Generalized Register Context-Free Grammars

Ryoma Senda[1(✉)], Yoshiaki Takata[2], and Hiroyuki Seki[1]

[1] Graduate School of Information Science, Nagoya University, Furo-cho, Chikusa, Nagoya 464-8601, Japan
ryoma.private@sqlab.jp, seki@i.nagoya-u.ac.jp
[2] Graduate School of Engineering, Kochi University of Technology, Tosayamada, Kami City, Kochi 782-8502, Japan
takata.yoshiaki@kochi-tech.ac.jp

Abstract. Register context-free grammars (RCFG) is an extension of context-free grammars to handle data values in a restricted way. This paper first introduces register type as a finite representation of the register contents and shows some properties of RCFG. Next, generalized RCFG (GRCFG) is defined by permitting an arbitrary relation on data values in the guard expression of a production rule. We extend register type to GRCFG and introduce two properties of GRCFG, the simulation property and the type oracle. We then show that ε-rule removal is possible and the emptiness and membership problems are EXPTIME solvable for GRCFG that satisfy these two properties.

1 Introduction

This paper focuses on register context-free grammars (abbreviated as RCFG), which were introduced by Cheng and Kaminsky in 1998 [6]. Recently, register automata (abbreviated as RA) [10] have been paid attention [11–13] 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 [6] 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 a *register 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 [6], properties of RCFG were

© Springer Nature Switzerland AG 2019
C. Martín-Vide et al. (Eds.): LATA 2019, LNCS 11417, pp. 259–271, 2019.
https://doi.org/10.1007/978-3-030-13435-8_19

shown including the decidability of the membership and emptiness problems, and the closure properties. In our previous study [17], 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. 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 conditions, namely, the simulation and the type oracle. We show that the emptiness and membership are decidable and ε-removal is possible for GRCFG that satisfies these two conditions. As a corollary, we also show that those properties hold for GRCFG with a total order on a dense set.

Related Work. Register automata (RA) was proposed in [10] 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 former two problems are analyzed in [7,16]. In [6], register context-free grammars (RCFG) as well as pushdown automata over an infinite alphabet 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 [6]. Extension of RA to a totally ordered set was discussed in [1,9], 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 [5], pebble automata (PA) [14] and nominal automata (NA) [4] are known. Libkin and Vrgoč [13] 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 [15]. 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 [4]. (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 [4], nominal CFG is also introduced but decidability of related problems is not discussed. Finiteness of orbits and that of register types in this paper are related, but deeper observation is left as future study. *LTL with freeze quantifier*: Linear temporal logic (LTL) was extended to LTL↓ with freeze quantifier [7,8]. The relationship among subclasses of LTL↓ and RA as well as the decidability and complexity of the satisfiability (nonemptiness) problems are investigated [7]. 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 $\mathrm{FO}^2(<,+1)$ 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 [3] that $\mathrm{FO}^2(\sim,<,+1)$ with data equality $\sim$ is decidable on data words. Note that $\mathrm{FO}^2(\sim,<,+1)$ is incomparable with LTL↓ of [7]. Also it was shown in [2] that $\mathrm{FO}^2(\sim,+1)$ and existential $\mathrm{MSO}^2(\sim,+1)$ are decidable on unranked data trees.

2 Register Context-Free Grammars

Let $\mathbb{N} = \{1,2,\ldots\}$ and $\mathbb{N}_0 = \{0\} \cup \mathbb{N}$. 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 $d \in D$, $\theta \in \Theta_k$ and $\psi \in F_k$, the satisfaction relation $d, \theta \models \psi$ is defined as $d, \theta \models x_i^{=}$ 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 $\|G\| = |V| + |R| \max\{(|\alpha|+1)(\log|V| + \log k) + \|\psi\| \mid (A, \psi, i)/(A, \psi) \to \alpha \in R\}$, where $\|\psi\|$ is the description length of ψ.

