On the Existence of Regular Approximations [1]

Brendan Cordy [a] Kai Salomaa [b]

[a] *Department of Mathematics and Statistics, McGill University, 805 Sherbrooke W., Montreal, Quebec H3A 2K6, Canada*

[b] *School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada*

Abstract

We approximate context-free, or more general, languages using finite automata. The degree of approximation is measured, roughly speaking, by counting the number of incorrect answers an automaton gives on inputs of length m and observing how these values behave for large m. More restrictive variants are obtained by requiring that the automaton never accepts words outside the language or that it accepts all words in the language. A further distinction is whether a given (context-free) language has a regular approximation which is optimal under the measure of approximation degree or an approximation which is arbitrarily close to optimal. We study closure and decision properties of the approximation measure.

Key words: regular languages, degree of approximation, decidability

1 Introduction

Different ways of measuring levels of reliability of finite automata have been considered in [9,10]. By allowing a finite automaton to give incorrect answers on some inputs we may obtain significant savings in the state-complexity of a language. Here, instead of comparing state-complexity of representations with different reliability, we consider various ways to give approximate representations of non-regular languages using finite automata. Strong non-approximability results for certain non-regular languages have been previously obtained in [7].

Email addresses: bcordy@math.mcgill.ca (Brendan Cordy), ksalomaa@cs.queensu.ca (Kai Salomaa).

[1] A preliminary version of this paper appeared in DCFS 2006 [4].

For the purpose of evaluating how well a language is approximated by another language, we need a way to measure the similarity of two languages. The existence of "good" approximations for non-regular languages naturally depends very much on the measures used to compare languages. We present examples for different types of measurements where good regular approximations exist or do not exist, respectively. We use variants of the measures considered in [7,9,10]. The measures, roughly speaking, count the number of inputs of a given length for which a finite automaton gives an incorrect answer, and then see how this value behaves for long inputs in the limit. We make some modifications to the earlier definitions, mainly, in order to avoid trivial good approximations. This issue did not occur in [7,9,10], perhaps, because the work there was concerned on the one hand with negative approximability results, that is results about the non-existence of approximations, and on the other hand with worst-case results for the descriptional complexity of approximations having different reliability. Thus, the work in [7,9,10] was for the most part not dealing with positive results on the existence of approximations.

We distinguish approximations of a language L by requiring that the approximation has to contain all strings in L (top approximation); the approximation must be included in L (bottom approximation); or by imposing no restriction (mixed approximation). A further distinction is whether L has an approximation that is optimal under the approximation degree or approximations that can become arbitrarily close to optimal (an approximation in the limit). We study closure properties of the different variants of approximability and establish an undecidability result.

Other types of regular approximations of languages have been considered e.g. in [1–3,5,6,12–14]. The automaticity descriptional complexity measure [14] and the model of cover automata [3] count the number of states of finite automata that recognize approximations of the language to be represented. Automaticity is a descriptional complexity measure for arbitrary languages whereas cover automata are used as an implementing method to reduce the size of automata that represent finite languages. These models differ from our approach in that the approximations are required to be correct for all words up to length n (with variable n) and do not consider the number of incorrect answers on words longer than n. The paper [6] investigates minimal covers and the approach is quite different from the measures considered here, as well. The paper [1] considers metrics that are required to have an additivity property with respect to catenation. This property guarantees that the metric preserves regularity of languages which means it would not be useful for our current purpose. In the context of probabilistic languages, it is common to use the relative entropy as a distance measure between languages [5]. Work along more practical lines on regular approximations of context-free languages can be found in [11].

The work [12,13] investigates lower and upper rough set approximations that converge to a given language L. Interestingly, it turns out that in most cases the lower and upper approximations of context-free languages are regular. It would be a useful topic for further research to determine how the rough set approximations relate to the measures we consider here.

2 Degree of approximation

In the following Σ denotes a finite alphabet and Σ^* is the set of all finite strings over Σ. The length of a string w is the number of occurrences of symbols of Σ in w and it is denoted $|w|$. The reversal (or mirror-image) of $w \in \Sigma^*$ is denoted w^R. For $L \subseteq \Sigma^*$ and $m \geq 0$, we denote by $|L|_m$ the number of strings of length m in L. An alphabet is said to be *minimal* for a language $L \subseteq \Sigma^*$ if each symbol of Σ occurs in some string of L. The symmetric difference of sets A and B is $A \triangle B$. The cardinality of a finite set is $|A|$.

A deterministic finite automaton (DFA) is a tuple $M = (\Sigma, Q, q_0, Q_F, \delta)$ where Σ is the input alphabet, Q is the finite set of states, $q_0 \in Q$ is the start state, $Q_F \subseteq Q$ is the set of accepting states, and $\delta : Q \times \Sigma \longrightarrow Q$ is the state transition function. The function δ is extended in the natural way to a function $Q \times \Sigma^* \longrightarrow Q$ and the language recognized by the DFA M is $L(M) = \{w \in \Sigma^* \mid \delta(q_0, w) \in Q_F\}$. Note that a DFA as defined above is complete, i.e., $\delta(q, \sigma)$ is defined for all $q \in Q$, $\sigma \in \Sigma$. The deterministic finite automata recognize exactly all the regular languages. For all unexplained notions in language theory we refer the reader e.g. to [8,15].

For integers $m, n \geq 0$, we define the "non-zero minimum" of m and n as $\min_1(m, n) = \max(\min(m, n), 1)$.

