Approximating Deterministic Lattice Automata

Shulamit Halamish* and Orna Kupferman

Hebrew University, School of Engineering and Computer Science, Jerusalem 91904, Israel
Email: {lamit,orna}@cs.huji.ac.il

Abstract. Traditional automata accept or reject their input, and are therefore Boolean. *Lattice automata* generalize the traditional setting and map words to values taken from a lattice. In particular, in a fully-ordered lattice, the elements are $0, 1, \ldots, n-1$, ordered by the standard $\leq$ order. Lattice automata, and in particular lattice automata defined with respect to fully-ordered lattices, have interesting theoretical properties as well as applications in formal methods. *Minimal deterministic automata* capture the combinatorial nature and complexity of a formal language. Deterministic automata have many applications in practice.

In [13], we studied minimization of deterministic lattice automata. We proved that the problem is in general NP-complete, yet can be solved in polynomial time in the case the lattices are fully-ordered. The multi-valued setting makes it possible to combine reasoning about lattice automata with *approximation*. An approximating automaton may map a word to a range of values that are close enough, under some pre-defined distance metric, to its exact value. We study the problem of finding minimal approximating deterministic lattice automata defined with respect to fully-ordered lattices. We consider approximation by absolute distance, where an exact value x can be mapped to values in the range $[x-t, x+t]$, for an approximation factor t, as well as approximation by separation, where values are mapped into t classes. We prove that in both cases the problem is in general NP-complete, but point to special cases that can be solved in polynomial time.

1 Introduction

Automata theory is one of the longest established areas in computer science. Applications of automata theory include pattern matching, syntax analysis, and formal verification. In recent years, novel applications of automata-theoretic concepts have emerged from numerous sciences, like biology, physics, cognitive sciences, control, and linguistics. These novel applications require significant advances in fundamental aspects of automata theory [23]. One such advance is a transition from a Boolean to a multi-valued setting: while traditional automata accept or reject their input, and are therefore Boolean, novel applications, for example in speech recognition and image processing [19], are based on weighted automata, which map an input word to a value from a semi-ring over a large domain [8].

Focusing on applications in formal verification, the multi-valued setting arises directly in *quantitative verification* [14], and indirectly in applications like *abstraction methods*, in which it is useful to allow the abstract system to have unknown assignments to atomic propositions and transitions [24], *query checking* [6], which can be reduced to model checking over multi-valued systems, and verification of systems from *inconsistent viewpoints* [15], in which the value of the atomic propositions is the composition of their values in the different viewpoints.

In the above examples, the semi-ring used in the automata is often a *finite distributive lattice*. A lattice $\langle A, \leq \rangle$ is a partially ordered set in which every two elements $a, b \in A$ have

* Supported in part by the Israeli Ministry of Science and Technology.

a least upper bound (a *join* b) and a greatest lower bound (a *meet* b). In particular, in a *fully-ordered* lattice (a.k.a. *linearly-* or *totally-ordered* lattice: one in which $a \leq b$ or $b \leq a$ for all elements a and b in the lattice), join and meet correspond to max and min, respectively.

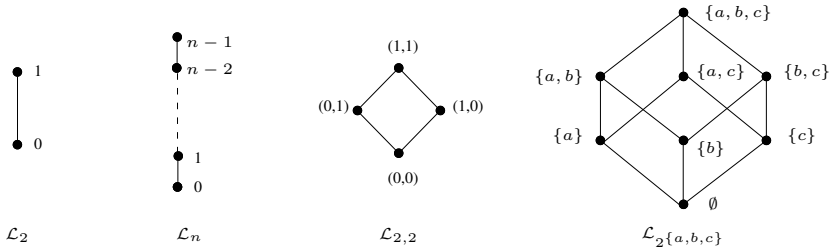

Fig. 1. Some lattices

For example (see Figure 1), in the abstraction application, researchers use the lattice $\mathcal{L}_3$ of three fully-ordered values [3, 24], as well as its generalization to $\mathcal{L}_n$ [7]. In query checking [6], the lattice elements are sets of formulas, ordered by the inclusion order, as in $\mathcal{L}_{2^{\{a,b,c\}}}$ [4]. When reasoning about inconsistent viewpoints, each viewpoint is Boolean, and their composition gives rise to products of the Boolean lattice, as in $\mathcal{L}_{2,2}$ [9, 15]. In addition, when specifying prioritized properties of systems, one uses lattices in order to specify the priorities [1]. Finally, LTL has been extended to *latticed LTL* (LLTL, for short), where the atomic propositions can take lattice values. The semantics of LLTL is defined with respect to multi-valued Kripke structures and it can specify their quantitative properties [7, 17].

In a *nondeterministic lattice automaton* on finite words (LNFA, for short) [17], each transition is associated with a *transition value*, which is a lattice element (intuitively indicating the truth of the statement "the transition exists"), and each state is associated with an *initial value* and an *acceptance value*, indicating the truth of the statements "the state is initial/accepting", respectively. Each run r of an LNFA $\mathcal{A}$ has a value, which is the *meet* of the values of all the components of r: the initial value of the first state, the transition value of all the transitions taken along r, and the acceptance value of the last state. The value of a word w is then the *join* of the values of all the runs of $\mathcal{A}$ on w. Accordingly, an LNFA $\mathcal{A}$ over an alphabet Σ and lattice $\mathcal{L}$ induces an $\mathcal{L}$*-language* $L(\mathcal{A}) : \Sigma^* \to \mathcal{L}$. Note that traditional finite automata (NFAs) can be viewed as a special case of LNFAs over the lattice $\mathcal{L}_2$. In a *deterministic* lattice automaton on finite words (LDFA, for short), at most one state has an initial value that is not $\bot$ (the least lattice element), and for every state q and letter σ, at most one state q' is such that the value of the transition from q on σ to q' is not $\bot$. Thus, an LDFA $\mathcal{A}$ has at most one run whose value is not $\bot$ on each input word, and the value of this run is the value of the word in the language of $\mathcal{A}$. In case such a run does not exist, the value of the word is $\bot$.

For example, the LDFA $\mathcal{A}$ in Figure 2 is over the alphabet $\Sigma = \{a, b, c, \#\}$ and the lattice $\mathcal{L} = \langle\{0, 1, 2, 3\}, \leq\rangle$. All states have acceptance value 3, and this is also the initial value of the single initial state. The $\mathcal{L}$-language of $\mathcal{A}$ is $L : \Sigma^* \to \mathcal{L}$ such that $L(\epsilon) = 3$, $L(a) = 3$, $L(a \cdot \#) = 1$, $L(b) = 1$, $L(b \cdot \#) = 1$, $L(c) = 3$, $L(c \cdot \#) = 2$, and $L(w) = 0$ for all other $w \in \Sigma^*$.

Minimal deterministic automata capture the combinatorial nature and complexity of formal languages. Beyond this theoretical importance, deterministic automata have many applications in practice. They are used in run-time monitoring, pattern recognition, and modeling systems.

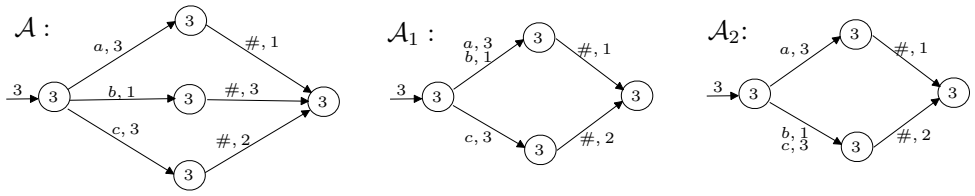

Fig. 2. An LDFA $\mathcal{A}$ over a fully ordered lattice, with two different minimal LDFAs.

Thus, the minimization problem for deterministic automata is of great interest, both theoretically and in practice. For deterministic traditional automata on finite words (DFAs, for short), a minimization algorithm, based on the Myhill-Nerode right congruence on the set of words, generates in polynomial time a canonical minimal deterministic automaton [20, 21]. A polynomial algorithm based on a right congruence is known also for deterministic weighted automata over the tropical semi-ring [19].

In [13], we studied the minimization problem for LDFAs. We showed that it is impossible to define a right congruence in the context of latticed languages, and that no canonical minimal LDFA exists. The difficulty is demonstrated in the LDFA $\mathcal{A}$ in Figure 2. It is not hard to see that both $\mathcal{A}_1$ and $\mathcal{A}_2$, which are not isomorphic, are minimal LDFAs equivalent to $\mathcal{A}$. The lack of a right congruence makes the minimization problem much more complicated than in the Boolean setting, and in fact also than in the setting of the tropical semi-ring. It is shown in [13] that the minimization problem for LDFAs is NP-complete in general. As good news, it is also shown in [13] that even though it is impossible to define a right congruence even when the LDFA is defined with respect to a fully-ordered lattice (indeed, the LDFA in Figure 2 is defined with respect to $\mathcal{L}_4$), it is possible to minimize such LDFAs in polynomial time.

The multi-valued setting makes it possible to combine reasoning about weighted and lattice automata with *approximation*. In this context, an approximating LDFA may map a word to a range of values that are close enough, under some pre-defined distance metric, to its exact value.

Approximations are widely used in computer science in cases where finding an exact solution is impossible or too complex. In the context of automata, approximation is used already in the Boolean setting: DFAs are used in order to approximate regular [25] and non-regular [10] languages. Applications of approximating DFAs include network security pattern matching, where one-sided errors are acceptable, and abstraction-refinement methods in formal methods, where DFAs that over- and under-approximate the concrete language of the system or the specification are used [18]. In the weighted setting, researchers used approximating automata in order to cope with the fact that not all weighted automata can be determinized [2, 5]. For example, [2] introduces t-determinization: given a weighted automaton $\mathcal{A}$ and an approximation factor $t > 1$, constructs a deterministic weighted automaton $\mathcal{A}'$ such that for all words $w \in \Sigma^*$, it holds that $L(A)(w) \leq L(A')(w) \leq t \cdot L(A)(w)$. Approximation not only makes determinization possible but also leads to automata that are significantly smaller [2, 5].

In this paper we study the approximation of lattice automata and the problem of finding minimal approximating LDFAs. We focus on LDFAs defined with respect to fully-ordered lattices. While it is possible to define distance metrics also in the setting of partially-ordered lattices, for example by using lattice chains, we find the notion of approximation cleaner in the setting of fully-ordered lattices. Also, since the minimization problem for LDFAs over partially ordered lattices is NP-hard already without approximations, we cannot expect the problem of

finding a minimal approximating LDFA to be easier. Finally, as we discuss below, applications of approximated minimization exist already for fully-ordered lattices.

