Taiwan, 2010, pp. 4161–4168. ⇒199

[55] B. Triggs, Camera pose and calibration from 4 or 5 known 3d points, *Proc. of IEEE International Conference on Computer Vision (ICCV'99)*, volume 1, Kerkyra, Greece, 1999, pp. 278–284. ⇒191

[56] B. Triggs, P. F. McLauchlan, R. I. Hartley, A. W. Fitzgibbon, Bundle adjustment — a modern synthesis, *Proc. of International Workshop on Vision Algorithms: Theory and Practice*, Springer Berlin Heidelberg, Corfu, Greece, 1999, pp. 298–372. ⇒192

[57] S. Weiss, R. Siegwart, Real-time metric state estimation for modular vision-inertial systems, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'11)*, Shanghai, China, 2011, pp. 4531–4537. ⇒199

[58] C. Wu, SiftGPU: A GPU implementation of scale invariant feature transform (SIFT), `http://cs.unc.edu/~ccwu/siftgpu`, 2007. ⇒210

[59] C. Wu, S. Agarwal, B. Curless, S. M. Seitz, Multicore bundle adjustment, *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR'11)*, Colorado Springs, USA, 2011, pp. 3057–3064. ⇒210

Received: May 17, 2016 • Revised: November 11, 2016

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **8,** 2 (2016) 216–240

DOI: 10.1515/ausi-2016-0010

Internal quality evolution of a large test system–an industrial study

Attila KOVÁCS
Eötvös Loránd University
email: attila.kovacs@inf.elte.hu

Kristóf SZABADOS[1]
Eötvös Loránd University
email:
kristof.szabados@ericsson.com

Abstract.

This paper presents our empirical observations related to the evolution of a large automated test system. The system observed is used in the industry as a test tool for complex telecommunication systems, itself consisting of more than one million lines of source code. This study evaluates how different changes during the development have changed the number of observed Code Smells in the test system. We have monitored the development of the test scripts and measured the code quality characteristics over a five years period.

The observations show that the introduction of continuous integration, the existence of tool support for quality improvements in itself, changing the development methodologies (from waterfall to agile), changing technical and line management structure and personnel caused no measurable change in the trends of the observed Code Smells. Internal quality improvements were achieved mainly by individuals intrinsic motivation. Our measurements show similarities with earlier results on software systems evolutions presented by Lehman.

Computing Classification System 1998: D.2.2, D.2.9
Mathematics Subject Classification 2010: 68N99
Key words and phrases: code smells; empirical study; software evolution; test systems; Lehman's laws; TTCN-3

[1]Corresponding author

1 Introduction

Do we really know how to build large test sets? Do we really know how test systems evolve, how can their development be managed, their quality ensured?

Nowadays the usage of software systems belongs to the everyday life of the society, yet testing these systems is still a challenging and not really understood activity. Software helps in navigating to locations, supports communication with other people, drives the production, distribution and consumption of energy resources. Software controls companies, trades on the markets, takes care of people's health.

To support the testing these systems need, ETSI[2] developed the TTCN-3[3] language, which can be used in testing of reactive systems. By now test systems developed by ETSI, 3GPP[4] and in the industry, have evolved to be comparable to the tested systems in both size and complexity ([26]). Standardized test systems were shown to contain design problems similar to those present in other (C / C++ / Java) systems ([29, 28]).

In this paper we show empirical observations on the evolution of two large test systems developed in the industry and our measurements on the number of code smells in them. We ask the following research questions: Was the number of measured code smells affected by the introduction of Continuous Integration (RQ1), by tool support for detecting code smells (RQ2), by the merging of 2 test systems (RQ3), by doing the development using different methodologies (RQ4), by changing leaders on the project (RQ5) ? Our final research question is: Do code smells in test systems follow predictable patterns during the system's evolution (RQ6) ?

In this study we examine the evolution of TTCN-3 test systems from a software quality point of view. In our research we treat test systems as software products used for testing, rather than tests. We present historical information on changes in line and project management, development practices, organizational and technical structures, tool support that happened during their five years development period. By comparing our measurements with historical information we show how the evolution could affect the quality of the observed large scale test system.

[2]European Telecommunication Standardization Institute
[3]Testing and Test Control Notation Version 3
[4]3rd Generation Partnership Project

1.1 Structure of this paper

This paper is organized as follows. In Section 2 we present earlier results related to this subject. Section 3 contains the history of the studied projects and the measurement environment. In Section 4 we analyze the measured data and correlate the measurements to publicly known items. Section 5 presents the measured results from the enterprise' point of view (significance of development methods, leadership styles, tool support) to emphasize their influences. Section 6 lists the factors that might be a threat to the validity of our results. Finally, Section 7 summarizes our findings.

2 Previous work

Before presenting our findings it is necessary to understand the importance and limitations of code smells, software evolution and the state of how this knowledge is translated to testing.

2.1 Code smell studies

Code smells were introduced by Fowler [4] as issues that are not necessarily technically incorrect codes and do not disable the program from functioning, but might indicate architectural problems or misunderstandings, issues which are very hard to detect. Since then, the initial list of 22 code smells has been extensively extended (see e.g. [33, 19, 21]), and code smells have become a metaphor for software design aspects that may cause problems during further development and maintenance of software systems.

Empirical work on code smells revealed that smelly codes in software systems are changed more frequently than other codes ([9, 22]). Moser at al. found [20] that in the context of small teams working in volatile domains (e.g. mobile development) correcting smelly code increased software quality, and measurably increased productivity.

On the other hand the value of code smells has been questioned by many. Yamashita et al. found [37] that only 30% of the maintenance problems were related to files containing code smells. Sjøberg at al. found [25] that none of the code smells they investigated was significantly associated with increased maintenance effort when adjusted by file size. Macia et al. observed [18] that more than 60% of the automatically detected code anomalies were not correlated with architectural problems.

In order to understand software aging better the lifespan of code smells was

studied by many (see e.g. [23]). Chatzigeorgiou et al. published [1] that code smells are usually introduced with new features, accumulating as the project matures, persisting up to the latest examined version. The disappearance of smell instances was usually the side effect of maintenance works, not the result of targeted correcting activities. Peters and Zaidman concluded [24] that developers might be aware of code smells, but are usually not concerned by their presence. In each system inspected there were only one or two developers who resolved code smell instances intentionally, or resolved significantly more instances than others (possibly unintentionally).

In their 2013 paper Yamashita et al. [35] conducted a survey on 85 software professionals in order to understand the level of knowledge about code smells and their perceived usefulness. They found that 32% of the respondents did not know about code smells, nor did they care. Those who were at least somewhat concerned about code smells indicated difficulties with obtaining organizational support and tooling. In their empirical studies ([34, 36]) they observed that code smells covered only some of the maintainability aspects considered important by developers. They also observed, that developers did not take any conscious action to correct bad smells that were found in the code.

2.2 Software evolution studies

Lehman [14] described the evolution of software as the study and management of repeatedly changing software over time for various reasons.

Out of Lehman's *laws of software evolution* the following are the most relevant for this study [16]:

- Law 2: "As an E-type[5] system is evolved its complexity increases unless work is done to maintain or reduce it"
- Law 4: "Unless feedback mechanisms are appropriately adjusted, average effective global activity rate in an evolving E-type system tends to remain constant over product lifetime"
- Law 5: "In general, the incremental growth and long term growth rate of E-type systems tend to decline"
- Law 8: "E-type evolution processes are multi-level, multi-loop, multi-agent feedback systems"

Lehman and Ramil [15], and Lawrence [10] found that commercial systems have a clear linear growth, viewed over a number of releases. Izurieta and

[5]systems actively used and embedded in a real world domain.

Bieman found [6] that Open Source Software products FreeBSD and Linux also appear to grow at similar rates.

Turski showed ([32]) that the gross growth trends can be predicted, with a mean absolute error of order 6%, with

$$S_{i+1} = S_i + \hat{e}/S_i^2, \tag{1}$$

where S_i is the system size at the i-th measurement, and $\hat{e}$ can be calculated as $(S_{i-1} - S_1)/(\sum_{k=1}^{i-1} 1/S_k^2)$.

There are plenty of researches ([17, 13, 8, 7, 5]) in which the authors show that the laws seem to be supported by solid evidence. But applying them currently requires the understanding of human, technical, usage and organizational contexts of the measures, they were derived from.

2.3 Test quality and evolution studies

Deursen at al. [3] noticed while working on a Java project that tests in their test system have their own set of problems and repertoire of solutions, which they translated into code smells and refactorings for the JUnit framework.

Zaidman et al. [38] witnessed both phased and synchronous co-evolution of tests and production codes.

Zeiss et al. [39] published a model for TTCN-3 test specification derived from ISO 9126, by translating the quality standard for testing.

2.4 Our contributions

In our long term research we explore similarities between systems of tests and software systems. We look at tests as software systems, re-interpreting test systems and script as software products.

In [26] we have shown that automated test systems written in TTCN-3 can grow large and complex similar to the structures studied in [31]. In [29] we have defined 86 code smells for TTCN-3 and their relations to international software quality standards. In order to understand the quality of such huge test systems 35 selected code smells were implemented and measured on 16 projects.

The updated list of code smells, used in this measurement, can be found at [30].

To the best of our knowledge the evolution of code smells in test systems was not yet studied in the domain of testing communication systems. We have also not found any work presenting the relation between real world events and what effects they had on the quality of test suites.

3 History of the studied systems

3.1 Background

Current test systems have grown large with many different parts, which might be developed separately in different organizations. Although these parts are designed to become test suites or serve as components of test suites, most of them can not be called tests (ex. the software layer converting between abstract TTCN-3 messages and actual bit stream messages). For this reason in this article we use the term "test system" to describe software components of test suites and the test suites built of them.

We have studied two test systems developed and used at our industry partner. The history of these systems goes back to 2005. We started to analyze them in 2012. At the end of 2012 the two systems were merged to form a single solution.

Both test systems are built on a set of libraries and tools in a hierarchical structure. We will call this set of systems `Common`. Parts of `Common` in the lower abstraction layers support (1) sending and receiving messages of a specific protocol, (2) the protocol logic (3) and the forming of a glue layer between a generic product and some specific usage.

`System-1` was originally designed for demonstrating and testing the features of `Common`, containing a set of project independent, reusable data structures and algorithms that can be used for creating high levels of load in TTCN-3.