When a finite automaton A is used as an approximation of a language L, the value $| L(A) \triangle L |_m$ gives the number of strings of length m for which A gives an incorrect answer. More generally, if R and L are two languages over Σ, we define the *degree of approximation of L by R* as the quantity

$$App_\Sigma(R, L) = \limsup_{n \to \infty} \frac{| R \triangle L |_n}{\min_1(| L |_n, | \Sigma^* - L |_n)} \tag{1}$$

When the alphabet Σ is known from the context we denote the approximation degree simply as $App(R, L)$. It can be noted that the degree-of-approximation relation is not symmetric. In our terminology, a degree-of-approximation value (close to) zero means a "good approximation". The definition of $App(R, L)$ does not require that R is regular, however, in the following we are mainly concerned with cases where the language used as an approximation is regular.

The relation (1) resembles the reliability measure of [10] or the approximation measure used in [7]. These measures use the number of all words of length n, $\mid \Sigma^* \mid_n$ (or equivalently $\mid \Sigma \mid^n$), as the denominator. The main reason for introducing the definition (1) is that if we would use $\limsup_{n\to\infty} \frac{\mid R \triangle L \mid_n}{\mid \Sigma^* \mid_n}$ as the right side of (1), all very "dense" languages would always have Σ^* as a good approximation and, similarly, all "sparse" languages could be approximated by $\emptyset$. Above by a dense (respectively, sparse) language we mean a language L such that the limit of $\mid \Sigma^* - L \mid_n$ (respectively, $\mid L \mid_n$) divided by $|\Sigma|^n$ approaches zero when n approaches infinity.

In fact, the degree of approximation can be thought of as a measure of how well some language R approximates L relative to the best trivial approximation ($\emptyset$ or Σ^*). If we were to use the maximum of $\mid L \mid_n$ and $\mid \Sigma^* - L \mid_n$ in the denominator in place of the minimum, this could be viewed as comparing an approximation relative to the worse of the two trivial approximations.

In (1) we use the non-zero minimum in the denominator because it is possible that $\min(\mid L \mid_n, \mid \Sigma^* - L \mid_n)$ is zero. In particular, if L contains no words of length n (or all words of length n) where n ranges over infinitely many non-periodic values, this could cause the approximation degree to become infinite for any regular language. This could happen even if L is very sparse, in which case $\emptyset$ would intuitively be a "reasonably good" approximation of L.

First we consider the question of what values the approximation degree can have. The below example gives a construction showing that $App(R, L)$ with R regular and L context-free can have any rational value between 0 and 1.

Example 2.1 Let $\Sigma = \{0, 1, b, c\}$. For any integers $0 \leq x \leq n$ we construct a regular language R and a context-free language L over Σ such that $App(R, L) = \frac{x}{n}$.

Let $f : \{1, \ldots, n\} \longrightarrow \{0, 1\}^r$ be a bijective mapping where $r = \lceil \log n \rceil$. We define

$$L = \bigcup_{i=1}^{n} f(i)\{b^j c^k \mid j \neq k, \quad j, k \geq 0\}.$$

All strings in L begin with a sequence of r symbols 0 and 1, followed by a string of b's and c's. Let $0 \leq k \leq n$ and define

$$R_k = \bigcup_{i=1}^{k} f(i) b^* c^*.$$

Let $u = r + 2m$, $m \geq 1$. All strings of L having length u begin with a sequence of r symbols 0 and 1, and followed by a string of i symbols b and j symbols c where $i + j = 2m$ and $i \neq j$. Thus,

$$\mid L \mid_u = n \cdot (2m). \tag{2}$$

The set $\mid R_k \triangle L \mid_u$ has all strings of L of length u beginning with $f(i)$, $k < i \leq n$, and for any $1 \leq i \leq k$ it contains one string $f(i)b^j c^j$, where $j = m$. Thus,

$$\mid R_k \triangle L \mid_u = (n-k) \cdot (2m) + k. \tag{3}$$

Next let $v = r + 2m - 1$, $m \geq 1$. Strings of L having length v again begin with a sequence of r symbols 0 and 1, followed by all strings in b^*c^* having length $2m-1$. Thus,

$$\mid L \mid_v = n \cdot (2m). \tag{4}$$

Also, similarly as above it is easy to see that

$$\mid R_k \triangle L \mid_v = (n-k) \cdot (2m). \tag{5}$$

Always when $i > r$, $\mid L \mid_i < \mid \Sigma^* \mid_i$. Hence using (2), (3), (4), and (5), we see that the $(r+2m)^{th}$ term in the limit on the right side of (1) is $\frac{2m(n-k)+k}{2mn}$ and the $(r+2m-1)^{th}$ term is $\frac{2m(n-k)}{2mn}$. Thus as the limit we obtain

$$App(R_k, L) = \frac{n-k}{n}.$$

Since k can be an arbitrary integer between 0 and n, the claim follows. ■

Similar to Example 2.1 we can, of course, construct languages R and L such that $App(R, L)$ is any rational number greater than one. This can be done by making $R \triangle L$ sufficiently large, and this would correspond to situations where R is a bad approximation of L.

Next we define what we mean by "good" regular approximations of a language L. We distinguish the notions of *top approximation* that contains L, *bottom approximation* that is contained in L and *mixed approximation* (or just approximation) that can have any relation with L.

Definition 2.1 *Let $L \subseteq \Sigma^*$. The language L has a* regular m-approximation *(mixed approximation) if there exists a regular language R such that $App(R, L) = 0$. We say that L has a* regular t-approximation, *or top approximation, (respectively,* b-approximation, *or bottom approximation) if above R can be chosen such that $L \subseteq R$ (respectively, $R \subseteq L$).*

The language L is said to have a regular m-approximation in the limit *if for any $\epsilon > 0$ there exists a regular language R_ϵ such that $App(R_\epsilon, L) < \epsilon$. Again we say that L has a regular* t-approximation *(respectively,* b-approximation*)*

in the limit *if above R_ϵ can always be chosen such that $L \subseteq R_\epsilon$ (respectively, $R_\epsilon \subseteq L$).*

In the following, we call regular x-approximations simply x-approximations, where $x \in \{m, b, t\}$. Note that any t-approximation or b-approximation is by definition also an m-approximation.

