When Stars Control a Grammar's Work

Henning Fernau

Abteilung Informatikwissenschaften
Universität Trier, Germany

fernau@uni-trier.de

Lakshmanan Kuppusamy

School of Computer Science and Engineering
VIT, Vellore, India

klakshma@vit.ac.in

Indhumathi Raman

Department of Computing Technologies,
School of Computing, SRMIST, Chennai, India

indhumar2@srmist.edu.in

Graph-controlled insertion-deletion (GCID) systems are regulated extensions of insertion-deletion systems. Such a system has several components and each component contains some insertion-deletion rules. The components are the vertices of a directed control graph. A rule is applied to a string in a component and the resultant string is moved to the target component specified in the rule. The language of the system is the set of all terminal strings collected in the final component. We impose the restriction in the structure of the underlying graph to be a star structure where there is a central, control component which acts like a master and transmits a string (after applying one of its rules) to one of the components specified in the (applied) rule. A component which receives the string can process the obtained string with any applicable rule available in it and sends back the resultant string only to the center component. With this restriction, we obtain computational completeness for some descriptional complexity measures.

1 Introduction

Insertion-deletion systems are part of formal languages which are extensively analyzed. The motivation for the systems comes from both linguistics [14, 16] and molecular biology. The action of inserting or deleting some strands do occur often in DNA processing [17] and RNA editing [2]. These two operations together were introduced as a formal languages theory framework in [11] and further studied in [10, 19]. The corresponding grammatical mechanism is called *insertion-deletion system* (abbreviated as ins-del system). The insertion operation means inserting a string η in between the strings w_1 and w_2, whereas the deletion operation is deleting a substring δ from the string $w_1 \delta w_2$.

In the literature, several variants of ins-del systems have been considered. We refer to the survey article [20] for details concerning the state-of-the-art around 2010. One of the important variants of ins-del systems is *graph-controlled ins-del systems* (abbreviated as GCID systems), introduced in [6] and further studied in [8]. In such a system, the concept of *components* is introduced, which are associated with insertion or deletion rules. The transition is performed by choosing any applicable rule from the set of rules of the current component and by moving the resultant string to the target component specified in the rule in order to continue processing it. Several restrictions of graph control have been studied, e.g., matrix ins-del systems (see [18, 5] and more papers cited there), time-varying ins-del systems [1], or path-controlled ins-del systems [4]. In this paper, we consider star control (which also has been considered in [9] in an implicit way when dealing with graph-controlled systems with two components. This models a kind of master-slave system in the sense that the central component always dispatches work to the slave components who, after finishing their work, return the result to the master component.

k	$=$	the number of components			
n	$=$	$\max\{\|\eta\|: (i,(u,\eta,v)_I, j) \in R\}$	m	$=$	$\max\{\|\delta\|: (i,(u,\delta,v)_D, j) \in R\}$
i'	$=$	$\max\{\|u\|: (i,(u,\eta,v)_I, j) \in R\}$	j'	$=$	$\max\{\|u\|: (i,(u,\delta,v)_D, j) \in R\}$
i''	$=$	$\max\{\|v\|: (i,(u,\eta,v)_I, j) \in R\}$	j''	$=$	$max\{\|v\|: (i,(u,\delta,v)_D, j) \in R\}$

Table 1: Size $(k;n,i',i'';m,j',j'')$ of a GCID system

Graph-controlled insertion-deletion systems whose underlying control graph is a tree are equivalent to ins-del P systems [13, 8]. Hence, our restriction can be viewed as a special case of ins-del P systems. We want to point to one technicality here: with P systems (and similarly with several restrictions of graph control), there is the possibility to stay in the same membrane with the *here* command (which corresponds to allowing loops in graph control); in the model that we consider in this paper, this is disallowed: when the master saw the 'current work' (and worked on it one step), it has to pass it to some slave immediately, and after the slave performed one step, the work is handed back to the master, etc. Therefore, results in the literature concerning seemingly related models do not always compare well to this star model.

The *descriptional complexity* of a GCID system is measured by its *size* $s = (k;n,i',i'';m,j',j'')$, where the parameters represent resource bounds as given in Table 1. Slightly abusing notation, the language class generated by GCID systems of size s is denoted by GCID(s). We attach subscripts P and S when referring to path-controlled or star-controlled GCID systems, respectively.

The main results of this paper are the following ones.

1. (Theorem 9) $RE = GCID_S(6;1,1,0;2,0,0) = GCID_S(6;1,0,1;2,0,0)$;

2. (Theorem 11) $RE = GCID_S(4;2,1,1;1,0,0)$.

Our proofs are based on the *Special Geffert Normal Form* of type-0 grammars, which characterizes the class RE, the recursively enumerable languages. Formal definitions follow in the next section.

2 Preliminaries

We assume that the readers are familiar with the standard notations used in formal language theory. Here, we recall a few notations for the understanding of the paper. Let $\mathbb{N}$ denote the set of positive integers, and $[\ell \ldots k] = \{i \in \mathbb{N}: \ell \le i \le k\}$. Given an *alphabet* (finite set) Σ, Σ^* denotes the free monoid generated by Σ. The elements of Σ^* are called *strings* or *words*; λ denotes the empty string. For a string $w \in \Sigma^*$, $|w|$ is the length of w and w^R denotes the reversal (mirror image) of w. L^R and $\mathscr{L}^R$ are also understood for languages L and language families $\mathscr{L}$, collecting all reversals of words from L and all reversals of languages from $\mathscr{L}$, respectively. For the computational completeness results, as our main tool we use the fact that type-0 grammars in Special Geffert Normal Form (SGNF) describe RE.

Definition 1 *A type-0 grammar $G = (N,T,P,S)$ is said to be in SGNF if*

- *N decomposes as $N = N' \cup N''$, where $N'' = \{A,B,C,D\}$ and N' contains at least the two nonterminals S and S';*

- *the only non-context-free rules in P are $AB \to \lambda$ and $CD \to \lambda$;*

- *the context-free rules are of the form (i) $S' \to \lambda$, or (ii) $X \to Y_1Y_2$, where $X \in N' \setminus \{S'\}$ and $Y_1Y_2 \in ((N' \setminus \{X\})(T \cup \{B,D\})) \cup (\{A,C\}(N' \setminus \{X\}))$.*

The way the normal form is constructed is described in [6], based on [7]. We can assume that S' does not appear on the left-hand side of any non-erasing rule. This also means that the derivation in G undergoes two phases. In phase I, only context-free rules are applied. This phase ends with applying the context-free deletion rule $S' \to \lambda$ (which is the only rule that has S' on its left-hand side). Then only, non-context-free deletion rules $AB \to \lambda$ and $CD \to \lambda$ are applied in phase II. Notice that the symbol from N', as long as present, separates A and C from B and D; this prevents a premature start of phase II. One of the features of SGNF derivations is that any string that can be derived can contain at most one substring AB or CD in its so-called *center*. If such a substring is present, the derivation is in phase II; also, then no nonterminal from N' occurs. In phase I, exactly one such nonterminal is present (in the center). Therefore, we can differentiate two cases within (ii) for $X, Y \in N' \setminus \{S'\}$ with $X \neq Y$: either, we have a rule $X \to bY$, with $b \in \{A, C\}$, or we have a rule $X \to Yb$, with $b \in T \cup \{B, D\}$. This case distinction is often necessary when simulating type-0 grammars in SGNF, as we will see later.

We write $\Rightarrow_r$ to denote a single derivation step using rule r, and $\Rightarrow_G$ (or $\Rightarrow$ if no confusion arises) denotes a single derivation step using any rule of G. Then, $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$, where $\Rightarrow^*$ is the reflexive transitive closure of $\Rightarrow$.

2.1 Graph-Controlled Insertion-Deletion Systems

Definition 2 *A* graph-controlled insertion-deletion system *(GCID for short) with k components is a construct* $\Pi = (k, V, T, A, H, i_0, F, R)$, *where (i) k is the number of components, (ii) V is an alphabet, (iii) $T \subseteq V$ is the terminal alphabet, (iv) $A \subset V^*$ is a finite set of strings, called axioms, present in the initial component, (v) H is a set of labels associated (in a one-to-one manner) to the rules in R, (vi) $i_0 \in [1 \dots k]$ is the initial component, (vii) $F \subseteq [1 \dots k]$ is the set of final components and (viii) R is a finite set of rules of the form $l : (i, r, j)$, where $l \in H$ is the label of the rule, r is an insertion rule of the form $(u, \eta, v)_I$, with insertion string η and context (u, v), or deletion rule of the form $(u, \delta, v)_D$, with deletion string δ and context (u, v), where $u, v \in V^*$, $\eta, \delta \in V^+$ and $i, j \in [1 \dots k]$.*

Often, the component is part of the label name. This will also (implicitly) define H. We shall omit the label l of the rule wherever it is not necessary for the discussion.

