ON THE DENSITY OF LANGUAGES ACCEPTED BY TURING MACHINES AND OTHER MACHINE MODELS

Oscar H. Ibarra (A)  Ian McQuillan(B,C)

(A) Department of Computer Science, University of California, Santa Barbara, CA 93106, USA
ibarra@cs.ucsb.edu

(B) Department of Computer Science, University of Saskatchewan, 110 Science Place, Saskatoon, SK, Canada
mcquillan@cs.usask.ca

ABSTRACT

A language is dense if the set of all infixes (or subwords) of the language is the set of all words. Here, it is shown that it is decidable whether the language accepted by a nondeterministic Turing machine with a one-way read-only input and a reversal-bounded read/write worktape (the read/write head changes direction at most some fixed number of times) is dense. From this, it is implied that it is also decidable for one-way reversal-bounded queue automata, one-way reversal-bounded stack automata, and one-way reversal-bounded k-flip pushdown automata (machines that can “flip” their pushdowns up to k times). However, it is undecidable for deterministic Turing machines with two 1-reversal-bounded worktapes (even when the two tapes are restricted to operate as 1-reversal-bounded pushdown stacks).

Keywords: density, Turing machines, store languages, pushdowns, queues, stacks

1. Introduction

A language \( L \subseteq \Sigma^* \) is said to be dense if the set of all infixes of \( L \) is equal to \( \Sigma^* \). This is an interesting property especially relevant to the theory of codes [15]. The notion has been investigated as it pertains to independent sets, maximal independent sets, and disjunctive languages [13, 18]. Later, the notion was generalized from the set of infixes of a language being the universe, to arbitrary relations used in place of the infix relation [14]. For example, a language \( L \) is suffix-dense if the set of all suffixes of \( L \) is equal to the universe. Homomorphisms that preserve different types of density were investigated as well [16].

Recently, these generalized notions of density were studied as applied to types of pushdown automata and counter machines [4]. It was surprisingly found that it is...
decidable whether a language accepted by a reversal-bounded nondeterministic pushdown automaton (NPDA) is dense (a pushdown is \( l \) reversal-bounded if the number of changes in direction between pushing and popping is at most \( l \), and is reversal-bounded if it is \( l \) reversal-bounded for some \( l \)). This is true despite the fact that it is undecidable whether the language accepted by such a machine is equal to \( \Sigma^* \) (even if the pushdown is a 1-reversal-bounded counter). Therefore, deciding whether the set of infixes is the universe is easier than deciding whether the set itself is equal to the universe. However, density was found to be undecidable for deterministic pushdown automata (without a reversal-bound), and nondeterministic one counter automata (again without a reversal-bound). Decidability and undecidability results for other variants of density are also presented in [4].

A key property used to prove decidability of density for reversal-bounded NPDA

was that of the store language. The store language of a pushdown is the set of all state/store content pairs, \( qx \) where \( q \) is a state and \( x \) is a word over the pushdown alphabet, that can appear in any accepting computation. It is known that the store language of every NPDA is a regular language [1]. This inspired the authors to investigate the store language of other machine models, especially some types of Turing machines [12]. Of particular interest was the recent result that the store language of every nondeterministic Turing machine with a one-way read-only input tape, and a reversal-bounded worktape (the store; where there is a bound on the number of changes of direction of the read/write head) is a regular language.

In this paper, density of languages accepted by this same type of nondeterministic Turing machines is similarly shown to be decidable using the regularity of the store languages. A corollary to this is decidability of density for languages accepted by several types of one-way nondeterministic reversal-bounded machine models: stack automata [3,5] (machines that can enter the “contents” of the pushdown in read-only mode), queue automata [7], and \( k \)-flip pushdown automata [8] (machines that can flip their pushdown contents at most \( k \) times). However, undecidability is obtained for one-way deterministic Turing machines with a one-way read-only input and two 1-reversal-bounded pushdowns, or for nondeterministic Turing machines over a unary alphabet. Decidability of density for several types of two-way machine models is also investigated.

2. Preliminaries

In this paper, we assume knowledge of formal language and automata theory, referring to [9] for an introduction. This includes nondeterministic finite automata (NFA), nondeterministic pushdown automata (NPDA), nondeterministic Turing machines, and their deterministic variants (obtained by replacing \( N \) with \( D \)).