We observe that it is possible that $L \subseteq \Sigma^*$ does not have an x-approximation when viewed as a language over Σ, but L has an x-approximation over some larger alphabet. For example, choose $\Sigma = \{a, b\}$ and

$$L_1 = \Sigma^* - \{a^n b^n \mid n \geq 0\}.$$

Now for all $n \geq 1$, $\min_1(|L_1|_n, |\Sigma^* - L_1|_n) = 1$, and for any regular language R the value of $App_\Sigma(R, L_1)$ cannot be zero. On the other hand, consider L_1 to be a language over the alphabet $\Omega = \{a, b, c\}$ and choose $R_1 = \{a, b\}^*$. Now for large values of n, $|L_1|_n \leq |\Omega^* - L_1|_n$ and hence $App_\Omega(R_1, L_1) = 0$.

The above situation is caused by the fact that $|L_1|_n$, $n \geq 0$, is close to $|\Sigma^*|_n$ where Σ is the alphabet containing all symbols occuring in words of L_1.

Lemma 2.1 *Let $L \subseteq \Sigma^*$ where Σ is the minimal alphabet for L and $x \in \{m, b, t\}$. If there is a constant n_0 such that $|L|_n \leq \frac{1}{2}|\Sigma^*|_n$ for all $n \geq n_0$, then L has an x-approximation when viewed as a language over Σ if and only if L has an x-approximation when viewed as a language over any extension of Σ.*

Proof. When $n \geq n_0$, the term in the denominator of (1) will be $|L|_n$ (or 1), and this holds independently of whether L is viewed as a language over Σ or over some extension of Σ. ■

Lemma 2.1 means, roughly speaking, that extending the alphabet does not affect the existence of approximations, so long as the language in question contains at most half of the words of any given length in the original alphabet. In the following sections we often implicitly rely on Lemma 2.1 in our constructions.

To conclude this section we recall a result from [7]. Denote

$$L_{majority} = \{w \in \{a, b\}^* \mid w \text{ has more } a\text{'s than } b\text{'s }\}.$$

In [7] it is shown that for any regular language $R \subseteq \Sigma^*$,

$$\lim_{n \to \infty} \frac{| R \triangle L_{majority} |_n}{| \Sigma^* |_n} = \frac{1}{2}.$$

Using our definitions, this gives the following corollary. Note that the property of having an m-approximation in the limit defines the largest class of languages among the different properties considered in Definition 2.1.

Corollary 2.1 [7] *The language $L_{majority}$ does not have an m-approximation in the limit.*

3 Different types of approximations

We show that the classes of languages having different types of approximations as introduced in Definition 2.1 are distinct. First we show that there exist languages having top approximations but no bottom approximations and vice versa. We consider the linear context-free language

$$T = \{a^i b^j \mid i, j \geq 0, i \neq j\} \tag{6}$$

Example 3.1 We show that $R_t = a^*b^*$ is a t-approximation of T. We note that $T \subseteq R_t$ and $\mid R_t \triangle T \mid_n$, $n \geq 0$, is always 0 or 1. The number of strings in T of length m is either m or $m+1$ and clearly $\mid T \mid_m < \mid \Sigma^* - T \mid_m$, for all $m \geq 2$. Thus

$$App(R_t, T) = \limsup_{m \to \infty} \frac{1}{\mid T \mid_m} \leq \limsup_{m \to \infty} \frac{1}{m} = 0.$$

Lemma 3.1 *The language T from (6) has no b-approximation.*

Proof. Let R_b be any regular language such that $R_b \subseteq T$. It is sufficient to show that $App(R_b, T) > 0$.

Let $\Sigma = \{a, b\}$ and $M = (\Sigma, Q, q_0, Q_F, \delta)$ be a complete DFA that recognizes R_b. Consider computations of M on strings in a^* and find the first state that repeats. That is, we find $0 \leq i < |Q|$ and $0 < j \leq |Q|$ such that $\delta(q_0, a^i) = p_1$ and $\delta(p_1, a^j) = p_1$. Since M is complete, the state p_1 always exists.

Now we consider computations of M starting from state p_1 on strings in b^* and find the first cycle. We find $0 \leq k < |Q|$ and $0 < m \leq |Q|$ such that $\delta(p_1, b^k) = p_2$ and $\delta(p_2, b^m) = p_2$.

We choose an integer r as follows. If $k \leq i$, then $r = i - k$. If $k > i$, then we choose r to be the smallest integer such that

$$j \text{ divides } k - i + r. \tag{7}$$

The integer r can always be found such that $r \leq |Q|$. Now there exists $p_3 \in Q$ such that for all $x, y \geq 0$, we have

$$\delta(q_0, a^{i+xj}b^{k+ym+r}) = p_3. \tag{8}$$

By (7), we can choose x such that $i + xj = k + r + 0 \cdot m$. Thus if p_3 is an accepting state, $R_b = L(M)$ contains some string not in T and R_b is not a b-approximation. Hence p_3 cannot be an accepting state and from (8) we know that R_b does not contain any strings of the form $a^{i+xj}b^{k+ym+r}$, where $x, y \geq 0$.

