Dense Completeness

Andreas Krebs and Klaus-Jörn Lange

University of Tübingen, Germany
{krebs,lange}@informatik.uni-tuebingen.de

Abstract. We introduce *dense completeness*, which gives tighter connection between formal language classes and complexity classes than the usual notion of completeness. A family of formal languages $\mathcal{F}$ is *densely complete* in a complexity class $\mathcal{C}$ iff $\mathcal{F} \subseteq \mathcal{C}$ and for each $C \in \mathcal{C}$ there is an $F \in \mathcal{F}$ such that F is many-one equivalent to C.
For AC^0-reductions we show the following results: the family CFL of context-free languages is densely complete in the complexity class SAC^1. Moreover, we show that the indexed languages are densely complete in NP and the nondeterministic one-counter languages are densely complete in NL. On the other hand, we prove that the regular languages are not densely complete in NC^1.

1 Introduction

In this work we observe differences and similarities between *complexity classes* (like NP or NL) and *classes of formal languages* (like the regular or the context-free languages).

Despite the difference between the two areas there are surprisingly close connections between them. A common behavior is that for a family of formal languages $\mathcal{F}$ and a complexity class $\mathcal{C}$ we have on the one hand $\mathcal{F} \subset \mathcal{C}$ and on the other hand that for every $C \in \mathcal{C}$ there is an $F \in \mathcal{F}$ such that $C \leq_m F$.

For the following complexity classes we have multiple complete formal language classes([Lan93]:

complexity class	formal language classes
NC^1	REG, visibly push-down languages
NL	linear context-free languages, one-counter languages
SAC^1	context-free languages, IO
NP	indexed languages = OI

In this article we will show that some of these relations are in fact much closer than thought before. To this end we introduce the notion of *dense completeness* which requires that for every $C \in \mathcal{C}$ there is an $F \in \mathcal{F}$ such that not only $L \leq_m F$ but $F \leq_m L$ holds as well.

In this paper when we talk about many-one reductions we think about DLOGTIME-uniform AC^0 reductions. This type of reducibility is finer than the more usual logspace or polynomial time reducibilities. It adequately demonstrates the closeness of the relationship between complexity classes and families

of formal languages exhibited in this work. We write $L_1 \leq_m^{\mathrm{AC}^0} L_2$ if there is a many-one DLOGTIME-uniform AC^0 reduction from L_1 to L_2, and $L_1 \approx_m^{\mathrm{AC}^0} L_2$ if $L_1 \leq_m^{\mathrm{AC}^0} L_2$ and $L_2 \leq_m^{\mathrm{AC}^0} L_1$.

Definition 1 (Dense Completeness). *Let $\mathcal{C}$, $\mathcal{D}$ be classes of languages. We say $\mathcal{C}$ is densely complete in $\mathcal{D}$ if*

- $\mathcal{C} \subseteq \mathcal{D}$ *and*
- $\forall D \in \mathcal{D}\ \exists C \in \mathcal{C}\ :\ D \approx_m^{\mathrm{AC}^0} C$.

We observe that this notion is transitive in the sense that if both $\mathcal{C}$ is densely complete for $\mathcal{D}$ and $\mathcal{D}$ is densely complete for $\mathcal{E}$ then $\mathcal{C}$ is densely complete for $\mathcal{E}$.

The main difference between the two areas lies in the combinatorial intractability of complexity classes, in contrast to the setting of formal languages, which are more amenable to combinatorial analysis via pumping lemmata, and where properties such as emptiness and finiteness are decidable. This difference is expressed by the abundance of notoriously open questions in complexity theory while in the area of formal languages many of the basic questions like the relation between determinism and nondeterminism are settled.

We observe some typical differences between these two areas:

typical properties	formal language class	complexity class
closure under reducibilities	No	Yes
closure under intersection	No	Yes
closure under morphism	Yes	No
emptiness decidable	Yes	No

In this paper we understand a class of formal languages as the following:

Definition 2. *A family F of formal languages is a class of languages finitely presented by some grammar or automata model such that the emptiness problem is decidable and which is closed under morphism, inverse morphisms, and intersection with regular sets, i.e. F is a full trio.*

Well established members of this kind are the regular languages, the linear context-free languages, the one-counter languages, the context-free languages, the indexed languages and the macro language families IO and OI ([Fis68,Aho68,Aho69]).

