Monoid-based Approach to the Inclusion Problem on Superdeterministic Pushdown Automata

Yuya Uezato[1] and Yasuhiko Minamide[2]

[1] Department of Computer Science, University of Tsukuba, Ibaraki, Japan
uezato@logic.cs.tsukuba.ac.jp
[2] Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Japan

Abstract. We present a new and simple decidability proof for the language inclusion problem between context-free languages and languages accepted by superdeterministic pushdown automata (SDPDAs). The language class of SDPDAs is one of the largest language classes $\mathcal{C}$ for which the inclusion $L_{\mathrm{cfl}} \subseteq L_{\mathcal{C}}$ is decidable for an arbitrary context-free language L_{cfl} and arbitrary language $L_{\mathcal{C}}$ in $\mathcal{C}$. We introduce generalized pushdown automata and reformulate SDPDAs as a subclass of them. This reformulation naturally leads to a monoid that captures SDPDAs. The monoid is key to our simple decidability proof because we translate the inclusion problem on SDPDAs to the corresponding monoid inclusion problem. In addition to the decidability result, we present a new undecidability result regarding the inclusion problem on indexed languages.

1 Introduction

A superdeterministic pushdown automaton (SDPDA) is a deterministic pushdown automaton that is finite delay and satisfies a peculiar condition—for any state p and word w, there is a state q and $z \in \mathbb{Z}$ such that for any configuration $\langle p, \alpha\rangle$, if a computation starting from $\langle p, \alpha\rangle$ ends in $\langle p', \beta\rangle$ after consuming w, then $p' = q$ and $|\alpha| - |\beta| = z$. Greibach and Friedman showed the decidability of $\mathit{Incl}(\mathbf{CFL}, \mathbf{SDPDA})$, i.e., the inclusion $L_{\mathrm{cfl}} \subseteq L(M)$ is decidable for an arbitrary context-free language L_{cfl} and arbitrary SDPDA M [6]. They also showed that the language class **SDPDA** includes some important classes, i.e., the class of regular languages (**REG**), Dyck languages (**DYCK**), and generalized parenthesis languages [13]. Moreover, a language class $\mathcal{C}$ for which $\mathit{Incl}(\mathbf{CFL}, \mathcal{C})$ is decidable and $\mathcal{C}$ strictly includes **SDPDA** is not yet known. Our aim is to obtain a simple decidability proof for $\mathit{Incl}(\mathbf{CFL}, \mathbf{SDPDA})$ and extend it to a larger class. The original proof [6], however, is elaborated by pumping arguments and it remains unclear why $\mathit{Incl}(\mathbf{CFL}, \mathbf{SDPDA})$ is decidable.

We introduce generalized pushdown automata (GPDAs) and a subclass, real-time GPDAs (RGPDAs). Each transition rule of GPDAs is of the form $p \xrightarrow[\sigma]{\alpha \,/\, \beta} q$, which consumes an input σ, pops a sequence of symbols α, and then pushes β to a stack. Although usual pushdown automata require $|\alpha| = 1$, GPDAs allow $|\alpha| > 1$ and pop multiple symbols in one transition. On the basis of the multiple

pop feature, we translate SDPDAs to RGPDAs that satisfy the following property: if we have $p \xrightarrow[\sigma]{\alpha\,/\,\beta} q$ and $p \xrightarrow[\sigma]{\alpha'\,/\,\beta'} q'$, then $q = q'$, $|\alpha| = |\alpha'|$, and $|\beta| = |\beta'|$. This translation simplifies our decidability proof; thus, RGPDAs with the above condition are adequate normal forms of SDPDAs.

The formalization of RGPDAs instinctively leads to a monoid for RGPDAs and this monoid is the basis of our decidability proof. Our approach generalizes the monoid-based approach for $\mathit{Incl}(\mathbf{CFL}, \mathbf{REG})$ and $\mathit{Incl}(\mathbf{CFL}, \mathbf{DYCK})$ [11] where the author applied the classical notion of language recognition by monoids to translate inclusion problems to corresponding monoid inclusion problems. For a finite automaton A, there is a finite monoid $\mathbf{M}$, a subset $U \subseteq \mathbf{M}$, and a homomorphism $\mathcal{H} : \Sigma^* \to \mathbf{M}$ that recognize $L(A)$ as $L(A) = \mathcal{H}^{-1}(U)$. This equation translates the inclusion $L(G) \subseteq L(A)$ to the monoid inclusion $\mathcal{H}(L(G)) \subseteq U$ where G is a context-free grammar. Since $\mathbf{M}$ is a finite monoid, we can decide whether $\mathcal{H}(L(G)) \subseteq U$ and this implies the decidability of $\mathit{Incl}(\mathbf{CFL}, \mathbf{REG})$. This argument, however, cannot be directly applied to $\mathit{Incl}(\mathbf{CFL}, \mathbf{DYCK})$ because a monoid that recognizes a Dyck language is infinite. In [11], to manage this unavoidable infiniteness, the author rephrased an argument given by Berstel and Boasson [2] for the decidability of $\mathit{Incl}(\mathbf{CFL}, \mathbf{DYCK})$ in terms of monoids. The present paper generalizes their argument to accommodate $\mathit{Incl}(\mathbf{CFL}, \mathbf{SDPDA})$ on the basis of our monoid for RGPDAs. Tsukada and Kobayashi gave a procedure similar to ours in a type-theoretical framework [14]. We compare our approach with their type-theoretical approach in Section 6.

Recently, higher-order PDAs [10] have received much attention for their use in higher-order program verification [8]. Hence, it is a natural attempt to extend the decidability of $\mathit{Incl}(\mathbf{CFL}, \mathbf{SDPDA})$ to a class of languages accepted by higher-order PDAs. However, unfortunately, we show that such an attempt is even undecidable for $\mathit{Incl}(\mathbf{IL}, \mathbf{DYCK})$ where $\mathbf{IL}$ is the class of indexed languages [1, 7] accepted by second-order PDAs [10].

Context-free Grammar and Normal Form. A *context-free grammar* (CFG) is a 4-tuple $G = (V, \Sigma, P, S)$ where V is a finite set of *variables*, Σ is a finite set of *terminal symbols*, $P \subseteq V \times (V \cup \Sigma)^*$ is a finite set of *production rules*, and $S \in V$ is the *start variable*. We write $X \to \alpha$ instead of $(X, \alpha) \in P$. To denote a one-step derivation, we write $\alpha X \beta \Rightarrow \alpha\,\xi\,\beta$ if there is a rule $X \to \xi \in P$ where $\alpha, \beta, \xi \in (V \cup \Sigma)^*$. The words generated by a variable X is $L(X) \coloneqq \{w \in \Sigma^* : X \Rightarrow^* w\}$ and the language of G, $L(G)$, is defined by $L(G) \coloneqq L(S)$.

We primarily use the Chomsky normal form (CNF). A CFG is in CNF if all of its production rules are of the form $X \to YZ$, $X \to \sigma$, or $S \to \epsilon$ where $X, Y, Z \in V$ and $\sigma \in \Sigma$. By the standard translation from CFG to CNF [7], we assume that each variable X is *reachable*, i.e., there exists a pair of terminal strings (w_1, w_2) such that $S \Rightarrow^* w_1 X w_2$.

