Decision problems for Clark-congruential languages

Makoto Kanazawa[*†] *Hosei University, Tokyo, Japan* KANAZAWA@HOSEI.AC.JP

Tobias Kappé[*‡] *University College London, London, United Kingdom* TKAPPE@CS.UCL.AC.UK

Editors: Olgierd Unold, Witold Dyrka, and Wojciech Wieczorek

Abstract

A common question when studying a class of context-free grammars (CFGs) is whether equivalence is decidable within this class. We answer this question positively for the class of *Clark-congruential grammars*, which are of interest to grammatical inference. We also consider the problem of checking whether a given CFG is Clark-congruential, and show that it is decidable given that the CFG is a deterministic CFG.

Keywords: decidability, context-free grammars, Clark-congruential grammars

1. Introduction

Given two context-free grammars (CFGs), the *equivalence problem* asks whether they represent the same language; this is well known to be undecidable in general (Bar-Hillel et al., 1961). In contrast, the equivalence problem is decidable within some families of CFGs, such as deterministic CFGs (Sénizergues, 2001) and (pre-)NTS grammars (Sénizergues, 1985; Autebert and Boasson, 1992). Thus, a reasonable question to ask when studying a subclass of CFGs is whether equivalence is decidable for members of this class.

One subclass of CFGs of interest to grammatical inference consists of the CFGs considered in (Clark, 2010), which we refer to as *Clark-congruential (CC) grammars*. There it is shown that, given an oracle called the "teacher", an algorithm can infer a language known to the teacher by posing questions about the language in a fixed format. In particular, one type of question that the teacher can answer is an *equivalence query*, where the algorithm supplies a CFG and asks whether it represents the language that the teacher has in mind. A similar (if slightly less general) teacher can be used to infer regular languages (Angluin, 1987).

In analogy to other classes of CFGs, one might ask whether the equivalence problem for CC grammars is decidable; in analogy to regular languages, one might ask whether it is in principle possible to implement a teacher that answers equivalence queries for a CC grammar. Motivated by these questions, we investigate decision problems surrounding CC grammars. Our main contribution is a proof that equivalence and congruence problems for these grammars are decidable, based on arguments of that ilk for pre-NTS grammars (Autebert and Boasson, 1992). We also show that it is decidable whether a deterministic CFG is CC.

The remainder of this paper is organised as follows. In Section 2, we recall some preliminary notions. In Section 3, we discuss the congruence, equivalence and recognition

* This work performed at the National Institute of Informatics in Tokyo, Japan.
† Supported by JSPS KAKENHI Grant Number 17K00026.
‡ Partially supported by the ERC Starting Grant ProFoundNet (grant code 679127).

problems for CC grammars. We list directions for further work in Section 4. To preserve the narrative, some proofs appear in the technical report (Kanazawa and Kappé, 2018).

2. Preliminaries

A relation $R \subseteq S \times S$ is said to be *Noetherian* if it does not admit an infinite chain, i.e., there exist no infinite sequence $(s_n)_{n\in\mathbb{N}}$ such that for all $n \in \mathbb{N}$ it holds that $s_n \neq s_{n+1}$ and $s_n \; R \; s_{n+1}$. R is *confluent on* $S' \subseteq S$ if it is transitive and when for all $s, s', s'' \in S'$ such that $s \; R \; s'$ and $s \; R \; s''$, there exists a $t \in S'$ with $s' \; R \; t$ and $s'' \; R \; t$.

Words and languages We fix a finite set Σ, called the *alphabet*, and write Σ^* for the *language* of words over Σ. We write Γ for another finite alphabet that contains Σ, and the symbol \$, which is not in Σ. The *empty word* is denoted by ϵ. We write $|w|$ for the *length* of $w \in \Sigma^*$. We also fix an (arbitrary) total order $\preceq$ on Σ, and extend $\preceq$ to an order on Σ^* by defining $x \preceq y$ if and only if either $|x| < |y|$, or $|x| = |y|$ and x precedes y lexicographically. A *prefix* (resp. *suffix*) of $w \in \Sigma^*$ is a $w' \in \Sigma^*$ such that there exists a $y \in \Sigma^*$ with $w'y = w$ (resp. $yw' = w$); w *overlaps* with x if a non-empty suffix of w is a prefix of x, or vice versa.

A function $h : \Sigma^* \to \Sigma^*$ is a *morphism* when for $w, x \in \Sigma^*$ it holds that $h(wx) = h(w)h(x)$. If we define a function $h : \Sigma \to \Sigma^*$, then h uniquely extends to a morphism $h : \Sigma^* \to \Sigma^*$, by defining for $a_0, a_1, \ldots, a_{n-1} \in \Sigma$ that $h(a_0a_1 \cdots a_{n-1}) = h(a_0)h(a_1) \cdots h(a_{n-1})$. If for all $a \in \Sigma$ we have that $h(a) \in \Sigma$, we say that h is *strictly alphabetic*. When L is a language, we write $h^{-1}(L)$ for the language given by $\{w \in \Sigma^* : h(w) \in L\}$.

A *semi-Thue system* (Book and Otto, 1993) is a reflexive and transitive relation $\rightsquigarrow$ on Σ^* such that if $w \rightsquigarrow w'$ and $x \rightsquigarrow x'$, then $wx \rightsquigarrow w'x'$. A *reduction* is a Noetherian semi-Thue system. We say that $x \in \Sigma^*$ is *irreducible* by a reduction $\rightsquigarrow$ if $x \rightsquigarrow x'$ implies that $x = x'$.

A *congruence* is an equivalence $\sim$ on Σ^* such that when $u \sim v$ and $w \sim x$, also $uw \sim vx$. If $\sim$ is a congruence on Σ^*, we write $[w]_\sim$ for the *congruence class* containing $w \in \Sigma^*$. A congruence $\sim$ is *finitely generated* if for some finite $S \subseteq \Sigma^* \times \Sigma^*$, $\sim$ is the smallest congruence containing S; the set S is said to *generate* $\sim$. Any language L induces a *syntactic congruence*, denoted $\equiv_L$, which is the relation where $w \equiv_L x$ holds precisely when, for all $u, v \in \Sigma^*$, we have $uwv \in L$ if and only if $uxv \in L$. The *language of contexts* of $w \in \Sigma^*$ w.r.t. a language L, denoted $L[w]$, is $\{u\sharp v : uwv \in L\}$ (for a distinguished symbol $\sharp$). It should be clear that $w \equiv_L x$ if and only if $L[w] = L[x]$.

A language L is *congruential* (Book and Otto, 1993) if there exists a finitely-generated congruence $\sim$ and a finite set $T \subseteq \Sigma^*$ such that $L = \bigcup_{t\in T} [t]_\sim$. We say that L is *regular* if its syntactic congruence induces finitely many congruence classes (Nerode, 1958).

Decidability of congruence and of equivalence are closely related for congruential languages, as witnessed by the following lemma from (Sénizergues, 1985).

