An Intersection Type System for Deterministic Pushdown Automata

Takeshi Tsukada[1] and Naoki Kobayashi[2]

[1] Tohoku University
[2] University of Tokyo

Abstract. We propose a generic method for deciding the language inclusion problem between context-free languages and deterministic context-free languages. Our method extends a given decision procedure for a subclass to another decision procedure for a more general subclass called a refinement of the former. To decide $\mathcal{L}_0 \subseteq \mathcal{L}_1$, we take two additional arguments: a language $\mathcal{L}_2$ of which $\mathcal{L}_1$ is a refinement, and a proof of $\mathcal{L}_0 \subseteq \mathcal{L}_2$. Our technique then refines the proof of $\mathcal{L}_0 \subseteq \mathcal{L}_2$ to a proof or a refutation of $\mathcal{L}_0 \subseteq \mathcal{L}_1$. Although the refinement procedure may not terminate in general, we give a sufficient condition for the termination. We employ a type-based approach to formalize the idea, inspired from Kobayashi's intersection type system for model-checking recursion schemes. To demonstrate the usefulness, we apply this method to obtain simpler proofs of the previous results of Minamide and Tozawa on the inclusion between context-free languages and regular hedge languages, and of Greibach and Friedman on the inclusion between context-free languages and superdeterministic languages.

Update

– [5th September 2016] Correction of the definition of reading mode

1 Introduction

The language inclusion problem, which asks whether $\mathcal{L}_0 \subseteq \mathcal{L}_1$ for languages $\mathcal{L}_0$ and $\mathcal{L}_1$, is a fundamental problem in the field of formal language theory. We are interested in its decidability, mainly motivated by applications to program verification [1, 7, 12]. We consider the case that $\mathcal{L}_0$ and $\mathcal{L}_1$ range over context-free languages. It is well known that the inclusion $\mathcal{L}_0 \subseteq \mathcal{L}_1$ is undecidable for context-free languages $\mathcal{L}_0$ and $\mathcal{L}_1$. For some subclasses of context-free languages, however, the inclusion is decidable [3].

In the present paper, we propose a generic method for deciding the inclusion problem. Our method extends a decision procedure for a subclass of context-free languages to another decision procedure for a more general subclass. For example, consider the languages consisting of open and close tags, like XML documents. It is known to be decidable whether a given context-free language is included in the Dyck language, which is the set of all words consisting of correctly nested tags. Using our method, we can extend this result to obtain a new proof of the decidability of inclusion between context-free languages and regular hedge languages [12].

Our method can be outlined as follows. Suppose that a decision procedure is given, which takes a language $\mathcal{L}_0$ and decides whether $\mathcal{L}_0 \subseteq \mathcal{L}_2$ for a fixed language $\mathcal{L}_2$ (in the example above, the language of all correctly nested tags). We assume that the procedure returns a "proof" of $\mathcal{L}_0 \subseteq \mathcal{L}_2$ if it is the case. By using this procedure, our method provides a way of deciding whether $\mathcal{L}_0 \subseteq \mathcal{L}_1$, where $\mathcal{L}_1$ is a subset of $\mathcal{L}_2$, called a *refinement* [19] of $\mathcal{L}_2$ (in the above example, a regular hedge language). To decide $\mathcal{L}_0 \subseteq \mathcal{L}_1$, we first decide whether $\mathcal{L}_0 \subseteq \mathcal{L}_2$, using the decision procedure. If $\mathcal{L}_0 \not\subseteq \mathcal{L}_2$, we conclude $\mathcal{L}_0 \not\subseteq \mathcal{L}_1$. If $\mathcal{L}_0 \subseteq \mathcal{L}_2$, the procedure returns a "proof" of it, and we decide the inclusion $\mathcal{L}_0 \subseteq \mathcal{L}_1$ by refining the "proof" of $\mathcal{L}_0 \subseteq \mathcal{L}_2$.

To formalize the idea, we employ a type-based approach inspired by Kobayashi's intersection type system [7] for the model checking of higher-order recursion schemes. For each deterministic context-free language $\mathcal{L}_i$, we develop a type system characterizing context-free grammars $\mathcal{G}$ such that $\mathcal{L}_\mathcal{G} \subseteq \mathcal{L}_i$, i.e., a type system $\mathcal{T}_i$ such that $\mathcal{G}$ is typable in $\mathcal{T}_i$ if and only if $\mathcal{L}_\mathcal{G} \subseteq \mathcal{L}_i$. Then, the inclusion problem $\mathcal{L}_\mathcal{G} \subseteq \mathcal{L}_i$ is reduced to the typability of $\mathcal{G}$ in $\mathcal{T}_i$. We check it by (i) first checking whether $\mathcal{G}$ is typable in a "simpler" type system $\mathcal{T}_2$, and (ii) if $\mathcal{G}$ is typable in $\mathcal{T}_1$, enumerating "refinements" of the type derivation of $\mathcal{T}_2 \vdash \mathcal{G}$ and checking whether there exists a type derivation for $\mathcal{G}$ in $\mathcal{T}_1$ among them. (We will substantiate the meaning of "simpler type system" and "refinements" in later sections.)

We demonstrate the usefulness of the method by giving simpler proofs of two previous decidability results: (1) The result of Minamide and Tozawa [12] on the inclusion between context-free languages and regular hedge languages; (2) The result of Greibach and Friedman [5] on the inclusion between context-free languages and superdeterministic languages, which is, to our knowledge, one of the strongest results about the inclusion problems.

The rest of the paper is organized as follows. In Section 2, we define some notions and notations about context-free grammars and pushdown automata. In Section 3, we construct an intersection type system characterizing the inclusion problem. In Section 4, we develop a procedure which refines a type derivation and we give a sufficient condition for the termination of the procedure. In Section 5, we apply our method to prove some decidability results. In Section 6, we discuss the related work and we conclude in Section 7.

2 Preliminaries

Context-free Grammars We present context-free grammars for words in the form of (a special case of) context-free tree grammars generating monadic trees (i.e., trees of the form $a_1(a_2(\dots(a_n(\$))\dots)))$. The definition is consistent with the standard definition of the context-free grammars.

We use a special letter \$, which can occur only at the end of a word, and distinguish between two kinds of words: those that end with \$, called *terminating words*, and those that end with a normal letter, called *normal words* (or simply, words). A *sort* κ is o describing terminating words, or $o \to o$ describing normal words. A normal word w can be considered as a function that takes a terminating word $w'\$$ and returns the terminating word $ww'\$$; that is why we assign a function sort to normal words. A *context-free grammar* (CFG, for short) is a quadruple $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R}, S)$, where:

1. $\mathcal{N}$ is a finite set of symbols called *non-terminals.* They have the sort $o \to o$. Non-terminals are ranged over by F.
2. Σ is a finite set of symbols called *terminals.* We use metavariables a and b for terminals. They also have the sort $o \to o$.
3. $\mathcal{R}$ is a set of rewriting rules of the form $F\ x \to t$, where x is a variable of the sort o and t is a term of the form $\alpha_1(\alpha_2(\dots(\alpha_n(x))\dots))$ with $\alpha_i \in \Sigma \cup \mathcal{N}$. There can be more than one rule for the same non-terminal.
4. S is a distinguished non-terminal, called the *initial symbol.*

We use t and s as metavariables of terms and α as a metavariable ranging over $\Sigma \cup \mathcal{N}$. The *rewriting relation* $\Rightarrow_{\mathcal{R}}$ is defined by:

$$F\ s \Rightarrow_{\mathcal{R}} t[s/x] \text{ if } (F\ x \to t) \in \mathcal{R} \qquad \alpha\ t \Rightarrow_{\mathcal{R}} \alpha\ t' \text{ if } t \Rightarrow_{\mathcal{R}} t'$$

Here $t[s/x]$ is the term obtained by substituting s for x in t. We write $\Rightarrow_{\mathcal{R}}^*$ for the reflexive and transitive closure of $\Rightarrow_{\mathcal{R}}$. We often omit $\mathcal{R}$ if it is clear from the context. For a given non-terminal F, we define the *language generated by* F as $\mathcal{L}_{\mathcal{G}}(F) = \{a_1 a_2 \dots a_n \in \Sigma^* \mid F\ \$ \Rightarrow^* a_1(a_2(\dots(a_n(\$))\dots))\}$. The *language generated by* $\mathcal{G}$, written $\mathcal{L}_{\mathcal{G}}$, is $\mathcal{L}_{\mathcal{G}}(S)$.

Example 1. For a given alphabet Σ, we define the set of open tags $\acute{\Sigma} = \{\acute{a} \mid a \in \Sigma\}$ and close tags $\grave{\Sigma} = \{\grave{a} \mid a \in \Sigma\}$. Let $\mathcal{G}_0 = (\{S, F_a, F_b\}, \acute{\Sigma}_0 \cup \grave{\Sigma}_0, \mathcal{R}, S)$, where $\Sigma_0 = \{\mathbf{a}, \mathbf{b}\}$ and $\mathcal{R} = \{S\,x \to x,\ S\,x \to \acute{\mathbf{a}}(F_a(x)),\ S\,x \to F_b(\grave{\mathbf{b}}(x)),\ F_a\,x \to S(\grave{\mathbf{a}}(x)),\ F_b\,x \to \acute{\mathbf{b}}(S(x))\}$. The language $\mathcal{L}_{\mathcal{G}}$ consists of words of the form $\acute{a}_1 \acute{a}_2 \dots \acute{a}_n \grave{a}_n \dots \grave{a}_1$, where $a_i \in \{\mathbf{a}, \mathbf{b}\}$ for all $1 \le i \le n$.

The rules of this CFG can be written in the standard notation as:

$$S \to \varepsilon \mid \acute{\mathbf{a}}\,F_a \mid F_b\,\grave{\mathbf{b}}, \qquad F_a \to S\,\grave{\mathbf{a}}, \qquad F_b \to \acute{\mathbf{b}}\,S,$$

where ε denotes the empty word. □

Pushdown Automaton A *pushdown automaton* (PDA, for short) is a quadruple $M = (Q, \Sigma, \Gamma, \delta)$, where (1) Q is a finite set of *states*; (2) Σ is an alphabet; (3) Γ is a finite set of *stack symbols* (we use metavariables A and B for stack symbols), and (4) $\delta \subseteq Q \times \Gamma \times (\Sigma \cup \{\varepsilon\}) \times Q \times \Gamma^*$ is a *transition relation.* We use $\widetilde{A}$ and $\widetilde{B}$ to denote (possibly empty) sequences of stack symbols. For $q \in Q$, $A \in \Gamma$ and $a \in \Sigma \cup \{\varepsilon\}$, we define $\delta(q, A, a) = \{(q', \widetilde{A}') \mid (q, A, a, q', \widetilde{A}') \in \delta\}$. A pushdown automaton is *deterministic* if for any $q \in Q$, $A \in \Gamma$ and $a \in \Sigma$, the set $\delta(q, A, a) \cup \delta(q, A, \varepsilon)$ has exactly one element. In the rest of the paper, we consider only deterministic pushdown automata.

We call an element of $Q \times \Gamma^*$ a *configuration.* If $(q, A, a, q', \widetilde{A'}) \in \delta$ (here $a \in \Sigma \cup \{\varepsilon\}$), we write $(q, \widetilde{B}A) \Vdash^a_M (q', \widetilde{B}\widetilde{A'})$. We say a configuration $c = (q, \widetilde{B}A)$ is in *reading mode* if $\bigcup_{a \in \Sigma}(q, A, a) \neq \emptyset$ (note that a configuration with the empty stack is not a reading mode).[3] For configurations c and c' in reading mode and $a \in \Sigma$, we write $c \vDash^a_M c'$ if

$$c \Vdash^a_M d_1 \Vdash^\varepsilon_M d_2 \Vdash^\varepsilon_M \cdots \Vdash^\varepsilon_M d_n \Vdash^\varepsilon_M c' \nVdash^\varepsilon_M .$$

For $w = a_1 a_2 \ldots a_n \in \Sigma^*$, we write $c \vDash^w_M c'$ if $c \vDash^{a_1}_M d_1 \vDash^{a_2}_M d_2 \vDash^{a_3}_M \cdots \vDash^{a_n}_M c'$.

For a given configuration c in reading mode and a given set $\mathcal{F}$ of configurations in reading mode, we define $\mathcal{L}_M(c, \mathcal{F}) = \{w \in \Sigma^* \mid \exists c' \in \mathcal{F}.\ c \vDash^w_M c'\}$. Here c indicates the initial configuration and $\mathcal{F}$ the set of accepting configurations.