We now describe how GCID systems work. Applying an insertion rule of the form $(u, \eta, v)_I$ means that the string η is inserted between u and v; this corresponds to the rewriting rule $uv \to u\eta v$. Similarly, applying a deletion rule of the form $(u, \delta, v)_D$ means that the string δ is deleted between u and v; this corresponds to the rewriting rule $u\delta v \to uv$. A *configuration* of Π is represented by $(w)_i$, where $i \in [1 \dots k]$ is the number of the current component and $w \in V^*$ is the current string. We also say that w has entered or moved to component Ci. We write $(w)_i \Rightarrow_l (w')_j$ if there is a rule $l : (i, r, j)$ in R, and w' is obtained by applying the insertion or deletion rule r to w. For brevity, we write $(w)_i \Rightarrow (w')_j$ if there is some rule l in R such that $(w)_i \Rightarrow_l (w')_j$. To avoid confusion with traditional grammars, we write $\Rightarrow_*$ for the transitive reflexive closure of $\Rightarrow$ between configurations. The language $L(\Pi)$ generated by Π is defined as $\{w \in T^* \mid (x)_{i_0} \Rightarrow_* (w)_{i_f} \text{ for some } x \in A \text{ and some } i_f \in F\}$.

The *underlying control graph* of a GCID system Π with k components is defined to be a graph with k nodes labelled $C1$ through Ck and there exists a directed edge from Ci to Cj if there exists a rule of the form (i, r, j) in R of Π. We also associate an undirected graph on k nodes to a GCID system of k components as follows: there is an undirected edge from a node Ci to Cj if there exists a rule of the form (i, r_1, j) or (j, r_2, i) in R of Π. We call a GCID system with k components *star-controlled* if its underlying undirected control graph has the edge set $\{\{C1, Ci\} \mid i \in [2 \dots k]\}$. This means that the corresponding directed control graph may contain arcs like $(C1, Ci)$, $(Ci, C1)$, but no loops.

Below, we provide a few examples for a better understanding of how the above-defined system works. As star-controlled systems have to have at least two components to produce anything non-trivial, it is interesting to observe that even with only two components, non-regular languages can be obtained.

Example 3 *The language $\{a^n b^n \mid n \geq 0\}$ can be generated by a star-controlled insertion-deletion system with two components, alphabet $\{a,b,A,B\}$, the axiom set $\{AB\}$ in C1, final component $\{C1\}$ and the following rules:* $r1.1 : (1,(A,a,\lambda)_I,2)$, $r2.1 : (2,(B,b,\lambda)_I,1)$, $r1.2 : (1,(\lambda,A,\lambda)_D,2)$ *and* $r2.2 : (2,(\lambda,B,\lambda)_D,1)$. *A possible derivation of a terminal string is:*

$$(AB)_1 \Rightarrow_{r1.1} (AaB)_2 \Rightarrow_{r2.1} (AaBb)_1 \Rightarrow_{r1.2} (aBb)_2 \Rightarrow_{r2.2} (ab)_1.$$

Observe that $(aBb)_2 \Rightarrow_{r2.1} (aBbb)_1$ *is possible, but now the derivation is stuck, as any rule in C1 checks for the presence of the nonterminal A. Yet, as the nonterminal B is present, the configuration* $(aBbb)_1$ *cannot lead to a terminal word. The size of this system is* $(2;1,1,0;1,0,0)$. *A very similar system can be given for this language that is of size* $(2;1,0,1;1,0,0)$. *For the very similar language* $\{a^n b^n \mid n \geq 1\}$, *even a system with two rules* $r1 : (1,(a,a,\lambda),2)$ *and* $r2 : (2,(b,b,\lambda),1)$ *would suffice, with axiom ab.*

Recall that the class REG of regular languages is the lowest class of the Chomsky hierarchy. It can be characterized by right-linear grammars whose rules have the form $A \to Ba$ or $A \to \lambda$ for nonterminal symbols A,B and a terminal symbol a. We use this characterization to prove that star-controlled GCID systems can generate all regular languages. The previous example then shows that even non-regular languages can be generated.

Theorem 4 *Each regular language (and also some non-regular languages) can be generated by a GCID$_S$ system of size* $(2;3,0,1;2,0,0)$, *where the initial component C1 is also the only final one.*

Later, we will see that with both components being final, many more languages can be described.
<u>Proof</u> We only sketch the construction in the following. For each rule of the form $A \to aB$ of a right-linear grammar G that we start with, we introduce the insertion rule $(\lambda,aB\$,A)$ into the first component of the simulating GCID$_S$ system Π. For each erasing rule $A \to \lambda$, we add the insertion rule $(\lambda,a\$,A)$ into the first component of Π. In both cases, $\$$ is a special marker symbol that is taken care of in the second component that contains all possible deletion rules of the form $(\lambda,\$A,\lambda)$ for any nonterminal A of G. For instance, if G contains the rules $S \to aX$ and $X \to \lambda$, enabling the derivation $S \Rightarrow aX \Rightarrow a$, then the simulation is performed as follows: $(S)_1 \Rightarrow (aX\$S)_2 \Rightarrow (aX)_1 \Rightarrow (a\$X)_2 \Rightarrow (a)_1$. $\qquad\square$

By adding more nonterminal symbols, one can also achieve this result with GCID$_S$ systems of size $(2;2,0,1;2,0,0)$. We leave it open if GCID$_S$ systems with only two components and only one final component can generate each recursively enumerable language.

Example 5 *The copy language $\{ww \mid w \in \{a,b\}^*\}$ can be generated by a star-controlled insertion-deletion system $\Pi = (3;\{a,b,A,B\},\{a,b\},\{AB\},H,1,\{1\},R)$, where $H = \{r1.1,r1.2,r1.3,r2.1,r2.2, r3.1\}$ and R is the set of rules depicted in Table 2; Π has size $(3;1,0,1;1,0,0)$. Starting with the axiom AB in C1, if we apply $r1.3$, then we can apply $r2.2$ only in C2 and that produces λ in C1. The nonterminals A and B serve as markers and if an a is introduced to the left of A in C1 (by $r1.1$), then one a is introduced to the left of B (by $r2.1$) in C2. Similarly, if one b is introduced to the left of A in C1 (by $r1.2$), then a b is introduced to the left of B (by $r3.1$) in C3. This guarantees to have the pattern of the copy language produced by the system Π. But, there is a caveat here. If one can apply $r1.1$ in C1, then in C2, $r2.2$ can also be applied and in such a case, the pattern of the copy language is not followed. However, then back in C1, only $r1.3$ can be applied, which means for the string to move to C2 and there, the derivation stops.*

Component C1	Component C2	Component C3
$r1.1 : (1,(\lambda,a,A)_I,2)$	$r2.1 : (2,(\lambda,a,B)_I,1)$	$r3.1 : (3,(\lambda,b,B)_I,1)$
$r1.2 : (1,(\lambda,b,A)_I,3)$	$r2.2 : (\lambda,A,\lambda)_D,1)$	
$r1.3 : (1,(\lambda,B,\lambda)_D,2)$		

Table 2: Star-controlled ins-del system for generating $\text{Copy}_{a,b} = \{ww \mid w \in \{a,b\}^*\}$.

(a) Star-shaped control graphs underlying the star-controlled systems depicted in Example 5.

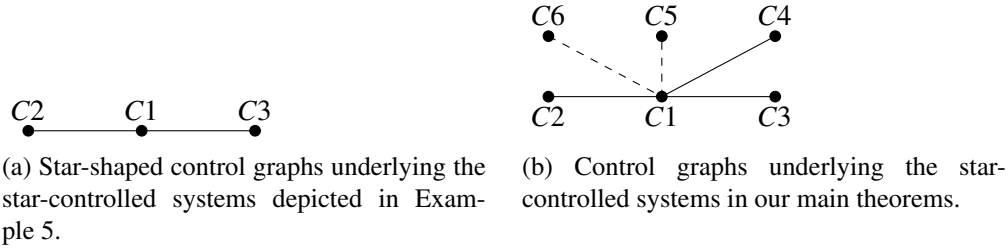(b) Control graphs underlying the star-controlled systems in our main theorems.

Figure 1: Control graphs underlying different GCID systems in this paper.

As C2 is not a final component, by definition the strings over the terminal alphabet $\{a,b\}$ that are not also leading into the final component are not collected into the language generated by Π.

A sample derivation for aabaab is given below.

$$(AB)_1 \Rightarrow_{r1.1} (aAB)_2 \Rightarrow_{r2.1} (aAaB)_1 \Rightarrow_{r1.1} (aaAaB)_2 \Rightarrow_{r2.1} (aaAaaB)_1 \Rightarrow_{r1.2} (aabAaaB)_3$$
$$\Rightarrow_{r3.1} (aabAaabB)_1 \Rightarrow_{r1.3} (aabAaab)_2 \Rightarrow_{r2.2} (aabaab)_1.$$

The control graph underlying the construction is shown in Figure 1.

3   Computational Completeness

In this section, we present the main results of our paper. First, we discuss some limitations for getting computational completeness results and then, we describe two important cases of resource restrictions that characterize RE.

3.1   GCID$_S$ systems with insertion and deletion length one

In [19], it has been proved that ins-del systems with size $(1,1,1;1,1,1)$ characterize RE. Notice that it is proved in [12, 15] that ins-del systems of size $(1,1,1;1,1,0)$ or $(1,1,0;1,1,1)$ cannot characterize RE. It is therefore obvious that we need at least 2 components in a graph-controlled ins-del system of sizes $(1,1,1;1,1,0)$ and $(1,1,0;1,1,1)$ to characterize RE. In [3], we characterized RE by path-controlled GCID systems of size $(k;1,i',i'';1,j',j'')$ for different combinations of $k \geq 1, i', i'', j', j''$.