Lemma 1 *Let $\sim_1$ and $\sim_2$ be congruences generated by finite sets $S_1, S_2 \subseteq \Sigma^* \times \Sigma^*$ respectively, and let $T_1, T_2 \subseteq \Sigma^*$ be finite. Let L_1 and L_2 be given by*

$$L_1 = \bigcup_{t\in T_1} [t]_{\sim_1} \qquad\qquad L_2 = \bigcup_{t\in T_2} [t]_{\sim_2}$$

If we can decide L_1 and L_2, as well as $\equiv_{L_1}$ and $\equiv_{L_2}$, then we can decide whether $L_1 = L_2$.

Proof Observe that $L_1 = L_2$ precisely when $T_1 \subseteq L_2$ and $T_2 \subseteq L_1$, as well as $\sim_1 \subseteq \equiv_{L_2}$ and $\sim_2 \subseteq \equiv_{L_1}$. The first two inclusions are decidable, since T_1 and T_2 are finite, and L_1 and L_2 are decidable. The latter two inclusions are also decidable, for they are equivalent to checking whether $S_1 \subseteq \equiv_{L_2}$ and $S_2 \subseteq \equiv_{L_1}$. Thus, we can decide whether $L_1 = L_2$. ■

Context-free grammars A *(context-free) grammar* (*CFG*) is a tuple $G = \langle V, P, I\rangle$ where V is a finite set of symbols called *nonterminals* with $I \subseteq V$ the *initial nonterminals*, and $P \subseteq V \times (\Sigma \cup V)^*$ is a finite set of pairs called *productions*. We denote $\langle A, \alpha\rangle \in P$ by $A \to \alpha$. We use G to denote an arbitrary CFG $\langle V, P, I\rangle$, implicitly quantifying over all CFGs.

We write $\widehat{\Sigma}$ for the set $\Sigma \cup V$ and define $\Rightarrow_G$ as the smallest relation on $\widehat{\Sigma}^*$ such that for all $\alpha, \gamma \in \widehat{\Sigma}^*$ and $B \to \beta \in P$, we have $\alpha B \gamma \Rightarrow_G \alpha\beta\gamma$. For $\alpha \in \widehat{\Sigma}^*$, the *language of* α *in* G, denoted $L(G, \alpha)$ is $\{w \in \Sigma^* : \alpha \Rightarrow_G^* w\}$; the *language of* G, denoted $L(G)$, is $\bigcup_{A \in I} L(G, A)$. We say that $L \subseteq \Sigma^*$ is a *context-free language* (*CFL*) if $L = L(G)$ for some CFG G.

As an example of a CFG, let us fix $G_D = \langle V_D, P_D, I_D\rangle$ as a CFG over the alphabet $\{\texttt{[}, \texttt{]}\}$, where $V_D = I_D = \{S\}$, and P_D contains the rules $S \to \epsilon$ and $S \to \texttt{[}S\texttt{]}$ and $S \to SS$. The language of G_D is the well-known *Dyck language*, which consists of strings of well-nested parentheses, and which we shall use as a recurring example throughout this paper.

If $L(G, \alpha)$ is non-empty, we write $\vartheta_G(\alpha)$ for the $\preceq$-minimum of $L(G, \alpha)$. Now, if $L(G, \alpha\beta)$ is non-empty, then $\vartheta_G(\alpha\beta) = \vartheta_G(\alpha)\vartheta_G(\beta)$. We define $\rightsquigarrow_G$ as the smallest semi-Thue system such that whenever $A \to \alpha \in P$ and $L(G, \alpha) \neq \emptyset$, also $\vartheta_G(\alpha) \rightsquigarrow_G \vartheta_G(A)$. As an example, for G_D we see that $\vartheta_{G_D}(S) = \epsilon$, and hence $\rightsquigarrow_{G_D}$ is generated solely by the rule $\texttt{[]} \rightsquigarrow_{G_D} \epsilon$.

We observe that $\rightsquigarrow_G$ is a reduction (regardless of G), and that for all $A \in V$ and $w \in L(G, A)$ it holds that $w \rightsquigarrow_G \vartheta_G(A)$. We write $\mathcal{I}_G$ for the set of words irreducible by $\rightsquigarrow_G$. Note that $\mathcal{I}_G$ is regular: it is the complement of the regular language of words containing the left-hand side of a rule defining $\rightsquigarrow_G$, and regular languages are closed under complementation. For instance, it is not hard to see that $\mathcal{I}_{G_D} = \{\texttt{]}^n\texttt{[}^m : n, m \geq 0\}$.

We say that G is *weakly ω-reduced* when for $A \in V \setminus I$ we have that $L(G, A)$ is infinite, and for all productions $A \to \alpha$ where $L(G, A)$ is finite, we have that $\alpha \in \Sigma^*$.

Lemma 2 [1] *Let $G = \langle V, P, I\rangle$ be a CFG, let R be a regular language and let $h : \Sigma^* \to \Sigma^*$ be a strictly alphabetic morphism. All of the following hold:*

(i) We can construct a weakly ω-reduced CFG $G_\omega = \langle V_\omega, P_\omega, I_\omega\rangle$ such that $L(G_\omega) = L(G)$ and $V_\omega \subseteq V$; moreover, when $A \in V_\omega$ it holds that $L(G, A) = L(G_\omega, A)$.

(ii) We can construct a CFG $G^h = \langle V^h, P^h, I^h\rangle$ such that $L(G^h) = h^{-1}(L(G))$ and $V^h \subseteq V$; moreover, when $A \in V^h$ it holds that $h^{-1}(L(G, A)) = L(G^h, A)$.

(iii) We can construct a CFG $G_R = \langle V_R, P_R, I_R\rangle$ such that $L(G_R) = L(G) \cap R$; moreover, when $A \in V_R$ there exist $A' \in V$ and $w \in \Sigma^*$ such that $L(G_R, A) = L(G, A') \cap [w]_{\equiv_R}$.

Pushdown automata A *pushdown automaton* (*PDA*) is a tuple $M = \langle Q, \to, q^0, F\rangle$ where Q is a finite set of *states*, $q^0 \in Q$ is the *initial state*, $F \subseteq Q$ are the *accepting states* and $\to \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \times \Gamma^* \times Q$ is the (finite) *transition relation*. When $\langle q, a, \sigma, \rho, q'\rangle \in \to$, we write $q \xrightarrow{a, \sigma/\rho} q'$. The set of *configurations* of M, denoted $\mathcal{C}_M$, is $Q \times \Sigma^* \times \Gamma^*$. We define

1. Details appear in the technical report (Kanazawa and Kappé, 2018).

$\models_M$ as the smallest relation on $\mathcal{C}_M$ such that whenever $q \xrightarrow{a,\sigma/\rho} q'$ and $w \in \Sigma^*$ as well as $\tau \in \Gamma^*$, it holds that $\langle q, aw, \sigma\tau\rangle \models_M \langle q', w, \rho\tau\rangle$. The *language* of M, denoted $L(M)$, is[2]

$$\left\{w \in \Sigma^* : \left\langle q^0, w, \$\right\rangle \models_M^* \langle q, \epsilon, \$\rangle\,,\ q \in F\right\}$$

M is a *deterministic PDA* if, (i) for all $q \in Q$, $a \in \Sigma \cup \{\epsilon\}$, and $\sigma \in \Gamma$, there is at most one $\rho \in \Gamma^*$ and at most one $q' \in Q$ such that $q \xrightarrow{a,\sigma/\rho} q'$, and, (ii) for all $q' \in Q$ and $\rho \in \Gamma^*$ such that $q \xrightarrow{\epsilon,\sigma/\rho} q'$, there are no $q'' \in Q$, $a \in \Sigma$ and $\rho' \in \Gamma^*$ such that $q \xrightarrow{a,\sigma/\rho'} q''$.