An alphabet \( \Sigma \) is a finite set of symbols, and \( \Sigma^* \) is the set of all words over \( \Sigma \). A language \( L \) is any subset of \( \Sigma^* \). Let \( w \in \Sigma^* \). Then \( |w| \) is the length of \( w \), and \( |w|_a \) is the number of \( a \)'s in \( w \), for \( a \in \Sigma \). The reverse of \( w \), \( w^R \) is the word obtained by reversing the order of the letters of \( w \). A word \( y \) is an infix of \( w \) if \( w = xyx \), for some \( x, z \in \Sigma^* \). Given \( L \subseteq \Sigma^* \), \( \text{inf}(L) = \{ y \mid y \text{ is an infix of } w \in L \} \). The left quotient of \( L \) by \( R \), \( R^{-1}L = \{ y \mid xy \in L, x \in R \} \).
Next, we will define nondeterministic Turing machines, which we will define to have a one-way read-only input, and a separate read/write bi-infinite worktape.

A nondeterministic Turing machine (NTM) is a tuple $M = (Q, \Sigma, \Gamma, \downarrow, \delta, q_0, F)$, where $Q$ is the finite set of states, $\Sigma$ is the input alphabet, $\Gamma$ is the worktape alphabet, $\downarrow \in \Gamma$ is the blank symbol, $\downarrow \in \Gamma$ is the worktape head, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, $\Gamma_0 = \Gamma - \{\downarrow\}$, and $\delta$ is a relation from $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma_0$ to $Q \times \Gamma_0 \times \{L, S, R\}$.

A configuration of $M$ is a tuple $(q, w, x)$, where $q \in Q$ is the current state, $w \in \Sigma^*$ is the remaining input, and $x \in \Gamma^*$ is the worktape contents with $|x|_\downarrow = 1$. Next, we will describe how configurations change. Below, $q, q' \in Q; a \in \Sigma \cup \{\lambda\}, x \in (\Gamma_0 - \Gamma_0^\uparrow), y \in (\Gamma_0 - \Gamma_0^\downarrow), c, d, c', d' \in \Gamma_0$.

- $(q, aw, x \downarrow cy) \vdash_M (q', w, x \downarrow dy)$, if $(q', d, S) \in \delta(q, a, c)$,
- $(q, aw, x \downarrow cy) \vdash_M (q', w, xd' \downarrow y')$, if $(q', d, R) \in \delta(q, a, c), (y = \lambda \implies y' = \downarrow, \text{otherwise } y' = y), (x = \lambda \land d = \downarrow \implies d' = \lambda, \text{otherwise } d' = d)$,
- $(q, aw, x \downarrow cy) \vdash_M (q', x, x' \downarrow c'd'y)$, if $(q', d, L) \in \delta(q, a, c), (x = \lambda \implies x' = \lambda, \text{otherwise } x = x'c'), (y = \lambda \land d = \downarrow \implies d' = \lambda, \text{otherwise } d' = d)$.

Let $\vdash^*_M$ be the reflexive and transitive closure of $\vdash_M$. Then, the language accepted by $M$, denoted by $L(M)$ is defined to be:

$$L(M) = \{w \mid (q_0, w, \downarrow) \vdash^*_M (q_f, \lambda, x), w \in \Sigma^*, q_f \in F\}.$$ 

Furthermore, the store language of $M$, $S(M)$, is defined to be:

$$S(M) = \{qx \mid (q_0, w, \downarrow) \vdash^*_M (q_f, \lambda', x), q_f \in F, w, w' \in \Sigma^*\}.$$ 

An NTM of this type is said to be reversal-bounded, if there is a $k$ such that $M$ makes at most $k$ changes between moving the worktape left and right on every accepting computation. Such a machine can be assumed to be deterministic in the usual fashion \cite{12}.

In \cite{12}, the following was shown:

**Proposition 1.** Given a nondeterministic Turing machine $M$ with a one-way read-only input tape, and a reversal-bounded worktape, then $S(M)$ is a regular language that can be effectively constructed from $M$.

It is clear that should the reversal-bounded condition be removed, then $S(M)$ could be an arbitrary recursively enumerable language after a left quotient by a state symbol, since, given an arbitrary Turing machine with a two-way read/write input tape $M$ and initial state $q_0$ (that without loss of generality is never re-entered), a TM with a one-way read-only input tape and a worktape $M'$ could be constructed that copies the input to the store, then simulates $M$ on the worktape. Then $(q_0 \downarrow)^{-1}S(M) = L(M)$.