However, if we impose star structure as the underlying control graph and the resultant string has to move in/move out during every derivation step, then it is interesting to notice that the context-free rules of SGNF, namely $p : X \to bY$, $q : X \to Yb$ and $h : S' \to \lambda$ can never be *directly simulated* by rules of GCID$_S(k;1,i',i'';1,j',j'')$ for any value of $k \geq 2, i', i'', j', j'' \geq 0$. Here, by a *direct simulation* of a rule r, we mean that, assuming a sentential form w may yield the sentential form v by applying rule r within the original grammar (which is, in our case, in SGNF), then the simulating star-controlled GCID system will

start in the central component $C1$ with the sentential form w and derive after a number of steps, possibly, during the simulation, introducing and deleting symbols specific to r (called *markers* in the following), the sentential form v and moving back to $C1$ to be ready for the next simulation step.

Proposition 6 *The context-free rules of a grammar in SGNF, namely $p : X \to bY$ and $q : X \to Yb$ (with $X \neq Y$), as well as $h : S' \to \lambda$, can never be directly simulated by rules of $\text{GCID}_S(k; 1, i', i''; 1, j', j'')$ for any value of $k \geq 2, i', i'', j', j'' \geq 0$.*

<u>Proof</u> To directly simulate $p : X \to bY$ using insertion-deletion rules, we need two insertion rules (one to insert b and one to insert Y; here we recall that the insertion length is 1) and one deletion rule to delete X. Hence, we need three *basic* insertion-deletion rules. Further, if we need to introduce $r \geq 1$ markers, then we can insert only one marker at a time using an insertion rule which will account for r insertion rules. At the end of the derivation, we need to delete all the r markers using r deletion rules (since we can only delete one symbol at a time). This amounts to having r insertion rules and r deletion rules to deal with the markers and 3 basic insertion-deletion rules to simulate $p : X \to bY$. This sums up to $2r + 3$ rules.

We need to distribute these $2r + 3$ rules among the k components of the GCID system. Let $C1$ be the central (initial and final) component. As the system is star-structured, in a rule $(i, (x, y, z)_\delta, j), \delta \in \{I, D\}, i, j \in [1 \ldots k]$, we have $i \neq j$, as loops are not allowed, and $|\{i, j\} \cap \{1\}| = 1$. Hence, the order of rule applications in any derivation will start at the central node and then alternate between central and non-central nodes. Therefore, (i) the last rule of the simulation should be placed in a non-central component and not in $C1$ and (ii) the total number of rules for simulation is even. Since $2r + 3$ is not even, the statement follows. $\square$

By its definition, a derivation of a GCID_S system has to alternate between the central component and any other component. By putting exactly the same rules in two components and declaring one of the two components as being central, while both are final, one obtains:

Proposition 7 $\text{GCID}(1; n, i', i''; m, j', j'') \subseteq \text{GCID}_S(2; n, i', i''; m, j', j'')$ *holds for any value of $i', i'', j', j'' \geq 0$ and $n, m \geq 1$.*

For example, this entails $\text{GCID}_S(2; 1, 1, 1; 1, 1, 1) = \text{RE}$ and similar computational completeness results based on what is known for ins-del systems. By way of contrast, computational completeness results for 2-component graph-controlled systems do not necessarily carry over to our star-controlled systems, as there, 'loops' might be allowed.

3.2 GCID$_s$ systems with insertion or deletion length of more than one

To simplify the presentation and proofs of our further results, the following observations from [3] are used, adapted to our case.

Proposition 8 *[3] Let k, n, i', i'', m, j, j'' be non-negative integers.*

1. $\text{GCID}_S(k; n, i', i''; m, j', j'') = [\text{GCID}_S(k; n, i'', i'; m, j'', j')]^R$;

2. $\text{RE} = \text{GCID}_S(k; n, i', i''; m, j', j'')$ *iff* $\text{RE} = \text{GCID}_S(k; n, i'', i'; m, j'', j')$.

Theorem 9 $\text{RE} = \text{GCID}_S(6; 1, 1, 0; 2, 0, 0)$ *and* $\text{RE} = \text{GCID}_S(6; 1, 0, 1; 2, 0, 0)$.

<u>Proof</u> Consider a type-0 grammar $G = (N, T, P, S)$ in SGNF as in Definition 1. We construct a GCID system $\Pi = (6, V, T, \{S\}, H, 1, \{1\}, R)$ of size $(6; 1, 1, 0; 2, 0, 0)$ such that $L(\Pi) = L(G)$. The alphabet V contains rule markers, apart from the symbols of G. More specifically, for each rule $\gamma \to \delta \in P$ labeled r, we have $r \in V$. Moreover, if $\gamma \to \delta \neq S' \to \lambda$, we also have single-primed markers $r' \in V$. Finally, for

C1	C2	C3
$p1.1:(1,(X,p,\lambda)_I,2)$	$p2.1:(2,(\lambda,X,\lambda)_D,1)$	$p3.1:(3,(p',p^v,\lambda)_I,1)$
$p1.2:(1,(p,p',\lambda)_I,3)$	$p2.2:(2,(\lambda,p''p''',\lambda)_D,1)$	
$p1.3:(1,(p',b,\lambda)_I,5)$		
$p1.4:(1,(\lambda,pp',\lambda)_D,4)$		
$p1.5:(1,(p''',p^{iv},\lambda)_I,4)$		
$p1.6:(1,(p''',Y,\lambda)_I,2)$		
$q1.1:(1,(X,q,\lambda)_I,2)$	$q2.1:(2,(\lambda,X,\lambda)_D,1)$	$q3.1:(3,(q',b,\lambda)_I,1)$
$q1.2:(1,(q,q',\lambda)_I,3)$		
$q1.3:(1,(q',Y,\lambda)_I,4)$		
$h1.1:(1,(S',h,\lambda)_I,2)$	$h2.1:(2,(\lambda,S'h,\lambda)_D,1)$	
$f1.1:(1,(A,f,\lambda)_I,6)$	$f2.1:(2,(\lambda,Af',\lambda)_D,1)$	
$f1.2:(1,(\lambda,fB,\lambda)_D,2)$		
$g1.1:(1,(C,g,\lambda)_I,6)$	$g2.1:(2,(\lambda,Cg',\lambda)_D,1)$	
$g1.2:(1,(\lambda,gD,\lambda)_D,2)$		

C4	C5	C6
$p4.1:(4,(p'',p''',\lambda)_I,1)$	$p5.1:(5,(b,p'',\lambda)_I,1)$	
$p4.2:(4,(\lambda,p^{iv}p^v,\lambda)_D,1)$		
$q4.1:(4,(\lambda,qq',\lambda)_D,1)$		
		$f6.1:(6,(B,f',\lambda)_I,1)$
		$g6.1:(6,(D,g',\lambda)_I,1)$

Table 3: Star-controlled $\text{GCID}_S(6;1,1,0;2,0,0)$ simulating the rules of SGNF.

context-free rules of the form $X \to bY$, even markers $r'',r''',r^{iv},r^v \in V$. We refer to Table 3, showing the simulation of the different rule types of SGNF. The columns of the table correspond to the components of Π. The rows of Table 3 correspond to the rules simulating the 'linear rules' $p:X \to bY$ and $q:X \to Yb$, with $X \in N'$ and $b \in N'' \cup T$, $h:S' \to \lambda$, as well as $f:AB \to \lambda$ and $g:CD \to \lambda$.

We now prove that $L(G) \subseteq L(\Pi)$ as follows. We show that if $w \Rightarrow w'$ in G, then $(w)_1 \Rightarrow_* (w')_1$ according to Π. The claim then follows by induction.

Context-free rule $q:X \to Yb$. Here, $w = \alpha X\beta$ and $w' = \alpha Yb\beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\} \cup T)^*$. The simulation performs as follows:

$$(\alpha X\beta)_1 \Rightarrow_{q1.1} (\alpha Xq\beta)_2 \Rightarrow_{q2.1} (\alpha q\beta)_1 \Rightarrow_{q1.2} (\alpha qq'\beta)_3 \Rightarrow_{q3.1} (\alpha qq'b\beta)_1$$
$$\Rightarrow_{q1.3} (\alpha qq'Yb\beta)_4 \Rightarrow_{q4.1} (\alpha Yb\beta)_1.$$

Context-free rule $p:X \to bY$. Here, $w = \alpha X\beta$ and $w' = \alpha bY\beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\} \cup T)^*$. The simulation performs as follows:

$$(\alpha X\beta)_1 \Rightarrow_{p1.1} (\alpha Xp\beta)_2 \Rightarrow_{p2.1} (\alpha p\beta)_1 \Rightarrow_{p1.2} (\alpha pp'\beta)_3 \Rightarrow_{p3.1} (\alpha pp'p^v\beta)_1$$
$$\Rightarrow_{p1.3} (\alpha pp'bp^v\beta)_5 \Rightarrow_{p5.1} (\alpha pp'bp''p^v\beta)_1 \Rightarrow_{p1.4} (\alpha bp''p^v\beta)_4 \Rightarrow_{p4.1} (\alpha bp''p'''p^v\beta)_1$$
$$\Rightarrow_{p1.5} (\alpha bp''p'''p^{iv}p^v\beta)_4 \Rightarrow_{p4.2} (\alpha bp''p'''\beta)_1 \Rightarrow_{p1.6} (\alpha bp''p'''Y\beta)_2 \Rightarrow_{p2.2} (\alpha bY\beta)_1.$$

One might wonder that the $h : S' \to \lambda$ and the $f : AB \to \lambda$ rules could easily be simulated by the rules $(1,(\lambda,S',\lambda)_D,1)$ and $(1,(\lambda,AB,\lambda)_D,1)$, respectively. However, the underlying control graph of our star-controlled GCID forbids loops and hence, we have given a different simulation for these rules. Since the correctness of the h-rule simulation is trivial, it remains to discuss the simulation of the rule $f : AB \to \lambda$. The simulation of $g : CD \to \lambda$ is similar and hence omitted.