`System-2` was aimed at testing IMS[6] products. At the end of 2012 these two test systems were merged into one, which we will call the `Merged` System.

`System-1`, `System-2` and `Merged` offer complex and computationally intensive functionalities. They are used to test if the System Under Test is able to: (1) handle large amount of users, (2) handle large data traffic coming in a mix of several supported traffic type and (3) stay stable for long durations (days or even weeks).

Titanium is our open source ([27]), static code analyzer, developed as part of our research to support detecting issues in TTCN-3 source codes.

3.2 History of the tracked systems

In this section we provide a list of the most important events which could have influenced the quality of the studied systems.

[6]IP Multimedia Core Network Subsystem is an architectural framework designed by 3GPP for evolving mobile networks beyond GSM

- 2005 - 2006: The development on Core Library started.
- Mid. 2007: First Core Library release.
- Early 2008: `System-1` was born. Developers were dedicated to independent customers with little coordination among them.
- Mid. 2009: A team in `System-1` switched to Scrum methodology for development, led by an experienced Scrum Master. Strong coordination appeared for the teams but there were still external developers working on the same source codes.
- End of 2009: The Scrum Master moved to a different unit inside the company. Her place was filled with people she trained earlier.
- 2010: `System-2` was moved from abroad to in-house. The in-house team decided to rewrite the code from ground up.
- 2010 - 2011: The team of `System-1` was experimenting with Kanban and custom methodologies designed specifically for the project.
- February 2012: Work starts on Titanium.
- 2012 beginning: `System-2` changed to a new version handling repository. This was the first version of its source code available for us to study.
- 2012 first half year: New Scrum Master and Product Owner were selected for `System-1`. One system architect was selected from each team to analyze requirements, write implementation studies and guidelines. A System Architect Forum was created, fostering information sharing between system architects.
- 2012 second half year: The organizational structure of `System-1` was changed. The Scrum Master and the Product Owner were replaced. From this point in time there were no external developers changing the source code in parallel with the team.
- Dec. 2012: `System-1` and `System-2` were merged forming the `Merged` System. The source codes were stored in a new source code repository.
- May 2013: during a "Boost day" event Titanium is integrated into the continuous integration server of `Merged`. The effect of every change is measured and displayed on web pages accessible by all developers and managers in the project.
- 11 July 2013: "Titanium Quest" was organized. Among others, the participants removed 10% of fixme and todo comments, reduced the number of "circular importations" by 57% and the number of "unused imports" by 50%. The removal of the circular imports enabled a 3% improvement in the build time of the `Merged` System.

- 2014 first half year: All of the system architects of the `Merged` System are replaced by a single System Architect.
- 17 July 2014: The "Green Day" event is organized. Among others, most of the remaining "unused imports" were removed.
- 4th December 2014: the "Black Thursday" event is organized. Participants removed 0.6% of the code, reviewing readonly variables, inout and out parameters, unused local definitions

"Titanium Quest", "Green Day" and "Black Thursday" were 24 hour code fixing challenges.

3.3 Subjective information

From organizational point of view these systems were developed by several teams. The size, structure and responsibilities of the teams changed with time. All teams were working within the same organizational unit, sitting together in the same part of the building. Communication among members of teams and among teams was not obscured.

Developers of `System-1`, `System-2` and `Merged` have mentioned that between 2008 and 2011 the system architect was always available for questions but it was not mandatory to ask him. Members of the System Architect Forum mentioned that they had no tools to enforce their proposals as the teams were following agile methodologies (particularly Scrum) where reviewing and accepting the implementations of features/requirements was the responsibility of the PO role.

3.4 Trainings on Code Smells and usage of Titanium

Between 22 July 2013 and 17th July 2014 there were 73 issues reported for the `Merged` System. These issues range from product and structural issues via performance and code duplications to code complexity and inefficient variable scoping. All reports contained the location and a description of the specific defect. Some reports contain advises for possible corrections as well.

During 2014 we organized trainings to spread knowledge about code smells with the following agendas:

- January: Handling lists efficiently in TTCN-3,
- Mids of February: Introduction to code smells and their relevance,
- End of February: Advanced uses of Altsteps

- March: How to efficiently assign a value?
- April: Parameter passing in TTCN-3 in theory and practice.

3.5 Effort

Table 1 shows the actual efforts (in ratios of man-hours) reported for the test systems at different points in time. For each year we show data for the months January and October[7] to represent the starting and closing of the year.

Name	2009		2010		2011		2012		2013		2014	
	Jan	Oct	Jan	Oct	Jan	Oct	Jan	Oct	Jan	Oct	Jan	Oct
`Common`	1.00	2.06	1.70	1.92	1.54	1.97	1.90	1.56	1.30	1.50	1.39	1.36
`System-1`	1.20	0.52	0.64	0.76	0.76	0.78	0.81	1.14				
`System-2`				0.68	0.42	1.07	1.06	1.13				
`Merged`									2.63	2.65	3.35	3.51

Table 1: The actual effort (ratios of man-hours) reported on the investigated systems at different points in time. The values are shown as ratios compared to the effort reported for `Common` in January, 2009.

The efforts invested into the products show a growing trend with some fluctuations. Since the work started in 2009 the number of Man-Hours reported for the project have almost doubled by the end of 2014.

After the merge all previous efforts invested into `System-1` and `System-2` were redirected to `Merged` taking away some resources from `Common`.

4 Code smell measurements

In this section we present our measurements. For each day in the investigated range we checked out the source code in the state it was at midnight and measured the number of code smells (listed at [30]) present.

4.1 Size

We analyzed the size growth of `System-1` and `Merged` systems measured in LOC. Figure 1 shows the measured data[8] and a quadratic trend line fitted.

[7] In November and December employees tend to go on vacations, significantly changing the amount of work reported on each project.

[8] Measuring the lines of code was an afterthought in our case. For System-1 we measured the lines of code of released software versions, for Merged we show monthly measurement

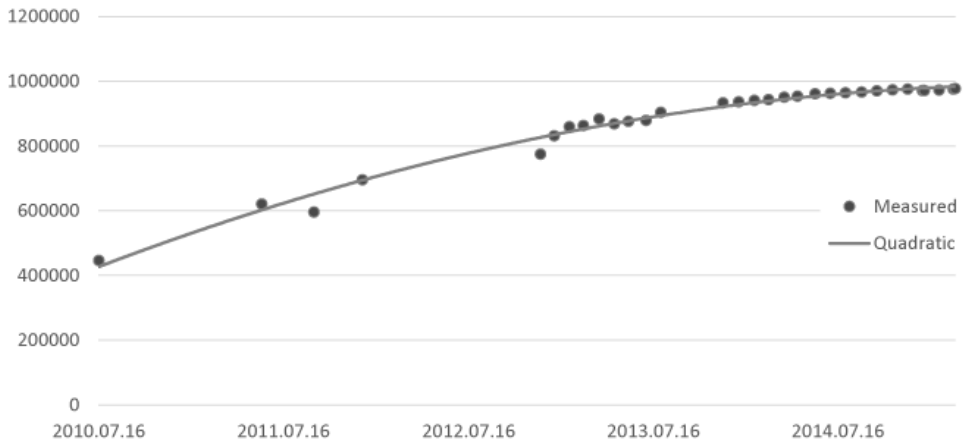

Figure 1: Size evolution of the `System-1` and `Merged` systems.

When we used the Lehman's prediction according to equation (1) on the lines of code in `Merged`, we measured a maximal absolute error between the measured data and the predicted model is about 3%.

4.2 Correlations among code smells

For each possible pair of code smells we calculated the Pearson correlation between the data series of the code smells ([30]) on the `Common` + `System-1` + `Merged` system evolution (Table 2). We excluded code smells having less than 50 occurrences at every measurement point during the development of the systems, as even small changes can appear to break trends using such small numbers. Based on the correlation values the code smells could be separated into 3 groups:

1. In the largest group, the correlation was at least 0.95 between the smell pairs. These are exactly the code smells that have never been addressed during special events: *FIXME tags, TODO tags, empty statement block, if instead altguard, magic numbers, magic strings, logic inversion, definition should be private, readonly inout formal parameter, size check in loop, switch on boolean, too complex expression, too many parameters, uncommented function, uninitialized variable, unused function return values, visibility in definition.*