3. Density of Turing Machines

A language $L \subseteq \Sigma^*$ is **dense** if $\inf(L) = \Sigma^*$. Even though it has long been known that it is undecidable for even a one-way nondeterministic one counter automaton that
makes only one reversal on the counter, whether the language it accepts is equal to $\Sigma^*$ \[2\], the infix operator perhaps counter-intuitively makes the problem easier.

Here, we will prove that it is decidable whether a language accepted by a one-way nondeterministic Turing machine with a reversal-bounded read/write worktape is dense. Then, it will follow that this is true for a number of different machine models. The main tool of the proof is that for these types of Turing machines, the store language is a regular language, by Proposition 1.

Proposition 2. It is decidable, given $L$ accepted by a NTM with a one-way read-only input and a $k$-reversal-bounded read/write worktape, whether $L$ is dense.

Proof. Let $M = (Q, \Sigma, \Gamma, \downarrow, \delta, q_0, F)$ be an NTM with a one-way input and a $k$-reversal-bounded worktape. Assume without loss of generality that all transitions that stay on the worktape do not change the worktape (any changes can be remembered in the state and applied when moving).

Next, assume without loss of generality that the states of $M$ are partitioned into subsets $Q_0, Q_1, \ldots, Q_k$, where $Q_i$ contains all states defined on or after the $i$th reversal, but before the $(i + 1)$st reversal. Let $T$ be a set of labels in bijective correspondence with the transitions of $M$ (each transition $(p', d, x) \in \delta(p, a, c)$ has an associated label in $T$).

Consider $S(M)$, which is a regular language by Proposition 1. For each $q \in Q$, consider $R^q = qS(M)$, also regular.

Consider the language

$L^q = \{ w \mid (q_0, uwv, \downarrow) \vdash^* M (q, u, v, \alpha) \vdash^* M (q, v, \beta) \vdash^* M (q_f, \lambda, \gamma), u, v, w \in \Sigma^*, q_f \in F \}.$

It will be proven that $L^q$ is regular for each $q \in Q$.

A sequence of transition labels $y = t_1 \cdots t_m, t_i \in T, 1 \leq i \leq m$, is valid for state $q$ if $t_1$ starts in state $q$, and $t_m$ switches to $q$, and for each $i, 1 \leq i < m$, the outgoing state of $t_i$ is the same as the incoming state of $t_{i+1}$, and $t_i$ being a transition that stays on letter $d \in \Gamma$ of the store implies $t_{i+1}$ also reads $d$. There is no restriction on the input word read while transitioning via $y$. Given a valid $y$, let $\tilde{y} \in \Gamma^*$ be the word with the first letter of it being the store letter $t_1$ is defined on, and subsequent letters are obtained from $y$ from left-to-right by concatenating the letters read from the store after a transition that moves either left or right but not stay on the store. Also, given a valid $y$, let $\hat{y}$ be obtained from $y$ from left-to-right by concatenating all letters written on the store during a transition that moves either left or right (but not stay) on the store. Notice that since $\tilde{y}$ and $\hat{y}$ are only defined on valid $y$, since $y$ transitions from state $q$ to itself, and since the states are partitioned by reversal-bounds, then $\tilde{y}$ and $\hat{y}$ are only defined on $y$ such that all transitions in $y$ only move right or stay, or move left or stay. Lastly, let $\check{y}$ be $\lambda$ if $y$ ends with a transition that moves right or left on the store, and the last store letter read by the last transition of $y$ otherwise (if a stay transition).

Let $h_\Sigma$ be a homomorphism from $(T \cup \Gamma)^*$ to $\Sigma^*$ that erases all letters of $\Gamma$ and maps each $t \in T$ to the input in $\Sigma \cup \{\lambda\}$ read by $t$. 

Density of Turing Machines

For \( q \in Q_i \), with \( i \) even, then there are no transitions moving left on the store between states \( q \) and \( q \). Then, an NFA \( M' \) is created accepting an intermediate language \( L_i^q \subseteq \Gamma^*\Gamma^* \), where \( M' \) does the following in parallel:

- verifies that the input is of the form \( \mu \nu \), where \( \mu, \nu \in \Gamma_0^* \); \( y = t_1 \cdots t_m, t_i \in T, 1 \leq i \leq m \), and that \( y \) is valid for \( q \);
- verifies that \( \mu \downarrow \overline{y} \nu \in R^q \);
- verifies that \( \mu \downarrow \overline{y} \nu \in R^q \).