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 $r = (A, \psi, i) \to c_1 \dots c_n \in R$ (resp. $r = (A, \psi) \to c_1 \dots c_n \in R$) such that

$$d, \theta \models \psi, X = c'_1 \dots c'_n, \theta' = \theta[i \leftarrow d] \text{ (resp. } \theta' = \theta) \text{ where}$$
$$c'_j = \begin{cases} (B, \theta') & \text{if } c_j = B \in V, \\ (b, \theta'(l)) & \text{if } c_j = (b, l) \in \Sigma \times [k]. \end{cases}$$

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 $X \Rightarrow^d_{G,r} Y$.

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

Example 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, $L(G) = \{(a, d_0)(b, d_1) \dots (b, d_n)(a, d_0)(b, d_n) \dots (b, d_1)(a, d_0) \mid n \geq 0, d_i \neq d_0 \text{ for } i \in [n]\}$.

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 2. *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 a register 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 $r = (A,\psi,i)/(A,\psi) \to \alpha$, r can be applied for any (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:

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

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 $\psi^{=} = \{i_1,\ldots,i_m\}$ and $\psi^{\neq} = \{j_1,\ldots,j_n\}$. We assume $\psi^{=} \cap \psi^{\neq} = \emptyset$ (the rule with $\psi^{=} \cap \psi^{\neq} \neq \emptyset$ can be removed). For $\gamma \in \Gamma_k$ and $\psi \in F_k$ in the form of (1), we define

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

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 $\gamma \models \psi$.

$$\theta \models \gamma \;\Rightarrow\; (\gamma \models \psi \Longleftrightarrow \exists d.\ d,\theta \models \psi).$$

Lemma 3. *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:* $x_1^{=}, x_2^{=}, \ldots, x_k^{=}$, $x_1^{\neq} \wedge \cdots \wedge x_k^{\neq}$.

Proof. We assume that the guard expression of every rule of G has the form of (1). For such a guard expression ψ and a register type $\gamma \in \Gamma_k$, let $[\psi, \gamma]$ be the set of guard expressions defined as follows.

$$[\psi, \gamma] = \begin{cases} \{x_i^{=} \mid i = \min \psi^{=}\} & \text{if } \psi^{=} \neq \emptyset, \\ \{x_i^{=} \mid i \in [k] \setminus \bigcup_{j \in \psi^{\neq}} \gamma[j]\} \cup \{x_1^{\neq} \wedge \ldots \wedge x_k^{\neq}\} & \text{if } \psi^{=} = \emptyset. \end{cases}$$

Then the following properties hold.

(i) $\psi' \in [\psi, \gamma] \;\Rightarrow\; \forall \theta\, \exists d.\ d, \theta \models \psi'$ (The guard in $[\psi, \gamma]$ is always satisfiable.)
(ii) $\theta \models \gamma \;\Rightarrow\; \forall d.\ (d, \theta \models \psi \iff \exists \psi' \in [\psi, \gamma].\ d, \theta \models \psi')$
(The same input value can be used for ψ and $[\psi, \gamma]$.)
(iii) $\psi' \in [\psi, \gamma] \;\Rightarrow\; \forall i\, \exists \gamma'\, \forall \theta, d.\ ((\theta \models \gamma \wedge d, \theta \models \psi') \;\Rightarrow\; \theta[i \leftarrow d] \models \gamma')$.
(The register type after the rule application is unique.)

The register type γ' in the above third property is uniquely determined by γ, ψ', and i, and we write the register type as $\mathit{after}(\gamma, \psi', i)$.