Code Smells	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1 FIXME tags	1.00																										
2 TODO tags	0.98	1.00																									
3 Circular importation	0.42	0.40	1.00																								
4 Empty statement block	0.99	0.98	0.43	1.00																							
5 If instead altguard	0.99	0.97	0.43	0.98	1.00																						
6 If without else	0.87	0.87	0.44	0.91	0.87	1.00																					
7 Magic numbers	0.98	0.96	0.47	0.99	0.96	0.90	1.00																				
8 Magic strings	0.99	0.98	0.42	0.99	0.98	0.90	0.99	1.00																			
9 Module name in definition	0.86	0.85	0.39	0.90	0.86	0.99	0.89	0.90	1.00																		
10 Logic inversion	0.97	0.97	0.43	0.99	0.95	0.92	0.98	0.99	0.93	1.00																	
11 Definition should be private	0.98	0.96	0.45	0.99	0.96	0.89	0.99	0.99	0.90	0.98	1.00																
12 Readonly local variable	0.68	0.69	0.35	0.72	0.67	0.67	0.72	0.68	0.66	0.74	0.67	1.00															
13 Readonly out formal parameter	-0.42	-0.45	-0.31	-0.49	-0.44	-0.79	-0.47	-0.47	-0.74	-0.51	-0.43	-0.37	1.00														
14 Readonly inout formal parameter	0.97	0.97	0.46	0.98	0.96	0.86	0.98	0.97	0.85	0.97	0.97	0.75	-0.42	1.00													
15 Size check in loop	1.00	0.98	0.41	0.99	0.98	0.86	0.98	0.99	0.86	0.98	0.98	0.67	-0.40	0.98	1.00												
16 Switch on boolean	0.98	0.97	0.39	0.98	0.95	0.81	0.97	0.97	0.81	0.97	0.97	0.68	-0.33	0.97	0.99	1.00											
17 Too complex expression	0.99	0.98	0.42	0.99	0.98	0.90	0.99	1.00	0.90	0.99	0.99	0.67	-0.47	0.97	0.99	0.97	1.00										
18 Too many parameters	0.99	0.98	0.41	0.99	0.98	0.85	0.98	0.99	0.85	0.97	0.98	0.68	-0.39	0.97	0.99	0.98	0.99	1.00									
19 Typename in definition	0.94	0.93	0.42	0.93	0.95	0.80	0.92	0.95	0.80	0.92	0.96	0.56	-0.32	0.93	0.96	0.93	0.95	0.93	1.00								
20 Uncommented function	0.97	0.95	0.47	0.98	0.96	0.95	0.98	0.98	0.95	0.98	0.98	0.68	-0.57	0.95	0.97	0.94	0.98	0.96	0.92	1.00							
21 Uninitialized variable	0.99	0.99	0.41	0.99	0.98	0.87	0.98	0.99	0.86	0.98	0.98	0.70	-0.42	0.98	1.00	0.98	0.99	0.99	0.95	0.96	1.00						
22 Unnecessary control	0.86	0.87	0.44	0.91	0.88	1.00	0.89	0.90	0.98	0.92	0.88	0.67	-0.80	0.86	0.86	0.82	0.90	0.85	0.80	0.94	0.87	1.00					
23 Unused function return values	0.97	0.94	0.40	0.96	0.97	0.91	0.96	0.98	0.90	0.95	0.96	0.57	-0.53	0.92	0.97	0.93	0.98	0.96	0.93	0.97	0.96	0.90	1.00				
24 Unused global definition	0.91	0.92	0.38	0.93	0.89	0.79	0.93	0.92	0.80	0.95	0.91	0.82	-0.32	0.93	0.92	0.94	0.92	0.93	0.84	0.89	0.93	0.79	0.83	1.00			
25 Unused import	-0.72	-0.72	-0.43	-0.75	-0.75	-0.87	-0.74	-0.75	-0.84	-0.73	-0.74	-0.34	0.79	-0.70	-0.72	-0.64	-0.76	-0.70	-0.73	-0.81	-0.71	-0.87	-0.84	-0.49	1.00		
26 Unused local definition	0.04	0.05	-0.11	0.01	-0.01	-0.32	0.02	0.00	-0.28	0.02	0.01	0.34	0.69	0.09	0.05	0.14	-0.01	0.09	0.01	-0.11	0.07	-0.32	-0.17	0.31	0.61	1.00	
27 Visibility in definition	0.98	0.97	0.38	0.97	0.95	0.83	0.97	0.98	0.83	0.96	0.97	0.64	-0.36	0.96	0.99	0.98	0.98	0.99	0.94	0.95	0.98	0.82	0.94	0.93	-0.67	0.10	1.00

Table 2: The Pearson correlation between the data series of the code smells. To save on space the numbers in the header represent the code smells, numbered in the first column.

2. Code smells with correlation values related to the first group, lying between 0.3 and 0.95, were addressed during special events, but only a fraction of their appearances were removed: *Module name in definition, If without else, Unnecessary control, readonly local variable, typename in definition, unused global definition, circular importation.*

3. Three code smells have zero or negative medium correlation values (−0.42, −0.72 and 0.04) compared to the members of the first group. Most of the occurrences of these code smells were addressed during special events or in personal efforts: *readonly out formal parameter, unused import, Unused local definition.*

4.3 Code smell trends

In this section we show how the different events in the history of the test systems have correlated with the changes in the number of code smells.

4.3.1 First correlation group

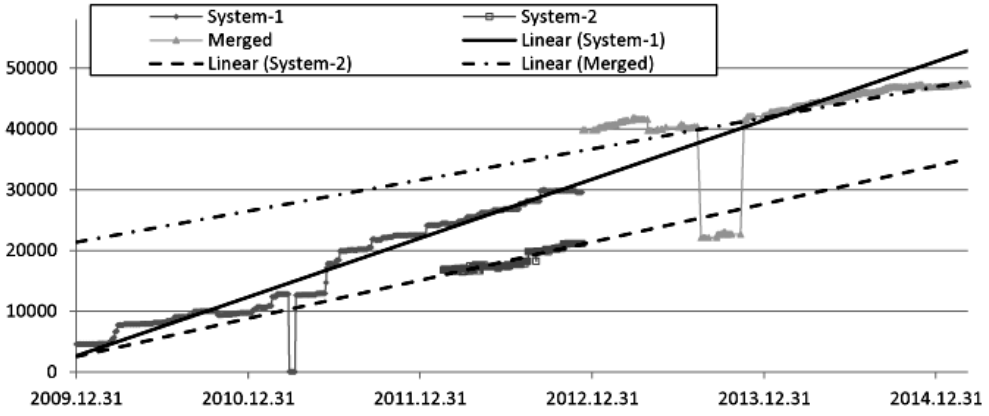

Figure 2: *Number of magic string* issues and its linear approximations.

From the first correlation group we present the *magic strings* code smell. The data series of other code smells from this group have high correlation with this data series, hence, we omit to show them.

In both systems the cumulative number of magic strings was increasing following a nearly linear trend (Figure 2). Before the merge the number of

magic strings was growing by 5152/7923/7027 instances in `System-1` and by 4225 instances in `System-2` per year. Directly after the merge the growth dropped to 2378 instances per year for most of the year 2013. The growth speed reached 4733 instances per year in 2014.

It is interesting to point out that the reduction of growth after the merge, lasted approximately until the numbers were fitting to the original growth trend of `System-1`. From 2014 the growth of `Merged` followed a trend much closer to that of `System-2` than to `System-1`.

The sudden increases in the measured data in `System-1` till the middle of 2011 indicates 3 months development cycles and developers working on branches separate from the main development branch. Later in `System-1` and `System-2` these increases are not present, indicating frequent changes to the main development branch. This fits to the part of the history: the development was not done as a team, but rather individuals serving the needs of separate customers.

Between April and May 2011 the number of most code smells in this group temporarily dropped. The project descriptor was corrupted in both cases. The build system used a forgiving way for extracting information from the project descriptor, but for our tool this made the project appear as if large amounts of files were removed. At the end of 2013, already after agile and continuous integration was introduced, the same problem reappeared while code quality measurements were displayed in publicly available places.

4.3.2 Second correlation group

From the second correlation group we show each code smell separately.

In case of the *Module name in definition* code smell (Figure 3) the trends of `System-1` and `System-2` seems to be added together, and following the growth trend of `System-2`. After the merge the smell occurences of `Merged` followed the growth of `System-2`.

In case of the *Readonly local variable* code smell (Figure 4) the growth trend slowed down after the merge, creating a different trend from that of its source systems. In `System-1` the growth was 118 instances in 2012, and 89 in `System-2`. The trend continued by 9 in 2013 and 11 in 2014 after the merge until the occurrences were greatly decreased at the "Black Thursday" event.

The *Typename in definition* trends (Figrure 5) also slowed down after the merge. The reason behind the drop in `System-1` from around mid 2010 till mid 2011 was a naming convention change.

In the case of the *Unused global definition* code smell the trends in `System-1`

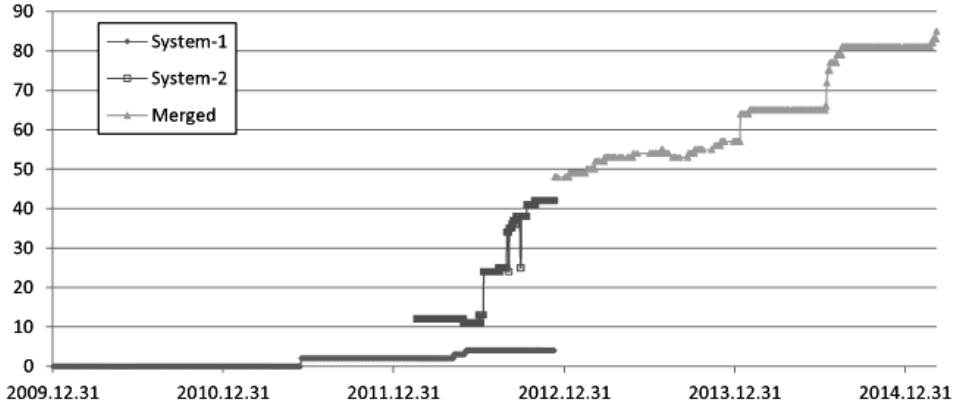

Figure 3: *Module name in definition* smell trends

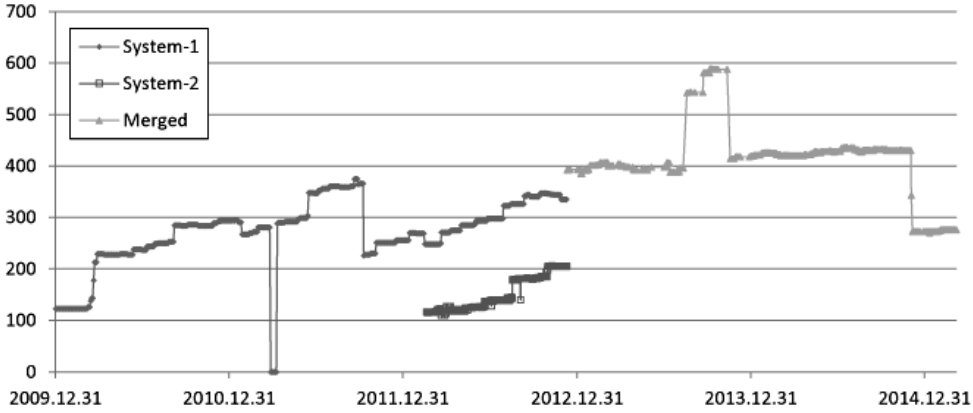

Figure 4: *Readonly local variable* smell trends

continued in `Merged` (Figure 6) also slowed down after the merge. Several instances of this code smell were handled during the "Green Day" and "Black Thursday" events. The corruption of the project descriptor caused a temporal drop in April 2011, and a temporal increase at the end of 2013. In the first case files containing *unused global definitions* disappeared from our measurements, in the second case the files disappearing caused the increase in the number of *unused global definitions*.

