PAPER Special Section on Foundations of Computer Science —Frontiers of Theory of Computation and Algorithm—

Generalized Register Context-Free Grammars

Ryoma SENDA[†a)], *Nonmember*, Yoshiaki TAKATA[††b)], *Member*, *and* Hiroyuki SEKI[†c)], *Fellow*

SUMMARY Register context-free grammars (RCFG) is an extension of context-free grammars to handle data values in a restricted way. In RCFG, a certain number of data values in registers are associated with each nonterminal symbol and a production rule has the guard condition, which checks the equality between the content of a register and an input data value. This paper starts with RCFG and introduces register type, which is a finite representation of a relation among the contents of registers. By using register type, the paper provides a translation of RCFG to a normal form and ε-removal from a given RCFG. We then define a generalized RCFG (GRCFG) where an arbitrary binary relation can be specified in the guard condition. Since the membership and emptiness problems are shown to be undecidable in general, we extend register type for GRCFG and introduce two properties of GRCFG, simulation and progress, which guarantee the decidability of these problems. As a corollary, these problems are shown to be EXPTIME-complete for GRCFG with a total order over a dense set.

key words: *register context-free grammar, register type, computational complexity, data word, data language*

1. Introduction

This paper focuses on register context-free grammars (abbreviated as RCFG), which were introduced by Cheng and Kaminsky in 1998 [5]. Recently, register automata (abbreviated as RA) [9] have been paid attention [10]–[12] as a core computational model of query languages for structured documents with data values such as XPath. For example, XPath can specify both a regular pattern of node labels (e.g., element names) and a constraint on data values (e.g., attribute values and PCDATA) in a tree representing an XML document. While RA have a power sufficient for expressing regular patterns on *paths* of a tree or a graph, it cannot represent tree patterns (or patterns over branching paths) that can be represented by some query languages such as XPath. Hence, a computational model that can represent both local tree patterns and constraints on data values is expected.

RCFG [5] is defined as an extension of CFG in a similar way to extending finite automata to RA. In a derivation of a k-RCFG, k data values are associated with each occurrence of a nonterminal symbol (called an *assignment*) and a production rule can be applied only when the guard condition of the rule, which is a Boolean combination of the equality check between an input data value and the data value in a register, is satisfied. In [5], properties of RCFG were shown including the decidability of the membership and emptiness problems, and the closure properties. In our previous study [16], the membership problem for RCFG, ε-rule free RCFG and growing RCFG are shown EXPTIME-complete, PSPACE-complete and NP-complete, respectively, and the emptiness problem for these classes are shown EXPTIME-complete.

In this paper, we first show that ε-rules can be removed from a given RCFG without changing the generated language. To prove this property, we introduce a notion called *register type*, which is the quotient of registers by the equivalence classes induced by equality relation among the contents of registers. Next, we move to the main topic of this paper, a generalization of RCFG abbreviated as GRCFG. As we mentioned, what an RCFG (and also an RA) can do when applying a rule is the equality check between the content of a register and an input data value. Then, we come to a natural question that what happens if we allow the check of an arbitrary relation (such as the total order on numbers). Generally, basic problems including membership and emptiness become undecidable even if the relations used in the guard conditions are all decidable. Hence, we want to introduce appropriate conditions for such extensions of RCFG to keep the decidability and complexity of those problems unchanged. For this aim, we extend the above mentioned *register type* for an arbitrary relation and then we introduce two properties, namely, simulation and progress. We then provide an example of decidable relations that satisfy the simulation property but do not satisfy the progress property, and the membership and emptiness are still undecidable. We show that the emptiness and membership become decidable and ε-removal is possible for GRCFG that have both the simulation and progress properties. As a corollary, we show that those problems are decidable for GRCFG with a total order on a dense set.

Related work Register automata (RA) was proposed in [9] as finite-memory automata where they show that the membership and emptiness problems are decidable, and the class of languages recognized by RA are closed under union, concatenation and Kleene-star. Later, the computational complexity of the above two problems are analyzed in [6], [15]. In [5], register context-free grammars (RCFG) as well as pushdown automata over an infinite al-

Manuscript received April 26, 2019.
Manuscript revised September 5, 2019.
Manuscript publicized November 21, 2019.

[†]The authors are with Graduate School of Informatics, Nagoya University, Nagoya-shi, 464–8601 Japan.
[††]The author is with Graduate School of Engineering, Kochi University of Technology, Kami-shi, 782–8502 Japan.
a) E-mail: ryoma.private@sqlab.jp
b) E-mail: takata.yoshiaki@kochi-tech.ac.jp
c) E-mail: seki@i.nagoya-u.ac.jp

DOI: 10.1587/transinf.2019FCP0010

phabet were introduced and the equivalence of the two models were shown. Also, the decidability of membership and emptiness problems and the closure under union, concatenation, Kleene-star were shown in [5]. Extension of RA to a totally ordered set was discussed in [1], [8], which is also provided in RCFG in the last section of this paper.

There have been many studies on other extensions of finite models to deal with data values in restricted ways. *Other automata for data words*: As extensions of finite automata other than RA, data automata [4], pebble automata (PA) [13] and nominal automata (NA) [3] are known. Libkin and Vrgoč [12] argue that register automata (RA) is the only model that has efficient data complexity for membership among the above mentioned formalisms. Neven, et al. consider variations of RA and PA, which are either one way or two ways, deterministic, nondeterministic or alternating. They show inclusion and separation relationships among these automata, FO($\sim, <$) and EMSO($\sim, <$), and give the answer to some open problems including the undecidability of the universality problem for RA [14]. Nominal (G-)automata (NA) is defined by a data set with symmetry and finite supports, and properties of NA are investigated including Myhill-Nerode theorem, closure and determinization in [3]. (Usual) RA with equality and RA with total order can be regarded as NA where the data sets have equality symmetry and total order symmetry, respectively. In [3], nominal CFG is also introduced but decidability of related problems is not discussed. Finiteness of orbits and that of register types in this paper may be related, but deeper observation is left as future study. *LTL with freeze quantifier*: Linear temporal logic (LTL) was extended to LTL$\downarrow$ with freeze quantifier [6], [7]. The relationship among subclasses of LTL$\downarrow$ and RA as well as the decidability and complexity of the satisfiability (nonemptiness) problems are investigated [6]. They especially showed that the emptiness problem for (both nondeterministic and deterministic) RA are PSPACE-complete. *Two-variable logics with data equality*: It is known that two-variable FO$^2(<, +1)$ where $<$ is the ancestor-descendant relation and $+1$ is the parent-child relation is decidable and corresponds to Core XPath. The logic was extended to those with data equality. It was shown in [2] that FO$^2(\sim, <, +1)$ with data equality $\sim$ is decidable on data words. Note that FO$^2(\sim, <, +1)$ is incomparable with LTL$\downarrow$ of [6].

This paper is an extended version of [17] by adding non-trivial formal proofs, revising some important concepts including simulation and progress and showing some undecidability results when these two properties do not hold.

2. Register Context-Free Grammars

Let $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ be the set of natural numbers including 0. We assume an infinite set D of *data values* as well as a finite alphabet Σ. For a given $k \in \mathbb{N}_0$ specifying the number of *registers*, a mapping $\theta : [k] \to D$ is called an *assignment* (of data values to k registers) where $[k] = \{1, 2, \ldots, k\}$. We assume that a data value $\perp \in D$ is designated as the initial value of a register. Let Θ_k denote the collection of assignments to k registers. For $\theta, \theta' \in \Theta_k$, we write $\theta' = \theta[i \leftarrow d]$ if $\theta'(i) = d$ and $\theta'(j) = \theta(j)$ for $j \neq i$. Let F_k denote the set of *guard expressions* over k registers defined by $\psi := \mathtt{tt} \mid x_i^= \mid \neg\psi \mid \psi \vee \psi$ where $x_i \in \{x_1, \ldots, x_k\}$. Let $\mathtt{ff}, x_i^{\neq}, \psi_1 \wedge \psi_2$ denote $\neg\mathtt{tt}, \neg x_i^=, \neg(\neg\psi_1 \vee \neg\psi_2)$, respectively. The description length of a guard expression ψ, denoted as $\|\psi\|$, is defined as usual where $\|x_i^=\| = 1 + \log k$. For $\theta \in \Theta_k$, $d \in D$ and $\psi \in F_k$, the satisfaction relation $\theta, d \models \psi$ is defined as $\theta, d \models x_i^=$ iff $\theta(i) = d$ and is recursively defined for $\neg$ and $\vee$ in a usual way.