In fully-ordered lattices, there is a natural way to define the distance between two lattice elements: finite fully-ordered lattices are all isomorphic to $\mathcal{L}_n$, for some $n \geq 1$, and we define the distance between two elements a and b in the lattice to be $|a - b|$.[1] We refer to approximations with respect to this metric as *distance approximations*: Consider an integer $n \geq 1$, an approximation factor $0 \leq t \leq n-1$, and two LDFAs $\mathcal{A}$ and $\mathcal{A}'$ over $\mathcal{L}_n$. We say that $\mathcal{A}'$ *t-approximates* $\mathcal{A}$ iff for all words $w \in \Sigma^*$ we have that $|L(\mathcal{A}')(w) - L(\mathcal{A})(w)| \leq t$. That is, $\mathcal{A}'$ t-approximates $\mathcal{A}$ if it maps every word w to a value whose distance from $L(\mathcal{A})(w)$ is at most t. We first show that the problem of deciding whether $\mathcal{A}'$ t-approximates $\mathcal{A}$ is NLOGSPACE-complete. We then turn to the problem of finding a minimal LDFA that t-approximates a given LDFA. We study the corresponding decision problem, and then conclude about the search problem. It follows from [13] that the special case of finding a minimal 0-approximating LDFA can be solved in polynomial time. We show that when $1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$, the problem is NP-complete, and in fact is even inapproximable. On the other hand, for $\lfloor \frac{n}{2} \rfloor \leq t \leq n-1$, the problem can be solved in constant time (simply since a t-approximating LDFA of size one always exists).

A different natural way to define approximations in a fully-ordered lattice involves the introduction of *separators*. Formally, a *t-separation* of $\mathcal{L}_n$, for $1 \leq t \leq n$, is a partition $\mathcal{P} = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{t-1}\}$ of $\{0, 1, \ldots, n-1\}$ such that all the sets in the partition are not empty and contain only successive elements. An approximation by t-separation maps a set of successive values to a single value. Formally, consider an integer $n \geq 1$, an approxomation factor $1 \leq t \leq n$, an LDFA $\mathcal{A}$ over $\mathcal{L}_n$ and an LDFA $\mathcal{A}'$ over $\mathcal{L}_t$. We say that $\mathcal{A}'$ *t-separates* $\mathcal{A}$ iff there is a t-separation $\mathcal{P}$ of $\mathcal{L}_n$ such that for all words $w \in \Sigma^*$ we have that $L(\mathcal{A}')(w) = i$ iff $L(\mathcal{A})(w) \in \mathcal{P}_i$. For example, when $n = 10$ and $\mathcal{P} = \{\{0\}, \{1\}, \{2, \ldots, 9\}\}$, the LDFA $\mathcal{A}'$ may agree with $\mathcal{A}$ on all words that are mapped to 0 and 1, and map to 2 all words that are mapped by $\mathcal{A}$ to $\{2, \ldots, 9\}$. We first show that the problem of deciding whether $\mathcal{A}'$ t-separates $\mathcal{A}$ according to a given t-separation is NLOGSPACE-complete. We then turn to the problem of finding a minimal LDFA that t-separates a given LDFA. We show that the problem can be solved in polynomial time for a fixed t, and is NP-complete when t is a parameter to the problem.

Beyond the theoretical motivation for studying minimization of approximating LDFAs with respect to the two distance metrics, both metrics are useful in practice. Recall the use of fully-ordered lattices in the specification of prioritized properties of systems [1]. Consider an LDFA over $\mathcal{L}_n$ that corresponds to the specification. Assume that the (Boolean) language of all words that are assigned some value i has a large Myhill-Nerod index, thus a DFA for it, and hence also the LDFA, is big. It is often possible to approximate the assignment of priorities so that "problematic" values are no longer problematic and an LDFA for the specification is much smaller. Using the distance metric, the approximation may map a value x to values in $[x - t, x + t]$. Such a metric is useful when we tolerate a change of order between the prioritized properties but want to limit the range in which priorities are changed. Using separators, the approximation is not restricted to a certain range, but bounds the number of classes to which successive priorities can be mapped. Such a metric is useful when we care about the order of the prioritized properties and do not care about their values. As another example, recall the application of the three-value lattice $\mathcal{L}_3$ in abstraction. A 2-separation of $\mathcal{L}_3$ corresponds to either an under-approximation of the concrete system, in case the "unknown" value is grouped

[1] Another popular isomorphic lattice uses the range $0, \ldots, 1$, with the elements being $0, \frac{1}{n}, \frac{2}{n}, \ldots, 1$. We find it simpler to work with $\mathcal{L}_n$. Clearly, our results can be easily adjusted to all fully-order lattices.

with "false", or to an over-approximation, in case "unknown" value is grouped with "true". Finally, both types of approximation may be useful when using LDFAs for the specification of quantitative properties.

2 Fully-Ordered Lattices and Lattice Automata

Let $\langle A, \leq \rangle$ be a partially ordered set, and let P be a subset of A. An element $a \in A$ is an *upper bound* on P if $a \geq b$ for all $b \in P$. Dually, a is a *lower bound* on P if $a \leq b$ for all $b \in P$. An element $a \in A$ is the *least element of* P if $a \in P$ and a is a lower bound on P. Dually, $a \in A$ is the *greatest element of* P if $a \in P$ and a is an upper bound on P. A partially ordered set $\langle A, \leq \rangle$ is a *lattice* if for every two elements $a, b \in A$ both the least upper bound and the greatest lower bound of $\{a, b\}$ exist, in which case they are denoted $a \vee b$ (a *join* b) and $a \wedge b$ (a *meet* b), respectively. For an integer $n \geq 1$, let $[n] = \{0, 1, \ldots, n-1\}$. The *fully-ordered* lattice (a.k.a. *linearly-* or *totally-ordered* lattice) with n elements is $\mathcal{L}_n = \langle [n], \leq \rangle$, where $\leq$ is the usual "less than" relation, thus $0 \leq 1 \leq \cdots \leq n-1$. For a set $P \subseteq [n]$ of elements, the least upper bound of P, namely the join of the elements in P, is $\max P$. The greatest lower bound of P, namely the meet of the elements in P, is $\min P$. In particular, the meet and join of $[n]$ are 0 and $n-1$, also referred to as $\bot$ and $\top$, respectively.

Consider a lattice $\mathcal{L} = \langle A, \leq \rangle$. For a set $\mathcal{X}$ of elements, an *$\mathcal{L}$-set over $\mathcal{X}$* is a function $S : \mathcal{X} \to A$ assigning to each element of $\mathcal{X}$ a value in A. It is convenient to think about $S(x)$ as the truth value of the statement "x is in S". We say that an $\mathcal{L}$-set S is *Boolean* if $S(x) \in \{\top, \bot\}$ for all $x \in \mathcal{X}$. In particular, all $\mathcal{L}_2$-sets are Boolean.

Consider a lattice $\mathcal{L} = \langle A, \leq \rangle$ and an alphabet Σ. An *$\mathcal{L}$-language* is an $\mathcal{L}$-set over Σ^*. Thus, an $\mathcal{L}$-language $L : \Sigma^* \to A$ assigns a value in A to each word over Σ.

A *deterministic lattice automaton* on finite words (LDFA) is $\mathcal{A} = \langle \mathcal{L}, \Sigma, Q, Q_0, \delta, F \rangle$, where $\mathcal{L}$ is a finitie lattice, Σ is a finite alphabet, Q is a finite set of states, $Q_0 \in \mathcal{L}^Q$ is an $\mathcal{L}$-set of initial states, $\delta \in \mathcal{L}^{Q \times \Sigma \times Q}$ is an $\mathcal{L}$-transition-relation, and $F \in \mathcal{L}^Q$ is an $\mathcal{L}$-set of accepting states. The fact that $\mathcal{A}$ is deterministic is reflected in two conditions on Q_0 and δ. First, there is at most one state $q \in Q$, called the *initial state* of $\mathcal{A}$, such that $Q_0(q) \neq \bot$. In addition, for every state $q \in Q$ and letter $\sigma \in \Sigma$, there is at most one state $q' \in Q$, called the *σ-destination* of q, such that $\delta(q, \sigma, q') \neq \bot$. The *run* of an LDFA on a word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_l$ is a sequence $r = q_0, \ldots, q_l$ of $l+1$ states, where q_0 is the initial state of $\mathcal{A}$, and for all $1 \leq i \leq l$ it holds that q_i is the *σ_i-destination* of q_{i-1}. The *value* of w is $val(w) = Q_0(q_0) \wedge \bigwedge_{i=1}^{l} \delta(q_{i-1}, \sigma_i, q_i) \wedge F(q_l)$. Intuitively, $Q_0(q_0)$ is the value of q_0 being initial, $\delta((q_{i-1}, \sigma_i, q_i))$ is the value of q_i being a successor of q_{i-1} when σ_i is the input letter, $F(q_l)$ is the value of q_l being accepting, and the value of w is the meet of all these values. The $\mathcal{L}$-language of $\mathcal{A}$, denoted $L(\mathcal{A})$, maps each word w to the value of its run in $\mathcal{A}$. In case $\mathcal{A}$ does not have a run on w, this word is mapped to $\bot$. In the special case of a lattice automaton over $\mathcal{L}_n$, the value of a word is simply the minimal value appearing in its run. An LDFA is *simple* if Q_0 and δ are Boolean. An example of an LDFA over a fully-ordered lattice can be found in Figure 2.

Note that traditional deterministic automata over finite words (DFA) correspond to LDFA over the lattice $\mathcal{L}_2$. Indeed, over $\mathcal{L}_2$, a word is mapped by $L(\mathcal{A})$ to the value $\top$ iff the run on it uses only transitions with value $\top$ and its final state has acceptance value $\top$.

Analyzing the size of $\mathcal{A}$, one can refer to $|\mathcal{L}|$, $|Q|$, and $|\delta|$. Since the emphasize in this paper is on the size of the state space, we use $|\mathcal{A}|$ to refer to the size of its state space.

3 Approximation by Distance

In this section we study the problem of approximating LDFAs over fully-ordered lattices using the natural distance metric, in which the distance between two elements a and b is $|a - b|$. We first formally define approximation with respect to this metric.

Definition 1. *Consider an integer $n \geq 1$, an approximation factor $0 \leq t \leq n-1$, and two LDFAs $\mathcal{A}$ and $\mathcal{A}'$ over $\mathcal{L}_n$. We say that $\mathcal{A}'$ t-*approximates *$\mathcal{A}$ iff for all words $w \in \Sigma^*$ we have that $|L(\mathcal{A}')(w) - L(\mathcal{A})(w)| \leq t$.*

Intuitively, the distance between the exact and the approximated values of words is at most t. It is not hard to see that the quality of the approximation improves as t becomes smaller. For $t = n - 1$ (in fact, already for $t \geq \lfloor \frac{n}{2} \rfloor$) all words can simply be mapped to the same element. As t reduces, the values of the words get closer to their exact values, till they coincide for $t = 0$, where $L(\mathcal{A}') = L(\mathcal{A})$.

Example 1. Figure 3 depicts an LDFA $\mathcal{A}$ over the lattice $\mathcal{L}_7$ and the alphabet $\Sigma = \{a_1, \ldots, a_6, \#\}$. It also depicts an LDFA $\mathcal{A}'$ that 1-approximates $\mathcal{A}$. One can see that $\mathcal{A}'$ agrees with $\mathcal{A}$ on the values of the words $\epsilon, a_1, \ldots, a_6, a_2\#, a_5\#$, while the values of the words $a_1\#, a_3\#, a_4\#, a_6\#$ differs by 1. All other words are mapped by both $\mathcal{A}$ and $\mathcal{A}'$ to 0. The approximation enables $\mathcal{A}'$ to merge the upper and lower three states that are reachable in one transition in $\mathcal{A}$.