Example 2. Recall Σ_0 and $\mathcal{G}_0$ defined in Example 1. We define $\mathcal{A}_2 = \langle \{q\}, \acute{\Sigma}_0 \cup \grave{\Sigma}_0, \{\star\}, \delta_{\mathcal{A}_2} \rangle$, where $\delta_{\mathcal{A}_2} = \{(q, \star, \acute{a}, q, \star\star), (q, \star, \grave{a}, q, \varepsilon) \mid a \in \Sigma_0\}$. The automaton $\mathcal{A}_2$ counts and records the difference between the numbers of open tags and close tags, ignoring their labels. Let $L = \mathcal{L}_{\mathcal{A}_2}((q, \star), \{(q, \star)\})$. Then L is the set of all balanced tags, e.g., $\acute{\mathbf{a}}\grave{\mathbf{b}} \in L$ but $\acute{\mathbf{a}}\grave{\mathbf{a}}\grave{\mathbf{b}}\acute{\mathbf{b}} \notin L$. It is obvious that $\mathcal{L}_{\mathcal{G}_0} \subseteq \mathcal{L}_{\mathcal{A}_2}((q, \star), \{(q, \star)\})$.

We define a different PDA $\mathcal{A}_1 = \langle \{q_1, q_2\}, \acute{\Sigma}_0 \cup \grave{\Sigma}_0, \Sigma \cup \{\bot\}, \delta_{\mathcal{A}_1} \rangle$, where $\delta_{\mathcal{A}_1} = \{(q_1, A, \acute{a}, q_1, Aa) \mid A \in \Sigma_0 \cup \{\bot\}, a \in \Sigma_0\} \cup \{(q_1, a, \grave{a}, q_2, \varepsilon), (q_2, a, \grave{a}, q_2, \varepsilon) \mid a \in \Sigma_0\}$. In addition to counting the difference of open tags and close tags, $\mathcal{A}_1$ records labels of open tags on its stack, and checks if end tags are already read, by using its state. Let $L' = \mathcal{L}_{\mathcal{A}_1}((q_1, \bot), \{(q_1, \bot), (q_2, \bot)\})$. Then L' is the set of all words of the form $\acute{a}_1 \acute{a}_2 \ldots \acute{a}_n \grave{a}_n \ldots \grave{a}_2 \grave{a}_1$, where $a_i \in \Sigma_0$. Thus $\mathcal{L}_{\mathcal{G}_0} = L'$. □

3 Type System

We construct a type system $\mathcal{T}_M$ for each PDA M which characterizes the CFGs generating languages accepted by M. In the rest of this section, we fix a PDA M and discuss the definition and properties of the type system $\mathcal{T}_M$.

The syntax of types is defined by: $\tau ::= c \mid \bigwedge \Theta \to c$, where c ranges over configurations of M in reading mode and Θ is a (possibly infinite) set of configurations in reading mode. We often abbreviate $\bigwedge\{d\} \to c$ as $d \to c$. We say a type c has the sort o (written as $c :: o$) and a type $\bigwedge \Theta \to c$ has the sort $o \to o$ (written as $\bigwedge \Theta \to c :: o \to o$). Intuitively, the type c is for terminating words accepted from c (by ignoring \$ at the end). Interpretations of $\to$ and $\bigwedge$ are standard: $d \to c$ describes functions from d to c and $c_1 \bigwedge c_2$ describes terminating words accepted from the both of c_1 and c_2. Thus a normal word $w = a_1 \ldots a_n$, which can be considered as a function $\lambda x.\, a_1(a_2(\ldots(a_n(x))\ldots))$, has a type $d \to c$ if $c \vDash^{a_1 a_2 \ldots a_n}_M d$.

[3] [5th September 2016: Correction] As discussed in [21], the previous definition of the notion of reading mode was wrong. We are grateful to Uezato and Minamide for pointing out this.

A *type environment* is a (possible infinite) set of bindings of the form $x : \tau$ or $F : \tau$. We allow multiple bindings for the same variable (or the same non-terminal), as in $\{x : \tau_1, x : \tau_2\}$. We often omit curly brackets, and simply write $x_1 : \tau_1, \ldots, x_n : \tau_n$ for $\{x_1 : \tau_1, \ldots, x_n : \tau_n\}$. We abbreviate $\{x : c \mid c \in \Theta\}$ as $x : \bigwedge \Theta$. We define $\Delta(x) = \{\tau \mid x : \tau \in \Delta\}$. A type environment Δ is *well-formed* if it respects the sort, i.e., $x : \tau \in \Delta$ implies $\tau :: o$ and $F : \tau \in \Delta$ implies $\tau :: o \to o$. We assume that all type environments appearing in the sequel are well-formed.

The typing rules are listed as follows.

$$\frac{x : \tau \in \Delta}{\Delta \vdash_M x : \tau} \qquad \frac{F : \tau \in \Delta}{\Delta \vdash_M F : \tau} \qquad \frac{\Delta \vdash_M t_1 : \bigwedge \Theta \to c \quad \Delta \vdash_M t_2 : d \text{ (for all } d \in \Theta)}{\Delta \vdash_M t_1\, t_2 : c} \qquad \frac{c \vDash^a_M c'}{\Delta \vdash_M a : c' \to c}$$

These are standard rules for intersection type systems except for the last rule for constants, which is inspired by Kobayashi's type system [7]. Types of constants depend on the transition rule of the automaton, as explained below. Assume $c \vDash^a_M c'$. Then for any (normal) word w accepted from c', aw is accepted from c. By using type-based notations, for any (terminated) word $w(\$) : c'$, we have $a(w(\$)) : c$. Thus a can be considered as a function of type $c' \to c$.

We say that a type environment Δ is an *invariant* of the rules $\mathcal{R}$, written $\Delta \vdash_M \mathcal{R}$, if $\Delta, x : \bigwedge \Theta \vdash_M t : c$ holds for all $F : \bigwedge \Theta \to c \in \Delta$ and $F\, x \to t \in \mathcal{R}$. We write $\Delta \vdash_M (\mathcal{R}, S) : \bigwedge \Theta \to c$ if $\Delta \vdash_M \mathcal{R}$ and $\Delta, \$: \bigwedge \Theta \vdash_M S\$: c$ (in the type system, \$ is treated as a variable).

Theorem 1. *Let $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R}, S)$ be a CFG, M be a PDA, c be a configuration of M and $\mathcal{F}$ be a set of configurations of M. Then $\mathcal{L}_\mathcal{G}(S) \subseteq \mathcal{L}_M(c, \mathcal{F})$ if and only if $\Delta \vdash_M (\mathcal{R}, S) : \bigwedge \mathcal{F} \to c$ for some type environment Δ.*

Proof. The "if" direction follows from the facts that typing is preserved by reductions of $S\$$, and that $\$: \bigwedge \mathcal{F} \vdash_M w\$: c$ implies $w \in \mathcal{L}_M(c, \mathcal{F})$. For the other direction, let $\Delta = \{F : \bigwedge \Theta \to d \mid \mathcal{L}_\mathcal{G}(F) \subseteq \mathcal{L}_M(d, \Theta)\}$. □

By Theorem 1, the pair of the initial configuration c and the set $\mathcal{F}$ of accepting configurations can be identified with the type $\bigwedge \mathcal{F} \to c$. We call the type $\iota = \bigwedge \mathcal{F} \to c$ the *initial type* and write $\mathcal{L}_M(\iota)$ for $\mathcal{L}_M(c, \mathcal{F})$. When $\Delta \vdash_M (\mathcal{R}, S) : \tau$, the environment Δ is called a *witness of* $\vdash_M (\mathcal{R}, S) : \tau$.

We introduce a partial order on witnesses and show the existence of the minimum witness.

Definition 1. *The* refinement ordering $\sqsubseteq$ *is the smallest partial order that satisfies: (1) $\Theta_1 \sqsubseteq \Theta_2$ if $\Theta_1 \subseteq \Theta_2$, (2) $(\bigwedge \Theta_1 \to c_1) \sqsubseteq (\bigwedge \Theta_2 \to c_2)$ if $c_1 = c_2$ and $\Theta_1 \sqsubseteq \Theta_2$, and (3) $\Delta_1 \sqsubseteq \Delta_2$ if $\Delta_1(x) \sqsubseteq \Delta_2(x)$ for every x.* □

Lemma 1. *Let $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R}, S)$ be a CFG, M be a PDA and ι be its initial type. Assume that $\mathcal{L}_\mathcal{G}(S) \subseteq \mathcal{L}_M(\iota)$. Then the set of witnesses of $\vdash_M (\mathcal{R}, S) : \iota$, i.e., $\{\Delta \mid \Delta \vdash_M (\mathcal{R}, S) : \iota\}$, has the minimum element with respect to $\sqsubseteq$.*

Proof. Let $\iota = \bigwedge \Theta \to c$. For a non-terminal F, we define $\mathrm{pre}(F) = \{w \mid S\$ \Rightarrow^*_\mathcal{R} wFv\$\}$. Let $\Delta_0 = \{F : \bigwedge \Theta' \to c' \mid \exists w \in \mathrm{pre}(F).\ c \vDash^w_M c'$ and $\Theta' = \{d' \mid \exists u \in$

$\mathcal{L}_{\mathcal{G}}(F).\ c' \models^{u}_{M} d'\}\}$. Then $\Delta_0 \vdash_M (\mathcal{R}, S) : \iota$ and Δ_0 is minimum: See Appendix A for more details. □

Example 3. Let $\mathcal{G}_0$ be the CFG defined in Example 1, $\mathcal{A}_2$ be the PDA defined in Example 2 and $\iota_2 = (q, \star) \to (q, \star)$. Since $\mathcal{L}_{\mathcal{G}_0} \subseteq \mathcal{L}_{\mathcal{A}_2}(\iota_2)$, by Theorem 1, there is Δ such that $\Delta \vdash_{\mathcal{A}_2} (\mathcal{R}, S) : \iota_2$. The minimum witnesses is given by $\{S : (q, \widetilde{A}) \to (q, \widetilde{A}), F_a : (q, \widetilde{A}) \to (q, \widetilde{A}\star), F_b : (q, \widetilde{A}\star) \to (q, \widetilde{A}) \mid \widetilde{A} \in \{\star\}^+\}$, where $\{\star\}^+$ is the set of non-empty sequences of $\star$. □

Note that a minimum type environment may be infinite as in Example 3. In the rest of this section, we develop a way to finitely describe (some of) infinite type environments.

An important property of pushdown automata is that only the top of the stack affects its transition. Especially, we can add any stack symbols to the bottom, preserving the transition. For example, let $\mathcal{A}_1$ be the automaton defined in Example 2 and $w = \acute{\mathbf{a}}\grave{\mathbf{a}}\grave{\mathbf{b}}$. Then we have a transition $(q_1, \mathbf{bbb}) \models^{w}_{\mathcal{A}_1} (q_2, \mathbf{bb})$. By adding $\perp\mathbf{aa}$ to the bottom of the stack, we obtain $(q_1, \perp\mathbf{aabbb}) \models^{w}_{\mathcal{A}_1} (q_2, \perp\mathbf{aabb})$. More generally, for any sequence $\widetilde{A}$ of stack symbols, we have $(q_1, \widetilde{A}\mathbf{bbb}) \models^{w}_{\mathcal{A}_1} (q_2, \widetilde{A}\mathbf{bb})$. This does not depend on the choice of w, i.e., for any w such that $(q_1, \mathbf{bbb}) \models^{w}_{\mathcal{A}_1} (q_2, \mathbf{bb})$, we have $(q_1, \widetilde{A}\mathbf{bbb}) \models^{w}_{\mathcal{A}_1} (q_2, \widetilde{A}\mathbf{bb})$.

We will formally state this fact in terms of intersection types (see Lemma 2).

Definition 2. *For a given (possible empty) sequence $\widetilde{B}$ of stack symbols and a given configuration $(q, \widetilde{A})$, we define the* stack extension $(q, \widetilde{A}) \Uparrow \widetilde{B}$ *as* $(q, \widetilde{B}\widetilde{A})$. *We define $(\Theta \Uparrow \widetilde{B}) = \{c \Uparrow \widetilde{B} \mid c \in \Theta\}$ for the set of configurations, $(\bigwedge \Theta \to c) \Uparrow \widetilde{B} = \bigwedge(\Theta \Uparrow \widetilde{B}) \to (c \Uparrow \widetilde{B})$ for the type, $\Delta \Uparrow \widetilde{B} = \{x : (\tau \Uparrow \widetilde{B}) \mid x : \tau \in \Delta\}$ for the type environment and $(\Delta \vdash t : \tau) \Uparrow \widetilde{B} = (\Delta \Uparrow \widetilde{B}) \vdash t : (\tau \Uparrow \widetilde{B})$ for the judgement. We define $\Delta^{\Uparrow} = \cup_{\widetilde{B}}(\Delta \Uparrow \widetilde{B})$.* □

Lemma 2. *If $\Delta \vdash_M t : \tau$, then for any $\widetilde{B}$, we have $(\Delta \vdash_M t : \tau) \Uparrow \widetilde{B}$.*

Proof. Easy induction on $\Delta \vdash_M t : \tau$. □

We write $\Delta \vdash^{\Uparrow}_{M} \mathcal{R}$, read "$\Delta$ is an invariant of $\mathcal{R}$ up-to stack extensions", if for every $F : \bigwedge \Theta \to c \in \Delta$ and $F\, x \to t \in \mathcal{R}$, we have $(\Delta^{\Uparrow}), x : \bigwedge \Theta \vdash_M t : c$. Note that while $F : \bigwedge \Theta \to c$ is chosen from Δ, the environment to type the body of F is $\Delta^{\Uparrow}$. The judgement $\Delta \vdash^{\Uparrow}_{M} (\mathcal{R}, S) : \bigwedge \Theta \to c$ is defined as $\Delta \vdash^{\Uparrow}_{M} \mathcal{R}$ and $(\Delta^{\Uparrow}), \$: \bigwedge \Theta \vdash_M S\$: c$.