For a finite alphabet Σ and a set D of data values disjoint from Σ, a *data word* over $\Sigma \times D$ is a finite sequence of elements of $\Sigma \times D$ and a *data language* over $\Sigma \times D$ is a subset of $(\Sigma \times D)^*$. $|\beta|$ denotes the cardinality of β if β is a set and the length of β if β is a finite sequence.

For $k \in \mathbb{N}_0$, a *k-register context-free grammar* (*k*-RCFG) over Σ and D is a triple $G = (V, R, S)$ where

- V is a finite set of nonterminal symbols (abbreviated as nonterminals) where $V \cap (\Sigma \cup D) = \emptyset$,
- R is a finite set of production rules (abbreviated as rules) having either of the following forms: $(A, \psi, i) \to \alpha$ or $(A, \psi) \to \alpha$ where $A \in V$, $\psi \in F_k$, $i \in [k]$ and $\alpha \in (V \cup (\Sigma \times [k]))^*$; we call (A, ψ, i) (or (A, ψ)) the left-hand side and α the right-hand side of the rule, and,
- $S \in V$ is the start symbol.

A rule whose right-hand side is ε is an *ε-rule*. If R contains no ε-rule, G is called *ε-rule free*. A k-RCFG G for some $k \in \mathbb{N}_0$ is just called an RCFG.

In the following, we write $(A, \psi, i)/(A, \psi) \to \alpha \in R$ to represent $(A, \psi, i) \to \alpha \in R$ or $(A, \psi) \to \alpha \in R$. The description length of a k-RCFG $G = (V, R, S)$ is defined as $\|G\| = |V| + |R| \max\{(|\alpha| + 1)(\log|V| + \log k) + \|\psi\| \mid (A, \psi, i)/(A, \psi) \to \alpha \in R\}$.

We define $\Rightarrow_G$ as the smallest relation containing the instantiations of rules in R and closed under the context as follows. For $A \in V$, $\theta \in \Theta_k$ and $X \in ((V \times \Theta_k) \cup (\Sigma \times D))^*$, we say (A, θ) directly derives X, written as $(A, \theta) \Rightarrow_G X$ if there exist $d \in D$ (regarded as an input data value) and $r = (A, \psi, i) \to c_1 \ldots c_n \in R$ (resp. $r = (A, \psi) \to c_1 \ldots c_n \in R$) such that

$$\theta, d \models \psi,\ X = c_1' \ldots c_n',\ \theta' = \theta[i \leftarrow d] \text{ (resp. } \theta' = \theta) \text{ where}$$
$$c_j' = \begin{cases} (B, \theta') & \text{if } c_j = B \in V, \\ (b, \theta'(l)) & \text{if } c_j = (b, l) \in \Sigma \times [k]. \end{cases}$$

For $X, Y \in ((V \times \Theta_k) \cup (\Sigma \times D))^*$, we also write $X \Rightarrow_G Y$ if there are $X_1, X_2, X_3 \in ((V \times \Theta_k) \cup (\Sigma \times D))^*$ such that $X = X_1(A, \theta)X_2$, $Y = X_1X_3X_2$ and $(A, \theta) \Rightarrow_G X_3$. If we want to emphasize the applied rule r and the input data value d, we write $X \Rightarrow_{G,r}^{d} Y$.

Let $\overset{*}{\Rightarrow}_G$ and $\overset{+}{\Rightarrow}_G$ be the reflexive transitive closure and the transitive closure of $\Rightarrow_G$, respectively, called the derivation relation of zero or more steps (resp. the derivation relation of one or more steps). We abbreviate $\Rightarrow_G$, $\overset{*}{\Rightarrow}_G$ and $\overset{+}{\Rightarrow}_G$

as $\Rightarrow$, $\overset{*}{\Rightarrow}$ and $\overset{+}{\Rightarrow}$ if G is clear from the context.

We denote by $\theta_\perp$ the assignment that assigns the initial value $\perp$ to every register. We let $L(G) = \{w \mid (S, \theta_\perp) \overset{+}{\Rightarrow} w \in (\Sigma \times D)^*\}$. $L(G)$ is called the data language generated by G. $(S, \theta_\perp) \overset{+}{\Rightarrow} w$ is called a derivation of w in G. RCFGs G_1 and G_2 are *equivalent* if $L(G_1) = L(G_2)$.

Example 2.1: For $\Sigma = \{a, b\}$, let $G = (\{S, A\}, R, S)$ be a 2-RCFG where $R = \{(S, \mathtt{tt}, 1) \to (a, 1)A(a, 1),\ (A, x_1^{\neq}, 2) \to (b, 2)A(b, 2),\ (A, x_1^{=}) \to (a, 1)\}$. Then, $L(G) = \{(a, d_0)(b, d_1) \dots (b, d_n)(a, d_0)(b, d_n) \dots (b, d_1)(a, d_0) \mid n \geq 0, d_i \neq d_0 \text{ for } i \in [n]\}$.

Theorem 2.1: For an arbitrary k-RCFG $G = (V, R, S)$, we can construct an equivalent $(k + 1)$-RCFG $G' = (V', R', S')$ such that R' has no rule of the form $(A, \psi) \to \alpha$.

Proof We construct G' by using the $(k + 1)$th register as dummy:

- $V' = V, S' = S$,
- $R' = \{(A, \psi, i) \to \alpha \mid (A, \psi, i) \to \alpha \in R \text{ or } ((A, \psi) \to \alpha \in R \wedge i = k + 1)\}$.

3. Register Type, Normal Forms and ε-Rule Removal

3.1 Register Type

In this subsection, we will define register type, which is useful in expressing equalities among the contents of registers, transforming a given RCFG into a certain normal form and proving some important properties of RCFG. The idea is simple; instead of remembering concrete data values in registers, it suffices to remember the induced equivalence classes of the indices of registers as long as the equalities among data values in the registers are concerned.

Definition 3.1: A decomposition of $[k]$ into disjoint non-empty subsets is called a *register type* of k-RCFG. Let Γ_k denote the collection of all register types of k-RCFG. For a register type $\gamma \in \Gamma_k$, let $\gamma[i]$ ($i \in [k]$) denote the subset containing i. □

For example, $\gamma_1 = \{\{1, 2\}, \{3, 5\}, \{4\}\}$ is a register type of 5-RCFG and $\gamma_1[1] = \{1, 2\}$, $\gamma_1[5] = \{3, 5\}$. For an assignment $\theta \in \Theta_k$ and a register type $\gamma \in \Gamma_k$, we define the typing relation as:

$$\theta \models \gamma :\Longleftrightarrow \forall i, j.(\theta(i) = \theta(j) \Longleftrightarrow \gamma[i] = \gamma[j]).$$

For example, $\theta_1 \in \Theta_5$ such that $\theta_1(1) = \theta_1(2) = 8$, $\theta_1(3) = \theta_1(5) = 10$, $\theta_1(4) = 5$ satisfies $\theta_1 \models \gamma_1$. By definition, for each $\theta \in \Theta_k$, there is exactly one $\gamma \in \Gamma_k$ such that $\theta \models \gamma$. In this case, we say that the type of θ is γ.

3.2 Normal Forms for Guard Expressions

By using register types, we show that a given RCFG can be transformed into an equivalent RCFG G' such that for any rule $r = (A, \psi, i)/(A, \psi) \to \alpha$, there must exist an input data value d for any (A, θ) that enables r to be applied to (A, θ), that is, the guard ψ never blocks any (A, θ) and only specifies the equality or inequality among an input data value d and the current contents of the registers. This transformation is the key of the ε-rule removal shown in the next subsection.

First, it is easy to transform a given k-RCFG into an equivalent k-RCFG where the guard expression ψ of every rule has the following form:

$$\psi = (x_{i_1}^{=} \wedge \dots \wedge x_{i_m}^{=}) \wedge (x_{j_1}^{\neq} \wedge \dots \wedge x_{j_n}^{\neq}). \tag{1}$$

The above guard can be obtained by the following equivalence transformations:

1. Transform the guard expression of every rule to an equivalent disjunctive normal form.
2. Replace a rule $(A, \psi_1 \vee \psi_2, i) \to \alpha$ into $(A, \psi_1, i) \to \alpha$ and $(A, \psi_2, i) \to \alpha$.