Let z be an arbitrary positive integer and denote

$$n = i + k + r + z \cdot j \cdot m.$$

We know that T contains, in total, n or $n+1$ strings of length n. By (8) we know that the following strings of length n cannot be in R_b:

$$a^i b^{k+r+zjm}, \quad a^{i+jm}b^{k+r+(z-1)jm}, \quad \ldots, \quad a^{i+zjm}b^{k+r},$$

and it follows that $| R_b \triangle T |_n \geq z + 1$. Since for large n, $| T |_n < | \Sigma^* - T |_n$, we get

$$\begin{aligned} App(R_b, T) &= \limsup_{n\to\infty} \frac{| R_b \triangle T |_n}{| T |_n} \geq \limsup_{n\to\infty} \frac{| R_b \triangle T |_n}{n+1} \\ &\geq \limsup_{z\to\infty} \frac{z+1}{i+k+r+z\cdot j\cdot m+1} = \frac{1}{jm} > 0. \end{aligned}$$

■

The above proof shows that the value of $App(R_b, T)$ is positive, but does not give any positive lower bound for it. Indeed it is easy to see that the language T has a b-approximation in the limit. If a DFA M_k checks that the input is of the form $a^i b^j$ where $i \not\equiv j$ modulo some of the integers $2, \ldots, k$, then $L(M_k) \subseteq T$ and for large enough k, $App(L(M_k), T)$ becomes smaller than any given positive constant.

Above we have seen that the language T has a t-approximation but no b-approximation. This naturally begs the question whether there exists a language with a b-approximation but no t-approximation. The following correspondence between b- and t-approximations turns out to be useful.

Proposition 3.1 *Let L be a language over the alphabet Σ.*

(i) *If L has a t-approximation, then $\Sigma^* - L$ has a b-approximation.*
(ii) *If L has a b-approximation, then $\Sigma^* - L$ has a t-approximation.*

Proof. Assume that R_t is a regular language such that $L \subseteq R_t$ and $App(R_t, L) = 0$. We observe that $(\Sigma^* - L) \triangle (\Sigma^* - R_t) = L \triangle R_t = R_t - L$. Thus,

$$\begin{aligned} App(\Sigma^* - R_t, \Sigma^* - L) &= \limsup_{m\to\infty} \frac{|\ (\Sigma^* - R_t) \triangle (\Sigma^* - L)\ |_m}{\min_1(|\ \Sigma^* - L\ |_m, |\ L\ |_m)} \\ &= \limsup_{m\to\infty} \frac{|\ R_t \triangle L\ |_m}{\min_1(|\ L\ |_m, |\ \Sigma^* - L\ |_m)} = App(R_t, L) = 0. \end{aligned}$$

This proves (i) since $\Sigma^* - R_t \subseteq \Sigma^* - L$ and $\Sigma^* - R_t$ is regular. The case (ii) is completely symmetric. ■

Now Example 3.1, Lemma 3.1 and Proposition 3.1 give the following.

Corollary 3.1 *With T as in (6), the language $\Sigma^* - T$ has a b-approximation but no t-approximation.*

Next we address the question of whether there exist languages having an m-approximation but no b- or t-approximations. In the following let $\Sigma = \{a, b, c, d\}$ and denote

$$X = \{a^i b^i \mid i \geq 0\} \cup \{c^i d^j \mid i \neq j,\ \ i, j \geq 0\} \tag{9}$$

Example 3.2 The language c^*d^* is an m-approximation of X. To see this we observe that for any even length $2n$, c^*d^* contains all strings of X except $a^n b^n$, and for any odd length $2n+1$, c^*d^* contains all strings of X. Similarly, for any even length $2n$, c^*d^* contains the string $c^n d^n$ not in X and for any odd length $2n+1$, all strings of length $2n+1$ in c^*d^* are also in X. Since for all non-negative integers n, $|\ X\ |_n = n + 1$, we get

$$App(c^*d^*, X) \leq \limsup_{n\to\infty} \frac{2}{n+1} = 0.$$

Lemma 3.2 *The language X as in (9) does not have a b-approximation or a t-approximation.*

Proof. First we show that X does not have a b-approximation. Let R_b be any regular language such that $R_b \subseteq X$ and let M be a complete DFA having q states, that recognizes R_b. Exactly as in the proof of Lemma 3.1 we can find integers $i, j, k, m, r \leq q$, where j divides $(k - i + r)$ such that M reaches the same state p after reading any string $c^{i+xj} d^{k+ym+r}$ independently of the values $x, y \geq 0$, and it is then observed that p cannot be an accepting state. Then similarly as in the proof of Lemma 3.1 we can calculate that $App(R_b, X)$ is at least $\frac{1}{j \cdot m}$. The only difference in the estimation is that now, for any length n, X contains exactly $n+1$ strings of length n, and the value $n+1$ was used in the estimation for the lower bound in the proof of Lemma 3.1.

Next we show that the language X cannot have a t-approximation. Assume that R_t is a regular language, $X \subseteq R_t$ and let $M = (\Sigma, Q, q_0, Q_F, \delta)$ be a complete DFA recognizing R_t. We consider computations of M on strings of a^* and find the first cycle. That is, we find $0 \leq i < |Q|$, $1 \leq m \leq |Q|$ such that for some $p_1 \in Q$, $\delta(q_0, a^i) = p_1$ and $\delta(p_1, a^m) = p_1$.

Then we consider computations of M starting in state p_1 on strings in b^* and find the first cycle. That is, we find $0 \leq j < |Q|$ and $1 \leq n \leq |Q|$ such that for some $p_2 \in Q$, $\delta(p_1, b^j) = p_2$ and $\delta(p_2, b^n) = p_2$.