Non-context-free rules $f : AB \to \lambda$. This means that we expect $w = \alpha AB\beta$ and $w' = \alpha\beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\}\cup T)^*$. Within Π, this can be simulated as follows.

$$(\alpha AB\beta)_1 \Rightarrow_{f1.1} (\alpha Af B\beta)_6 \Rightarrow_{f6.1} (\alpha Af Bf'\beta)_1 \Rightarrow_{f1.2} (\alpha Af'\beta)_2 \Rightarrow_{f2.1} (\alpha\beta)_1.$$

Conversely, a derivation $(w)_1 \Rightarrow^* (w')_1$, with $w \neq w'$ and $w,w' \in \{A,C\}^*(N'\cup\{\lambda\})(\{B,D\}\cup T)^*$ has to start like $(w)_1 \Rightarrow (v)_j$ in Π. If some rule from $C1$ is applied to w, the rule will insert a rule marker into the string w and branch to $C2$ (when simulating context-free rules) or to $C3$ (when simulating non-context-free rules). The introduction of rule markers in $C1$ will take care of the non-interference among the non-context-free and context-free rules. We now discuss the possibilities in detail. In our discussion, we distinguish between sentential forms containing or not containing a symbol from N'.

Our inductive arguments will also show that, in Π, no sentential form is derivable that contains two occurrences of symbols from N'. More generally, we can show the following. Assume that we can derive some configuration $(w)_1$ in Π such that the string v contains no marker symbols, where $v = \mu(w)$ is obtained by applying the morphism μ that acts like the identity on V apart from the letters f, f', g, g' that will be erased. Then the sentential form v is also derivable in G. In particular, if $w \in T^*$, then $v = w$, i.e., each word in $L(\Pi)$ also belongs to $L(G)$.

We will also prove by induction that, if $(w)_1$ is derivable in Π and if w contains at most one occurrence of N' and no markers but possibly f, f', g, g', then also $\mu(w)$ is derivable in Π and then, only using the rules $f1.1$, $f6.1$, $g1.1$ and $g6.1$, we can derive $\mu(w)$ in Π from w. This also means that, in each such string w derivable in Π, the number of occurrences of symbols from $\{f,g\}$ equals the number of occurrences of symbols from $\{f',g'\}$. We also call this the *balance condition*. Therefore, we can start our inductive hypothesis with strings that can be derived both in G and in Π and observe the maintenance of the balance condition along our arguments.

Let us first assume (by induction) that the sentential form $w^1 = \alpha X\beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\}\cup T)^*$ and $X \in N'$ is derivable in G and the configuration $(w^1)_1$ is derivable in Π. We will prove (as induction step) that if $(w^1)_1 \Rightarrow_* (u)_1$, $w^1 \neq u$, and $u \in \{A,C\}^*(N'\cup\{\lambda\})(\{B,D\}\cup T)^*$ is the first sentential form from $\{A,C\}^*(N'\cup\{\lambda\})(\{B,D\}\cup T)^*$ that appears in a derivation of Π in $C1$, then $w^1 \Rightarrow u$ holds in G, except from a premature start of simulating non-context-free rules also discuss below and where we argue that the balance condition is maintained.

Applying $f1.1$ to $w^1 = \alpha X\beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\}\cup T)^*$ and $X \in N'$, the only nonterminal from N', is possible but unintended. (A similar discussion applies to $g1.1$.) A successful application yields the configuration $(w^2)_6$, with $w^2 = \alpha_1 Af \alpha_2 X\beta$, where $\alpha_1 A\alpha_2 = \alpha$. Now, the only applicable rules are $f6.1$ or $g6.1$. We get a configuration $(w^3)_1$ with $\mu(w^3) = w^1$. It can be observed that on w^3, neither $f1.2$ nor $g1.2$ are applicable, as these rules require AB or CD sitting in the center of w^1, which was not the case by assumption. Therefore, we could either continue our journey with inserting further markers f, f', g, g' (but always maintaining the balance condition) or finally apply $r1.1$, belonging to a context-free rule r. Now, observe that, instead of applying $r1.1$ and then $r2.1$ (because $f2.1$ and $g2.1$ are inapplicable), yielding a configuration $(u)_1$, we could also first apply $r1.1$ and $r2.1$ to w^1, and then the same f- and g-simulation rules as before, arriving at $(u)_1$ in a different way. This proves (here as part of the induction step) that, as claimed, we can exchange the sequence of rule applications in a way that we

apply f- and g-simulation rules after the other rules that are meant to simulate the context-free rules. We also see by induction that the balance condition is always maintained.

Applying $q1.1$ to $w^1 = \alpha X \beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\} \cup T)^*$ and $X \in N'$, the only nonterminal from N', yields the configuration $(w^2)_2$ with $w^2 = \alpha X q \beta$. In $C2$, all rules (except $q2.1$ and $p2.1$) are guarded by markers and the only applicable rule are $q2.1$ or $p2.1$ (in case there is a rule $X \to b'Y'$ in P) which delete X, yielding $(w^3)_1 = (\alpha q \beta)_1$. Due to the rule markers, apart from the intended rule $q1.2$, one could also apply $f1.1$ or $g1.1$. In that case, having moved to $C6$, $f6.1$ or $g6.1$ are applicable, yielding some configuration $(w')_1$ with $\mu(w') = w^3$. As discussed before, we can even continue like this, but in order to make any further progress, we will have to apply $q1.2$ in some configuration $(w'')_1$ with $\mu(w'') = w^3$. As we could also apply the non-context-free simulation rules afterwards, it suffices to discuss what happens if we apply $q1.2$ to w^3 as intended. Hence, we arrive at the configuration $(w^4)_3$ with $w^4 = \alpha q q' \beta$. The required rule markers cause $q3.1$ to be the only applicable rule as desired. Therefore, we arrive at the configuration $(w^5)_1 = (\alpha q q' b \beta)_1$. Clearly, one could now (again) apply $f1.1$ or $g1.1$, but this would only lead to prematurely introducing the markers f, f', g, g' similar as discussed before, again always maintaining the balance condition. Therefore, the only applicable rule that needs to be discussed (apart from the intended one, which is $q1.3$) is $q1.2$ (again). With the string $\alpha q q' q' b \beta$, we are back to $C3$. Now, there are two possible subsequent configurations: (a) $(\alpha q q' b q' b \beta)_1$, or (b) $(\alpha q q' q' b b \beta)_1$. In Case (a), we claim that there is no way to delete the second occurrence of q' in the future. Namely, the only way to delete q' is if left to it, q is sitting. But as now some $b \in \{B,D\} \cup T$ is to the left of q', there is no way to introduce q in this position later, because the marker q always works as a symbol that replaces the former N'-symbol. Therefore, a derivation following (a) cannot terminate. The situation is different in Case (b). For instance, we can apply $q1.3$ to string $\alpha q q' q' b b \beta$, followed by $q4.1$. Again, we have two configurations to study: (A) $(\alpha q' Y b b \beta)_1$ or (B) $(\alpha Y q' b b \beta)_1$. In Case (A), we can argue similarly to Case (a) above to see that this configuration cannot lead to a terminal string: left to q' will sit some symbol A or C. Case (B) is indeed different. Assuming that only rules of the form $Z \to b'Z'$ are simulated subsequently, there may be a derivation $Y \Rightarrow^* \gamma X$ with $\gamma \in \{A,C\}^+$ that is simulated by the GCID system as intended. Hence, we see now a configuration $(\alpha \gamma X q' b b \beta)_1$ and then, after a short excursion into $C2$, we see $(\alpha \gamma q q' b b \beta)_1$. Now, we can actually terminate by using the rules $q1.3$ and $q4.1$, leading to $(\alpha \gamma Y b b \beta)_1$. However, we would arrive at the same string if we had followed our intended plan. Then, we could get from $(\alpha Y b \beta)_1$ via $(\alpha \gamma X b \beta)_1$ to $(\alpha \gamma q b \beta)_1$. Now, after applying $q1.2$ and $q2.2$ as intended, we can also see $(\alpha \gamma q q' b b \beta)_1$ and continue as above. This argument is also valid (by a separate yet straightforward induction) if we happen to produce a string $(\alpha q (q')^k b^k \beta)_1$ for an arbitrary $k > 1$. Therefore, we can avoid this process that we call *rule inversion*, and always follow our standard derivation instead. We can hence assume that we apply $q1.3$ to w^5 as desired. Therefore, we arrive at the configuration $(w^5)_1 = (\alpha q q' Y b \beta)_4$. If we actually apply $q4.1$, then we arrive at $(\alpha Y b \beta)_1$ as intended, proving the inductive step in this case. But by the very structure of this component, no other rules are applicable.