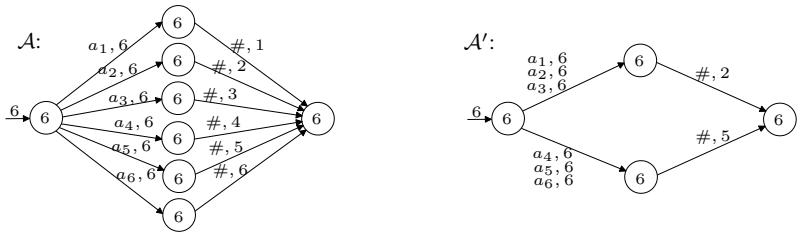

Fig. 3. An LDFA $\mathcal{A}$ with a 1-approximating LDFA $\mathcal{A}'$.

Theorem 1. *Let $n \geq 1$ and $t \geq 0$. Given two LDFAs $\mathcal{A}$ and $\mathcal{A}'$ over $\mathcal{L}_n$, deciding whether $\mathcal{A}'$ t-approximates $\mathcal{A}$ is NLOGSPACE-complete.*

Proof. We start with the upper bound and rely on the fact that co-NLOGSPACE = NLOGSPACE [16]. To decide whether $\mathcal{A}'$ does not t-approximate $\mathcal{A}$, it is enough to find a word $w \in \Sigma^*$ such that $|L(A')(w) - L(A)(w)| > t$. We show that if such a word exists, then there also exists a word of size at most $n^2|\mathcal{A}'||\mathcal{A}|$ satisfying this condition[2]. Let $w \in \Sigma^*$ be such a word. Consider two corresponding simple LDFAs $\mathcal{S}$ and $\mathcal{S}'$, such that $L(\mathcal{S}) = L(\mathcal{A})$ and $L(\mathcal{S}') = L(\mathcal{A}')$. In [17] it was shown that there are such automata with $n|\mathcal{A}|$ and $n|\mathcal{A}'|$ states, respectively. Consider now the product automaton of $\mathcal{S}$ and $\mathcal{S}'$ which is also simple and is of size $n^2|\mathcal{A}'||\mathcal{A}|$. Let q_w and $q_{w'}$ be the states reached by w in $\mathcal{S}$ and $\mathcal{S}'$, respectively. Note that the run of the product automaton on w reaches the state $\langle q_w, q_{w'} \rangle$. Since the product automaton is simple, the value of w is induced only by the acceptance values of the states q_w and $q_{w'}$. Therefore,

[2] In fact, one can prove that there exists such a word of size at most $|\mathcal{A}'||\mathcal{A}|$, but this is not required for our purpose.

we can ignore all cycles along the run if any, and get a word of size at most $n^2|\mathcal{A}'||\mathcal{A}|$ which gets the same value as w on both $\mathcal{A}$ and $\mathcal{A}'$. Therefore, one can guess a word of size at most $n^2|\mathcal{A}'||\mathcal{A}|$, compute its value on both $\mathcal{A}$ and $\mathcal{A}'$ in logarithmic space, and return that $\mathcal{A}'$ does not t-approximate $\mathcal{A}$ iff $|L(A')(w) - L(A)(w)| > t$. Finally, the lower bound is by an easy reduction from the reachability problem in directed graphs. □

Approximation can lead to a significant reduction in the size of the automata. Formally, for all $t \geq 1$ there is a family of LDFAs for which t-approximation allows for an exponential reduction in the state space whereas $(t-1)$-approximation allows no reduction. To see this, note that applying 1-approximation on languages over $\mathcal{L}_2$ results in an LDFA with one state, no matter how complicated the original automaton was. This trivial example can be naturally extended to all $t \geq 1$ by considering lattices over $[n]$ and $\mathcal{L}_n$-languages that map the accepted and non-accepted words to two lattice values l and l', respectively, such that $|l - l'| = t$.

Motivated by the importance of generating small automata, our goal is to find an LDFA $\mathcal{A}'$ with a minimal number of states that t-approximates a given LDFA $\mathcal{A}$. We show that the problem is trivial when t is large enough. For the non-trivial interesting case where t is small with respect to the lattice, the problem becomes much more complicated, and we prove that it is NP-complete. However, for $t = 0$, where the problem coincides with minimization, it gets back to the easy side and can be solved in polynomial time [13].

Consider the corresponding decision problem: APRXLDFA=$\{\langle \mathcal{A}, n, t, k\rangle : \mathcal{A}$ is an LDFA over $\mathcal{L}_n$ with a t-approximating $\mathcal{A}'$ over $\mathcal{L}_n$ such that $|\mathcal{A}'| \leq k\}$. As we shall see, the complexity of APRXLDFA depends on the relation between n and t. We therefore study also the family of problems (n,t)-APRXLDFA, in which n and t are not parameters and rather are fixed. That is, (n,t)-APRXLDFA=$\{\langle \mathcal{A}, k\rangle : \mathcal{A}$ is an LDFA over $\mathcal{L}_n$ with a t-approximating $\mathcal{A}'$ over $\mathcal{L}_n$ such that $|\mathcal{A}'| \leq k\}$.

Theorem 2. *Let $n \geq 1$. The problem (n,t)-APRXLDFA:*

- *[13] Can be solved in polynomial time for $t = 0$.*
- *Is NP-complete for $1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$.*
- *Can be solved in constant time for $t \geq \lfloor \frac{n}{2} \rfloor$.*

Proof. We start with the second case. For the upper bound, given $\mathcal{A}$ and k, a witness for their membership in (n,t)-APRXLDFA is an LDFA $\mathcal{A}'$ as required. Assuming $k \leq |\mathcal{A}|$ (otherwise, $\langle A, k\rangle$ clearly belongs to (n,t)-APRXLDFA), the size of $\mathcal{A}'$ is linear in the input. By Theorem 1, we can verify that $\mathcal{A}'$ t-approximates $\mathcal{A}$ in polynomial time.

For the lower bound, we show a polynomial-time reduction from the Minimal Automaton Identification problem (MAI, for short), proved to be NP-complete in [12]. The MAI problem refers to the minimal DFA whose language agrees with a set of observations.[3] Formally, let Σ be an alphabet of size two. A *data* is a set $D = \{(w_1, y_1), \ldots, (w_n, y_n)\}$, where for all $1 \leq i \leq n$, we have that $w_i \in \Sigma^*$ and $y_i \in \{0,1\}$, and $w_i \neq w_j$ for all $1 \leq i \neq j \leq n$. A DFA $\mathcal{A}$ *agrees* with D iff $L(\mathcal{A})(w_i) = y_i$ for all $1 \leq i \leq n$. Then, MAI $= \{\langle D, k\rangle : D$ is data, and there exists a DFA $\mathcal{A}$ with $|\mathcal{A}| \leq k$ that agrees with $D\}$.

We now turn to describe the reduction, starting with the case n is even and $t = \frac{n}{2} - 1$. Note that for t to be at least 1, it must be that $n \geq 4$. Given an input $\langle D, k\rangle$ for MAI, let Σ be the alphabet and let $D = \{(w_1, y_1), \ldots, (w_m, y_m)\}$. We construct an LDFA $\mathcal{A} = \langle \mathcal{L}_n, Q, \Sigma, \delta, Q_0, F\rangle$ as described below.

[3] The result in [12] is stated by means of Mealy machines. It can, however, be easily adapted to DFAs.

– The states Q are defined as follows. Let P be the set of all prefixes of the words $w_1, \ldots, w_m$. Each prefix $w \in P$ induces a state q_w. Note that prefixes that are common to more than one word contribute one element to P, and therefore induce one state in Q. In addition, there is a state q_{sink}.
– For all $w \in P$ and $\sigma \in \Sigma$, if $w\sigma \in P$, then $\delta(q_w, \sigma, q_{w\sigma}) = \top$; Otherwise, $\delta(q_w, \sigma, q_{sink}) = \top$. In addition, $\delta(q_{sink}, \sigma, q_{sink}) = \top$ for all $\sigma \in \Sigma$.
– $Q_0(q_\epsilon) = \top$, and $Q_0(q) = \bot$ for all other states $q \in Q$.
– The acceptance values are defined as follows. Consider first the states $\{q_{w_1}, \ldots, q_{w_m}\}$ corresponding to the words appearing in D. For all $1 \leq i \leq m$, if $y_i = 1$, then $F(q_{w_i}) = \top$; Otherwise, $F(q_{w_i}) = \bot$. For all other $q \in Q$, we define $F(q) = \frac{n}{2}$.

Note that $\mathcal{A}$ is deterministic. Also, since the components of $\mathcal{A}$ are all of size polynomial in the data D, the reduction is polynomial.

Example 2. Let $n = 4$. Figure 4 depicts the LDFA $\mathcal{A}_D$ corresponding to the data $D = \{(aa, 1), (aab, 0), (ab, 0), (bba, 1), (bbb, 0)\}$ over $\Sigma = \{a, b\}$. All transitions described in the figure have the value $\top$. The states with acceptance value $\top$ correspond to words $w \in \{a, b\}^*$ such that $(w, 1) \in D$, and symmetrically, the states with acceptance value $\bot$ correspond to words $w \in \{a, b\}^*$ such that $(w, 0) \in D$. The states with acceptance value of 2 on the left correspond to the strict prefixes of the words in D. Finally, the rightmost state is q_{sink}, and its acceptance value is also 2.

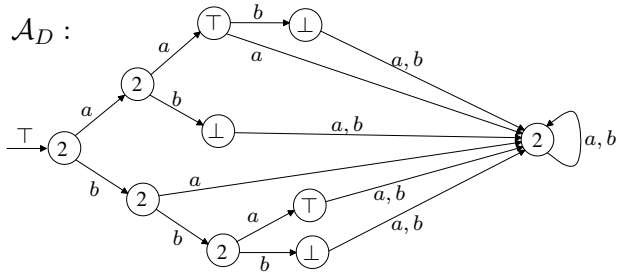

Fig. 4. The LDFA $\mathcal{A}_D$ induced by the data D.

Intuitively, the goal of $\mathcal{A}$ is to keep the information stored in D in a way that would enable to restore it from an LDFA that t-approximates $\mathcal{A}$. By mapping to 0 and $n-1$ and defining t to be strictly smaller than $\frac{n}{2}$, we ensure that the t-approximating LDFAs do not mix up words that are mapped in D to different values. This way, we can associate a t-approximation $\mathcal{A}'$ of $\mathcal{A}$ with a DFA $\mathcal{B}$ that agrees with D and vice versa.

We prove formally that $\langle D, k\rangle \in$ MAI iff $\langle A, k\rangle \in (n, t)$-APRXLDFA. That is, there exists a DFA $\mathcal{B}$ with $|\mathcal{B}| \leq k$ that agrees with D iff there exists a t-approximating $\mathcal{A}'$ for $\mathcal{A}$ with $|\mathcal{A}'| \leq k$.

Assume first that there exists a DFA $\mathcal{B}$ as required. Recall that a DFA can be viewed as an LDFA over the Boolean lattice $\mathcal{L}_2 = \{0, 1\}$. We thus derive $\mathcal{A}'$ from $\mathcal{B}$ by viewing $\mathcal{B}$ as an LDFA over $\mathcal{L}_2$, and replacing all $0'$s and $1'$s appearing as values in $\mathcal{A}'$ with $\frac{n}{2} - 1$ and $\frac{n}{2}$, respectively. Note that $|\mathcal{A}'| \leq k$, as we do not add states to $\mathcal{B}$.