If M is a PDA and L a language such that $L(M) = L$, we say that M *accepts* L. It is well-known that a language is a CFL if and only if it is accepted by a PDA (Chomsky, 1962). A language accepted by a deterministic PDA is said to be a *deterministic CFL* (*DCFL*). A CFG G whose language is a DCFL is said to be a *deterministic CFG* (DCFG).

As an example of a PDA, consider $M_D = \langle\{q\}, \to_D, q, \{q\}\rangle$, where $\to_D$ contains the rules $q \xrightarrow{\texttt{[},\$/\texttt{[}\$}_D q$, $q \xrightarrow{\texttt{[},\texttt{[}/\texttt{[[}}_D q$ and $q \xrightarrow{\texttt{]},\texttt{[}/\epsilon}_D q$. This PDA happens to be deterministic, and it is not hard to see that it accepts the Dyck language, $L(G_D)$; this makes G_D a DCFG.

3. Clark-congruential languages

We now turn our attention to *Clark-congruential languages*. These are context-free languages that are defined by grammars where every nonterminal has a language that is contained in a congruence class of its grammar; more formally, we work with the following definition.

Definition 3 *G is* Clark-congruential *(CC) if for all $A \in V$, there exists an $x_A \in \Sigma^*$ s.t. $L(G, A)$ is a subset of $[x_A]_{\equiv_{L(G)}}$. A language L is CC if $L = L(G)$ for a CC grammar G.*

As an example of a CC grammar, consider G_D. There, we find that if $w \in L(G_D, S)$, then w consists of a string of balanced parentheses; hence, if $uwv \in L(G_D, S)$, then $uv \in L(G_D, S)$, and vice versa. Consequently, it holds that for $w \in L(G_D, S)$ we have $w \equiv_{L(G_D)} \epsilon$.

CC grammars can be seen as a generalization of pre-NTS grammars (Autebert and Boasson, 1992), which are themselves a generalization of NTS grammars (Boasson, 1980; Sénizergues, 1985; Boasson and Sénizergues, 1985). While the class of CC *grammars* strictly contains the class of pre-NTS grammars, and thus the class of pre-NTS languages is contained in the class of CC languages, it remains an open question whether this inclusion is strict on the level of languages; likewise, the class of NTS grammars is contained in the class of pre-NTS grammars, but the question of equal expressiveness remains open.

3.1. Congruence and equivalence

We now consider the question of deciding equivalence of CC grammars. Our strategy here will be to verify the preconditions of Lemma 1 w.r.t. CC languages. Thus, our first task is to show that all CC languages are congruential; this is indeed the case.

Lemma 4 *If L is a CC language, then L is congruential.*

2. This definition is non-standard, in that upon acceptance the machine should be in an accepting state, *and* the stack contains exactly \$. A (D)PDA with this acceptance condition can easily be converted into an equivalent (D)PDA with the standard acceptance condition, provided that its transitions preserve the end-of-stack marker; this is the case for all DPDAs in this paper. We omit details for the sake of brevity.

Proof Let G be a CC grammar such that $L = L(G)$ and choose $\sim$ as the smallest congruence containing $\rightsquigarrow_G$. Obviously, $\sim$ is finitely generated. We now claim that

$$L(G) = \bigcup_{A \in I,\, L(G,A) \neq \emptyset} [\vartheta_G(A)]_\sim$$

For the inclusion from left to right, note that if $w \in L(G, A)$ for an $A \in I$, then $w \rightsquigarrow_G \vartheta_G(A)$, and hence $w \sim \vartheta_G(A)$; thus, $w \in [\vartheta_G(A)]_\sim$. For the other inclusion, note that since G is CC, $\sim \subseteq \equiv_{L(G)}$. Hence, if $w \sim \vartheta_G(A)$ with $A \in I$, then $w \equiv_{L(G)} \vartheta_G(A)$, and thus $w \in L(G)$. ■

We use G to denote an arbitrary CC grammar, and set out to validate the second assumption of Lemma 1, i.e., to show that if G is CC, then $\equiv_{L(G)}$ is decidable. To this end, we observe the following; details are in the technical report (Kanazawa and Kappé, 2018).

Lemma 5 *The grammar transformations from Lemma 2 preserve Clark-congruentiality.*

The algorithm that we describe to decide $\equiv_{L(G)}$ is essentially a generalization of the one found in (Autebert and Boasson, 1992). Before we dive into formal details, it helps to sketch a high-level roadmap of the steps required to establish the desired result, in analogy with the steps in op. cit. We proceed as follows:

(I) We argue that, when G is CC, $\rightsquigarrow_G$ is almost confluent: it can be used to decide $w \in L(G)$ by reducing w using any strategy, until we reach an irreducible word.

(II) We show that, for a given $w \in \Sigma^*$, we can use the transformations discussed earlier to construct a particular CC grammar G_w, which has a number of useful properties.

(III) From G_w, we create a DPDA M_w accepting a language very close to $L(G)[w]$; this DPDA exploits the almost-confluent nature of $\rightsquigarrow_{G_w}$ and the properties of G_w.

(IV) We argue that $w \equiv_{L(G)} x$ if and only if $L(M_w) = L(M_x)$. Since the latter is decidable (Sénizergues, 2001), we can decide the former.

Step (I): reduction is (almost) confluent If G is pre-NTS, then $\rightsquigarrow_G$ is confluent on $L(G)$, but not necessarily on Σ^* (Autebert and Boasson, 1992). For CC languages, this property is lost. As an example, consider the CC grammar G' with the rules $S \to aS$, $S \to a$, $T \to aaT$ and $T \to \epsilon$, and both S and T initial. We find that $\vartheta_{G'}(S) = a$ and $\vartheta_{G'}(T) = \epsilon$, and hence $aa \rightsquigarrow_{G'} a$ as well as $aa \rightsquigarrow_{G'} \epsilon$, but both a and ϵ are irreducible in $\rightsquigarrow_{G'}$.

