Minimizing 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. In contrast, weighted automata map each word to a value from a semiring over a large domain. The special case of *lattice automata*, in which the semiring is a finite lattice, has interesting theoretical properties as well as applications in formal methods. A *minimal deterministic automaton* captures the combinatorial nature and complexity of a formal language. Deterministic automata are used in run-time monitoring, pattern recognition, and modeling systems. Thus, the minimization problem for deterministic automata is of great interest, both theoretically and in practice.
For deterministic traditional automata on finite words, a minimization algorithm, based on the Myhill-Nerode right congruence on the set of words, generates in polynomial time a canonical minimal deterministic automaton. A polynomial algorithm is known also for deterministic weighted automata over the tropical semiring. For general deterministic weighted automata, the problem of minimization is open. In this paper we study minimization of deterministic lattice automata. We show that it is impossible to define a right congruence in the context of lattices, and that no canonical minimal automaton exists. Consequently, the minimization problem is much more complicated, and we prove that it is NP-complete. As good news, we show that while right congruence fails already for finite lattices that are fully ordered, for this setting we are able to combine a finite number of right congruences and generate a minimal deterministic automaton in polynomial time.

Categories and Subject Descriptors: F.1.1 [Computation by Abstract Devices]: Models of Computation–Automata, Relations between models; F.1.3 [Computation by Abstract Devices]: Complexity Measures and Classes–Reducibility and completeness; F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems–Computations on discrete structures; F.4.3 [Mathematical Logic and Formal Languages]: Formal Languages–Classes defined by grammars or automata, Decision problems

General Terms: Algorithms, Languages, Theory, Verification

Additional Key Words and Phrases: Deterministic Finite Automata, Minimization, Lattice Automata and Languages

1 Introduction

Automata theory is one of the longest established areas in Computer Science. Standard 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 [2]. 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 semiring over a large domain [7].

Focusing on applications in formal verification, the multi-valued setting arises directly in *quantitative verification* [11], 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 [10], *query checking* [5], which can be reduced to model checking over multi-valued systems, and verification of systems from *inconsistent viewpoints* [12], in which the value of the atomic propositions is the composition of their values in the different viewpoints.

Recall that in the multi-valued setting, the automata map words to a value from a semiring over a large domain. A *distributive finite lattice* is a special case of a semiring. A lattice $\langle A, \leq \rangle$ is a partially ordered set in which every two elements $a, b \in A$ have a least upper bound (a *join* b) and a greatest lower bound (a *meet* b). In many of the applications of the multi-valued setting described above, the values are taken from finite lattices. For example (see Figure 2), in the abstraction application, researchers use the lattice $\mathcal{L}_3$ of three fully ordered values [3], as well as its generalization to $\mathcal{L}_n$ [6]. In query checking, the lattice elements are sets of formulas, ordered by the inclusion order [4]. When reasoning about inconsistent viewpoints, each viewpoint is Boolean, and their composition gives rise to products of the Boolean lattice, as in $\mathcal{L}_{2,2}$ [8]. Finally, when specifying prioritized properties of system, one uses lattices in order to specify the priorities [1].

In [14], the authors study lattice automata, their theoretical properties, and decision problems for them. In a *nondeterministic lattice automaton* on finite words (LNFA, for short), 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 over an alphabet Σ and lattice $\mathcal{L}$ induces an $\mathcal{L}$-language $L : \Sigma^* \to \mathcal{L}$. 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 1 below is over the alphabet $\Sigma = \{0, 1, 2\}$ and the lattice $\mathcal{L} = \langle 2^{\{a,b,c,d\}}, \subseteq \rangle$. All states have acceptance value $\top$ (the greatest value in the lattice, which equals $\{a, b, c, d\}$), and this is also the initial value of the single initial state. The *join* and *meet* operators coincide with union and intersection, respectively. Thus, the $\mathcal{L}$-language of $\mathcal{A}$ is $L : \Sigma^* \to \mathcal{L}$ such that $L(\epsilon) = \top$, $L(0) = \{c, d\}$, $L(0 \cdot 0) = \{d\}$, $L(1) = \{a, b\}$, $L(1 \cdot 0) = \{a\}$, $L(2) = \{c, d\}$, $L(2 \cdot 0) = \{c\}$, and $L(x) = \bot$ for all other $x \in \Sigma^*$.

A *minimal deterministic automaton* captures the combinatorial nature and complexity of a formal language. Deterministic automata are used in run-time monitoring, pattern recognition, and modeling systems. 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 [21, 22]. In more detail, given a regular language L over Σ, then the relation $\sim_L \subseteq \Sigma^* \times \Sigma^*$, where $x \sim_L y$ iff for all $z \in \Sigma^*$ we have that $x \cdot z \in L$ iff $y \cdot z \in L$, is an equivalence relation, its equivalence classes correspond to the states of a minimal automaton for $\mathcal{A}$, and they also uniquely induce the transitions of such an automaton. Further, given a deterministic automaton for L, it is possible to use the relation $\sim_L$ in order to minimize it in polynomial time. From a theoretical point of view,

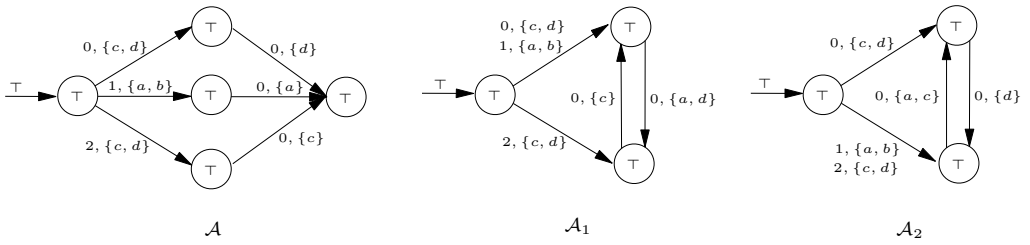

Fig. 1. An LDFA with two different minimal LDFAs.

the insights obtained from the study of minimization of DFAs have turned out to be useful also in the study of the complexity of regular languages and their algebraic properties. One theoretical motivation for our study is to try to obtain similar observations in the latticed setting.

A polynomial algorithm based on a right congruence is known also for deterministic weighted automata over the *tropical semiring* [19]. In such automata, each transition has a value in $\mathbb{R}$, each state has an initial and acceptance values in $\mathbb{R}$, and the value of a run is the sum of the values of its components. Unlike the case of DFAs, here there may be several different minimal automata. They all, however, have the same graph topology, and only differ by the values assigned to the transitions and states. In other words, there is a canonical minimal topology, but no canonical minimal automaton. For semirings that are not the tropical semiring, and in particular, for lattice automata, the minimization problem is open.

In this work we study the minimization problem for lattice automata. An indication that the problem is not easy is the fact that in the latticed setting, the "canonical topology" property does not hold. To see this, consider again the LDFA $\mathcal{A}$ in Figure 1. Note that an automaton for $L(\mathcal{A})$ must have at least three states. Indeed, if it has at most two then the word $w = 000$ would not get the value $\bot$, whereas $L(000) = \bot$. Hence, the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ presented on its right are two minimal automata for L. Their topologies differ by the transition leaving the initial state with the letter 1.[1] The absence of the "canonical topology" property suggests that efforts to construct the minimal LDFA by means of a right congruence are hopeless, as a right congruence induces a canonical minimal topology (in which the states are the equivalence classes and reading a letter σ from a state $[x]$ leads to the state $[x\sigma]$).

We formalize these discouraging indications by showing that the problem of LDFA minimization is NP-complete. This is a quite surprising result, as this is the first parameter in which lattice automata are more complex than weighted automata over the tropical semiring. In particular, lattice automata can always be determinized [14, 16], which is not the case for weighted automata over the tropical semiring [19]. Also, language containment between nondeterministic lattice automata can be solved in PSPACE [14, 15], whereas the containment problem for weighted automata on the tropical semiring is undecidable [17]. In addition, lattices have some appealing properties that general semirings do not, which make them seem simpler. Specifically, the idempotent laws (i.e., $a \vee a = a$ and $a \wedge a = a$) as well as the absorption laws (i.e., $a \vee (a \wedge b) = a$ and $a \wedge (a \vee b) = a$), do not hold in a general semiring, and do hold for lattices. Nevertheless, as mentioned, we are able to prove that their minimization is NP-complete.

[1] Note that the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are not simple, in the sense that the transitions are associated with values from the lattice that are not $\bot$ or $\top$. The special case of simple lattice automata, where the value of a run is determined only by the value associated with the last state of the run is simpler, and has been solved in the context of fuzzy automata [18, 23]. We will get back to it in Section 2.3.

Our NP-hardness proof is by a reduction from the vertex cover problem [13]. The lattice used in the reduction is $\mathcal{L} \subset 2^E$, for the set E of edges of the graph, with the usual set-inclusion order. The reduction strongly utilizes the fact that the elements of 2^E are not fully ordered. The most challenging part of the reduction is to come up with a lattice that, on the one hand, strongly uses the fact that $\mathcal{L}$ is not fully ordered, yet on the other hand is of size polynomial in E (and still satisfies the conditions of closure under meet and join).

As pointed above, the NP-hardness proof involved a partially ordered lattice, embodied in the "subset lattice", and strongly utilizes the order being partial. This suggests that for fully ordered lattices, we may still be able to find a polynomial minimization algorithm. Yet, as we shall show, the property of no canonical minimal LDFA is valid already in the case of fully ordered lattice, which suggests that no polynomial algorithm exists. As good news, we show that minimization of LDFAs over fully ordered lattices can nevertheless be done in polynomial time. The idea of the algorithm is to base the minimization on linearly many minimal DFAs that correspond to the different lattice values. The fact that the values are fully ordered enables us to combine these minimal automata into one minimal LDFA.

2 Preliminaries

This section introduces the definitions and notations related to lattices and lattice automata, as well as some background about the minimization problem.