We show that $\mathcal{A}'$ t-approximates $\mathcal{A}$. It is not hard to see that $L(\mathcal{A})$ and $L(\mathcal{A}')$ are as follows:

$$L(\mathcal{A})(w) = \begin{cases} \bot & w = w_i \text{ for some } 1 \leq i \leq m \text{ and } y_i = 0 \\ \top & w = w_i \text{ for some } 1 \leq i \leq m \text{ and } y_i = 1 \\ \frac{n}{2} & \text{otherwise} \end{cases}$$

$$L(\mathcal{A}')(w) = \begin{cases} \frac{n}{2} - 1 & w = w_i \text{ for some } 1 \leq i \leq m \text{ and } y_i = 0 \\ \frac{n}{2} & w = w_i \text{ for some } 1 \leq i \leq m \text{ and } y_i = 1 \\ \frac{n}{2} \quad or \quad \frac{n}{2} - 1 & \text{otherwise} \end{cases}$$

We show that for all $w \in \Sigma^*$ it holds that $|L(A')(w) - L(A)(w)| \leq t$. For words $w \in \Sigma^*$ such that $w = w_i$ for some $1 \leq i \leq m$ and $y_i = 0$, it holds that $|L(A')(w) - L(A)(w)| = |(\frac{n}{2} - 1) - 0| = \frac{n}{2} - 1 = t$. For words $w \in \Sigma^*$ such that $w = w_i$ for some $1 \leq i \leq m$ and $y_i = 1$, it holds that $|L(A')(w) - L(A)(w)| = |\frac{n}{2} - (n-1)| = (n-1) - \frac{n}{2} = \frac{n}{2} - 1 = t$. Finally, for words that are not of the form w_i for any $1 \leq i \leq m$ the apprixomation is obvious, since $t \geq 1$. We conclude that $\mathcal{A}'$ t-approximates $\mathcal{A}$.

As for the other direction, assume now that there exists a t-approximating $\mathcal{A}'$ for $\mathcal{A}$ with at most k states. We show that there is a DFA $\mathcal{B}$ that agrees with D and $|\mathcal{B}| \leq k$.

We derive $\mathcal{B}$ from $\mathcal{A}'$ as follows. We replace by 0 all values between 0 and $\frac{n}{2} - 1$ appearing in $\mathcal{A}'$, and symmetrically, we replace by 1 all values between $n-1$ and $\frac{n}{2}$. Now we have an LDFA over $\mathcal{L}_2$. Such an LDFA can be viewed as a DFA, and we define $\mathcal{B}$ to be this DFA. Note that $|\mathcal{B}| \leq k$, as we do not add states to $\mathcal{A}'$.

It is left to show that $\mathcal{B}$ agrees with D. For this purpose we show that $L(\mathcal{B})(w_i) = y_i$ for all $1 \leq i \leq m$. Consider a word w_i for $1 \leq i \leq m$. Assume first that $y_i = 1$. Recall that in such a case it holds that $L(\mathcal{A})(w_i) = \top = n - 1$. Also, since $\mathcal{A}'$ t-approximates A, it holds that $|L(\mathcal{A}')(w_i) - (n-1)| \leq t = \frac{n}{2} - 1$, and therefore $L(\mathcal{A}')(w_i)$ must be between $n-1$ and $\frac{n}{2}$. This implies that all values read along the run of $\mathcal{A}'$ on w_i are between $n-1$ and $\frac{n}{2}$. Finally, recall that such values have all been replaced by $\frac{n}{2}$ and then by 1, meaning that $L(\mathcal{B})(w_i) = 1$ as required. The proof for w_i with $y_i = 0$ is similar. We conclude that $\mathcal{B}$ agrees with D, and we are done.

Now, recall that the reduction above assumes that n is even and $t = \frac{n}{2} - 1$. Hence, we still have to show that (n,t)-APRXLDFA is NP-hard for all $n \geq 1$ and $1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$.

Let $n \geq 1$ and $1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$. Since $2t+2$ is even and $t = \frac{2t+2}{2} - 1$, the reduction above shows that $(2t+2,t)$-APRXLDFA is NP-hard. We show a polynomial-time reduction from $(2t+2,t)$-APRXLDFA to (n,t)-APRXLDFA. Let $\langle \mathcal{A}, k \rangle$ be an input for $(2t+2,t)$-APRXLDFA. Since $2t+2 \leq 2(\lfloor \frac{n}{2} \rfloor - 1) + 2 = 2\lfloor \frac{n}{2} \rfloor \leq n$, we have that $2t+2 \leq n$. We extend the lattice $\mathcal{L}_{2t+2}$ to $\mathcal{L}_n$ by adding $n - (2t+2)$ new elements at the top of the lattice. We then produce $\langle \mathcal{U}, k \rangle$ as an input for (n,t)-APRXLDFA, where $\mathcal{U}$ coincides with $\mathcal{A}$, except that $\mathcal{U}$ is defined over $\mathcal{L}_n$ rather than $\mathcal{L}_{2t+2}$. It is not hard to see that $\langle \mathcal{A}, k \rangle \in (2t+2,t)$-APRXLDFA iff $\langle \mathcal{U}, k \rangle \in (n,t)$-APRXLDFA. Indeed, if $\mathcal{A}'$ t-approximates $\mathcal{A}$ when both are over $\mathcal{L}_{2t+2}$, then, as $L(\mathcal{A}) = L(\mathcal{U})$, we have that $\mathcal{A}'$ also t-approximates $\mathcal{U}$ when both are over $\mathcal{L}_n$. On the other hand, if an LDFA $\mathcal{U}'$ t-approximates $\mathcal{U}$, then replacing all occurrences of the new elements of $\mathcal{L}_n$ that appear in $\mathcal{U}'$ by $2t+1$ resulted in an LDFA over $\mathcal{L}_{2t+2}$ that still t-approximates $\mathcal{U}$, and therefore t-approximates $\mathcal{A}$ as well.

We now turn to prove the third part of the theorem. Let $t \geq \lfloor \frac{n}{2} \rfloor$. A constant time algorithm that is given $\langle \mathcal{A}, k \rangle$ and decides whether $\langle \mathcal{A}, k \rangle \in (n,t)$-APRXLDFA can simply return "yes" for all inputs. To prove correctness, we show that every LDFA $\mathcal{A}$ over $\mathcal{L}_n$ has a t-approximating $\mathcal{A}'$ of size one. To see this, consider an LDFA with one state that maps all words to the value $\lfloor \frac{n}{2} \rfloor$. Clearly, there exists such LDFA. It is left to show that $|\lfloor \frac{n}{2} \rfloor - L(A)(w)| \leq t$ for all $w \in \Sigma^*$. Since $0 \leq L(\mathcal{A})(w) \leq n-1$ for all $w \in \Sigma^*$, it is enough to show that $|\lfloor \frac{n}{2} \rfloor - 0| \leq t$

and $|\lfloor \frac{n}{2} \rfloor - (n-1)| \leq t$. The first inequality is clear, since $\lfloor \frac{n}{2} \rfloor \leq t$. For the second one $|\lfloor \frac{n}{2} \rfloor - (n-1)| = (n-1) - \lfloor \frac{n}{2} \rfloor \leq (n-1) - (\frac{n}{2} - \frac{1}{2}) = \frac{n}{2} - \frac{1}{2} \leq \lfloor \frac{n}{2} \rfloor \leq t$, and we are done. □

Going back to the problem APRXLDFA, we can now reduce (n,t)-APRXLDFA to APRXLDFA for some fixed $n \geq 1$ and $1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$. By the second part of Theorem 2, we conclude with the following.

Theorem 3. *APRXLDFA is NP-complete.*

Remark 1. By Theorem 3, it is unlikely that there can ever be an efficient exact algorithm for APRXLDFA. A natural question to ask is whether the problem can be efficiently approximated. In [22], the authors show that the MAI problem, from which we have reduced, cannot be approximated within any polynomial. That is, assuming P$\neq$ NP, there does not exist a polynomial time algorithm that on a data input D can always return a DFA of size at most polynomially larger than opt, where opt is the smallest DFA that agrees with D. Therefore, the reduction described in the proof of Theorem 2 in fact gives a stronger result: APRXLDFA is inapproximable.

Theorems 2 and 3 study the complexity of deciding whether there is a t-approximating LDFA of size at most k. Below we discuss the corresponding search problem, of constructing a minimal LDFA. Consider the three cases stated in Theorem 2. For $t = 0$, the approximation problem coincides with minimization, and there is an algorithm generating a minimal LDFA in polynomial time [13]. For $1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$, we can first perform a binary search to find the minimal k, and then verify a guessed LDFA of size k. Therefore, the problem of constructing a minimal LDFA belongs to the class FNP (of problems where the goal is to return a witness in an NP decision problem). Finally, for $\lfloor \frac{n}{2} \rfloor \leq t$, we have seen that a t-approximating LDFA can map all words to the value $\lfloor \frac{n}{2} \rfloor$, and a minimal one can do it with a single state.

4 Approximation by Separation

Approximation by distance poses a uniform requirement on the elements of the lattice. In this section we introduce and study another natural metric, based on lattice separation.

Definition 2. *Let $n \geq 1$ and $1 \leq t \leq n$. A t-separation of $\mathcal{L}_n$ is a partition $\mathcal{P} = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{t-1}\}$ of $[n]$ into t non-empty sets such that each set contains only successive elements.*

Definition 3. *Consider an integer $n \geq 1$, an approximation factor $1 \leq t \leq n$, an LDFA $\mathcal{A}$ over $\mathcal{L}_n$ and an LDFA $\mathcal{A}'$ over $\mathcal{L}_t$. We say that $\mathcal{A}'$ t-separates $\mathcal{A}$ iff there is a t-separation $\mathcal{P}$ of $\mathcal{L}_n$ such that for all words $w \in \Sigma^*$ we have that $L(\mathcal{A}')(w) = i$ iff $L(\mathcal{A})(w) \in \mathcal{P}_i$.*

Intuitively, an approximation by t-separation maps a set of successive values to a single value. One can see that the quality of the approximation improves as t grows. Indeed, for $t = 1$ we have only one set containing all elements, allowing us to map all words to the same value. On the other hand, when $t = n$, each element constitutes a singleton set, and $L(\mathcal{A}') = L(\mathcal{A})$.

Example 3. Figure 5 depicts an LDFA $\mathcal{A}$ over the lattice $\mathcal{L}_7$ and the alphabet $\Sigma = \{a_1, \ldots, a_6, \#\}$. The LDFA $\mathcal{A}'$ over $\mathcal{L}_3$ 3-separates $\mathcal{A}$ with respect to the 3-separation $\mathcal{P}_0 = \{0,1,2,3,4\}$, $\mathcal{P}_1 = \{5\}$, and $\mathcal{P}_2 = \{6\}$. One can see that the words $\epsilon, a_1, \ldots, a_6$, and $a_6\#$ are mapped by $\mathcal{A}$ to 6 and by $\mathcal{A}'$ to 2, and that the word $a_5\#$ is mapped by $\mathcal{A}$ to 5 and by $\mathcal{A}'$ to 1. All other words are mapped by $\mathcal{A}$ to values taken from the set $\{0,1,2,3,4\}$, and by $\mathcal{A}'$ to 0.