2 Generalized PDA and Superdeterministic PDA

First, we introduce *generalized pushdown automata* (GPDAs). Next, we define *realtime* GPDAs (RGPDAs) and a monoid for GPDAs. Third, we define push-

down automata (PDAs) as a subclass of GPDAs and superdeterministic PDAs (SDPDAs) by following [6]. Finally, after pointing out a problem in the formalization of SDPDAs, we translate SDPDAs into RGPDAs.

Generalized PDA. A GPDA is a 7-tuple $M = (Q, \Sigma, \Gamma, \Delta, q_{\text{init}}, \mathcal{F}, Z)$ where Q is a finite set of *states*, Σ is a finite *input alphabet*, Γ is a finite *stack alphabet*, $\Delta \subseteq (Q \times (\Sigma \cup \{\epsilon\}) \cup \Gamma^+) \times (Q \times \Gamma^*)$ is a finite set of *transition rules*, $q_{\text{init}} \in Q$ is the *initial state*, $\mathcal{F} \subseteq Q \times \Gamma^*$ is a finite set of *final configurations*, and $Z \in \Gamma$ is the *initial stack symbol*. We use Γ^+ to denote $\Gamma^* \setminus \{\epsilon\}$.

A *configuration* $\mathbf{c}$ is a pair $\langle p, \alpha \rangle \in Q \times \Gamma^*$ of a state p and a stack α. We define the set of *transitions* $\mathbb{T} \coloneqq Q \times \Gamma^* \times \Gamma^* \times Q$ and write $p \xrightarrow{\alpha \,/\, \beta} q$ to denote a transition $(p, \alpha, \beta, q) \in \mathbb{T}$. A transition $\delta = p \xrightarrow{\alpha \,/\, \beta} q$ rewrites a configuration $\mathbf{c}$ to another one $\mathbf{c}'$ as $\mathbf{c} \overset{\delta}{\hookrightarrow} \mathbf{c}'$ if $\mathbf{c} = \langle p, \alpha\xi \rangle$, $\mathbf{c}' = \langle q, \beta\xi \rangle$, and $\xi \in \Gamma^*$. We use Δ as a function from $\Sigma \cup \{\epsilon\}$ to $2^{\mathbb{T}}$ defined by $\Delta(a) \coloneqq \{p \xrightarrow{\alpha \,/\, \beta} q : ((p, a, \alpha), (q, \beta)) \in \Delta\}$.

We define a *single move* $\mathbf{c} \underset{a}{\vdash} \mathbf{c}'$ if $\mathbf{c} \overset{\delta}{\hookrightarrow} \mathbf{c}'$ for some $\delta \in \Delta(a)$ where $a \in \Sigma \cup \{\epsilon\}$ and a *multiple move* $\mathbf{c}_1 \overset{*}{\underset{a_1 a_2 \ldots a_n}{\vdash}} \mathbf{c}_{n+1}$ if $\mathbf{c}_i \underset{a_i}{\vdash} \mathbf{c}_{i+1}$ for all $i \in [1..n]$. The language of M, $L(M)$, is defined as follows:

$$L(M) \coloneqq \{w \in \Sigma^* : \langle q_{\text{init}}, Z \rangle \overset{*}{\underset{w}{\vdash}} \langle q_f, \xi \rangle,\ \langle q_f, \xi \rangle \in \mathcal{F}\}.$$

Realtime* GPDA *and Transition Monoid. We introduce a subclass of GPDAs, *realtime* GPDAs, and define a monoid and homomorphism to recognize the language accepted by a realtime GPDA.

A GPDA $M = (Q, \Sigma, \Gamma, \Delta, q_{\text{init}}, \mathcal{F}, Z)$ is *realtime* (RGPDA) if there are no ϵ-moves; namely, $\Delta \subseteq (Q \times \Sigma \times \Gamma^+) \times (Q \times \Gamma^*)$.

We define a *composition* operator $\odot$ on $\mathbb{T}_\perp (= \mathbb{T} \cup \{\perp\})$ as follows:

$$\delta_1 \odot \delta_2 \coloneqq \begin{cases} p \xrightarrow{\alpha \,/\, \zeta\xi} r & \text{if } \delta_1 = p \xrightarrow{\alpha \,/\, \beta\xi} q \text{ and } \delta_2 = q \xrightarrow{\beta \,/\, \zeta} r, \\ p \xrightarrow{\alpha\xi \,/\, \zeta} r & \text{if } \delta_1 = p \xrightarrow{\alpha \,/\, \beta} q \text{ and } \delta_2 = q \xrightarrow{\beta\xi \,/\, \zeta} r, \\ \perp & \text{otherwise,} \end{cases} \quad \begin{array}{l} \perp \odot \_ \coloneqq \perp, \\ \\ \_ \odot \perp \coloneqq \perp, \end{array}$$

where the element $\perp$ denotes a composition failure, e.g., $p \xrightarrow{a \,/\, b} q \odot q \xrightarrow{c \,/\, d} r = \perp$ because it means to push b to a stack and then try to pop c but we cannot.

The operator $\odot : \mathbb{T}_\perp \times \mathbb{T}_\perp \to \mathbb{T}_\perp$ is associative; thus, the pair $(\mathbb{T}_\perp, \odot)$ forms a semigroup. This semigroup leads to a monoid $\mathbf{T}_M$ for the RGPDA M: $\mathbf{T}_M \coloneqq (2^{\mathbb{T}}, \otimes, \mathbf{1} = \{q \xrightarrow{\epsilon \,/\, \epsilon} q : q \in Q\})$ where the multiplication $\otimes$ is defined by extending $\odot$ to the sets of transitions:

$$T_1 \otimes T_2 \coloneqq \{\delta_1 \odot \delta_2 : \delta_1 \in T_1, \delta_2 \in T_2, \delta_1 \odot \delta_2 \neq \perp\}.$$

Since there are no ϵ-moves in RGPDAs, we can see Δ as a function $\Delta : \Sigma \to 2^{\mathbb{T}}$ and this derives a homomorphism $\widetilde{\Delta} : \Sigma^* \to \mathbf{T}_M$ as follows:

$$\widetilde{\Delta}(\epsilon) \coloneqq \mathbf{1}, \quad \widetilde{\Delta}(\sigma) \coloneqq \Delta(\sigma), \quad \widetilde{\Delta}(\sigma_1 \ldots \sigma_n) \coloneqq \Delta(\sigma_1) \otimes \cdots \otimes \Delta(\sigma_n).$$

The homomorphism $\widetilde{\Delta}$ naturally interprets moves of M as follows.

Proposition 1. – *If* $p \xrightarrow{\alpha\,/\,\beta} q \in \widetilde{\Delta}(w)$, *then* $\langle p, \alpha\xi\rangle \vdash_w^* \langle q, \beta\xi\rangle$ *for any* $\xi \in \Gamma^*$.
– *If* $\langle p, \alpha\rangle \vdash_w^* \langle q, \beta\rangle$, *then there exists* $\xi \in \Gamma^*$ *such that* $\alpha = \alpha'\xi$, $\beta = \beta'\xi$, *and* $p \xrightarrow{\alpha'\,/\,\beta'} q \in \widetilde{\Delta}(w)$.

Thus, the homomorphism $\widetilde{\Delta}$ recognizes the language $L(M)$.

Lemma 2. $L(M) = \widetilde{\Delta}^{-1}(\{T : q_{init} \xrightarrow{Z\,/\,\xi} q_f \in T,\ \langle q_f, \xi\rangle \in \mathcal{F}\})$.