For a guard expression ψ in (1), we let $\psi^{=} = \{i_1, \dots, i_m\}$ and $\psi^{\neq} = \{j_1, \dots, j_n\}$. For $\gamma \in \Gamma_k$ and $\psi \in F_k$ in the form of (1), we define

$$\gamma \models \psi :\Longleftrightarrow \bigwedge_{i \in \psi^{=}} (\bigwedge_{j \in \psi^{=}} \gamma[i] = \gamma[j] \wedge \bigwedge_{j \in \psi^{\neq}} \gamma[i] \neq \gamma[j]).$$

Note that $\psi^{=} = \emptyset$ implies $\gamma \models \psi$ for any γ. It is easy to see that the following property holds, which means that for an assignment θ that conforms to γ, there is a data value d that satisfies ψ if and only if $\gamma \models \psi$.

$$\theta \models \gamma \ \Rightarrow\ (\gamma \models \psi \Longleftrightarrow \exists d.\ \theta, d \models \psi). \tag{2}$$

Lemma 3.1: For an arbitrary k-RCFG G, we can construct a k-RCFG G' such that $L(G') = L(G)$ and the guard expression of every rule in G' is one of the following $k + 1$ expressions: $x_1^{=}, x_2^{=}, \dots, x_k^{=}, x_1^{\neq} \wedge \dots \wedge x_k^{\neq}$.

Proof Let $\varphi_j = x_j^{=}$ for $j \in [k]$ and $\varphi_* = x_1^{\neq} \wedge \dots \wedge x_k^{\neq}$. Obviously, the following property holds:

$$\begin{array}{l}\text{For any } \theta, d, \text{ and } \psi,\ \theta, d \models \psi \text{ iff } \theta, d \models \psi \wedge \varphi_j \\ \text{for some } j \in [k] \cup \{*\}.\end{array} \tag{3}$$

For an assignment θ whose type is γ, consider an updated assignment $\theta[i \leftarrow d]$ obtained by a single derivation step. Then one of the following possibilities must hold:

(i) $d = \theta(j)$ for some $j \in [k]$ (i.e. $\theta, d \models \varphi_j$).

(ii) $d \neq \theta(j)$ for any $j \in [k]$ (i.e. $\theta, d \models \varphi_*$).

In case (i), the type of $\theta[i \leftarrow d]$ is uniquely determined by γ, i, and j and regardless of θ. Similarly, the type of $\theta[i \leftarrow d]$ in case (ii) is uniquely determined by γ and i. We denote the type of $\theta[i \leftarrow d]$ in cases (i) and (ii) by *after*(γ, i, j) and *after*$(\gamma, i, *)$, respectively. These register types can be specifically described as follows:

- For $j \in [k]$, *after*$(\gamma, i, j) = \gamma'$ where $\gamma'[i] = \gamma'[j] = \gamma[j] \cup \{i\}$ and $\gamma'[j'] = \gamma[j'] \setminus \{i\}$ for $\forall j' \in [k] \setminus \gamma'[i]$.

- $\mathit{after}(\gamma, i, *) = \gamma'$ where $\gamma'[i] = \{i\}$ and $\gamma'[j'] = \gamma[j'] \setminus \{i\}$ for $\forall j' \in [k] \setminus \gamma'[i]$.

We construct k-RCFG $G' = (V', S', R')$ from $G = (V, S, R)$ where $V' = V \times \Gamma_k$, $S' = (S, \{[k]\})$, and R' is the smallest set that satisfies the following inference rules, where $\alpha^{\mathrm{aug}(\gamma')}$ is the sequence obtained from α by replacing every occurrence of every nonterminal A in V with (A, γ'); that is, $(X_1 \dots X_n)^{\mathrm{aug}(\gamma')} = X'_1 \dots X'_n$ where $X'_\ell = (X_\ell, \gamma')$ if $X_\ell \in V$ and $X'_\ell = X_\ell$ otherwise for every $\ell \in [n]$. In the following rules, φ_j is the expression defined in the beginning of the proof of this lemma. We assume that the guard expression of every rule of G has the form of (1).

$$\frac{\begin{array}{l}(A, \psi, i) \to \alpha \in R \\ j \in [k] \cup \{*\} \quad \gamma \models \psi \wedge \varphi_j \quad \gamma' = \mathit{after}(\gamma, i, j)\end{array}}{((A, \gamma), \varphi_j, i) \to \alpha^{\mathrm{aug}(\gamma')} \in R'}$$

$$\frac{(A, \psi) \to \alpha \in R \quad \gamma \models \psi}{((A, \gamma), \varphi_1) \to \alpha^{\mathrm{aug}(\gamma)} \in R'}$$

Note that φ_1 in the second inference rule is used as an equivalent of `tt`, because for any assignment θ, there must be a data value d ($= \theta(1)$) that satisfies φ_1, and d is not stored in a register when that rule is applied.

We can show the following properties, which establish the lemma.

- For every $((A, \gamma), \theta)$ appearing in a derivation in G' starting from $(S', \theta_\perp)$, it holds that $\theta \models \gamma$. This can be proved by induction on the length of the derivation.
- For a derivation of a data word w in G, if we replace every occurrence of $(A, \theta) \in V \times \Theta_k$ with $((A, \gamma), \theta)$ where γ is the type of θ, then we obtain a derivation of w in G'. This can be proved as follows: For each derivation step of G where a rule $(A, \psi, i) \to \alpha$ is applied to (A, θ) with an input data value d (i.e. $\theta, d \models \psi$), there must exist some φ_j ($j \in [k] \cup \{*\}$) such that $\theta, d \models \psi \wedge \varphi_j$ by property (3). By property (2), $\gamma \models \psi \wedge \varphi_j$ holds, and thus G' has the rule $((A, \gamma), \varphi_j, i) \to \alpha^{\mathrm{aug}(\gamma')}$. This rule allows the simulating derivation step in G' applied to $((A, \gamma), \theta)$ using the same input data value d, which yields the same updated assignment $\theta[i \leftarrow d]$ as the derivation step in G.
- For a derivation of a data word w in G', if we replace each $((A, \gamma), \theta)$ with (A, θ), then we obtain a derivation of w in G. This can be proved as follows: By the property before the previous one, $\theta \models \gamma$ holds. Consider a derivation step of G' where a rule $((A, \gamma), \varphi_j, i) \to \alpha^{\mathrm{aug}(\gamma')}$ is applied to $((A, \gamma), \theta)$ with an input data value d (i.e. $\theta, d \models \varphi_j$). Then G must have a rule $r = (A, \psi, i) \to \alpha$ and $\gamma \models \psi \wedge \varphi_j$. By property (2), there is some d' such that $\theta, d' \models \psi \wedge \varphi_j$. If $j \in [k]$, then d' and d are the same ($= \theta(j)$). If $j = *$, then d must satisfy $\theta, d \models \psi \wedge \varphi_*$ because $\psi \wedge \varphi_*$ must be equivalent to φ_* (if not, $\theta, d' \not\models \psi \wedge \varphi_*$ for any d'). Since $\theta, d \models \psi \wedge \varphi_j$ implies $\theta, d \models \psi$, we can apply r to (A, θ) with the same input data value d.

3.3 ε-Rule Removal

Theorem 3.1: For an arbitrary k-RCFG G, we can construct an ε-rule free k-RCFG G' that satisfies $L(G') = L(G) \setminus \{\varepsilon\}$.

Proof By Lemma 3.1, we can transform any k-RCFG $G = (V, R, S)$ into another k-RCFG $G'' = (V'', R'', S'')$ such that $L(G'') = L(G)$ and the guard expression of every rule in G'' is either $x_1^=, \dots, x_k^=$, or $x_1^{\neq} \wedge \dots \wedge x_k^{\neq}$. Because the guard expressions of G'' never block the application of each rule, we can compute the set Nu of nullable nonterminals (i.e. the set that consists of every nonterminal A such that $(A, \theta) \Rightarrow^*_{G''} \varepsilon$ regardless of θ) in the same way as CFG; that is, we can compute Nu as the smallest set that satisfies the following conditions:

- If $(A, \psi, i)/(A, \psi) \to \varepsilon \in R''$, then $A \in Nu$.
- If $(A, \psi, i)/(A, \psi) \to \alpha \in R''$ and α consists of nonterminals in Nu, then $A \in Nu$.

And thus we can remove the ε-rules of G'' also in the same way as CFG; that is, for each $(A, \psi, i)/(A, \psi) \to \alpha \in R''$ such that $\alpha \neq \varepsilon$, we construct $(A, \psi, i)/(A, \psi) \to \alpha'$ where $\alpha' \neq \varepsilon$ and α' is obtained from α by removing zero or more occurrence of nonterminals in Nu. Let $G' = (V'', R', S'')$ be the resultant RCFG. We can show that for every $A \in V''$, $\theta \in \Theta_k$, and a data word w, $(A, \theta) \Rightarrow^*_{G''} w$ and $w \neq \varepsilon$ if and only if $(A, \theta) \Rightarrow^*_{G'} w$, by induction on the length of the derivation.