First, it might seem impossible to find a formal language class that is densely complete in a complexity class, since by by Ladner ([Lad75]) we know that in between any pair of languages L_1 and L_2 such that $L_1 \leq L_2$ but not $L_1 \geq L_2$ there is an L_3 such that $L_1 \leq L_3 \leq L_2$ and neither $L_1 \geq L_3$ nor $L_3 \geq L_2$. The proof is done by merging the languages L_1 and L_2 according to the length of words. In case L_1 is the empty set this means "punching large holes" into the language L_2. This idea seems to contradict the concept of any pumping lemma. Thus a language like L_3 appears to lack any relation to a typical formal language with a semilinear Parikh-image, since languages with a semilinear Parikh image can only have linear sized holes in their set of lengths of words.

Despite this fact, we will show that in some cases there are in fact densely complete subfamilies of formal languages. In particular we show that the complexity classes NL, SAC^1, and NP indeed possess dense subfamilies of formal languages.

The paper is structured as follows: After listing the notions used in this paper we treat the nondeterministic classes SAC^1, NL, and NP and show that they indeed contain dense subfamilies of formal languages. We then show that the regular languages, which contain NC^1-complete problems, are not densely complete in NC^1. In a closing discussion we look at perspectives to generalize our approach.

2 Preliminaries

We assume the reader to be acquainted with some basic families of formal languages like the regular languages or the context-free languages and their subfamilies, the linear context-free languages and the one-counter languages.

In this paper we use the following notations for one-way and two-way k-head nondeterministic finite automata with and without a stack. We denote by 1-NFA(k) (resp. 2-NFA(k)) the one-way (resp. two-way) finite automata. By 1-NPDA(k) (resp. 2-NPDA(k)) we denote the one-way (resp. two way) nondeterministic pushdown automaton, where the 2-way pushdown automata are restricted to work in polynomial time. Finally by 1-NOCA we denote the one-way one nondeterministic counter automata.

Aho introduced the family INDEX of indexed languages and characterized them by nondeterministic nested stack automata ([Aho68,Aho69]). The common idea in both characterizations is to equip the nonterminals of a context-free grammar or the stack symbols of a pushdown automaton by a stack of *index flags* which are manipulated appropriately. This results in working with a stack of stacks as memory structure.

We will refer to a variety of standard complexity classes such as NP, P, SAC^1, NC^1, ACC^0, ACC^0_q, AC^0, CC^0 (see e.g. [Joh90,Str94]). For a variety of different versions of auxiliary push-down-automata and for the class $\mathbf{LOG(}$CFL$\mathbf{)}$ we refer to [Coo71,Sud78]. This class coincides with SAC^1 [Ven91].

Throughout the paper we will use AC^0-many-one-reducibilities, i.e. mappings between free monoids computed by polynomial sized DLOGTIME-uniform circuits of constant depth using both and- and or-gates (see e.g. [BIS90]).

We will use the notion of a syntactic monoid as it is dealt with for the finite case by Eilenberg ([Eil76]) or Pin ([Pin86]).

3 The Context-free Languages and the class SAC^1

Sudborough showed that the context-free languages are complete for SAC^1 ([Sud78]). For this he used the equivalent characterization of SAC^1 by polynomial time bounded pushdown automata with a two-way input tape and an

auxiliary logspace tape. We will extend this result and show that the context-free languages are densely complete in SAC^1.

Theorem 1. *The context-free languages are densely complete in* SAC^1.

Before we start with the proof we will review the proof in [Sud78]. He applied the well known technique of replacing a logspace tape by a multitude of two-way input heads. He then reduced the number of input heads down to one by reductions which intensively repeated the input word. It is quite easy to see that these steps preserve denseness since we reduce the word problem of an automaton A working on input w to the word problem of an automaton B on a transformed input word $w' = T(w)$ and B with its two-way input head together with its stack can check w' to be of the right format first before simulating A and reject otherwise.