By using this up-to technique, we can sometimes (but not always) finitely describe a witness type environment as shown in the example below.

Example 4. Recall Example 3. We have $\Delta \vdash^{\Uparrow}_{\mathcal{A}_2} (\mathcal{R}, S) : \iota_2$, where $\Delta = \{S : (q, \star) \to (q, \star), F_a : (q, \star) \to (q, \star\star), F_b : (q, \star\star) \to (q, \star)\}$. Note that Δ is a finite set. □

This up-to technique is sound in the sense that if a CFG is typable up-to stack expansions, then it is typable without using the up-to technique.

Theorem 2. *$\Delta \vdash^{\Uparrow}_{M} (\mathcal{R}, S) : \iota$ implies $(\Delta^{\Uparrow}) \vdash_M (\mathcal{R}, S) : \iota$.*

Proof. We should show that $(\Delta^{\Uparrow}) \vdash_M \mathcal{R}$ and $(\Delta^{\Uparrow}), \$: \bigwedge \Theta \vdash_M S\$: c$, where $\iota = \bigwedge \Theta \to c$. The latter comes from the assumption. To show the former, assume $F : \tau \in (\Delta^{\Uparrow})$ and $F\,x \to t \in \mathcal{R}$. Then we have $F : \sigma \in \Delta$ and $\tau = (\sigma \Uparrow \widetilde{A})$ for some σ and $\widetilde{A}$. Let $\sigma = \bigwedge \Xi \to d$. Then $\tau = \bigwedge(\Xi \Uparrow \widetilde{A}) \to (d \Uparrow \widetilde{A})$. We should show that $(\Delta^{\Uparrow}), (x : \bigwedge \Xi \Uparrow \widetilde{A}) \vdash_M t : (d \Uparrow \widetilde{A})$. By the assumption, $(\Delta^{\Uparrow}), x : \bigwedge \Xi \vdash_M t : d$. By the previous lemma, we have $((\Delta^{\Uparrow}) \Uparrow \widetilde{A}), (x : \bigwedge \Xi \Uparrow \widetilde{A}) \vdash_M t : (d \Uparrow \widetilde{A})$. Because $((\Delta^{\Uparrow}) \Uparrow \widetilde{A}) \subseteq (\Delta^{\Uparrow})^{\Uparrow} = \Delta^{\Uparrow}$, we conclude $(\Delta^{\Uparrow}), (x : \bigwedge \Xi \Uparrow \widetilde{A}) \vdash_M t : (d \Uparrow \widetilde{A})$. □

4 Refining Witnesses

It is in general difficult (in fact undecidable) to check whether a given CFG $\mathcal{G}$ is typable in $\mathcal{T}_{M_1}$ for a given PDA M_1, so that we first consider a simpler PDA M_2 and check whether G is typable in $\mathcal{T}_{M_2}$. If we choose M_2 so that (i) we have a witness of typability of $\mathcal{G}$ in $\mathcal{T}_{M_2}$ and (ii) M_1 is a refinement of M_2, then $\mathcal{G}$ is typable in $\mathcal{T}_{M_1}$ if and only if there is a witness that is a refinement of the witness in $\mathcal{T}_{M_2}$ (Section 4.1). Moreover, if a witness in $\mathcal{T}_{M_2}$ is finite, then the set of its refinements is a finite set. Thus, we can decide the typability in $\mathcal{T}_{M_1}$ by exhaustively searching a witness from the (finite) set of refinements of the witness in $\mathcal{T}_{M_2}$ (Section 4.2).

4.1 Refinements of Automata

We first define the notion of *refinements* of automata. As we will see below, if M_1 is a refinement of M_2, then M_2 is a good over-approximation of M_1.

Definition 3 (Refinement of Automata). *Let $M_1 = \langle Q_1, \Sigma, \Gamma_1, \delta_1 \rangle$ and $M_2 = \langle Q_2, \Sigma, \Gamma_2, \delta_2 \rangle$ be pushdown automata. A* homomorphism $f : M_1 \to M_2$ *is a pair of mappings $f^Q : Q_1 \to Q_2$ and $f^\Gamma : \Gamma_1 \to \Gamma_2$ such that for any $(q, A, a, q', \widetilde{B}) \in \delta_1$, $(f^Q(q), f^\Gamma(A), a, f^Q(q'), f^\Gamma(\widetilde{B})) \in \delta_2$, where $f^\Gamma(B_1 B_2 \dots B_n) = f^\Gamma(B_1) f^\Gamma(B_2) \dots f^\Gamma(B_n)$. We often omit superscripts Q and Γ, and simply write $f(q)$ and $f(\widetilde{A})$.* □

The homomorphism $f : M_1 \to M_2$ can be naturally extended to mappings on configurations, types, type environments and judgements, e.g., the mapping on configurations is defined by $f(\ (q, \widetilde{A})\ ) = (f^Q(q), f^\Gamma(\widetilde{A}))$.

When there is a homomorphism $f : M_1 \to M_2$, we say M_2 is an *approximation* of M_1 and M_1 is a *refinement* of M_2. A type τ_1 in $\mathcal{T}_{M_1}$ is a refinement of τ_2 in $\mathcal{T}_{M_2}$ if $f(\tau_1) \sqsubseteq \tau_2$. Refinements of type environments are defined similarly. We can always find a homomorphism $f : M_1 \to M_2$ if it exists, since both of $Q_1 \to Q_2$ and $\Gamma_1 \to \Gamma_2$ are finite. We write $f : (M_1, \iota_1) \to (M_2, \iota_2)$ if $f : M_1 \to M_2$ and $f(\iota_1) = \iota_2$. The next lemma justifies to say that M_2 is an (over-)approximation of M_1.

Lemma 3. *If $f : (M_1, \iota_1) \to (M_2, \iota_2)$, then $\mathcal{L}_{M_1}(\iota_1) \subseteq \mathcal{L}_{M_2}(\iota_2)$.* □

Example 5. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be automata defined in Example 2. Then $\mathcal{A}_1$ is a refinement of $\mathcal{A}_2$ by a homomorphism $(h^Q, h^\Gamma) : \mathcal{A}_1 \to \mathcal{A}_2$ given by $h^Q(q_1) = h^Q(q_2) = q$ and $h^\Gamma(\mathbf{a}) = h^\Gamma(\mathbf{b}) = h^\Gamma(\bot) = \star$. □

In the following, we fix two pushdown automata (with their initial types) (M_1, ι_1) and (M_2, ι_2) and a homomorphism $f : (M_1, \iota_1) \to (M_2, \iota_2)$ between them. For readability, we write $\mathcal{T}_1$ instead of $\mathcal{T}_{M_1}$, $\mathcal{L}_1$ instead of $\mathcal{L}_{M_1}$ and so on.

Validity of type judgements and minimality of a witness are preserved by f.

Theorem 3. *Let $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R}, S)$ be a CFG, M_1 and M_2 be PDAs, ι_1 and ι_2 be their initial types and $f : (M_1, \iota_1) \to (M_2, \iota_2)$ be a homomorphism.*

1. *If $\Delta \vdash_{M_1} (\mathcal{R}, F) : \iota_1$, then $f(\Delta) \vdash_{M_2} (\mathcal{R}, F) : \iota_2$.*
2. *If Δ is the minimum witness of $\vdash_{M_1} (\mathcal{R}, F) : \iota_1$, then $f(\Delta)$ is the minimum witness of $\vdash_{M_2} (\mathcal{R}, F) : \iota_2$.*

Proof. It is easy to prove that $\Delta \vdash_{M_1} t : \tau$ implies $f(\Delta) \vdash_{M_2} t : f(\tau)$ by induction on t. The first part of the claim is an easy consequence of this proposition. The second part is clear from the construction of the minimum witness in the proof of Lemma 1. □

A witness Δ_2 in $\mathcal{T}_2$ ensures the existence of a "smaller" witness in $\mathcal{T}_1$.

Theorem 4. *Let $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R}, S)$ be a CFG, M_1 and M_2 be PDAs, ι_1 and ι_2 be their initial types and $f : (M_1, \iota_1) \to (M_2, \iota_2)$ be a homomorphism. Assume that $\Delta_2 \vdash^{\Uparrow}_{M_2} (\mathcal{R}, S) : \iota_2$. If $\Delta_1 \vdash^{\Uparrow}_{M_1} (\mathcal{R}, S) : \iota_1$, then there exists Δ'_1 such that $\Delta'_1 \vdash^{\Uparrow}_{M_1} (\mathcal{R}, S) : \iota_1$ and $f(\Delta'_1) \sqsubseteq \Delta_2$.*

Proof. Here, we give a proof sketch. Since $\Delta_1 \vdash^{\Uparrow}_{M_1} (\mathcal{R}, S) : \iota_1$, there is the minimum witness type environment by Lemma 1. Let Δ^0_1 be the minimum witness of $\vdash_{M_1} (\mathcal{R}, S) : \iota_1$. Note that $f(\Delta^0_1) \sqsubseteq \Delta^{\Uparrow}_2$ by Theorem 3.

We shorten the types in Δ^0_1, appropriately. We define $(q, A_1 A_2 \dots A_m) \Downarrow n = (q, A_{n+1} \dots A_m)$ if $m > n$ (and undefined otherwise). This operation is extended to types by $(\bigwedge \Theta \to c) \Downarrow n = \bigwedge\{d \Downarrow n \mid d \in \Theta\} \to (c \Downarrow n)$. Let $(F, \tau^0_1, \tau_2, \widetilde{A}_2)$ be a quadruple such that $F : \tau^0_1 \in \Delta^0_1$, $F : \tau_2 \in \Delta_2$ and $f(\tau^0_1) \sqsubseteq (\tau_2 \Uparrow \widetilde{A}_2)$. The corresponding type binding $F : \tau'_1$ of the quadruple is defined by $\tau'_1 = \tau^0_1 \Downarrow n$, where n is the length of $\widetilde{A}_2$. Let Δ'_1 be the set of all such bindings $F : \tau'_1$. Then Δ'_1 satisfies the above conditions: See Appendix B for a more detailed proof. □

4.2 Procedure and Sufficient Condition for Termination

Recall the overall picture of our method to understand the role of the procedure developed here. The final goal is to decide whether $\mathcal{G}$ is typable in $\mathcal{T}_1$. To solve the problem, we first check whether $\mathcal{G}$ is typable in $\mathcal{T}_2$, and if so, use the derivation for $\mathcal{T}_2$ and Theorem 4 to check whether $\mathcal{G}$ is typable in $\mathcal{T}_1$. The procedure developed here takes care of this last step.

Refine$(\mathcal{G}, (M_1, \iota_1), (M_2, \iota_2), f, \Delta_2)$.

1. Let $n := 0$ and $\Delta_1^0 := \{F : \tau_1 \mid \exists \tau_2.\ F : \tau_2 \in \Delta_2 \text{ and } f(\tau_1) \sqsubseteq \tau_2\}$.
2. Compute a fixed-point Δ_1 of $\mathcal{H}$ starting from Δ_1^0 as follows:
   (a) Let $\Delta_1^{n+1} := \mathcal{H}(\Delta_1^n)$.
   (b) If $\Delta_1^n = \Delta_1^{n+1}$, then Δ_1^n is a fixed-point of $\mathcal{H}$.
   (c) Otherwise, let $n := n + 1$ and goto (a).
3. Check whether $S : \iota_1 \in \Delta_1$. If so, return Δ_1. Otherwise, return **untypable**.

Fig. 1. The procedure to refine a witness

Before describing the procedure, we define the notion of *finiteness*. We say that any base type q is *finite* and a type $\bigwedge \Theta \to c$ is finite if Θ is a finite set. A type environment Δ is finite if Δ is a finite set and for every type binding $x : \tau \in \Delta$, τ is finite.