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 rule, 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, $(X_1 \ldots X_n)^{\mathrm{aug}(\gamma')} = X_1' \ldots X_n'$ where $X_\ell' = (X_\ell, \gamma')$ if $X_\ell \in V$ and $X_\ell' = X_\ell$ otherwise for every $\ell \in [n]$.

$$\frac{(A, \psi, i) \to \alpha \in R \quad (\text{resp. } (A, \psi) \to \alpha \in R) \qquad \gamma \models \psi,\ \psi' \in [\psi, \gamma],\ \gamma' = \mathit{after}(\gamma, \psi', i) \quad (\text{resp. } \gamma' = \gamma)}{((A, \gamma), \psi', i) \to \alpha^{\mathrm{aug}(\gamma')} \in R' \quad (\text{resp. } ((A, \gamma), \psi') \to \alpha^{\mathrm{aug}(\gamma')} \in R')}$$

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

- For a derivation of a data word w in G, if we replace each (A, θ) with $((A, \gamma), \theta)$ where γ is the register type that satisfies $\theta \models \gamma$, then we obtain a derivation of w in G'. Note that by the above property (ii), there must exist $\psi' \in [\psi, \gamma]$ that allows this derivation in G' where ψ is the guard expression of the rule used for the derivation in G.
- For every $((A, \gamma), \theta)$ appearing in a derivation in G', it holds that $\theta \models \gamma$.
- 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.

3.3 ε-rule Removal

Theorem 4. *For an arbitrary k-RCFG, we can construct an equivalent k-RCFG having no ε-rule.*

Proof. By Lemma 3, 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 $x_1^{=}, \ldots, x_k^{=}$, or $x_1^{\neq} \wedge \ldots \wedge x_k^{\neq}$. 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 $A \Rightarrow_{G'}^{*} \varepsilon$) 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.

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 $\mathbb{D} = (D, \mathcal{R})$ 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

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

for any $d \in D$ and $\theta \in \Theta_k$. 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.

The following properties can be proved in a similar way to the case of k-GRCFG(=) [6].

Theorem 5. *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.*

4.2 Simulation Property and Type Oracle

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 6. 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 $\varphi = x_1^< \wedge x_2^>$ be a guard expression of 3-GRCFG($\mathbb{Z}$) and consider register 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 $d, \theta_2 \models \varphi$ while $5, \theta_1 \models \varphi$. □

Similarly, the membership and emptiness lose decidability for GRCFG because a binary relation appearing in a guard expression may be an undecidable relation. To limit the influence of binary relations in a data structure so that GRCFG have mild expressive power, we introduce two properties of a GRCFG, namely, the simulation property and (the existence) of type oracle.

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 $\gamma : ([k] \times [k]) \backslash \{(i,i) \mid i \in [k]\} \to \{\mathtt{tt}, \mathtt{ff}\}$[1]. We say that the type of a register assignment θ is γ (and write $\theta \models \gamma$) iff for all $i, j \in [k]$ $(i \neq j)$,

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

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

Definition 7 (Simulation). *Let G be a k-GRCFG($\bowtie$) $G = (V, R, S)$. G has the simulation property (with respect to $\bowtie$) if the following condition is met.*

For all $\theta, \theta' \in \Theta_k$, $d \in D$, $r = (A, \varphi, i)/(A, \varphi) \to \alpha \in R$ such that $\theta \sim_{\bowtie} \theta'$ and $d, \theta \models \varphi$, there exists $d' \in D$ such that $d', \theta' \models \varphi$, and if the left-hand side of r is (A, φ, i), then $\theta[i \leftarrow d] \sim_{\bowtie} \theta'[i \leftarrow d']$.

The following diagram illustrates the condition of the simulation property.

$$\begin{array}{ccc} (A,\ \theta) & \Longrightarrow_r^d & \ldots (B,\ \theta[i \leftarrow d]) \ldots \\ \wr & & \wr \\ (A, \theta') & \Longrightarrow_r^{d'} & \ldots (B, \theta'[i \leftarrow d']) \ldots \end{array}$$

Definition 8 (Type Oracle). *Let $O : \Gamma_k \times F_k \to \{\mathtt{tt}, \mathtt{ff}\}$ be the predicate defined by: for $\gamma \in \Gamma_k$ and $\psi \in F_k$, $O(\gamma, \psi) = \mathtt{tt}$ iff*

there are $\theta \in \Theta_k$ and $d \in D$ such that $\theta \models \gamma$ and $d, \theta \models \psi$.