In a next essential step Sudborough reduced the word problem of a one-head two-way pushdown automaton A on an input w working in polynomial time $p(|w|)$ to the word problem of a usual one-way pushdown automaton B working on the input $T(w) := (w\$\overleftarrow{w}\$)^{p(|w|)}$. When A reverses the direction of its input head at the i-th position of a subword w of $T(w)$, B reads the remaining $|w| - i$ symbols of w against the first $|w| - i$ symbols of the following subword $\overleftarrow{w}$.

This seems to need an additional counter which while in use freezes the stack (and does not use it). But this *freezing counter* can be simulated within the stack of the PDA by simply pushing an intermediate bottom of stack symbol separating the frozen stack from the active counter. After successfully counting the remaining $|w|-i$ symbols of w against the first $|w|-i$ symbols of the following subword $\overleftarrow{w}$ the intermediate bottom of stack symbol is popped and the unfrozen stack below continues its work - now on the reversed word.

It is now tempting to speculate that this reduction preserves denseness. But this should not be the case: consider the case of undirected graph reachability. By Reingold's ([Rei08]) celebrated result this problem is in deterministic logspace and thus in SAC^1. It is possible to code it as a SAC^1-language in a way that after the last step of Sudborough's proof going from a two-way automaton A to a one-way automaton B we can cheat by giving B inputs which are only nearly of the form $T(w)$ but are too short and thus with the help of B can solve the shortest path problem for undirected graphs which is NL-complete ([Tan07]).

This problem is caused by the inability of one-way automaton B to check its input being of the correct form, i.e. being a member of $T(\Sigma^*)$. To avoid this problem we make use of the simple observation that the complement L_{BAD} of $T(\Sigma^*)$ is both a context-free language (in fact it is even a linear context-free language) and is in AC^0. We now reduce $L(A)$ via $T(\cdot)$ to the context-free language $L' := L(B) \cup L_{\mathrm{BAD}}$. But since $L' = T(L) \cup L_{\mathrm{BAD}}$ (observe that $T(L) = L(B) \cap T(\Sigma^*)$) we can reduce $L(A)$ via a AC^0-many-one reduction to L'.

With this trick in mind we change the route from 2-NPDA(k) to 1-NPDA(1) by not going via 2-NPDA(1) but instead via 1-NPDA(k) which seems to give easier proofs and thus these constructions can be easier transferred to the classes NP and NL.

But first we need some additional definitions. Given a word $w \in \Sigma^*$ of length n, and an integer $0 \leq k < n^c$, for some natural number c, define a mapping γ_c that maps w, k injectively to a word of length n over a new alphabet. We extend the alphabet by c markers, i.e. $\Sigma' = \Sigma \times 2^{1,\dots,c}$. The position of the markers will allow us to encode the number k. In order to represent the number k by c numbers in the range $1, \dots, n$ we pick $1 \leq k_1, \dots, k_c \leq n$ such that $k = \sum_{i=0}^{c-1} (k_{i+1} - 1) \cdot n^i$. Finally, we define $\gamma_c : \Sigma^* \times \mathbb{N} \to (\Sigma \times 2^{\{1,\dots,c\}})^*$ by

$$\gamma_c(w, k) = (w_1, X_1) \dots (w_n, X_n),$$

where $m_i \in X_l$ iff $k_i = l$.

Given a word $\triangleright w \triangleleft$ with start and end markers we can repeat the word a polynomial number of times and add increasing numbers to the repeated words. In this way we can later check that the repetition is bounded by a polynomial.

Definition 3. *Define* $\Gamma_c : \Sigma^* \to (\Sigma \times 2^{\{1,\dots,c\}} \cup \{\triangleright, \triangleleft\})^*$ *as*

$$w \mapsto \triangleright \gamma_c(w, 0) \triangleleft \triangleright \gamma_c(w, 1) \triangleleft \dots \triangleright \gamma_c(w, |w|^c - 1) \triangleleft .$$

Lemma 1. *The languages accepted by* $\bigcup_k$ 1-NPDA(k) *are densely complete for the languages accepted by* $\bigcup_k$ 2-NPDA(k).

Proof. Pick an automaton A from 2-NPDA(k) that recognizes a language $L \subseteq (\Sigma^*)$. Choose c such that the runtime of A is bounded by n^c.