On the positive side, $\rightsquigarrow_G$ is still useful in deciding membership of $L(G)$:

Lemma 6 *There exists an $A \in I$ with $x \rightsquigarrow_G \vartheta_G(A)$ if and only if $x \in L(G)$.*

Proof For the direction from left to right, note that if $x \rightsquigarrow_G \vartheta_G(A)$, then $x \equiv_{L(G)} \vartheta_G(A)$, and therefore $x \in [\vartheta_G(A)]_{\equiv_{L(G)}}$. Since $\vartheta_G(A) \in L(G)$, also $x \in L(G)$. For the other direction, note that if $x \in L(G)$, then $x \in L(G, A)$ for some $A \in I$, and therefore $x \rightsquigarrow_G \vartheta_G(A)$. ■

Using Lemma 6, we can simply apply reductions (using any strategy) to $w \in \Sigma^*$ from $\rightsquigarrow_G$, until we reach an irreducible word w_r. This process terminates, since $\rightsquigarrow_G$ is Noetherian. At

that point, either $w_r = \vartheta_G(A)$ for some $A \in I$, in which case $w \in L(G)$, or $w_r \neq \vartheta_G(A)$ for all $A \in I$, in which case $w_r \notin L(G)$ (since $w_r \in \mathcal{I}_G$), and since $w \equiv_{L(G)} w_r$, also $w \notin L(G)$.

As an example, consider the word `[[] []]`, which can be reduced using $\rightsquigarrow_{G_D}$ as follows:

$$\texttt{[[]}\,\underline{\texttt{[]}}\,\texttt{]}\,\texttt{[]} \rightsquigarrow_{G_D} \texttt{[}\,\underline{\texttt{[]}}\,\texttt{]}\,\texttt{[]} \rightsquigarrow_{G_D} \texttt{[]}\,\underline{\texttt{[]}} \rightsquigarrow_{G_D} \underline{\texttt{[]}} \rightsquigarrow_{G_D} \epsilon = \vartheta_{G_D}(S)$$

And hence `[[] []]` $\in L(G_D)$. On the other hand, the word `[[]` can be reduced to `[` only, and therefore Lemma 6 allows us to conclude that `[[]` $\notin L(G_D)$.

The (implicit) precondition that G is CC is necessary to establish Lemma 6. As an example, consider the grammar G' with rules $S \to a$, $S \to b$ and $T \to ab$, with both S and T initial. This grammar is not CC. If we assume that $a \preceq b$, then $\rightsquigarrow_G$ is generated by the rule $b \rightsquigarrow_G a$. We then find that $bb \rightsquigarrow_G ab$ and $\vartheta_G(T) = ab$, while $bb \notin L(G)$.

Step (II): construct G_w We now proceed to construct a CC grammar G_w from G. This is done by progressively applying the CC-preserving transformations described in Lemma 2.

First, we augment Σ by adding for $a \in \Sigma$ the (unique) letter a', i.e., every letter gains a "primed" version; this does not change $L(G)$, or the fact that G is CC. We write Σ_0 for the original alphabet, and Σ_1 for the set of newly added letters. Moreover, let $h : \Sigma^* \to \Sigma^*$ be the morphism that removes the primes from $w \in \Sigma^*$, i.e., the morphism defined by setting $h(a) = a$ for $a \in \Sigma_0$ and $h(a') = a$ for $a' \in \Sigma_1$. We write w' for the "primed copy" of w, i.e., the unique element of Σ_1^* such that $h(w') = w$. We proceed to define G_w in steps, as follows:

- Let $G' = \langle V', P', I' \rangle$ be such that $L(G') = h^{-1}(L(G))$.
- Let $G'_w = \langle V'_w, P'_w, I'_w \rangle$ be such that $L(G'_w) = L(G') \cap Rw'R$, where $R = \mathcal{I}_G \cap \Sigma_0^*$.
- Let $G_w = \langle V_w, P_w, I_w \rangle$ be such that $L(G_w) = L(G'_w)$, and G_w is weakly ω-reduced.

By Lemma 2, these grammars are CC. Without trying to get ahead of ourselves, we note that $L(G_w)$ is already somewhat close to $L[w]$. After all, we know that $L(G_w) = \{uw'v : u, v \in \mathcal{I}_G, uwv \in L\}$. The difference between $L[w]$ and $L(G_w)$ comes down to having $\sharp$ or w' separate the parts of the words, and whether those parts need to be in $\mathcal{I}_G$.

Some analysis of G'_w now gives us the following.

Lemma 7 *Let $A \in I'_w$. If $L(G'_w, A) \cap \Sigma_0^* \neq \emptyset$ and $w' \neq \epsilon$, then $L(G'_w, A) = \{\vartheta_G(A)\}$.*

Proof Suppose that $y \in L(G'_w, A) \cap \Sigma_0^*$. First note that we can (without loss of generality) find $u, v \in \Sigma^*$ such that $uL(G'_w, A)v \subseteq L(G'_w) \subseteq Rw'R$. Consequently, there exist $p, q \in R$ such that $uyv = pw'q$. Since $w' \neq \epsilon$, this means that y is a substring of p or q, and thus $y \in R$. For the remainder, it suffices to show that $y = \vartheta_G(A)$, and $L(G'_w, A) \setminus \Sigma_0^* = \emptyset$.

First, note that $y \in L(G', A)$, and so $h(y) = y \in L(G, A)$; thus, $y \rightsquigarrow_G \vartheta_G(A)$. Since $y \in \mathcal{I}_G$, we have $y = \vartheta_G(A)$. Also, suppose towards a contradiction that $z \in L(G'_w, A) \setminus \Sigma_0^*$. Then z contains at least one primed letter. By choice of u and v, we find that $uzv \in L(G'_w)$. Now uzv contains strictly more primed letters than uyv; since all words in $L(G'_w)$ contain exactly $|w'|$ primed letters, we have reached a contradiction. We conclude that $L(G'_w, A) \setminus \Sigma_0^* = \emptyset$. ■

Since G_w is the weakly ω-reduced version of G'_w, we can show the following:

Lemma 8 *Let $A \to \alpha \in P_w$ with $L(G_w, \alpha) \neq \emptyset$. Then $\vartheta_{G_w}(A)$ and $\vartheta_{G_w}(\alpha)$ either contain or share an overlap with w'; more formally, one of the following holds:*

(i) $\vartheta_{G_w}(A) = x_A w'_\ell$ *and* $\vartheta_{G_w}(\alpha) = x_\alpha w'_\ell$, *for* $x_A, x_\alpha \in \Sigma_0^*$ *and* w'_ℓ *a nonempty prefix of* w'

(ii) $\vartheta_{G_w}(A) = w'_r y_A$ *and* $\vartheta_{G_w}(\alpha) = w'_r y_\alpha$, *for* $y_A, y_\alpha \in \Sigma_0^*$ *and* w'_r *a nonempty suffix of* w'

(iii) $\vartheta_{G_w}(A) = x_A w' y_A$ *and* $\vartheta_{G_w}(\alpha) = x_\alpha w' y_\alpha$, *for* $x_A, y_A, x_\alpha, y_\alpha \in \Sigma_0^*$.