Applying $p1.1$ to $w^1 = \alpha X \beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\} \cup T)^*$ and $X \in N'$, the only nonterminal from N', will insert a marker p to the right of X, yielding $(w^2)_2 = (\alpha X p \beta)_2$. Recall that we are trying to simulate the rule $X \to bY$ for some $X, Y \in N'$ and $b \in \{A,C\}$. In $C2$, all rules (except $p2.1$ and $q2.1$) are guarded by markers and the only applicable rule are $q2.1$ or $p2.1$ (in case there is a rule $X \to b'Y'$ in P) which delete the nonterminal X, yielding a string $w^3 = \alpha p \beta$, i.e., we arrive at the configuration $(w^3)_1$. Since X is deleted in the previous step and there is no p', p'', the only applicable rule is $p1.2$ which inserts a p' after p, yielding the configuration $(w^4)_3 = (\alpha p p' \beta)_3$. In $C3$, guarded by rule markers, we have to apply $p3.1$ as intended. Hence, we arrive at $(w^5)_1 = (\alpha p p' p^v \beta)_1$. If we re-apply $p1.2$, we achieve

an imbalance of the number of occurrences of p and p'. This is problematic insofar, as pp' is deleted together. Also, we would have to then re-apply $p3.1$ again, creating another imbalance. Alternatively, we could apply $f1.1$ or $g1.1$, which introduces a pair of non-primed and primed f/g-markers prematurely, but maintaining their balance. This brings us to the conclusion that we should apply $p1.3$ in $(w^5)_1$.

Hence, we arrive at $(w^6)_5 = (\alpha pp'bp^v\beta)_5$, with $b \in \{A,C\}$. In $C5$, we have to apply a rule that puts some marker r'' to the right of an occurrence of b. As $b \in \{A,C\}$, the b occurring between p' and p^v is the rightmost of all occurrences of A,C within w^6. Let us first discuss what happens if we do apply some $r5.1$ (but possibly $r \neq p$) to this described rightmost occurrence and mark the situation when some b within α is affected as $(*)$, not to forget its discussion. We get to $(w^7)_1 = (\alpha pp'br''p^v\beta)_1$. Apart from now applying $f1.1$ or $g1.1$, we could also apply $p1.2$ or $p1.3$, and finally also $p1.4$ (as intended). The scenario of prematurely introducing the f/g-markers has been sufficiently discussed before. If we apply rule $p1.2$, we again create an imbalance concerning p/p'. Let us defer the discussion of applying $p1.3$ at this configuration $(w^7)_1$ a bit as $(+)$; we rather discuss applying $p1.4$. We arrive at the configuration $(w^8)_4 = (\alpha br''p^v\beta)_4$. Now, only $r4.1$ would be applicable, leading to $(w^9)_1 = (\alpha br''r'''p^v\beta)_1$. We are now in safer waters, as we have to use rule $r1.5$ to get to $(w^{10})_4 = (\alpha br''r'''r^{iv}p^v\beta)_4$, because if we apply $p1.6$ directly, we have no chance to delete p^v in the future. If we apply $r4.1$ again on w^{10}, we create an imbalance between the number of r'' and r''', but this balance is necessary for deleting these markers in $C2$. By using $r4.2$ alternatively on w^{10}, one can see that the only chance to continue the route is when we have $r = p$. In that case, we move to $C1$ with $w^{11} = \alpha bp''p'''\beta$. If we now re-apply $p1.5$, at $C4$, we have to apply $p4.1$ and create an imbalance between p'' and p''', hence preventing us from a terminating derivation. If we introduce f- or g-markers, we are forced to introduce primed versions in $C6$; we have discussed these premature but balanced introductions of these markers before. Hence, we have to discuss applying $p1.6$ as intended. We enter $C2$ with $w^{12} = \alpha bp''p'''Y\beta$. Now, we can either delete Y with some fitting rule $s2.1$ and return to the configuration $(w^{11})_1$, hence making no progress, or we apply $p2.2$ as intended, finally getting to the configuration $(w^{12})_1 = (\alpha bY\beta)_1$ as desired.

In order to conclude that the induction step has been shown, we still have to consider two scenarios, marked as $(*)$ and $(+)$ above. In $(+)$, we look at $(w^7)_1 = (\alpha pp'br''p^v\beta)_1 \Rightarrow_{p1.3} (\alpha pp'bbr''p^v\beta)_5$. Assume we apply a rule $s5.1$ next. As the case when we find $b \in \{A,C\}$ within α is similar to the discussion $(*)$ that is still to come, we focus on two cases of configurations: (1) $(\alpha pp'bs''br''p^v\beta)_1$ or (2) $(\alpha pp'bbs''r''p^v\beta)_1$. In both configurations, we can again apply $p1.3$, but this makes the whole case fail even more. We can now derive (under the conclusion that $r = p$) in the same way as in the main line of derivation, leading to $(\alpha bs''bY\beta)_1$ (Case (1)) or to $(\alpha bbs''Y\beta)_1$ (Case (2)). In both cases, there is no way to make use of s'', because this means we have to move to $C4$, or we mis-use another p-type rule at some point, when $p1.4$ makes us enter $C4$ again, but then continuing with the s-markers (using $s4.1$). Let us clarify this by assuming that we simulate $t : Y \to b'Y'$ next. Following the standard simulation up to $t1.4$, we get $(\alpha bs''bb't''t^v\beta)_4$ (Case (1)) or $(\alpha bbs''b't''t^v\beta)_4$ (Case (2)). We could now use $s4.1$, $s1.6$ and $s2.2$ to introduce another nonterminal from N' at the position of s'', but behold: we have now another left-over double-primed marker t'' whose removal can only be achieved by switching between two rule simulations in the 'next round'. Therefore, we will never be able to terminate this derivation.

For scenario $(*)$, we reconsider $(w^6)_5 = (\alpha pp'bp^v\beta)_5$, with $\alpha = \alpha_1 b\alpha_2$, so that for a suitable rule r that should introduce $Z \in N'$, $(w^7)_1 = (\alpha_1 br''\alpha_2 pp'bp^v\beta)_1$. We could try to continue with $(w^7)_1 \Rightarrow_{p1.4}$

$$(\alpha_1 br''\alpha_2 bp^v\beta)_4 \Rightarrow_{r4.1} (\alpha_1 br''r'''\alpha_2 bp^v\beta)_1 \Rightarrow_{r1.6} (\alpha_1 br''r'''Z\alpha_2 bp^v\beta)_2 \Rightarrow_{r2.2} (\alpha_1 bZ\alpha_2 bp^v\beta)_1$$

but then there is never a chance to lose p^v again. Therefore, also this scenario will never see a derivation producing a terminal string.

Component C1	Component C2	Component C3	Component C4	Component C5
$p1.1 : (1, (X, p, \lambda)_I, 2)$	$p2.1 : (2, (\lambda, X, \lambda)_D, 1)$	$p3.1 : (3, (\lambda, p'', \lambda)_D, 1)$	$p4.1 : (4, (p, p', \lambda)_I, 1)$	$p5.1 : (5, (\lambda, pp', \lambda)_D, 1)$
$p1.2 : (1, (p, p'', \lambda)_I, 4)$				
$p1.3 : (1, (p', b, \lambda)_I, 5)$				
$p1.4 : (1, (p'', Y, \lambda)_I, 3)$				

Table 4: A direct simulation attempt for a *p*-rule $p : X \to bY$.

We are now discussing a string w derivable in G and as configuration $(w)_1$ in Π, with $w = \alpha AB\beta$, with $\alpha \in \{A, C\}^*$ and $\beta \in (\{B, D\} \cup T)^*$. The case of a string of the form $\alpha CD\beta$ can be discussed in a very similar fashion. First observe that we cannot apply any rule $p1.x$ or $q1.x$ or $h1.x$ due to the absence of nonterminals from N' or of required markers. We could in fact start with $g1.1$, followed by $g6.1$, and even repeat this, so that some g-markers are attached to C-occurrences. Similarly, we can consider such derivations to occur prematurely, because finally we have to use the f-rule as explained next.

Applying $f1.1$ to $w = \alpha AB\beta$, we get a string w_1 by inserting f anywhere after an A-occurrence within w. Let $\alpha A = \alpha_1 A \alpha_2$ indicate this position, i.e., $w_1 = \alpha_1 A f \alpha_2 B\beta$. w_1 is transferred to component $C6$. So, $f6.1$ is applied and the string, yielding $w_2 = \alpha_1 A f \alpha_2 \beta_1 B f' \beta_2$, which enters $C1$, where $B\beta = \beta_1 B \beta_2$. Notice that the configuration $(w_2)_1$ could have also been created by a premature application of $f1.1$ and $f6.1$ in some earlier phase of the derivation. This explains how a string that satisfies the balance condition could finally yield a terminal string, although it is not following the standard simulation as described in the beginning of the proof. As α_2 cannot contain any B-occurrence, now applying $f1.2$ necessitates $\alpha_2 = \beta_1 = \lambda$, and then, $w_3 = \alpha_1 A f' \beta_2$ is sent to $C2$. There, the only applicable rule is $f2.1$ as intended, producing $w_4 = \alpha_1 \beta_2$, sent to $C1$ as intended. As mentioned at several places, instead of applying $f1.2$ on w_2, one could also possibly apply $f1.1$ again, or also $g1.1$. We can consider all these attempts as premature ones, they only affect the left part of the string and have to be finally successfully matched by using rules $f1.2$ or $g1.2$, followed by executing another deletion rule in $C2$.