We will show that there is a 1-NPDA$(k+1)$ automaton B that recognizes $L' = \Gamma_c(L) \cup L_{\text{BAD}}$, where $L_{\text{BAD}} = (\Sigma \times 2^{\{1,\dots,c\}} \cup \{\triangleright, \triangleleft\})^* \setminus \Gamma_c(\Sigma^*)$.

This will suffice for the proof since the mapping Γ_c can be computed by a DLOGTIME-uniform AC^0 reduction and since $L' \leq_m^{\text{AC}^0} L$ (see discussion at the beginning of this section) we get $L \approx_m^{\text{AC}^0} L'$.

First we show that we can find a 1-NPDA(2), that accepts all words in L_{BAD}. This automaton will not even use the stack. For this we choose two heads and place them randomly at two neighboring symbols $\triangleright$. Then we move them simultaneously right till they reach $\triangleleft$. If they do not reach $\triangleleft$ in the same step, we accept, since in the image of Γ_c the distance between $\triangleright$ and $\triangleleft$ is always the same. During the movement we can check that the sequences scanned encode $\gamma_c(w, i)$ and $\gamma_c(w, i+1)$ for a value i, if this is not the case we accept.

We now show how we can simulate A working on $\triangleright w \triangleleft$ by a 1-NPDA$(k+1)$ B working on $w' := \Gamma_c(\triangleright w \triangleleft)$. For each of the k heads in the automaton A automaton B has a corresponding head and in addition B has one auxiliary head. Observe that w' when projected to the Σ-components losing all information concerning numbers consists in $|w|^c$ repetitions of the word $\triangleright w \triangleleft$. Let $n = |\triangleright w \triangleleft|$, we want to show that during the simulation of A by B the following invariant holds: When B completed the simulation of a single step of A the position of each of the k heads of A within $\triangleright w \triangleleft$ coincides with the position of the corresponding head of B on w' modulo n. This is clear for the initial configuration.

Since the positions are the same modulo n, the corresponding heads of B see which symbols are seen by the simulated automaton A. We can assume that

in the automaton A only one of the heads moves during one step. If the head makes a move to the right or stays at the same position, we do the same with the corresponding head of B. If the head of A moves to the left, we want to move the corresponding head of B $n-1$ steps to the right. This position always exists since the runtime of A is bounded by n^c. In order to move the head $n-1$ steps to the right, we move the auxiliary head right until we encounter a $\triangleright$ symbol, and then move the auxiliary head of B and the corresponding head simultaneously to the right, until the auxiliary head sees a $\triangleleft$ symbol. Hence we can always keep our condition about the positions of the heads.

We mention that we choose in our construction to use in the one-way automaton an additional head and avoided to do the counting by the push-down store in order to use this proof for other automata than push-down automata. □

We define another mapping to encode the position of two heads by a single head. Let $x, y \in \Sigma^n$ then we write (x, y) for the word

$$(x_1, y_1)(x_2, y_2)\dots(x_n, y_n) \in (\Sigma^2)^n$$

also we let $\pi_1((x_i, y_i)) = x_i$ and $\pi_2((x_i, y_i)) = y_i$.

Given a word $w_1 \dots w_n \in \Sigma^*$, we define a word over $\Sigma \times \Sigma \cup \{\$\}$. We let $v_i = (w, w_i^n)\$^n$ for $i \in \{1, \dots, n\}$. Also we let $w' = v_1 v_2 \dots v_n$, so the length of w' is $2n^2$ We let $\mu : \Sigma^* \to \Sigma''^*$ be the mapping of w to w'. Also we let $\nu_n : \{1, \dots, n\}^2 \to \{1, \dots, 2n^2\}$ be the map

$$\nu_n(i, j) = i + 2n \cdot (j - 1).$$

Thus, given two heads at the positions i and j on a word w, the construction guarantees that $w'_{\nu_n(i,j)} = (w_i, w_j)$. In the following lemma we show that we can simulate the movement of multiple heads on the word w by only one head on the word w'.

Lemma 2. 1-NPDA(1) *are densely complete for* 1-NPDA*(k).*

Proof. Pick an automaton A from 1-NPDA(2) that recognizes a language $L \subseteq (\Sigma^*)$. We will show that there is a 1-NPDA(1) B that recognizes $L' = \mu(L) \cup L_{\mathrm{BAD}}$, where $L_{\mathrm{BAD}} = \Sigma^* \backslash \mu(\Sigma^*)$. This will suffice for the proof since the mapping μ can be computed by a DLOGTIME-uniform AC^0 reduction and since $L' \leq_m^{\mathrm{AC}^0} L$ (see discussion at the beginning of this section) we get $L \approx_m^{\mathrm{AC}^0} L'$.