Figure 1 shows the procedure that refines a *finite* witness in $\mathcal{T}_2$ to one in $\mathcal{T}_1$. Here for a given grammar $\mathcal{G}$ and its rewriting relation $\mathcal{R}$, the function $\mathcal{H}$ on type environments in $\mathcal{T}_1$ is defined by

$$\mathcal{H}(\Delta_1) = \{F : \bigwedge \Theta \to c \in \Delta_1 \mid \forall (F\,x \to t) \in \mathcal{R}.\ \Delta_1, x : \bigwedge \Theta \vdash^{\Uparrow}_{M_1} t : c\}.$$

The procedure takes five arguments: a grammar $\mathcal{G}$, two PDAs with the initial types (M_1, ι_1) and (M_2, ι_2), a homomorphism $f : (M_1, \iota_1) \to (M_2, \iota_2)$ and a finite type environment Δ_2 in $\mathcal{T}_2$ such that $\Delta_2 \vdash^{\Uparrow}_{M_2} (\mathcal{R}, S) : \iota_2$. The finiteness of the type environment ensures the termination of the procedure. The procedure returns a witness if it exists, and otherwise returns **untypable**.

Example 6. Let $\mathcal{G}_0$ be the CFG defined in Example 1, $\mathcal{A}_1$ and $\mathcal{A}_2$ be PDAs defined in Example 2, Δ' be the finite witness of $\vdash_{\mathcal{A}_2} (\mathcal{R}, S) : \iota_{\mathcal{A}_2}$ defined in Example 4, $f : \mathcal{A}_1 \to \mathcal{A}_2$ be the homomorphism defined in Example 5 and $\iota_{\mathcal{A}_1} = (q_1, \bot) \wedge (q_2, \bot) \to (q_1, \bot)$. We compute a witness of $\vdash_{\mathcal{A}_1} (\mathcal{R}, S) : \iota_{\mathcal{A}_1}$ by our procedure **Refine**.

The starting point Δ_1^0 for computing a fixed-point of $\mathcal{H}$ is the set of all refinements of type bindings in Δ'. For example, $\Delta_1^0(S)$ is given by

$$\left\{\begin{array}{lll}
\bigwedge\emptyset \to (q_1, \mathbf{a}), & \bigwedge\emptyset \to (q_1, \mathbf{b}), & \bigwedge\emptyset \to (q_1, \bot) \\
\bigwedge\emptyset \to (q_2, \mathbf{a}), & \bigwedge\emptyset \to (q_2, \mathbf{b}), & \bigwedge\emptyset \to (q_2, \bot) \\
(q_1, \mathbf{a}) \to (q_1, \mathbf{a}), & (q_1, \mathbf{a}) \to (q_1, \mathbf{b}), & (q_1, \mathbf{a}) \to (q_1, \bot) \\
(q_1, \mathbf{a}) \to (q_2, \mathbf{a}), & (q_1, \mathbf{a}) \to (q_2, \mathbf{b}), & (q_1, \mathbf{a}) \to (q_2, \bot) \\
(q_1, \mathbf{b}) \to (q_1, \mathbf{a}), & (q_1, \mathbf{b}) \to (q_1, \mathbf{b}), & (q_1, \mathbf{b}) \to (q_1, \bot) \\
\vdots & & \\
(q_1, \mathbf{a}) \wedge (q_1, \mathbf{b}) \to (q_1, \mathbf{a}), & \cdots & \\
(q_1, \mathbf{a}) \wedge (q_1, \mathbf{b}) \wedge (q_2, \mathbf{a}) \to (q_1, \mathbf{b}), & \cdots &
\end{array}\right\}$$

since $\Delta'(S) = \{(q, \star) \to (q, \star)\}$. The type $\tau = (q_1, \mathbf{a}) \to (q_2, \mathbf{ab})$ does not belong to $\Delta_1^0(S)$, since $f(\tau) = (q, \star) \to (q, \star\star) \not\sqsubseteq (q, \star) \to (q, \star)$. The set $\Delta_1^0(S)$ contains $2^6 \times 6$ elements, because there are 6 refinements of $(q, \star)$. Similarly, $\Delta_1^0(F_a)$ contains $2^6 \times 18$ elements and $\Delta_1^0(F_b)$ contains $2^{18} \times 6$ elements.

Then we filter out wrong type bindings such as $S : \bigwedge\emptyset \to (q_1, \mathbf{b}) \in \Delta_1^0$ by iteratively applying $\mathcal{H}$. For example, $S : \bigwedge\emptyset \to (q_1, \mathbf{b}) \notin \mathcal{H}(\Delta_1^0)$ because $S\,x \to x \in \mathcal{R}$ and $\Delta_1^0, x : \bigwedge\emptyset \nvdash_{\mathcal{A}_1} x : (q_1, \mathbf{b})$.

Be repeated applications of $\mathcal{H}$, we obtain the following fixed-point:

$$\Delta_1 = \left\{ \begin{array}{ll|c} S & : \bigwedge\Big(\{(q_1, B), (q_2, B)\} \quad \cup\Theta_1\Big) \to (q_1, B) & B \in \{\mathbf{a}, \mathbf{b}, \bot\} \\ F_a & : \bigwedge\Big(\{(q_2, B)\} \quad \cup\Theta_1\Big) \to (q_1, B\mathbf{a}) & f(\Theta_1) \subseteq \{(q, \star)\} \\ F_b & : \bigwedge\Big(\{(q_1, B\mathbf{b}), (q_2, B\mathbf{b})\} \cup\Theta_2\Big) \to (q_1, B) & f(\Theta_2) \subseteq \{(q, \star\star)\} \end{array} \right\}.$$

Δ_1 is an invariant of $\mathcal{R}$ and contains $S : \iota_{\mathcal{A}_1}$. So Δ_1 is a witness and returned by **Refine**. □

We show the correctness and termination of **Refine**.

Lemma 4. *Let M_1 be a PDA. Given a finite environment Δ_1, a term t and a finite type τ, whether $\Delta_1 \vdash^{\Uparrow}_{M_1} t : \tau$ is decidable.*

Proof. Induction on the structure of t. □

Lemma 5. *Let (M_1, ι_1) and (M_2, ι_2) be PDAs with the initial symbols, $f : (M_1, \iota_1) \to (M_2, \iota_2)$ be a homomorphism and Δ_2 be a finite type environment in $\mathcal{T}_2$. Then the type environment Δ_1^0 defined in Fig. 1 is finite.*

Proof. We first show that the following two propositions hold for any finite type τ_2 by induction on τ_2: (i) for any type τ_1 in $\mathcal{T}_1$, $f(\tau_1) \sqsubseteq \tau_2$ implies finiteness of τ_1 and (ii) the set $\{\tau_1 \mid f(\tau_1) \sqsubseteq \tau_2\}$ is a finite set. Since there are finitely many type bindings in Δ_2, propositions (i) and (ii) imply finiteness of Δ_1^0. □

Theorem 5. *Let $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R}, S)$ be a CFG, (M_1, ι_1) and (M_2, ι_2) be PDAs with the initial types, $f : (M_1, \iota_1) \to (M_2, \iota_2)$ be a homomorphism and Δ_2 be a finite witness of $\vdash^{\Uparrow}_{M_2} (\mathcal{R}, S) : \iota_2$. Then* **Refine**$(\mathcal{G}, (M_1, \iota_2), (M_2, \iota_2), f, \Delta_2)$ *always terminates, and returns a witness of $\vdash^{\Uparrow}_{M_1} (\mathcal{R}, S) : \iota_1$ if and only if it exists.*

Proof. First, we show the termination of the step 2 in Figure 1. It is easy to show that Δ_1^n is a finite type environment by induction on n (for the base case, we use Lemma 5). Thus Lemma 4 implies that we can compute $\mathcal{H}(\Delta_1^n)$. Since $\mathcal{H}$ is decreasing with respect to the set inclusion ordering, i.e., $\mathcal{H}(\Delta_1) \subseteq \Delta_1$ for any environment Δ_1, and Δ_1^0 is a finite set, the fixed-point iteration must terminate. So the procedure **Refine** terminates.

Let Δ_1' be a witness of $\vdash^{\Uparrow}_{M_1} (\mathcal{R}, S) : \iota_1$. Theorem 4 ensures that we can assume without loss of generality that $f(\Delta_1') \sqsubseteq \Delta_2$. Thus $\Delta_1' \subseteq \Delta_1^0$ because Δ_1^0 is the set of all refinement type bindings. By induction on n, we have $\Delta_1' = \mathcal{H}^n(\Delta_1') \subseteq \mathcal{H}^n(\Delta_1^0) = \Delta_1^n$, since Δ_1' is a fixed-point of $\mathcal{H}$ and $\mathcal{H}$ is monotonic. So $S : \iota_1 \in \Delta_1^n$ for any n, especially $S : \iota_1 \in \Delta_1$. □

5 Applications: Some Decidability Results

5.1 Balanced Parenthesis and Regular Hedge Languages

Let Σ be an alphabet. We define a PDA $\mathcal{B} = (\{q\}, \acute{\Sigma} \cup \grave{\Sigma}, \Sigma \cup \{\bot\}, \delta)$, where $\delta = \{(q, A, \acute{a}, q, Aa) \mid A \in \Sigma \cup \{\bot\}, a \in \Sigma\} \cup \{(q, a, \grave{a}, q, \varepsilon) \mid a \in \Sigma\}$ with the initial type $\iota_{\mathcal{B}} = (q, \bot) \to (q, \bot)$. Then $\mathcal{L}_{\mathcal{B}}(\iota_{\mathcal{B}})$ is the set of all balanced tags. For example, $\acute{a}\acute{b}_1\grave{b}_1\acute{b}_2\grave{b}_2\grave{a} \in \mathcal{L}_{\mathcal{B}}(\iota_{\mathcal{B}})$ and $\acute{b}_1\grave{b}_2 \notin \mathcal{L}_{\mathcal{B}}(\iota_{\mathcal{B}})$, where $a, b_1, b_2 \in \Sigma$. It is known that, for a given CFG $\mathcal{G}$, whether $\mathcal{L}_{\mathcal{G}} \subseteq \mathcal{L}_{\mathcal{B}}$ is decidable. Moreover, if $\mathcal{L}_{\mathcal{G}} \subseteq \mathcal{L}_{\mathcal{B}}$, we can construct a finite type environment Δ such that $\Delta \vdash^{\Uparrow}_{\mathcal{B}} (\mathcal{R}, S) : \iota_{\mathcal{B}}$.

Assume that (M, ι) is a refinement of $(\mathcal{B}, \iota_{\mathcal{B}})$, i.e., there is $f : (M, \iota) \to (\mathcal{B}, \iota_{\mathcal{B}})$. Then we can decide $\mathcal{L}_{\mathcal{G}} \subseteq \mathcal{L}_M$ in the following way. First, we decide whether $\mathcal{L}_{\mathcal{G}} \subseteq \mathcal{L}_{\mathcal{B}}$. If not, then $\mathcal{L}_{\mathcal{G}} \not\subseteq \mathcal{L}_M$ by Lemma 3. If $\mathcal{L}_{\mathcal{G}} \subseteq \mathcal{L}_{\mathcal{B}}$, we construct a finite witness Δ and call **Refine**$(\mathcal{G}, (M, \iota), (\mathcal{B}, \iota_{\mathcal{B}}), f, \Delta)$.

This argument leads to the following decidability result.

Theorem 6. *Let $\mathcal{G}$ be a CFG and M be a refinement of $\mathcal{B}$. Then $\mathcal{L}_{\mathcal{G}} \subseteq \mathcal{L}_M(\iota)$ is decidable.* □

We have the following theorem for the class of refinements of $\mathcal{B}$.

Theorem 7. *A language is accepted by a refinement of $\mathcal{B}$ if and only if it is a regular hedge language [14].*

Proof. It is easy to prove using an algebraic representation of a regular hedge language, called binoid [12, 18]. □

The above argument therefore gives a new definition of the class of regular hedge languages and a new decidability proof of the inclusion problem between CFLs and regular hedge languages.

5.2 Counting Automata and Superdeterministic Languages

We define the class of PDAs named $\mathcal{C}$-machines.

Definition 4. *A PDA (M, ι_M) with the initial type is called a* $\mathcal{C}$-machine *if its stack alphabet is singleton and ι_M is finite.* □

A configuration of a $\mathcal{C}$-machine is expressed by a pair (q, n) of a state q and a natural number n representing the length of the stack sequence. We define the stack extension $\Uparrow m$ for $\mathcal{C}$-machines by $(q, n) \Uparrow m = (q, n+m)$ and $(\bigwedge \Theta \to c) \Uparrow m = \bigwedge\{d \Uparrow m \mid d \in \Theta\} \to (c \Uparrow m)$.