Thus for all $x, y \geq 0$, $\delta(q_0, a^{i+xm}b^{j+yn}) = p_2$. For large values of x, $i + xm > j$. Since $X \subseteq L(M)$ and X contains all strings $a^r b^r$, $r \geq 0$, this implies that an accepting state must be reachable from p_2 by reading a string of b's. Thus there exists $k < |Q|$ and $p_3 \in Q_F$ such that for all $x, y \geq 0$,

$$\delta(q_0, a^{i+xm}b^{j+k+yn}) = p_3. \tag{10}$$

Let z be a positive integer and denote

$$n_z = i + j + k + z \cdot m \cdot n.$$

By (10) we know that M must accept the following strings of length n_z,

$$a^i b^{j+k+zmn}, \quad a^{i+mn}b^{j+k+(z-1)mn}, \quad \ldots, \quad a^{i+zmn}b^{j+k}.$$

At most one of the above $z+1$ strings can be in X and hence we conclude that for any integer n_z, $| L(M) \triangle X |_{n_z} \geq z$. Now we get

$$\begin{aligned} App(R_t, X) &\geq \limsup_{z\to\infty} \frac{| R_t \triangle X |_{n_z}}{\min_1(| X |_{n_z}, | \Sigma^* - X |_{n_z})} \\ &\geq \limsup_{z\to\infty} \frac{z}{1 + i + j + k + zmn} = \frac{1}{mn} > 0. \end{aligned}$$

■

For easier readability, the language X used in Example 3.2 and Lemma 3.2 is defined over a four letter alphabet. By using a simple coding, exactly the same argument can be used to show that there is a language over a binary alphabet having an m-approximation but no b-approximation or t-approximation.

Next we show that there exist context-free languages that do not have m-approximations, or even m-approximations in the limit. A convenient language for this purpose is

$$L_0 = \{a^k b^k \mid k \geq 0\}. \tag{11}$$

Lemma 3.3 *Let $\Sigma = \{a, b\}$. The language L_0 does not have an m-approximation in the limit.*

Proof. Observe for all $n \geq 1$, $\min_1(\mid L_0 \mid_n, \mid \Sigma^* - L_0 \mid_n) = 1$, and we have for any language R,

$$App(R, L_0) = \limsup_{n\to\infty} \frac{\mid R \triangle L_0 \mid_n}{\min_1(\mid L_0 \mid_n, \mid \Sigma^* - L_0 \mid_n)} = \limsup_{n\to\infty} \mid R \triangle L_0 \mid_n.$$

Clearly then, $App(R, L_0) < 1$ if and only if there is a positive integer n_0 such that $\mid R \triangle L_0 \mid_n = 0$ for all $n > n_0$. But this means that R and L_0 differ only on finitely many strings, and therefore R must be non-regular. ■

It can be argued that L_0 has no m-approximations (in the limit) because of the fact that L_0 has at most one string of any length, and this is of course what makes the proof quite easy. We have established the following result in [4].

Lemma 3.4 [4] *Let $L_0' = \{w\$w^R \mid w \in \{a, b\}^*\}$. The language L_0' does not have an m-approximation in the limit.*

The proof of Lemma 3.4 uses a density based argument. The language L_0' has, roughly, $2^{\frac{n}{2}}$ words of length n which makes a density argument straightforward to use. The proof required to show that $L_{majority}$ does not have an m-approximation in the limit is considerably more involved, see [7].

Combining the results of Lemmas 3.1, 3.2 and 3.3, Corollary 3.1 and Examples 3.1 and 3.2 we can summarize the situation as follows.

Theorem 3.1 *There exist context-free languages L_t, L_b, L_m and L_0 such that*

(i) *L_t has a t-approximation but no b-approximation.*
(ii) *L_b has a b-approximation but no t-approximation.*
(iii) *L_m has an m-approximation but no b-approximation or t-approximation.*
(iv) *L_0 does not have any m-approximation, and not even an m-approximation in the limit.*

Since t- and b-approximations are always also m-approximations, Theorem 3.1 says, in particular, that for any combination of $x, y \in \{t, b, m\}$ such that $x \neq y$ and x-approximations are not a special case of y-approximations, there exists a context-free language L such that L has an x-approximation and L does not have an y-approximation.

In Lemma 3.1 we saw that the language T from (6) does not have a b-approximation, and after this result it was observed that T, on the other hand, has a b-approximation in the limit. Using the correspondence between b- and t-approximations as in Proposition 3.1 it follows that there exist languages that

do not have a t-approximation but do have a t-approximation in the limit. Theorem 3.1 (iv) leaves open the question whether there exist languages having an m-approximation in the limit and do not have an m-approximation. When using a density argument, as in the proof of Lemma 3.4, to establish the non-existence of an m-approximation, it is not clear whether the same languages can have an m-approximation in the limit, and this is clearly the case also for the language L_0 from Lemma 3.3.

When considering the types of approximations introduced in Definition 2.1, a language has the strongest approximation properties when it has both a t-approximation and a b-approximation. The obvious question is then whether non-regular languages can have both t- and b-approximations. This is answered affirmatively by the following example.

Example 3.3 Let $\Sigma = \{a, b, c, d\}$ and

$$L_1 = \{a^i b^i \mid i \geq 0\} \cup \{c, d\}^*. \tag{12}$$

Denote $L_2 = \{c, d\}^*$ and $L_3 = a^*b^* \cup \{c, d\}^*$. Clearly $L_2 \subseteq L_1 \subseteq L_3$. It is easy to verify that L_2 is a b-approximation and L_3 is a t-approximation of L_1.

Corollary 3.2 *There exist non-regular context-free languages that have both a t-approximation and a b-approximation.*

4 Closure properties

Let $x \in \{b, t, m\}$. If R_i is an x-approximation of L_i, $1 \leq i \leq n$, it is natural to ask whether $\sigma(R_1, \ldots, R_n)$ is an x-approximation for $\sigma(L_1, \ldots, L_n)$ when σ is some n-ary operation on languages. If σ has this property we say that σ is *x-approximation preserving.* More generally we say that σ is *x-approximability preserving,* if $\sigma(L_1, \ldots, L_n)$ has an x-approximation whenever the languages L_i, $i = 1, \ldots, n$, each have an x-approximation. An approximation preserving operation is always approximability preserving but not vice versa.