Intuitively, \( \mu \downarrow \overline{y} \nu \in R^q \) enforces that this word is in the store language, which implies that there is some input word which can reach this configuration in an accepting computation. Then, \( M' \) enforces that \( y \) encodes a valid change of configurations between the two configurations involving \( q \), which is possible since the worktape head only moves to the right. Since \( y \) is valid, the sequence of transitions switches \( \overline{y} \) to \( \overline{y} \) in \( M \). So \( M \) can reach \( \mu \overline{y} \downarrow \mu \nu \), which is then verified to be in \( R^q \), enforcing that this is in the store language, so an input word can lead it to acceptance.

**Claim 1.** \( h_\Sigma(L_i^q) = L_i^q \).

**Proof.** “\( \subseteq \)” Let \( s \in h_\Sigma(L_i^q) \). Thus, there exists \( r \in \Gamma^*\Gamma^* \) such that \( h_\Sigma(r) = s \) and \( r \in L_i^q \). Then \( r = \mu \nu, \mu, \nu \in \Gamma^* \), \( y \in T^* \), where \( M' \) verifies \( \mu \downarrow \overline{y} \nu \in R^q \). Then \( (q_0, u, \downarrow \omega) \vdash_M (q, \lambda, \mu \downarrow \overline{y} \nu) \) for some \( \nu \in \Sigma^* \). Then, \( (q_0, u, s, \downarrow \omega) \vdash_M (q, s, \mu \downarrow \overline{y} \nu) \). Then, on each letter of \( y \), since \( y \) is valid, \( M' \) simulates \( M \) on the last letter of reading each letter from \( \Gamma \) reading each letter from \( s \in \Sigma^* \) and replacing each letter of \( \overline{y} \) with the next one from \( \overline{y} \), while starting and finishing in state \( q \). Assume first that the last letter of \( y \) is a transition that moves right on the store. Thus,

\[
(q, s, \mu \downarrow \overline{y} \nu) \vdash_M (q, \lambda, \mu \overline{y} \downarrow = \mu \overline{y} \downarrow \nu),
\]

since \( \nu = \lambda \). Similarly if the last letter of \( y \) is a transition that stays on the store, then \( (q, s, \mu \downarrow \overline{y} \nu) \vdash_M (q, \lambda, \nu \overline{y} \downarrow \nu) \), where \( \nu \in \Gamma \). In either case, then since \( M' \) verified \( \mu \overline{y} \downarrow \nu \in R^q \), and this implies \( (q, v, \mu \overline{y} \downarrow \nu) \vdash_M (q, \lambda, \gamma), qf \in F \), for some \( v \in \Sigma^* \). Hence \( h_\Sigma(y) = s \in L_i^q \).

“\( \supseteq \)” Let \( s \in L_i^q \). Thus,

\[
(q_0, u, s, \downarrow \omega) \vdash_M (q, \nu, \alpha_1 \downarrow \alpha_2 \alpha_4) \vdash_M (q, v, \alpha_1 \alpha_3 \downarrow \alpha_4) \vdash_M (q, f, \lambda, \gamma),
\]

(1)

for some \( \alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \Gamma_0^* \). Let the derivation above between states \( q \) and \( q \) be via transitions \( t_1, \ldots, t_n \) respectively. Let \( y = t_1 \cdots t_n \). First, \( y \) must be valid for \( q \). Assume first that this sequence ends with a transition that moves right on the store. Then \( \alpha_2 \) is \( \overline{y} \) and \( \alpha_3 = \overline{y} \) and \( \nu = \lambda \). Indeed, \( \alpha_1 \downarrow \alpha_2 \alpha_4 \in R^q \) by Equation (1), and \( \alpha_1 \downarrow \alpha_2 \alpha_4 = \alpha_1 \downarrow \overline{y} \alpha_4 \in R^q \). Further, \( \alpha_1 \alpha_3 \downarrow \alpha_4 \in R^q \) by (1), and \( \alpha_1 \alpha_3 \downarrow \alpha = \alpha_1 \overline{y} \downarrow \overline{y} \alpha_4 \in R^q \). Thus, \( s \in h_\Sigma(L_i^q) \). Similarly if the sequence ends with a stay transition.

For \( q \in Q_i \), with \( i \) odd, then another language \( L_i^q \) is created. Hence, that there are no right transitions on the store defined between state \( q \) and itself. This case is similar in principal, however, it is slightly trickier since the transitions work in a right-to-left
fashion, and because the read/write head is placed to the left of the scanned symbol, thereby causing a mild complication at the beginning and end of the word $y$.