Theorem 8. *For a given CFG $\mathcal{G}$ and $\mathcal{C}$-machine (M, ι_M), whether $\mathcal{L}_{\mathcal{G}} \subseteq \mathcal{L}_M(\iota_M)$ is decidable. Moreover, when $\mathcal{L}_{\mathcal{G}} \subseteq \mathcal{L}_M(\iota_M)$, we can construct a finite type environment Δ such that $\Delta \vdash^{\Uparrow}_M (\mathcal{R}, S) : \iota_M$.*

Proof. We give a proof sketch: See Appendix C for more details. For simplicity, we assume that $\iota_M = c_E \to c_S$. Let $c_E = (q_E, n_E)$ and $c_S = (q_S, n_S)$. Let N be a finite-state automaton obtained by removing the counter of M, i.e., $q \vDash_N^a p$ if and only if $(q, n) \vDash_M^a (p, m)$ for some n and m. Roughly speaking, N is an "approximation" of M. So we can "refine" a witness in $\mathcal{T}_N$ to a witness in $\mathcal{T}_M$. Since N is finite-state, we can decide whether $\mathcal{L}_\mathcal{G} \subseteq \mathcal{L}_N(q_E \to q_S)$. If not, then $\mathcal{L}_\mathcal{G} \not\subseteq \mathcal{L}_M(\iota_M)$. Assume $\mathcal{L}_\mathcal{G} \subseteq \mathcal{L}_N(q_E \to q_S)$ and let Δ_N be the minimum witness of $\vdash_N (\mathcal{R}, S) : q_E \to q_S$ (here $\mathcal{T}_N$ is the type system whose base types are states of N, instead of configurations).

For a given type binding $F : \bigwedge\{q_1, \ldots, q_m\} \to q \in \Delta_N$, we construct a corresponding type binding in $\mathcal{T}_M$. Since Δ_N is minimum, from the construction of the minimum witness (see the proof of Lemma 1), we have $w \in \mathrm{pre}(F)(= \{v \mid \exists u.\ S\$ \Rightarrow^* vFu\$\})$ and $w_i \in \mathcal{L}_\mathcal{G}(F)$ $(1 \le i \le m)$ such that $q_S \vDash_N^w q$ and $q \vDash_N^{w_i} q_i$ for all i (a different choice of w and w_i gives a different upper-bound of witnesses). We define n and n_i by $(q_E, n_E) \vDash_M^w (q, n)$ and $(q, n) \vDash_M^{w_i} (q_i, n_i)$. Then the corresponding type binding is $F : \bigwedge\{(q_1, n_1), \ldots, (q_m, n_m)\} \to (q, n)$.

Let Δ'_M be the type environment collecting such type bindings. We define $\Delta_M = \{F{:}\tau \mid \exists \sigma, k.\ F{:}\sigma \in \Delta'_M \text{ and } \tau \Uparrow k = \sigma\}$. Then Δ_M gives an upper-bound in the sense that if a witness of $\vdash_M^{\Uparrow} (\mathcal{R}, S) : \iota_M$ exists, then a witness included by Δ_M exists. □

Similarly to the argument in the previous subsection, Theorem 8 leads to the following decidability result.

Theorem 9. *For a given context-free grammar $\mathcal{G}$ and a pushdown automaton M which is a refinement of a $\mathcal{C}$-machine N, whether $\mathcal{L}_\mathcal{G} \subseteq \mathcal{L}_M$ is decidable.* □

The class of refinements of $\mathcal{C}$-machines is closely related to the class of *superdeterministic pushdown automata* proposed by Greibach and Friedman [5].

Definition 5 (Superdeterministic PDAs [5]). *A pushdown automaton M is of* delay d *if for any series of one-step transitions by ε, its length is less than or equal to d, i.e., if $c_0 \Vdash_M^\varepsilon c_1 \Vdash_M^\varepsilon \cdots \Vdash_M^\varepsilon c_n$ then $n \le d$. A pushdown automaton $M(\iota)$ is* superdeterministic *if it satisfies the following properties: (1) M is of delay d for some finite number d, (2) if $(q, \widetilde{A}_1) \vDash_M^w (p_1, \widetilde{B}_1)$ and $(q, \widetilde{A}_2) \vDash_M^w (p_2, \widetilde{B}_2)$, then $p_1 = p_2$ and $|\widetilde{B}_1| - |\widetilde{A}_1| = |\widetilde{B}_2| - |\widetilde{A}_2|$, here $|\widetilde{A}|$ is the length of A, and (3) ι is finite. A language $\mathcal{L}$ is superdeterministic if $\mathcal{L} = \mathcal{L}_M$ for some superdeterministic pushdown automaton M.* □

The class of refinements of $\mathcal{C}$-machines and of superdeterministic PDAs are incomparable as classes of PDAs. However, they are equally expressive in the sense that the class of languages accepted by refinements of $\mathcal{C}$-machines is equivalent to the one accepted by superdeterministic PDAs.

Theorem 10. *A language is superdeterministic if and only if it is accepted by a refinement of a $\mathcal{C}$-machine.*

Proof. We give a proof sketch. We first prove the right-to-left direction. A state q of $\mathcal{C}$-machine C has a ε-loop if there is a sequence of ε-transitions starting from and ending with q, i.e., $(q, n) \Vdash_C^{\varepsilon} \cdots \Vdash_C^{\varepsilon} (q, m)$ for some n and m. By removing states which have ε-loops, we can construct an equivalent $\mathcal{C}$-machine that is of finite delay. Similarly, we can assume without loss of generality that any refinement of a $\mathcal{C}$-machine is of finite delay. Consider condition (2) in Definition 5. The condition on the stack length must be satisfied by all refinements of $\mathcal{C}$-machines, but the condition on the state may not in general. However we can always construct another refinement that satisfies the condition by moving the refined state information to the stack top, i.e., instead of refining a configuration of the $\mathcal{C}$ machine (q, n) to $(q', A_1 \ldots A_n)$, refining it to $(q, \langle A_1, q_1\rangle \ldots \langle A_n, q'\rangle)$. So for all refinements of $\mathcal{C}$-machines, we can construct another refinement which is superdeterministic and accepts the same language.

For the other direction, let M be a superdeterministic PDA and d be its delay. Note that for any configuration $(q, \widetilde{B}A_{d+1} \ldots A_1)$, only $d+1$ stack symbols at the top (i.e., $A_{d+1} \ldots A_1$) affect a transition $(q, \widetilde{B}\widetilde{A}) \vDash_M^a (q', \widetilde{B}\widetilde{C})$. So we can construct another superdeterministic PDA M', whose transition coincides with the transition of M and is normalized as follows:

$$\begin{aligned}
(q, \widetilde{B}\widetilde{A}) &\Vdash_{M'}^{a} (\langle q, a\rangle, \widetilde{B}\widetilde{A}) \\
&\Vdash_{M'}^{\varepsilon} (\langle q, a, A_1\rangle, \widetilde{B}A_{d+1} \ldots A_2) \\
&\ \ \vdots \\
&\Vdash_{M'}^{\varepsilon} (\langle q, a, \widetilde{A}\rangle, \widetilde{B}) \\
&\Vdash_{M'}^{\varepsilon} (q', \widetilde{B}\widetilde{C}).
\end{aligned}$$

In the first stage of the transition, M' records a on its state, pops its stack d times and records them on the state. Then the state is a triple of the form $\langle q, a, \widetilde{A}\rangle$. In the last stage, M' computes q' and $\widetilde{C}$ from its state $\langle q, a, \widetilde{A}\rangle$. See Appendix D for more details about the construction of M'.

Let $\natural(\cdot)$ be a mapping which forgets stack symbols such as

$$\natural((\langle q, a, A_n \ldots A_1\rangle, B_m \ldots B_1)) = (\langle q, a, n\rangle, m).$$

The mapping $\natural(\cdot)$ and the transition relation δ of M induces a transition relation $\natural(\delta)$ of some $\mathcal{C}$-machine, which is an approximation of M. Condition (2) in Definition 5 ensures that $\natural(\delta)$ is deterministic. $\square$

The decidability of the inclusion problem between context-free languages and superdeterministic languages has been proved by Greibach and Friedman [5]. The proof of Theorem 9 with Theorem 10 is an alternative and arguably simpler proof of the result.

6 Related Work

There have been a number of studies on the inclusion problems for subclasses of context-free languages (see [3] for a survey).

One of the strongest decidability results is about the inclusion between context-free languages and superdeterministic languages, proved by Greibach and Friedman [5]. Nguyen and Ogawa [15] gave a new proof by simplifying the technique used in [5]. Greibach and Friedman [5] reduced the problem to the emptiness problem for a pushdown automaton and Nguyen and Ogawa [15] gave simpler construction of a pushdown automaton.

Minamide and Tozawa [12] have proposed an algorithm for inclusion between context-free languages and regular hedge languages, motivated by the validation of dynamically generated HTML documents. As demonstrated in Section 5.1, our method gives an alternative algorithm for the same problem, although our algorithm may not be as efficient as Minamide and Tozawa's. Møller and Schwarz [13] have developed an algorithm to validate a context-free grammar against SGML DTDs, dealing with tag omissions and exceptions. It is not clear whether our method can provide a similar result.

The subclass of the context-free languages named *visibly pushdown languages* [1, 2] has many good properties such as boolean closure and decidability of the emptiness problem in polynomial time. Some researchers have extended the class preserving such properties. Caucal [4] has introduced a notion of *synchronized pushdown automata* and Nowotka and Srba [16] have proposed *height-deterministic pushdown automata*. The refinement of a counter machine is similar to those notions. Since the class of visibly pushdown automata can be defined as the class of refinements of a certain automaton, our notion of refinements may give an extension of them.

Recently, type-based approaches to model-checking, verification and language inclusion problems have been extensively studied [7–9, 11, 19, 20]. Kobayashi and Ong [7, 9] have proposed a type system for recursion schemes that is equivalent to the modal μ-calculus model-checking of recursion schemes (the decidability of the model-checking problem has been proved by Ong [17]). These type systems have been applied to verification of higher-order programs [7, 11, 10], and practically effective typability checkers have been developed [6, 8]. The present work extends type systems to deal with infinite state systems, namely deterministic pushdown automata. Types are now configurations of pushdown automata, rather than states of automata, which are finite a priori.

In our previous work [20], we gave a type-based proof for the inclusion problem between context-free languages and superdeterministic languages. But the proof is specific to superdeterministic languages, and difficult to generalize.

7 Conclusion and Future Work

We have proposed an intersection type system characterizing the inclusion by a deterministic context-free language, and given a sufficient condition of decidability of its typability. Future work includes extensions in two directions, extending grammars and automata. A naive extension to higher-order recursion schemes fails to establish the counterpart of Theorem 4. That is because the up-to technique used in this paper is too crude to deal with them. To extend automata is easier than grammars. For example, we can develop a framework for higher-order pushdown automata. So what we should do is to find a language accepted by a higher-order pushdown automaton which has decidable inclusion problem and a practical use.

Acknowledgement. The authors would like to thank the anonymous reviewers for their valuable comments. This work is partially supported by Kakenhi 23220001 and 22·3842.

References

1. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Babai, L. (ed.) STOC. pp. 202–211. ACM (2004)
2. Alur, R., Madhusudan, P.: Adding nesting structure to words. J. ACM 56(3), 1–43 (2009)
3. Asveld, P.R.J., Nijholt, A.: The inclusion problem for some subclasses of context-free languages. Theor. Comput. Sci. 230(1-2), 247–256 (2000)
4. Caucal, D.: Synchronization of pushdown automata. In: Ibarra, O.H., Dang, Z. (eds.) Developments in Language Theory. Lecture Notes in Computer Science, vol. 4036, pp. 120–132. Springer (2006)
5. Greibach, S.A., Friedman, E.P.: Superdeterministic PDAs: A subcase with a decidable inclusion problem. J. ACM 27(4), 675–700 (1980)
6. Kobayashi, N.: Model-checking higher-order functions. In: Porto, A., López-Fraguas, F.J. (eds.) PPDP. pp. 25–36. ACM (2009)
7. Kobayashi, N.: Types and higher-order recursion schemes for verification of higher-order programs. In: Shao, Z., Pierce, B.C. (eds.) POPL. pp. 416–428. ACM (2009)
8. Kobayashi, N.: A practical linear time algorithm for trivial automata model checking of higher-order recursion schemes. In: FoSSaCS. pp. 260–274. Springer (2011)
9. Kobayashi, N., Ong, C.H.L.: A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In: LICS. pp. 179–188. IEEE Computer Society (2009)
10. Kobayashi, N., Sato, R., Unno, H.: Predicate abstraction and CEGAR for higher-order model checking. In: Hall, M.W., Padua, D.A. (eds.) PLDI. pp. 222–233. ACM (2011)
11. Kobayashi, N., Tabuchi, N., Unno, H.: Higher-order multi-parameter tree transducers and recursion schemes for program verification. In: Hermenegildo, M.V., Palsberg, J. (eds.) POPL. pp. 495–508. ACM (2010)
12. Minamide, Y., Tozawa, A.: XML validation for context-free grammars. In: Kobayashi, N. (ed.) APLAS. Lecture Notes in Computer Science, vol. 4279, pp. 357–373. Springer (2006)
13. Møller, A., Schwarz, M.: HTML validation of context-free languages. In: FoSSaCS, pp. 426–440. Springer (2011)
14. Murata, M.: Hedge automata: a formal model for XML schemata (1999), `http://www.xml.gr.jp/relax/hedge_nice.html`
15. Nguyen, V.T., Ogawa, M.: Alternate stacking technique revisited: Inclusion problem of superdeterministic pushdown automata. IPSJ Transactions on Programming 1(1), 36–46 (2008)

16. Nowotka, D., Srba, J.: Height-deterministic pushdown automata. In: Kucera, L., Kucera, A. (eds.) MFCS. Lecture Notes in Computer Science, vol. 4708, pp. 125–134. Springer (2007)
17. Ong, C.H.L.: On model-checking trees generated by higher-order recursion schemes. In: LICS. pp. 81–90. IEEE Computer Society (2006)
18. Pair, C., Quéré, A.: Définition et etude des bilangages réguliers. Information and Control 13(6), 565–593 (1968)
19. Tsukada, T., Kobayashi, N.: Untyped recursion schemes and infinite intersection types. In: Ong, C.H.L. (ed.) FOSSACS. Lecture Notes in Computer Science, vol. 6014, pp. 343–357. Springer (2010)
20. Tsukada, T., Kobayashi, N.: A type-theoretic proof of the decidability of the language containment between context-free languages and superdeterministic languages (in japanese). IPSJ Transactions on Programming 4(2), 31–47 (2011)
21. Uezato, Y., Minamide, Y.: Monoid-based approach to the inclusion problem on superdeterministic pushdown automata. In: DLT. Lecture Notes in Computer Science, vol. 9840, pp. 393–405. Springer (2016)

A Detailed Proof of Lemma 1

Let $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R}, S)$ be a CFG, M be a PDA and $\iota = \bigwedge \Theta \to c$ be its initial type. Assume that $\mathcal{L}_\mathcal{G} \subseteq \mathcal{L}_M(\iota)$.