Lemma 4.1 *Complementation is m-approximation preserving. Complementation is not b-approximability preserving or t-approximability preserving.*

Proof. Similarly as in the proof of Proposition 3.1 it can be computed that if R is an m-approximation of L then $\Sigma^* - R$ is an m-approximation of $\Sigma^* - L$.

Complementation is not b-approximability or t-approximability preserving by Lemma 3.1 and Corollary 3.1. ■

Lemma 4.2 *Intersection and union are not x-approximability preserving for $x \in \{m, b, t\}$.*

Proof. Let $\Sigma = \{a, b, c, d, e, f\}$ and let L_0 be as in (11). Define $L_1 = \{c, d\}^* \cup L_0$ and $L_2 = \{e, f\}^* \cup L_0$. Let $R_1 = \{c, d\}^* \cup a^*b^*$ and $R_2 = \{e, f\}^* \cup a^*b^*$. Then R_i is a t-approximation of L_i, $i = 1, 2$, and, by Lemma 3.3, $L_1 \cap L_2 = L_0$ does not have an m-approximation (and hence no t-approximation). Note that we have extended the alphabet but, by Lemma 2.1, the language L_0 has no m-approximation even with respect to the extended alphabet.

For the case of b-approximability we can choose $R_1' = \{c, d\}^*$ and $R_2' = \{e, f\}^*$ and we note that R_i' is a b-approximation of L_i, $i = 1, 2$.

Next we consider union. As in Lemma 4.1 and Proposition 3.1 we observe that R is an x-approximation of $L \subseteq \Sigma^*$ if and only if $\Sigma^* - R$ is an y-approximation of $\Sigma^* - L$ where $\{x, y\} = \{b, t\}$. Since intersection can be expressed in terms of union and complement, from the fact that intersection is not x-approximability preserving it follows that union is not y-approximability preserving, $\{x, y\} = \{b, t\}$. ■

Next we consider closure under morphisms and inverse morphisms. We need two lemmas. The proof of the first lemma is basically the same as the proof of Lemma 3.3 – observe that B has at most two strings of any given length.

Lemma 4.3 *Let $\Sigma = \{a, b, c\}$. The language*

$$B = \{a^k b^k \mid k \geq 0\} \cup c^*. \tag{13}$$

has no m-approximation.

Lemma 4.4 *Let $\Sigma = \{a, b, c, d\}$. The language $L_2 = a^*b^* \cup \{w \in \{c, d\}^* \mid |w| = 2^k, k \geq 0\}$ has no regular m-approximation.*

Proof. Note that if R is an m-approximation of L_2, then for any sufficiently large length $n = 2^k$, R contains at least 2^{n-1} words of $\{c, d\}^n$, as otherwise $App(R, L_2) \geq \frac{1}{2}$.

Now assume that R is regular and let q be the number of states of the minimal DFA for R. By the pumping lemma, each of the 2^{n-1} words of length $n = 2^k$ that is chosen to be greater than q have a decomposition $w = xyz$ where $w = xy^2z \in R$ and $|xy| \leq q$. By the pigeonhole principle, there is some length m, with $n < m < n + q$, such that R has at least $\frac{1}{q} \cdot 2^{n-1}$ strings in $\{c, d\}^m$. Since $n > q$, we know that $2^{n+1} - 2^n > q$, and hence there are no strings of

$\{c,d\}^m$ in L_2, since m cannot be a power of 2. Thus, $|R \triangle L_2|_m > \frac{1}{q} \cdot 2^{n-1}$, and since there are infinitely many lengths m with these properties, it is easy to verify that $App(R, L_2)$ will not be zero.

We have shown that L_2 cannot have an m-approximation. ■

Lemma 4.5 *Nonerasing morphisms and inverse nonerasing morphisms are not x-approximability preserving, $x \in \{b, t, m\}$.*

Proof. In Example 3.3 it was observed that the language L_1 defined by (12), with $\Sigma = \{a, b, c, d\}$, has both a b-approximation and a t-approximation. Consider a morphism $\phi : \Sigma^* \to \{a, b, c\}^*$ defined by the conditions $\phi(a) = a$, $\phi(b) = b$ and $\phi(c) = \phi(d) = c$. Then $\phi(L_1)$ is the language B given in (13), and $\phi(L_1)$ does not have an m-approximation by Lemma 4.3.

Define $M = a^*b^* \cup \{w \in c^* \mid |w| = 2^k, k \geq 0\}$. It is easy to verify that M has a b-approximation a^*b^* and a t-approximation $a^*b^* + c^*$. Now $\phi^{-1}(M)$ is the language L_2 in Lemma 4.4 and we know that $\phi^{-1}(M)$ does not have an m-approximation. ■

Finally we consider the catenation and quotient operations. Catenation is not approximability preserving and the same holds even for catenation with a regular language. Below in Lemmas 4.6 and 4.7 we use separate constructions for t- and b-approximations.