Proof If $L(G_w, A)$ is finite, then $A \in I_w$ (since G_w is weakly ω-reduced), and therefore $\vartheta_{G_w}(A), \vartheta_{G_w}(\alpha) \in L(G_w) \subseteq \mathcal{I}_G w' \mathcal{I}_G$; thus, $\vartheta_{G_w}(A)$ and $\vartheta_{G_w}(\alpha)$ satisfy the third condition.

Otherwise, suppose that $L(G_w, A)$ is infinite. First, note that there exist $x, y \in \Sigma^*$ such that $xL(G_w, A)y \subseteq L(G_w)$. Thus, there exist $u, v \in \mathcal{I}_G \subseteq \Sigma_0^*$ such that $x\vartheta_{G_w}(A)y = uw'v$. Suppose, towards a contradiction, that $\vartheta_{G_w}(A)$ neither contains nor overlaps with w'. In that case, $\vartheta_{G_w}(A) \in \Sigma_0^*$, and $w' \neq \epsilon$; then, since $A \Rightarrow_{G_w}^* \vartheta_{G_w}(A)$, also $A \Rightarrow_{G'_w}^* \vartheta_{G_w}(A)$. By Lemma 7, we have that $L(G'_w, A)$ is finite. But since $L(G'_w, A) = L(G_w, A)$ and the latter is infinite, we have a contradiction. Therefore $\vartheta_{G_w}(A)$ must contain or overlap with w'.

Suppose $\vartheta_{G_w}(A) = x_A w'_\ell$ for $x_A \in \Sigma_0^*$ and w'_ℓ a nonempty prefix of w'; other cases are similar. Write $w' = w'_\ell w'_r$ and $y = w'_r v$. By choice of x and y, we have $x\vartheta_{G_w}(\alpha)w'_r v = x\vartheta_{G_w}(\alpha)y \in L(G_w) \subseteq \mathcal{I}_G w' \mathcal{I}_G$. Therefore, $\vartheta_{G_w}(\alpha) = x_\alpha w'_\ell$ for some $x_\alpha \in \Sigma_0^*$. ■

This lemma tells us something about $\rightsquigarrow_{G_w}$: all of its generating rules overlap with w', and moreover each rule preserves w'. Thus, to decide whether $uw'v \in L(G_w)$, we can apply the rules of $\rightsquigarrow_{G_w}$ as described above; since every step involves (and preserves) part of w', we also know that reductions must be clustered around the locus of w'.

Step (III): creating a DPDA The above analysis allows us to construct a DPDA that accepts $\{u\sharp v : uw'v \in L(G_w)\}$, by going through the following phases:

1. Read symbols and push them on the stack, until we encounter $\sharp$.
2. From that point on, read from the stack or the input and apply reductions whenever possible, but with $\sharp$ standing in for the part of w'.
3. When no reductions are possible (i.e., we have reached an element if $\mathcal{I}_{G_w}$), check whether the buffer corresponds to a $\vartheta_{G_w}(A)$ for some $A \in I_w$.

In the second step, the state of the DPDA holds a buffer to the left and the right of $\sharp$, large enough to detect any possible reductions. Since $\rightsquigarrow_{G_w}$ is Noetherian, this phase must end after finitely many reductions; furthermore, since $\rightsquigarrow_{G_w}$ is length-decreasing, we can choose the size of the buffer appropriately. Formally, this DPDA is defined as follows:

Definition 9 *We build the PDA $M_w = \langle Q, \to, q_0, F\rangle$ as follows. First, let N be the maximum length of $\vartheta_{G_w}(\alpha)$ for $A \to \alpha$ in G_w. Also, Q and F are the smallest sets satisfying*

$$\frac{}{q_0 \in Q} \qquad \frac{u, v \in \Sigma_0^* \qquad |u|, |v| \leq N}{u\sharp v \in Q} \qquad \frac{A \in I_w \qquad \vartheta_{G_w}(A) = uw'v}{u\sharp v \in F}$$

Furthermore, $\to$ is the smallest transition relation satisfying

$$\frac{a \neq \sharp}{q_0 \xrightarrow{b,\, a/ba} q_0} \qquad \frac{}{q_0 \xrightarrow{\sharp,\, a/a} \sharp} \qquad \frac{u\sharp v \in Q \qquad |u| < N \qquad uw'v \in \mathcal{I}_{G_w} \qquad a \neq \$}{u\sharp v \xrightarrow{\epsilon,\, a/\epsilon} au\sharp v}$$

$$\frac{u\sharp v \in Q \qquad |v| < N \qquad uw'v \in \mathcal{I}_{G_w} \qquad a = \$ \vee |u| = N}{u\sharp v \xrightarrow{b,\, a/a} u\sharp vb}$$

$$\frac{u\sharp v \in Q \qquad uw'v \notin \mathcal{I}_{G_w} \qquad uw'v \rightsquigarrow_{G_w} xw'y \text{ such that } xy \text{ is } \preceq\text{-minimal}}{u\sharp v \xrightarrow{\epsilon,\, a/a} x\sharp y}$$

The first two rules take care of the first phase, where input is read onto the stack until we reach $\sharp$. The third and fourth rule are responsible for reading symbols from the stack and from the input buffer respectively; the last rule applies reductions. The set of accepting states makes sure that, upon acceptance, the buffer represents $\vartheta_{G_w}(A)$ for an $A \in I_w$.

We note that M_w is deterministic: if M_w is in state q_0, then the input is either equal to $\sharp$ (in which case the first rule applies) or not (in which case the second rule applies); otherwise, we are in some state $u\sharp v$, then either $uw'v \notin \mathcal{I}_{G_w}$ (and so the last rule applies), or the (mutually exclusive) third or fourth rule apply.

We can then show that M_w indeed accepts $\{u\sharp v : uw'v \in L(G_w)\}$. We give a sketch of the proof below; details are in Appendix A.

Lemma 10 $L(M_w) = \{u\sharp v : uw'v \in L(G_w)\}$.

Proof sketch For the inclusion from left to right, show that every change in configuration of M_w corresponds to a step in the reduction of the input according to $\rightsquigarrow_{G_w}$, and that a configuration where M_w accepts corresponds to this reduction reaching $\vartheta_{G_w}(A)$ for $A \in I_w$.

For the other inclusion, first note that if $u\sharp v$ is such that $uw'v \in L(G_w)$, we can let M_w read up to and including $\sharp$, putting u on the stack. Subsequently, inspect the halting configuration reached by M_w from that point on (which exists uniquely, for $\models_{M_w}$ is Noetherian), and show that it is a state where M_w can accept — i.e., that the remaining input and stack is empty, and that the buffer corresponds to an accepting state of M_w. ■

Step (IV): wrapping up Now we can show the following.

Lemma 11 $L(M_w) = L(M_x)$ *if and only if* $w \equiv_{L(G)} x$.