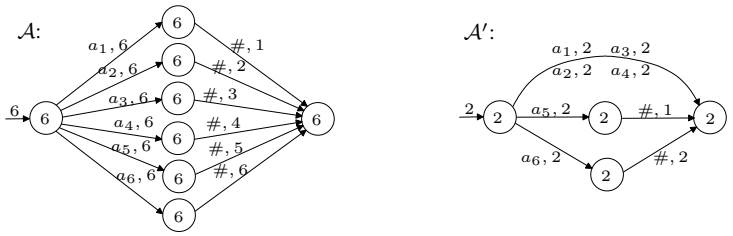

Fig. 5. An LDFA $\mathcal{A}$ with a 3-separating LDFA $\mathcal{A}'$.

Theorem 4. *Let $n \geq 1$ and $1 \leq t \leq n$. Given an LDFA $\mathcal{A}$ over $\mathcal{L}_n$, an LDFA $\mathcal{A}'$ over $\mathcal{L}_t$, and a t-separation $\mathcal{P} = \{\mathcal{P}_0, \ldots, \mathcal{P}_{t-1}\}$ of $\mathcal{L}_n$, deciding whether $\mathcal{A}'$ t-separates $\mathcal{A}$ with respect to $\mathcal{P}$ is NLOGSPACE-complete.*

Proof. We use the same considerations used in Theorem 1. Here we are looking for a word $w \in \Sigma^*$ such that $L(A)(w) \in \mathcal{P}_i$ but $L(A)(w) \neq i$ for some $0 \leq i \leq t-1$. As there, we can bound the length of such a word from above by $n^2|\mathcal{A}||\mathcal{A}'|$ and conclude that the problem can be solved in NLOGSPACE. Hardness in NLOGSPACE follows from hardness of reachability in directed graphs. □

We now turn to consider the problem of finding a minimal t-separating LDFA. As we have seen in Section 1, there are practical situations in which the input includes a specific t-separation of $\mathcal{L}_n$, and the goal is to find a minimal $\mathcal{A}'$ that t-separates $\mathcal{A}$ with respect to that separation. We show below that in such a case the problem can be solved in polynomial time.

Theorem 5. *Let $n \geq 1$ and $1 \leq t \leq n$. Given an LDFA $\mathcal{A}$ over $\mathcal{L}_n$ and a t-separation $\mathcal{P} = \{\mathcal{P}_0, \ldots, \mathcal{P}_{t-1}\}$ of $\mathcal{L}_n$, constructing a minimal LDFA $\mathcal{A}'$ over $\mathcal{L}_t$ that t-separates $\mathcal{A}$ with respect to $\mathcal{P}$ can be done in polynomial time.*

Proof. Let $\mathcal{B}$ be the LDFA over $\mathcal{L}_t$ obtained from $\mathcal{A}$ by replacing each value $j \in [n]$ appearing in $\mathcal{A}$ by the value $i \in [t]$ for which $j \in \mathcal{P}_i$. Let $\mathcal{A}'$ be a minimal LDFA equivalent to $\mathcal{B}$. By [13], $\mathcal{A}'$ can be constructed in time polynomial in $\mathcal{B}$, which can clearly be constructed in time polynomial in $\mathcal{A}$. We claim that $\mathcal{A}'$ is a minimal LDFA that t-separates $\mathcal{A}$ with respect to $\mathcal{P}$.

We first show that for all $w \in \Sigma^*$, we have that $L(\mathcal{A}')(w) = i$ iff $L(\mathcal{A})(w) \in \mathcal{P}_i$. Since $L(\mathcal{A}') = L(\mathcal{B})$, it is enough to show that for all $w \in \Sigma^*$ we have $L(\mathcal{B})(w) = i$ iff $L(\mathcal{A})(w) \in \mathcal{P}_i$. Let $w \in \Sigma^*$. It holds that $L(\mathcal{A})(w) \in \mathcal{P}_i$ iff at least one of the values read along the run of $\mathcal{A}$ on w belong to $\mathcal{P}_i$, and the other values belong to $\mathcal{P}_i, \ldots, \mathcal{P}_{t-1}$. This holds iff at least one of the values read along the run of $\mathcal{B}$ on w equals to i, and the other values are in $\{i, \ldots, t-1\}$. Finally, this holds iff $L(\mathcal{B})(w) = i$.

The fact that $\mathcal{A}'$ has a minimal number of states follows from the correctness of the minimization algorithm, and from the fact that all LDFAs that t-separate $\mathcal{A}$ with respect to a specific t-separation have the same language. □

We now turn to consider the case where the user does not provide a specific t-separation. That is, we are given an LDFA $\mathcal{A}$ over $\mathcal{L}_n$ and $t \geq 1$, and we seek an LDFA $\mathcal{A}'$ with a minimal number of states that t-separates $\mathcal{A}$. For example, as discussed in Section 1, when the user does not care about the way priorities are grouped, or, in the case of abstraction, when the user hesitates between working with an over- or an under-approximation, the t-separation is not given. Consider the corresponding decision problem SEPLDFA=$\{\langle \mathcal{A}, n, t, k\rangle : \mathcal{A}$ is an LDFA

over $\mathcal{L}_n$ with a t-separating $\mathcal{A}'$ over $\mathcal{L}_t$ such that $|\mathcal{A}'| \leq k\}$. As in Section 3, we study also the family of problems (n,t)-SEPLDFA, in which n and t are not parameters and rather are fixed. That is, (n,t)-SEPLDFA=$\{\langle \mathcal{A}, k\rangle : \mathcal{A}$ is an LDFA over $\mathcal{L}_n$ with a t-separating $\mathcal{A}'$ over $\mathcal{L}_t$ such that $|\mathcal{A}'| \leq k\}$.

Theorem 6. *For all $n \geq 1$ and $t \geq 1$, the problem (n,t)-SEPLDFA can be solved in polynomial time.*

Proof. For fixed $n \geq 1$ and $t \geq 1$, there is a fixed number of possible t-separations of $\mathcal{L}_n$. Therefore, one can go over all t-separations, construct for each the corresponding minimal LDFA and return"yes" iff one of them has at most k states. By Theorem 5, each check, and therefore also the whole procedure, can be done in polynomial time. □

It is not hard to see that the algorithm above shows that the problem stays solvable in polynomial time also when n is not fixed and is a parameter to the problem.

We now turn to study the problem SEPLDFA, in which n and t are parameters, and show that this problem is NP-complete. In Section 3, the NP-hardness of APRXLDFA follows directly from the hardness of (n,t)-APRXLDFA for $1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$. Here, however, the problem (n,t)-SEPLDFA can be solved in polynomial time for all $n \geq 1$ and $t \geq 1$, so the fact that n and t are parameters is crucial.

Theorem 7. *The problem SEPLDFA is NP-complete.*

Proof. As in the case of (n,t)-APRXLDFA, membership in NP follows directly from Theorem 4.

For the lower bound, we show a polynomial time reduction from the NP-complete Maximum-Bisection problem on regular graphs (MBRG, for short) [11]. The Maximum Bisection of a graph $G = \langle V, E\rangle$, for V of an even size, is a partition of V into two equally sized sets that maximizes the number of edges between those sets. For regular graphs, in which all vertices have the same degree, the problem coincides with the problem of finding $T \subseteq V$ such that $|T| = \frac{|V|}{2}$ and $e(T)$ is minimal, where $e(T)$ is the number of edges among the vertices of T. Formally, $e(T) = |E \cap (T \times T)|$. The corresponding decision problem can therefore be formulated as MBRG = $\{\langle G, k\rangle : G = \langle V, E\rangle$ is an undirected regular graph with an even number of vertices, such that there is a set $T \subseteq V$ with $|T| = \frac{|V|}{2}$ and $e(T) \leq k\}$.

For technical convenience, instead of reducing the MBRG problem to SEPLDFA directly, we go through the following variant of the problem: MBRG$' = \{\langle G, v, k\rangle : G = \langle V, E\rangle$ is an undirected graph with an odd number of vertices, the vertex v touches all other vertices, and there is a set $T \subseteq V$ with $|T| = \frac{|V|-1}{2}$, $v \notin T$, and $e(T) \leq k\}$.

Lemma 1. *MBRG$'$ is NP-complete.*

Proof. Membership in NP is trivial. We show a polynomial time reduction from MBRG to MBRG$'$, proving that it is NP-hard. Let $\langle G, k\rangle$ be an input for MBRG, where $G = \langle V, E\rangle$. We add a new vertex v to V, and connect v to all other vertices. Note that the number of vertices is now odd. We denote by $G' = \langle V', E'\rangle$ the graph obtained, and produce $\langle G', v, k\rangle$ as an input for MBRG$'$. It is not hard to see that $\langle G, k\rangle \in$ MBRG iff $\langle G', v, k\rangle \in$ MBRG$'$. Indeed, the same set T works for both graphs. That is, $T \subseteq V$ is a set of $\frac{|V|}{2}$ vertices such that $e(T) \leq k$ iff $T \subseteq V'$ is a set of $\frac{|V'|-1}{2}$ vertices such that $v \notin T$ and $e(T) \leq k$. □

We now turn to describe the reduction from MBRG′ to SEPLDFA. Let $\langle G, v, k\rangle$ be an input to MBRG′, where $G = \langle V, E\rangle$ is such that $V = \{v_1, \ldots, v_n, v\}$ and $E = \{e_1, \ldots, e_m\}$. Note that $n = |V| - 1$ and $m = |E|$. We construct an LDFA $\mathcal{A} = \langle \mathcal{L}_{n+1}, \Sigma, Q, \delta, Q_0, F\rangle$, where:

- $\Sigma = \{a_1, \ldots, a_m, b_1, \ldots, b_n, c_1, \ldots, c_n\}$. Thus, each edge $e_i \in E$ induces a letter a_i, and each vertex $v_i \in V \setminus \{v\}$ induces two letters, b_i and c_i.
- $Q = \{q_1, \ldots, q_m, q_1^1, q_1^2, q_2^1, q_2^2, \ldots, q_n^1, q_n^2, q_{init}, q_{fin}\}$. Thus, each edge $e_i \in E$ induces a state q_i, and each vertex $v_i \in V \setminus \{v\}$ induces two states q_i^1 and q_i^2. In addition there are two states q_{init} and q_{fin}.
- The transition relation is defined as follows.
  - For all $1 \leq i \leq m$, we have $\delta(q_{init}, a_i, q_i) = \top$.
  - For all $1 \leq i \leq m$ and $1 \leq j \leq n$, if e_i touches v_j, then $\delta(q_i, b_j, q_j^1) = \top$, otherwise $\delta(q_i, b_j, q_j^2) = \top$.
  - For all $1 \leq j \leq n$, we have $\delta(q_j^2, b_j, q_j^1) = \delta(q_j^1, b_j, q_j^1) = \top$.
  - For all $1 \leq j \leq n$, we have $\delta(q_j^1, c_j, q_{fin}) = \delta(q_j^2, c_j, q_{fin}) = \top$.
  - For all other $q, q' \in Q$ and $\sigma \in \Sigma$, we have $\delta(q, \sigma, q') = \bot$.
- $Q_0(q_{init}) = \top$, and $Q_0(q) = \bot$ for all other $q \in Q$.
- For all $1 \leq j \leq n$, we have $F(q_j^1) = j$ and $F(q_j^2) = j - 1$. For all other $q \in Q$, we have $F(q) = \top$.