First, we show that we can recognize L_{BAD}. Let w' be the input of B and let $n' = |w'|$. We can use the stack of B as a counter to accept if it is not the case that each block of \$'s, and each block between the \$-blocks has the same length. Also we can check that the number of \$-blocks is the same as the number \$'s in the first \$-block. If one of these conditions is *not* fulfilled we accept the word, since we want to accept L_{BAD}. We continue to check if there are mistakes in the repetition of the word w, i.e. that $\pi_1(w'_{\nu_n(i,j)}) \neq \pi_1(w'_{\nu_n(i,j+1)})$ or $\pi_1(w'_{\nu_n(i,1)}) \neq \pi_2(w'_{\nu_n(j,i)})$ for $i, j \in \{1, \dots, n'\}$. (See appendix for details.)

Second, we show how we can simulate A working on w by a 1-NPDA(1) working on $w' = \mu(\Sigma^*)$. For the two heads of A at positions i, j we have one corresponding head in B at position $\nu_n(i, j)$, where $n = |w|$. This is clear for the initial configuration.

Since $w'_{\nu_n(i,j)} = (w_i, w_j)$, the corresponding heads sees which symbols are seen by the simulated automaton. We can assume that in the automaton we simulate only one of the head's moves during one step. If the first head makes a move to the right, we do the same. If the second head moves to the right, we want the head to move $2n$ steps to the right.

If had an additional freezing counter, we would move the head to right increasing the counter, till we see a \$ symbol. While we see \$ symbols we move the head further to the right and decrease the counter for each step. After the last \$ symbol we move further right increasing the counter with each step till the counter equals zero. Since there are n \$ symbols in a block, we move the head $2n$ positions to the right. Also a counter can be simulated with the stack and two additional stack symbols. Hence we can always keep our invariant condition about the positions of the heads.

Since we can use this reduction also in the presence of additional heads, by induction we get our result. □

Proof (Proof of Theorem 1). Since the languages accepted by 2-NPDA(k) equal the the languages in SAC^1, by the transitivily of dense completeness and the previous two lemmas the result follows. □

4 The One-Counter Languages and the class NL

In our other proofs in the previous section we use the ability of push-down automata to simulate an additional freezing counter. For One-Counter Languages we face the problem that they cannot simulate an additional freezing counter, otherwise they would be able to recognize, for example, $\{a^n b^m c^m d^n\}$ which is not a one-counter language. Nevertheless, we can show that the one-counter languages are dense in NL.

The general idea is two show that 1-NFA(2) are densely complete in NL. This is similar to the previous section. Then we use a 1-NOCA to simulate a 1-NFA(2), where we can make use of the counter since 1-NFA(2) do not posses a counter.

Lemma 3. *The languages accepted by* $\bigcup_k$ 1-NFA(k) *are densely complete for the languages accepted by* $\bigcup_k$ 2-NFA(k).

Proof. This is the same as in the proof of Lemma 1, where we used an automaton with an additional head from 1-NPDA($k+1$). □

For the following reduction we will need a new mapping. The new mapping should help us to encode the position of three heads by a two heads. This is similar to the context free case, but here we need one additional head.

Let $x, y \in \Sigma^n$ then we write (x, y) for the word $(x_1, y_1)(x_2, y_2)\dots(x_n, y_n) \in (\Sigma^2)^n$ Given a word $w_1 \dots w_n \in \Sigma^*$, we define words over $\Sigma \times \Sigma \cup \{\$_u\}$. We let $u_i = (w, w_i^n)\$^n$ for $i \in \{0, \dots, n-1\}$. Further, we define words over $\Sigma'' = \Sigma' \times \Sigma \cup \{\$_v\}$. We let $v_i = (u, w_i^{2n^2})\$_v^{2n^2}$ and $w' = v_1 v_2 \dots v_n v_1 v_2 \dots v_n$, so the length of w' is $8n^3$. Please note that we repeated $v_1 \dots v_n$ twice, this is a rather technical reason, which we explain in the following lemma.