Note that $\widetilde{\Delta}(w)$ is finite for any $w \in \Sigma^*$ because $\Delta(\sigma)$ is finite for any $\sigma \in \Sigma$. This finiteness is important to obtain a decision procedure for inclusion problems. Although we can show properties similar to Proposition 1 for GPDAs, we require ϵ-closures to build a homomorphism $\mathcal{H}$ and then $\mathcal{H}(w)$ is infinite in general. The existence of ϵ-moves is harmful to give a decision procedure.

PDA and Deterministic PDA. A GPDA $M = (Q, \Sigma, \Gamma, \Delta, q_{\text{init}}, \mathcal{F}, Z)$ is a *pushdown automaton* (PDA) if $\Delta \subseteq (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$. In contrast to GPDAs, PDAs cannot pop multiple symbols in a single move.

A PDA M is *deterministic* (DPDA) if M satisfies the following conditions:

– For each $a \in \Sigma \cup \{\epsilon\}$, if $\mathbf{c}_1 \vdash_a \mathbf{c}_2$ and $\mathbf{c}_1 \vdash_a \mathbf{c}_3$, then $\mathbf{c}_2 = \mathbf{c}_3$.
– If we have $\mathbf{c}_1 \vdash_\epsilon \mathbf{c}_2$, then $\mathbf{c}_1 \not\vdash_\sigma \mathbf{c}_3$ for all $\sigma \in \Sigma$ and $\mathbf{c}_3 \in Q \times \Gamma^*$.

A configuration $\mathbf{c}$ is a *reading configuration* if $\mathbf{c} \vdash_\sigma \mathbf{c}'$ for some $\sigma \in \Sigma$. To emphasize a move between reading configurations, we write $\mathbf{c} \Vdash_w^* \mathbf{c}'$ instead of $\mathbf{c} \vdash_w^* \mathbf{c}'$ where $\mathbf{c}$ and $\mathbf{c}'$ are reading configurations.

Superdeterministic PDA. A DPDA M is *superdeterministic* (SDPDA) [6] if M satisfies the following conditions:

1. M accepts words with the empty stack: if $\langle q_f, \xi\rangle \in \mathcal{F}$, then $\xi = \epsilon$.
2. M is *finite delay*, i.e., any sequence of ϵ-moves is d-bound for some $d \in \mathbb{N}$: there are no configurations $\mathbf{c}$ such that $\mathbf{c} = \mathbf{c}_0 \vdash_\epsilon \mathbf{c}_1 \vdash_\epsilon \cdots \vdash_\epsilon \mathbf{c}_{d-1} \vdash_\epsilon \mathbf{c}_d$.
3. Let $\sigma \in \Sigma$ and $\langle p, \alpha\rangle$ and $\langle p, \alpha'\rangle$ be reading configurations with the same state. If $\langle p, \alpha\rangle \Vdash_\sigma^* \langle q, \beta\rangle$ and $\langle p, \alpha'\rangle \Vdash_\sigma^* \langle r, \beta'\rangle$, then $q = r$ and $|\alpha| - |\beta| = |\alpha'| - |\beta'|$.

As mentioned above, the presence of ϵ-moves in the formalization of SDPDAs prevents us from directly defining a homomorphism that recognizes $L(M)$. Thus, we remove ϵ-moves from SDPDAs by translating them to RGPDAs.

Theorem 3. *Let* $M = (Q, \Sigma, \Gamma, \Delta, q_{init}, \mathcal{F}, Z)$ *be an* SDPDA. *There exists* RG-PDA N *such that (1)* $\$L(M) = L(N)$ *where* $\$ \notin \Sigma$ *and (2) if* $p \xrightarrow{\alpha\,/\,\beta} q \in \Delta_N(\sigma)$ *and* $p \xrightarrow{\alpha'\,/\,\beta'} r \in \Delta_N(\sigma)$, *then* $q = r$, $|\alpha| = |\alpha'|$, *and* $|\beta| = |\beta'|$.

Proof. We can easily translate M to an SDPDA M' that satisfies the following:

$$M : \langle q_{\text{init}}, Z\rangle \vdash_w^* \langle q_f, \epsilon\rangle \iff M' : \langle q'_{\text{init}}, \natural\rangle \vdash_\epsilon \langle q_{\text{init}}, Z\natural\rangle \vdash_w^* \langle q_f, \natural\rangle \vdash_\epsilon \langle q'_f, \epsilon\rangle$$

where $\langle q_f, \epsilon\rangle \in \mathcal{F}$ and the two states $q'_{\text{init}}, q'_f \notin Q$ are the unique initial and final states of M'. Thus, $L(M) = L(M')$. We assume that M' is d-bound and the special symbol $\natural \notin \Gamma$ is the stack bottom symbol of M'.

We take the ϵ-closure of the initial configuration of M'. If $\langle q'_{\text{init}}, \natural\rangle \overset{*}{\vdash_\epsilon} \langle q, \epsilon\rangle$, then $L(M') = \emptyset$ or $L(M') = \{\epsilon\}$ and these cases are easy. We assume $\langle q'_{\text{init}}, \natural\rangle \overset{*}{\vdash_\epsilon} \langle q_\star, \alpha_\star\natural\rangle$ where $\langle q_\star, \alpha_\star\natural\rangle$ is a reading configuration. If $\sigma_1 \dots \sigma_n\sigma_{n+1} \in L(M')$, then we have $\langle q'_{\text{init}}, \natural\rangle \overset{*}{\vdash_\epsilon} \langle q_\star, \alpha_\star\natural\rangle \parallel^*_{\sigma_1} \langle q_1, \alpha_1\natural\rangle \parallel^*_{\sigma_2} \cdots \parallel^*_{\sigma_n} \langle q_n, \alpha_n\natural\rangle \vdash^*_{\sigma_{n+1}} \langle q'_f, \epsilon\rangle$.

We build an RGPDA $N = (Q \cup \{q''_{\text{init}}, q'_f\}, \Sigma \cup \{\$\}, \Gamma \cup \{\natural, \sharp\}, \Delta_N, q''_{\text{init}}, \mathcal{F}_N, \natural)$ as follows. Let $p \in Q$ and $\sigma \in \Sigma$. Since M' is finite delay, we can compute two sets $A = \{(q, \alpha, \beta) : \langle p, \alpha\rangle \parallel^*_\sigma \langle q, \beta\rangle, \beta \neq \epsilon\}$ and $B = \{(q'_f, \xi\natural) : \langle p, \xi\natural\rangle \vdash_\sigma \mathbf{c} \overset{*}{\vdash_\epsilon} \langle q'_f, \epsilon\rangle\}$. Since M' is an SDPDA, (when $A \neq \emptyset$) there are $r \in Q$ and $k \in \mathbb{Z}$ such that if $(q, \alpha, \beta) \in A$, then $q = r$ and $|\alpha| - |\beta| = k$. We add transitions to Δ_N as follows:

- Let $(r, \alpha, \beta) \in A$. Add $p \xrightarrow{\alpha\zeta \,/\, \beta\zeta} r \in \Delta_N(\sigma)$ where $\alpha\zeta \in \Gamma^*\natural^*$ and $|\alpha\zeta| = d$.
- Let $(q'_f, \xi\natural) \in B$.

  Case $A \neq \emptyset$**:** Add $p \xrightarrow{\xi\natural\natural^i \,/\, \sharp\sharp^j} r \in \Delta_N(\sigma)$ where $i = d - |\xi\natural|$ and $j = (d-k)-1$.

  Case $A = \emptyset$**:** Add $p \xrightarrow{\xi\natural\natural^i \,/\, \sharp} q'_f \in \Delta_N(\sigma)$ where $i = d - |\xi\natural|$.