Note that $\mathcal{A}$ is indeed deterministic, and has $m + 2n + 2$ states. Also, since the components of $\mathcal{A}$ are all of size polynomial in the input graph, the reduction is polynomial. We refer to the states q_i as "the left column" and to the states q_j^1 and q_j^2 as "the right column" (see Figure 6).

Example 4. Figure 6 depicts a graph G and its induced LDFA $\mathcal{A}_G$. For clarity, we do not draw all edges in the middle, only a symbolic sample. In addition, we omit transition values, which are all $\top$, and omit acceptance values, which are all $\top$ except for the states on the right column, where $F(q_j^1) = j$ and $F(q_j^2) = j - 1$ for all $1 \leq j \leq 4$.

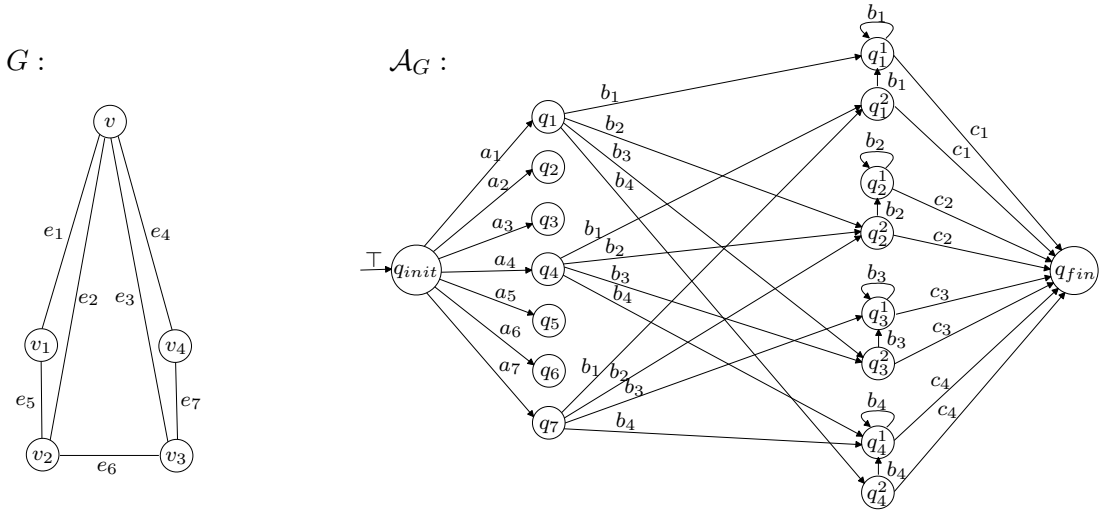

Fig. 6. A graph G and its induced LDFA $\mathcal{A}_G$.

We prove below that there exists a set $T \subseteq V$ such that $|T| = \frac{n}{2}$, $v \notin T$, and $e(T) \leq k$ iff there exists an LDFA $\mathcal{A}'$ over $\mathcal{L}_t$ with at most $k + 2n + 3$ states that t-separates $\mathcal{A}$, for $t = \frac{n}{2} + 1$.

For the sake of the proof, we redefine below the notion of a t-separating LDFA. The new definition is equivalent to the original one in the sense that for a given LDFA $\mathcal{A}$, there exists $\mathcal{A}'$ with k states that t-separates it by the original definition iff there exists $\mathcal{A}''$ with k states that t-separates it by the new definition.

Definition 4. *Consider an integer $n \geq 1$, an approximation factor $1 \leq t \leq n$, and two LDFAs $\mathcal{A}$ and $\mathcal{A}'$ over $\mathcal{L}_n$. We say that $\mathcal{A}'$ t-separates $\mathcal{A}$ iff there is a t-separation $\mathcal{P}$ of $\mathcal{L}_n$ such that for all words $w \in \Sigma^*$ we have that $L(\mathcal{A}')(w) \in \mathcal{P}_i$ iff $L(\mathcal{A})(w) \in \mathcal{P}_i$.*

Back to the proof, assume first that G has a set $T \subseteq V$ s.t. $|T| = \frac{n}{2}$, $v \notin T$, and $e(T) \leq k$. Let $v_{i_1}, v_{i_2}, \ldots, v_{i_{\frac{n}{2}}}$ be the vertices of T. For $t = \frac{n}{2} + 1$, we define a t-separation $\mathcal{P} = \{\mathcal{P}_0, \ldots, \mathcal{P}_{\frac{n}{2}}\}$ of $\mathcal{L}_{n+1}$ such that $\mathcal{P}_0 = \{0, 1, .., i_1 - 1\}$, $\mathcal{P}_1 = \{i_1, \ldots, i_2 - 1\}$, $\mathcal{P}_2 = \{i_2, \ldots, i_3 - 1\}$, $\ldots$, $\mathcal{P}_{\frac{n}{2}} = \{i_{\frac{n}{2}}, \ldots, n\}$. That is, $\mathcal{P}$ separates $\mathcal{L}_{n+1}$ on the indices of the vertices of T. We construct an LDFA $\mathcal{A}' = \langle \mathcal{L}_{n+1}, Q', \Sigma, \delta', Q'_0, F' \rangle$ that t-separates $\mathcal{A}$ with respect to $\mathcal{P}$, as decribed below (see Figure 7).

- The states Q' are defined as follows.
  - Consider the relation $\sim_T \subseteq E \times E$, where $e_1 \sim_T e_2$ iff for all $v \in T$ it holds that e_1 touches v iff e_2 touches v. In other words, e_1 and e_2 are equivalent iff they agree on touching the vertices of T. It is easy to see that $\sim_T$ is an equivalence relation. For each equivalence class C we add one state q_C to Q'.
  - For each $v_i \in V$, if $v_i \in T$ we add two states q_i^1 and q_i^2, otherwise we add one state $q_i^{1,2}$ (one can think of this state as a merge of q_i^1 and q_i^2, as explained in the sequel).
  - Finally, we add an initial and final states q_{init} and q_{fin}.
- The transition relation is defined as follows.
  - For all $1 \leq i \leq m$ and equivalence class C, if $a_i \in C$, then $\delta'(q_{init}, a_i, q_C) = \top$.
  - For all $1 \leq j \leq n$ and equivalence class C:
    - If $v_j \notin T$, then $\delta'(q_C, b_j, q_j^{1,2}) = \top$.
    - If $v_j \in T$ and the edges of C touch v_j, then $\delta'(q_C, b_j, q_j^1) = \top$.
    - If $v_j \in T$ and the edges of C do not touch v_j, then $\delta'(q_C, b_j, q_j^2) = \top$.
  - For all $1 \leq j \leq n$ such that $v_j \in T$, we have $\delta(q_j^2, b_j, q_j^1) = \delta(q_j^1, b_j, q_j^1) = \top$.
    For all $1 \leq j \leq n$ such that $v_j \notin T$, we have $\delta'(q_j^{1,2}, b_j, q_j^{1,2}) = \top$.
  - For all $1 \leq j \leq n$ such that $v_j \in T$, we have $\delta'(q_j^1, c_j, q_{fin}) = \delta'(q_j^2, c_j, q_{fin}) = \top$.
    For all $1 \leq j \leq n$ such that $v_j \notin T$, we have $\delta'(q_j^{1,2}, c_j, q_{fin}) = \top$.
  - For all other $q, q' \in Q$ and $\sigma \in \Sigma$, we have $\delta(q, \sigma, q') = \bot$.
- $Q'_0(q_{init}) = \top$, and $Q'_0(q) = \bot$ for all other $q \in Q'$.
- For all $1 \leq j \leq n$ such that $v_j \in T$, we have $F'(q_j^1) = j$ and $F'(q_j^2) = j - 1$, and for all $1 \leq j \leq n$ such that $v_j \notin T$, we have $F'(q_j^{1,2}) = j - 1$. For all other $q \in Q'$, we have $F'(q) = \top$.

Example 5. Figure 7 depicts a set $T = \{v_2, v_4\}$ in which $e(T)$ is minimal, together with the corresponding minimal LDFA $\mathcal{A}_G^{min}$ that t-separates $\mathcal{A}$, for $t = \frac{n}{2} + 1 = 3$. For clarity, we omit transition values, which are all $\top$, and omit acceptance values, which are all $\top$ except for the states on the right column, where for all $1 \leq j \leq 4$, if $v_j \in T$, then $F'(q_j^1) = j$ and $F'(q_j^2) = j - 1$, and if $v_j \notin T$, then $F'(q_j^{1,2}) = j - 1$. (As explained above, it is technically convenient to define $\mathcal{A}'$ over $\mathcal{L}_{n+1}$ rather than over $\mathcal{L}_t$. We can later easily group

the elements and map them to $[t]$). Note that equivalent edges are drawing the same in the graph G, and that the equivalence classes corresponding to the states on the left column of $\mathcal{A}_G^{min}$ are $C_1 = \{e_1, e_3\}, C_2 = \{e_2, e_5, e_6\}$ and $C_3 = \{e_4, e_7\}$.

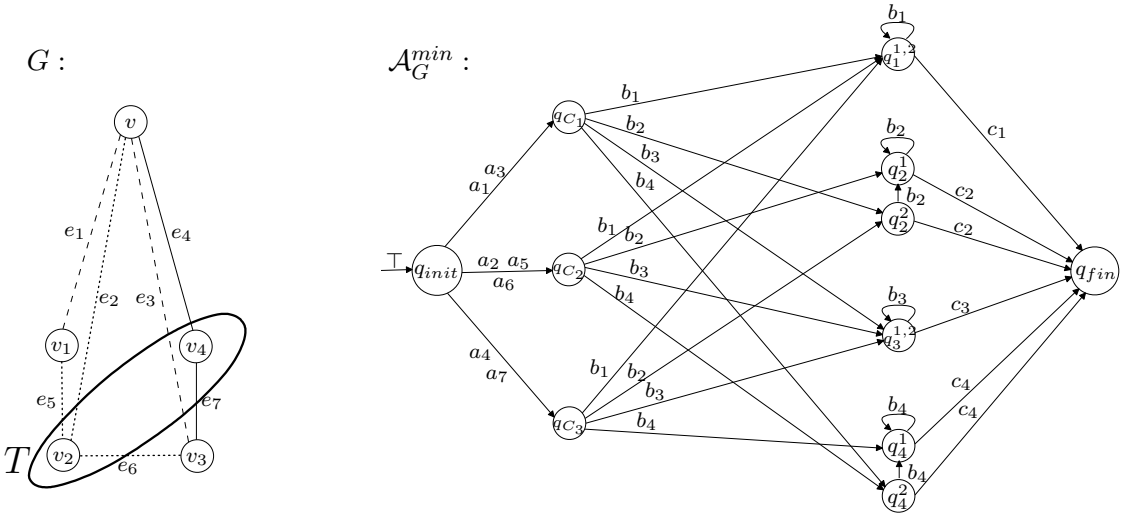

Fig. 7. A set $T = \{v_2, v_4\}$ in which $e(T)$ is minimal, and the corresponding minimal LDFA $\mathcal{A}_G^{min}$ that t-separates $\mathcal{A}$, for $t = 3$.

Lemma 2. $|\mathcal{A}'| \leq k + 2n + 3$.