For a sequence $\alpha_1 \dots \alpha_n$ of non-terminals and terminals, we define

$$\begin{aligned}\mathcal{L}_\mathcal{G}(\alpha_1 \dots \alpha_n) &= \{w \mid \alpha_1(\alpha_2(\dots(\alpha_n(\$))\dots)) \Rightarrow^*_\mathcal{R} w\$\}\\ \mathrm{pre}(\alpha_1 \dots \alpha_n) &= \{w \mid S\$ \Rightarrow^*_\mathcal{R} w(\alpha_1(\alpha_2(\dots(\alpha_n(v(\$)))\dots)))\}.\end{aligned}$$

We can assume without loss of generality that for every sequence $\alpha_1 \dots \alpha_n$, $\mathcal{L}_\mathcal{G}(\alpha_1 \dots \alpha_n) \neq \emptyset$ and $\mathrm{pre}(\alpha_1 \dots \alpha_n) \neq \emptyset$. Let

$$\Delta_0 = \{F{:}\bigwedge \Xi \to d \mid \exists w \in \mathrm{pre}(F).\ c \vDash^w_M d \text{ and } \Xi = \{d' \mid \exists u \in \mathcal{L}_\mathcal{G}(F). d \vDash^u_M d'\}\}.$$

Lemma 6. *Let F be a non-terminal and $F : \tau \in \Delta_0$. Then $\mathcal{L}_\mathcal{G}(F) \subseteq \mathcal{L}_M(\tau)$.*

Proof. Let $\tau = \bigwedge \Xi \to d$. Assume $w \in \mathcal{L}_\mathcal{G}(F)$. By the definition of Δ_0, we have u and v such that $S\$ \Rightarrow^*_\mathcal{R} u(F(v(\$)))$ and $c \vDash^u_M d$. Since $u(w(v(\$))) \in \mathcal{L}_\mathcal{G} \subseteq \mathcal{L}_M(\iota) = \mathcal{L}_M(\bigwedge \Theta \to c)$, we have $c \vDash^u_M d \vDash^w_M d' \vDash^v_M c'$ for some d' and c'. By the definition of Δ_0, we have $d' \in \Xi$. So $w \in \mathcal{L}_M(\bigwedge \Xi \to d)$ as required. □

Lemma 7. *Let $\alpha_1 \dots \alpha_n$ be a sequence of non-terminals and terminals, and $\bigwedge \Xi \to d$ be a type. If $\mathcal{L}_\mathcal{G}(\alpha_1 \dots \alpha_n) \subseteq \mathcal{L}_M(\bigwedge \Xi \to d)$ and there exists $w \in \mathrm{pre}(\alpha_1 \dots \alpha_n)$ such that $c \vDash^w_M d$, then $\Delta_0, x : \bigwedge \Xi \vdash_M \alpha_1(\alpha_2(\dots(\alpha_n(x))\dots)) : d$.*

Proof. By induction on the length n of the sequence. The base case $n = 0$ is trivial. We prove the induction step. We assume that $\alpha_1 = F \in \mathcal{N}$. The case $\alpha_1 = a \in \Sigma$ can be proved by the same way.

Let w be a normal word such that $w \in \mathrm{pre}(F\alpha_2 \dots \alpha_n)$ and $c \vDash^w_M d$. By the definition of Δ_0, we have $(F : \bigwedge \Xi' \to d) \in \Delta_0$ for some Ξ'. Let $d' \in \Xi'$. We should show that $\Delta_0, x : \bigwedge \Xi \vdash_M \alpha_2(\dots(\alpha_n(x))\dots) : d'$. Because $F : \bigwedge \Xi' \to d \in \Delta_0$ and $d' \in \Xi'$, by the definition of Δ_0, we have $u \in \mathcal{L}_\mathcal{G}(F)$ such that $d \vDash^u_M d'$. We have $\mathcal{L}_\mathcal{G}(\alpha_2 \dots \alpha_n) \subseteq \mathcal{L}_M(\bigwedge \Xi \to d')$ because $\mathcal{L}_\mathcal{G}(w\alpha_2 \dots \alpha_n) \subseteq \mathcal{L}_M(\bigwedge \Xi \to d)$ and $d \vDash^w_M d'$. Therefore, $wu \in \mathrm{pre}(\alpha_2 \dots \alpha_n)$, $c \vDash^{wu}_M d'$ and $\mathcal{L}_\mathcal{G}(\alpha_2 \dots \alpha_n) \subseteq \mathcal{L}_M(\bigwedge \Xi \to d')$. So by the induction hypothesis, we have $\Delta_0, x : \bigwedge \Xi \vdash_M \alpha_2(\dots(\alpha_n(x))\dots) : d'$. □

Lemma 8. $\Delta_0 \vdash_M (\mathcal{R}, S) : \iota$.

Proof. It is easy to show that $\Delta_0, \$: \bigwedge \Theta \vdash_M S\$: c$ from $\mathcal{L}_\mathcal{G}(S) \subseteq \mathcal{L}_M(\iota)$ and the definition of Δ_0. We show $\Delta_0 \vdash_M \mathcal{R}$.

Let $F : \bigwedge \Xi \to d \in \Delta_0$ and $F\,x \to \alpha_1(\alpha_2(\dots(\alpha_n(x))\dots)) \in \mathcal{R}$. By the definition of Δ_0, there is w such that $w \in \mathrm{pre}(F)$ and $c \vDash^w_M d$. Then $w \in \mathrm{pre}(\alpha_1 \dots \alpha_n)$ because $\mathrm{pre}(F) \subseteq \mathrm{pre}(\alpha_1 \dots \alpha_n)$. We have $\mathcal{L}_\mathcal{G}(\alpha_1 \dots \alpha_n) \subseteq \mathcal{L}_M(\bigwedge \Xi \to d)$ by $\mathcal{L}_\mathcal{G}(\alpha_1 \dots \alpha_n) \subseteq \mathcal{L}_\mathcal{G}(F)$ and Lemma 6. So by Lemma 7, we have $\Delta_0, x : \bigwedge \Xi \vdash_M \alpha_1(\alpha_2(\dots(\alpha_n(x))\dots)) : d$. □

Lemma 9. *For all Δ such that $\Delta \vdash_M (\mathcal{R}, S) : \iota$, we have $\Delta_0 \sqsubseteq \Delta$.*

Proof. Let $F : \bigwedge \Xi_0 \to c_0 \in \Delta_0$.

First, we show that $F : \bigwedge \Xi \to c_0 \in \Delta$ for some Ξ. By the definition of Δ_0, we have $S\$ \Rightarrow^*_{\mathcal{R}} w(F(v(\$)))$ and $c \vDash^w_M c_0$ for some w and v. Since $\Delta, \$: \bigwedge \Theta \vdash_M S\$: c$ and the typing is preserved by reductions, we have $\Delta, \$: \bigwedge \Theta \vdash_M w(F(v(\$))) : c$. By induction on the length of w, we have $\Delta, \$: \bigwedge \Theta \vdash_M Fv\$: c_0$, using determinism of M. Thus $F : \bigwedge \Xi \to c_0 \in \Delta$ for some Ξ.

Second, we show that $\Xi_0 \subseteq \Xi$. Let $d_0 \in \Xi_0$. By the definition of Δ_0, we have $w \in \mathcal{L}_{\mathcal{G}}(F)$ such that $c_0 \vDash^w_M d_0$. Since $F : \bigwedge \Xi \to c_0 \in \Delta$, we have $\Delta, \$: \bigwedge \Xi \vdash_M F\$: c_0$. Thus $\Delta, \$: \bigwedge \Xi \vdash_M w\$: c_0$ because $w \in \mathcal{L}_{\mathcal{G}}(F)$ and the typing is preserved by reductions. Since $c_0 \vDash^w_M d_0$, we have $\Delta, \$: \bigwedge \Xi \vdash_M \$: d_0$. Therefore $d_0 \in \Xi$ as required. □

Theorem 1 is a consequence of Lemma 8 and Lemma 9.

B Detailed Proof of Theorem 4

Claim

Let $\mathcal{G} = (\mathcal{N}, \Sigma, \mathcal{R}, S)$ be a CFG, M_1 and M_2 be PDAs, ι_1 and ι_2 be their initial types and $f : (M_1, \iota_1) \to (M_2, \iota_2)$ be a homomorphism. Assume that $\Delta_2 \vdash^{\Uparrow}_{M_2} (\mathcal{R}, S) : \iota_2$. If $\Delta_1 \vdash^{\Uparrow}_{M_1} (\mathcal{R}, S) : \iota_1$, then there exists Δ'_1 such that $\Delta'_1 \vdash^{\Uparrow}_{M_1} (\mathcal{R}, S) : \iota_1$ and $f(\Delta'_1) \sqsubseteq \Delta_2$.

Proof

Since $\Delta_1 \vdash^{\Uparrow}_{M_1} (\mathcal{R}, S) : \iota_1$, there is the minimum witness type environment by Lemma 1. Let Δ^0_1 be the minimum witness of $\vdash_{M_1} (\mathcal{R}, S) : \iota_1$. Note that $f(\Delta^0_1) \sqsubseteq \Delta^{\Uparrow}_2$ by Theorem 3.

We define $(q, A_1 A_2 \dots A_m) \Downarrow n = (q, A_{n+1} \dots A_m)$ if $m > n$ (and undefined otherwise). This operation is extended to types by $(\bigwedge \Theta \to c) \Downarrow n = \bigwedge \{d \Downarrow n \mid d \in \Theta\} \to (c \Downarrow n)$.

Let $(F, \tau^0_1, \tau_2, \widetilde{A}_2)$ be a quadruple such that $F : \tau^0_1 \in \Delta^0_1$, $F : \tau_2 \in \Delta_2$ and $f(\tau^0_1) \sqsubseteq (\tau_2 \Uparrow \widetilde{A}_2)$. The corresponding type binding $F : \tau'_1$ of the quadruple is defined by $\tau'_1 = \tau^0_1 \Downarrow n$, where n is the length of $\widetilde{A}_2$. Let Δ'_1 be the set of all such bindings $F : \tau'_1$.

We show that Δ'_1 satisfies the requirements of the above claim. It is trivial that $f(\Delta'_1) \sqsubseteq \Delta_2$ by the construction. We prove $\Delta'_1 \vdash^{\Uparrow}_{M_1} (\mathcal{R}, S) : \iota_1$.

Lemma 10. *Let c^0_1 and c_2 be configurations of M_1 and M_2, respectively. Assume*

1. $f(c^0_1) \sqsubseteq (c_2 \Uparrow \widetilde{A}_2)$,
2. $F : \bigwedge \Theta^0_1 \to c^0_1 \in \Delta^0_1$ *and*
3. $F : \bigwedge \Theta_2 \to c_2 \in \Delta_2$.

Then $f(\bigwedge \Theta^0_1 \to c^0_1) \sqsubseteq ((\bigwedge \Theta_2 \to c_2) \Uparrow \widetilde{A}_2)$.

Proof. It is enough to show that for every $d_1^0 \in \Theta_1^0$, there is $d_2 \in \Theta_2$ such that $f(d_1^0) = (d_2 \Uparrow \widetilde{A}_2)$.