Create an intermediate 2-way NFA $M'$ accepting a language $L'_2 \subseteq \Gamma^* T^* \Gamma^*$, where $M'$ does all of the following:

- verifies that the input is of the form $\mu \nu \mu$, where $\mu, \nu \in \Gamma_0^*$, $z = y^R = t_1 \cdots t_m, t_i \in T, 1 \leq i \leq m$, and that $z$ is valid for $q$.
- let $z'$ be obtained from $(\overrightarrow{z})^R$ by inserting $\downarrow$ in the second last position; it verifies that $\mu z' \nu \in R^3$ (by using the reverse of $R^3$) as it reads $y$ (from right-to-left),
- if $y$ starts with a symbol that stays on the store, then $M'$ verifies that $\mu \downarrow (\overrightarrow{z})^R \nu \in R^3$; otherwise, $M'$ verifies that $\mu_1 \downarrow b(\overrightarrow{z})^R \nu \in R^3, b \in \Gamma, \mu = \mu_1 b$ if $\mu \neq \lambda$, and $\mu_1 = \lambda, b = \uparrow$ if $\mu = \lambda$.

**Claim 2.** $h_\Sigma(L'_2) = L_q$.

**Proof.** "$\subseteq$" Let $s \in h_\Sigma(L'_2)$. Thus, there exists $r \in \Gamma^* T^* \Gamma^*$ such that $h_\Sigma(r) = s$ and $r \in L'_2$. Then $r = \mu \nu \mu, \nu \in \Gamma^*, y \in T^*, z = y^R$, where $M'$ verifies that $\mu_1 z' \nu \in R^3$ with $z'$ obtained from $(\overrightarrow{z})^R$ by inserting $\downarrow$ in the second last position. Then $(q_0, u, \downarrow, \uparrow) \vdash_M^+ (q, \lambda, \mu z' \nu)$ for some $u \in \Sigma^*$. Then, $(q_0, u s, \downarrow, \uparrow) \vdash_M^+ (q, \lambda, v \mu z' \nu)$. Then, on each letter of $y$ from right-to-left, since $z$ is valid, $M'$ simulates $M$ on the store letter read from $\Gamma$, reading each letter from $s$ and replacing each letter of $\overrightarrow{z}$ with the next one from $\overrightarrow{y}$, while starting and finishing in state $q$. If the first letter of $y$ is a left transition, then

$$(q, s, \mu z' \nu) \vdash_M^+ (q, \lambda, \mu_1 \downarrow b(\overrightarrow{z})^R \nu),$$

where $\mu = \mu_1 b$ if $\mu \neq \lambda$, and $\mu_1 = \lambda, b = \uparrow$ if $\mu = \lambda$. Then since $M'$ verified $\mu_1 \downarrow b(\overrightarrow{z})^R \nu \in R^3$, this implies $(q, v, \mu_1 \downarrow b(\overrightarrow{z})^R \nu) \vdash_M^+ (q, \lambda, \gamma), q_f \in F$, for some $v \in \Sigma^*$. Hence $h_\Sigma(y) = s \in L_q$. Similarly if the first letter of $y$ is a stay transition.

"$\supseteq$" Let $s \in L_q$. Thus,

$$(q_0, u s v, \downarrow, \uparrow) \vdash_M^+ (q, v, \alpha_1 \alpha_2 \downarrow a \alpha_3) \vdash_M^+ (q, v, \alpha_1 \downarrow \alpha_4 \alpha_3) \vdash_M^+ (q, \lambda, \gamma),$$

$q_f \in F$, for some $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \Gamma_0^*, a \in \Gamma_0$. Let the derivation above between states $q$ and $q$ be via transitions $t_1, \ldots, t_n$ respectively. Let $y = t_n \cdots t_1$, and $z = y^R$. First, $z$ must be valid for $q$. Assume first that this ends with a left transition.