Proof. We first consider the states at the right column, that is, states that are associated with vertices. It is not hard to see that there are exactly $\frac{n}{2} \cdot 2 + \frac{n}{2} \cdot 1 = \frac{3}{2}n$ such states, because vertices that belong to T induce two states, and vertices that do not belong to T induce one state. Next, we consider the states at the left column that are associated with edges, and claim that there are at most $k + \frac{n}{2} + 1$ such states. Since the number of those states equals the number of the equivalence classes of the relation $\sim_T$ defined above, it is enough to show that there are at most $k + \frac{n}{2} + 1$ equivalence classes of that relation. To see this, we consider three different types of edges, counting the number of classes contributed by each type. First, we consider edges $e = (u, v)$ such that $u, v \in T$. Note that edge of that type cannot be equivalent to any other edge, as there is exactly one edge touching both u and v. Therefore, each such edge contributes exactly one class. By the assumption, there are at most k edges of that type, meaning that at most k classes are contributed by this type of edges. We now turn to consider edges $e = (u, v)$ such that exactly one of u or v belongs to T. We can see that edges of that type that are thouching the same vertex of T are equivalent. Therefore, each of the vertices of T contributes at most one class, meaning that at most $\frac{n}{2}$ classes are contributed by the second type of edges. As for the third type, we consider edges $e = (u, v)$ such that $u, v \notin T$. If there exist edges of that type, then they are all equivalent, so at most one class is contributed by that type. Altogether, we get that there are at most $k + \frac{n}{2} + 1$ equivalence classes of the relation $\sim_T$, meaning that there are at most $k + \frac{n}{2} + 1$ states at the left column of $\mathcal{A}'$. Finally, summing up

the states of the right and left columns together with the initial and final states, we get a total of at most $k + 2n + 3$ states, and we are done. □

Lemma 3. *$\mathcal{A}'$ t-separates $\mathcal{A}$ with respect to $\mathcal{P}$, for $t = \frac{n}{2} + 1$.*

Proof. We show that for all $w \in \Sigma^*$ and for all $0 \leq i \leq \frac{n}{2}$ it holds that $L(A)(w) \in \mathcal{P}_i$ iff $L(A')(w) \in \mathcal{P}_i$. It is not hard to see that $L(\mathcal{A})$ and $L(\mathcal{A}')$ are as follows.

$$L(\mathcal{A})(w) = \begin{cases} \top & w = \epsilon \\ \top & w = a_i \\ j & (w = a_i b_j) \wedge (e_i \text{ touches } v_j) \\ j-1 & (w = a_i b_j) \wedge (e_i \text{ does not touch } v_j) \\ j & w = a_i b_j b_j^+ \\ \top & w = a_i b_j^+ c_j \\ \bot & \text{otherwise} \end{cases}$$

$$L(\mathcal{A}')(w) = \begin{cases} \top & w = \epsilon \\ \top & w = a_i \\ j & (w = a_i b_j) \wedge (e_i \text{ touches } v_j) \wedge (v_j \in T) \\ j-1 & (w = a_i b_j) \wedge (e_i \text{ does not touch } v_j) \wedge (v_j \in T) \\ j & (w = a_i b_j b_j^+) \wedge (v_j \in T) \\ j-1 & (w = a_i b_j^+) \wedge (v_j \notin T) \\ \top & w = a_i b_j^+ c_j \\ \bot & \text{otherwise} \end{cases}$$

Comparing $L(\mathcal{A})$ and $L(\mathcal{A}')$, we can see that the first two lines and the last two lines remain identical, and that lines 3-5 remain identical as long as $v_j \in T$. The only difference occurs for words of the form $a_i b_j^+$ with $v_j \notin T$ (6'th line in the definition of $L(\mathcal{A}')$), because those words were mapped by $L(A)$ to either j or $j-1$, while $L(\mathcal{A}')$ maps them all to $j-1$. This difference does not break the approximation though. To see this, it is enough to notice that j and $j-1$ belong to the same set $\mathcal{P}_i$. This is implied by the fact that the sets $\mathcal{P}_0, \ldots, \mathcal{P}_{\frac{n}{2}}$ were separated exactly at the indices $\{i_1, \ldots, i_{\frac{n}{2}}\}$ of the vertices of T, and by the fact that $v_j \notin T$. We conclude that for all $w \in \Sigma^*$ and for all $0 \leq i \leq \frac{n}{2}$ it holds that $L(A)(w) \in \mathcal{P}_i$ iff $L(A')(w) \in \mathcal{P}_i$, and we are done. □

By Lemma 2 and Lemma 3 we get that $\mathcal{A}'$ has at most $k + 2n + 3$ states, and that it t-separates $\mathcal{A}$, for $t = \frac{n}{2} + 1$, as required.

On the other hand, assume that there exists an LDFA $\mathcal{A}' = \langle \mathcal{L}_{n+1}, Q', \Sigma, \delta', Q'_0, F' \rangle$ with at most $k + 2n + 3$ states that t-separates $\mathcal{A}$, for $t = \frac{n}{2} + 1$. We show that G has a set $T \subseteq V$ s.t. $|T| = \frac{n}{2}$, $v \notin T$, and $e(T) \leq k$.

The fact that $\mathcal{A}'$ t-separates $\mathcal{A}$, for $t = \frac{n}{2} + 1$, implies that there is a partition of $\mathcal{L}$ into $\frac{n}{2} + 1$ non-empty sets $\mathcal{P}_0, \ldots, \mathcal{P}_{\frac{n}{2}}$ of successive elements, such that for all $w \in \Sigma^*$ and for all $0 \leq i \leq \frac{n}{2}$ it holds that $L(A)(w) \in \mathcal{P}_i$ iff $L(A')(w) \in \mathcal{P}_i$. Let $\{i_1, \ldots, i_{\frac{n}{2}}\}$ be the indices that separate those sets, that is, the sets are $\mathcal{P}_0 = \{0, 1, .., i_1 - 1\}$, $\mathcal{P}_1 = \{i_1, \ldots, i_2 - 1\}$, $\mathcal{P}_2 = \{i_2, \ldots, i_3 - 1\}$,..., $\mathcal{P}_{\frac{n}{2}} = \{i_{\frac{n}{2}}, \ldots, n\}$. We define $T = \{v_{i_1}, \ldots, v_{i_{\frac{n}{2}}}\}$. Of course, it holds that $|T| = \frac{n}{2}$ and $v \notin T$, so it is left to show that $e(T) \leq k$.

Recall that $\mathcal{A}'$ is deterministic, and therefore has exactly one initial state. Let q'_{init} be that state, and let $B = \{q \in Q' : \delta'(q'_{init}, a_i, q) \neq \bot \text{ for some } a_i \in \Sigma\}$. That is, $B \subseteq Q'$ contains

all states reachable from the initial state by $a_1, \ldots, a_m$. Recall that the states in B correspond to edges between vertices in T. Accordingly, we now turn to show that $|B| \leq k + \frac{n}{2} + 1$.

Lemma 4. $|B| \leq k + \frac{n}{2} + 1$.

Proof. Recall that $|\mathcal{A}'| \leq k + 2n + 3$. Hence, it is enough to show that $|Q' \setminus B| \geq \frac{3}{2}n + 2$. We show it by considering words that do not reach the states of B, proving that these words must reach at least $\frac{3}{2}n + 2$ different states.

We start with an observation that will be repeatedly used along the proof. Let $w_1, w_2 \in \Sigma^*$ be two words for which there exist $w_1', w_2' \in \Sigma^*$ such that $L(\mathcal{A})(w_1 w_1') = \top$ and $L(\mathcal{A})(w_2 w_2') = \top$. This fact implies that the traversal values $tr\_val(w_1)$ and $tr\_val(w_2)$ read in $\mathcal{A}'$ must belong to $\mathcal{P}_t$. Assume further that there exists $z \in \Sigma^*$ such that $L(\mathcal{A})(w_1 z)$ and $L(\mathcal{A})(w_2 z)$ belong to two different sets $\mathcal{P}_i$ and $\mathcal{P}_j$, respectively. We claim that in such a case the words w_1 and w_2 cannot reach the same state in $\mathcal{A}'$. Assume by contradiction that w_1 and w_2 are reaching the same state, and let q be that state. Consider the state q' reached when z is read from q, and consider the value $x_z = \delta(q, z, q') \wedge F(q')$ (where δ is naturally extended to apply on words rather than on letters). Recall that both w_1 and w_2 are reaching q with traversal value taken from $\mathcal{P}_t$. Now, if $x_z \in \mathcal{P}_i$, then $L(\mathcal{A}')(w_2 z) \in \mathcal{P}_i$, which contradicts the fact that $L(\mathcal{A}')(w_2 z)$ should be taken from $\mathcal{P}_j$. Symmetrically, if $x_z \in \mathcal{P}_j$, then $L(\mathcal{A}')(w_1 z) \in \mathcal{P}_j$, which contradicts the fact that $L(\mathcal{A}')(w_1 z)$ should be taken from $\mathcal{P}_i$. If $x_z \notin \mathcal{P}_i \cup \mathcal{P}_j$ then of course the contradiction is met for both $w_1 z$ and $w_2 z$. We conclude that w_1 and w_2 cannot reach the same state. In such a case we say that z is a *distinguishing tail* between w_1 and w_2.

Going back to our proof, we start by considering words of the form $a_i b_j^+$. We show that these words cannot reach any of the states of B. For this purpose, we have to find a distinguishing tail between words of the form a_l and $a_i b_j^+$ for all $1 \leq l \leq m$, $1 \leq i \leq m$ and $1 \leq j \leq n$. Let l, i, j be such indices. Applying the observation above on $w_1 = a_l$, $w_2 \in a_i b_j^+$, $w_1' = \epsilon$, $w_2' = c_j$ and $z = c_j$ implies that words of the form $a_i b_j^+$ cannot reach any of the states of B, as c_j is a distinguishing tail. Therefore, we have at least one state in $Q' \setminus B$.

Next up, we distinguish between words of the form $a_i b_j^+$ and $a_i b_l^+$ for all $1 \leq i \leq m$ and $1 \leq j \neq l \leq n$, by applying the observation on $w_1 \in a_i b_j^+$, $w_2 \in a_i b_l^+$, $w_1' = c_j$, $w_2' = c_l$ and $z = c_j$. We conclude that there must be at least n states in $Q' \setminus B$.

To distinguish further between the words, we now turn to focus on words of the form $a_i b_j^+$ such that $v_j \in T$. Let e_l be an edge touching v_j and let e_r be an edge that does not[4]. We show that the words $a_l b_j$ and $a_r b_j$ cannot reach the same state. To see this, recall that $v_j \in T$, and that the indices of the vertices of T are exctly those that separate between the sets of $\mathcal{P}$. This implies that j and $j-1$ necessarily do not belong to the same set. Now, we can apply the observation on $w_1 = a_l b_j$, $w_2 = a_r b_j$, $w_1' = c_j$, $w_2' = c_j$ and $z = \epsilon$ and conclude that $a_l b_j$ and $a_r b_j$ cannot reach the same state. Going back to the n different states we have already found, we conclude that $\frac{n}{2}$ of them must be splitted into two states, resulting in at least $n + \frac{n}{2} = \frac{3}{2}n$ states in $Q' \setminus B$ so far.