We let $\mu' : \Sigma^* \to \Sigma''^*$ be the mapping of w to w'. Also we let $\nu'_n : \{1, \dots, n\}^3 \to \{1, \dots, 4n^3\}$ be the map

$$\nu'_n(i, j, k) = i + 2n \cdot (j-1) + 4n^2 \cdot (k-1).$$

So given three heads at the positions i, j, k on a word w by the construction $w'_{\nu_n(i,j,k)} = (w_i, w_j, w_k)$. In the following lemma we show that we can simulate the movement of three heads on the word w by only two heads on the word w'.

Lemma 4. *The languages accepted by* 1-NFA(2) *are densely complete for the languages accepted by* $\bigcup_k$ 1-NFA(k).

Proof. Pick an 1-NFA(3) A that recognizes a languages $L \subseteq (\Sigma^*)$. We will show that there is a 1-NFA(2) B that recognizes $L' = \mu'(L) \cup L_{\text{BAD}}$, where $L_{\text{BAD}} = \Sigma^* \setminus \mu'(\Sigma^*)$.

Similar to Lemma 1 we can show that we can accept all words in L_{BAD}; since we have two heads this is straight forward. (Note that Lemma 2 is different, since we have only one head there.)

Second, we show how we can simulate A working on w by a 1-NFA(2) working on $w' = \mu'(\Sigma^*)$. For the three heads of A at positions i, j, k we have one corresponding head in B at position $\nu'_n(i, j, k)$, where $n = |w|$, and one auxiliary head. This is clear for the initial configuration.

Since $w'_{\nu_n(i,j,k)} = (w_i, w_j, w_k)$, the head corresponding heads sees which symbols are seen by the simulated automaton. We can assume that in the automaton we simulate only one of the heads moves during one step. If the first head makes a move to the right, we do the same. If the second (resp. third) head moves to the right, we want the head of B $2n$ (resp. $4n^2$) steps to the right.

The auxiliary head will be at the beginning of one of the v_i. We now move both heads right till the auxiliary head is at the first letter beyond of the next $\$_u$ (resp. $\$_v$) block. If we just simulated a move of the second head of A, we will continue to move the auxiliary head, just one character beyond the next $\$_v$-block. So with every more of the second and third head of A, the auxiliary head of B move $4n^2$ symbols to the right. This is the reason we doubled the length of w' in the definition of μ'.

Using induction on the number of heads we get our result. □

Since in Lemma 2 we use the stack only to simulate a freezing counter, we can use the (otherwise unused) counter here to get the lemma:

Lemma 5. *The languages accepted by* 1-NOCA *are densely complete for the languages accepted by* 1-NFA(2).

Since the languages accepted by 2-NFA(k) are equal to the languages in NL(see e.g. [Sud77]), by the transitively of dense completeness and the previous lemmas the result follows.

Theorem 2. *The one-counter languages are densely complete for* NL.

5 The Indexed Languages and the class NP

Aho introduced the family INDEX of indexed languages and characterized them by nondeterministic nested stack automata ([Aho68,Aho69]). The indexed languages coincide with Fisher's class OI of outside-in macro-languages ([Fis68]). The indexed languages are contained in NP and there are NP-complete indexed languages ([Rou73]).

Theorem 3. *The indexed languages are densely complete for* NP.

Proof sketch. (For details see appendix.) The starting point is that polynomial time bounded nondeterministic two-way nested stack multihead automata accept exactly the languages in NP. Again, we first reduce densely the polynomial time bounded nondeterministic two-way nested stack multihead automata to the corresponding one-way model.

In the second step, we decrease the number of heads to one preserving density by the use of a freezing counter. Pushdown automata can simulate a freezing counter by pushing an intermediate designated stack symbol as described in the previous section. The same observation holds for nested stack automata which gives us the dense NP-completeness of the languages accepted by nondeterministic nested stack automata (with one-way input head), which are the indexed languages. □

Observe that the simulation of a freezing counter within the nested stack automaton does not go through for other variants of stack automata , like stack automata and non-erasing stack automata.