Proof For the direction from left to right, suppose that $L(M_w) = L(M_x)$, and that $uwv \in L(G)$. We can then find $u', v' \in \mathcal{I}_G$ such that $u \rightsquigarrow_G u'$ and $v \rightsquigarrow_G v'$. Now, since G is CC and $u \equiv_{L(G)} u'$ and $v \equiv_{L(G)} v'$, we know that $u'wv' \in L(G)$. Consequently, $u'\sharp v' \in L(M_w) = L(M_x)$, and therefore $u'xv' \in L(G)$, meaning that $uxv \in L(G)$. By symmetry, $uxv \in L(G)$ also implies $uwv \in L(G)$; this allows us to conclude that $w \equiv_{L(G)} x$.

For the other direction, suppose that $y \in L(M_w)$. Then $y = u\sharp v$ such that $u, v \in \mathcal{I}_G$, and $uwv \in L(G)$. Since $w \equiv_{L(G)} x$, it then follows that $uxv \in L(G)$, and thus $y = u\sharp v \in L(M_x)$. This shows that $L(M_w) \subseteq L(M_x)$; the other inclusion follows symmetrically. ■

The above characterises the syntactic congruence of $L(G)$ in terms of the equivalence of two DPDAs, constructible from G, w and x. Since equivalence of DPDAs is decidable (Sénizergues, 2001), it follows that we can decide $\equiv_{L(G)}$. The main result then follows.

Theorem 12 *It is decidable, given a CFG G that is CC and $w, x \in \Sigma^*$, whether $w \equiv_{L(G)} x$. It is furthermore decidable, given CFGs G and G' that are CC, whether $L(G) = L(G')$.*

Like in (Autebert and Boasson, 1992), M_w is *one-turn*, i.e., it processes input first in a phase where the stack does not shrink (when it is still in q^0), and subsequently in a phase where the stack does not grow (in all other states). Thus, an algorithm to test equivalence of finite-turn DPDAs (Valiant, 1974; Beeri, 1975) suffices. Complexity-wise, this also helps: the equivalence problem for one-turn DPDAs is known to be in co-NP (Sénizergues, 2003), while the problem for general DPDAs is known only to be primitive recursive (Stirling, 2002).

3.2. Recognition

The *recognition problem* for a class of CFGs $\mathcal{C}$ asks, given a CFG G, whether G is in $\mathcal{C}$. This problem is decidable for NTS grammars (Sénizergues, 1985), yet undecidable for a proper subclass of pre-NTS grammars (Zhang, 1992).[3]

Given that our earlier decidability proofs were based on proofs of the same statement for pre-NTS grammars, one might ask whether we could extend the result from (Zhang, 1992) to CC grammars. This turns out not to be the case. The proof in op. cit. constructs, given a Turing machine M and an input w, a CFG which is in the studied class if and only if M does not halt on input w; this construction relies heavily on adding nonterminals with an empty language. However, we can easily adapt the first construction from Lemma 2 to show that we can remove all such nonterminals from a CFG G to obtain an (equivalent) CFG G'; furthermore, G is CC if and only if G' is CC. Thus, to decide whether a given CFG is CC, we can assume without loss of generality that no nonterminal has an empty language. Hence, the undecidability proof from (Zhang, 1992) does not generalize to CC grammars.

We therefore turn our attention to finding a novel approach to the recognition problem for CC grammars, independent of (un)decidability proofs of the recognition problem for its subclasses. To this end, it is useful to introduce the following notion.

Definition 13 *Let $\sim$ be a congruence. G is* $\sim$-aligned *if, for every $A \in V$, there exists a $w_A \in \Sigma^*$ such that $L(G, A) \subseteq [w_A]_\sim$.*

Note that, by definition, G is CC if and only if it is $\equiv_{L(G)}$-aligned. As it turns out, $\sim$-alignment is decidable, provided that $\sim$ is decidable.

Lemma 14 *Given a decidable congruence $\sim$, it is decidable whether a CFG G is $\sim$-aligned.*

Proof Without loss of generality, assume that all nonterminals of G have a non-empty language; if this is not the case, we can create a CFG G' that does have this property, and which is $\sim$-aligned if and only if G is. Since $\vartheta_G : \widehat{\Sigma}^* \to \Sigma^*$ is computable, it now suffices to prove that G is $\sim$-aligned if and only if for all $A \to \alpha \in P$, it holds that $\vartheta_G(A) \sim \vartheta_G(\alpha)$.

3. We note that the class of CFGs considered in (Zhang, 1992) was originally claimed to coincide with pre-NTS grammars (Boasson and Sénizergues, 1985), but this is not strictly true: Zhang's class is a strict subclass of the pre-NTS grammars, although the languages that they can express are the same.

For the direction from left to right, we know that if $A \to \alpha \in P$, then $\vartheta_G(A), \vartheta_G(\alpha) \in L(G, A) \subseteq [w_A]_\sim$ for some $w_A \in \Sigma^*$; hence, $\vartheta_G(A) \sim w_A \sim \vartheta_G(\alpha)$. For the direction from right to left, a straightforward inductive argument shows that for all $\alpha, \beta \in \widehat{\Sigma}^*$ such that $\alpha \Rightarrow_G^* \beta$, we have that $\vartheta_G(\alpha) \sim \vartheta_G(\beta)$. Hence, if $A \Rightarrow_G^* w$, then we know that $\vartheta_G(A) \sim \vartheta_G(w) = w$, and thus it suffices to choose $w_A = \vartheta_G(A)$. ■

As an application of the above, let $\sim$ be the smallest congruence on $\{\texttt{[}, \texttt{]}\}^*$ such that $\texttt{[]} \sim \epsilon$. Without too much effort, we can then show that we can *uniquely* compute $m, n \in \mathbb{N}$ such that $w \sim \texttt{]}^m\texttt{[}^n$. Therefore, we can conclude that $\sim$ is decidable: to decide whether $w \sim x$, check whether the m and n computed for w are the same as the m and n computed for x. Thus, by Lemma 14, we find that we can decide whether a given grammar G over the alphabet $\{\texttt{[}, \texttt{]}\}^*$ is $\sim$-aligned. Indeed, $\sim$ turns out to be exactly $\equiv_{L(G_D)}$.

Lemma 14 would also show that the recognition problem for CC grammars is decidable, provided that the congruence problem were decidable for arbitrary CFGs. Unsurprisingly, this is not the case, as witnessed by the following lemma.

Lemma 15 *It is undecidable, given a CFG G and words $w, x \in \Sigma^*$, whether $w \equiv_{L(G)} x$.*

Proof We claim that $L(G) = \Sigma^*$ if and only if $\epsilon \in L(G)$, and for all $a \in \Sigma$ it holds that $a \equiv_{L(G)} \epsilon$. First, suppose $L(G) = \Sigma^*$; then $\epsilon \in L(G)$ immediately. Furthermore, for $a \in \Sigma$ and $u, v \in \Sigma^*$, we have that $uav, uv \in L(G)$, and thus $a \equiv_{L(G)} \epsilon$. For the other direction, let $w \in \Sigma^*$. An argument by induction on $|w|$ then shows that $w \equiv_{L(G)} \epsilon$, and hence $w \in L(G)$.