4. Generalized RCFG

4.1 Definitions

We define generalized register context-free grammar by allowing an arbitrary binary relation on the set of data values. Let Σ be a finite alphabet, D be a set of data values such that $\Sigma \cap D = \emptyset$ equipped with a finite set of binary relations $\mathcal{R}$. We call $(D, \mathcal{R})$ a *data structure*. For $k \in \mathbb{N}_0$, a *generalized k-register context-free grammar* (k-GRCFG) is a triple $G = (V, R, S)$ where V, R and S are the same as in k-RCFG except that an atomic formula in a guard expression is $x_i^{\bowtie}$ and $x_i^{\bowtie^{-1}}$ ($i \in [k]$, $\bowtie \in \mathcal{R}$) and its semantics is defined by

$$\theta, d \models x_i^{\bowtie} \text{ iff } \theta(i) \bowtie d \text{ and } \theta, d \models x_i^{\bowtie^{-1}} \text{ iff } d \bowtie \theta(i)$$

for any $\theta \in \Theta_k$ and $d \in D$. We sometimes write k-GRCFG($\mathcal{R}$) to emphasize $\mathcal{R}$ and abbreviate it as k-GRCFG($\bowtie$) when $\mathcal{R} = \{\bowtie\}$. Notions and notations for RCFG such as ε-rule, derivation relation $\Rightarrow$, the data language $L(G)$ generated by G are defined in the same way. We also write k-GRCFG($=$) to denote a (usual) k-RCFG.

Closure and non-closure properties of GRCFG described in the following Theorem 4.1 can be proved in a similar way to the case of the usual RCFG [5], [9]. In that theorem, a homomorphism is a function $h : (\Sigma \times D)^* \to (\Sigma' \times D')^*$

where Σ, Σ' are finite alphabets and D, D' are infinite sets of data values such that $h(w_1 \cdots w_n) = h(w_1) \cdots h(w_n)$ holds for $w_1, \ldots, w_n \in (\Sigma \times D)^*$. Theorem 4.2 is an extension of 2.1 to GRCFG and can be proved in the same way.

Theorem 4.1: The class of data languages generated by k-GRCFG($\mathcal{R}$) is closed under union, concatenation and Kleene-closure. It is not closed under intersection, complement, homomorphisms or inverse homomorphisms.

Theorem 4.2: For an arbitrary k-GRCFG $G = (V, R, S)$, we can construct an equivalent $(k+1)$-GRCFG $G' = (V', R', S')$ such that R' has no rule of the form $(A, \psi) \to \alpha$.

4.2 Simulation and Progress Properties

In Sect. 3, we showed that a given RCFG can be transformed to an equivalent RCFG where the guard expression of a production rule never blocks its application by associating a register type with each nonterminal symbol. We can extend register type to GRCFG in a natural way, but the above transformation cannot guarantee the equivalence because the register type no longer has information enough to represent the applicability of a rule in GRCFG.

Example 4.1: Consider the set of integers with the usual strict total order $\mathbb{Z} = (Z, \{<_Z, >_Z\})$ as a data structure. We might extend register type of GRCFG(=) by introducing $<_Z$ among the equivalence classes of $[k]$. For example, let $\psi = x_1^{<_Z} \wedge x_2^{>_Z}$ be a guard expression of 3-GRCFG($\{<_Z, >_Z\}$) and consider assignments $\theta_1, \theta_2 \in \Theta_3$ such that $\theta_1(1) = \theta_1(3) = 4$, $\theta_1(2) = 7$ and $\theta_2(1) = \theta_2(3) = 5$, $\theta_2(2) = 6$. Also let γ be the register type (informally) defined as $\gamma = \{\{1, 3\} <_Z \{2\}\}$. Both $\theta_1 \models \gamma$ and $\theta_2 \models \gamma$ hold. However, there is no $d \in Z$ such that $\theta_2, d \models \psi$ while $\theta_1, 5 \models \psi$. □

Similarly, the membership and emptiness lose decidability for GRCFG even if a binary relation appearing in a guard expression is a decidable relation.

Theorem 4.3: There exists a data structure $(D, \mathcal{R})$ such that the membership and emptiness problems for 1-GRCFG($\mathcal{R}$) are undecidable.

Proof We show a reduction from the Post's correspondence problem (PCP). Let $\{(u_1, v_1), \ldots, (u_n, v_n)\} \subseteq \Sigma^+ \times \Sigma^+$ be an instance of PCP over an alphabet Σ. Then let $D = \Sigma^* \times \Sigma^*$ and $\mathcal{R} = \{\bowtie_i \mid i \in [n]\} \cup \{EQ\}$ where

$$(x_1, x_2) \bowtie_i (y_1, y_2) \iff x_1 u_i = y_1 \wedge x_2 v_i = y_2,$$
$$(x_1, x_2)\ EQ\ (y_1, y_2) \iff x_1 = x_2.$$

We assume the initial register data value $\bot = (\varepsilon, \varepsilon) \in D$. We construct 1-GRCFG($\mathcal{R}$) $G = (V, R, S)$ where R consists of

$(S, x_1^{\bowtie_i}, 1) \to A$ and $(A, x_1^{\bowtie_i}, 1) \to A$ for $\forall i \in [n]$ and $(A, x_1^{EQ}) \to \varepsilon$.

We can easily show that $\varepsilon \in L(G)$ (iff $L(G) \neq \emptyset$) iff the instance $\{(u_1, v_1), \ldots, (u_n, v_n)\}$ of PCP has a solution. □

In the rest of this paper, we assume $\mathcal{R}$ is a singleton $\mathcal{R} = \{\bowtie\}$ for simplicity. The properties we show below can be extended in a general case that $\mathcal{R}$ has more than one binary relation. We first extend a register type as a binary relation $\gamma : ([k] \times [k]) \backslash \{(i, i) \mid i \in [k]\} \to \{\texttt{tt}, \texttt{ff}\}$[†]. We say that the type of an assignment θ is γ (and write $\theta \models \gamma$) iff for all $i, j \in [k]$ $(i \neq j)$,

$$\gamma(i, j) = \texttt{tt} \text{ iff } \theta(i) \bowtie \theta(j).$$

We write $\theta \sim_{\bowtie} \theta'$ if the types of assignments θ and θ' are the same. The collection of all register types of k-GRCFG is denoted by Γ_k as before.

In Example 4.1, we see that a register type cannot always decide the applicability of a rule because there may exist $\gamma \in \Gamma_k, \psi \in F_k$ such that $\exists d.\ \theta, d \models \psi$ and $\forall d'.\ \theta', d' \not\models \psi$ for some assignments $\theta, \theta' \models \gamma$. We would like to avoid the above case and also to uniquely determine the register-type $\gamma' \in \Gamma_k$ after a rule application. For this purpose, we introduce two predicates (Definition 4.1) to describe the desirable property (Definition 4.2).

Definition 4.1: Let $pr, npr : \Gamma_k \times F_k \times [k] \times \Gamma_k \to \{\texttt{tt}, \texttt{ff}\}$ be the predicates defined as follows: For $\gamma, \gamma' \in \Gamma_k$, $i \in [k]$, and $\psi \in F_k$,

$$pr(\gamma, \psi, i, \gamma') = \texttt{tt} \iff \gamma' \in after(\theta, \psi, i) \text{ for } \forall\theta \models \gamma,$$
$$npr(\gamma, \psi, i, \gamma') = \texttt{tt} \iff \gamma' \notin after(\theta, \psi, i) \text{ for } \forall\theta \models \gamma,$$

where $after(\theta, \psi, i) = \{\gamma' \mid \exists d.\ (\theta, d \models \psi \wedge \theta[i \leftarrow d] \models \gamma')\}$.

Definition 4.2 (simulation): We say that a data structure $(D, \mathcal{R})$ has the simulation property for k registers if the following condition is met:

For all $\gamma, \gamma' \in \Gamma_k$, $i \in [k]$, and $\psi \in F_k$, either $pr(\gamma, \psi, i, \gamma') = \texttt{tt}$ or $npr(\gamma, \psi, i, \gamma') = \texttt{tt}$ holds.