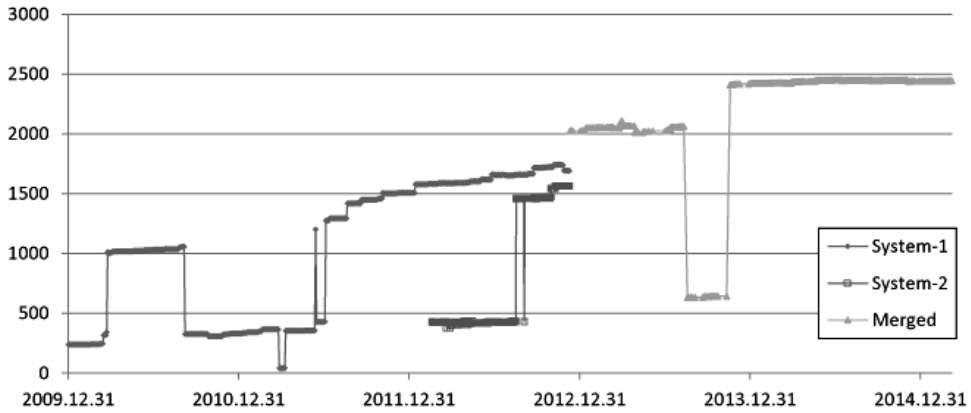

Figure 5: *Typename in definition* smell trends

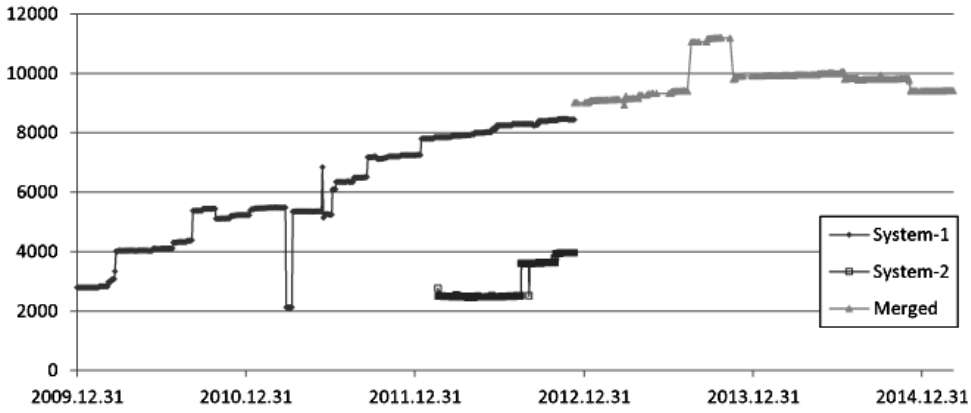

Figure 6: *Unused global definition* smell trends

Circular importation followed a different behavior. In `System-1` the occurrences were rare and stable. In `System-2` their occurrences were higher and changing frequently (this smell is reported for every module in the circle individually in our tool, allowing for small changes in the source leading to large changes in reported numbers of this smell). After the merge the trend stabilized.

In `System-1` the growth was 4 instances in 2012, in `System-2` chaotic till

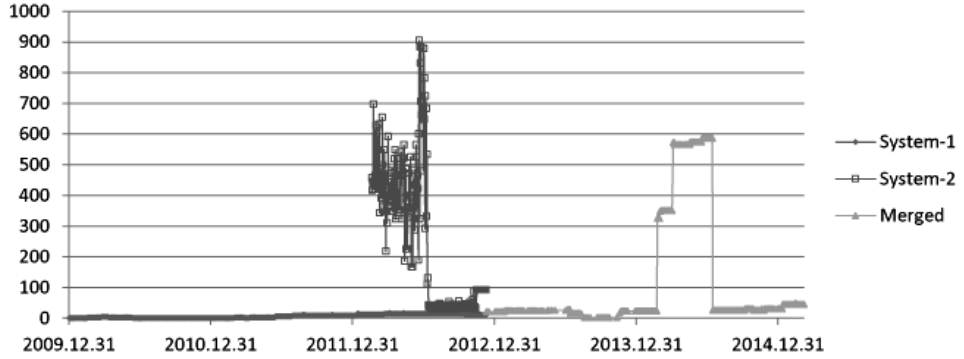

Figure 7: *Circular importation* smell trends

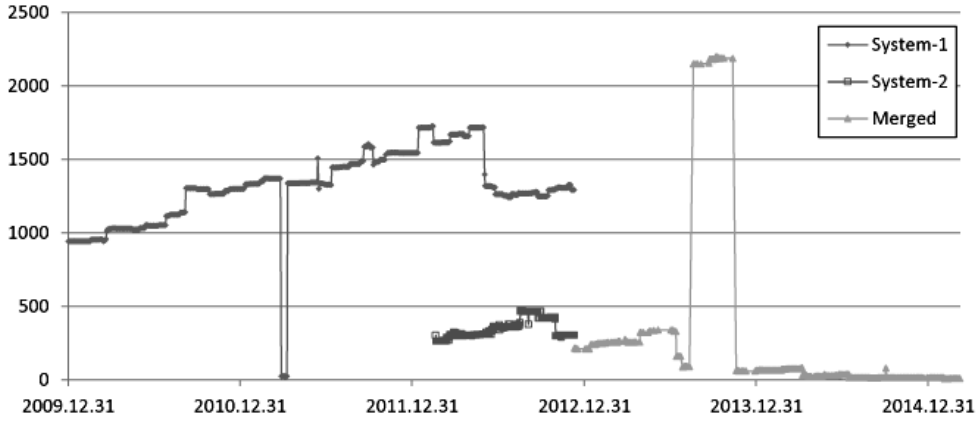

Figure 8: Number of *unused imports* smell trends.

the half of that year. Which continued with 2 in 2013 and 7 in 2014 after the merge. When two libraries developed on separate branches were merged in February and March 2014, the numbers increased to 351 and 566. Only to be reduced to 45 during the "Green Day" event.

The code smells *Readonly local variable*, *Circular importation* and *Unused global definition* were addressed on special events, but only a portion of their numbers could have been corrected.

4.3.3 Third correlation group

From this group we show only the *unused imports* smell trends.

The occurrences of this smell in `System-1` drops from 1717 to 1398 between June and July and to 215 till the end of December 2012 (Figure 8). In `System-2` the occurrences of *unused imports* falls from 420 to 298 on October and to 215 on December, 2012. We found that all of these code quality improvements were related to one employee. After learning that Titanium had support for detecting unused imports she/he decided to clean up some of the code.

Shortly after July 2013 the occurrences of *unused imports* drops from 329 to 84 during the "TitaniumQuest" event.

The large fallback at end of 2013 appeared as an increment of issue numbers. The imports to missing modules were reported as unused.

5 Analysis of our research questions

5.1 Was the number of measured code smells affected by the introduction of Continuous Integration (RQ1)?

Continuous Integration was introduced together with the Agile methodology. As many systems had to adapt to it the process took months, with fine tuning happening even at the time of writing this article. Quality checking was introduced into continuous integration during the "Boost day" (May 2013), with the integration of Titanium.

We found no direct connection between the number of code smells present in the source code and the introduction of quality checking to continuous integration, or continuous integration itself.

Most of the observed code smell occurrences followed the same or similar trends after continuous integration was introduced.

We also observed two cases when project descriptors were corrupted (one before, one after continuous integration was introduced). In neither of the cases did the build and test system notice the corruption. Although during the second case, the code quality displays, driven by continuous integration, showed the changes, they did not evoke immediate action.

Our experience on the influence of using continuous integration aligns with earlier published results of others ([1, 24, 35]).

5.2 Was the number of measured code smells affected by the introduction of tool support for detecting Code Smells (RQ2) ?

We have created Titanium to detect and report internal quality issues. Titanium was integrated into the continuous integration system during the "Boost day" (May 2013). We have organized tutorials: we explained (1) the usage of the tool, (2) the meaning of the reported code smells and (3) what kind of problems the smells can create. In order to reduce the entry barrier of correction we analyzed the observed systems and reported some issues found together with a guide on what to correct, where and how. 73 issues were reported between July 2013 and July 2014 (one year interval) as improvement proposals.

We have found no evidence, breaks in the trends, showing that tool support in itself motivates project members to clean up their code.

Yet, measurements show that, when personal motivation is present, or special events are organized, tool support increases productivity. One person can review and correct numerous of instances of issues otherwise unnoticed.

These results align with the earlier results of others ([24]).

5.3 Was the number of measured code smells affected by the merging of 2 test systems (RQ3) ?

We measured that the merge increased the amount of code smells present and also decreased their previous growth rate.

These results align with the 5th law of software evolution ([16]) and other earlier results ([1, 24, 35]).

It is interesting to note, that the growth of the merged system is between the original growths of the two systems it consists of. At the time of writing, we do not know whether this growth rate will stay longer or will follow one of the original system's growth rate.

5.4 Was the number of measured code smells affected by the different development methodologies (RQ4) ?

During the history of the observed projects the development was performed sometimes by individuals, sometimes by teams. Teams used company specific methods in the beginning, Scrum and Kanban for some time, tailored Agile-like methods for other periods of time.

We have seen that before the middle of 2011 the changes in the numbers

of code smells indicated 3 month development period. After this time the changes became smaller and more frequent. Although this might indicate an effect custom methodologies or maturing in agile methodologies might have had, there was no change in the general trend lines. The changes became more frequent, but followed the same trends in their effects.

Other than the changes becoming more frequent we were not able to find any change correlating to the methodologies, or lack of in our measurements.

5.5 Was the number of measured code smells affected by changing leaders of the projects (RQ5) ?

Conway's law [2] suggests that there is a mirroring effect between the structure of an organization and the structure of the product it creates. In our case there were several organizational changes on the lower levels: teams were formed, team internal processes were changed, system architects were appointed, product ownership changed.

In the measured data we were not able to find any evidence that could be related to these changes. We assume that changes in the immediate leadership were not able to affect the systems. The reason for this is not clear: there could be higher-level organizational structures that binded the immediate leaders, or code smells and lines of code might not correlate with such structures.

Based on the information we collected from the system architects and developers we believe the former assumption. There were no organizational tools in place for enforcing the system architect's guides. Tasks were selected for implementation and prioritized for dedicated developers by the distinct customers they support. This relation might have circumvented the power of technical and managerial leaders.

5.6 Do code smells in test systems follow predictable patterns during the system's evolution (RQ6) ?

In this section we show how our findings detailed in section 4 relate to Lehman's laws of Software Evolution ([16]).