2.1 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. A lattice is *complete* if for every subset $P \subseteq A$ both the least upper bound and the greatest lower bound of P exist, in which case they are denoted $\bigvee P$ and $\bigwedge P$, respectively. In particular, $\bigvee A$ and $\bigwedge A$ are denoted $\top$ (*top*) and $\bot$ (*bottom*), respectively. A lattice $\langle A, \leq \rangle$ is finite if A is finite. Note that every finite lattice is complete. A lattice $\langle A, \leq \rangle$ is *distributive* if for every $a, b, c \in A$, we have $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ and $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$.

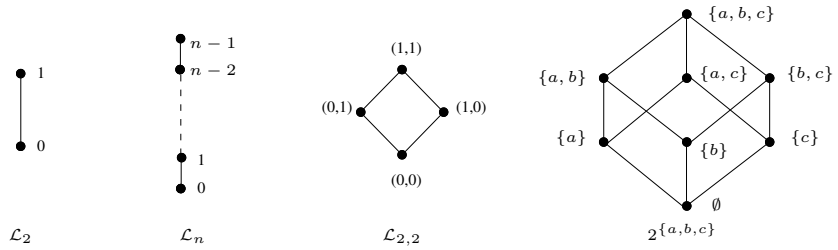

Fig. 2. Some lattices

In Figure 2 we describe some finite lattices. The elements of the lattice $\mathcal{L}_2$ are the usual truth values 1 (*true*) and 0 (*false*) with the order $0 \leq 1$. The lattice $\mathcal{L}_n$ contains the values $0, 1...n-1$,

with the order $0 \leq 1 \leq ... \leq n-1$. The lattice $\mathcal{L}_{2,2}$ is the Cartesian product of two $\mathcal{L}_2$ lattices, thus $(a, b) \leq (a', b')$ if both $a \leq a'$ and $b \leq b'$. Finally, the lattice $2^{\{a,b,c\}}$ is the power set of $\{a, b, c\}$ with the set-inclusion order. In this lattice, *join* and *meet* coincide with union and intersection, respectively, and we have, for example, $\{a\} \vee \{b\} = \{a, b\}$, $\{a\} \wedge \{b\} = \bot$, $\{a, c\} \vee \{b\} = \top$, and $\{a, c\} \wedge \{b\} = \bot$.

Consider a lattice $\mathcal{L}$ (we abuse notation and refer to $\mathcal{L}$ also as a set of elements, rather than a pair of a set with an order on it). For a set X of elements, an *$\mathcal{L}$-set over X* is a function $S : X \to \mathcal{L}$ assigning to each element of X a value in $\mathcal{L}$. Thus, $S \in \mathcal{L}^X$. 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 $x \in X$.

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

A *deterministic lattice automaton* on finite words (LDFA, for short) is a tuple $\mathcal{A} = \langle \mathcal{L}, \Sigma, Q, Q_0, \delta, F \rangle$, where $\mathcal{L}$ is a finite 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 $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n$ is a sequence $r = q_0, \ldots, q_n$ of $n + 1$ states, where q_0 is the initial state of $\mathcal{A}$, and for all $1 \leq i \leq n$ it holds that q_i is the *σ_i-destination* of q_{i-1}. The *value* of w is $val(w) = Q_0(q_0) \wedge \bigwedge_{i=1}^{n} \delta(q_{i-1}, \sigma_i, q_i) \wedge F(q_n)$. 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, $F(q_n)$ is the value of q_n being accepting, and the value of w is the meet of all these values. The *traversal value* of w is $tr\_val(w) = Q_0(q_0) \wedge \bigwedge_{i=1}^{n} \delta(q_{i-1}, \sigma_i, q_i)$, and its *acceptance value* is $F(q_n)$. 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 such a run does not exist, the value of the word is $\bot$. An example of an LDFA can be found in Figure 1.

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

An LDFA is *simple* if Q_0 and δ are Boolean. Note that the traversal value of a run r of a simple LDFA is either $\bot$ or $\top$, thus the value of r is induced by F. Simple LDFAs have been studied in the context of *fuzzy* logic and automata [18, 23].

Analyzing the size of $\mathcal{A}$, one can refer to $|Q|$, $|\delta|$, and $|\mathcal{L}|$ (where $|\mathcal{L}|$ denotes the number of elements in $\mathcal{L}$). 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. Our complexity results, however, refer to the size of the input, and thus take into account the other components of $\mathcal{A}$ as well, and in particular the size of $\mathcal{L}$.

2.2 Minimizing DFAs

This section is a reminder of the algorithm for minimizing DFAs using the Myhill-Nerode theorem [21, 22, 9]. As we shall see, the properties of DFAs that make the algorithm valid are not maintained in the context of lattice automata. Still the algorithm and the related properties are going to be relevant for understanding our contribution.

Given a Boolean language L over an alphabet Σ, consider the relation $\sim_L \subseteq \Sigma^* \times \Sigma^*$, where $x \sim_L x'$ iff for all $z \in \Sigma^*$ it holds that $x \cdot z \in L$ iff $x' \cdot z \in L$. In other words, x and x' are equivalent iff there is no tail z that distinguishes between them. It is well known that $\sim_L$ is an equivalence relation. We present here the proof for completeness: Reflexivity and symmetry are

obvious, so left to show is transitivity. Given $x, x', x'' \in \Sigma^*$ such that $x \sim_L x'$ and $x' \sim_L x''$, we show that $x \sim_L x''$. Assume by way of contradiction that this is not the case. Then, there exists some $z \in \Sigma^*$ such that w.l.o.g $x \cdot z \in L$ but $x'' \cdot z \notin L$. The contradiction is met when considering $x' \cdot z$. If $x' \cdot z \in L$ then $x'' \cdot z \in L$, contradicting the fact that $x'' \cdot z \notin L$. On the other hand, if $x' \cdot z \notin L$ then $x \cdot z \notin L$, contradicting the fact that $x \cdot z \in L$.

The equivalence relation $\sim_L$ induces a canonical minimal DFA $\mathcal{A}_{min}$. The states of $\mathcal{A}_{min}$ correspond to the equivalence classes of $\sim_L$, and the transitions between them are uniquely determined such that reading σ from a state corresponding to $[x]$ leads to the state corresponding to $[x \cdot \sigma]$. The initial state is the equivalence class of ϵ, and a state $[x]$ is accepting iff $x \in L$.

The fact that $L(\mathcal{A}_{min}) = L$ can be easily proved by induction on the length of an input word w, showing that $\mathcal{A}_{min}$ accepts w iff $w \in L$. To see that $\mathcal{A}_{min}$ is minimal and canonical, consider a DFA for L, and let δ be its transition function. For every two words $x, x' \in \Sigma^*$, if $\delta(q_0, x) = \delta(q_0, x')$, then it must be that $x \sim_L x'$. Otherwise, there is $z \in \Sigma^*$ such that w.l.o.g $x \cdot z \in L$ and $x' \cdot z \notin L$, whereas the runs of $\mathcal{A}$ on $x \cdot z$ and $x' \cdot z$ are either both accepting or both rejecting. It follows that the state space of a DFA for L can only refine the state space of $\mathcal{A}_{min}$, showing that $\mathcal{A}_{min}$ is minimal and canonical.

The algorithm for minimizing a given DFA therefore partitions the states of the DFA according to $\sim_L$, and this process can be done in polynomial time [9].

2.3 Difficulties in Applying Existing Minimization Methods on LDFAs

As discussed above, Myhill and Nerode suggest a general paradigm for minimization, based on the right congruence $\sim_L$. This paradigm has the "canonical topology" property: the right congruence directly induces a canonical minimal topology, in which the states correspond to the equivalence classes and the transitions are uniquely determined such that reading σ from a state $[x]$ leads to the state $[x \cdot \sigma]$. The canonicity derives from the fact that any other automaton for the same language can only refine the state space of the automaton induced by the right congruence.

The above general minimization paradigm is widely used in variety of settings. In paricular, it is used for minimizing weighted automata over the tropical semiring [19].

In this section we try to apply this paradigm on lattice automata, and explore the difficulties arises in the latticed setting.

In fact, for the case of simple LDFAs, the plan proceeds smoothly (see [18, 23], where the problem is discussed by means of fuzzy automata)[2]: Given an $\mathcal{L}$-language L, we extend the definition of $\sim_L$ to fit the nature of $\mathcal{L}$-languages. For all $x, x' \in \Sigma^*$, we say that $x \sim_L x'$ iff for all $z \in \Sigma^*$ it holds that $L(x \cdot z) = L(x' \cdot z)$. Clearly, $\sim_L$ is an equivalence relation, since it is based on the equality relation. As in the case of DFA, we can build a minimal simple LDFA $\mathcal{A}_{min}$ for L such that $|\mathcal{A}_{min}| = |\sim_L|$. We construct it in the same manner, only that here the acceptance values are defined such that $F([x]) = L(x)$. We show that every simple LDFA for L must have at least $|\sim_L|$ states. Indeed, if this is not the case, then we have two words $x, x' \in \Sigma^*$ reaching the same state q, while $x \not\sim_L x'$. The contradiction is reached when reading the distinguishing tail z from q, as in the case of DFA, due to the fact that in simple LDFAs the value of a word is solely determined by the final state. We get that simple LDFAs can be minimized in polynomial time, and the key for proving it was a generalization of the right congruence to require agreement on the values of the words. As in [18, 23], we conclude with the following.

[2] Several variants of fuzzy automata are studied in the literature. The difficulties we cope with in the minimization of lattice automata are not applied to them, and indeed they can be minimized by variants of the minimization construction described here [20].

Theorem 1. *Simple LDFAs can be minimized in polynomial time.*

Encouraged by this, we now turn to apply the above extended definition of $\sim_L$ on general LDFAs. Unfortunately, the extension does not seem to work here. To see the difficulties, consider an $\mathcal{L}$-language L over $\Sigma = \mathcal{L}$ such that $L(l_1 \cdot l_2 \cdots l_n) = \bigwedge_{i=1}^{n} l_i$. The language L can be recognized by an LDFA $\mathcal{A}$ with a single state q. The initial and acceptance values of q are $\top$, and for every $l \in \Sigma$, there is an l-transition with value l from q to itself. Thus, the single run of $\mathcal{A}$ on an input word maps it to the meet of its letters (see Figure 3).