Then, in Equation (2), let $\alpha_2$ be such that $\alpha_2 = b a \alpha_3, b \in \Gamma_0$ if $\alpha_2 \neq \lambda$, and $\alpha_2 = \lambda, b = \uparrow$ otherwise, and let $\alpha_4$ be such that $\alpha_4 = b a \alpha_3$ (it must start with $b$ since $z$ ends with a left transition). Then $(\overrightarrow{z})^R = \alpha_2 a$ and $(\overrightarrow{z})^R = \alpha_4$. Indeed, $\alpha_1 b a \alpha_2 \downarrow a \alpha_3 \in R^3$ by Equation (2), and $\alpha_1 b a \alpha_4 \downarrow a \alpha_3 = \alpha_1 b z \alpha_3$, where $z'$ is obtained from $(\overrightarrow{z})^R$ by inserting $\downarrow$ in the second last position. Further, $\alpha_1 b a \alpha_3 \in R^3$ by Equation (2), and $\alpha_1 \downarrow b a \alpha_3 = \alpha_1 \downarrow (\overrightarrow{z})^R \alpha_3 \in R^3$. Thus, $s \in h_\Sigma(L'_2)$. Similarly if the sequence ends with a stay transition.

Hence, $L_q$ is regular since regular languages are closed under homomorphism. Thus, $L' = \bigcup_{q \in Q} L_q$ is also regular.

Lastly, it is shown that $\inf(L) = \Sigma^*$ if and only if $\inf(L') = \Sigma^*$. Indeed, it is immediate that if $\inf(L') = \Sigma^*$, then $\inf(L) = \Sigma^*$, since $\inf(L') \subseteq \inf(L)$. For the
Density of Turing Machines

opposite direction, assume \( \inf(L) = \Sigma^* \). Let \( w \in \Sigma^* \). Then \( w \in \inf(L) \). Considering the word \( w' = w^{|Q|+1} \), then \( w' \in \inf(L) \) as well by the assumption. By the pigeonhole principal, an entire copy of \( w \) has to be read between some state \( q \) and itself. Hence, \( w \in \inf(L') \), and \( \inf(L') = \Sigma^* \).

The next result involves the definitions of pushdown automata \([9]\), queue automata \([7]\), stack automata \([5, 6]\), and flip pushdown automata \([8]\), which will not be defined formally here. A pushdown automaton, queue automaton, or flip pushdown automaton are reversal-bounded if there is a bound on the number of switches between increasing and decreasing the size of the store. For stack automata, it is defined to be reversal-bounded if there is a bound on the number of changes of direction of the read/write head (which is influenced both by changing and reading the stack). From the definitions, the following is straightforward.

**Proposition 3.** Given \( M \), a reversal-bounded pushdown automaton, then \( L(M) \) can be accepted by a nondeterministic Turing machine with a one-way read-only input, and a reversal-bounded read/write worktape. This result also holds for \( M \), a reversal-bounded queue machine, a reversal-bounded stack machine, and a reversal-bounded \( k \)-flip pushdown automaton.

Then, from Proposition \([8]\) the following are true.

**Corollary 4.** It is decidable, given \( L \) accepted by a reversal-bounded queue automaton, whether \( L \) is dense.

**Corollary 5.** It is decidable, given \( L \) accepted by a reversal-bounded stack automaton, whether \( L \) is dense.

**Corollary 6.** It is decidable, given \( L \) accepted by a reversal-bounded \( k \)-flip pushdown automaton, whether \( L \) is dense.

Next, we will address undecidability. The first result applies to most of the deterministic and nondeterministic families that have an undecidable emptiness problem. The right marked concatenation of \( L \) with a language \( R \) is \( L\$R \) where \( \$ \) is not a letter of \( L \) (but it can be a letter of \( R \)).

**Proposition 7.** Let \( \mathcal{L} \) be a family of languages with an undecidable emptiness problem that is closed under right marked concatenation with regular languages. Then it is undecidable whether \( L \in \mathcal{L} \) is dense.

**Proof.** Let \( L \in \mathcal{L} \) over \( \Sigma \). Let \( \$ \) be a new symbol, and let \( \Sigma' = \Sigma \cup \{\$\} \). Let \( L' = L\$\Sigma'^* \in \mathcal{L} \).

If \( L \) is empty, then \( L' \) is empty too and so \( L' \) is not dense.

If \( L \) is not empty then say \( w \in L \). Thus \( w\$\Sigma'^* \) is a subset of \( L' \) and so every word of \( \Sigma'^* \) is an infix of \( L' \). Hence, \( L' \) is dense. □

Next, we describe a construction which was essentially presented in \([2]\).
Construction 8. Let $Z$ be a DTM operating on an initially blank tape. Assume that if $Z$ halts, it makes $2k$ moves for some $k \geq 2$.

Let $w = ID_1#ID_3#ID_5\ldots#ID_{2k-1}$§ID_{2k}^R#\ldots#ID_6^R#ID_4^R#ID_2^R$, where the $ID_i$’s are configurations of $Z$.