This construction ensures that (i) if $p \xrightarrow{\alpha \,/\, \beta} q, p \xrightarrow{\alpha' \,/\, \beta'} q' \in \Delta_N(\sigma)$, then $q = q'$, $|\alpha| = |\alpha'|$, and $|\beta| = |\beta'|$, (ii) $\langle q_\star, \alpha_\star\natural\rangle \parallel^*_{\sigma_1\dots\sigma_n} \langle q_n, \alpha_n\natural\rangle \vdash^*_{\sigma_{n+1}} \langle q'_f, \epsilon\rangle$ in M' iff $q_\star \xrightarrow{\alpha_\star\natural^d \,/\, \sharp\sharp^c} r \in \Delta_N(\sigma_1 \dots \sigma_n\sigma_{n+1})$ for some $c \in [1..d]$ and $r \in Q$.

We define $\Delta_N(\$) \coloneqq \{q''_{\text{init}} \xrightarrow{\natural \,/\, \alpha_\star\natural^d} q_\star\}$ so that $\langle q'_{\text{init}}, \natural\rangle \vdash^*_{\sigma_1\dots\sigma_{n+1}} \langle q'_f, \epsilon\rangle$ in M' iff $q''_{\text{init}} \xrightarrow{\natural \,/\, \sharp\sharp^c} r \in \Delta_N(\sigma_1 \dots \sigma_{n+1})$ for some $c \in [1..d]$ and $r \in Q$. By defining $\mathcal{F}_N \coloneqq \{\langle p, \sharp\natural^c\rangle : p \in Q \cup \{q'_f\}, c \leq d\}$, we have $\$L(M') = L(N)$. □

We call the following condition of Theorem 3 a *uniformity condition* that is stronger than the third condition of SDPDAs:

$$\text{If } p \xrightarrow{\alpha \,/\, \beta} q,\ p \xrightarrow{\alpha' \,/\, \beta'} r \in \Delta(\sigma), \text{ then } |\alpha| = |\alpha'|,\ |\beta| = |\beta'|, \text{ and } q = r.$$

We denote RGPDAs that satisfy the uniformity condition as RGPDA+**U**.

The normalization through Theorem 3 and the uniformity condition are crucial for the proof of our key lemma, Lemma 9 of Section 4. A construction similar to Theorem 3 appears in [14, Theorem 10]. However, they normalized an SDPDA to a corresponding SDPDA that satisfies a property like the uniformity condition; thus, they did not remove ϵ-moves in their proof.

3 Decidability of *Incl*(CFL, DYCK) Revisited

Before giving a decision procedure for *Incl*(**CFL**, RGPDA+**U**), we consider the subcase *Incl*(**CFL**, **DYCK**). We rephrase the decidability proof of *Incl*(**CFL**, **DYCK**) given by Berstel and Boasson [2] as a constraint solving problem on a monoid for the Dyck languages. The argument of this section will be naturally extended to *Incl*(**CFL**, RGPDA+**U**) in the next section.

Dyck Language. Let Σ be a finite alphabet $\Sigma = \{\sigma_1, \dots, \sigma_n\}$. We use $\acute{\sigma}$ and $\grave{\sigma}$ to denote an open and a close parenthesis labelled by σ, respectively, by following [13]. We define the open parentheses $\acute{\Sigma}$ and close parentheses $\grave{\Sigma}$ obtained from Σ by $\acute{\Sigma} \coloneqq \{\acute{\sigma}_1, \dots, \acute{\sigma}_n\}$ and $\grave{\Sigma} \coloneqq \{\grave{\sigma}_1, \dots, \grave{\sigma}_n\}$. A word $w \in (\acute{\Sigma} \cup \grave{\Sigma})^*$ is a

Dyck word if w is well-matched. For example, $á à$ and $á b̂ b̀ à$ are Dyck words, but $à á$ and $á b̀$ are not. To formally define this, we build a monoid and homomorphism that recognize the set of Dyck words.

We define a function $\mu : \acute{\Sigma} \cup \grave{\Sigma} \to \mathbb{T}_*$ where $\mathbb{T}_* = \{*\} \times \Sigma^* \times \Sigma^* \times \{*\}$ by interpreting a open parenthesis $\acute{\sigma}$ and close parenthesis $\grave{\sigma}$ as transitions of push σ and pop σ: $\mu(\acute{\sigma}) \coloneqq * \xrightarrow{\epsilon\,/\,\sigma} *$ and $\mu(\grave{\sigma}) \coloneqq * \xrightarrow{\sigma\,/\,\epsilon} *$. For the sake of readability, we write α/β to denote $* \xrightarrow{\alpha\,/\,\beta} *$. The triple $\mathbf{D} = (\mathbb{T}_* \cup \{\bot\}, \odot, \mathbf{1} = \epsilon/\epsilon)$ forms a monoid because $\delta \odot \epsilon/\epsilon = \epsilon/\epsilon \odot \delta = \delta$ for any $\delta \in \mathbb{T}_* \cup \bot$. We call the monoid $\mathbf{D}$ *Dyck monoid* and deal the function μ as a homomorphism $\mu : (\acute{\Sigma} \cup \grave{\Sigma})^* \to \mathbf{D}$. For example, $\mu(\epsilon) = \mu(á à) = \mu(á b̂ b̀ à) = \epsilon/\epsilon$, $\mu(à á) = a/a$, and $\mu(á b̀) = \bot$.

A word $w \in (\acute{\Sigma} \cup \grave{\Sigma})^*$ is a Dyck word if $\mu(w) = \epsilon/\epsilon$; thus the Dyck language over Σ is defined as $\mathbf{Dyck}(\Sigma) \coloneqq \mu^{-1}(\{\epsilon/\epsilon\})$.

Inclusion Problem as Constraint Solving. We fix a CFG $G = (V, \acute{\Sigma} \cup \grave{\Sigma}, P, S)$ and provide a procedure to decide whether $L(G) \subseteq \mathbf{Dyck}(\Sigma)$. For this purpose, we consider the equivalent monoid inclusion $\mu(L(G)) \subseteq \{\epsilon/\epsilon\}$ that is obtained from $\mathbf{Dyck}(\Sigma) = \mu^{-1}(\{\epsilon/\epsilon\})$. We introduce a notation to solve this.

A mapping $\varphi : V \to 2^{\mathbf{D}}$ is a *solution* if it satisfies the following constraint over the Dyck monoid $\mathbf{D}$:

$$\forall X \in V. \begin{cases} \varphi(X) \supseteq \{\mu(w)\} & \text{if } X \to w \in P, \\ \varphi(X) \supseteq \varphi(Y) \odot \varphi(Z) & \text{if } X \to YZ \in P. \end{cases}$$

If φ is a solution, then $\varphi(X) \supseteq \mu(L(X))$ holds for all variable X. Thus, it suffices to search a solution φ such that $\varphi(S) \subseteq \{\epsilon/\epsilon\}$ to solve $\mu(L(G)) \subseteq \{\epsilon/\epsilon\}$.

Proposition 4. *$\mu(L(G)) \subseteq \{\,\epsilon/\epsilon\,\}$ if and only if $\exists(\varphi : \textit{solution}).\ \varphi(S) \subseteq \{\,\epsilon/\epsilon\,\}$.*

By this proposition, if we find a solution φ satisfying $\varphi(S) \subseteq \{\,\epsilon/\epsilon\,\}$, we can decide whether $L(G) \subseteq \mathbf{Dyck}(\Sigma)$. Unfortunately, however, we cannot find such a solution φ directly in general because the Dyck monoid $\mathbf{D}$ is infinite and thus the set of mappings is infinite.