Since it is decidable whether $\epsilon \in L(G)$, the above equivalence tells us that we can decide $L(G) = \Sigma^*$ if we can decide the congruence problem for G. Because the former is undecidable for CFGs in general (Bar-Hillel et al., 1961), the claim follows. ■

Fortunately, some classes of CFGs do have a decidable congruence problem. This leads us to formulate our main result regarding the recognition problem, as follows.

Theorem 16 *It is decidable, given a DCFG G, whether G is CC.*

Proof Let us write $L = L(G)$. By Lemma 14, it suffices to show that we can effectively obtain a decision procedure for $\equiv_L$. We employ a technique similar to the method we used to decide $\equiv_L$ when G is CC: we reduce the problem to checking equivalence of DCFLs.

Without loss of generality, let $\Sigma = \Sigma_0 \cup \{\sharp\}$, with $\sharp \notin \Sigma_0$, such that $L \subseteq \Sigma_0^*$. For $w \in \Sigma^*$, we define the morphism $g_w : \Sigma^* \to \Sigma^*$ by setting $g_w(\sharp) = w$ and $g(a) = a$ for $a \in \Sigma_0$.

We now claim that $L[w] = g_w^{-1}(L) \cap \Sigma_0^*\sharp\Sigma_0^*$. To see this, suppose that $u\sharp v \in L[w]$; then, since $g_w(u\sharp v) = uwv \in L$ and $u\sharp v \in \Sigma_0^*\sharp\Sigma_0^*$, we find that $u\sharp v \in g_w^{-1}(L)$. For the other inclusion, suppose that $x \in g_w^{-1}(L) \cap \Sigma_0^*\sharp\Sigma_0^*$. Since $x \in \Sigma_0^*\sharp\Sigma_0^*$, we can write $x = u\sharp v$ for $u, v \in \Sigma_0^*$. Since $uwv = g_w(u\sharp v) = g(x) \in L$, we find that $u\sharp v \in L[w]$.

Since L is a DCFL, we have a DPDA M such that $L = L(M)$. Furthermore, because DCFLs are closed under inverse morphism and intersection with regular languages (Ginsburg and Greibach, 1966), we can create for $w \in \Sigma^*$ a DPDA M_w such that $L(M_w) = L[w]$. Since it is decidable whether $L(M_w) = L(M_x)$ (Sénizergues, 2001), we can decide whether $L[w] = L[x]$, and hence whether $w \equiv_L x$. ■

4. Further work

With regard to implementing a teacher for a given CC language, one detail remains to be settled. The algorithm to learn CC languages from (Clark, 2010) assumes the presence of an *extended MAT*, in which the representation of the language in the equivalence query need not guarantee that the hypothesis language is in the class of languages being learned. More concretely, this means that the algorithm might query the teacher with grammars that are not CC, and thus the decision procedure outlined in this paper need not apply. Consequently, we wonder whether the learning algorithm can be adapted to work with a (proper) MAT, or alternatively, whether the decision procedure of this paper can be extended to accommodate the class of grammars that can be produced by the learning algorithm.

One possible direction for generalization of the decision procedure is the setting of *multiple context-free grammars* (*MCFGs*) (Seki et al., 1991). A notion corresponding to Clark-congruentiality for MCFGs is already known, and the class of languages generated by such MCFGs is also known to be learnable (Yoshinaka and Clark, 2010). We conjecture that the decidability results can be lifted to Clark-congruential MCFGs, and that such a lifting would employ n-*turn* DPDAs instead of one-turn DPDAs.

Equivalence and congruence are decidable for both DCFLs and CC languages. To see if the case for CC languages follows from the case for DCFLs, one would have to investigate whether all CC grammars define a DCFL. For what it's worth, the fact that we can decide whether a DCFG is CC appears to at least not contradict this possibility, and we have been unsuccessful in finding a counterexample thus far.

The question about the connection between CC languages and DCFLs can be seen as analogous to the (open) question of whether all pre-NTS grammars define a DCFL (Autebert and Boasson, 1992). Since all NTS grammars are pre-NTS, and all pre-NTS grammars are in turn CC, it follows that every NTS language is a pre-NTS language, and in turn every pre-NTS language is a CC language; whether this inclusion is strict remains an open question. It has been conjectured that these families of languages coincide (Clark, 2010).

Acknowledgments

We would like to thank the anonymous referees of LearnAut and ICGI for their comments, which helped improve this paper.

References

Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75 (2):87–106, 1987. URL https://doi.org/10.1016/0890-5401(87)90052-6.

Jean-Michel Autebert and Luc Boasson. The equivalence of pre-NTS grammars is decidable. *Mathematical Systems Theory*, 25(1):61–74, 1992. URL https://doi.org/10.1007/BF01368784.

Yehoshua Bar-Hillel, Micha Perles, and Eli Shamir. On formal properties of simple phrase structure grammars. *Sprachtypologie und Universalienforschung*, 14:143–172, 1961.

Catriel Beeri. An improvement of Valiant's decision procedure for equivalence of deterministic finite-turn pushdown automata. In *Proc. Foundations of Computer Science (FOCS)*, pages 128–134, 1975. URL https://doi.org/10.1109/SFCS.1975.4.

Luc Boasson. Derivations et redutions dans les grammaires algebriques. In *Proc. Automata, Languages and Programming*, pages 109–118, 1980. URL https://doi.org/10.1007/3-540-10003-2_64.

Luc Boasson and Géraud Sénizergues. NTS languages are deterministic and congruential. *J. Comput. Syst. Sci.*, 31(3):332–342, 1985. URL https://doi.org/10.1016/0022-0000(85)90056-X.

Ronald V. Book and Friedrich Otto. *String-Rewriting Systems*. Texts and Monographs in Computer Science. Springer, 1993. ISBN 978-3-540-97965-4. URL https://doi.org/10.1007/978-1-4613-9771-7.

Noam Chomsky. Context-free grammars and pushdown storage. *MIT. Res. Lab. Electron. Quart. Prog. Report*, 65:187–194, 1962.

Alexander Clark. Distributional learning of some context-free languages with a minimally adequate teacher. In *Proc. Grammatical Inference (ICGI)*, pages 24–37, 2010. URL https://doi.org/10.1007/978-3-642-15488-1_4.

Seymour Ginsburg and Sheila A. Greibach. Deterministic context free languages. *Information and Control*, 9(6):620–648, 1966. URL https://doi.org/10.1016/S0019-9958(66)80019-0.

Makoto Kanazawa and Tobias Kappé. Decision problems for Clark-congruential languages, May 2018. URL https://arxiv.org/abs/1805.04402.

Anil Nerode. Linear automaton transformations. *Proc. American Mathematical Society*, 9 (4):541–544, 1958. URL https://doi.org/doi:10.2307/2033204.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context-free grammars. *Theor. Comput. Sci.*, 88(2):191–229, 1991. URL https://doi.org/10.1016/0304-3975(91)90374-B.