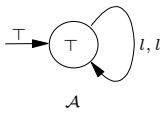

Fig. 3. An automaton $\mathcal{A}$ for the language L, in which $L(l_1 \cdot l_2 \cdots l_n) = \bigwedge_{i=1}^{n} l_i$.

Clearly, there exist $x, x' \in \Sigma^*$ such that $L(x) \neq L(x')$, and still $\mathcal{A}$ has only one state. Thus, despite being mapped to different values, x and x' reach the same state in $\mathcal{A}$. This observation shows a crucial difference between the setting of DFAs or simple LDFAs and the one of general LDFAs. It is only in the latter, that words with different values can reach the same state, as the value of a word is determined not only by the state it reaches, but also by the traversal value it accumulates. This fact implies that an attempt to distinguish between words according to their values results in LDFAs with needlessly more states.

However, the above example suggests another way to distinguish between words. In this example, all words could reach the same state as they all "want" to read the value l upon reading the letter l. Accordingly, a right congruence that may be helpful has to take into account the values that should be read with any possible tail, and in particular, in the case of L above, should have only one equivalence class.

Following the above discussion, we now try to define such a right congruence. Let us first consider a simpler model of LDFAs in which all acceptance values are $\top$. Note that in this model, for all $x, z \in \Sigma^*$ we have that $L(x \cdot z) \leq L(x)$. Let L be an $\mathcal{L}$-language in the model above. We define a relation $\sim_L \subseteq \Sigma^* \times \Sigma^*$ such that $x \sim_L x'$ iff for all $z \in \Sigma^*$ there exists $l \in \mathcal{L}$ such that $L(x \cdot z) = L(x) \wedge l$ and $L(x' \cdot z) = L(x') \wedge l$. That is, x and x' are equivalent iff for all tails $z \in \Sigma^*$ there exists some $l \in \mathcal{L}$ such that z can be read with the value l after reading either x or x'. Note that the relation $\sim_L$ indeed takes into a consideration the values that should be read after x and x' are read. Also, this relation fits the automaton in Figure 3, as every two words are equivalent under this relation (for a tail $z = l_1 ... l_n$, take $l = l_1 \wedge ... \wedge l_n$), and we therefore result in one class containing all words.

Unfortunately, the relation suffers another crucial problem - it is not even transitive. For example, for the language L of the LDFA $\mathcal{A}$ in Figure 1, we have $0 \sim_L 1$ (for $z = 0$, take $l = \{a, d\}$) and $1 \sim_L 2$ (for $z = 0$, take $l = \{a, c\}$), but $0 \not\sim_L 2$ (no l works for $z = 0$). The lack of transitivity in this example implies that the words 0 and 2 must reach different states, while the word 1 is free to join either of them. However, this results in two minimal automata with different topologies for $\mathcal{A}$! Their topologies differ by the transition leaving the initial state with the letter 1, as shown in Figure 1.

Having seen that the "canonical topology" property does not hold in the latticed setting, it is now clear why we could not come up with a right congruence. Indeed, as stated above, a right congruence always induces a canonical minimal topology.

In conclusion, we have seen some evidences that minimization of LDFAs cannot follow the minimization paradigm for DFAs and even not the one for deterministic weighted automata over the tropical semiring. In the rest of the paper we show that in fact the minimization problem for LDFAs is NP-complete in general. On the other hand, the evidences above do not contradict the existence of a polynomial time algorithm which does not follow this paradigm, and indeed we also describe a polynomial time algorithm for minimization of LDFAs over fully ordered lattices – a special case for which the evidences still apply.

3 Minimizing General LDFAs

In this section we study the problem of minimizing LDFAs and show that unlike the case of DFAs, and even the case of weighted DFAs over the tropical semiring, which can be minimized in polynomial time, here the problem is NP-complete. We consider the corresponding decision problem MINLDFA=$\{\langle \mathcal{A}, k\rangle : \mathcal{A}$ is an LDFA and there exists an LDFA $\mathcal{A}'$ with at most k states such that $L(\mathcal{A}') = L(\mathcal{A})\}$.

Theorem 2. *MINLDFA is NP-complete.*

Proof. We start with membership in NP. Given $\mathcal{A}$ and k, a witness to their membership in MINLDFA is an LDFA $\mathcal{A}'$ as required. Assuming $k \leq |\mathcal{A}|$ (otherwise, $\langle A, k\rangle$ clearly belongs to MINLDFA), the size of $\mathcal{A}'$ is linear in the input. Deciding language equivalence between LDFAs is NLOGSPACE-complete [14], thus we can verify that $L(\mathcal{A}') = L(\mathcal{A})$ in polynomial time.

For the lower bound, we show a polynomial time reduction from the Vertex Cover problem (VC, for short), proved to be NP-complete in [13]. Recall that VC=$\{\langle G, k\rangle : G$ is an undirected graph with a vertex cover of size $k\}$, where a *vertex cover* of a graph $G = \langle V, E\rangle$ is a set $C \subseteq V$ such that for all edges $(u, v) \in E$ we have $u \in C$ or $v \in C$.

Before we describe the reduction, we need some definitions. Consider an undirected graph $G = \langle V, E\rangle$. Let $n = |V|$ and $m = |E|$. For simplicity, we assume that V and E are ordered, thus we can refer to the minimum element in a set of vertices or edges. In particular, let $E = \{e_0, ..., e_{m-1}\}$. For $v \in V$, let $touch(v) = \{e : e = (v, u)$ for some $u\}$. For $e = (v_1, v_2) \in E$, let $far(e) = \min\{e' : e' \notin touch(v_1) \cup touch(v_2)\}$. That is, $far(e)$ is the minimal edge that is not adjacent to e. Note that if $\{e' : e' \notin touch(v_1) \cup touch(v_2)\} = \emptyset$, then $\{v_1, v_2\}$ is a VC of size two, so we can assume that $far(e)$ is well defined.

Example 1. In the graph G below, we have, for example, $touch(1) = \{a, b\}$, $touch(2) = \{d, e\}$, $far(a) = d$, $far(b) = e$, and $far(c) = e$.

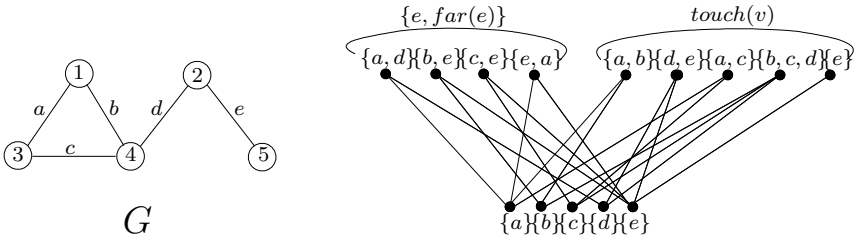

Fig. 4. A graph and its corresponding lattice.

We now turn to describe the reduction. Given an input $G = \langle V, E\rangle$, we construct an LDFA $\mathcal{A} = \langle \mathcal{L}, Q, \Sigma, \delta, Q_0, F\rangle$ as follows.

- $\mathcal{L} \subseteq 2^E$ contains the following elements: $\{\emptyset, E\} \cup \{\{e\} : e \in E\} \cup \{\{e, far(e)\} : e \in E\} \cup \{touch(v) : v \in V\}$, with the usual set-inclusion relation. In particular, $\bot = \emptyset$ and $\top = E$. Note that $\mathcal{L}$ contains at most $2 + n + 2m$ elements ("at most" since $\{e, far(e)\}$ may be identical to $\{far(e), far(far(e))\}$). For example, the graph in Example 1 induces the lattice shown on its right (for clarity, we omit the elements $\top$ and $\bot$ in the figure). Note that in this example we have $\{a, far(a)\} = \{far(a), far(far(a))\}$, so we omit the unnecessary element. We claim that though $\mathcal{L}$ does not contain all the elements in 2^E, the operators $join$ and $meet$ are well defined for all $l_1, l_2 \in \mathcal{L}$.[3] In the case l_1 and l_2 are ordered, the closure for both $join$ and $meet$ is obvious. Otherwise, we handle the operators separately as follows. We start with the case of $meet$. Closure to $meet$ is easy since $l_1 \wedge l_2$ never contains more than one edge. Indeed, if l_1, l_2 are of the form $touch(v_1)$, $touch(v_2)$ then their $meet$ is the single edge (v_1, v_2). In all other possibilities for l_1 and l_2 that are not ordered, one of them contains at most two edges, so the fact they are not ordered implies that they have at most one edge in their $meet$. As for $join$, given l_1 and l_2 let $S = \{l : l \geq l_1 \text{ and } l \geq l_2\}$. We claim that all the elements in S are ordered, thus we can define $l_1 \vee l_2$ to be the minimal element in S. Assume by way of contradiction that S contains two elements l and l' that are not ordered. On the one hand, $l \wedge l' \geq l_1 \vee l_2$. Since l_1 and l_2 are not ordered, this implies that $l \wedge l'$ is of size at least two. On the other hand, as we argued above, the $meet$ of two elements that are not ordered contains at most one edge, and we have reached a contradiction.
- $Q = \{q_{init}, q_0, ..., q_{m-1}\}$.
- $\Sigma = \{e_0, ..., e_{m-1}\} \cup \{\#\}$.
- For $0 \leq i < m$, we define $\delta(q_{init}, e_i, q_i) = \{e_i, far(e_i)\}$ and $\delta(q_i, \#, q_{(i+1)modm}) = \{e_i\}$. For all other $q, q' \in Q$ and $\sigma \in \Sigma$, we define $\delta(q, \sigma, q') = \bot$.
- $Q_0(q_{init}) = \top$, and $Q_0(q) = \bot$ for all other $q \in Q$.
- $F(q) = \top$ for all $q \in Q$.