A Procedure for $\mathit{Incl}(\mathbf{Cfl}, \mathbf{Dyck})$. In order to limit the space in which we explore solutions, we rephrase a result of Berstel and Boasson [2], which shows the decidability of $\mathit{Incl}(\mathbf{Cfl}, \mathbf{Dyck})$, in our framework.

For each variable X, there exists a pair of terminal strings $w_1, w_2 \in (\acute{\Sigma} \cup \grave{\Sigma})^*$ such that $S \Rightarrow^* w_1 X w_2$ because each variable of CFGs is reachable. We define $\mathcal{K}(X) \coloneqq (w_1, w_2)$ by taking such a pair of terminal strings for each variable. Note that our argument does not depend on the choice of terminal strings. The pair of terminal strings $\mathcal{K}(X)$ serves as an upper-bound for $\mu(L(X))$ as follows.

Lemma 5 ([2]). *Let $\mathcal{K}(X) = (w_1, w_2)$ and $w \in L(X)$. If the following holds, then $w_1 w w_2 \in L(G) \setminus \mathbf{Dyck}(\Sigma)$:*

$$\mu(w) = \bot \quad \textit{or} \quad \mu(w) = \alpha/\beta \textit{ such that } |\alpha| > |w_1| \textit{ or } |\beta| > |w_2|.$$

Proof. We use the property $\mu(w_1w_2w_3) = \mu(\mu(w_1)\mu(w_2)\mu(w_3))$. By this property, if $\mu(w_1) = \bot$, $\mu(w) = \bot$, or $\mu(w_2) = \bot$, then $\mu(w_1ww_2) \neq \epsilon/\epsilon$.

By the definition of the operator $\odot$, if $\alpha_1/\beta_1 \odot \alpha_2/\beta_2 = \alpha_3/\beta_3$, then $|\alpha_3| \geq |\alpha_1|$ and $|\beta_3| \geq |\beta_2|$. Thus, we can assume $\mu(w_1) = \epsilon/\sigma_1 \dots \sigma_n$ and $\mu(w_2) = \sigma_1 \dots \sigma_m/\epsilon$ where $n, m \geq 0$. If not, we have $\mu(w_1ww_2) \neq \epsilon/\epsilon$. Moreover, by the definition of μ, we have $|w_1| \geq n$ and $|w_2| \geq m$.

If $|\alpha| > |w_1| \geq n$, then $\mu(w_1w) = \sigma'_1 \dots \sigma'_{|\alpha|-n}/\beta$ or $\mu(w_1w) = \bot$; thus, $\mu(w_1ww_2) \neq \epsilon/\epsilon$. Similarly, if $|\beta| > |w_2| \geq m$, then $\mu(ww_2) = \alpha/\sigma'_1 \dots \sigma'_{|\beta|-m}$ or $\mu(ww_2) = \bot$; thus, $\mu(w_1ww_2) \neq \epsilon/\epsilon$. These arguments complete the proof. □

Lemma 5 states that the space in which we explore solutions is finitely bounded and this leads to the decidability of $\mathit{Incl}(\mathbf{CFL}, \mathbf{DYCK})$. To formally state this argument, we introduce *bounded* mappings.

A mapping φ is *bounded by* $\mathcal{K}$ if it satisfies the following property:

$$\forall X \in V.\ \forall\, \alpha/\beta \in \varphi(X).\ |\alpha| \leq |w_1| \text{ and } |\beta| \leq |w_2| \text{ where } \mathcal{K}(X) = (w_1, w_2).$$

We can solve the inclusion problem by searching an adequate bounded solution.

Theorem 6.

$$\begin{aligned} \mu(L(G)) \subseteq \{\, \epsilon/\epsilon \,\} &\iff \exists(\varphi : \textit{solution}).\ \varphi(S) \subseteq \{\, \epsilon/\epsilon \,\} \\ &\iff \exists(\varphi' : \textit{bounded solution}).\ \varphi'(S) \subseteq \{\, \epsilon/\epsilon \,\}. \end{aligned}$$

Proof. It suffices to show that if a solution φ satisfies $\varphi(S) \subseteq \{\, \epsilon/\epsilon \,\}$, then φ must be bounded. If not, then there exists $w \in L(X)$ such that $|w_1| > |\alpha|$ or $|w_2| > |\beta|$ where $\mu(w) = \alpha/\beta$ and $\mathcal{K}(X) = (w_1, w_2)$. By Lemma 5, $w_1ww_2 \notin \mathbf{DYCK}(\Sigma)$ and $L(G) \not\subseteq \mathbf{DYCK}(\Sigma)$. However, the presence of φ implies $L(G) \subseteq \mathbf{DYCK}(\Sigma)$. □

Proposition 7. *The set of bounded mappings* $\{\varphi : \varphi \textit{ is bounded by } \mathcal{K}\}$ *is finite.*

Since we can decide whether a given bounded mapping is a solution, Theorem 6 and Proposition 7 imply the following result.

Corollary 8 ([2]). *The inclusion problem* $\mathit{Incl}(\mathbf{CFL}, \mathbf{DYCK})$ *is decidable.*

4 Decidability of $\mathit{Incl}(\mathbf{CFL}, \text{RGPDA+U})$

On the basis of the argument of the previous section, this section provides a procedure to decide whether $L(G) \subseteq L(M)$ where $G = (V, \Sigma, P, S)$ is a CFG and $M = (Q, \Sigma, \Gamma, \Delta, q_{\text{init}}, \mathcal{F}, Z)$ is an RGPDA+U.

A Property Corresponding to Lemma 5. We show a crucial property that corresponds to Lemma 5 and limits a space of mappings in which we explore a solution. To state it formally, we use three constants PUSH, POP, and H where

$$\forall \sigma \in \Sigma.\ \forall p \xrightarrow{\alpha\ /\ \beta} q \in \Delta(\sigma).\ \big(\textsc{Push} \geq |\beta| \ \wedge\ \textsc{Pop} \geq |\alpha|\big); \quad \forall \langle q_f, \xi \rangle \in \mathcal{F}.\ H \geq |\xi|.$$

PUSH and POP are upper bounds of the numbers of symbols that are pushed onto or popped from a stack in a single move. H is an upper bound of the heights of final configurations.

Lemma 9. *Assume that* $S \Rightarrow^* w_1 X w_2$, $q_{init} \xrightarrow{\alpha\,/\,\beta} p \in \widetilde{\Delta}(w_1)$, *and* $w \in L(X)$. *If the following holds, then* $w_1 w w_2 \in L(G) \setminus L(M)$*:*

There is $p \xrightarrow{\alpha'\,/\,\beta'} q \in \widetilde{\Delta}(w)$ *such that* $|\alpha'| > \textsc{Push}\cdot|w_1|$ *or* $|\beta'| > \textsc{Pop}\cdot|w_2| + H$.

Proof. If M fails to consume w_1, then it means $w_1 w w_2 \notin L(G)$. Hence, we assume M succeeds on consuming w_1 and have $\langle q_{\text{init}}, Z\rangle \overset{*}{\underset{w_1}{\vdash}} \langle p, \xi\rangle$ where $|\xi| = |\beta| \leq \textsc{Push}\cdot|w_1|$ by the uniformity condition of M.