6 The Regular Languages and the class NC^1

The regular languages are complete for NC^1, since every regular language with a non-solvable monoid is complete for NC^1 ([Bar89]). In this section we will prove unconditionally that the regular languages are not densely complete in NC^1. For regular languages we have the fact that a language is either in AC^0, or its syntactic monoids is non-aperiodic ([Sch65]). While this is a rather sharp boundary on the formal language side, on the complexity side we can find languages very close to AC^0 but still outside. Similar to Ladner's theorem [Lad75] that shows if $P \neq NP$ we can find languages that are neither in P nor NP-complete, we construct a language that is neither in AC^0 nor any non-aperiodic regular language can be reduced to it.

Theorem 4. *The regular languages are not densely complete in* NC^1.

Proof. Assume by contradiction that the regular languages are densely complete in NC^1. Then for every language $L \in \mathrm{NC}^1$ there is a regular language R such that there are DLOGTIME-uniform AC^0 reductions from L to R and conversely.

The idea is to find a language in NC^1 that is not many-one equivalent to any regular language. We apply Ladner's result as generalized by Vollmer in [Vol90] to the parity-language, yielding a language $L \subseteq L_{parity}$, such that L is reducible to L_{parity} but not vice versa.

Essentially $L = \{w \in L_{parity} \mid |w| \in F\}$ where F is a set of positive integers such that both F and its complement consist in intervals of rapidly growing size. Observe that L is not in AC^0 by Håstad's result ([Hås86]), but clearly in ACC^0_2 since it is AC^0-reducible to L_{parity}.

By assumption there is a regular set R which is many-one equivalent to L. Let M be the (finite) syntactic monoid of R. There are three cases to consider.
Case 1: M contains no group. Then R is aperiodic and hence $R \in \mathrm{AC}^0$ and $L \in \mathrm{AC}^0$ ([Str94],[Sch65]) - a contradiction.
Case 2: M contains a group of even order, then M also contains Z_2. But then L_{parity} is recognizable by M, and thus is many-one reducible to R and hence to L - a contradiction.
Case 3: M contains a group of some odd order. Then for some odd n the cyclic group Z_n divides M, thus we could reduce a language whose syntactic monoid is Z_n to the language R. And hence also to L_{parity} in contradiction to [Smo87], who proved that ACC^0_2 cannot count modulo any odd number. □

We can apply the exact same proof to any complexity class between NC^1 and ACC^0_2 and its corresponding class within the regular languages. For example the regular languages with solvable syntactic monoid are complete for ACC^0 ([BCST92]) , but the proof of the previous theorem shows:

Theorem 5. *The regular languages with solvable syntactic monoid are not densely complete in* ACC^0.

We are left with class AC^0 and the aperiodic or star-free regular languages. In order to treat this question it would be necessary to use a finer notion of reducibility since with respect to AC^0-reducibility every nonempty subset of AC^0 is densely complete in AC^0.

7 Discussion

The principal construction used in this paper to establish the dense completeness of a language family F in a complexity class C was transforming a language $L \in C$ by padding-like repetitions into some $L' \in F$ such that L' was the union of the transformation of L and the set L_{BAD} of ill-formed words which are not in the image of the transformation. This construction needs F to be closed under union and to contain L_{BAD}. This could force F both to be of a non-deterministic

nature and to be able to simulate a counter which would induce the NL-hardness of F.

This may explain the fact that the regular languages are not dense in their complexity class NC^1. Considering the similarities between the regular, contextfree, and indexed languages (the contextfree languages are the yields of the regular tree languages and the indexed languages are the yields of the contextfree tree languages) this is a bit surprising.

This still leaves open the question whether there exists a dense subfamily of formal languages in NC^1. A natural candidate are the visibly pushdown languages VPL studied in [AM04], which are contained in NC^1 and contain NC^1-hard languages since all regular languages are in VPL. They were characterized in [AKMV05] by the finiteness of a certain congruence relation. We conjecture that the visibly push-down languages are not dense in NC^1. It might be possible to show along the lines of the proof of Theorem 4 a conditional result like "If $\mathrm{TC}^0 \neq \mathrm{NC}^1$ then VPL is not dense in NC^1". If it were possible to exhibit a dense subfamily of formal languages in NC^1 one should not expect this to be a trio, since the trio generated by the mirror language, i.e. the set of all palindromes (as a simple example of a nonregular language) already contains NL-complete languages.