Lemma 4.6 *Let $\Sigma = \{a, b, c, d\}$ and let T be the language of (6). If $L = T \cdot \{c, d\}^*$, then L has no m-approximation.*

Proof. First we calculate an upper bound for $|L|_n$, $n \geq 0$. Note that for any $k \geq 1$, T has at most $k + 1$ strings of length k. Any word of L of length n is a catenation of a word of T of length k and a word of $\{c, d\}^*$ of length $n - k$, $0 \leq k \leq n$. Hence we get

$$\begin{aligned} |L|_n &\leq 2^n + 2 \cdot 2^{n-1} + 3 \cdot 2^{n-2} + \ldots + (n+1) \cdot 2^0 \qquad (14) \\ &= \sum_{i=0}^{n} \sum_{j=0}^{n-i} 2^j \leq 2^{n+1} + 2^n + \ldots + 2^1 + 2^0 \leq 2^{n+2} \end{aligned}$$

For the sake of contradiction assume that there exists a regular language R that is an m-approximation of L and let $M = (\Sigma, Q, q_0, Q_F, \delta)$ be a complete DFA that recognizes R. Using an argument similar to that at the beginning of the proof of Lemma 3.1, we find strings $w_1 \in T$ and $w_2 \in a^*b^* - T$ such that $\delta(q_0, w_1) = \delta(q_0, w_2)$ and denote this state by q_1.

Since $w_1 u \in L$ whenever $u \in \{c,d\}^*$, it follows that there exists a bound n_0 such that, for all $k \geq n_0$, the computations of M starting with q_1 accept at least half of the strings of $\{c,d\}^k$. Note that otherwise, using (14), the right side of (1) would be, for infinitely many lengths $n = |w_1| + k$, at least

$$\frac{\frac{1}{2} \cdot 2^{n-|w_1|}}{2^{n+2}} \tag{15}$$

and hence $App(R,L)$ could not be zero. But since, on the other hand $w_2 u \notin L$ whenever $u \in \{c,d\}^*$, the existence of the above bound n_0 implies that for lengths $n = |w_2| + k$, where $k \geq n_0$, the right side of (1) has a lower bound that is obtained from (15) by replacing $|w_1|$ with $|w_2|$. This is again impossible. ■

Lemma 4.7 *Let $\Sigma = \{a,b,c,d,e,f\}$ and define $T' = \{a,b\}^* - T$, where T is as in (6). If we define $L' = T' \cdot \{c,d,e,f\}^*$, then L' has no m-approximation.*

Proof. The proof is similar to that of Lemma 4.6 and we only outline the idea. Similarly as in (14) we establish that there exists a constant $h > 0$ such that for all $n \geq 0$, $|L'|_n \leq h \cdot 4^n = h \cdot |\{c,d,e,f\}^*|_n$. If M' is a DFA recognizing an m-approximation of L', we find strings $v_1 \in T'$ and $v_2 \in \{a,b\}^* - T'$ such that M' is in the same state q after reading v_1 and v_2, respectively. Again there are two possibilities. Either computations starting from q, for infinitely many k, reject more than half of the strings of $|\{c,d,e,f\}|_k$, or the negation of the above property holds. In both cases, as in the proof of Lemma 4.6 we see that $L(M')$ cannot be an m-approximation of L'. ■

Corollary 4.1 *Let $x \in \{b,t,m\}$. Catenation from the right with a regular language is not x-approximability preserving.*

Proof. Since, by Example 3.1, T as in (6) has a t-approximation, Lemma 4.6 implies that catenation with a regular language is not t- or m-approximability preserving. Similarly Corollary 3.1 and Lemma 4.7 give the negative result for b-approximability. ■

The above result holds completely symmetrically if we consider catenation from the left with a regular language.

We conjecture that Kleene-star is not approximability preserving. The star operation is at least not x-approximation preserving, $x \in \{m,b,t\}$ which is seen easily by considering finite languages. If L_1 and L_2 are finite, $App(L_1,L_2)$ is always zero but it is easy to choose finite $L_1 \subset L_2$ or $L_1 \supset L_2$ such that $App(L_1^*, L_2^*) > 0$.

Lemma 4.8 *Right or left quotient with respect to individual alphabet symbols is not x-approximability preserving, $x \in \{b, t, m\}$.*

Proof. Let $\Sigma = \{a, b, c\}$ and $L = \{a^i b^i c \mid i \geq 0\} \cup \{a, b\}^*$. It is easy to verify that $\{a, b\}^*$ is a b-approximation and $a^* b^* c \cup \{a, b\}^*$ is a t-approximation of L. We note that $L/\{c\}$ does not have an m-approximation by Lemma 3.3 (and Lemma 2.1). The construction for left quotient is symmetric. ■

Above we have seen that most of the standard operations on languages are neither approximation preserving nor approximability preserving. Complement is the single exception, and the fact that complementation is m-approximation preserving follows easily from the definition of the degree of approximation. We note that none of the operations considered are approximability preserving without being approximation preserving; perhaps it is worth asking whether any such natural operations do exist.

5 Undecidability

In this section we show that the question of whether a given regular language approximates a given context-free language is undecidable.

Theorem 5.1 *Let $x \in \{m, b, t\}$. Given a context-free language L and a regular language R, it is undecidable whether R is an x-approximation for L.*

Proof. Given an instance $C = (\Omega, (u_1, ..., u_n), (v_1, ..., v_n))$, $n \geq 1$, $\Omega = \{a, b\}$, of the Post correspondence problem (PCP) [8], construct the following context-free languages over the alphabet $\Sigma = \Omega \cup \{c, d\}$:

$$L_{C,1} = \{u_{i_1} \cdots u_{i_k} d c^{i_k} \cdots d c^{i_1} \mid 1 \leq i_j \leq n,\ j = 1, \ldots, k,\ k \geq 1\},$$

$$L_{C,2} = \{v_{i_1} \cdots v_{i_k} d c^{i_k} \cdots d c^{i_1} \mid 1 \leq i_j \leq n,\ j = 1, \ldots, k,\ k \geq 1\}.$$

Note that the PCP instance C has a solution if and only if $L_{C,1} \cap L_{C,2} \neq \emptyset$. We observe that although $L_{C,1} \cap L_{C,2}$ is not context-free in general, $\overline{L_{C,i}}$ is context-free, $i = 1, 2$, and hence so is $\overline{L_{C,1}} \cup \overline{L_{C,2}}$. Here $\overline{L}$ stands for complement with respect to Σ^*. Since $\overline{L_{C,1}} \cup \overline{L_{C,2}} = \overline{L_{C,1} \cap L_{C,2}}$, it is clear that $\overline{L_{C,1}} \cup \overline{L_{C,2}} = \Sigma^*$ exactly when the PCP instance C has no solution.