First, we consider the case $|\alpha'| > \textsc{Push}\cdot|w_1| \geq |\xi|$. By the uniformity condition, M have to pop just $|\alpha'|$-symbols from the stack while reading w. However, after consuming w_1, i.e., $\langle q_{\text{init}}, Z\rangle \overset{*}{\underset{w_1}{\vdash}} \langle p, \xi\rangle$, we have only $|\xi|$-symbols on its stack. Thus, M cannot pop $|\alpha'|$-symbols and $w_1 w w_2 \notin L(M)$.

Next, we consider the case $|\beta'| > \textsc{Pop}\cdot|w_2| + H$. We assume that M succeeds on consuming $w_1 w$ and $\langle q_{\text{init}}, Z\rangle \overset{*}{\underset{w_1 w}{\vdash}} \langle q, \zeta\rangle$ where $|\zeta| = |\beta| - |\alpha'| + |\beta'|$ and $|\beta| \geq |\alpha'|$. If M succeeds on consuming w_2 from $\langle q, \zeta\rangle$, then M pops at most ($\textsc{Pop}\cdot|w_2|$)-symbols: $\langle q_{\text{init}}, Z\rangle \overset{*}{\underset{w_1 w w_2}{\vdash}} \langle r, \zeta'\rangle$ where $|\zeta'| \geq |\zeta| - \textsc{Pop}\cdot|w_2|$. Furthermore, we have $\zeta' > H$ because $|\zeta| \geq |\beta'| > \textsc{Pop}\cdot|w_2| + H$, and so $w_1 w w_2 \notin L(M)$. □

This lemma provides upper-bounds for states p and variables X if there exists a pair of terminal strings (w_1, w_2) such that $S \Rightarrow^* w_1 X w_2$ and $q_{\text{init}} \xrightarrow{\alpha\,/\,\beta} p \in \widetilde{\Delta}(w_1)$; in other words, we need a witness (w_1, w_2) to use this lemma. However, unfortunately, the following proposition says that we cannot compute such pairs (w_1, w_2) in general; thus, we cannot use this lemma directly for $\mathit{Incl}(\textbf{CFL}, \text{RGPDA}+\textbf{U})$.

Proposition 10. *Let* p *be a state and* X *be a variable. It is unsolvable to decide if there is a pair* (w_1, w_2) *such that* $S \Rightarrow^* w_1 X w_2$ *and* $q_{init} \xrightarrow{\alpha\,/\,\beta} p \in \widetilde{\Delta}(w_1)$.

To bypass this undecidability, we consider an *underlying automaton* of M where properties similar to Lemma 9 and Proposition 10 hold.

Underlying Automaton. We obtain the underlying automaton of M by forgetting the stack contents from the transition rules of M, i.e., a rule $p \xrightarrow{\alpha\,/\,\beta} q \in \Delta(\sigma)$ is translated to $p \xrightarrow{\sigma} q$ in the underlying automaton. The underlying automaton is a pair $\mathcal{A}_M = (Q, E)$ where Q is the set of *states* of M and $E \subseteq Q \times \Sigma \times Q$ has an edge $p \xrightarrow{\sigma} q$ if $p \xrightarrow{\alpha\,/\,\beta} q \in \Delta(\sigma)$ for some $\alpha, \beta \in \Gamma^*$. As the usual notation of finite automata, we write $p \xrightarrow{w} q$ if $w = \sigma_1 \sigma_2 \ldots \sigma_n$ and $p = p_0 \xrightarrow{\sigma_1} p_1 \xrightarrow{\sigma_2} p_2 \xrightarrow{\sigma_3} \cdots \xrightarrow{\sigma_n} p_n = q$. By the uniformity condition of M, $\mathcal{A}_M$ is deterministic.

A state q is *quasi-reachable* to a variable X if there exists a pair (w_1, w_2) such that $S \Rightarrow^* w_1 X w_2$ and $q_{\text{init}} \xrightarrow{w_1} q$. The next proposition says that we can determine if a given state q is quasi-reachable to X, and it means that Proposition 10 is solvable in the underlying automaton. A similar construction to consider underlying automata appears in [14, Theorem 8]. It seems that in [14] they also adopted such a construction to avoid the undecidable result of Proposition 10.

Proposition 11. *There is a partial function* $\mathcal{K} : Q \times V \rightharpoonup \Sigma^* \times \Sigma^*$ *such that:*

– *If* $\mathcal{K}(p, X) = (w_1, w_2)$*, then* $S \Rightarrow^* w_1 X w_2$ *and* $q_{init} \xrightarrow{w_1} p$ *in* $\mathcal{A}_M$*.*
– *If* $\mathcal{K}(p, X)$ *is* undefined*, then* p *is not quasi-reachable to* X*.*

Proof. For a variable X, the language $L_X = \{w_1 \# w_2 : S \Rightarrow^* w_1 X w_2, w_i \in \Sigma^*\}$ is context-free. For a state p, the language $L_p = \{w \# w' : q_{\text{init}} \xrightarrow{w} p,\ w' \in \Sigma^*\}$ is regular. Hence, the intersection $L_X \cap L_p$ is a context-free language. By the decidability of the emptiness problem of context-free languages, we can decide whether p is quasi-reachable to X and compute a witness (w_1, w_2). □

Furthermore, the property corresponding to Lemma 9 holds.

Lemma 12. *Assume that* $\mathcal{K}(p, X) = (w_1, w_2)$ *and* $w \in L(X)$*. If the following holds, then* $w_1 w w_2 \in L(G) \setminus L(M)$*:*

$$\text{There is } p \xrightarrow{\alpha' \,/\, \beta'} q \in \widetilde{\Delta}(w) \text{ such that } |\alpha'| > \textsc{Push}\cdot|w_1| \text{ or } |\beta'| > \textsc{Pop}\cdot|w_2| + H.$$

A Procedure for $\mathit{Incl}(\mathbf{CFL}, \mathbf{rgPDA{+}U})$. Although Lemma 12 constrains the pairs of a state p and variable X where p is quasi-reachable to X, this lemma does not constrain non quasi-reachable states. To avoid this problem, we redefine bounded mappings whose codomain is bounded by $\mathcal{K}$.

For a mapping $\psi : V \to 2^{\mathbf{T}_M}$, we define the *restriction* $\psi \restriction \mathcal{K} : V \to 2^{\mathbf{T}_M}$:

$$(\psi \restriction \mathcal{K})(X) \coloneqq \{T \cap \mathcal{K}_X : T \in \psi(X)\}, \quad \mathcal{K}_X \coloneqq \{\, p \xrightarrow{\alpha \,/\, \beta} q : \mathcal{K}(p, X) \text{ is defined}\}.$$

A mapping ψ is *bound by* $\mathcal{K}$ if $\psi = \psi \restriction \mathcal{K}$ and ψ satisfies the following:

$$\forall X \in V. \forall T \in \psi(X). \forall p \xrightarrow{\alpha \,/\, \beta} q \in T. \left[\begin{array}{c} |\alpha| \le \textsc{Push} \cdot |w_1| \ \wedge\ |\beta| \le \textsc{Pop} \cdot |w_2| + H \\ \text{where } \mathcal{K}(p, X) = (w_1, w_2). \end{array} \right]$$

Proposition 13. *The set of bounded mappings* $\{\psi : \psi \text{ is bound by } \mathcal{K}\}$ *is finite.*

A mapping ψ is a *solution* if it satisfies the following constraint over the transition monoid $\mathbf{T}_M$ of M:

$$\forall X \in V. \begin{cases} \psi(X) \sqsupseteq \{\widetilde{\Delta}(w)\} & \text{if } X \to w \in P, \\ \psi(X) \sqsupseteq \psi(Y) \otimes \psi(Z) & \text{if } X \to YZ \in P. \end{cases}$$

where $\mathcal{F}_1 \sqsupseteq \mathcal{F}_2$ if $\forall T_2 \in \mathcal{F}_2.\ \exists T_1 \in \mathcal{F}_1.\ T_1 \subseteq T_2$. If ψ is a solution, then $\psi(X) \sqsupseteq \widetilde{\Delta}(L(X))$ for all variable X.