Géraud Sénizergues. The equivalence and inclusion problems for NTS languages. *J. Comput. Syst. Sci.*, 31(3):303–331, 1985. URL https://doi.org/10.1016/0022-0000(85)90055-8.

Géraud Sénizergues. $L(A) = L(B)$? Decidability results from complete formal systems. *Theor. Comput. Sci.*, 251(1-2):1–166, 2001. URL https://doi.org/10.1016/S0304-3975(00)00285-1.

Géraud Sénizergues. The equivalence problem for t-turn DPDA is co-NP. In *Proc. Automata, Languages and Programming (ICALP)*, pages 478–489, 2003. URL https://doi.org/10.1007/3-540-45061-0_39.

Colin Stirling. Deciding DPDA equivalence is primitive recursive. In *Proc. Automata, Languages and Programming (ICALP)*, pages 821–832, 2002.

Leslie G. Valiant. The equivalence problem for deterministic finite-turn pushdown automata. *Information and Control*, 25(2):123–133, 1974. URL https://doi.org/10.1016/S0019-9958(74)90839-0.

Ryo Yoshinaka and Alexander Clark. Polynomial time learning of some multiple context-free languages with a minimally adequate teacher. In *Proc. Formal Grammar (FG)*, pages 192–207, 2010. URL https://doi.org/10.1007/978-3-642-32024-8_13.

Louxin Zhang. The pre-NTS property is undecidable for CFGs. *Inf. Process. Lett.*, 44(4): 181–184, 1992. URL https://doi.org/10.1016/0020-0190(92)90082-7.

Appendix A. The language of M_w

To analyze the behavior of M_w, we first note that if it is in a configuration with a state of the form $u\sharp v$, then all reachable configurations are related to that configuration by $\rightsquigarrow_{G_w}$. In effect, this shows that M_w proceeds according to $\rightsquigarrow_{G_w}$.

Lemma 17 *If* $u_0, u_1, v_0, v_1, x_0, x_1, y_0, y_1 \in \Sigma^*$ *s.t.* $\langle u_0\sharp v_0, y_0, x_0^R\$\rangle \models_{M_w} \langle u_1\sharp v_1, y_1, x_1^R\$\rangle$ *then it follows that* $x_0u_0w'v_0z_0 \rightsquigarrow_{G_w} x_1u_1w'v_1y_1$.

Proof There are three cases to consider. First, if $uw'v$ is reducible, then $u_0w'v_0 \rightsquigarrow_{G_w} u_1w'v_1$, as well as $y_0 = y_1$ and $x_0 = x_1$; the claim then follows. Second, if $uw'v$ is irreducible and x_0 is non-empty, with $|u_0| < N$, then $x_0 = x_1a$ and $u_1 = au_0$, as well as $y_0 = y_1$ and $v_0 = v_1$; we derive that $x_0u_0w'v_0y_0 = x_1au_0w'v_0y_0 = x_1u_1w'v_0y_0 = x_1u_1w'v_1y_1$. Lastly, if $uw'v$ is irreducible and either x_0 is empty or $|u_0| = N$, then $ay_1 = y_0$ and $v_1 = v_0a$, as well as $x_0 = x_1$ and $u_0 = u_1$; thus, $x_0u_0w'v_0y_0 = x_0u_0w'v_0ay_1 = x_0u_0w'v_1y_1 = x_1u_1w'v_1y_1$. ■

With this in hand, we can show that M_w accepts the desired language.

Lemma 18 $L(M_w) = \{u\sharp v : uw'v \in L(G_w)\}$.

Proof For the inclusion from left to right, suppose that $x \in L(M_w)$. We then know that $\langle q_0, x, \$\rangle \models_{M_w}^* \langle u_1\sharp v_1, \epsilon, \$\rangle$ such that there exists an $A \in I$ with $\vartheta_{G_w}(A) = u_1w'v_1$. Thus, $x = u_0\sharp v_0$ such that $\langle q_0, u_0\sharp v_0, \$\rangle \models_{M_w}^* \langle \sharp, v_0, u_0^R\$\rangle \models_{M_w}^* \langle u_1\sharp v_1, \epsilon, \$\rangle$. By Lemma 17, we have $u_0w'v_0 \rightsquigarrow_{G_w} u_1w'v_1 = \vartheta_{G_w}(A)$. By Lemma 6, also $u_0w'v_0 \in L(G_w, A) \subseteq L(G_w)$.

For the inclusion from right to left, suppose that $u, v \in \Sigma^*$ are such that $uw'v \in L(G_w)$; our aim is to show that $u\sharp v \in L(M_w)$. By construction of M_w, this DPDA first processes the input up to $\sharp$ to reach $C^\sharp = \langle \sharp, \epsilon, v, u^R\$\rangle$.

Let $C = \langle u_1\sharp v_1, y, z^R\$\rangle$ be the unique halting configuration of M_w starting from $C^\sharp$; this configuration exists uniquely, because every transition of M_w either advances the input, or performs a reduction using $\rightsquigarrow_{G_w}$. We then have that $u_1w'v_1 \in \mathcal{I}_{G_w}$, otherwise C would not be halting. Now, we observe that (i) either z is empty, or $|u_1| = N$ — for otherwise M_w could pop letters off the stack into the left buffer, meaning that C would not be halting,

and (ii) either y is empty, or $|v_1| = N$ — for otherwise M_w could consume letters from the input into the right buffer, and so C would again not be halting.

A reducible substring of $zu_1w'v_1y$ must start at least N positions before the start of w', and end at most N positions from the end of w' (by Lemma 8). Consequently, $\rightsquigarrow_{G_w}$ cannot reduce (a) a substring overlapping z — otherwise $|u_1| < N$ and $z \neq \epsilon$, nor (b) a substring overlapping y — otherwise $|v_1| < N$ and $y \neq \epsilon$. Thus, if $zu_1w'v_1y$ were reducible, then the reducible substring must occur in $u_1w'v_1$ — but this is a contradiction, since $u_1w'v_1 \in \mathcal{I}_{G_w}$; hence, $zu_1w'v_1y$ is irreducible.

By Lemma 17, we know that $uw'v \rightsquigarrow_{G_w} zu_1w'v_1y$; also, by (the proof of) Lemma 4, we have that $uw'v \equiv_{L(G_w)} zu_1w'v_1y$, and hence $zu_1w'v_1y \in L(G_w)$. By Lemma 6, it follows that there exists an $A \in I$ such that $zu_1w'v_1y \rightsquigarrow_{G_w} \vartheta_{G_w}(A)$. Consequently, $\vartheta_{G_w}(A) = zu_1w'v_1y$, and so either $|u_1| < N$, and thus $z = \epsilon$, or $|u_1| = N$, in which case $z = \epsilon$ again, as $|\vartheta_{G_w}(A)| \leq N$. By a similar argument, we find that $y = \epsilon$. But then $\vartheta_{G_w}(A) = u_1w'v_1$, and thus $u_1\sharp v_1 \in F$. We can conclude that $u\sharp v \in L(M_w)$. ■