- Our measurements support the 2nd law: in all examined test systems all code smells measured followed an increasing trend unless work was done to reduce them.
- Our measurements support the 4th law: the work rate in each test system studied stayed approximately the same during their whole lifetime.

The invariant work rate was not significantly affected by the changes in history. Lehman showed [12] that although corporate and local management certainly has control over resource allocation and activity targets their ability to do this was constrained by external forces, like the availability of personnel with appropriate skills and trade unions.

- Our measurements support the 5th law: the average incremental growth of successive releases was largely invariant. This property was not affected by most of the changes in history. Only individual efforts and the merge of the two systems has disturbed the trends. Lehman conjectured [17] that this effect is caused by the rate of acquisition of the necessary information by the participants.
- The 8th law is usually proved with showing ripples in the measured data, which are believed to reflect self-stabilization through positive and negative feedback. We believe that the slowdown right after the merge was the result of this feedback mechanism. The merge of the test systems increased the amount of code to be maintained and developed further, but at the same time, the growth trends were somewhat decreased.

6 Threats to validity

This study might suffer from the usual threats to external validity. There might be limits to generalizing our results beyond our settings (programming language used, project setups and possible industry specific effects).

This study was performed on two test systems, developed at the same organization. The field of software evolution studies has limited information sources. Publications in the field analyze only a few open source systems ([6, 10]) and few commercial systems ([11, 17]). Our efforts are an addition to the growing body of knowledge to this field.

To the best of our knowledge these are the first results for the evolution of test systems from software quality point of view, and also the first observation of the effects of products merging. Although it is a valid question if our results can be generalized to other testing languages and domains of software development, we believe this to be true as our results align with previous results in the field of software evolution ([5, 6, 7, 8, 10, 11, 13, 15, 16, 17, 32]).

The study might suffer from not measuring the metrics which were changed by the historical happenings. We have measured several code smells and presented our observations of their changes in this article. These code smells were either collected from a wide range of tools supporting other languages and

adapted to TTCN-3, or defined by us based on our earlier observations related to the language ([29]). We believe that these metrics are correctly measuring internal quality and exhaustive for the TTCN-3 language.

This study also faces the threat of delayed influence: as the work on the studied systems is still going on, it could happen that the influence of some change in the past, will only appear after the publication of this paper. We don't believe this to be a big threat, as the projects studied have been in development for 5 years, our tool support appeared 3 years ago and we have organized several code improvement special events in the last 2 years.

It is an unlikely but theoretically possible scenario that all changes happened at the right time: the changes were necessary to keep the rate of growth; all transitions were smooth and all changes stack up to keep up the same rate of growth.

7 Summary

We have previously defined ([29]) several code smells for test systems written in TTCN-3 and have shown ([28]) that publicly available TTCN-3 test systems have room for improvement. We have also already shown ([26]), that test systems written in TTCN-3, can become large and complex structures. In this article we studied the long term evolution of a large test system in the industry.

We have monitored the development of a test system and measured the code quality characteristics for a five years period at our industry partner. Changing the development processes, project leaders, team and technical leaders, introducing Continuous Integration and automated quality checks did not cause significant difference in the number of code smell instances present. We can conclude that the development of the observed test system follows predictable tendencies.

Just like Lehman's law predicted and observed in [1, 35].

The presence of tool support only made a difference when code smell reductions were the target of personal motivations. According to our observations the best way to improve a software's internal quality is to provide people with dedicated time and tools. This way people, who were already motivated [34, 24, 36, 35], could focus on a few lines of the source code instead of analyzing all of it by hand. This phenomenon was also observed in [24].

Our observation on the evolution of the studied test systems show similarity with the evolution of software systems. This is the main conclusion of the paper.

Acknowledgements

We thank the referee for providing constructive comments and help in improving the contents of this paper.

The authors would like to thank the Faculty of Informatics of Eötvös Loránd University for supporting this research.

We would also like to thank Gábor Jenei, Dániel Poroszkai and Dániel Góbor for their help in implementing features that were crucial to our investigation. Their work allowed us to quickly process large amount of data.

References

[1] A. Chatzigeorgiou, A. Manakos, Investigating the evolution of bad smells in object-oriented code, *Proc. 2010 Seventh International Conference on the Quality of Information and Communications Technology*, QUATIC'10, pp. 106–115, Washington, DC, USA, 2010. IEEE Computer Society. ⇒219, 232, 233, 236

[2] M. E. Conway, How do committees invent?, *Datamation*, **14** ,5 (1968) 28–31. http://www.melconway.com/research/committees.html [accessed 26-Aug-2015]. ⇒234

[3] A. v. Deursen, L. Moonen, A. v. d. Bergh, G. Kok, Refactoring test code, *Proc. 2nd International Conference on Extreme Programming and Flexible Processes* (XP2001), pp. 92–95. University of Cagliari, 2001. ⇒220

[4] M. Fowler, *Refactoring: Improving the Design of Existing Code*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. ⇒218

[5] A. Israeli, D. G. Feitelson, The linux kernel as a case study in software evolution, J. Syst. Softw., **83,** 3 (2010) 485–501. ⇒220, 235

[6] C. Izurieta, J. Bieman, The evolution of freebsd and linux, *Proc. 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ISESE'06, pp. 204–211, New York, NY, USA, 2006. ACM. ⇒220, 235

[7] K. Johari, A. Kaur, Effect of software evolution on software metrics: An open source case study, SIGSOFT Softw. Eng. Notes, **36,** 5 (2011) 1–8. ⇒220, 235

[8] C. F. Kemerer, S. Slaughter, An empirical approach to studying software evolution, IEEE Trans. Softw. Eng., **25,** 4, (1999) 493–509. ⇒220, 235

[9] F. Khomh, M. Di Penta, Y.-G. Gueheneuc, An exploratory study of the impact of code smells on software change-proneness, *Proc. 16th Working Conference on Reverse Engineering*, WCRE'09, pp. 75–84, Washington, DC, USA, 2009. IEEE Computer Society. ⇒218

[10] M. J. Lawrence, An examination of evolution dynamics, *Proc. 6th International Conference on Software Engineering*, ICSE'82, pp. 188–196, Los Alamitos, CA, USA, 1982. IEEE Computer Society Press. ⇒219, 235

[11] M. M. Lehman, *The programming process*, 1969. IBM Research Report RC 2722. ⇒235

[12] M. M. Lehman, Laws of software evolution revisited, *Proc. 5th European Workshop on Software Process Technology*, EWSPT '96, pp. 108–124, London, UK, UK, 1996, Springer-Verlag. ⇒235

[13] M. M. Lehman, *Feast/2 final report – grant number gr/m44101*, 2001. ⇒220, 235

[14] M. M. Lehman, J. F. Ramil, Towards a theory of software evolution - and its practical impact (working paper), *Proc. Intl. Symposium on Principles of Softw. Evolution (invited talk)*, ISPSE 2000, 1-2 Nov, pp. 2–11. Press, 2000. ⇒219

[15] M. M. Lehman, J. F. Ramil, Evolution in software and related areas, *Proc. 4th International Workshop on Principles of Software Evolution*, IWPSE'01, pages 1–16, New York, NY, USA, 2001, ACM. ⇒219, 235

[16] M. M. Lehman, J. F. Ramil, Rules and tools for software evolution planning and management, *Ann. Softw. Eng.,* **11,** 1 (2001) 15–44. ⇒219, 233, 234, 235

[17] M. M. Lehman, J. F. Ramil, D. E. Perry, On evidence supporting the feast hypothesis and the laws of software evolution, *Proc. 5th International Symposium on Software Metrics*, METRICS '98, pp. 84 –, Washington, DC, USA, 1998. IEEE Computer Society. ⇒220, 235

[18] I. Macia, J. Garcia, D. Popescu, A. Garcia, N. Medvidovic, A. von Staa, Are automatically-detected code anomalies relevant to architectural modularity?: An exploratory analysis of evolving systems, *Proc. 11th Annual International Conference on Aspect-oriented Software Development*, AOSD '12, pp. 167–178, New York, NY, USA, 2012, ACM. ⇒218

[19] N. Moha, Y.-G. Gueheneuc, L. Duchien, A.-F. Le Meur, Decor: A method for the specification and detection of code and design smells, IEEE Trans. Softw. Eng., **36,** 1 (2010) 20–36. ⇒218

[20] R. Moser, P. Abrahamsson, W. Pedrycz, A. Sillitti, G. Succi, A case study on the impact of refactoring on quality and productivity in an agile team, in *Balancing Agility and Formalism in Software Engineering*, pp. 252–266, Springer-Verlag, Berlin, Heidelberg, 2008. ⇒218

[21] H. Neukirchen, M. Bisanz, Utilising code smells to detect quality problems in ttcn-3 test suites, *Proc. 19th IFIP TC6/WG6.1 International Conference, and 7th International Conference on Testing of Software and Communicating Systems*, TestCom'07/FATES'07, pp. 228–243, Berlin, Heidelberg, 2007, Springer-Verlag. ⇒218

[22] S. Olbrich D. S. Cruzes, V. Basili, N. Zazworka, The evolution and impact of code smells: A case study of two open source systems, *Proc. 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM '09, pp. 390–400, Washington, DC, USA, 2009. IEEE Computer Society. ⇒218

[23] D. L. Parnas, Software aging, *Proc. 16th International Conference on Software Engineering*, ICSE '94, pp. 279–287, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press. ⇒219

[24] R. Peters, A. Zaidman, Evaluating the lifespan of code smells using software repository mining, *Proc. 2012 16th European Conference on Software Maintenance and Reengineering*, CSMR'12, pp. 411–416, Washington, DC, USA, 2012. IEEE Computer Society. ⇒219, 232, 233, 236

[25] D. I. K. Sjoberg, A. Yamashita, B. Anda, A. Mockus, T. Dyba, Quantifying the effect of code smells on maintenance effort, *IEEE Trans. Softw. Eng.*, **39,** 8 (2013) 1144–1156. ⇒218

[26] K. Szabados, Structural analysis of large ttcn-3 projects. *Proc. 21st IFIP WG 6.1 International Conference on Testing of Software and Communication Systems and 9th International FATES Workshop*, TESTCOM '09/FATES '09, pp. 241–246, Berlin, Heidelberg, 2009, Springer-Verlag. ⇒217, 220, 236