The reason why we redefine solutions is to establish the following property. Indeed, even if $\psi(X) \supseteq \widetilde{\Delta}(L(X))$, then $(\psi \restriction \mathcal{K})(X) \not\supseteq \widetilde{\Delta}(L(X))$ in general.

Proposition 14. *If a mapping* ψ *is a solution then* $\psi \restriction \mathcal{K}$ *is also a solution.*

This proposition is crucial to obtain our main theorem.

Theorem 15. $L(G) \subseteq L(M) \iff$
$\exists(\psi : \textit{solution}).\ \psi(S) \subseteq \{T : q_{init} \xrightarrow{Z \,/\, \xi} q_f \in T, \langle q_f, \xi\rangle \in \mathcal{F}\} \iff$
$\exists(\psi' : \textit{bounded solution.}).\ \psi'(S) \subseteq \{T : q_{init} \xrightarrow{Z \,/\, \xi} q_f \in T, \langle q_f, \xi\rangle \in \mathcal{F}\}.$

As with $\mathit{Incl}(\mathbf{CFL}, \mathbf{Dyck})$, to decide if $L(G) \subseteq L(M)$, it suffices to explore a solution in the finite set of bounded mappings. This implies our main result.

Corollary 16. *The inclusion problem* $\mathit{Incl}(\mathbf{CFL}, \text{rgPDA+U})$ *is decidable.*

5 Attempt at Generalization or Undecidability

We have given a decidability proof of $\mathit{Incl}(\mathbf{CFL}, \mathbf{SDPDA})$ where each problem is of the form $L(G) \subseteq L(M)$. One possible generalization is to consider the inclusion of the form $L(G) \subseteq L'(M)$ where $L'(M)$ is the language defined by $L'(M) \coloneqq \{w : \langle q_{\mathrm{init}}, Z\rangle \vdash_w^* \langle q_f, \alpha\rangle, \langle q_f, \epsilon\rangle \in \mathcal{F}_M, \alpha \in \Gamma^*\}$. However, as Friedman and Greibach showed in [5], the above problem becomes undecidable. Indeed, the empty-stack acceptance condition is crucial in the proof of Lemma 9.

The second attempt is to replace the third condition of SDPDAs by the condition: if $\langle p, \alpha\rangle \Vdash_\sigma^* \langle q, \beta\rangle$ and $\langle p, \alpha'\rangle \Vdash_\sigma^* \langle r, \beta'\rangle$, then $q = r$ but maybe $|\alpha| - |\beta| \neq |\alpha'| - |\beta'|$. The relaxed condition enables us to simulate simple machines. A realtime DPDA M is a *simple machine* if M has only one state and accepts words by the empty-stack. Since simple machines are realtime DPDAs that satisfy the above relaxed condition, the undecidability of the inclusion problem on simple machines [4] implies the undecidability of the inclusion problem between CFGs and the relaxed SDPDAs. A similar argument shows that the inclusion problem also becomes undecidable if we adopt the following condition: if $\langle p, \alpha\rangle \Vdash_\sigma^* \langle q, \beta\rangle$ and $\langle p, \alpha'\rangle \Vdash_\sigma^* \langle r, \beta'\rangle$, then $|\alpha| - |\beta| = |\alpha'| - |\beta'|$ but maybe $q \neq r$.

These results illustrate the difficulty of finding a class of languages $\mathcal{B}$ that makes $\mathit{Incl}(\mathbf{CFL}, \mathcal{B})$ decidable with $\mathbf{SDPDA} \subsetneq \mathcal{B}$. Now we consider finding a class $\mathcal{A}$ such that $\mathbf{CFL} \subsetneq \mathcal{A}$ and $\mathit{Incl}(\mathcal{A}, \mathbf{SDPDA})$ is decidable. For this purpose, we consider the class **IL** of indexed languages [1, 7]. This class and higher-order indexed languages [10] is known as a natural generalization of **CFL** and have received attention for higher-order program verification [8]. If we could solve $\mathit{Incl}(\mathbf{IL}, \mathbf{SDPDA})$, then this becomes a base for program verification. However, unfortunately, the inclusion problem on **IL** is unsolvable for $\mathit{Incl}(\mathbf{IL}, \mathbf{DYCK})$.

An *indexed grammar* is a 5-tuple $G = (V, \Sigma, F, S, P)$ where $V = \{A, B, \ldots\}$ is a finite set of *variables*, $S \in V$ is the *start variable*, $\Sigma = \{\sigma_1, \sigma_2, \ldots\}$ is a finite set of *terminal symbols*, and $F = \{f, g, \ldots\}$ is a finite set of *indices*, and P is a finite set of *production rules* [1, 7]. Each rule in P is one of the following: $A \to BC$, $A_f \to B$, $A \to B_f$, $A \to \sigma$ where $A, B, C \in V$, $f \in F$, and $\sigma \in \Sigma$. A *sentential form* ψ is of the form $\psi = \alpha_1(A_1, x_1)\alpha_2(A_2, x_2)\ldots\alpha_n(A_n, x_n)\alpha_{n+1} \in (\Sigma \cup (V \times F^*))^*$ where $\alpha_i \in \Sigma^*$ and $(A_i, x_i) \in V \times F^*$. Instead of $(A, x) \in V \times F^*$, we write A_x. Each production rule rewrites a sentential form as follows: (1) a rule $A \to BC$ rewrites as $\psi_1 A_x \psi_2 \Rightarrow \psi_1 B_x C_x \psi_2$, (2) a rule $A_f \to B$ rewrites as $\psi_1 A_{fx} \psi_2 \Rightarrow \psi_1 B_x \psi_2$, (3) a rule $A \to B_f$ rewrites as $\psi_1 A_x \psi_2 \Rightarrow \psi_1 B_{fx} \psi_2$, and (4) a rule $A \to \sigma$ rewrites as $\psi_1 A_x \psi_2 \Rightarrow \psi_1 \sigma \psi_2$. The language of an indexed grammar G is defined by $L(G) \coloneqq \{w \in \Sigma^* : S \Rightarrow^* w\}$. The class of indexed languages **IL** is the languages generated by indexed grammars.

To obtain the undecidability of $\mathit{Incl}(\mathbf{IL}, \mathbf{DYCK})$, we use an undecidability result of DT0L-systems. A *DT0L-system* is a tuple $G = (\Sigma, g_1, \ldots, g_n, \alpha)$ where Σ is a finite *alphabet*, $g_i : \Sigma^* \to \Sigma^*$ is a *homomorphism* for each $i \in [1..n]$, and the non-empty word $\alpha \in \Sigma^+$ is the *axiom* of G [12]. On a DT0L G, we define the function $\mathcal{F}_G : [1..n]^* \to \Sigma^*$ recursively: $\mathcal{F}_G(\epsilon) \coloneqq \alpha$ and $\mathcal{F}_G(i_1 \ldots i_{n-1} i_n) \coloneqq g_{i_n}(\mathcal{F}_G(i_1 \ldots i_{n-1}))$. The following theorem is the immediate consequence of Theorem II.12.1 and III.7.1 of [12] and is key to showing our undecidability result.