For example, the graph G in Example 1 induces the LDFA $\mathcal{A}_G$ below (for clarity, we omit the acceptance values in the figure, as they are all $\top$).

It is not hard to see that the $\mathcal{L}$-language induced by $\mathcal{A}$, denoted L, is such that for all $e \in \Sigma$, we have that $L(e) = \{e, far(e)\}$ and $L(e{\cdot}\#) = \{e\}$. In addition, $L(\epsilon) = \top$, and for all other $w \in \Sigma^*$, we have that $L(w) = \bot$. Also, $\mathcal{A}$ is indeed deterministic, and has $m + 1$ states. Finally, since the components of $\mathcal{A}$ are all of size polynomial in the input graph, the reduction is polynomial.

We now turn to prove that G has a k-VC iff there is an LDFA with $k + 1$ states for L. Assume first that G has a k-VC $\{v_0, ..., v_{k-1}\}$. We construct an LDFA $\mathcal{A}' = \langle \mathcal{L}, Q', \Sigma, \delta', Q'_0, F'\rangle$ for L with $k + 1$ states as follows.

- $Q' = \{q_{init}, q_{v_0}, ..., q_{v_{k-1}}\}$.
- The transition relation is defined as follows.
  - Consider an edge $e = (u_1, u_2)$. We distinguish between two cases: (1) if only one of the vertices of e is in the cover, that is, there is a single i such that $u_1 = v_i$ or $u_2 = v_i$, then $\delta'(q_{init}, e, q_{v_i}) = \{e, far(e)\}$. (2) if both vertices of e are in the cover, that is, there are i and j such that $u_1 = v_i$ and $u_2 = v_j$, then if $j = (i{+}1)modk$, we define $\delta'(q_{init}, e, q_{v_j}) = \{e, far(e)\}$. Otherwise, we define $\delta'(q_{init}, e, q_{v_i}) = \{e, far(e)\}$. In other words, if v_1 and

[3] We note that this point has been the most challenging part of the reduction, as on the one hand, we have to strongly use the fact that $\mathcal{L}$ is not fully ordered (as we show in Section 4, polynomial minimization is possible for LDFAs over fully ordered lattice), yet on the other hand the reduction has to be polynomial and thus use only polynomially many values of the subset lattice. See also Remark 1.

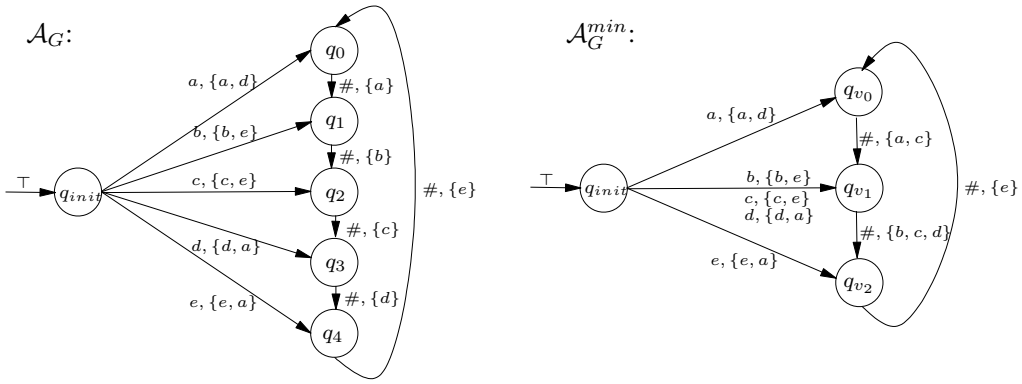

Fig. 5. The LDFA induced by G, and the minimal LDFA that corresponds to the 3-cover $\{3, 4, 5\}$.

v_2 are adjacent, we choose the right one, where $v_{(i+1)modk}$ is considered "right" with respect to v_i.
  - For all $0 \leq i < k$, we define $\delta'(q_{v_i}, \#, q_{v_{(i+1)modk}}) = touch(v_i)$.
  - For all other $q, q' \in Q$ and $\sigma \in \Sigma$, we define $\delta'(q, \sigma, q') = \bot$.
- $Q'_0(q_{init}) = \top$, and $Q_0(q) = \bot$ for all other $q \in Q$.
- $F'(q) = \top$ for all $q \in Q$.

In Example 1, the set $\{3, 4, 5\}$ is a minimal VC of G. Let $v_0 = 3, v_1 = 4, v_2 = 5$. The corresponding LDFA $\mathcal{A}_G^{min}$ shown on the right of Figure 3. Intuitively, in $\mathcal{A}_G^{min}$ it is possible to merge states of $\mathcal{A}_G$ that are associated with edges that are covered by the same vertex.

We prove that $L(\mathcal{A}) = L(\mathcal{A}')$. Let $L' = L(\mathcal{A}')$. For all $e \in E$, there is some v in the coverage such that $e \in touch(v)$. In addition, all the acceptance values are $\top$. Thus, for all $e \in E$ we have that $L'(e) = \{e, far(e)\}$. Also, after reading e, the letter $\#$ must be read with the value $touch(v_i)$ such that $e \in touch(v_i)$. Hence, since $\{e, far(e)\} \wedge touch(v_i) = \{e\}$, we get that $L'(e \cdot \#) = \{e\}$ for all $e \in \Sigma$, and the meet with $\top$ while reading the acceptance value has no additional effect. We turn on to consider words of the form $e \cdot \# \cdot \#^+$ and show that $\mathcal{A}'$ maps them to $\bot$. It is enough to show that $tr\_val(e \cdot \# \cdot \#) = \bot$. Note that $tr\_val(e \cdot \# \cdot \#) = \{e, far(e)\} \wedge touch(v_i) \wedge touch(v_{(i+1)modk})$, for some $0 \leq i < k$ such that $e \in touch(v_i)$. We already know that $\{e, far(e)\} \wedge touch(v_i) = \{e\}$. Thus, it is enough to show that $e \notin touch(v_{(i+1)modk})$. Recall that $e \in touch(v_i)$. Assume by way of contradiction that $e \in touch(v_{(i+1)modk})$ as well, then according to the order we induced on the vertices and since $v_{(i+1)modk}$ is the right with respect to v_i, we would get that the e-destination of q_{init} is $v_{(i+1)modk}$ rather than v_i. Thus, we get that $e \notin touch(v_{(i+1)modk})$, and this implies that $\mathcal{A}'$ maps words of the form $e \cdot \# \cdot \#^+$ to $\bot$. Finally, it is obvious that $L(\epsilon) = \top$ and that $L(w) = \bot$ for all other $w \in \Sigma^*$. Also, the number of the states in the automaton is indeed $k + 1$, as required.

Assume now that there is an LDFA $\mathcal{A}' = \langle \mathcal{L}, Q', \Sigma, \delta', Q'_0, F' \rangle$ with $k + 1$ states for $L(\mathcal{A})$. We show that G has a k-VC. Let us consider two subsets of Q' as follows:

- $T_1 = \{q_0\}$, where q_0 is the initial state of $\mathcal{A}'$.
- $T_2 = \{q \in Q' : \delta'(q_0, e, q) \neq \bot \text{ for some } e \in \Sigma \setminus \{\#\}\}$.

As we will see later, the states in T_2 correspond to vertices in the cover. Accordingly, we now turn to show that $|T_2| \leq k$. Recall that $|Q'| = k + 1$, thus $|T_1 \cup T_2| \leq k + 1$, and that $|T_1| = 1$. Hence, it is enough to show that $T_1 \cap T_2 = \emptyset$.

Assume by way of contradiction that $T_1 \cap T_2 \neq \emptyset$. Thus, since $T_1 = \{q_0\}$, it follows that $q_0 \in T_2$, which means that there is some edge e for which $\delta'(q_0, e, q_0) \neq \bot$. To reach the contradiction, we look at $L(\mathcal{A}')$ and show that $L(\mathcal{A}')(e \cdot e) \neq L(\mathcal{A})(e \cdot e)$. Recall that $L(\epsilon) = \top$, and note that therefore we have $Q_0(q'_0) = F(q'_0) = \top$. Now, together with the fact that $\delta'(q_0, e, q_0) \neq \bot$, we get that $L(\mathcal{A}')(e \cdot e) \neq \bot$, whereas $L(\mathcal{A})(e \cdot e) = \bot$. We conclude that $|T_2| \leq k$.

Consider the states in T_2. Since $L(\mathcal{A}')(e) \neq \bot$ for all letters $e \in \Sigma \setminus \{\#\}$, then all these letters reach some state in T_2. We show that for each $q \in T_2$ there exists a matching vertex $v_q \in V$ that covers all edges corresponding to the letters reaching q in $\mathcal{A}'$. The set of these vertices then covers E.