[27] K. Szabados, *Titanium*, https://projects.eclipse.org/proposals/titan, 2015. [Online; accessed 26-Aug-2015]. ⇒221

[28] K. Szabados, A. Kovács, Advanced ttcn-3 test suite validation with titan, *Proc. 9th Conference on Applied Informatics*, pp. 273–281, 2014. ⇒217, 236

[29] K. Szabados, A. Kovács, Test software quality issues and connections to international standards, *Acta Universitatis Sapientiae, Informatica*, **5,** 1 (2014) 77–102. ⇒217, 220, 236

[30] K. Szabados and A. Kovács, *Up-to-date list of code smells*, `http://compalg.inf.elte.hu/~attila/TestingAtScale.htm`, 2015. [Online; accessed 26-Aug-2015]. ⇒220, 224, 225

[31] C. Taube-Schock, R. J. Walker, I. H. Witten, Can we avoid high coupling?, *Proc. 25th European Conference on Objectoriented Programming*, ECOOP'11, pp. 204–228, Berlin, Heidelberg, 2011, Springer-Verlag. ⇒220

[32] W. M. Turski, The reference model for smooth growth of software systems revisited, *IEEE Trans. Softw. Eng.,* **28,** 8, (2002) 814–815. ⇒220, 235

[33] E. Van Emden, L. Moonen, Java quality assurance by detecting code smells, *Proc. Ninth Working Conference on Reverse Engineering* (WCRE'02) pp. 97–106, Washington, DC, USA, 2002. IEEE Computer Society. ⇒218

[34] A. Yamashita, L. Moonen, Do code smells reflect important maintainability aspects?, *Proc. 2012 IEEE International Conference on Software Maintenance*, ICSM '12, pp. 306–315,Washington, DC, USA, 2012. IEEE Computer Society. ⇒219, 236

[35] A. Yamashita, L. Moonen, Do developers care about code smells? an exploratory survey, *Proc. 20th Working Conference on Conference: Reverse Engineering*, pp. 242–251. IEEE Computer Society, 2013. ⇒219, 232, 233, 236

[36] A. Yamashita, L. Moonen, Exploring the impact of inter-smell relations on software maintainability: An empirical study, *Proc. 2013 International Conference on Software Engineering*, ICSE '13, pp. 682–691, Piscataway, NJ, USA, 2013. IEEE Computer Society Press. ⇒219, 236

[37] A. Yamashita, L. Moonen, *To what extent can maintenance problems be predicted by code smell detection? - an empirical study*, *Inf. Softw. Technol.*, **55,** 12 (2013) 2223–2242. ⇒218

[38] A. Zaidman, B. Rompaey, A. Deursen, S. Demeyer, Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining, *Empirical Softw. Engg.*, **16,** 3 (2011) 325–364. ⇒ 220

[39] B. Zeiß, D. Vega, I. Schieferdecker, H. Neukirchen, J. Grabowski, Applying the ISO 9126 Quality Model to Test Specifications–Exemplified for TTCN-3 Test Specifications, *Software Engineering 2007*, *Lecture Notes in Informatics*, Copyright Gesellschaft für Informatik, Mar. 2007. ⇒220

Received: June 9, 2016 • Revised: August 20, 2016

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **8,** 2 (2016) 241–253

DOI: 10.1515/ausi-2016-0011

Satellite image fusion using fuzzy logic

Suda KUMARASWAMY
Department of IT, VNRVJIET, Hyderabad, India
email: kumaraswamy_it@vnrvjiet.ac.in

Dammavalam SRINIVASA RAO
Department of IT, VNRVJIET, Hyderabad, India
email: srinivasarao_d@vnrvjiet.ac.in

Nuthanapati NAVEEN KUMAR
CSE department, SIT,JNTU Hyderabad, India
email: naveen.cse.mtech@jntuh.ac.in

Abstract. Image fusion is a method of combining the Multispectral (MS) and Panchromatic (PAN) images into one image contains more information than any of the input. Image fusion aim is to decrease unknown and weaken common data in the fused output image at the same time improving necessary information. Fused images are helpful in various applications like, remote sensing, computer vision, biometrics, change detection, image analysis and image classification. Conventional fusion methods are having some side effects like assertive spatial information and uncertain color information is an usually the problem in PCA and wavelet transform based fusion is a computationally in depth process. In order to overcome these side effects and to propose alternative soft computing fusion approach for conventional fusion methods we exploit image fusion using fuzzy logic technique to fuse two source images obtained from different sensors to enhance both spectral and spatial information. The proposed work here further compared with two common fusion methods like, principal component analysis (PCA) and wavelet transform along with quality assessment metrics. Exploratory outputs demonstrated in

Computing Classification System 1998: I.4.3, I.4.9
Mathematics Subject Classification 2010: 68U10
Key words and phrases: Fusion, MS, PAN, Fuzzy Logic, PCA, Wavelet Transform

order that fuzzy based image fusion technique can actively retains more information compared to PCA and wavelet transform approaches while enhancing the spatial and spectral resolution of the satellite images.

1 State-of-the art

The coordinates in the small frequency region with high incisiveness core dimensions are chosen as coefficients of the output image, and a most adjacent intensity stationed fusion method is presented to choose giant frequency coordinates [3]. In [15], novel image fusion approach for confining image sensor network is presented where the sharpen density of the fresh compressive value are not acquired from the arbitrary specimen data still in distinction to the chosen Hadamard conjoined with whatever can additionally be generated against constrict imaging process adequately. Fuzzy based fusion approach compared with discrete wavelet transform (DWT) and weighted average DWT using Genetic algorithm (GA) approaches and shown that fuzzy based image fusion technique out performs DWT and DWT using GA approaches. Image fusion using Pulse-Coupled Neural Network (PCNN) is proposed where input images are flattened through scrambled block Hadamard ensemble (SBHE) in compressed domain and local standard variance is input to drive PCNN and coefficients with huge ignited times are chosen as the fusion coefficients. Later fusion coefficients are whipped by sliding window in order to avoid blocking effect [16]. In [10], image fusion using fuzzy logic and neuro fuzzy logic approaches are compared and concluded that in some cases fuzzy based fusion results gave better results in some other cases neuro fuzzy based fusion generated better results. . A new method of satellite image fusion have been build on Otsu's Multi-thresholding approach in two stages, i) shearlet transform is used Panchromatic and multi-spectral image distinctly, ii) the revised low frequency sub-band shearlet coefficients obtained from shearlet transform are composed by the Otsu's Multi-thresholding approach and choose most low-pass band naturally [2]. A innovative multifocus image fusion approach [17] built on human visual system and neural network back propagation given with three facets which echo brightness of a pixel are extracted first and used to train a BP neural network to decide the clarity pixel. Those pixels are then used to build the initial fused image. Later the focused regions are identified by calculating the coincidence in mid of satellite images and the first time fused image proceed by morphological operations and the final fused image is attained by applying a fusion rule for those concentrated regions. In [8], novel

fusion method is introduced to invent full utilization of structural compactness for fusion of the common and structured layers. In [4], authors demonstrated a new fusion technique where it separates the input image decomposition technique into two consecutive filtrated activities by applying spectral factorization filter. The concrete image fusion attain after involution along with the early filter couple. Its important lower guide volume directed to the miniaturize of the undesirable expansion of conjoined values about overlaying image peculiarities. In [6], a technique proposed for straight virtue evaluation of fusion process placed on the assessment of triple major elements of output image quality like diversity storage, incisiveness and anatomy preservation. Intuitive analysis is postured to construct a database with fusion to evaluate the achievement of the fusion process.

2 Wavelet transform based image fusion

In [7], wavelet transform applied a structure in which a input image is decomposed, where individual plain correlating to a mean decision, or reduced periodicity strip. Fusion using this methodology is a category of input model that can allow the density contended about the input at appropriate moment. The framework for wavelt based fusion illustrated in Fig. 1.

Algorithm for wavelet based image fusion [5]

1. Take two input images, K1 and K2 to be fused.

2. Apply the wavelet decomposition process on the two input images.

3. Employ the pixel based approach for similarity whatever contain fusion situated on considering the higher valued image pixels from likeness of source images K1and K2.

4. Placed on higher valued image pixels ,a binate determination map is produced and it gives the decision rule for the conduction fusion of two input images K1and K2.

5. The output fused transform interrelated to similarity over higher selection pixel rule is generated.

6. Connecting of fused resemblance and particulars produces the new coefficient matrix.

7. Execute the inverse wavelet transform process to build the output fused image.

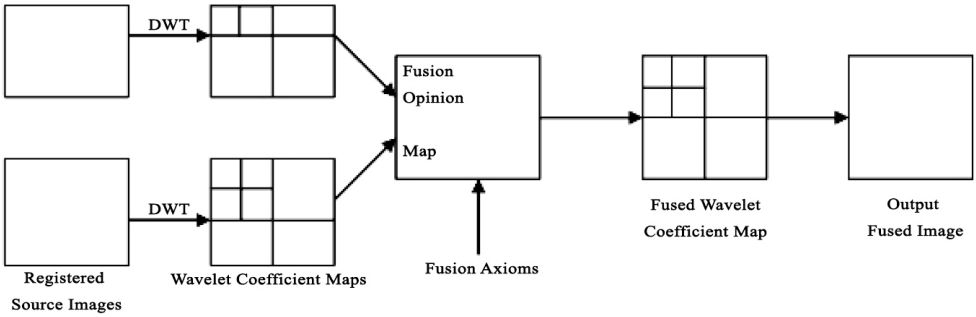

Figure 1: The generic structure for wavelet transform based image fusion

3 Principal component analysis based image fusion

A numerical concept that transforms a number of correlated input variables into a numeral unassociated variables through the Principal Component Analysis (PCA). Algorithm steps in PCA concept is as follows

Algorithm steps:

1. Input image are transformed in to column vectors initially.
2. Covariance matrix is calculated from two column vectors.
3. Compute eigenvalues and the corresponding eigenvectors.
4. Normalize both the characteristic values and characteristic vectors.
5. Through fusion process on two scaled matrices, final fused image matrix is generated.