In other words, if C has no solution, then certainly Σ^* is an x-approximation, $x \in \{m, b, t\}$, for $\overline{L_{C,1}} \cup \overline{L_{C,2}}$, since the two languages coincide.

On the other hand, if C has a solution, then there exists $w \in L_{C,1} \cap L_{C,2}$. Note that if we write $w = w_1 w_2$ where $w_1 \in \{a,b\}^*$ and $w_2 \in \{c,d\}^*$, from the structure of the language $L_{C,1} \cap L_{C,2}$ it follows that $w_1^i w_2^i \in L_{C,1} \cap L_{C,2}$ for all $i \geq 1$. Thus always when $n = i \cdot |w|$, $i \geq 1$, $|\Sigma^* - (\overline{L_{C,1}} \cup \overline{L_{C,2}})|_n \geq 1$. We get

$$App(\Sigma^*, \overline{L_{C,1}} \cup \overline{L_{C,2}}) \geq \limsup_{i \to \infty} \frac{|\Sigma^* - (\overline{L_{C,1}} \cup \overline{L_{C,2}})|_{i \cdot |w|}}{min_1(|\overline{L_{C,1}} \cup \overline{L_{C,2}})|_{i \cdot |w|}, |\Sigma^* - (\overline{L_{C,1}} \cup \overline{L_{C,2}})|_{i \cdot |w|})}$$

and here the right side is at least one. It follows that the degree of approximation has to be at least one. ■

6 Conclusion

Here we have continued the work of [7,9,10] in attempting to classify different types of regular approximations for non-regular languages. Naturally much more remains to be done in relating these results to previous work. In particular, it would be interesting to explore possible connections with the rough set approximations studied in [12,13].

Our results leave open the question whether there exists a (context-free) language L such that L does not have any m-approximation but L has an m-approximation in the limit. We have shown that for b-approximations and t-approximations such examples do exist.

We have shown that it is undecidable whether a given regular language is an x-approximation, $x \in \{m, b, t\}$, for a given context-free language. It remains an open question whether for a given context-free language L we can decide whether or not L has some x-approximation. Note that at first sight the undecidability of this question could seem to follow from Greibach's theorem [8]. However, the result is not applicable because the class of context-free languages that have x-approximations is not preserved under quotient with single alphabet symbols, as observed in Lemma 4.8.

References

[1] C. Calude, K. Salomaa and S. Yu, Additive distances and quasi-distances between words. J. Universal Computer Science 8 (2002) 141–152.

[2] C. Câmpeanu and A. Păun, Tight bounds for the state complexity of deterministic cover automata. In: H. Leung and G. Pighizzini (Eds.), Descriptional Complexity of Formal Systems, DCFS 2006, Las Cruces, NM, June 21–23, 2006, pp. 223–231

[3] C. Câmpeanu, N. Sântean and S. Yu, Minimal cover automata for finite languages. Theoret. Comput. Sci. 267 (2001) 3–16.

[4] B. Cordy and K. Salomaa, Regular approximations of non-regular languages. In: H. Leung and G. Pighizzini (Eds.), Descriptional Complexity of Formal Systems, DCFS 2006, Las Cruces, NM, June 21–23, 2006, pp. 118–129.

[5] C. Cortes, M. Mohri, A. Rastogi and M. Riley, Efficient computation of the relative entropy of probabilistic automata. 7th Latin American Symposium, LATIN 2006, Lect. Notes Comput. Sci. 3887, Springer, 2006, pp. 323–336.

[6] M. Domaratzki, J. Shallit and S. Yu, Minimal covers of formal languages. Developments in Language Theory, DLT 2001, Lect. Notes Comput. Sci. 2295, Springer, 2001, pp. 319–329.

[7] G. Eisman and B. Ravikumar, Approximate recognition of non-regular languages by finite automata. Australasian Computer Science Conference, ASCS 2005, pp. 219–228.

[8] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company, 1979.

[9] M. Kappes and C. Kintala, Tradeoffs between reliability and conciness of deterministic finite automata. J. Automata, Languages and Combinatorics 9 (2004) 281–292.

[10] M. Kappes and F. Niessner, Succinct representations of DFA with different levels of reliability. Theoret. Comput. Sci. 330 (2005) 299–310.

[11] M.-J. Nederhof, Practical experiments with regular approximation of context-free languages. Computational Linguistics 26 (2000) 17–44.

[12] Gh. Păun, L. Polkowski and A. Skowron, Rough-set-like approximations of context-free and regular languages. Proceedings of IPMU–96: Information Processing and Management of Uncertainty on Knowledge Based Systems, July 1–5, 1996, Granada, Spain, Universidad de Granada, vol. II, pp. 891–895.

[13] Gh. Păun, L. Polkowski and A. Skowron, Rough set approximations of languages. Fundamenta Informaticae 32 (1997) 149–162.

[14] J. Shallit and Y. Breitbart, Automaticity I: Properties of a measure of descriptional complexity. J. Comput. System Sci. 53 (1996) 10–25.

[15] S. Yu, Regular languages. In: G. Rozenberg and A. Salomaa (Eds.), Handbook of Formal Languages, Vol. I, Springer, 1997, pp. 41–110.