For every state $q \in T_2$, if there is only one letter $e = (v_1, v_2)$, $e \in \Sigma \setminus \{\#\}$, that reaches q, then we can arbitrarily take $v_q = v_1$. The case where there are several such letters is more complicated. In this case, we consider the state q' that is reached when $\#$ is read from q and the value $l = \delta(q, \#, q') \wedge F(q')$. Since $l \in \mathcal{L}$, it must be of the form $\bot, \top, touch(v), \{e\}$, or $\{e, far(e)\}$. We claim that l must be of the form $touch(v)$, and that the corresponding v covers all edges that reach q. Indeed, it is clear that $l \neq \bot$, since $L(e \cdot \#) \neq \bot$ for all $e \in \Sigma$. Also, $l \neq \top$, since $L(e \cdot \#) \neq L(e)$ for all $e \in \Sigma$. In order to show that l is not of the forms $\{e\}$ or $\{e, far(e)\}$, we use the fact that there are at least two letters e_1, e_2 reaching q. Assume by way of contradiction that $l = \{e\}$ for some $e \in E$. Thus, we get that $L'(e_1 \cdot \#) \leq \{e\}$ and $L'(e_2 \cdot \#) \leq \{e\}$. Also, since $L'(e_1 \cdot \#) = \{e_1\}$ and $L'(e_2 \cdot \#) = \{e_2\}$, it follows that $\{e_1\} \leq \{e\}$ and $\{e_2\} \leq \{e\}$. Of course, this is a contradiction since $e_1 \neq e_2$. It is left to show that l is not of the form $\{e, far(e)\}$. We assume by way of contradiction that $l = \{e, far(e)\}$ for some $e \in E$, and in the same manner as above we get that $\{e_1\} \leq \{e, far(e)\}$ and $\{e_2\} \leq \{e, far(e)\}$. Thus, it must be that $e = e_1$ or $e = e_2$. Assume w.l.o.g that $e = e_1$, and consider $L'(e_1 \cdot \#)$. Note that since $L'(e_1) = \{e_1, far(e_1)\}$, then after reading e_1 from q_0, we accumulate a value that is greater or equal to $\{e_1, far(e_1)\}$. Thus, meeting with $l = \{e_1, far(e_1)\}$, when reading $\#$, results in $\{e_1, far(e_1)\}$, instead of $\{e_1\}$. It follows that l must be of the form $touch(v)$. Now it is easy to see that the corresponding state v covers all edges that reach q. Indeed, for each such edge e it holds that $L(e \cdot \#) = \{e\}$, so the $meet$ with the value $touch(v)$ results in $\{e\}$, which means that $e \in touch(v)$. It follows that the set $\{v_q : q \in T_2\}$ is indeed a VC, it has at most k vertices, and we are done. □

Remark 1. In [24], the author proves NP-hardness of the problem of minimization for deterministic weighted finite-state automata, and in fact showed that the problem is NP-hard already for lattice automata. The reduction in [24] (from the minimum clique partition problem) uses a lattice that contains exponentially many elements. The lattice does have a polynomial description, thus the reduction is polynomial, yet the result is weaker than our result here. Indeed, the result in [24] only proves hardness for LDFAs over lattices with polynomial symbolic description (and possibly exponentially many elements), while our reduction proves hardness even for LDFAs over lattices with a polynomial number of elements.

4 Minimizing LDFAs Over Fully Ordered Lattices

In Section 3, we saw that the problem of minimizing LDFAs is NP-complete. The hardness proof involved a partially ordered lattice, embodied in the "subset lattice", and strongly utilized the order being partial. This suggests that for fully ordered lattices, we may still be able to find a polynomial minimization algorithm. On the other hand, as shown in Example 2, the property of no canonical minimal LDFA is valid already in the case of fully ordered lattices, which suggests that following the general "right-congruence paradigm" would fail in this case, and that different techniques should be applied here.

Example 2. The LDFA $\mathcal{A}$ in Figure 6 below is over the alphabet $\Sigma = \{a, b, c, \#\}$ and the lattice $\mathcal{L} = \langle\{0, 1, 2, 3\}, \leq\rangle$. The *join* and *meet* operators correspond to *max* and *min*, respectively. Thus, 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 $L(x) = 0$ for all other $x \in \Sigma^*$. The automata $\mathcal{A}_1$ and $\mathcal{A}_2$ presented on its right are two minimal automata for L. Their topologies differ by the transition leaving the initial state with the letter b. Note that L is monotonic, in the sense that for all $x, z \in \Sigma^*$, we have that $L(x \cdot z) \leq L(x)$. Thus, even for a monotonic language over a fully ordered lattice, there are two minimal LDFAs with different topologies.

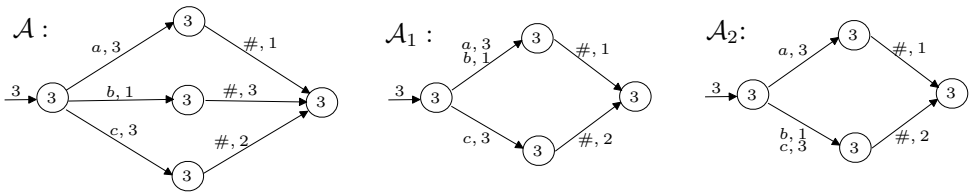

Fig. 6. An LDFA over a fully ordered lattice, with two different minimal LDFAs.

In this section we show that in spite of the non-canonicity, we are able to minimize such LDFAs in polynomial time. We describe a polynomial time algorithm that is given an LDFA $\mathcal{A} = \langle\mathcal{L},Q,\Sigma,\delta,Q_0,F\rangle$ over a fully ordered lattice, and returns an LDFA $\mathcal{A}_{min}$ with a minimal number of states, such that $L(\mathcal{A}_{min}) = L(\mathcal{A})$. The algorithm produces one of the possible minimal LDFAs.

Let $\mathcal{L} = \{0, 1, ..., n-1\}$ be the fully ordered lattice, let $L : \Sigma^* \to \mathcal{L}$ be the language of $\mathcal{A}$, and let $m = max_{w \in \Sigma^*} L(w)$; that is, m is the maximal value of a word in $L(\mathcal{A})$. Finally, let $q_0 \in Q$ be the single state with initial value that is not $\bot$. For each $1 \leq i \leq m$ we define a DFA $\mathcal{A}_i$ that accepts exactly all words w such that $L(w) \geq i$. Note that it is indeed enough to consider only the automata $\mathcal{A}_1, ..., \mathcal{A}_m$, as $\mathcal{A}_{m+1}, ..., \mathcal{A}_{n-1}$ are always empty and hence not needed, and $\mathcal{A}_0$ is not needed as well, as $L(\mathcal{A}_0) = \Sigma^*$.

For $1 \leq i \leq m$, we define $\mathcal{A}_i = \langle Q_i,\Sigma,\delta_i,q_0,F_i\rangle$ as follows:

- $Q_i \subseteq Q$ contains exactly all states that are both reachable from q_0 using transitions with value at least i, and also have some state with acceptance value at least i that is reachable from them using zero or more transitions with value at least i. Note that $q_0 \in Q_i$ for all i.
- δ_i contains all transitions such that their value in $\mathcal{A}$ is at least i and such that both their source and destination are in Q_i.
- $F_i \subseteq Q_i$ contains all states such that their acceptance value in $\mathcal{A}$ is at least i.

For readers that wonder why we do not define δ_i first, as these transitions with value at least i, and then define Q_i and F_i according to reachability along δ_i, note that such a definition would result in different automata that are not *trim*, as it may involve transitions that never lead to an accepting state in $\mathcal{A}_i$, and states that are equivalent to a rejecting sink. As we will see later, the fact that all the components in our $\mathcal{A}_i$ are not redundant is going to be important. Essentially, it has to do with the fact that the values of the transitions and states in the minimal LDFA are going to be influenced by their membership in δ_i and Q_i, respectively.

Note that for all $1 < i \leq m$, we have that $Q_i \subseteq Q_{i-1}$, $\delta_i \subseteq \delta_{i-1}$, and $F_i \subseteq F_{i-1}$. Also, it is not hard to see that $\mathcal{A}_i$ indeed accepts exactly all words w such that $L(w) \geq i$.

We now turn back to the given LDFA $\mathcal{A}$ and minimize it using $\mathcal{A}_1, ..., \mathcal{A}_m$. First, we apply a pre-processing on $\mathcal{A}$ that reduces the values appearing in $\mathcal{A}$ to be the minimal possible values (without changing the language)[4]. Formally, we define $\mathcal{A}' = \langle \mathcal{L}, Q, \Sigma, \delta', Q_0, F' \rangle$, where:

- For all $q, q' \in Q$ and $\sigma \in \Sigma$, we have that $\delta'(q, \sigma, q') = max\{i : (q, \sigma, q') \in \delta_i\}$.
- For all $q \in Q$, we have that $F'(q) = max\{i : q \in F_i\}$.

Note that since for all $1 < i \leq m$, we have that $\delta_i \subseteq \delta_{i-1}$ and $F_i \subseteq F_{i-1}$, then for all $1 \leq i \leq m$, we also have that $\delta'(q, \sigma, q') \geq i$ iff $(q, \sigma, q') \in \delta_i$ and $F'(q) \geq i$ iff $q \in F_i$.

Lemma 1. $L(\mathcal{A}) = L(\mathcal{A}')$.

Proof. Let $L' = L(\mathcal{A}')$. We prove that $L'(w) = L(w)$ for all words $w \in \Sigma^*$. Since we only reduced values, then clearly $L'(w) \leq L(w)$ for all $w \in \Sigma^*$. As for the other direction, let $w = \sigma_1 ... \sigma_k \in \Sigma^*$, and let $l = L(w)$. We prove that $L'(w) \geq l$. Let $r = q_0, q_1, ..., q_k$ be the run of $\mathcal{A}$ on w. Consider the values read along r, which are $Q_0(q_0), \delta(q_0, \sigma_1, q_1), \ldots, \delta(q_{k-1}, \sigma_k, q_k)$, and $F(q_k)$. Since $L(w) = l$, we know that all these values are at least l. Consider the automaton $\mathcal{A}_l$. By definition, the transitions in r belong to δ_l, and q_k belongs to F_l. Thus, the values $\delta'(q_0, \sigma_1, q_1), \ldots, \delta'(q_{k-1}, \sigma_k, q_k)$, and $F'(q_k)$ are all at least l. Together with the fact that the initial values in $\mathcal{A}$ and $\mathcal{A}'$ are the same, we get that $L'(w) \geq l$. □

By Lemma 1, it is enough to minimize $\mathcal{A}'$. We start with applying the algorithm for minimizing DFA on $\mathcal{A}_1, ..., \mathcal{A}_m$. As described in Section 2.2, each such application generates a partition of the states of $\mathcal{A}_i$ into equivalence classes.[5] Let us denote the equivalence classes produced for $\mathcal{A}_i$ by $\mathcal{H}_i = \{S_1^i, S_2^i, ..., S_{n_i}^i\}$.