□

Again, both pr and npr are undecidable in general because a binary relation appearing in ψ may be undecidable[††]. Hence, we will introduce the assumption that both pr and npr are decidable (in EXPTIME).

Definition 4.3 (progress): We say that a data structure $(D, \mathcal{R})$ has the progress property for k registers if the following condition is met:

For all $\gamma, \gamma' \in \Gamma_k$, $i \in [k]$, and $\psi \in F_k$, predicate pr and npr are decidable in EXPTIME.

□

For a data structure $\mathbb{D} = (D, \mathcal{R})$, if D is understood, we say that $\mathcal{R}$ (or even $\bowtie$, if $\mathcal{R} = \{\bowtie\}$) has the simulation or progress property.

Finally, we define *data type* as an extension of register type by adding the information on equality between data

[†]We exclude the diagonal elements $\{(i, i) \mid i \in [k]\}$ from the domain of a register type because the applicability of a rule does not depend on whether $\theta(i) \bowtie \theta(i)$.

[††]It is unknown whether pr and npr are decidable when we restrict a binary relation in ψ to be decidable.

values in the registers and data values appearing in a given data word w.

Definition 4.4 (data type): Let w be a data word and D_w be the set of data values appearing in w; i.e. $D_w = \{d_i \mid i \in [n],\ w = (a_1, d_1) \dots (a_n, d_n)\}$. Also let $d_{\neq} \notin D$ be a newly introduced symbol. We use a function $e : [k] \to D_w \cup \{d_{\neq}\}$ as a finite representation of an assignment. This function is the same as a given assignment except that every data value that does not appear in w is replaced with $d_{\neq}$. We write $\theta \models e$ iff for all $i \in [k]$,

$$e(i) = \theta(i) \text{ if } \theta(i) \in D_w \text{ and } e(i) = d_{\neq} \text{ otherwise.}$$

The collection of all such functions $e : [k] \to D_w \cup \{d_{\neq}\}$ is denoted by $E_{w,k}$. The *data type* of an assignment $\theta \in \Theta_k$ for a data word w is a pair $(\gamma, e) \in \Gamma_k \times E_{w,k}$. We write $\theta \models (\gamma, e)$ iff $\theta \models \gamma$ and $\theta \models e$. We define the simulation and progress properties with data type in the same way as in the case of register types.

5. Properties of GRCFG

5.1 ε-Rule Removal

Theorem 5.1: For an arbitrary GRCFG($\bowtie$) G such that $\bowtie$ has the simulation and progress properties, we can construct an ε-rule free GRCFG($\bowtie$) G'' that satisfies $L(G'') = L(G) \setminus \{\varepsilon\}$.

Proof The theorem can be proved in a similar way to Theorem 3.1 by using the simulation and progress properties. Let $G = (V, R, S)$ be a k-GRCFG($\bowtie$). Without loss of generality, we assume that R has no rule of the form $(A, \psi) \to \alpha$ (by Theorem 4.2) and the guard expression of every rule in R is a conjunction of literals (atomic formulas or their negations). We first construct k-GRCFG($\bowtie$) $G' = (V', R', S')$ from G where

- $V' = V \times \Gamma_k$,
- R' is the smallest set of rules satisfying the following conditions. Define the subset of guard expressions Φ as

  $$\Phi = \{\bigwedge_{i\in[k]} \zeta_i \wedge \bigwedge_{i\in[k]} \eta_i \mid \zeta_i \in \{x_i^{\bowtie}, \neg x_i^{\bowtie}\}, \eta_i \in \{x_i^{\bowtie^{-1}}, \neg x_i^{\bowtie^{-1}}\}\}.$$

  Let $r = (A, \psi, i) \to \alpha \in R$. Also let $\gamma, \gamma' \in \Gamma_k$ and $\varphi \in \Phi$. If $pr(\gamma, \psi \wedge \varphi, i, \gamma') = \mathtt{tt}$, which is decidable by the progress property,

  $$((A, \gamma), \psi \wedge \varphi, i) \to \alpha^{\mathrm{aug}(\gamma')} \in R'.$$

  (See the proof of Lemma 3.1 for the definition of $\alpha^{\mathrm{aug}(\gamma')}$.) Note that γ' is uniquely determined by $\gamma, \psi \wedge \varphi$ and i because γ specifies whether $\theta(i) \bowtie \theta(j)$ holds or not for each pair $i, j \in [k]$ $(i \neq j)$ and also φ specifies whether $\theta(i) \bowtie d$ and $d \bowtie \theta(i)$ hold or not for each $i \in [k]$ and an input data value d.
- $S' = (S, \gamma_0)$ where $\theta_\perp \models \gamma_0$.

See the example of the construction in Example 5.1. We will show $L(G) = L(G')$ by induction on the length of derivations in G and G', using the simulation property (to show $L(G) \subseteq L(G')$).

First we show that $L(G') \subseteq L(G)$. Assume $w \in L(G')$. Then, there exists a derivation $((S, \gamma_0), \theta_\perp) \overset{*}{\Rightarrow}_{G'} w$. We have to show $(S, \theta_\perp) \overset{*}{\Rightarrow}_G w$. For this purpose, we will show by induction on the length of the derivation in G' a little more general property that if $((S, \gamma_0), \theta_\perp) \overset{*}{\Rightarrow}_{G'} Y'_1 \dots Y'_m$ $(Y'_j \in ((V \times \Gamma_k) \times \Theta_k) \cup (\Sigma \times D)$ for each $j \in [m])$, then $(S, \theta_\perp) \overset{*}{\Rightarrow}_G Y_1 \dots Y_m$ where $Y_j = (B_j, \theta_j)$ and $\theta_j \models \gamma_j$ if $Y'_j = ((B_j, \gamma_j), \theta_j)$ and $Y_j = Y'_j$ otherwise. We call the former derivation t'.

(Basis) If the length of t' is zero, the claim holds because $\theta_\perp \models \gamma_0$.

(Induction) Assume that the length of t' is greater than zero and the last step in t' is $Y'_1 \dots Y'_{p-1}((A, \gamma), \theta) Y'_{q+1} \dots Y'_m \Rightarrow_{G'} Y'_1 \dots Y'_m$ with $((A, \gamma), \theta) \Rightarrow^d_{G', r'} Y'_p \dots Y'_q$ for some $1 \leq p, q \leq m$. Since $r' \in R'$, by the construction of R', there are $r = (A, \psi, i) \to \alpha \in R$, $\varphi \in \Phi$ and $\gamma' \in \Gamma_k$ such that r' is constructed from r, γ, γ' and φ. We assume that the form of r is $(A, \psi, i) \to \alpha$. By the induction hypothesis, $(S, \theta_\perp) \overset{*}{\Rightarrow}_G Y_1 \dots Y_{p-1}(A, \theta) Y_{q+1} \dots Y_m$ where $Y_j = (B_j, \theta_j)$ and $\theta_j \models \gamma_j$ if $Y'_j = ((B_j, \gamma_j), \theta_j)$ and $Y_j = Y'_j$ otherwise for $1 \leq j < p$ or $q < j \leq n$, and also $\theta \models \gamma$. Since the left-hand side of r' is $((A, \gamma), \psi \wedge \varphi, i)$ and r' is applied to $((A, \gamma), \theta)$ with the input data d, we have $\theta, d \models \psi \wedge \varphi$. Hence $\theta, d \models \psi$ holds obviously and thus r is applicable to (A, θ) also. Let $(A, \theta) \Rightarrow^d_{G,r} \alpha'$, then we obtain a desired derivation $(S, \theta_\perp) \overset{*}{\Rightarrow}_G Y_1 \dots Y_{p-1}(A, \theta) Y_{q+1} \dots Y_m \Rightarrow^d_{G,r} Y_1 \dots Y_{p-1} \alpha' Y_{q+1} \dots Y_m$.

Next we show that $L(G) \subseteq L(G')$. Assume $w \in L(G)$. Then, there exists a derivation $S \overset{*}{\Rightarrow}_G w$. We have to show $((S, \gamma_0), \theta_\perp) \overset{*}{\Rightarrow}_{G'} w$. We will show by induction on the length of the derivation in G that if $(S, \theta_\perp) \overset{*}{\Rightarrow}_G Y_1 \dots Y_m$ $(Y_j \in (V \times \Theta_k) \cup (\Sigma \times D)$ for each $j \in [m])$ then $((S, \gamma_0), \theta_\perp) \overset{*}{\Rightarrow}_{G'} Y'_1 \dots Y'_m$ where $Y'_j = ((B_j, \gamma_j), \theta_j)$ for γ_j such that $\theta_j \models \gamma_j$ if $Y_j = (B_j, \theta_j)$ and $Y'_j = Y_j$ otherwise. We call the former derivation t.