Let $\Sigma$ be the alphabet over which $w$ is defined.

We construct a 1-reversal 2-stack real-time (i.e. no $\lambda$-transitions) DPDA $M_1$ as follows, given input string $x$:

(i) $M_1$ enters state $q_r$ if $x$ is not in the above format (the finite control can detect this).

(ii) If $x$ is in the above format, then $M_1$ enters state $q_r$ if one of the following is not true:

   (A) $ID_1$ is not an initial configuration of $Z$ on blank tape,

   (B) $ID_{2k}$ is not a halting configuration of $Z$,

   (C) $ID_{i+1}$ is not a valid successor of $ID_i$ for some $i$.

Otherwise, $M_1$ enters state $q_h$.

Thus, on any input $x$, $M_1$ only enters $q_h$ if $x$ is a halting computation of $Z$; otherwise $M$ enters $q_r$.

As the halting problem is undecidable, it follows that the emptiness problem is undecidable for one-way 1-reversal-bounded 2-stack real-time DPDA.

From this, and Proposition 7, the following is true.

Corollary 9. It is undecidable, given a language $L$ accepted by a one-way deterministic 2-stack real-time DPDA where both pushdowns are 1-reversal-bounded, whether $L$ is dense.

Undecidability is therefore immediate for deterministic Turing machines with two 1-reversal-bounded worktapes, as each such worktape can simulate a pushdown.

Corollary 10. It is undecidable, given $L$ accepted by a deterministic real-time Turing machine with a one-way read-only input and two 1-reversal-bounded worktapes, whether $L$ is dense.

This shows that decidability changes between one and two reversal-bounded worktapes for these kinds of Turing machines.

It is not clear whether (or not) the result above can be extended to unary languages accepted by one-way deterministic two pushdown machines. In fact, we conjecture that density is decidable for this model. However, the next result shows that it is undecidable for unary languages when nondeterminism is used.

Proposition 11. It is undecidable, given unary $L$ accepted by a one-way nondeterministic real-time two pushdown machine, where both pushdowns are 1-reversal-bounded, whether $L$ is dense.

Proof. Starting with Construction 8 from $M_1$, now construct a 1-reversal 2-stack NPDA $M$ as follows. $M$ simulates $M_1$ described in the construction above by reading
symbol $a$ at each move of $M_1$ while nondeterministically guessing the input to $M_1$. If $M_1$ lands in $q_h$ after reading $a^k$ for some $k$, $M$ accepts any string $a^{k+i}$ for $i \geq 0$. Clearly, $\text{inf}(L(M)) = a^*$ if the DTM $Z$ halts on a blank tape. It $Z$ does not halt on a blank tape, $L(M)$ is empty and, hence, $\text{inf}(L(M))$ is also empty. It follows that $L(M)$ is dense if and only if $Z$ halts on blank tape. Moreover, $M$ is real-time.

There are some other interesting models with undecidable emptiness problems for which density will also be undecidable. It is undecidable, given a 2DCM(2) (a two-way DFA with two reversal-bounded counters) $M$ over a letter-bounded language, whether $L(M)$ is empty [10]. This model is also closed under right marked concatenation.

Corollary 12. It is undecidable, given a 2DCM(2) $M$, whether $L(M)$ is dense.

For 2NCM, i.e. two-way NFAs with $k$ reversal-bounded counters for some $k$, it is known that all such machines accepting unary languages can be effectively converted to NFAs [11]. Therefore:

**Proposition 13.** Given a unary 2NCM $M$, it is decidable whether $L(M)$ is dense.

In contrast:

**Proposition 14.** It is undecidable, given a 2DFA $M$ with one unrestricted counter over a unary language, whether $L(M)$ is dense.

**Proof.** It is well known that it is undecidable, given a machine $Z$ with no input but with two (unrestricted) counters $C_1$ and $C_2$ that are initially set to zero, whether $Z$ will halt [17]. In fact, we may assume that if $Z$ does not halt, it goes into an infinite loop with increasing counter values.

Given such a machine $Z$, we construct a 2DFA $M$ which, when given an input $w = a^k$ (with left and right end markers) simulates $Z$. $M$ has one counter $C_1$ to simulate $C_1$ of $Z$, and uses its unary two-way input tape to simulate counter $C_2$ of $Z$.

If, during the simulation, $M$ attempts to go past the right end-marker, $M$ accepts $w$. If, during the simulation, $Z$ halts, $M$ rejects. Clearly, if $Z$ does not halt, $L(M) = a^*$. If $Z$ halts, $L(M)$ is finite. It follows that $L(M)$ is dense if and only if $Z$ does not halt, which is undecidable.