We consider the input images denoted by A(i, j) and B (i, j) and convert these images in to equivalent dual column vectors and means are subtracted. The dimension of the output is m x 2, here m is the magnitude of the image. The eigenvalues and conform eigenvectors considering output is calculated also compute the eigenvectors correlated to the greater eigenvalues. P1 and P2 are normalized components computed from covariance matrix to obtain eigenvector and the fused image is obtained from it [12].

4 Fuzzy logic based image fusion

Two registered input images are used in the fusion process. Fuzzy logic properties are utilized to perform fusion. An innovative image fusion for in multi-view over the-wall radar imaging system to compute the variation among pixels us-

ing a local operator and concluded that method performs well compared to conventional fusion approaches [13]. A different method is proposed to fuse images by utilizing maximum, minimum operations in intuitionist fuzzy sets (IFSs). Entropy metric is used to generate the most favorable value of the parameter in membership functions. Later resulting IFIs are decomposed into image sections and the correlated sections of the images are combined by computing blackness and whiteness of the blocks [1]. An algorithm for image fusion is conferred stands on fuzzy logic and wavelet transform and evaluate the pixel-level image fusion approaches, and focus on a technique based on the discrete wavelet transform and fuzzy logic approaches. As part of the fusion process two fuzzy relations are determined and predicted the essence of each one wavelet coefficient with fuzzy hypothesis. Based on the priority of coefficients, the weighting average coefficients were computed. Finally the fused image is obtained through inverse wavelet transform operation [18]. A new image fusion technique based on fuzzy logic and Discrete Wavelet Transform (DWT). The fuzzy membership functions and fuzzy rules are composed properly to perfect adaption for the fusion of multifocus images. DWT has been utilized to enhance the attainment as fuzzy logic is practiced at every stage of DWT to perform fusion on similar coefficients [9].

4.1 Image processing with fuzzy logic

Image processing with fuzzy approach has triple essential steps. The encoding of input image and decoding the fused image are important stages that are get ready to operate the original image using fuzzy techniques. After the inputs are converted to the associates uniform adapted procedures update the fellows weights [11].

4.2 Procedure for fuzzy based image fusion

Fuzzy rules, Membership Functions (MSF) are utilized in the fusion process [12]

Axiom-1: $[I/P_1 isMSF_3] or [I/P_2 isMSF_3] \rightarrow [O/P_1 isMSF_2]$

Axiom-2: $[I/P_1 MSF_1] or [I/P_2 isMSF_3] \rightarrow [O/P_1 isMSF_1]$

Axiom-3: $[I/P_1 MSF_3] or [I/P_2 isMSF_2] \rightarrow [O/P_1 isMSF_3]$

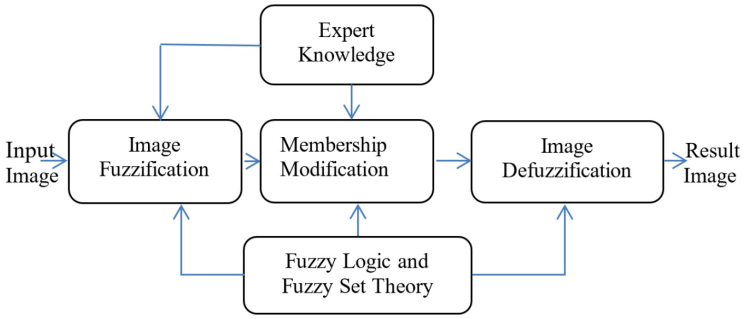

Figure 2: Block diagram for fuzzy image processing

Axiom-4: $[I/P_1 MSF_2] or [I/P_2 is MSF_2] \rightarrow [O/P_1 is MSF_2]$

Axiom-5: $[I/P_1 MSF_2] or [I/P_2 is MSF_2] \rightarrow [O/P_1 is MSF_2]$

Axiom-6: $[I/P_1 MSF_1] or [I/P_2 is MSF2] \rightarrow [O/P_1 is MSF_1]$

Sequence of steps in fuzzy logic based image fusion as follows [15].

1. Get the first image in K1 and its size (rows: row1, columns: col1).
2. Get the second image in K2 and its size (rows:row2, columns: col2).
3. Images K1 and K2 are in matrix form and each pixel value is in between 0-255. Apply Grey Colormap.
4. Select two input images which are in same size.
5. Transform two input images in column matrix which has S= rowl*coll entries.
6. Prepare a fis (Fuzzy) file, which has two input images.
7. Determine fuzzy membership functions for input images to be fused by adapting the membership functions
8. Prepare fuzzy rules for the fusion process
9. For num=l to S in steps of one, utilize fuzzification step by employing the fuzzy rules on input images

10. Transform the column form to matrix form and display the output fused image.

5 Quality metrics

Quality assessment parameters are applied to assess the fused image obtained from the fusion operation.

5.1 Quality index (QI)

QI calculates the affinity between two images (A & B) and QI is equivalent to 1 if both the images are exact [11]

$$QI = \frac{m_{ab}\ 2xy\ 2m_a\ 2m_b}{m_a m_b x^2 + y^2 m_a^2 + m_b^2} \tag{1}$$

where input images (A & B) mean values are denoted by x, y and variance, covariance of images are denoted by , M_a^2, M_b^2, and M_{ab},QI indicates the amount of the information presented in reference image has been converted into the output fused image. The ideal value 1 indicates fused image and reference images are similar.

5.2 Mutual information measure (MIM)

MIM contains the mutual information between A (i, j) and B (i, j) input images,

$$I_{AB} = \sum_{x,y} P_{AB}(x,y) log \frac{P_{AB}(x,y)}{P_A(x), P_B(y)} \tag{2}$$

where, P_A (x) and P_B (y) are the probability in the individual images, and P_{AB} (x, y)) is joint probability, higher value indicates better fused image quality.

5.3 Fusion factor (FF)

Two input images are A,B and F is their fused image [14], then

$$FF = I_{AF} + I_{BF} \tag{3}$$

where MIM values between input images and used image are denoted by I_{AF} and I_{BF} respectively. Maximum value of FF denotes that output fused image

consists of reasonably superior amount of information existent in both the images.

5.4 Fusion symmetry (FS)

FS is a notion of the intensity of equivalence in the image content of two images.

$$I_{AB} = abs\left(\frac{I_{AF}}{I_{AF} + I_{BF}} - 0.5\right) \tag{4}$$

Lower FS value indicates that the fused image obtains features from both source images.

5.5 Fusion index (FI)

Based on two fusion metrics, fusion symmetry and fusion factor the fusion index, FI is calculated as

$$FI = \frac{I_{AF}}{I_{BF}} \tag{5}$$

where IAF denotes mutual information between MS image and fused image and IBF is the mutual information between PAN image and fused image. The quality of fusion approach indicated by the degree of fusion indEx.

5.6 Root mean square error (RMSE)

The RMSE calculates the intensity of the pixel difference obtained from the fusion process.

$$RMSE = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(R(i,j) - F(i,j))} \tag{6}$$

lower RMSE value indicates better fusion approach.

5.7 Peak signal to noise ratio (PSNR)

PSNR can be determined by

$$PSNR = 20\ \log_{10}\ \left(\frac{G^2}{MSE}\right) \tag{7}$$

where G is the intensity of gray in the fused image, maximum PSNR value denotes better fused image quality.

Table 1: Quality metrics for outputs obtained from conventional and proposed approaches

Approach	QI	FF	FS	FI	MIM	RMSE	PSNR	E
Wavelet								
(Ex. 1)	0.9463	3.7629	0.0529	1.0779	1.6554	62.5529	11.2425	7.3415
(Ex. 2)	0.8550	3.7832	0.0218	0.8938	1.9775	18.8999	22.3648	7.1134
(Ex. 3)	0.9358	0.9278	0.0128	1.0527	0.4520	20.7690	21.7825	7.1454
(Ex. 4)	0.9425	1.2222	0.0391	1.0882	0.5848	13.8566	25.2977	7.2511
PCA								
(Ex. 1)	0.9450	1.5650	2.7964	0.0765	1.2913	9.4765	27.9806	7.2721
(Ex. 2)	0.9876	1.5890	3.2885	0.0110	0.7932	17.440	19.9290	7.3228
(Ex. 3)	0.9353	1.2194	0.0149	0.9422	0.8341	25.0039	20.2047	7.4532
(Ex. 4)	0.9397	1.5546	0.0402	1.1749	1.3800	25.6658	13.3336	7.4213
Fuzzy								
(Ex. 1)	0.9689	5.5324	0.2552	3.2875	4.3345	12.2001	23.8336	7.3865
(Ex. 2)	0.9955	8.6207	0.0498	1.2826	3.8823	16.8785	23.2336	7.3577
(Ex. 3)	0.9470	1.2919	0.0092	1.3133	0.9870	20.3423	21.8432	7.4322
(Ex. 4)	0.9431	1.5948	0.0384	1.2404	1.4086	21.7479	20.0220	7.4391

5.8 Entropy (E)

Entropy represents the quality of the source image. Entropy is a amount of volatility that can be used to discrminate the texture of the input image

$$E = -\sum p * log_2(p) \tag{8}$$

where p represents the scatter diagram count.

6 Results analysis

In this paper, we fused a MS and PAN images using our fuzzy based fusion algorithm. Ex. 1, Ex. 2 MS and PAN images, Hyderabad captured from LISS III. Ex. 2 images are collected from http://www.metapix.de/examples_r.htm [8], Ex. 3 images are taken from the NRSC test samples.