(Basis) If the length of t is zero, the claim holds because $\theta_\perp \models \gamma_0$.

(Induction) Assume that the length of t is greater than zero and the last step in t is $Y_1 \dots Y_{p-1}(A, \theta) Y_{q+1} \dots Y_m \Rightarrow_G Y_1 \dots Y_m$ with $(A, \theta) \Rightarrow^d_{G,r} Y_p \dots Y_q$ for some $1 \leq p, q \leq m$. By the induction hypothesis, $((S, \gamma_0), \theta_\perp) \overset{*}{\Rightarrow}_{G'} Y'_1 \dots Y'_{p-1}((A, \gamma), \theta) Y'_{q+1} \dots Y'_m$ where $Y'_j = ((B_j, \gamma_j), \theta_j)$ for γ_j such that $\theta_j \models \gamma_j$ if $Y_j = (B_j, \theta_j)$ and and $Y'_j = Y_j$ otherwise for $1 \leq j < p$ or $q < j \leq n$, and also $\theta \models \gamma$. We assume that the form of r applied to (A, θ) is $(A, \psi, i) \to \alpha$. This means that $\theta, d \models \psi$, and because of the definition of Φ, there must be a unique $\varphi \in \Phi$ such that $\theta, d \models \psi \wedge \varphi$. Hence, $\gamma' \in \mathit{after}(\theta, \psi \wedge \varphi, i)$ for the type γ' of $\theta[i \leftarrow d]$.

By the simulation property, we have $pr(\gamma, \psi \wedge \varphi, i, \gamma') = \mathtt{tt}$. Therefore, the rule $r' = ((A, \gamma), \psi \wedge \varphi, i) \rightarrow \alpha^{\mathrm{aug}(\gamma')}$ has been constructed. The rest of the proof is similar to the converse direction.

We can construct an equivalent ε-rule free k-GRCFG($\bowtie$) G'' from G' in a similar way to Theorem 3.1 (because every application of a rule in G' is never blocked by the guard condition and we can compute the set Nu of nullable nonterminals like Theorem 3.1).

Example 5.1: Let $k = 2$ and consider a rule $r = (A, \psi, 1) \rightarrow \alpha$ where $\psi = x_1^{\bowtie} \wedge x_2^{\bowtie^{-1}}$. The possible register types are $\gamma_1 = (\delta_{12} \wedge \delta_{21})$, $\gamma_2 = (\delta_{12} \wedge \neg\delta_{21})$, $\gamma_3 = (\neg\delta_{12} \wedge \delta_{21})$ and $\gamma_4 = (\neg\delta_{12} \wedge \neg\delta_{21})$ where $\delta_{12} = (\theta(1) \bowtie \theta(2))$ and $\delta_{21} = (\theta(2) \bowtie \theta(1))$[†]. After the elimination of the unsatisfiable ones and Boolean simplification, we can assume that $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ where $\varphi_1 = x_1^{\bowtie^{-1}} \wedge x_2^{\bowtie}$, $\varphi_2 = x_1^{\bowtie^{-1}} \wedge \neg x_2^{\bowtie}$, $\varphi_3 = \neg x_1^{\bowtie^{-1}} \wedge x_2^{\bowtie}$ and $\varphi_4 = \neg x_1^{\bowtie^{-1}} \wedge \neg x_2^{\bowtie}$. If $pr(\gamma, \psi \wedge \varphi_i, 1, \gamma') = \mathtt{tt}$, the possible register type γ' is $\gamma_1, \gamma_2, \gamma_1, \gamma_2$ for $\varphi_1, \varphi_2, \varphi_3, \varphi_4$, respectively. In this example, the type γ' is determined depending only on φ_j and independent of γ because $k = 2$ and an input data value is loaded to the first register when r is applied.

5.2 Emptiness and Membership

Theorem 5.2: The emptiness problem for GRCFG($\bowtie$) such that $\bowtie$ has the simulation and progress properties, is EXPTIME-complete.

Proof Let $G = (V, R, S)$ be an arbitrary k-GRCFG($\bowtie$) that has no rule of the form $(A, \psi) \rightarrow \alpha$ and $G' = (V', R', S')$ be the k-GRCFG($\bowtie$) constructed from G in the proof of Theorem 5.1. As shown in that proof, $L(G') = L(G)$. We construct CFG $G'' = (V', R'', S')$ from G' where

$R'' = \{(A, \gamma) \rightarrow X_1 \ldots X_n \mid ((A, \gamma), \psi, i) \rightarrow X'_1 \ldots X'_n \in R'$ for some ψ and i, and $X_j = X'_j$ if $X'_j \in V'$ and $X_j = a$ if $X'_j \notin V'$ for each $j \in [n]\}$.

As we will show below, $L(G') = \emptyset \Leftrightarrow L(G'') = \emptyset$ because a rule application is never blocked in G'.

Assume $L(G'') \neq \emptyset$. Then, there exists a derivation $S' \overset{*}{\Rightarrow}_{G''} w$ for some $w \in a^*$. It suffices to show $(S', \theta_\perp) \overset{*}{\Rightarrow}_{G'} w'$ for some $w' \in (\Sigma \times D)^*$. For this purpose, we will show by induction on the length of the derivation in G'' that if $t = S' \overset{*}{\Rightarrow}_{G''} Y_1 \ldots Y_m$ ($Y_j \in V'$ or $Y_j = a$ for each $j \in [m]$), then $(S', \theta_\perp) \overset{*}{\Rightarrow}_{G'} Y'_1 \ldots Y'_m$ where $Y'_j = (Y_j, \theta_j)$ for some $\theta_j \in \Theta_k$ such that $\theta_j \models \gamma_j$ if $Y_j = (B_j, \gamma_j) \in V'$ and $Y'_j \in (\Sigma \times D)$ if $Y_j = a$.
(Basis) If the length of t is zero, the claim trivially holds.
(Induction) Assume that the length of t is greater than zero and the last step in t is $Y_1 \ldots Y_{p-1}(A, \gamma)Y_{q+1} \ldots Y_m \Rightarrow_{G'',r} Y_1 \ldots Y_m$ with $(A, \gamma) \Rightarrow_{G'',r} Y_p \ldots Y_q$ for some $1 \leq p, q \leq m$. Since $r \in R''$, by the construction of R', there are $r' \in R'$ and $\gamma' \in \Gamma_k$ such that $r' = ((A, \gamma), \psi, i) \rightarrow \alpha^{\mathrm{aug}(\gamma')} \in R'$, $pr(\gamma, \psi, i, \gamma') = \mathtt{tt}$, and r is constructed from r'. We assume that the form of r' is $((A, \gamma), \psi, i) \rightarrow Y'_p \ldots Y'_q$. By the induction hypothesis, $(S', \theta_\perp) \overset{*}{\Rightarrow}_{G'} Y'_1 \ldots Y'_{p-1}((A, \gamma), \theta)Y'_{q+1} \ldots Y'_m$ for some $\theta \models \gamma$ and $Y'_j = (Y_j, \theta_j)$ and $\theta_j \models \gamma_j$ if $Y_j = (B_j, \gamma_j) \in V'$ and $Y'_j \in (\Sigma \times D)$ if $Y_j = a$ for $1 \leq j < p$ or $q < j \leq n$. Since $pr(\gamma, \psi, i, \gamma') = \mathtt{tt}$, the rule r' can be applied at $((A, \gamma), \theta)$ in the above derivation in G'. Hence, $L(G') \neq \emptyset$.

The converse direction $L(G') \neq \emptyset \Rightarrow L(G'') \neq \emptyset$ can be proved in a similar way.

$\|G'\|$ is single exponential to $\|G\|$ because $|V'| = |V| \times |\Gamma_k|$ (by the definition of $\Gamma_k : ([k] \times [k]) \backslash \{(i, i) \mid i \in [k]\} \rightarrow \{\mathtt{tt}, \mathtt{ff}\}$, $|\Gamma_k| \leq 2^{k^2}$) and $|R'| \leq |R| \times |\Gamma_k|^2 \times |\Phi|$ (Φ is defined in the proof of Theorem 5.1 and $|\Phi| = 2^{2k}$). In addition, $\|G''\|$ is obviously linear to $\|G'\|$. Therefore, $\|G''\|$ is single exponential to $\|G\|$. Because the emptiness problem for CFG is decidable in linear time, the emptiness problem for GRCFG is decidable in deterministic time exponential to k.