Now, we construct from $\mathcal{A}'$ a minimal automaton $\mathcal{A}_{min} = \langle \mathcal{L}, Q_{min}, \Sigma, \delta_{min}, Q_{0_{min}}, F_{min} \rangle$ as follows.

- We obtain the set Q_{min} by partitioning the states of $\mathcal{A}'$ into sets, each inducing a state in Q_{min}. The partitioning process is iterative: we maintain a disjoint partition $\mathcal{P}_i$ of the states Q, starting with one set containing all states, and refining it along the iterations. The refinement at the i-th iteration is based on $\mathcal{H}_i$, and guarantees that the new partition $\mathcal{P}_i$ agrees with $\mathcal{H}_i$, meaning that states that are separated in $\mathcal{H}_i$ are separated in $\mathcal{P}_i$ as well. At the end of this process, the sets of the final partition constitute Q_{min}.
  More specifically, the algorithm has $m + 1$ iterations, starting with $i = 0$, ending with $i = m$. Let us denote the partition obtained at the i-th iteration by $\mathcal{P}_i = \{T_1^i, ..., T_{d_i}^i\}$. At the first iteration, for $i = 0$, we have that $d_0 = 1$, and $T_1^0 = Q$. At the i-th iteration, for $i > 0$, we are given the partition $\mathcal{P}_{i-1} = \{T_1^{i-1}, ..., T_{d_{i-1}}^{i-1}\}$, and generate $\mathcal{P}_i = \{T_1^i, ..., T_{d_i}^i\}$ as follows. For each $1 \leq j \leq d_{i-1}$, we examine T_j^{i-1} and partition it further. We do it in two stages. First, we examine $S_1^i, S_2^i, ..., S_{n_i}^i$ and for each $1 \leq k \leq n_i$ we compute the set $U_{j,k}^i = T_j^{i-1} \cap S_k^i$, and if $U_{j,k}^i \neq \emptyset$, then we add $U_{j,k}^i$ to $\mathcal{P}_i$. Thus, we indeed separate the states that are separated in $\mathcal{H}_i$. At the second stage, we consider the states in T_j^{i-1} that do not belong to $U_{j,k}^i$ for all k. Note that these states do not belong to Q_i, so $\mathcal{A}_i$ is indifferent about them. This is the stage where we have a choice in the algorithm. We choose an arbitrary k for which $U_{j,k}^i \neq \emptyset$, and

[4] A "weight-pushing process" is useful also in the minimization of weighted automata over the tropical semiring [19].

[5] Note that, by definition, all the states in Q_i have some accepting state reachable from them, so the fact we do not have a rejecting state is not problematic, as such a state would have constitute a singleton state in all the partitions we are going to consider.

add these states to $U^i_{j,k}$. If no such k exists, we know that no state in T^{i-1}_j appears in Q_i, so we have no reason to refine T^{i-1}_j, and we can add T^{i-1}_j to $\mathcal{P}_i$. Finally, we define Q_{min} to be the sets of $\mathcal{P}_m$.

- The transition relation δ_{min} is defined as follows. Consider a state $T \in Q_{min}$. We choose one state $q^T_{rep} \in T$ to be a *representative* of T, as follows. Let $i^T_{max} = max\{i$: there is $q \in T$ s.t. $q \in Q_i\}$, and let q^T_{rep} be a state in $T \cap Q_{i^T_{max}}$. Note that $T \cap Q_{i^T_{max}}$ may contain more than one state, in which case we can assume Q is ordered and take the minimal[6], to make it well defined. We now define the transitions leaving T according to the original transitions of q^T_{rep} in $\mathcal{A}'$. For $\sigma \in \Sigma$, let $q_{dest} \in Q$ be the σ-destination of q^T_{rep} in $\mathcal{A}'$. For all $T' \in Q_{min}$, if $q_{dest} \in T'$ we define $\delta_{min}(T, \sigma, T') = \delta'(q^T_{rep}, \sigma, q_{dest})$; otherwise, $\delta_{min}(T, \sigma, T') = 0$.
- For all $T \in Q_{min}$, if $q_0 \in T$, where q_0 is the initial state of $\mathcal{A}'$, we define $Q_{0_{min}}(T) = Q_0(q_0)$; otherwise, $Q_{0_{min}}(T) = 0$.
- For all $T \in Q$, we define $F_{min}(T) = F'(q^T_{rep})$.

Example 3. We describe below how the minimization algorithm proceeds on an example. Let us start with the LDFA $\mathcal{U}$ shown in Figure 7, over $\Sigma = \{a, b\}$ and $\mathcal{L} = \langle\{0, 1, 2\}, \leq\rangle$.

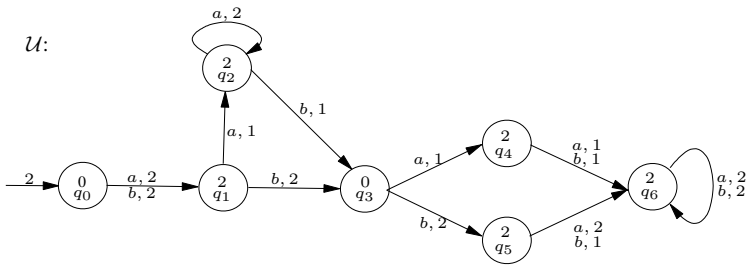

Fig. 7. An LDFA $\mathcal{U}$ to minimize.

The DFAs $\mathcal{U}_1$ and $\mathcal{U}_2$ induced by $\mathcal{U}$ are described in Figure 8. The dashed squares denote their partitions into equivalence classes according to the minimization algorithm for DFAs. Note, for example, that $q_2 \notin Q_2$, as q_2 is not reachable in $\mathcal{U}$ using transitions with values at least 2.

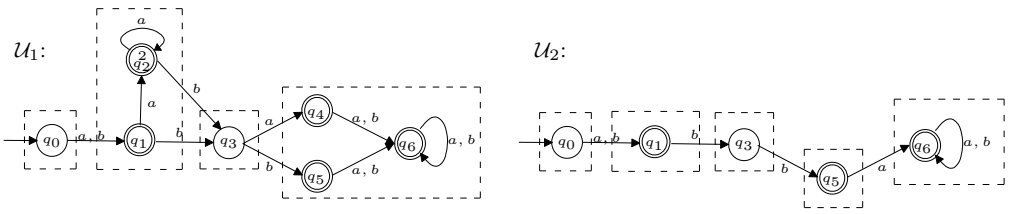

Fig. 8. $\mathcal{U}_1$ and $\mathcal{U}_2$ obtained from $\mathcal{U}$, with their partitions.

By using $\mathcal{U}_1$ and $\mathcal{U}_2$, we get $\mathcal{U}'$ as described in Figure 9. Note, for example, that the value of the a-transition from q_2 to itself has been reduced to 1.

[6] We could make here any arbitrary choice.

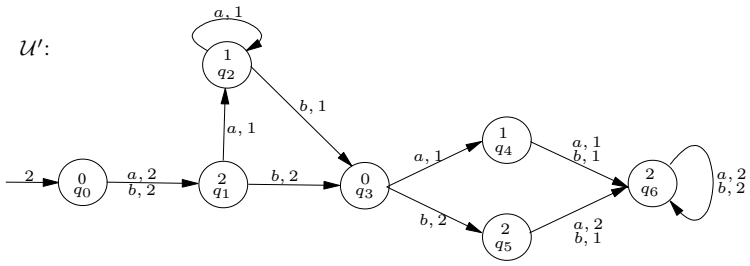

Fig. 9. $\mathcal{U}'$ obtained from $\mathcal{U}$

We now turn to obtain Q_{min}. We start with a partition $\mathcal{P}_0$ that contains all states. At the next iteration, $\mathcal{P}_0$ is partitioned according to the partition of $\mathcal{U}_1$, thus we have that $\mathcal{P}_1 = \{\{q_0\}, \{q_1, q_2\}, \{q_3\}, \{q_4, q_5, q_6\}\}$. At the last iteration, we partition $\mathcal{P}_1$ according to the partition of $\mathcal{U}_2$. Note, for example, that q_4 is not a member of Q_2. Accordingly, we can have q_4 in either the set of q_5 or the set of q_6, and we chose q_5. A similar choice could also be made for the state q_2 that is not a member of Q_2, but since it belongs to a set with only two states in the previous partition $\mathcal{P}_1$, we have no choice here and must have it in the set of q_1. We get $\mathcal{P}_2 = \{\{q_0\}, \{q_1, q_2\}, \{q_3\}, \{q_4, q_5\}, \{q_6\}\}$, as shown in Figure 10.

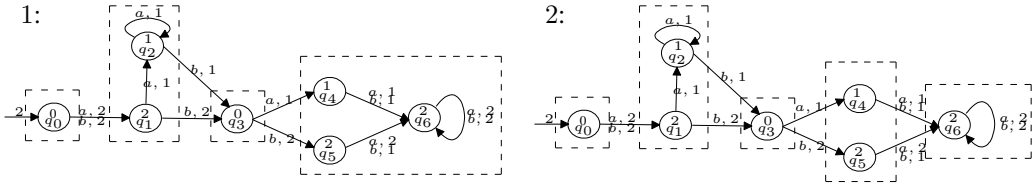

Fig. 10. The algorithm iterations.

The set Q_{min} is then $\mathcal{P}_2$, and the representatives of the sets in it are q_0, q_1, q_3, q_5, and q_6. Note that q_2 and q_4 cannot be representatives. The induced transitions are described in the minimal automaton $\mathcal{U}_{min}$ appearing in Figure 11.

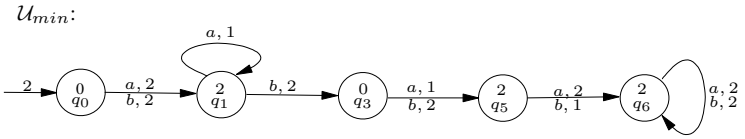

Fig. 11. A minimal automaton $\mathcal{U}_{min}$.