We say that D has the type oracle if there is a polynomial time algorithm that answers whether $O(\gamma, \psi) = \mathtt{tt}$ or $\mathtt{ff}$ for given $\gamma \in \Gamma_k$ and $\psi \in F_k$. □

[1] 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 $\theta(i) \bowtie \theta(i)$.

Finally, we define *data type* as an extension of register type by adding the information on equality between data values in the registers and data values appearing in a given data word w.

Definition 9 (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 $d_{\neq} \notin D_w$ be a newly introduced symbol. We use a function $e : [k] \to D_w \cup \{d_{\neq}\}$, whose codomain is finite, to represent the register assignment by replacing every data value that does not appear in w with $d_{\neq}$. We write $\theta \models e$ iff for all $i \in [k]$,*

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

The collection of all such functions $e : [k] \to D_w \cup \{d_{\neq}\}$ is denoted by $E_{w,k}$.

The data type *of a register 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 property with data type and the data type oracle $O_w(\gamma, e, \varphi)$ of $w \in (\Sigma \times D)^*$ defined for $\gamma \in \Gamma, e \in E_{w,k}, \varphi \in F_k$ in the same way as in the case of register types.*

5 Properties of GRCFG

5.1 ε-rule Removal

Theorem 10. *For an arbitrary GRCFG($\bowtie$) G such that G has the simulation property and D has the type oracle, we can construct an equivalent GRCFG($\bowtie$) G' having no ε-rule.*

Proof. The theorem can be proved in a similar way to Theorem 4 by using the simulation property and the type oracle. Let $G = (V, R, S)$ be a k-GRCFG($\bowtie$). We assume that the guard expression of every rule in R is the 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 defined as follows. Define the subset of guard expressions Ψ as

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

  Let $r = (A, \varphi, i) \to \alpha \in R$. (A rule $(A, \varphi) \to \alpha$ can be processed in a similar way.) Also let $\gamma \in \Gamma_k$ and $\psi \in \Psi$. If $O(\gamma, \varphi \wedge \psi) = \mathtt{tt}$,

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

  where $\gamma' \in \Gamma_k$ is a register type that satisfies $\theta[i \leftarrow d] \models \gamma'$ for any θ and d such that $\theta \models \gamma$ and $d, \theta \models \varphi \wedge \psi$ (see the proof of Lemma 3 for the definition of $\alpha^{\mathrm{aug}(\gamma')}$). Note that γ' must exist and γ' is uniquely determined by γ, $\varphi \wedge \psi$ 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 ψ 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 $\bot^k \models \gamma_0$.

See an example of the construction in Example 11. We can show $L(G) = L(G')$ by induction on the length of derivations in G and G', using the simulation property (to show $L(G') \subseteq L(G)$) and the type oracle (to show both inclusions).

The rest of the proof is similar to the one in Theorem 4.

Example 11. Let $k = 2$ and consider a rule $r = (A, \varphi, 1) \to \alpha$ where $\varphi = x_1^{\bowtie} \wedge x_2^{\bowtie^{-1}}$. 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 $\delta_{21} = (\theta(2) \bowtie \theta(1))$[2]. After the elimination of the unsatisfiable ones and Boolean simplification, we can assume that $\Psi = \{\psi_1, \psi_2, \psi_3, \psi_4\}$ where $\psi_1 = x_1^{\bowtie^{-1}} \wedge x_2^{\bowtie}$, $\psi_2 = x_1^{\bowtie^{-1}} \wedge \neg x_2^{\bowtie}$, $\psi_3 = \neg x_1^{\bowtie^{-1}} \wedge x_2^{\bowtie}$ and $\psi_4 = \neg x_1^{\bowtie^{-1}} \wedge \neg x_2^{\bowtie}$. If $O(\gamma_i, \varphi \wedge \psi_j) = \mathtt{tt}$, the register type γ' after the rule application is $\gamma_1, \gamma_2, \gamma_1, \gamma_2$ for $\psi_1, \psi_2, \psi_3, \psi_4$, respectively. In this example, the type γ' is determined depending only on ψ_j and independent of γ_i because $k = 2$ and an input data value is loaded to the first register when r is applied.