The lower bound can be obtained from EXPTIME-completeness of the emptiness problem for k-GRCFG(=) [16].

Theorem 5.3: The membership problem for GRCFG($\bowtie$) such that $\bowtie$ has the the simulation and progress properties with data type, is EXPTIME-complete.

Proof sketch This theorem can be proved in a similar way to Theorem 5.2 as follows. From a given k-GRCFG($\bowtie$) $G = (V, R, S)$ and a data word w, we construct k-GRCFG($\bowtie$) $G' = (V', R', S')$ in a similar way to Theorem 5.1, by using the set of data types $\Gamma_k \times E_{w,k}$ instead of Γ_k (Therefore, $V' = V \times \Gamma_k \times E_{w,k}$). We construct CFG $G'' = (V', R'', S')$ (over a finite alphabet $\Sigma \times (D_w \cup \{d_{\neq}\})$) from G' where

$R'' = \{(A, (\gamma, e)) \rightarrow X_1 \ldots X_n \mid ((A, (\gamma, e)), \psi, i) \rightarrow X'_1 \ldots X'_n \in R'$ for some ψ and i, and $X_j = X'_j$ if $X'_j \in V'$ and $X_j = (a, e(h))$ if $X'_j = (a, h) \in (\Sigma \times [k])$ for each $j \in [n]\}$.

Similar to the proof of Theorem 5.2, each derivation $S' \overset{*}{\Rightarrow}_{G''} Y_1 \ldots Y_m$ in G'' has a corresponding derivation $(S', \theta_\perp) \overset{*}{\Rightarrow}_{G'} Y'_1 \ldots Y'_m$ in G'. Moreover, if Y_j is a terminal $(a, d) \in \Sigma \times (D_w \cup \{d_{\neq}\})$, then Y'_j equals (a, d) if $d \in D_w$ and $Y'_j = (a, d')$ for some $d' \notin D_w$ if $d = d_{\neq}$. Therefore, we can prove $w \in L(G') \Leftrightarrow w \in L(G'')$ in a similar way to Theorem 5.2. □

If we drop the progress property, the membership and emptiness are not always decidable as shown in the next theorem.

Theorem 5.4: There exists a data structure $(D, \mathcal{R})$ that has the simulation property but does not have the progress property, and the membership and emptiness problems for 2-GRCFG($\mathcal{R}$) are undecidable.

Proof We use a reduction from the inclusion problem for CFG. Let $D = \Sigma^*$, and consider two CFG A and B over Σ and $\mathcal{R} = \{\bowtie\}$ such that

[†]For readability, we denote a register type as a Boolean formula on a register assignment θ. For example, $\gamma_2(1, 2) = \mathtt{tt}$ and $\gamma_2(2, 1) = \mathtt{ff}$ if we follow the notation defined in Sect. 4.2.

$$d_1 \bowtie d_2 \iff d_2 \in L(A) \setminus L(B).$$

As defined in Example 5.1, there are four possible register types $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ for $k = 2$ and a single binary relation $\bowtie$. For convenience, we give the definition of γ_3 and γ_4 again.

- $\gamma_3 = (\neg(\theta(1) \bowtie \theta(2)) \wedge \theta(2) \bowtie \theta(1))$
- $\gamma_4 = (\neg(\theta(1) \bowtie \theta(2)) \wedge \neg(\theta(2) \bowtie \theta(1)))$

For the binary relation $\bowtie$ defined above, $(\theta(1) \bowtie \theta(2))$ iff $(\theta(2) \in L(A)\setminus L(B))$ and $(\theta(2) \bowtie \theta(1))$ iff $(\theta(1) \in L(A)\setminus L(B))$.

We can prove that $\bowtie$ has the simulation property but does not have the progress property for two registers.

(The simulation property) For given $\psi \in F_2$, $\gamma' \in \Gamma_2$, $i \in [2]$, $\theta \in \Theta_2$, and $d \in D$, the condition $\theta, d \models \psi \wedge \theta[i \leftarrow d] \models \gamma'$ can be transformed into a Boolean combination of the following atomic propositions:

$$\theta(1) \in L(A) \setminus L(B), \tag{4}$$
$$\theta(2) \in L(A) \setminus L(B), \tag{5}$$
$$d \in L(A) \setminus L(B). \tag{6}$$

(Note that the atomic propositions of F_2 are $x_1^{\bowtie}$, $x_2^{\bowtie}$, $x_1^{\bowtie^{-1}}$, $x_2^{\bowtie^{-1}}$, and for example, $(\theta, d \models x_1^{\bowtie})$ iff $(d \in L(A) \setminus L(B))$ and $(\theta, d \models x_1^{\bowtie^{-1}})$ iff $(\theta(1) \in L(A) \setminus L(B))$.)

Whether the propositions (4) and (5) hold is determined uniquely by the type γ of θ (because γ uniquely determines whether $\theta(2) \bowtie \theta(1)$ and $\theta(1) \bowtie \theta(2)$ hold). Therefore, whether $\exists d.\ \theta, d \models \psi \wedge \theta[i \leftarrow d] \models \gamma'$ holds or not is determined independent of a choice of θ of type γ, and thus the simulation property holds.

(Absence of the progress property) We can show that $pr(\gamma_4, x_1^{\bowtie}, 1, \gamma_3)$ for the above-mentioned γ_4 and γ_3 is undecidable by a reduction from the inclusion problem $L(A) \subseteq L(B)$ for CFG A and B. We first decide whether $L(B) = \emptyset$. If $L(B) = \emptyset$, then $L(A) \subseteq L(B)$ iff $L(A) = \emptyset$, which is also decidable. Assume that $L(B) \neq \emptyset$. We can prove $L(A) \nsubseteq L(B) \Leftrightarrow pr(\gamma_4, x_1^{\bowtie}, 1, \gamma_3) = \mathtt{tt}$ in the following steps:

$$\begin{aligned} L(A) \nsubseteq L(B) &\Leftrightarrow \exists d.\ d \in L(A) \setminus L(B) \\ &\Leftrightarrow \forall \theta \models \gamma_4.\ \exists d.\ \theta, d \models x_1^{\bowtie} \wedge \theta[1 \leftarrow d] \models \gamma_3 \\ &\Leftrightarrow pr(\gamma_4, x_1^{\bowtie}, 1, \gamma_3) = \mathtt{tt}. \end{aligned}$$

$L(A) \nsubseteq L(B) \Leftrightarrow \exists d.\ d \in L(A) \setminus L(B)$ is obviously satisfied, and $\forall \theta \models \gamma_4.\ \exists d.\ \theta, d \models x_1^{\bowtie} \wedge \theta[1 \leftarrow d] \models \gamma_3$ $\Leftrightarrow pr(\gamma_4, x_1^{\bowtie}, 1, \gamma_3) = \mathtt{tt}$ is obtained by the definition of pr. We prove the middle step $\exists d.\ d \in L(A) \setminus L(B) \Leftrightarrow \forall \theta \models \gamma_4.\ \exists d.\ \theta, d \models x_1^{\bowtie} \wedge \theta[1 \leftarrow d] \models \gamma_3$ as follows.

- ($\Rightarrow$): Let θ be an arbitrary assignment that satisfies $\theta \models \gamma_4$. Note that $\theta(2) \notin L(A) \setminus L(B)$ by the definition of γ_4. By the assumption, there is a data value $d \in L(A) \setminus L(B)$. This d satisfies $\theta, d \models x_1^{\bowtie}$. Moreover, this d also satisfies $\theta[1 \leftarrow d] \models \gamma_3$ because $\theta[1 \leftarrow d](1) = d \in L(A) \setminus L(B)$ and $\theta[1 \leftarrow d](2) = \theta(2) \notin L(A) \setminus L(B)$.
- ($\Leftarrow$): Because $L(B) \neq \emptyset$, there exists some $d' \in L(B)$. Let θ be the assignment defined as $\theta(1) = \theta(2) = d'$, which satisfies $\theta \models \gamma_4$. By the assumption, there exists d such that $\theta, d \models x_1^{\bowtie}$, and this implies that $d \in L(A) \setminus L(B)$.

We construct 2-GRCFG($\bowtie$) $G = (V, R, S)$ such that $R = \{(S, x_1^{\bowtie}) \to \varepsilon\}$. We can easily show that $\varepsilon \in L(G)$ (iff $L(G) \neq \emptyset$) iff $L(A) \nsubseteq L(B)$.