Back to the formal proof. Let $L_{min} = L(\mathcal{A}_{min})$ and $L' = L(\mathcal{A}')$. We now turn to prove that the construction is correct, that is, we prove that $L_{min} = L'$, $|\mathcal{A}_{min}|$ is minimal, and the time complexity of the construction of $\mathcal{A}_{min}$ is polynomial.

We first prove that $L_{min} = L'$. For $q, q' \in Q$, we say that $q \sim_i q'$ iff there exists some class $S \in \mathcal{H}_i$ such that $q, q' \in S$. Also, we say that $q \equiv_i q'$ iff for all $j \leq i$ it holds that $q \sim_j q'$. Note that although $\sim_i$ and $\equiv_i$ are equivalence relations over $Q_i \times Q_i$, they are not equivalence relations over $Q \times Q$, as they are not reflexive. Indeed, for $q \in Q \setminus Q_i$, there is no class $S \in \mathcal{H}_i$ such that $q \in S$, so $q \not\sim_i q$ and of course $q \not\equiv_i q$. However, it is easy to see that $\sim_i$ and $\equiv_i$ are both symmetric and transitive over $Q \times Q$.

Lemma 2 below explains the essence of the relation $\equiv_i$. As we shall prove, if $q \equiv_i q'$ then q and q' agree on the transition and acceptance values in $\mathcal{A}'$, if these values are less than i.

Lemma 2. *For $q, q' \in Q$, if $q \equiv_i q'$ then for all $j < i$, the following hold.*

- *For all $\sigma \in \Sigma$, we have $\delta'(q, \sigma, s) = j$ iff $\delta'(q', \sigma, s') = j$, where $s, s' \in Q$ are the σ-destinations of q, q' in $\mathcal{A}'$, respectively. .*
- *$F'(q) = j$ iff $F'(q') = j$.*

Proof. We first prove that $\delta'(q, \sigma, s) = j$ iff $\delta'(q', \sigma, s') = j$ for all $j < i$. Given $j < i$, it holds by definition that $q \sim_j q'$, as $q \equiv_i q'$ and $j \leq i$. Thus, there exists some class $S \in \mathcal{H}_j$ such that $q, q' \in S$, which means that $(q, \sigma, s) \in \delta_j$ iff $(q', \sigma, s') \in \delta_j$. Recall that $(q, \sigma, s) \in \delta_j$ iff $\delta'(q, \sigma, s) \geq j$ and that $(q', \sigma, s') \in \delta_j$ iff $\delta'(q', \sigma, s') \geq j$. Thus, we have that $\delta'(q, \sigma, s) \geq j$ iff $\delta'(q', \sigma, s') \geq j$. Also, since $j < i$ we know that $j + 1 \leq i$, so we can apply the same arguments on $j + 1$ as well, and get that $\delta'(q, \sigma, s) \geq j + 1$ iff $\delta'(q', \sigma, s') \geq j + 1$. We conclude by noticing that $\delta'(q, \sigma, s) = j$ iff $\delta'(q, \sigma, s) \geq j$ and $\delta'(q, \sigma, s) < j + 1$. The proof for F follows the same arguments. □

In the case of DFA, we know that each state of the minimal automaton for L corresponds to an equivalence class of $\sim_L$, and the minimization algorithm merges all the states of the DFA that correspond to each class into a single state. Consequently, the transition function of the minimal automaton can be defined according to one of the merged states, and the definition is independent of the state being chosen. In the case of our $\mathcal{A}_{min}$, things are more complicated, as states that are merged in $\mathcal{A}_{min}$ do not correspond to equivalence classes. We still were able, in the definition of the transitions and acceptance values, to chose a state q^T_{rep}, for each state T. Lemma 3 below explains why working with the chosen state is sound.

The lemma considers a word $w \in \Sigma^*$, and examines the connection between a state q_i in the run of $\mathcal{A}'$ on w and the corresponding state T_i in the run of $\mathcal{A}_{min}$ on w. It shows that if $L'(w) \geq l$ for some $l \in \mathcal{L}$, then $q_i \equiv_l q^{T_i}_{rep}$ for all i. Thus, the states along the run of $\mathcal{A}_{min}$ behave the same as the corresponding states in $\mathcal{A}'$ on values that are less than l, and may be different on values that are at least l, as long as they are both at least l. Intuitively, this is valid since after reaching a value l, we can replace all subsequent values $l' \geq l$ along the original run with any other value $l'' \geq l$.

Lemma 3. *Let $w = \sigma_1\sigma_2...\sigma_k$ be a word in Σ^*, and let $q_0, q_1, ..., q_k$ and $T_0, T_1, ..., T_k$ be the runs of $\mathcal{A}'$ and $\mathcal{A}_{min}$ on w respectively. For $l \in \mathcal{L}$, if $L'(w) \geq l$ then for all $0 \leq i \leq k$ it holds that $q_i \equiv_l q^{T_i}_{rep}$.*

Proof. We prove the lemma by an induction on i. Let $i = 0$. To show that $q_0 \equiv_l q^{T_0}_{rep}$, we have to show that for all $j \leq l$ it holds that $q_0 \sim_j q^{T_0}_{rep}$. Let $j \leq l$. We first show that $q_0, q^{T_0}_{rep} \in Q_j$. Since $Q_m \subseteq Q_j$, it is enough to show that $q_0, q^{T_0}_{rep} \in Q_m$. We already know that $q_0 \in Q_m$. Also, the fact that $q_0 \in T_0$ implies that $i^{T_0}_{max} = m$, and therefore $q^{T_0}_{rep} \in Q_m$ as well. It is left to show that q_0 and $q^{T_0}_{rep}$ belong to the same class in $\mathcal{H}_j$. Indeed, if this is not the case then they would have been separated in $\mathcal{P}_j$, which contradicts the fact that they are in the same set in Q_{min}.

We now assume correctness for $i-1$, and prove it for i. Let $q \in T_i$ be the σ_i-destination of $q_{rep}^{T_{i-1}}$ in $\mathcal{A}'$. We show that $q_i \equiv_l q$ and that $q \equiv_l q_{rep}^{T_i}$. By the transitivity of $\equiv_l$, this implies that $q_i \equiv_l q_{rep}^{T_i}$.

First, we show that $q_i \equiv_l q$. By the induction hypothesis we know that $q_{i-1} \equiv_l q_{rep}^{T_{i-1}}$. Assume by way of contradiction that $q_i \not\equiv_l q$. Thus, there exists some $j \leq l$ for which $q_i \not\sim_j q$. Consider $\mathcal{A}_j$, and look at the transitions (q_{i-1}, σ_i, q_i) and $(q_{rep}^{T_{i-1}}, \sigma_i, q)$. We claim that (q_{i-1}, σ_i, q_i), $(q_{rep}^{T_{i-1}}, \sigma_i, q) \in \delta_j$. Since $L'(w) \geq l$ we know that $\delta'(q_{i-1}, \sigma_i, q_i) \geq l$. Applying Lemma 2 on q_{i-1} and $q_{rep}^{T_{i-1}}$, we get that $\delta'(q_{rep}^{T_{i-1}}, \sigma_i, q) \geq l$ as well. Since $j \leq l$, we get that $\delta'(q_{i-1}, \sigma_i, q_i) \geq j$ and $\delta'(q_{rep}^{T_{i-1}}, \sigma_i, q) \geq j$, and thus $(q_{i-1}, \sigma_i, q_i), (q_{rep}^{T_{i-1}}, \sigma_i, q) \in \delta_j$. Recall that $q_i \not\sim_j q$. Since both (q_{i-1}, σ_i, q_i) and $(q_{rep}^{T_{i-1}}, \sigma_i, q)$ are in δ_j, we can now conclude that $q_{i-1} \not\sim_j q_{rep}^{T_{i-1}}$ as well, which means that $q_{i-1} \not\equiv_l q_{rep}^{T_{i-1}}$, and we have reached a contradiction.

It is left to show that $q \equiv_l q_{rep}^{T_i}$. By definition, we have to show that for all $j \leq l$ it holds that $q \sim_j q_{rep}^{T_i}$. Given $j \leq l$, we first show that $q, q_{rep}^{T_i} \in Q_j$. We already know that $q \in Q_l$, since $q_i \equiv_l q$, and conclude that $q \in Q_j$, since $Q_l \subseteq Q_j$. To show that $q_{rep}^{T_i} \in Q_j$ as well, recall that $q \in T_i$, and consider $i_{max}^{T_i}$ defined above. Since $q \in Q_j$ we get that $i_{max}^{T_i} \geq j$, so $Q_{i_{max}^{T_i}} \subseteq Q_j$. Also, by the definition of $q_{rep}^{T_i}$ we have that $q_{rep}^{T_i} \in Q_{i_{max}^{T_i}}$, so we get that $q_{rep}^{T_i} \in Q_j$. Now, we show that $q, q_{rep}^{T_i}$ are in the same class in $\mathcal{H}_j$. Indeed, if they are separated in $\mathcal{H}_j$, then they must be separated in $\mathcal{P}_j$ as well, which is a contradiction, since they are in the same set in Q_{min}. □

Based on the above, we now turn to prove that $L_{min} = L'$. Let $w \in \Sigma^*$, and let $l = L'(w)$. We show that $L_{min}(w) = l$. Let $r' = q_0, q_1, ..., q_k$ and $r_{min} = T_0, T_1, ..., T_k$ be the runs of $\mathcal{A}'$ and $\mathcal{A}_{min}$ on w respectively.