5.2 Emptiness and Membership

Theorem 12. *The emptiness problem for GRCFG($\bowtie$) such that G has the simulation property and D has the type oracle, is EXPTIME-complete.*

Proof. Let $G = (V, R, S)$ be such a k-GRCFG($\bowtie$) and $G' = (V', R', S')$ be the k-GRCFG($\bowtie$) constructed from G in the proof of Theorem 10. As shown in that proof, $L(G') = L(G)$. We construct CFG $G'' = (V', R'', S')$ from G' where

$$R'' = \{(A, \gamma) \to X_1 \dots X_n \mid ((A, \gamma), \varphi, i)/((A, \gamma), \varphi) \to X'_1 \dots X'_n \in R' \text{ for some } \varphi \text{ and } i, \text{ and } X_j = X'_j \text{ if } X'_j \in V' \text{ and } X_j = a \text{ if } X_j \notin V' \text{ for each } j \in [n]\}.$$

We can easily show $L(G') = \emptyset \Leftrightarrow L(G'') = \emptyset$ because a rule application is never blocked in G'.

Because the size of the CFG constructed in this way is exponential to k and 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(=) [17].

Theorem 13. *The membership problem for GRCFG($\bowtie$) such that G has the simulation property with data type and D has the data type oracle, is EXPTIME-complete.*

(This theorem can be proved in a similar way to Theorem 12.)

[2] 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.

5.3 GRCFG with a Total Order on a Dense Set

Lemma 14. *Every GRCFG($<_Q$) has the simulation property and Q has the type oracle where $<_Q$ is the strict total order on the set Q of all rational numbers. Similarly, it has the simulation property with data type and Q has the data type oracle.*

Proof. We abbreviate $<_Q$ as $<$. Let $G = (V, R, S)$ be a k-GRCFG($<$), $\theta \in \Theta_k$, $\gamma \in \Gamma_k$ and $r = (A, \varphi, i) \to \alpha \in R$ where φ is the conjunction of literals (of the form $x_i^<$ or $\neg x_j^<$). (The case $r = (A, \varphi) \to \alpha \in R$ 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 $d, \theta \models \varphi$. The condition $d, \theta \models \varphi$ 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, if $\theta \models \gamma$, $\theta' \models \gamma$, $d, \theta \models \varphi$ and $\theta[i \leftarrow d] \models \gamma'$, there is $d' \in Q$ satisfying $d', \theta \models \varphi$ and $\theta'[i \leftarrow d'] \models \gamma'$ and the simulation property holds.

Similarly, for deciding $O(\gamma, \varphi) = \mathtt{tt}$, it suffices to represent the condition

$$d, \theta \models \varphi \ \wedge \ \theta \models \gamma$$

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

We can show the simulation property with data type and the existence of the data type oracle in a similar way.

Example 15. Consider a 2-GRCFG($<$)$=$ (V, R, S) and a rule $(A, \varphi, 1) \to B \in R$ where $\varphi = x_1^< \wedge \neg x_2^<$. We see that $d, \theta \models \varphi \Leftrightarrow \theta(1) < d \leq \theta(2)$. 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 $d, \theta \models \varphi$ 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 16. *For a given GRCFG($<_Q$), we can construct an equivalent GRCFG($<_Q$) having no ε-rule. The emptiness and membership problems are both EXPTIME-complete for GRCFG($<_Q$).*

Proof. By Lemma 14 and Theorems 10, 12 and 13.