Theorem 17 ([12]). *Let $G = (\Sigma, g_1, \ldots, g_n, \alpha)$ and $H = (\Sigma', h_1, \ldots, h_n, \beta)$ be DT0L-systems with n-homomorphisms. It is unsolvable to decide whether $|\mathcal{F}_G(w)| \geq |\mathcal{F}_H(w)|$ for all $w \in [1..n]^*$.*

Theorem 18. *Let L be an indexed language over $\{\acute{a}, \grave{a}\}$. It is unsolvable to decide whether $L \subseteq \mathbf{DYCK}(\{a\})$.*

Proof (Sketch). Let G and H be DT0L-systems with n-homomorphisms. On the basis of the construction of [7, Theorem 14.8], we can encode G and H into an indexed grammar I as $L(I) = \{\, \acute{a}^i \grave{a}^j \acute{a}^j \grave{a}^i : w \in [1..n]^*, i = |\mathcal{F}_G(w)|, j = |\mathcal{F}_H(w)| \,\}$.

Since $i \geq j \iff \acute{a}^i\, \grave{a}^j\, \acute{a}^j\, \grave{a}^i \in \mathbf{DYCK}(\{a\})$, we have the following:

$$L(I) \subseteq \mathbf{DYCK}(\{a\}) \iff |\mathcal{F}_G(w)| \geq |\mathcal{F}_H(w)| \text{ for all } w \in [1..n]^*.$$

This property and Theorem 17 imply the undecidability of $\mathit{Incl}(\mathbf{IL}, \mathbf{DYCK})$. □

6 Related Work

The idea of using monoids to solve inclusion problems follows previous work by the second author [11], which restated the decidability of the inclusion problems $\mathit{Incl}(\mathbf{CFL}, \mathbf{REG})$ and $\mathit{Incl}(\mathbf{CFL}, \mathbf{DYCK})$. We have extended the previous approach for $\mathit{Incl}(\mathbf{CFL}, \mathbf{SDPDA})$ by introducing RGPDAs and using a transition monoid of RGPDAs to accommodate the technique of Berstel and Boasson. In the paper [3], Bertoni et al. also used a monoid to solve the inclusion problem $\mathit{Incl}(\mathbf{CFL}, \mathbf{DYCK})$. Unfortunately, their proof is incorrect because they depended heavily on the incorrect assumption that Dyck monoids are *cancellative*. A monoid M is *cancellative* if $xy = xz$ implies $y = z$ and $yx = zx$ implies $y = z$ for every $x, y, z \in M$. However, the Dyck monoid over $\{a\}$ is not cancellative, i.e., $\mu(\grave{a}^3\acute{a}^3)\mu(\grave{a}^2\acute{a}^2) = a^3/a^3 = \mu(\grave{a}^3\acute{a}^3)\mu(\grave{a}^1\acute{a}^1)$ but $\mu(\grave{a}^2\acute{a}^2) \neq \mu(\grave{a}^1\acute{a}^1)$.

Tsukada and Kobayashi showed the decidability of $\mathit{Incl}(\mathbf{CFL}, \mathbf{SDPDA})$ by designing a type system for DPDAs [14]. They translated an inclusion problem $L(G) \subseteq L(M)$ for a CFG G and an SDPDA M into the type-theoretical problem deciding if G is typable under a type system obtained from M. In their type system, a typed term is of the form $w : \tau$ where a type τ is a set of pairs of configurations $\tau = \{\mathbf{c} \to \mathbf{c}' : \mathbf{c}' \overset{*}{\underset{w}{\vdash}} \mathbf{c}\}$. Since each type and each type environment are infinite in general, Tsukada and Kobayashi required extra notations to represent infinite objects in a finite form; this makes their proof elaborated overall. Conversely, as mentioned in Section 2, we consider the monoids obtained from transition rules of RGPDAs and thus the set $\widetilde{\Delta}(w)$ is finite for any word w. It is worthy to note that the definition of reading configurations of their paper differs from Greibach and Friedman [6] and us. A configuration $\mathbf{c}$ is a reading configuration if $\mathbf{c}$ is not expandable by ϵ-moves in their definitions; thus, each configuration with the empty stack $\langle q, \epsilon \rangle$ becomes a reading configuration in [14]. This difference is significant and their main theorems [14, Theorem 8 and 10] do not hold in their definition; however, it seems that their proofs and result hold in the original definition of reading configurations considered by Greibach and Friedman.

7 Conclusion and Future Work

We have extended the decidability proof of $Incl(\mathbf{Cfl}, \mathbf{Dyck})$ given by Berstel and Boasson to $Incl(\mathbf{Cfl}, \mathbf{Sdpda})$ by introducing rgPDAs and considering their monoid. **Sdpda** strictly includes the class of languages accepted by visibly pushdown automata with empty stack. Recently, the class of languages accepted by Floyd automata (operator precedence languages) [9] have received attention because it includes the class of visibly pushdown languages and enjoys many closure properties. To the best of our knowledge, the relationship between SdPDAs and Floyd automata with empty stack and the decidability of the inclusion problem between CFGs and them have not been studied to date. We would like to tackle these problems by extending the arguments presented in this paper.

References

1. Aho, A.V.: Indexed grammars—an extension of context-free grammars. J. ACM 15(4), 647–671 (1968)
2. Berstel, J., Boasson, L.: Formal properties of XML grammars and languages. Acta Informatica 38(9), 649–671 (2002)
3. Bertoni, A., Choffrut, C., Radicioni, R.: The inclusion problem of context-free languages: Some tractable cases. In: DLT'09, LNCS, vol. 5583, pp. 103–112. Springer (2009)
4. Friedman, E.P.: The inclusion problem for simple languages. TCS 1, 297–316 (1976)
5. Friedman, E.P., Greibach, S.A.: Superdeterministic DPDAs: The method of accepting does affect decision problems. JCSS 19(1), 79–117 (1979)
6. Greibach, S.A., Friedman, E.P.: Superdeterministic PDAs: A subcase with a decidable inclusion problem. J. ACM 27(4), 675–700 (1980)
7. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (1979)
8. Kobayashi, N.: Types and higher-order recursion schemes for verification of higher-order programs. In: POPL. pp. 416–428. ACM (2009)
9. Lonati, V., Mandrioli, D., Panella, F., Pradella, M.: Operator precedence languages: Their automata-theoretic and logic characterization. SIAM Journal on Computing 44(4), 1026–1088 (2015)
10. Maslov, A.N.: Multilevel stack automata. Prob. Inf. Trans. 12, 38–43 (1976)
11. Minamide, Y.: Verified decision procedures on context-free grammars. In: TPHOLs'07, LNCS, vol. 4732, pp. 173–188. Springer (2007)
12. Salomaa, A., Soittola, M.: Automata-Theoretic Aspects of Formal Power Series. Springer (1978)
13. Takahashi, M.: Generalizations of regular sets and their application to a study of context-free languages. Information and Control 27(1), 1–36 (1975)
14. Tsukada, T., Kobayashi, N.: An intersection type system for deterministic pushdown automata. In: TCS 2012, LNCS, vol. 7604, pp. 357–371. Springer (2012)