It could be that a string $w = \alpha\beta$ was derived in G (and hence possibly the configuration $(w)_1$ in Π by induction) with $\alpha \in \{A, C\}^*$ and $\beta \in (\{B, D\} \cup T)^*$ and neither α ends with A and β starts with B nor α ends with C and β starts with D. We can still apply rules $f1.1$ or $g1.1$, moving the resultant string to $C6$, where $f6.1$ or $g6.1$ are applicable, moving us back to $C1$. Yet, the crucial observation is that neither $f1.2$ nor $g1.2$ are ever applicable now, as they require the presence of the substring AB or CD (within w), respectively. Only then, the substrings fB or gD can be created.

This concludes our argument concerning the inductive step of the correctness proof of our suggested simulation.

Finally, Proposition 8 shows that star-controlled GCID systems of size $(6; 1, 0, 1; 2, 0, 0)$ are computationally complete, as well. □

Remark 10 *The reader might wonder if it would be possible to merge some of the components of the previous construction (Theorem 9), but this will create malicious derivations in each case. Also, the simulation of a p-rule cannot follow a simple pattern as that of the q-rule (see Rem. 10), as we want to avoid the derivation of strings with more than one occurrence of a symbol from N'. Here, we explain why a simple, not complex looking and seeming correct $p : X \to bY$ rule simulation does not work with the size $(5; 1, 1, 0; 2, 0, 0)$. Consider, if we attempt to construct a p-rule simulation for Π as in Table 4.*

A sample derivation of p-rule with the size $(5; 1, 1, 0; 2, 0, 0)$ is as follows.

$$(\alpha X \beta)_1 \Rightarrow_{p1.1} (\alpha X p \beta)_2 \Rightarrow_{p2.1} (\alpha p \beta)_1 \Rightarrow_{p1.2} (\alpha p p'' \beta)_4 \Rightarrow_{p4.1} (\alpha p p' p'' \beta)_1 \Rightarrow_{p1.3} (\alpha p p' b p'' \beta)_5$$
$$\Rightarrow_{p5.1} (\alpha b p'' \beta)_1 \Rightarrow_{p4} (\alpha b p'' Y \beta)_3 \Rightarrow_{p1.4} (\alpha b Y \beta)_1.$$

However, the simulation does not always work in the intended way as one need not apply p1.3 and instead p1.4 can be applied first. Therefore, the corresponding b is not inserted, however the Y has been inserted. With suitable a Y-rule that (finally) creates X again, later one could eliminate the markers pp' together and that will be a problem as a malicious string could be generated. For example, consider the grammar G contains the rules $p : X \to bY$ and $u : Y \to b'X$, $b, b' \in \{A, C\}$, $b \neq b'$, besides some other rules. Then, with the rules of Table 4, we can have the following derivation.

$$(\alpha X \beta)_1 \Rightarrow_{p1.1, p2.1, p1.2, p4.1, p1.4, p3.1} (\alpha pp'Y\beta)_1 \Rightarrow_{Y \to b'X}^{* \, simulating} (\alpha pp'b'X\beta)_1 \Rightarrow_{as \, earlier \, for \, X}^{*} (\alpha pp'b'pp'Y\beta)_1$$
$$\Rightarrow_{p1.3} (\alpha pp'b'pp'bY\beta)_5 \Rightarrow_{p5.1} (\alpha b'pp'bY\beta)_1 \Rightarrow_{p1.3} (\alpha b'pp'bbY\beta)_5 \Rightarrow_{p5.1} (\alpha b'bbY\beta)_1.$$

We are supposed to get $\alpha bb'bY\beta$ with G, but we could derive $\alpha b'bbY\beta$ with Π, but not in G.

Our next computational completeness result even further reduces the deletion complexity, making it context-free.

Theorem 11 $RE = GCID_S(4; 2, 1, 1; 1, 0, 0)$.

Component C1	Component C2	Component C3	Component C4
$p1.1 : (1, (\lambda, p, X)_I, 2)$ $p1.2 : (1, (p, bY, \lambda)_I, 3)$	$p2.1 : (2, (\lambda, X, \lambda)_D, 1)$	$p3.1 : (2, (\lambda, p, \lambda)_D, 1)$	
$q1.1 : (1, (\lambda, q, X)_I, 2)$ $q1.2 : (1, (q, Yb, \lambda)_I, 3)$	$q2.1 : (2, (\lambda, X, \lambda)_D, 1)$	$q3.1 : (2, (\lambda, q, \lambda)_D, 1)$	
$h1.1 : (1, (\lambda, hh', S')_I, 2)$ $h1.2 : (1, (\lambda, h', \lambda)_D, 3)$	$h2.1 : (3, (\lambda, S', \lambda)_D, 1)$	$h3.1 : (3, (\lambda, h, \lambda)_D, 1)$	
$f1.1 : (1, (\lambda, f', A)_I, 2)$ $f1.2 : (1, (\lambda, A, \lambda)_D, 4)$ $f1.3 : (1, (\lambda, f, \lambda)_D, 2)$ $f1.4 : (1, (\lambda, B, \lambda)_D, 4)$ $f1.5 : (1, (\lambda, f^3, \lambda)_D, 3)$ $f1.6 : (1, (\lambda, f''', \lambda)_D, 3)$ $f1.7 : (1, (\lambda, f^4, \lambda)_D, 2)$	$f2.1 : (2, (A, f, B)_I, 1)$ $f2.2 : (2, (B, f^4, \lambda)_I, 1)$ $f2.3 : (\lambda, f^2, \lambda)_D, 1)$	$f3.1 : (3, (3, (\lambda, f'', \lambda)_D, 1)$ $f3.2 : (3, (\lambda, f', \lambda)_D, 1)$	$f4.1 : (4, (f', f''f^2, f)_I, 1)$ $f4.2 : (4, (f^2, f'''f^3, f^4)_I, 1)$

Table 5: Star-controlled GCID of size $(4; 2, 1, 1; 1, 0, 0)$ simulating rules of SGNF.

<u>Proof</u> Consider a type-0 grammar $G = (N, T, P, S)$ in SGNF as in Definition 1. We construct a GCID system $\Pi = (4, V, T, \{S\}, H, 1, \{1\}, R)$ of size $(4; 2, 1, 1; 1, 0, 0)$ such that $L(\Pi) = L(G)$. The set V contains the symbols of G as well as some rule markers. We refer to Table 5 for the direct simulation of SGNF. The rules simulating $g : CD \to \lambda$ are similar to the ones simulating the $f : AB \to \lambda$ rule and hence omitted.

We now prove that $L(G) \subseteq L(\Pi)$. We show that if $w \Rightarrow w'$ in G, then $(w)_1 \Rightarrow_* (w')_1$ according to Π. From this fact, the claim follows by a simple induction, split into different cases, as discussed now.

<u>Context-free rule $p : X \to bY$.</u> Here, $w = \alpha X \beta$ and $w' = \alpha bY\beta$ for some $\alpha \in \{A, C\}^*$, $\beta \in (\{B, D\} \cup T)^*$. The simulation performs as follows:

$$(w)_1 = (\alpha X \beta)_1 \Rightarrow_{p1.1} (\alpha pX\beta)_2 \Rightarrow_{p2.1} (\alpha p\beta)_1 \Rightarrow_{p1.2} (\alpha pbY\beta)_3 \Rightarrow_{p3.1} (\alpha bY\beta)_1 = w'.$$

<u>Context-free rule $q : X \to Yb$</u> is simulated in a similar fashion to the simulation of the p-rule.

<u>Context-free rule $h : S' \to \lambda$</u> is simulated as follows for some $\alpha \in \{A, C\}^*$, $\beta \in (\{B, D\} \cup T)^*$:

$$(w)_1 = (\alpha S'\beta)_1 \Rightarrow_{h1.1} (\alpha hh'S'\beta)_2 \Rightarrow_{h2.1} (\alpha hh'\beta)_1 \Rightarrow_{h1.2} (\alpha h\beta)_3 \Rightarrow_{h3.1} (\alpha \beta)_1 = w'.$$

Non-context-free rule $f : AB \to \lambda$. This means that we expect $w = \alpha AB\beta$ and $w' = \alpha\beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\} \cup T)^*$. Within Π, this can be simulated as follows.

$$(\alpha AB\beta)_1 \Rightarrow_{f1.1} (\alpha f'AB\beta)_2 \Rightarrow_{f2.1} (\alpha f'AfB\beta)_1 \Rightarrow_{f1.2} (\alpha f'fB\beta)_4 \Rightarrow_{f4.1} (\alpha f'f''f^2fB\beta)_1 \Rightarrow_{f1.3} (\alpha f'f''f^2B\beta)_2$$
$$\Rightarrow_{f2.2} (\alpha f'f''f^2Bf^4\beta)_1 \Rightarrow_{f1.4} (\alpha f'f''f^2f^4\beta)_4 \Rightarrow_{f4.2} (\alpha f'f''f^2f'''f^3f^4\beta)_1 \Rightarrow_{f1.5} (\alpha f'f''f^2f'''f^4\beta)_3$$
$$\Rightarrow_{f3.1} (\alpha f'f^2f'''f^4\beta)_1 \Rightarrow_{f1.6} (\alpha f'f^2f^4\beta)_3 \Rightarrow_{f3.2} (\alpha f^2f^4\beta)_1 \Rightarrow_{f1.7} (\alpha f^2\beta)_2 \Rightarrow_{f2.3} (\alpha\beta)_1.$$

We next prove that $L(\Pi) \subseteq L(G)$. More precisely, we show the following (by induction). If $(u)_1$ is a configuration that is derivable in Π such that u contains the same number of markers from $\{f',g'\}$ as from $\{f^4,g^4\}$ (we call this property of the symbols from $\{f',g',f^4,g^4\}$ also a balanced situation) and such that the word u' that is obtained from u by deleting all symbols from $\{f',g',f^4,g^4\}$ belongs to $(N \cup T)^*$, then u is derivable in G. In particular, by induction we can assume that any such u that we discussed for proving the inductive step satisfies $u' \in \{A,C\}^*(N' \cup \{\lambda\})(\{B,D\} \cup T)^*$. To avoid clumsy formulations, we will discuss the markers from $\{f',g',f^4,g^4\}$ only in particular situations and argue why we maintain the property of being in a balanced situation. Hence, consider a derivation $(w)_1 \Rightarrow^* (w')_1$, with $w \neq w'$ and $w,w' \in \{A,C\}^*(N' \cup \{\lambda\})(\{B,D\} \cup T)^*$; it has to start like $(w)_1 \Rightarrow (v)_j$ in Π. If some rule from $C1$ is applied to w, the rule will insert a rule marker into the string w and move to $C2$, or an A or B is deleted and the string moves to $C4$. The introduction of rule markers in $C1$ will take care of the non-interference among the non-context-free and context-free rules. We will now discuss more details.