**Corollary 15.** It is undecidable, given a 2DFA $M$ with one unrestricted counter over a unary language, whether $L(M) = a^*$ (respectively, whether $L(M)$ is finite).

It is open whether or not a language accepted by a 2NCM with one reversal-bounded counter is empty, and density for this model is open as well

4. Conclusions and Future Directions

Here, we show that determining whether a language is dense is decidable for nondeterministic Turing machines with a one-way read-only input and one reversal-bounded worktape, and therefore this is also decidable for one-way nondeterministic machines
with several different reversal-bounded data structures, such as pushdowns, stacks, $k$-flip pushdowns, and queues. However, if the number of reversal-bounded Turing tapes is increased to two, then it is undecidable, even if the machine is deterministic, real-time, and $1$-reversal-bounded.

There are still several open problems related to determining density. It is still unknown whether the density of $L$ is decidable, if $L$ is accepted by a one-way nondeterministic (or deterministic) machine with multiple reversal-bounded counters. This is decidable when there is only one counter. The problem is also open for $L$ accepted by one-way deterministic machines with one counter (no reversal-bound), and for deterministic Turing machines with multiple reversal-bounded tapes accepting unary languages.

References

[1] J. Autebert, J. Berstel, L. Boasson, Handbook of Formal Languages, 1, chapter Context-Free Languages and Pushdown Automata, Springer-Verlag, Berlin, 1997.
[2] B. Baker, R. Book, Reversal-bounded multipushdown machines. Journal of Computer and System Sciences 8 (1974) 3, 315–332.
[3] J. Berstel, D. Perrin, Theory of Codes. Academic Press, Orlando, 1985.
[4] J. Eremondi, O. Ibarra, I. McQuillan, On the Density of Context-Free and Counter Languages. International Journal of Foundations of Computer Science (2017). Accepted.
[5] S. Ginsburg, S. Greibach, M. Harrison, One-way Stack Automata. J. ACM 14 (1967) 2, 389–418.
[6] S. Ginsburg, S. Greibach, M. Harrison, Stack Automata and Compiling. J. ACM 14 (1967) 1, 172–201.
[7] T. Harju, O. Ibarra, J. Karhumäki, A. Salomaa, Some Decision Problems Concerning Semilinearity and Commutation. Journal of Computer and System Sciences 65 (2002) 2, 278–294.
[8] M. Holzer, M. Kutrib, Flip-Pushdown Automata: Nondeterminism is Better than Determinism. In: Z. Ésik, Z. Fülöp (eds.), Developments in Language Theory. Lecture Notes in Computer Science 2710, 2003, 361–372.
[9] J. E. Hopcroft, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, MA, 1979.
[10] O. Ibarra, Reversal-Bounded Multicounter Machines and Their Decision Problems. Journal of the ACM 25 (1978) 1, 116–133.
[11] O. Ibarra, T. Jiang, N. Tran, H. Wang, New Decidability Results Concerning Two-Way Counter Machines. SIAM J. Comput. 23 (1995) 1, 123–137.
[12] O. Ibarra, I. McQuillan, On Store Languages of Language Acceptors. Technical Report arXiv:1702.07388, arXiv.org, 2017. https://arxiv.org/abs/1702.07388
[13] M. Ito, Dense and Disjunctive Properties of Languages. In: Z. Ésik (ed.), Fundamentals of Computation Theory 1993. Lecture Notes in Computer Science 710, Springer Berlin Heidelberg, 1993, 31–49.
[14] H. Jürgensen, L. Kari, G. Thierrin, Morphisms Preserving Densities. *International Journal of Computer Mathematics* **78** (2001), 165–189.

[15] H. Jürgensen, S. Konstantinidis, *Handbook of Formal Languages*, 1, chapter Codes, Springer-Verlag, Berlin, 1997.

[16] H. Jürgensen, I. McQuillan, Homomorphisms Preserving Types of Density. *Acta Cybernetica* **19** (2009) 2, 499–516.

[17] M. L. Minsky, Recursive Unsolvability of Post’s Problem of “Tag” and other Topics in Theory of Turing Machines. *Annals of Mathematics* **74** (1961) 3, pp. 437–455.

[18] H. J. Shiyr, *Free Monoids and Languages*. third edition, Hon Min Book Company, Taichung, 2001.