5.3 GRCFG with a Total Order on a Dense Set

Lemma 5.1: Every GRCFG($<_Q$) has the simulation and progress properties where $<_Q$ is the strict total order on the set Q of all rational numbers. Similarly, it has the simulation and progress properties with data type.

Proof We abbreviate $<_Q$ as $<$. Let $G = (V, R, S)$ be a k-GRCFG($<$), $\theta \in \Theta_k$, $\gamma \in \Gamma_k$ and $r = (A, \psi, i) \to \alpha \in R$ where ψ is the conjunction of literals (of the form $x_i^<$ or $\neg x_j^<$). (The case $r = (A, \psi) \to \alpha \in R$ can be treated in a similar way.) Assume that $\theta \models \gamma$. The rule r can be applied to (A, θ) iff there is $d \in Q$ such that $\theta, d \models \psi$. The condition $\theta, d \models \psi$ as well as the assumption $\theta \models \gamma$ can be represented as a set of inequations on $d, \theta(1), \ldots, \theta(k)$. Whether this set of inequations has a contradiction does not depend on the concrete values $\theta(1), \ldots, \theta(k)$, and if it does not have a contradiction, then there must exist $d \in Q$ that satisfies it because Q is dense. Moreover, whether $\theta[i \leftarrow d] \models \gamma'$ holds for a given γ', which can also be represented as the consistency of a set of inequations on $d, \theta(1), \ldots, \theta(k)$, does not depend on θ. Hence, for all $\theta \models \gamma$, whether there exists d such that $\theta, d \models \psi$ and $\theta[i \leftarrow d] \models \gamma'$ can be decided uniquely, and the simulation property holds.

Similarly, for deciding $pr(\gamma, \psi, i, \gamma') = \mathtt{tt}$, it suffices to represent the condition

$$\theta \models \gamma \ \wedge \ \theta, d \models \psi \ \wedge \ \theta[i \leftarrow d] \models \gamma'$$

as a set of inequations on $d, \theta(1), \ldots, \theta(k)$ as above and solve it.

We can show the simulation and progress properties with data type in a similar way.

Example 5.2: Consider a 2-GRCFG($<$) (V, R, S) and a rule $(A, \psi, 1) \to B \in R$ where $\psi = x_1^< \wedge \neg x_2^<$. We see that $\theta, d \models \psi \Leftrightarrow \theta(1) < d \leq \theta(2)$. Because $k = 2$ and $<$ is a total order on Q, there are three possible register types $\gamma_1 = (\theta(1) < \theta(2))$, $\gamma_2 = (\theta(2) < \theta(1))$ and $\gamma_3 = (\theta(1) = \theta(2))$. As easily known, (i) there is $d \in Q$ such that $\theta, d \models \psi$ and $\theta \models \gamma$ if and only if $\gamma = \gamma_1$, and (ii) if $\gamma = \gamma_1$ then such $d \in Q$ satisfies either (ii-a) $d < \theta(2)$, $\theta[1 \leftarrow d] \models \gamma_1$ or (ii-b) $d = \theta(2)$, $\theta[1 \leftarrow d] \models \gamma_3$.

Corollary 5.1: For a given GRCFG($<_Q$) G, we can construct an ε-rule free GRCFG($<_Q$) G' that satisfies $L(G') = L(G) \setminus \{\varepsilon\}$. The emptiness and membership problems are both EXPTIME-complete for GRCFG($<_Q$).

Proof By Lemma 5.1 and Theorems 5.1, 5.2 and 5.3.

6. Conclusion

We have introduced register type to RCFG and shown an equivalence transformation to RCFG that never blocks a rule application by associating a register type with each nonterminal symbol. Then we have defined generalized RCFG (GRCFG) that can use an arbitrary relation in the guard expression. Using the technique of register type and making two reasonable assumptions, the simulation and progress properties, the decidability of emptiness and membership for GRCFG and a transformation to an ε-free GRCFG have been provided.

Nominal CFG [3] with equality symmetry, total order symmetry and integer symmetry correspond to GRCFG(=), GRCFG($<_Q$) (Sect. 5.3) and GRCFG($<_Z$) (Example 4.1), respectively. Investigating the relation between nominal CFG and GRCFG in depth is future work.

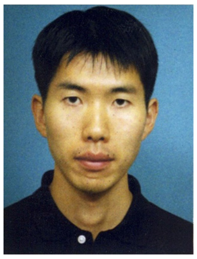
Acknowledgments

The authors thank the reviewers for providing valuable comments to the paper. This work was supported by JSPS KAKENHI Grant Number JP19H04083.

References

[1] M. Benedikt, C. Ley, and G. Puppis, "What you must remember when processing data words," 4th Alberto Mendelzon International Workshop on Foundations of Data Management, 2010.
[2] M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin, "Two-variable logic on data words," ACM Trans. Comput. Logic, vol.12, no.4, pp.1–26, July 2011.
[3] M. Bojańczyk, B. Klin, and S. Lasota, "Automata theory in nominal sets," Log. Meth. Comput. Sci., vol.10, no.3, Aug. 2014.
[4] P. Bouyer, "A logical characterization of data languages," Inform. Process. Lett., vol.84, no.2, pp.75–85, Oct. 2002.
[5] E.Y.C. Cheng and M. Kaminski, "Context-free languages over infinite alphabets," Acta Inform., vol.35, no.3, pp.245–267, March 1998.
[6] S. Demri and R. Lazić, "LTL with the freeze quantifier and register automata," ACM Trans. Comput. Logic, vol.10, no.3, pp.1–30, April 2009.
[7] S. Demri, R. Lazić, and D. Nowak, "On the freeze quantifier in constraint LTL: Decidability and complexity," Inform. Comput., vol.205, no.1, pp.2–24, Jan. 2007.
[8] D. Figueira, P. Hofman, and S. Lasota, "Relating timed and register automata," Math. Struct. Comp. Sci., vol.26, no.6, pp.993–1021, Sept. 2016.
[9] M. Kaminski and N. Francez, "Finite-memory automata," Theor. Comput. Sci., vol.134, no.2, pp.329–363, 1994.
[10] L. Libkin, W. Martens, and D. Vrgoč, "Querying graphs with data," J. ACM, vol.63, no.2, pp.1–53, March 2016.
[11] L. Libkin, T. Tan, and D. Vrgoč, "Regular expressions for data words," J. Comput. Syst. Sci., vol.81, no.7, pp.1278–1297, 2015.
[12] L. Libkin and D. Vrgoč, "Regular path queries on graphs with data," Proc. 15th International Conference on Database Theory (ICDT '12), pp.74–85, 2012.
[13] T. Milo, D. Suciu, and V. Vianu, "Typechecking for XML transformers," Proc. 19th ACM Symposium on Principles of Database Systems (PODS 2000), pp.11–22, 2000.
[14] F. Neven, T. Schwentick, and V. Vianu, "Finite state machines for strings over infinite alphabets," ACM Trans. Comput. Logic, vol.5, no.3, pp.403–435, July 2004.
[15] H. Sakamoto and D. Ikeda, "Intractability of decision problems for finite-memory automata," Theor. Comput. Sci., vol.231, no.2, pp.297–308, 2000.
[16] R. Senda, Y. Takata, and H. Seki, "Complexity results on register context-free grammars and register tree automata," Theoretical Aspects of Computing – ICTAC 2018, Lecture Notes in Computer Science, vol.11187, pp.415–434, Springer International Publishing, Cham, 2018.
[17] R. Senda, Y. Takata, and H. Seki, "Generalized register context-free grammars," Language and Automata Theory and Applications, Lecture Notes in Computer Science, vol.11417, pp.259–271, Springer International Publishing, Cham, 2019.

Ryoma Senda received his B.E. degree from Nagoya University in 2018. He has pursued Master's degree in the university. His research interests include formal language theory and its application to software engineering.



Yoshiaki Takata received the Ph.D. degree in information and computer science from Osaka University in 1997. He was with Nara Institute of Science and Technology as an Assistant Professor in 1997–2007. In 2007, he joined the faculty of Kochi University of Technology, where he has been an Associate Professor since 2009. His current research interests include formal specification and verification of software systems.

Hiroyuki Seki received his Ph.D. degree from Osaka University in 1987. He was an Assistant Professor, and later, an Associate Professor in Osaka University from 1987 to 1994. In 1994, he joined Nara Institute of Science and Technology, where he was a Professor during 1996 to 2013. Currently, he is a Professor in Nagoya University. His current research interests include formal language theory and formal approach to software development.