Let $d_1^0 \in \Theta_1^0$. Since Δ_1^0 is minimum, by the construction of the minimum witness (Lemma 1), we have $w \in \mathcal{L}_{\mathcal{G}}(F)$ such that $c_1^0 \vDash_{M_1}^w d_1^0$. Since $\Delta_2 \vdash_{M_2}^{\Uparrow} (\mathcal{R}, F){:}\bigwedge \Theta_2 \to c_2$, by soundness (Theorem 1 with Theorem 2), we have $\mathcal{L}_{\mathcal{G}}(F) \subseteq \mathcal{L}_{M_2}(\bigwedge \Theta_2 \to c_2)$. Therefore we have $c_2 \vDash_{M_2}^w d_2$ for some $d_2 \in \Theta_2$.

We show that $f(d_1^0) = (d_2 \Uparrow \widetilde{A}_2)$. We define d_2' by $(c_2 \Uparrow \widetilde{A}_2) \vDash_{M_2}^w d_2'$. Let n be the length of $\widetilde{A}_2$ and assume

$$\begin{aligned} c_1^0 &= (q_1, A_1^1 \dots A_1^m) \\ c_2 \Uparrow \widetilde{A}_2 &= (q_2, A_2^1 \dots A_2^m). \end{aligned}$$

Therefore $\widetilde{A}_2 = A_2^1 \dots A_2^n$ and $c = (q_2, A_2^{n+1} \dots A_2^m)$. Then we have the following transitions:

$$\begin{aligned} c_1^0 = (q_1, A_1^1 \dots A_1^m) &\vDash_{M_1}^w (q_1', B_1^1 \dots B_1^{m'}) = d_1^0 \\ (c_2 \Uparrow \widetilde{A}_2) = (q_2, A_2^1 \dots A_2^m) &\vDash_{M_2}^w (q_2', B_2^1 \dots B_2^{m'}) = d_2' \\ c_2 = (q_2, A_2^{n+1} \dots A_2^m) &\vDash_{M_2}^w (q_2', B_2^{n+1} \dots B_2^{m'}) = d_2. \end{aligned}$$

Because f is a homomorphism between M_1 and M_2, we have $f(A_1^i) = A_2^i$ for every $1 \le i \le m$ and $f(B_1^i) = B_2^i$ for every $1 \le i \le m'$.

We claim that $A_1^i = B_1^i$ for every $1 \le i \le n$. Assume it is not the case. Then A_1^n is popped during the transition $c_1^0 \vDash_{M_1}^w d_1^0$, i.e., for some prefix ua of w,

$$(q_1, A_1^1 \dots A_1^m) \vDash_{M_1}^u \Vdash_{M_1}^a \Vdash_{M_1}^\varepsilon \Vdash_{M_1}^\varepsilon \cdots \Vdash_{M_1}^\varepsilon (p, A_1^1 \dots A_1^n),$$

where we abbreviate $c \vDash_{M_1}^a c' \vDash_{M_1}^b c''$ for some c' to $c \vDash_{M_1}^a \vDash_{M_2}^b c''$. Then we have

$$\begin{aligned} (q_1, A_1^1 \dots A_1^m) &\vDash_{M_1}^u \Vdash_{M_1}^a \Vdash_{M_1}^\varepsilon \Vdash_{M_1}^\varepsilon \cdots \Vdash_{M_1}^\varepsilon (p_1, A_1^1 \dots A_1^n) \\ (q_2, A_2^1 \dots A_2^m) &\vDash_{M_2}^u \Vdash_{M_2}^a \Vdash_{M_2}^\varepsilon \Vdash_{M_2}^\varepsilon \cdots \Vdash_{M_2}^\varepsilon (p_2, A_2^1 \dots A_2^n) \\ (q_2, A_2^{n+1} \dots A_2^m) &\vDash_{M_2}^u \Vdash_{M_2}^a \Vdash_{M_2}^\varepsilon \Vdash_{M_2}^\varepsilon \cdots \Vdash_{M_2}^\varepsilon (p_2, \varepsilon), \end{aligned}$$

where ε denotes the empty sequence. So the transition starting from $(q_2, A_2^{n+1} \dots A_2^m)$ get stuck. It contradict to the assumption. Thus $A_1^i = B_1^i$ for every $1 \le i \le n$.

Then we have $A_2^i = B_2^i$ for every $1 \le i \le n$, because $A_2^i = f(A_1^i) = f(B_1^i) = B_2^i$. Therefore $f(d_1^0) = d_2' = (d_2 \Uparrow \widetilde{A}_2)$, as required. □

Lemma 11. *Let $\alpha_i \in \mathcal{N} \cup \Sigma$. Assume*

1. $f(\bigwedge \Theta_1^0 \to c_1^0) \sqsubseteq ((\bigwedge \Theta_2 \to c_2) \Uparrow \widetilde{A}_2)$,
2. $\Delta_1^0, x : \bigwedge \Theta_1^0 \vdash_{M_1} \alpha_1(\alpha_2(\dots(\alpha_n(x))\dots)) : c_1^0$ *and*
3. $\Delta_2^{\Uparrow}, x : \bigwedge \Theta_2 \vdash_{M_2} \alpha_1(\alpha_2(\dots(\alpha_n(x))\dots)) : c_2$.

Let $\bigwedge \Theta_1' \to c_1' = ((\bigwedge \Theta_1^0 \to c_1^0) \Downarrow m)$, where m is the length of $\widetilde{A}_2$. Then $(\Delta_1')^{\Uparrow}, x : \bigwedge \Theta_1' \vdash_{M_1} \alpha_1(\alpha_2(\dots(\alpha_n(x))\dots)) : c_1'$.

Proof. By induction on n. The base case $n = 0$ is trivial. We assume that $n > 0$. There are two cases. The case $\alpha_1 \in \Sigma$ is easy. Assume $\alpha_1 = F \in \mathcal{N}$. Let $t = \alpha_2(\dots(\alpha_n(x))\dots)$. Then we have the following derivations: the derivation in $\mathcal{T}_1$

$$\frac{\Delta_1^0, x : \bigwedge \Theta_1^0 \vdash_{M_1} F : \bigwedge \Xi_1^0 \to c_1^0 \qquad \Delta_1^0, x : \bigwedge \Theta_1^0 \vdash_{M_1} t : d_1^0 \quad (\text{for all } d_1^0 \in \Xi_1^0)}{\Delta_1^0, x : \bigwedge \Theta_1^0 \vdash_{M_1} F(t) : c_1^0},$$

and the derivation in $\mathcal{T}_2$

$$\frac{\Delta_2^{\Uparrow}, x : \bigwedge \Theta_2 \vdash_{M_2} F : \bigwedge \Xi_2 \to c_2 \qquad \Delta_2^{\Uparrow}, x : \bigwedge \Theta_2 \vdash_{M_2} t : d_2 \quad (\text{for all } d_2 \in \Xi_2)}{\Delta_2^{\Uparrow}, x : \bigwedge \Theta_2 \vdash_{M_2} F(t) : c_2}.$$

So $F : \bigwedge \Xi_1^0 \to c_1^0 \in \Delta_1^0$ and $F : \bigwedge \Xi_2 \to c_2 \in \Delta_2^{\Uparrow}$. By the definition of $\Delta^{\Uparrow}$, there are $\widetilde{B}$ and $F : \bigwedge \Xi_2' \to c_2' \in \Delta_2$ such that $\bigwedge \Xi_2 \to c_2 = (\bigwedge \Xi_2' \to c_2') \Uparrow \widetilde{B}$. Then, by Lemma 10, we have

$$f(\bigwedge \Xi_1^0 \to c_1^0) \sqsubseteq ((\bigwedge \Xi_2' \to c_2') \Uparrow (\widetilde{A}_2\widetilde{B}_2)) = ((\bigwedge \Xi_2 \to c_2) \Uparrow \widetilde{A}_2).$$

Especially, for any $d_1^0 \in \Xi_1^0$, we have $d_2 \in \Xi_2$ such that $f(d_1^0) = d_2 \Uparrow \widetilde{A}_2$. Therefore, for all $d_1^0 \in \Xi_1^0$, there is $d_2 \in \Xi_2$ such that

$$f(\bigwedge \Theta_1^0 \to d_1^0) \sqsubseteq ((\bigwedge \Theta_2 \to d_2) \Uparrow \widetilde{A}_2).$$

So by the induction hypothesis, we have $(\Delta_1')^{\Uparrow}, x : \bigwedge \Theta_2 \vdash_{M_2} t : (d_1^0 \Downarrow n)$. The rest of the proof is straightforward, using Lemma 10. $\square$

Then $\Delta_1' \vdash_{M_1}^{\Uparrow} (\mathcal{R}, S) : \iota_1$ is an easy consequence of $\Delta_1^0 \vdash_{M_1} (\mathcal{R}, S) : \iota_1$ and $\Delta_2 \vdash_{M_2}^{\Uparrow} (\mathcal{R}, S) : \iota_2$ and Lemma 11.

C Detailed Proof of Theorem 8

We fix a $\mathcal{C}$-machine $C(\iota)$ in this section. Let $\iota = \bigwedge \mathcal{F} \to (q_S, n_S)$ and L be the maximal number in $\mathcal{F}$, i.e., $L = \max\{n \mid (q, n) \in \mathcal{F}\}$. Since ι is finite, L is well-defined.

The followings are the key properties of $\mathcal{C}$-machines. They are easy to prove.

Lemma 12. *For any word w, if $(q, n_1) \vDash_C^w (q_1', n_1')$ and $(q, n_2) \vDash_C^w (q_2', n_2')$, then $q_1' = q_2'$ and $n_1' - n_1 = n_2' - n_2$.* $\square$

Lemma 13. *For any word w, if $w \in \mathcal{L}_C(\bigwedge \mathcal{F} \to (q, n_1))$ and $w \in \mathcal{L}_C(\bigwedge \mathcal{F} \to (q, n_2))$, then $|n_1 - n_2| \le L$.* $\square$

Let $\mathcal{G}$ be a CFG. Assume that $\mathcal{L}_{\mathcal{G}} \subseteq \mathcal{L}_C(\iota)$. Then there is the minimum witness of $\vdash_C (\mathcal{R}, S) : \iota$ by Lemma 1. Let Δ_0 be the minimum witness. We show some properties of Δ_0. Recall that Δ_0 is defined by

$$\Delta_0 = \{F{:}\bigwedge \Theta \to c \mid \Theta = \{d \mid \exists w \in \mathcal{L}_{\mathcal{G}}(F).\ c \vDash_M^w d\} \text{ and } \exists w \in \mathrm{pre}(F).\ (q_S, n_S) \vDash_C^w c\},$$

where $\mathrm{pre}(F) = \{w \mid S\$ \Rightarrow_{\mathcal{R}}^* wFv\$\}$. In other words, if $F : \bigwedge \Theta \to c \in \Delta_0$, we know that

1. c is reachable by $\mathrm{pre}(F)$ from the initial state, and
2. Θ is the set of all reachable configurations by $\mathcal{L}_\mathcal{G}(F)$ from c.

Lemma 14. *Assume* $F : \bigwedge \Theta_1 \to (q, n_1) \in \Delta_0$ *and* $F : \bigwedge \Theta_2 \to (q, n_2) \in \Delta_0$ *and* $n_2 \geq n_1$. *Then* $(\bigwedge \Theta_1 \to (q, n_1)) \Uparrow (n_2 - n_1) = \bigwedge \Theta_2 \to (q, n_2)$.

Proof. A consequence of the definition of Δ_0 and Lemma 12. We should show that $(q', n'_1) \in \Theta_1$ implies $(q', n'_1 + (n_2 - n_1)) \in \Theta_2$ and $(q', n'_2) \in \Theta_2$ implies $(q', n'_2 - (n_2 - n_1)) \in \Theta_1$. Here we prove the latter. The former can be proved in the same way.

Assume $(q', n'_2) \in \Theta_2$. We show that $(q', n'_2 - (n_2 - n_1)) \in \Theta_1$. By the definition of Δ_0, there is $w \in \mathcal{L}_\mathcal{G}(F)$ such that $(q, n_2) \vDash^w_C (q', n'_2)$. By soundness (Theorem 1), $\Delta_0 \vdash_C (\mathcal{R}, F) : \bigwedge \Theta_1 \to (q, n_1)$ and $w \in \mathcal{L}_\mathcal{G}(F)$ implies that there is some configuration d such that $(q, n_1) \vDash^w_C d$. By Lemma 12, we have $d = (q', n'_1)$, where $n'_1 - n_1 = n'_2 - n_2$. Thus $n'_1 = n'_2 - (n_2 - n_1)$. By the definition of Δ_0, we have $(q', n'_1 - (n_2 - n_1)) \in \Theta_1$ as required. ☐

A consequence of Lemma 14 is that for each F and q, there is a canonical type binding $F : \bigwedge \Theta \to (q, n) \in \Delta_0$ in the sense that all other type bindings of the form $F : \bigwedge \Theta' \to (q, n') \in \Delta_0$ are obtained by its extensions, i.e., there is k such that $(\bigwedge \Theta \to (q, n)) \Uparrow k = \bigwedge \Theta' \to (q, n')$. Let Δ_C be the set of all canonical bindings defined by

$$\Delta_C = \{F : \bigwedge \Theta_1 \to (q, n_1) \in \Delta_0 \mid \forall (F : \bigwedge \Theta_2 \to (q, n_2)) \in \Delta_0.\ n_1 \leq n_2\}.$$

Clearly, $\Delta_C \vdash^{\Uparrow}_C (\mathcal{R}, S) : \iota$.