Let us first assume (by induction) that the sentential form $w^1 = \alpha X\beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\} \cup T)^*$ and $X \in N'$ is derivable in G and the configuration $(w^1)_1$ is derivable in Π. We prove (as induction step) that if $(w^1)_1 \Rightarrow_* (u)_1$, $w^1 \neq u$, and $u \in \{A,C\}^*(N' \cup \{\lambda\})(\{B,D\} \cup T)^*$ is the first sentential form in $\{A,C\}^*(N' \cup \{\lambda\})(\{B,D\} \cup T)^*$ that appears in a Π-derivation, then $w^1 \Rightarrow u$ in G.

Caveat: The reader might wonder why a context-free deletion of A or B as in rules $f1.2$ or $f1.4$ could work at all. But notice that in $C4$, which is the target component of these rules, the presence of f-style markers is checked in each of the rules. This prevents any successful derivation that interferes with, say, a p-rule derivation by deleting an A or a B in the first component, simply because there are no p-rule simulation rules in $C4$. We will hence tacitly assume that $f1.2$ etc. are not applied within p-simulations.

Applying $p1.1$ to $w^1 = \alpha X\beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\} \cup T)^*$ and as $X \in N'$ being the only nonterminal of N' that is in w^1, the marker p will be inserted to the left of X, yielding $(w^2)_2 = (\alpha pX\beta)_2$. In $C2$, the only available and applicable rule among the rules meant to simulate context-free rules deletes the nonterminal X (which is not S') of N'. This results in $(w^3)_1 = (\alpha p\beta)_1$. Alternatively, rule $f2.2$ (or $g2.2$) would be applicable in $C2$, bringing the string back to $C1$. There, one could return to $C2$ by using $f1.1$, $g1.1$, $f1.7$ or $g1.7$. The last two possibilities (if applicable at all) would just undo the previous step and hence offer no progress in the derivation, or they would exchange an occurrence of g^4 with an occurrence of f^4 or vice versa. The first two mentioned rules would add a symbol from $\{f',g'\}$ to the string obtained from w^3 by previously adding a symbol from $\{f^4,g^4\}$. Hence, if we think of the statement that our inductive argument should prove, we keep up a balanced situation as required. Now, in $C1$, the rule $p1.1$ cannot be applied again, as no $X \in N'$ is present. So, the only applicable rule (within the rules in $C1$ that are meant to deal with simulating context-free rules) is $p1.2$ which results in $(w^3)_3 = (\alpha pbY\beta)_3$. In $C3$, the introduced marker p is deleted and the resultant string $w^4 = \alpha bY\beta$ is sent to $C1$, thus the intended and desired derivation is correctly simulated. Notice that we could also apply $f3.2$ or $g3.2$ instead of $p3.1$, which would delete some f' or g' that might have been introduced earlier. However, this would then create an imbalanced situation, with less occurrences of symbols from $\{f',g'\}$ than from $\{f^4,g^4\}$ within a string in $C1$. As we will see, such an imbalance can never be resolved, so

that such a string will not yield a terminal string. There is one possibility after applying $p1.1$ and $p2.1$ that still needs to be discussed. It might be possible to apply $f1.1$ on $w^3 = \alpha p\beta$. This is only possible if $\alpha = \alpha_1 A\alpha_2$, so that we arrive at $(u^4)_2 = (\alpha_1 f'A\alpha_2 p\beta)_2$. Now, $f2.2$ or $g2.2$ may be applicable, bringing us back to $C1$. However, this will maintain a balanced situation as claimed. Moreover, as the reader can check, this is indeed the only possible continuation, keeping in mind that AB will not occur as a substring in the present string due to SGNF. So, in various ways, along with a simulation of a context-free rule, we may add symbols from $\{f', g', f^4, g^4\}$, but this always happens in a balanced way if it might be fruitful.

The correctness of the simulation of the <u>h-rule</u> is easily seen. Notice that there are again possibilities to introduce or delete symbols from $\{f', g', f^4, g^4\}$ similar as discussed above for simulating p-rules.

<u>Applying $f1.1$</u> to $w^1 = \alpha X\beta$ for some $\alpha \in \{A,C\}^*$, $\beta \in (\{B,D\} \cup T)^*$ and $X \in N'$, the marker f' will be inserted to the left of some occurrence of A, yielding $(w^2)_2 = (\alpha_1 f'A\alpha_2 X\beta)_2$, with $\alpha = \alpha_1 A\alpha_2$. As $w^1 = \alpha X\beta$ indicates that we are simulating phase I of the work of the SGNF grammar G, the substring AB is absent, preventing us from applying $f2.1$. We might continue with $f2.2$ or $g2.2$, though, which introduces an occurrence of f^4 or g^4, respectively. This is one possibility how we can obtain 'pairs' of occurrences of symbols from $\{f', g'\}$ and $\{f^4, g^4\}$, but we clearly maintain a balanced situation. Still, we might delete X with $p2.1$, say, but then we arrive at a typical imbalanced situation. We would have to apply $f1.1$ or $f1.2$ or $f1.4$ next, as no markers from the context-free rule simulations are present. In the first case, we see that we maintain an imbalanced situation of which we can never get rid, while in the second and third case, the derivation is blocked in $C4$ because of the lack of appropriate f-markers.

We could lead a similar discussion for <u>applying $f1.7$</u> to a string that contains one occurrence of N'-symbols and at least one occurrence of f^4; in particular, the arguments concerning (im)balance remain the same, because for this condition, it does not matter whether we add f' or delete f^4.

We shall discuss the cases for the <u>f-rule simulation</u> next, including that the rules $f1.i$ were applied in a wrong manner. Recall the discussion of the *Caveat* above which ruled out a premature application of $f1.2$ or of $f1.4$. We cannot start with any rules that require the presence of f-markers, apart from those stemming from a balanced situation that we will discuss below. We therefore discuss a derivation starting with $f1.1$ on w^1 as intended. Again by the absence of the marker f'', we have to apply $f2.1$ next, shifting the discussion of a balanced situation as created after applying $f2.2$ to what we say further down. The role of rule $f2.1$ is crucial insofar as it checks that the substring AB is present in the current string. It is one of the important properties of SGNF that this substring can only occur once in a derived string, and this also means that we are in phase II of the SGNF derivation. By induction, we can assume this property also to hold for the string w^1 that is under discussion. In other words, we can assume that $w^1 = \alpha AB\beta$. Then, after applying $f1.1$ and $f2.1$, we are in the configuration $(w^2)_1 = (\alpha' f'\alpha''AfB\beta)_1$ with $\alpha = \alpha'\alpha''$, and α'' being a string that is either empty or it starts with an A. The intention would be to apply $f1.2$ on w^2. As the marker f^2 is absent, we have to apply $f4.1$ now. This is only possible if α'' is empty. Hence, $(w^2)_1 = (\alpha f'AfB\beta)_1$, and after applying $f1.2$ and $f4.1$, we necessarily arrive at the configuration $(w^3)_1 = (\alpha f'f''f^2fB\beta)_1$. We now check the other possibilities in configuration $(w^2)_1$. Due to the absence of f^3, f''' and f^4, only $f1.1$, $f1.3$, or $f1.4$ are applicable. If we apply $f1.1$, then as there is no second substring AB nor the f-marker f^2 present, we have to introduce f^4 next to obtain a configuration $(u)_1$ where u contains two occurrences of f' and one occurrence of f and one of f^4. So, on $(u)_1$, we might apply $f1.2$, followed by $f4.1$. As argued above for the main line of derivation, this means that we arrive at a configuration $(v)_1$ with $v = \alpha_1 f'\alpha_2 f'f''f^2 f\beta_1 Bf^4\beta_2$, with $\alpha_1\alpha_2 = \alpha$ and $\beta_1 B\beta_2 = \beta$. In fact, we could continue now with the derivation, closely following the main line, the only difference being an additional f' and f^4 being present in the string. This indicates that balanced situations are not necessarily a problem and also shows how they can arise. On applying $f1.3$, we can delete f, but this

takes us back to $(w^2)_2$. This analysis tells us that, in configuration $(w^2)_1$, we have to apply $f1.2$.