Consider now the empty word ϵ. We can see that ϵ cannot reach any of the states of B, by applying the observation on $w_1 = \epsilon$, $w_2 = a_i$, $w_1' = \epsilon$, $w_2' = \epsilon$ and $z = a_i$ for all $1 \leq i \leq m$. Also, we can distinguish between ϵ and words of the form $a_i b_j^+$ by applying the observation on $w_1 = \epsilon$, $w_2 \in a_i b_j^+$, $w_1' = \epsilon$, $w_2' = c_j$ and $z = a_i$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$. We conclude that ϵ cannot reach any of the states reached by words cosidered so far, so we gain one more state for $Q' \setminus B$, resulting in at least $\frac{3}{2}n + 1$ states.

[4] Note that such edges exist for all vertices of T. To see this, recall that G has a vertex touching all vertices. Hence, for a vertex $w \in T$, we can take $e_l = (w, v)$ and $e_r = (w', v)$ for some $w' \notin T$

Finally, we consider words of the form $a_i b_j^+ c_j$, for $1 \leq i \leq m$ and $1 \leq j \leq n$. We can see that such words cannot reach any of the states of B, by applying the observation on $w_1 \in a_i b_j^+ c_j$, $w_2 = a_l$, $w_1' = \epsilon$, $w_2' = \epsilon$ and $z = b_j c_j$ for all $1 \leq i \leq m$, $1 \leq l \leq m$, and $1 \leq j \leq n$. Also, we can distinguish between $a_i b_j^+ c_j$ and words of the form $a_l b_r^+$ by applying the observation on $w_1 \in a_i b_j^+ c_j$, $w_2 \in a_l b_r^+$, $w_1' = \epsilon$, $w_2' = c_r$ and $z = c_r$ for all $1 \leq i \leq m$, $1 \leq l \leq m$, $1 \leq j \leq n$ and $1 \leq r \leq n$. In addition, we can distinguish between $a_i b_j^+ c_j$ and ϵ, by applying the observation on $w_1 \in a_i b_j^+ c_j$, $w_2 = \epsilon$, $w_1' = \epsilon$, $w_2' = \epsilon$ and $z = a_i$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$. We conclude that words of the form $a_i b_j^+ c_j$ contribute at least one additional state for $Q' \setminus B$, ending up with at least $\frac{3}{2}n + 2$ states, and we are done.□

Lemma 5. *The relation $\sim_T$ has at most $k + \frac{n}{2} + 1$ equivalence classes.*

Proof. Assume by way of contradiction that there are more than $k + \frac{n}{2} + 1$ equivalence classes. By Lemma 4, we conclude that there must be two indices $i \neq j$ and some $q \in B$ such that $e_i \not\sim_T e_j$ but the words a_i and a_j are reaching the same state. Since $e_i \not\sim_T e_j$, there exists $v_l \in T$ such that w.l.o.g e_i touches v_l while e_j does not. To reach the contradiction, we show that the words a_i and a_j cannot reach the same state. Recall that $v_l \in T$, and that the indices of the vertices of T are exctly those that separate between the sets. This implies that l and $l-1$ necessaily do not belong to the same set. Now, we can apply the observation stated in the proof of Lemma 4 on $w_1 = a_i$, $w_2 = a_j$, $w_1' = \epsilon$, $w_2' = \epsilon$ and $z = b_l$ and conclude that a_i and a_j cannot reach the same state. We conclude that the number of equivalence classes of the relation $\sim_T$ is at most $k + \frac{n}{2} + 1$. □

We now turn to show that $e(T) \leq k$. As in the proof of Lemma 2, we consider three different types of edges, counting the number of classes contributed by each type. Recall that the types are as follows: The first type contains edges (u, v) such that $u, v \in T$, the second contains edges (u, v) such that exactly one of u and v belongs to T, and the third contains edges (u, v) such that $u, v \notin T$. We show that the second type contributes at least $\frac{n}{2}$ classes and the third contributes at least one. Consider the second type, and note that if two edges of that type touch two different vertices of T then those edges are not equivalent. Recall that G has a vertex v connected to all other vertices. Since $v \notin T$, the set $\{(v, u) : u \in T\}$ contains $\frac{n}{2}$ edges of the second type that are touching all $\frac{n}{2}$ different vertices of T. We therefore conclude that the second type contributes at least $\frac{n}{2}$ classes. As for the third type, it is enough to show that there is at least one edge that does not touch the vertices of T. This is clear, since we can take the edge (u, v) for some $u \neq v$ such that $u \notin T$. By Lemma 5, it follows that the first type contributes at most k classes. Now, as stated in the proof of Lemma 2, the number of classes of the first type equals $e(T)$, meaning that $e(T) \leq k$, and we are done. □

We note that although the SEPLDFA problem is generally NP-hard when n and t are given as parameters, there are still cases of parameters for which the problem can be solved in polynomial time. For example, consider the family of pairs $\langle n, t \rangle$ such that $t = n - c$, for a fixed $c \geq 0$. The number of possible t-separations in these cases is polynomial, so one can apply the same considerations as in Theorem 6 and solve the problem in polynomial time. Also, as in the case of t-approximation, the problem of returning the minimal LDFA that t-separates a given LDFA, for parameters t and n, is in FNP. Finally, comparing Theorems 5 and 7 we get that the computational bottleneck of SEPLDFA is the need to find a good t-separation. Once such a separation is given, finding a minimal LDFA can be done in polynomial time.

5 Discussion

We studied the problem of finding a minimal LDFA that approximates a given LDFA defined with respect to a fully-ordered lattice. We showed that the complexity of the problem depends on the relation between the lattice size and the approximation factor and also depends on whether we view them as fixed.

The complexity result of SEPLDFA may remind the reader of classic NP-complete problems, like vertex cover, where the goal is to decide the existence of some object ("the witness") of a certain size k. Typically, the existence of the required object can be decided in polynomial time for a fixed k, while the problem is NP-complete when k is a parameter to the problem. Despite of this resemblance, our setting here is very different, and the NP-hardness proof is quite challenging. To see the difference, note that the factors we fix in (n, t)-SEPLDFA do not include the size of the witness! The latter is k, which is part of the input. Another difficulty we face follows from the fact that, unlike in classic combinatorial problems, where, say, the vertices in the graph are not ordered, here we have no symmetry between the elements. For example, when an LDFA reads a lattice value that is greater than the values read already, the accumulated value is not affected. On the other hand, reading a value that is smaller affects the accumulated value. Coping with non-symmetry involves the design of languages that take into an account the order induced by the lattice, making our reductions complicated. The complication is reflected also in the fact that when $t = 0$, it is possible to use this non-symmetry and come up with a polynomial algorithm for (n, t)-APRXLDFA. Finally, note that our "fixed-parameter" variants fix both n and t, and still (n, t)-APRXLDFA is NP-hard when $1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$.

It is not hard to see that our bounds and proofs stay valid also for the t-APRXLDFA and t-SEPLDFA variants, when only t is fixed. In particular, t-SEPLDFA can be solved in polynomial time.

As discussed in Section 1, distance metrics can be defined also for partially-ordered lattices. In our future work we plan to study minimization of approximating LDFAs defined with respect to such lattices. Working with partially-ordered lattices, the notion of distance is more vague. We may therefore look for possible linearizations of the partial order, for example by using lattice chains, and apply techniques developed for fully-ordered lattices.

References

1. R. Alur, A. Kanade, and G. Weiss. Ranking automata and games for prioritized requirements. In *Proc. 20th Int. Conf. on Computer Aided Verification*, volume 5123 of *Lecture Notes in Computer Science*, pages 240–253. Springer, 2008.
2. B. Aminof, O. Kupferman, and R. Lampert. Formal analysis of online algorithms. In *9th Int. Symp. on Automated Technology for Verification and Analysis*, volume 6996 of *Lecture Notes in Computer Science*, pages 213–227. Springer, 2011.
3. G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *Proc. 11th Int. Conf. on Computer Aided Verification*, pages 274–287, 1999.
4. G. Bruns and P. Godefroid. Temporal logic query checking. In *Proc. 16th IEEE Symp. on Logic in Computer Science*, pages 409–420. IEEE Computer Society, 2001.
5. A. L. Buchsbaum, R. Giancarlo, and J. Westbrook. An approximate determinization algorithm for weighted finite-state automata. *Algorithmica*, 30(4):503–526, 2001.
6. W. Chan. Temporal-logic queries. In *Proc. 12th Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 450–463. Springer, 2000.
7. M. Chechik, B. Devereux, and A. Gurfinkel. Model-checking infinite state-space systems with fine-grained abstractions using SPIN. In *Proc. 8th Int. SPIN Workshop on Model Checking Software*, volume 2057 of *Lecture Notes in Computer Science*, pages 16–36. Springer, 2001.

8. M. Droste, W. Kuich, and H. Vogler (eds.). *Handbook of Weighted Automata*. Springer, 2009.
9. S. Easterbrook and M. Chechik. A framework for multi-valued reasoning over inconsistent viewpoints. In *Proc. 23rd Int. Conf. on Software Engineering*, pages 411–420. IEEE Computer Society Press, 2001.
10. Gerry Eisman and Bala Ravikumar. Approximate recognition of non-regular languages by finite automata. In *ACSC*, pages 219–228, 2005.
11. Uriel Feige, Marek Karpinski, and Michael Langberg. A note on approximating max-bisection on regular graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(43), 2000.
12. E. Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.
13. S. Halamish and O. Kupferman. Minimizing deterministic lattice automata. In *Proc. 14th Int. Conf. on Foundations of Software Science and Computation Structures*, volume 6604 of *Lecture Notes in Computer Science*, pages 199 – 213. Springer, 2011.
14. T.A. Henzinger. From Boolean to quantitative notions of correctness. In *Proc. 37th ACM Symp. on Principles of Programming Languages*, pages 157–158, 2010.
15. A. Hussain and M. Huth. On model checking multiple hybrid views. Technical Report TR-2004-6, University of Cyprus, 2004.
16. N. Immerman. Nondeterministic space is closed under complement. *Information and Computation*, 17:935–938, 1988.
17. O. Kupferman and Y. Lustig. Lattice automata. In *Proc. 8th Int. Conf. on Verification, Model Checking, and Abstract Interpretation*, volume 4349 of *Lecture Notes in Computer Science*, pages 199 – 213. Springer, 2007.
18. K.G. Larsen and G.B. Thomsen. A modal process logic. In *Proc. 3rd IEEE Symp. on Logic in Computer Science*, 1988.
19. M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
20. J. Myhill. Finite automata and the representation of events. Technical Report WADD TR-57-624, pages 112–137, Wright Patterson AFB, Ohio, 1957.
21. A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.
22. Leonard Pitt and Manfred K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM*, 40:95–142, 1993.
23. ESF Network programme. Automata: from mathematics to applications (AutoMathA). http://www.esf.org/index.php?id=1789, 2010.
24. S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In O. Grumberg, editor, *Proc. 9th Int. Conf. on Computer Aided Verification*, volume 1254, pages 72–83. Springer, 1997.
25. Bruce W. Watson, Derrick G. Kourie, Tinus Strauss, Ernest Ketcha Ngassam, and Loek G. Cleophas. Efficient automata constructions and approximate automata. *Int. J. Found. Comput. Sci.*, 19(1):185–193, 2008.