The next lemma restricts the shape of types in Δ_0 (and thus types in Δ_C).

Lemma 15. *Assume* $F : \bigwedge \Theta \to (q, n) \in \Delta_0$ *and* $(p, m), (p, m') \in \Theta$. *Then* $|m - m'| \leq L$.

Proof. A consequence of the definition of Δ_0 and Lemma 13. By the definition of Δ_0, we have $w, w' \in \mathcal{L}_\mathcal{G}(F)$ such that $(q, n) \vDash^w_C (p, m)$ and $(q, n) \vDash^{w'}_C (p, m')$. Moreover, by the definition of Δ_0, we have $v \in \mathrm{pre}(F)$ such that $(q_S, n_S) \vDash^v_C (q, n)$. By the definition of $\mathrm{pre}(F)$, we have u such that $S\$ \Rightarrow^*_\mathcal{R} vFu\$$. Since $\mathcal{L}_\mathcal{G} \subseteq \mathcal{L}_C(\bigwedge \mathcal{F} \to (q_S, n_S))$, we have $vwu, vw'u \in \mathcal{L}_C(\bigwedge \mathcal{F} \to (q_S, n_S))$. As a result, we have two transition sequences

$$(q_S, n_S) \vDash^v_C (q, n) \vDash^w_C (p, m) \vDash^u_C d$$
$$(q_S, n_S) \vDash^v_C (q, n) \vDash^{w'}_C (p, m') \vDash^u_C d'$$

where $d, d' \in \mathcal{F}$. Especially, $u \in \mathcal{L}_C(\bigwedge \mathcal{F} \to (p, m))$ and $u \in \mathcal{L}_C(\bigwedge \mathcal{F} \to (p, m'))$. Thus, by Lemma 13, we have $|m - m'| \leq L$. ☐

Now we construct Δ containing Δ_C.

Let N be a finite-state automaton obtained by removing the counter of C, i.e., the set of states of N is equivalent to C and $p \vDash^a_N q$ if and only if $(p, n) \vDash^a_C (q, m)$ for some m and n. Let $\mathcal{F}' = \{q \mid (q, n) \in \mathcal{F}\}$.

We solve the typability problem of $\mathcal{G}$ in $\mathcal{T}_N$. Since N is a finite-state automaton, the sets of types and type environments are finite. So we can decide whether there is a witness of $\vdash_N (\mathcal{R}, S) : \bigwedge \mathcal{F}' \to q_S$ and construct the minimum witness if it exists. Let Δ_N be the minimum witness of $\vdash_N (\mathcal{R}, S) : \bigwedge \mathcal{F}' \to q_S$.

For each $F : \bigwedge \Theta' \to q \in \Delta_N$, we construct a type binding in $\mathcal{T}_C$. Since Δ_N is minimum, there are words which satisfy the following conditions:

1. $v \in \mathrm{pre}(F)$ such that $q_S \vDash_N^v q$.
2. w_p for each $p \in \Theta'$ such that $q \vDash_N^{w_p} p$.

We define configurations (q, n) and (p, m_p) (for each $p \in \Theta'$) of C by $(q_S, n_S) \vDash_C^v (q, n)$ and $(q, n) \vDash_C^{w_p} (p, m_p)$ (if no such configurations exist, then $\mathcal{L}_\mathcal{G} \nsubseteq \mathcal{L}_C(\iota)$). Let

$$\tau = \bigwedge \{(p, m_p + k) \mid p \in \Theta' \text{ and } -L \leq k \leq L \text{ and } m_p + k > 0\} \to (q, n).$$

The corresponding type binding in $\mathcal{T}_C$ is $F : \tau$. Let Δ_1 be the set of all such bindings.

Lemma 16. *Let τ be the type constructed above. Then there is σ such that $F : \sigma \in \Delta_0$ and $\sigma \sqsubseteq \tau$.*

Proof. Since $v \in \mathrm{pre}(F)$, by the definition of Δ_0, there is $F : \bigwedge \Theta \to (q, n) \in \Delta_0$. Let $(p, m') \in \Theta$. We should show that $m' = m_p + k$ for some $-L \leq k \leq L$. By the definition of Θ', we have $p \in \Theta'$. By the definition of (p, m_p), we have $(p, m_p) \in \Theta$. From Lemma 15, we have $|m_p - m'| \leq L$. Thus $-L \leq m_p - m' \leq L$ as required. □

Then Δ_1 is bigger than Δ_C in the following sense.

Lemma 17. *For any $F{:}\sigma \in \Delta_C$, there is k and $F{:}\tau \in \Delta_1$ such that $(\sigma \Uparrow k) \sqsubseteq \tau$.*

Proof. Let $\sigma = \bigwedge \Theta \to (p, n)$. By the construction of Δ_C, we know that (p, n) is reachable by $\mathrm{pre}(F)$ from (q_S, n_S) in C. Thus p is reachable by $\mathrm{pre}(F)$ from q_S in N. By the construction of Δ_1, there is a type binding $F : \bigwedge \Theta' \to p \in \Delta_1$. Let $\tau = \bigwedge \Theta' \to p$.

By Lemma 16, there is a type binding $F : \tau' \in \Delta_0$ such that $\tau' \sqsubseteq \tau$. Since the type σ is canonical, there is k such that $\sigma \Uparrow k = \tau'$. Therefore $(\sigma \Uparrow k) \sqsubseteq \tau$. □

We define Δ as $\{F : \sigma \mid \exists k, \tau.\ (\sigma \Uparrow k) \sqsubseteq \tau \text{ and } F : \tau \in \Delta_1\}$.

Lemma 18. *$\Delta_C \subseteq \Delta$.*

Proof. A direct consequence of Lemma 17. □

Since Δ is finite, we can decide whether $\mathcal{G}$ is typable in $\mathcal{T}_C$. Moreover, if it is typable, we can construct a finite witness.

D Details of the Construction in the Proof of Theorem 10

Let $M = (Q, \Sigma, \Gamma, \delta)$ be a superdeterministic PDA of delay d_0 with the initial type ι and $d = d_0 + 1$. Here we construct the PDA $M' = (Q', \Sigma, \Gamma, \delta')$ in the proof of Theorem 10 and show its properties. The transition of M is normalized as follows:

$$\begin{aligned}(q, \widetilde{B} A_d \dots A_1) &\Vdash^{a}_{M'} (\langle q, a\rangle, \widetilde{B}\widetilde{A})\\ &\Vdash^{\varepsilon}_{M'} (\langle q, a, A_1\rangle, \widetilde{B} A_d \dots A_2)\\ &\Vdash^{\varepsilon}_{M'} (\langle q, a, A_2 A_1\rangle, \widetilde{B} A_d \dots A_3)\\ &\ \vdots\\ &\Vdash^{\varepsilon}_{M'} (\langle q, a, A_d \dots A_1\rangle, \widetilde{B})\\ &\Vdash^{\varepsilon}_{M'} (q', \widetilde{B}\widetilde{C}).\end{aligned}$$

We assume that M has a special stack symbol $\bot$ such that $(q, \bot) \nVdash^{a}_{M}$ and $(q, \bot) \nVdash^{\varepsilon}_{M}$, i.e., M gets stuck if it sees $\bot$. For a stack symbol A, we write A^n for $\overbrace{A \dots A}^{n}$.

The set Q' of states of M' is defined by

$$\begin{aligned}Q' = &\{q \mid q \in Q\}\\ &\cup \{\langle q, a\rangle \mid q \in Q, a \in \Sigma\}\\ &\cup \{\langle q, a, A_i \dots A_1\rangle \mid q \in Q, a \in \Sigma, i \leq d, \forall j \leq i (A_j \in \Gamma)\}\end{aligned}$$

and the transition relation is given by $\delta' = \delta'_1 \cup \delta'_2 \cup \delta'_3 \cup \delta'_4$, where

$$\begin{aligned}\delta'_1 &= \{(q, A_1, a, \langle q, a\rangle, A_1) \mid q \in Q, A_1 \in \Gamma, a \in \Sigma\}\\ \delta'_2 &= \{(\langle q, a\rangle, A_1, \varepsilon, \langle q, a, A_1\rangle, \varepsilon) \mid q \in Q, a \in \Sigma, A_1 \in \Gamma\}\\ \delta'_3 &= \{(\langle q, a, A_{i-1} \dots A_1\rangle, A_i, \varepsilon, \langle q, a, A_i \dots A_1\rangle, \varepsilon)\\ &\qquad \mid q \in Q, a \in \Sigma, i \leq d, \forall j \leq i (A_j \in \Gamma)\}\\ \delta'_4 &= \{(\langle q, a, A_d \dots A_1\rangle, B, \varepsilon, q', B\widetilde{C})\\ &\qquad \mid q \in Q, a \in \Sigma, \forall i \leq d (A_i \in \Gamma), B \in \Gamma,\ (q, A_d \dots A_1) \vDash^{a}_{M} (q', \widetilde{C})\}.\end{aligned}$$

The automaton M' records the letter on its state by δ'_1, records stack symbols by δ'_2 and δ'_3, and computes the next configuration from information recorded on the state by δ'_4. The definition of δ'_4 uses the transition of M.

Lemma 19. *Let $q, q' \in Q$, $a \in \Sigma$ and $\widetilde{A}, \widetilde{B}$ be sequences of stack symbols. Then $(q, \widetilde{A}) \vDash^{a}_{M} (q', \widetilde{B})$ if and only if $(q, \bot^d \widetilde{A}) \vDash^{a}_{M'} (q', \bot^d \widetilde{B})$.*

Proof. By the definition of δ' and the fact that $(q, \widetilde{A}) \vDash^{a}_{M} (q', \widetilde{B})$ if and only if $(q, \bot\widetilde{A}) \vDash^{a}_{M} (q', \bot\widetilde{B})$. To add $\bot^d$ to the stack is needed for the case that the length of $\widetilde{A}$ is less than or equal to d. □

Corollary 1. $\mathcal{L}_M(\iota) = \mathcal{L}_{M'}(\iota \Uparrow \bot^d)$. □

Then we define a $\mathcal{C}$-machine $\natural(M')$ and show that M' is a refinement of $\natural(M')$. Intuitively, $\natural(M')$ is given by forgetting stack symbols of M. We write $\star$ for the unique stack symbol of $\natural(M')$. The states $\natural(Q')$ of $\natural(M')$ is given by

$$\begin{aligned}\natural(Q') = \{q \mid q \in Q\} \\ \cup \{\langle q, a\rangle \mid q \in Q, a \in \Sigma\} \\ \cup \{\langle q, a, i\rangle \mid q \in Q, a \in \Sigma, 1 \le i \le d\}\end{aligned}$$

and the transition relation $\natural(\delta')$ is given by

$$\begin{aligned}\natural(\delta') = \{(q, \star, a, \langle q, a\rangle, \star) \mid q \in Q, a \in \Sigma\} \\ \cup \{(\langle q, a\rangle, \star, \varepsilon, \langle q, a, 1\rangle, \varepsilon) \mid q \in Q, a \in \Sigma\} \\ \cup \{(\langle q, a, i-1\rangle, \star, \varepsilon, \langle q, a, i\rangle, \varepsilon) \mid q \in Q, a \in \Sigma, i \le d\} \\ \cup \{(\langle q, a, d\rangle, \star, \varepsilon, q', \star\star^n) \\ \mid q \in Q, a \in \Sigma, \exists \widetilde{A}, \widetilde{C} \in \Gamma^* \big(|\widetilde{A}| = d \text{ and } (q, \widetilde{A}) \vDash^a_M (q', \widetilde{C}) \text{ and } |\widetilde{C}| = n\big)\},\end{aligned}$$

where $|\widetilde{A}|$ is the length of the sequence $\widetilde{A}$. In the last rule, n is uniquely determined for each q and a because M is superdeterministic.

We define two mappings $\natural^Q : Q' \to \natural(Q')$ and $\natural^\Gamma : \Gamma \to \{\star\}$ by

$$\begin{aligned}\natural^Q(q) &= q \\ \natural^Q(\,\langle q, a\rangle\,) &= \langle q, a\rangle \\ \natural^Q(\,\langle q, a, A_i \dots A_1\rangle\,) &= \langle q, a, i\rangle \\ \\ \natural^\Gamma(A) &= \star.\end{aligned}$$

The following lemma is easy to show.

Lemma 20. *The pair $(\natural^Q, \natural^\Gamma)$ is a homomorphism from M' to $\natural(M')$.* □

By the combination of Corollary 1 and Lemma 20, we conclude that every superdeterministic language is accepted by a refinement of a $\mathcal{C}$-machine.