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 property and the existence of type oracle, the decidability of emptiness and membership for GRCFG and a transformation to an ε-free GRCFG have been provided.

Nominal CFG [4] with equality symmetry, total order symmetry and integer symmetry correspond to GRCFG(=), GRCFG($<_Q$) (Sect. 5.3) and GRCFG($<_Z$) (Example 6), respectively. Investigating the relation between nominal CFG and GRCFG in depth is future work.

References

1. Benedikt, M., Ley, C., Puppis, G.: What you must remember when processing data words. In: 4th Alberto Mendelzon International Workshop on Foundations of Data Management (2010)
2. Bojańczyk, M., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data trees and XML reasoning. J. ACM **56**(3), 13:1–13:48 (2009). https://doi.org/10.1145/1516512.1516515
3. Bojańczyk, M., David, C., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data words. ACM Trans. Comput. Log. **12**(4), 27:1–27:26 (2011). https://doi.org/10.1145/1970398.1970403
4. Bojańczyk, M., Klin, B., Lasota, S.: Automata theory in nominal sets. Log. Methods Comput. Sci. **10**(3) (2014). https://doi.org/10.2168/LMCS-10(3:4)2014
5. Bouyer, P.: A logical characterization of data languages. Inf. Process. Lett. **84**(2), 75–85 (2002). https://doi.org/10.1016/S0020-0190(02)00229-6
6. Cheng, E.Y., Kaminski, M.: Context-free languages over infinite alphabets. Acta Inf. **35**(3), 245–267 (1998). https://doi.org/10.1007/s002360050120
7. Demri, S., Lazić, R.: LTL with the freeze quantifier and register automata. ACM Trans. Comput. Log. **10**(3), 16:1–16:30 (2009). https://doi.org/10.1145/1507244.1507246
8. Demri, S., Lazić, R., Nowak, D.: On the freeze quantifier in constraint LTL: decidability and complexity. Inf. Comput. **205**(1), 2–24 (2007). https://doi.org/10.1016/j.ic.2006.08.003
9. Figueira, D., Hofman, P., Lasota, S.: Relating timed and register automata. Math. Struct. Comput. Sci. **26**(6), 993–1021 (2016). https://doi.org/10.1017/S0960129514000322
10. Kaminski, M., Francez, N.: Finite-memory automata. Theor. Comput. Sci. **134**(2), 329–363 (1994). https://doi.org/10.1016/0304-3975(94)90242-9
11. Libkin, L., Martens, W., Vrgoč, D.: Querying graphs with data. J. ACM **63**(2), 14:1–14:53 (2016). https://doi.org/10.1145/2850413
12. Libkin, L., Tan, T., Vrgoč, D.: Regular expressions for data words. J. Comput. Syst. Sci. **81**(7), 1278–1297 (2015). https://doi.org/10.1016/j.jcss.2015.03.005
13. Libkin, L., Vrgoč, D.: Regular path queries on graphs with data. In: 15th International Conference on Database Theory (ICDT 2012), pp. 74–85 (2012). https://doi.org/10.1145/2274576.2274585
14. Milo, T., Suciu, D., Vianu, V.: Typechecking for XML transformers. In: 19th ACM Symposium on Principles of Database Systems (PODS 2000), pp. 11–22 (2000). https://doi.org/10.1145/335168.335171

15. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Trans. Comput. Log. **5**(3), 403–435 (2004). https://doi.org/10.1145/1013560.1013562
16. Sakamoto, H., Ikeda, D.: Intractability of decision problems for finite-memory automata. Theor. Comput. Sci. **231**(2), 297–308 (2000). https://doi.org/10.1016/S0304-3975(99)00105-X
17. Senda, R., Takata, Y., Seki, H.: Complexity results on register context-free grammars and register tree automata. In: Fischer, B., Uustalu, T. (eds.) ICTAC 2018. LNCS, vol. 11187, pp. 415–434. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02508-3_22