The fuzzy logic based image fusion process has been carried out using Matlab 10.0. In order to implement fuzzy based image fusion required fuzzy membership functions and fuzzy rules are tuned and determined precisely. Because of the potentiality of the fuzzy logic, similarity between fused image and reference image denoted by IQI value (0.9689, 0.9955, 0.9470 and 0.9431) obtained

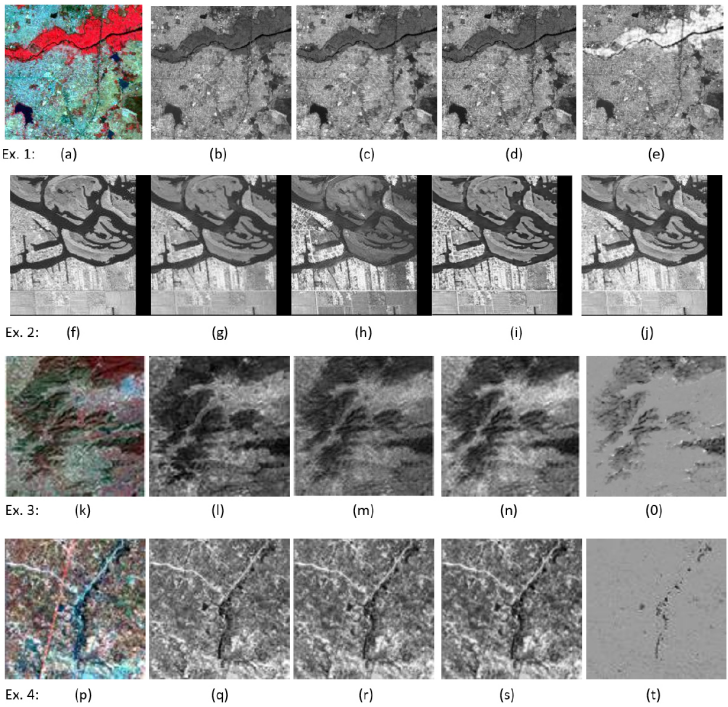

Figure 3: Case study: (a), (b), (f), (g),(k),(l),(p),(q): input images; (c), (h),(m),(r) are output images from wavelet transform and (d), (i),(n),(s) are fused from PCA and (e), (j),(o) and (t) fused images obtained from proposed fuzzy approach.

from fuzzy based fusion having better values compared to IQI values (0.9463, 0.8850, 0.9358 and 0.9425) from wavelet based fusion and IQI values (0.9450, 0.9876, 0.9353 and 0.9357) obtained from PCA based fusion approaches respectively. Typical assessment parameters like fusion factor values (5.5324, 8.6207, 1.2919 and 1.5948) and fusion index values (3.2875, 1.2826, 1.3133 and 1.2404) are also having better values indicates that proposed fuzzy based image fusion approach enhanced the fusion quality compared to traditional fusion methods. Higher values for FF obtained from proposed method indicates that information contained in the fuzzy based fused image is more possessing extremely good quality results compared with the wavelet transform and PCA based fusion approaches. Higher values for FI metric generated from the fuzzy based fusion approach indicates that fusion degree is higher for proposed method compared to other methods mentioned. Higher value for PSNR and Entropy obtained from proposed fuzzy based fusion method indicates that amount of information in the fused image is high compared to wavelet and PCA based fusion approaches respectively. Table 1 demonstrated that proposed fuzzy based fusion approach has exhibited conditionally more effective in QI, MIM and Entropy values while improving spectral and spatial information as well. Substantial variances are generated through fuzzy based fusion with lower values for RMSE, FS and having greater values for FF, FI and PSNR assessment metrics. Hence it is concluded from experimentation outputs that image fusion using fuzzy logic scheme out performs conventional wavelet transform and PCA based fusion approaches.

7 Conclusion and future work

In this paper, fuzzy logic based image fusion for satellite images obsolete conferred. The result analysis certainly proves that the proposed fuzzy logic based fusion provides a huge progress on the attainment of the process. The proposed approach can be applied iteratively and also applied to all categories of images and to integrate conclusive assessment parameter of different image fusion approaches. Classification of fused images may also improve accuracy in remote sensing objectives. So it has been examined from experimental outcomes that proposed fuzzy based image fusion algorithm conserve superior spatial and spectral information and also improved visual essence compared to conventional fusion methods, wavelet trans from and PCA methods.

References

[1] P. Balasubramaniam, V. P. Ananthi: Image fusion using intuitionistic fuzzy sets, *Elsever Journal of Information Fusion* **20** (2014) 21–30. ⇒245

[2] B. Biswas, K. N. Dey, A. Chakrabarti, Remote sensing image fusion using multithreshold Otsu method in Shearlet domain, *Procedia Computer Science* **57,** (2015) 554–562. ⇒242

[3] Y. Chen, Z. Qin, PCNN-based image fusion in compressed domain, *Mathematical Problems in Engineering*, Vol. 2015. ⇒242

[4] A. Ellmauthaler, C. L. Pagliari, E. A. B. da Silva, Multiscale image fusion using the undecimated wavelet transform with spectral factorization and nonorthogonal filter banks, *IEEE Transactions on Image Processing*, **22,** 3 (2013) 1005–1017. ⇒243

[5] D. L. A. Godse, D. S. Bormane, Wavelet based image fusion using pixel based maximum selection rule, *International Journal of Engineering Science and Technology*, **3,** 7, (2011) 5572–5557. ⇒243

[6] R. Hassen, Z. Wang, M. M. A. Salama, Objective quality assessment for multiexposure multifocus image fusion, *IEEE Transactions on Image Processing* **24,** 9 (2015) 2712–2724. ⇒243

[7] K. Kannan, S. A. Perumal, K. Arulmozhi, Performance comparision of various levels of fusion of multi-focused images using wavelet transform, I. J. Computer Applications, **1,** 6 (2010) 71–78. ⇒243

[8] S. Li, X. Kang, J. Hu, Image fusion with guided filtering, *IEEE Transactions on Image Processing*, **22,** 7 (2013) 2864–2875. ⇒242, 249

[9] A. N. Myna, J. Prakash, A novel hybrid approach for multi-focus image fusion using fuzzy logic and wavelets, *International Journal of Emerging Trends and Technology in Computer Science*, (IJETTCS), **3,** (2014) 131–138. ⇒245

[10] D. S. Rao, M. Seetha, M. H. M. Krishna Prasad, Comparison of fuzzy and neuro fuzzy image fusion techniques and its applications, *International Journal of Computer Applications*, **43,** 20 (2012) 31–37. ⇒242

[11] D. S. Rao, M. Seetha, M. H. M. Krishna Prasad, Quality assessment of pixel-level image fusion using fuzzy logic, *International Journal on Soft Computing*, **3,** 1, (2012) 13–25. ⇒245, 247

[12] D. S. Rao, M. Seetha, M. H. M. Krishna Prasad, Novel approach for iterative image fusion using fuzzy and neuro fuzzy logic, *International Journal of Geoinformatics* **11,** 2 (2015) 29–39. ⇒244, 245

[13] C. H. Seng, A. Bouzerdoum, F. H. C. Tivive, M. G. Amin, Fuzzy logic-based image fusion for multi-view through-the-wall radar, *Int. Conf. Digital Image Computing: Techniques and Applications (DICTA)*, 2010, pp. 423–428. ⇒245

[14] M. Seetha, I. V. Murali Krishna, B. L. Deekshatulu, Data fusion performance analysis based on conventional and wavelet transform techniques, *IEEE Proceedings on Geoscience and Remote Sensing Symposium* **4** (2005) 2842–2845. ⇒247

[15] C. Yang, B. Yang, Efficient compressive multi-focus image fusion, *Journal of Computer and Communications*, **2** (2014) 78–86. ⇒242, 246

[16] Y. Yang, S. Huang, J. Gao, Z. Qian: Multi-focus image fusion using an effective discrete wavelet transform based algorithm, *Measurement Science Review,* **14,** 2 (2014) 102–108. ⇒242

[17] Y. Yang, W. Zheng, S. Huang, Effective multifocus image fusion based on HVS and BP neural network, *The Scientific World Journal*, 2014, Article ID 281073, 10 pages. ⇒242

[18] M. Zhu, Y. Yang, A new image fusion algorithm based on fuzzy logic, *International Conference on Intelligent Computation Technology and Automation*, (ICICTA) 2008, pp. 83–86. ⇒245

Received: July 8, 2016 • Revised: November 27, 2016

Dumitru Dumitrescu
(1949–2016)

Our colleague, friend, editor of Acta Universitatis Sapientiae, Informatica, professor Dumitru Dumitrescu passed away after a long illness on October 14, 2016. He was 67.

Professor Dumitrescu obtained his MSc degree in Theoretical Physics in 1972, MSc degree in Mathematics in 1979, then PhD in Mathematics in 1990 from Babeş-Bolyai University, Cluj, where he worked until his death as professor in Informatics.

Among his research interests in Artificial Intelligence we can mention: fuzzy systems and fuzzy logic, neural networks, fuzzy clustering and pattern recognition, evolutionary computation, intelligent control, non-standard logics and their application in AI, cognitive sciences, biologically inspired systems.

In the last part of his life he worked on Complex Systems at CONEURAL (Centre for Cognitive and Neural Studies), Romanian Institute of Science and Technology, Cluj-Napoca.

With a terrible feeling of pain and loss, we say goodbye to our friend. We shall treasure his memory.

The Editorial Board

Acta Universitatis Sapientiae

The scientific journal of Sapientia Hungarian University of Transylvania publishes original papers and surveys in several areas of sciences written in English.
Information about each series can be found at
http://www.acta.sapientia.ro.

Acta Universitatis Sapientiae, Informatica

Each volume contains two issues.

Sapientia University
De Gruyter Open
Scientia Publishing House

ISSN 1844-6086
http://www.acta.sapientia.ro

Information for authors

Acta Universitatis Sapientiae, Informatica publishes original papers and surveys in various fields of Computer Science. All papers are peer-reviewed.

Papers published in current and previous volumes can be found in Portable Document Format (pdf) form at the address: http://www.acta.sapientia.ro.

The submitted papers should not be considered for publication by other journals. The corresponding author is responsible for obtaining the permission of coauthors and of the authorities of institutes, if needed, for publication, the Editorial Board is disclaiming any responsibility.

Submission must be made by email (acta-inf@acta.sapientia.ro) only, using the LaTeX style and sample file at the address http://www.acta.sapientia.ro. Beside the LaTeX source a pdf format of the paper is necessary too.

Prepare your paper carefully, including keywords, ACM Computing Classification System codes (http://www.acm.org/about/class/1998) and AMS Mathematics Subject Classification codes (http://www.ams.org/msc/).

References should be listed alphabetically based on the Intructions for Authors given at the address http://www.acta.sapientia.ro.

Illustrations should be given in Encapsulated Postscript (eps) format.

Contact address and subscription:
Acta Universitatis Sapientiae, Informatica
RO 400112 Cluj-Napoca
Str. Matei Corvin nr. 4.
Email: acta-inf@acta.sapientia.ro

Printed by Idea Printing House
Director: Péter Nagy

ISSN 1844-6086
http://www.acta.sapientia.ro