We left open the following question: what is a class of formal languages? Our decision to require the class to form a trio might be too restrictive since it seems to exclude deterministic classes. Another try would for instance be closure under intersection with regular sets only. But the following construction (which is only indicated and needs more details for a real proof) then provides us an arbitrary finitely presented class C with a dense subfamily F: Set $T(w) := w_1\$w_2$ where $w = w_1w_2$ and $|w_1| = |w_2|$ if $|w|$ is even resp. $|w_1| + 1 = |w_2|$ otherwise. Then the class $F := \{T(L) \cup L_{\mathrm{BAD}} \mid L \in C\} \cup \{\emptyset\}$ is dense in C, has a decidable emptiness problem, and is closed under intersection with regular sets. In this case, L_{BAD} is the set of words containing more than one \$-symbol, no \$-symbol, or one \$-symbol not in the middle of a word. But this class F is just a complexity class in disguise without any well-behavior of a typical formal language class.

Our results should not be regarded as an approach to separate complexity classes: If you could prove that NC^1 does not contain a dense subfamily of formal languages, you would have separated NC^1 from NL. This only shows how hard it will be to show that certain complexity classes do not have dense subfamilies of formal languages. But this motivates the question for the converse, i.e.: can we show under some standard assumptions from complexity theory non-denseness results for certain complexity classes? Can we for example show that $\mathrm{NC}^1 \neq \mathrm{NL}$ implies that NC^1 does not contain a dense subfamily of formal languages?

References

[Aho68] Alfred V. Aho. Indexed grammars - an extension of context-free grammars. *J. ACM*, 15(4):647–671, 1968.

[Aho69] Alfred V. Aho. Nested stack automata. *J. ACM*, 16(3):383–406, 1969.

[AKMV05] Rajeev Alur, Viraj Kumar, P. Madhusudan, and Mahesh Viswanathan. Congruences for visibly pushdown languages. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 1102–1114. Springer, 2005.

[AM04] Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In László Babai, editor, *STOC*, pages 202–211. ACM, 2004.

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

[BCST92] David A. Mix Barrington, Kevin J. Compton, Howard Straubing, and Denis Thérien. Regular languages in nc^1. *J. Comput. Syst. Sci.*, 44(3):478–499, 1992.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990.

[Coo71] Stephen A. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *J. ACM*, 18(1):4–18, 1971.

[Eil76] Samuel Eilenberg. *Automata, Languages and Machines, Vol. A+B.* Academic Press, 1976.

[Fis68] Michael J. Fischer. Grammars with macro-like productions. In *SWAT (FOCS)*, pages 131–142. IEEE Computer Society, 1968.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20. ACM, 1986.

[Joh90] David S. Johnson. A catalog of complexity classes. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 67–161. The MIT Press, 1990.

[Lad75] Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

[Lan93] Klaus-Jörn Lange. Complexity and structure in formal language theory. In *Structure in Complexity Theory Conference*, pages 224–238, 1993.

[Pin86] Jean-Eric Pin. *Varieties of formal languages.* Plenum, London, 1986.

[Rei08] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

[Rou73] William C. Rounds. Complexity of recognition in intermediate-level languages. In *SWAT (FOCS)*, pages 145–158. IEEE Computer Society, 1973.

[Sch65] Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *STOC*, pages 77–82, 1987.

[Str94] Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, Boston, 1994.

[Sud77] Ivan Hal Sudborough. Some remarks on multihead automata. *ITA*, 11(3):181–195, 1977.

[Sud78] Ivan Hal Sudborough. On the tape complexity of deterministic context-free languages. *J. ACM*, 25(3):405–414, 1978.

[Tan07] Till Tantau. Logspace optimization problems and their approximability properties. *Theory Comput. Syst.*, 41(2):327–350, 2007.

[Ven91] H. Venkateswaran. Properties that characterize logcfl. *J. Comput. Syst. Sci.*, 43(2):380–404, 1991.

[Vol90] Heribert Vollmer. The gap-language-technique revisited. In Egon Börger, Hans Kleine Büning, Michael M. Richter, and Wolfgang Schönfeld, editors,

CSL, volume 533 of *Lecture Notes in Computer Science*, pages 389–399. Springer, 1990.