Now, we study the configuration $(w^3)_1 = (\alpha f' f'' f^2 f B \beta)_1$. If we apply $f1.1$, we again have to introduce an occurrence of f^4 and can then follow the main line of derivation, which will finally lead to a balanced situation. If we delete A or B, the derivation is stuck in $C4$. Hence, the only applicable, promising rule is $f1.3$, deleting f and moving to $C2$. After deleting f^2, we get back to $C1$ with nearly the same (im)possibilities as just discussed, except that $f1.3$ is no longer available. Therefore, such a derivation cannot lead to a terminal string. Hence, in $C2$ the rule $f2.2$ must be applied, giving, with $\beta' \beta'' = \beta$ and β' being empty or ending with B, $(w^3)_1 = (\alpha f' f'' f^2 f B \beta' f^4 \beta'')_1$. Again, we might add a second f' with no fruitful continuation now, except for possibly creating other balanced situations finally. If we delete A, then the derivation is stuck in $C4$. If we delete f^4 with $f1.7$, we have to delete f^2 in $C2$ (unless we want to un-do $f1.7$ by applying $f2.2$) and then, whatever we apply next ($f1.1$ or $f1.2$ or $f1.4$), the derivation is stuck, ignoring the possibility to add more and more occurrences of f' and f^4. Therefore, on w^3, we have to apply $f1.4$ which deletes one B. In $C4$, we can only apply any rule if β' is empty, so that $f4.2$ is applicable, resulting in $(w^4)_1 = (\alpha f' f'' f^2 f''' f^3 f^4 \beta)_1$. The next six rule applications will delete all six f-markers; several ways to do this are possible. What could go wrong? Introducing a second f'-occurrence is again not interesting: it has to be matched by adding a f^4-occurrence; if we delete f^2 instead, then the addition of f^4 is indirect insofar, as $f1.7$ need not be executed to delete f^2. After applying $f1.5$, $f3.1$, $f1.6$ and $f3.2$, we end up with a string from which one could delete any A- or B-occurrence and try to restart with $f4.2$. Yet, now the symbols f''' and f^3 can only be deleted if one introduces two additional occurrences of $\{f', f''\}$, requiring a complete re-start of the simulation; it is not possible to get rid of f''' and f^3 otherwise. Such a re-start would require the substring AB which is currently not present. Hence, $(\alpha f^2 f^4 \beta)_1$ can only be continued as intended, applying $f1.7$ and $f2.3$.

<u>More on balanced situations.</u> As we have been mentioning these over and over again in the previous arguments, let us briefly discuss possibilities when we do have some balanced occurrences from $\{f', g', f^4, g^4\}$ in configuration $(w^1)_1$, e.g., $w^1 = \alpha A B \beta$ where $\alpha \in \{A, C\}^* \{f'\}$ and β contains exactly one occurrence from $\{f^4, g^4\}$. Then, instead of applying $f1.1$, we could start the derivation with $f1.7$ or $g1.7$. Notice that this yields a configuration $(w^2)_2$ that could have also been obtained when starting from $(\alpha' A B \beta')_1$ and applying $f1.1$, where α' is obtained from α by deleting f' and β' is obtained from β by deleting the unique occurrence of either f^4 or g^4. This shows that balanced situations can lead to terminal strings finally, as they may converge again to the main line of derivation. Importantly, no new terminal strings can be derived that are not following the possibilities given by G.

This concludes the main arguments concerning the inductive step and hence the claim follows.   □

4   Summary and Open Problems

In this paper, we focused on examining the computational power of graph-controlled ins-del systems with a star as a control graph. We lowered the resource requirements to describe RE, all recursively enumerable languages. We leave it open to explore the following possibilities.

1. $\text{GCID}_S(k; 2, i', i''; 1, j', j'') \overset{?}{=} \text{RE}$ for $i' + i'' \leq 1$ and $j' + j'' \leq 1$ and some k as small as possible,

2. $\text{GCID}_S(k'; 2, 0, 0; 1, i', i'') \overset{?}{=} \text{RE}$ for $i' + i'' \leq 1$ and some k' as small as possible.

Here we only considered GCID systems where the underlying graph is star-controlled and does not contain loops. One may also consider a tree structure and / or the possibility to allow loops (i.e., rules have option 'here' and the resultant string can stay back in the same component if such rules are applied), which may give additional power and connect closer to ins-del P systems and also to the results of [9].

References

[1] A. Alhazov, R. Freund, S. Ivanov & S. Verlan (2022): *Regulated Insertion-Deletion Systems*. Journal of Automata, Languages and Combinatorics 27(1-3), pp. 15–45, doi:10.25596/jalc-2022-015.

[2] R. Benne, editor (1993): *RNA Editing: The Alteration of Protein Coding Sequences of RNA*. Ellis Horwood.

[3] H. Fernau, L. Kuppusamy & I. Raman (2017): *On the computational completeness of graph-controlled insertion-deletion systems with binary sizes*. Theoretical Computer Science 682, pp. 100–121, doi:10.1016/j.tcs.2017.01.019.

[4] H. Fernau, L. Kuppusamy & I. Raman (2019): *On path-controlled insertion-deletion systems*. Acta Informatica 56(1), pp. 35–59, doi:10.1007/s00236-018-0312-2.

[5] H. Fernau, L. Kuppusamy & I. Raman (2021): *On the generative capacity of matrix insertion-deletion systems of small sum-norm*. Natural Computing 20(4), pp. 671–689, doi:10.1007/s11047-021-09866-y.

[6] R. Freund, M. Kogler, Y. Rogozhin & S. Verlan (2010): *Graph-Controlled Insertion-Deletion Systems*. In I. McQuillan & G. Pighizzini, editors: *Proceedings Twelfth Annual Workshop on Descriptional Complexity of Formal Systems, DCFS*, EPTCS 31, pp. 88–98, doi:10.4204/EPTCS.31.11.

[7] V. Geffert (1991): *Normal forms for phrase-structure grammars*. RAIRO Informatique théorique et Applications/Theoretical Informatics and Applications 25, pp. 473–498, doi:10.1051/ita/1991250504731.

[8] S. Ivanov & S. Verlan (2014): *About One-Sided One-Symbol Insertion-Deletion P Systems*. In A. Alhazov, S. Cojocaru, M. Gheorghe, Y. Rogozhin, G. Rozenberg & A. Salomaa, editors: *Membrane Computing - 14th Int. Conf., CMC 2013*, LNCS 8340, Springer, pp. 225–237, doi:10.1007/978-3-642-54239-8_16.

[9] S. Ivanov & S. Verlan (2017): *Universality and Computational Completeness of Controlled Leftist Insertion-Deletion Systems*. Fundamenta Informaticae 155(1-2), pp. 163–185, doi:10.3233/FI-2017-1580.

[10] L. Kari, Gh. Păun, G. Thierrin & S. Yu (1999): *At the crossroads of DNA computing and formal languages: Characterizing recursively enumerable languages using insertion-deletion systems*. In: Discrete Mathematics and Theretical Computer Science, DIMACS 48, AMS, pp. 329–338, doi:10.1090/dimacs/048/23.

[11] L. Kari & G. Thierrin (1996): *Contextual Insertions/Deletions and Computability*. Information and Computation 131(1), pp. 47–61, doi:10.1006/inco.1996.0091.

[12] A. Krassovitskiy, Y. Rogozhin & S. Verlan (2008): *Further Results on Insertion-Deletion Systems with One-Sided Contexts*. In C. Martín-Vide, F. Otto & H. Fernau, editors: *Language & Automata Theory & Applications, LATA*, LNCS 5196, Springer, pp. 333–344, doi:10.1007/978-3-540-88282-4_31.

[13] S. N. Krishna & R. Rama (2002): *Insertion-Deletion P Systems*. In N. Jonoska & N. C. Seeman, editors: *DNA Computing, 7th Int. Workshop on DNA-Based Computers, 2001*, LNCS 2340, Springer, pp. 360–370, doi:10.1007/3-540-48017-X_34.

[14] S. Marcus (1969): *Contextual grammars*. Revue Roumaine de Mathématiques Pures et Appliquées 14, pp. 1525–1534.

[15] A. Matveevici, Y. Rogozhin & S. Verlan (2007): *Insertion-Deletion Systems with One-Sided Contexts*. In: *MCU*, LNCS 4664, Springer, pp. 205–217, doi:10.1007/978-3-540-74593-8_18.

[16] Gh. Păun (1997): *Marcus Contextual Grammars*. Studies in Linguistics and Philosophy 67, Kluwer, doi:10.1007/978-94-015-8969-7_4.

[17] Gh. Păun, G. Rozenberg & A. Salomaa (1998): *DNA Computing: New Computing Paradigms*. Springer, doi:10.1007/978-3-662-03563-4.

[18] I. Petre & S. Verlan (2012): *Matrix insertion-deletion systems*. Theoretical Computer Science 456, pp. 80–88, doi:10.1016/j.tcs.2012.07.002.

[19] A. Takahara & T. Yokomori (2003): *On the computational power of insertion-deletion systems*. Natural Computing 2(4), pp. 321–336, doi:10.1023/B:NACO.0000006769.27984.23.

[20] S. Verlan (2010): *Recent Developments on Insertion-Deletion Systems*. The Computer Science Journal of Moldova 18(2), pp. 210–245.