We first show that $L_{min}(w) \geq l$. Consider the values read along r', which are $Q_0(q_0)$, $\delta'(q_0, \sigma_1, q_1)$, ..., $\delta'(q_{k-1}, \sigma_k, q_k)$, and $F'(q_k)$. Since $L'(w) = l$ we know that all these values are at least l. By Lemma 3 we get that for all $0 \leq i \leq k$ it holds that $q_i \equiv_l q_{rep}^{T_i}$. Then by applying the first part of Lemma 2 on $q_0, ..., q_{k-1}$ and the second part on q_k, we get that the values $\delta'(q_{rep}^{T_0}, \sigma_1, s_0)$, ..., $\delta_{min}(q_{rep}^{T_{k-1}}, \sigma_k, s_{k-1})$ and $F_{min}(q_{rep}^{T_k})$ are all at least l, where s_i is the σ_i-destination of $q_{rep}^{T_i}$ for all $0 \leq i < k$. Thus, we get that $\delta_{min}(T_0, \sigma_1, T_1), \ldots, \delta_{min}(T_{k-1}, \sigma_k, T_k)$ and $F_{min}(T_k)$ are all at least l as well, since these values are defined according to $q_{rep}^{T_0}, ..., q_{rep}^{T_k}$. Together with the fact that the initial value remains the same in $\mathcal{A}_{min}$, we get that $L_{min}(w) \geq l$.

In order to prove that $L_{min}(w) \leq l$, we show that at least one of the values read along r_{min} is l. Since $L'(w) = l$, at least one of the values read along r' is l. If this value is $Q_0(q_0)$ then we are done, since by definition $Q_0(T_0) = Q_0(q_0)$. Otherwise, it must be one of the values $\delta'(q_0, \sigma_1, q_1), \ldots, \delta'(q_{k-1}, \sigma_k, q_k)$ or $F'(q_k)$. Let q_d be the state from which the value l is read for the first time along r', either as a transition value ($d < k$) or as an acceptance value ($d = k$). We claim that $q_d \in Q_{l+1}$ (note that if $l = m$, then clearly $L_{min}(w) \leq l$, thus, we assume that $l < m$, so Q_{l+1} is well defined). If $d = 0$, then we are done, since $q_0 \in Q_{l+1}$ by definition. Otherwise, we look at the transition (q_{d-1}, σ_d, q_d). By the definition of q_d, we know that $\delta'(q_{d-1}, \sigma_d, q_d) \geq l+1$, and by the definition of δ' it then follows that $(q_{d-1}, \sigma_d, q_d) \in \delta_{l+1}$. Thus, we get that $q_d \in Q_{l+1}$. Now, by the definition of Q_{l+1}, there exists some state $q_{acc}^{l+1} \in Q$ with acceptance value at least $l+1$ that is reachable from q_d in $\mathcal{A}$ using zero or more transitions with value at least $l+1$. Let w' be the word read along these transitions from q_d to q_{acc}^{l+1}, and let $w'' = \sigma_1 ... \sigma_d \cdot w'$. It is easy to see that $L'(w'') \geq l+1$. Thus, we can apply Lemma 3, and get that $q_d \equiv_{l+1} q_{rep}^{T_d}$. Then, by applying Lemma 2 on q_d and $q_{rep}^{T_d}$ we conclude that the value l is read from T_d along r_{min}, and we are done.

We now turn to prove that $|\mathcal{A}_{min}|$ is minimal. Let $N = |\mathcal{A}_{min}|$. We describe below N different words $w_1, ..., w_N \in \Sigma^*$, and prove that for all $i \neq j$ the words w_i and w_j cannot reach the same

state in an LDFA for L. Clearly, this implies that an LDFA for L must contain at least N states, so $|\mathcal{A}_{min}|$ is minimal.

We define the words $w_1, ..., w_N$ as follows. Let $T_1, ..., T_N$ be the states of $\mathcal{A}_{min}$, and let $q_{rep}^{T_1}, ..., q_{rep}^{T_N}$ be their representatives respectively. We go back to the original automaton $\mathcal{A}$, and for each such representative q, we define the following (with the expected extension of δ to words, thus $\delta(q, w, q')$ is the value of traversing w from q to q'):

- $reach(q) = \{w \in \Sigma^* : \delta(q_0, w, q) > 0\}$, where q_0 is the initial state of $\mathcal{A}$.
- $maxval(q) = max\{\delta(q_0, w, q) : w \in reach(q)\}$. Note that $maxval(q)$ considers only the traversal values of the words reaching q.
- $maxw(q)$ is $w \in \Sigma^*$ for which $\delta(q_0, w, q) = maxval(q)$. Note that there may be several such words, so we can take the lowest one by lexicographic order, to make it well defined.

For all $1 \leq i \leq N$, we now define $w_i = maxw(q_{rep}^{T_i})$. Note that these words are indeed different, as they reach different states in the deterministic automaton $\mathcal{A}$. Consider two different indices i and j. We prove that the words $maxw(q_{rep}^{T_i})$ and $maxw(q_{rep}^{T_j})$ cannot reach the same state in an LDFA for L. Consider the states $q_{rep}^{T_i}$ and $q_{rep}^{T_j}$. These states belong to different sets in Q_{min}. Let $1 \leq l \leq m$ be the index of the iteration in which they were first separated.

We use the following lemmas:

Lemma 4. *The states $q_{rep}^{T_i}$ and $q_{rep}^{T_j}$ belong to different classes in $\mathcal{H}_l$.*

Proof. Before proving the lemma, let us note that it does not follow directly from the fact that $q_{rep}^{T_i}$ and $q_{rep}^{T_j}$ were first separated at the l-th iteration, as they could also be separated in the case where one of them does not belong to Q_l. So we start with proving that $q_{rep}^{T_i}, q_{rep}^{T_j} \in Q_l$, and together with the fact that $q_{rep}^{T_i}$ and $q_{rep}^{T_j}$ were separated at the l-th iteration, we get that they must belong to different classes in $\mathcal{H}_l$.

We denote by T^{l-1} the set that contains both $q_{rep}^{T_i}$ and $q_{rep}^{T_j}$ at the $(l-1)$-th iteration. At the l-th iteration, we compute the sets $T^{l-1} \cap S_k^l$ for $1 \leq k \leq n_l$. Since $q_{rep}^{T_i}$ and $q_{rep}^{T_j}$ were separated at this iteration, at least two of these sets are not empty. Let us denote these non-empty sets by T_i^l and T_j^l, with $q_{rep}^{T_i} \in T_i^l$ and $q_{rep}^{T_j} \in T_j^l$. Since T_i^l and T_j^l are obtained by intersection with $\mathcal{H}_l$, there must be two states $q_i, q_j \in Q_l$ such that $q_i \in T_i^l$ and $q_j \in T_j^l$. We show that $q_{rep}^{T_i} \in Q_l$. The proof that $q_{rep}^{T_j} \in Q_l$ follows the same arguments. Consider $i_{max}^{T_i}$. Since $q_{rep}^{T_i} \in Q_{i_{max}^{T_i}}$, it is enough to show that $i_{max}^{T_i} \geq l$, as it implies that $Q_{i_{max}^{T_i}} \subseteq Q_l$ and thus $q_{rep}^{T_i} \in Q_l$. To show that $i_{max}^{T_i} \geq l$, we show that there is a state $q \in T_i \cap Q_l$. To show that, we prove that in all iterations $l' \geq l$, the set containing the state $q_{rep}^{T_i}$ also contains some state of Q_l. The proof proceeds by an induction on l'. For the case $l' = l$, we know that, at the l-th iteration, the set T_i^l that contains $q_{rep}^{T_i}$ also contains the state $q_i \in Q_l$. Now, let $l' > l$ be some subsequent iteration, and let $T_i^{l'-1}$ be the set containing $q_{rep}^{T_i}$ at the $(l'-1)$-th iteration. If $T_i^{l'-1}$ is not separated at the l'-th iteration, then, by the induction hypothesis, we are done. Otherwise, it must be that any newly created set contains a state $q \in Q_{l'}$, as these sets are obtained by intersections with $\mathcal{H}_{l'}$. Then, since $Q_{l'} \subseteq Q_l$, we are done. Applying this claim to the m-th iteration, we conclude that there is a state $q \in T_i \cap Q_l$, and thus $q_{rep}^{T_i} \in Q_l$. □

Lemma 5. *$tr\_val(maxw(q_{rep}^{T_i})) \geq l$ and $tr\_val(maxw(q_{rep}^{T_j})) \geq l$ in all LDFA for L.*

Proof. We prove the lemma for $maxw(q_{rep}^{T_i})$, and the proof for $maxw(q_{rep}^{T_j})$ follows the same arguments. By Lemma 4 we know that $q_{rep}^{T_i} \in Q_l$, so by definition $q_{rep}^{T_i}$ is reachable in $\mathcal{A}$ from the initial state using transitions with value at least l. Thus, $tr\_val(maxw(q_{rep}^{T_i})) \geq l$ in $\mathcal{A}$. Also, by definition there is some state $q \in Q$ with acceptance value at least l that is reachable in $\mathcal{A}$ from $q_{rep}^{T_i}$ using zero or more transitions with value at least l. Let $w \in \Sigma^*$ be the word read along these transitions. Consider the word $maxw(q_{rep}^{T_i}) \cdot w$, it is now easy to see that $L(\mathcal{A})(maxw(q_{rep}^{T_i}) \cdot w)) \geq l$. Now, assume by way of contradiction that $tr\_val(maxw(q_{rep}^{T_i})) < l$ in some LDFA $\mathcal{A}'$ for L. Then, since $\mathcal{A}'$ is deterministic it follows that $L(\mathcal{A}')(maxw(q_{rep}^{T_i}) \cdot w)) < l$, which is a contradiction. □

Based on the above, we prove that the words $maxw(q_{rep}^{T_i})$ and $maxw(q_{rep}^{T_j})$ cannot reach the same state in an LDFA for L. By Lemma 4, there is a distinguishing tail $z \in \Sigma^*$. That is, without loss of generality, z is read in $\mathcal{A}$ from $q_{rep}^{T_i}$ with value at least l, and is read from $q_{rep}^{T_j}$ with value less than l. Let us examine the words $maxw(q_{rep}^{T_i}) \cdot z$ and $maxw(q_{rep}^{T_j}) \cdot z$. By applying Lemma 5 on $\mathcal{A}$, we get that $L(\mathcal{A})(maxw(q_{rep}^{T_i}) \cdot z) \geq l$ and that $L(\mathcal{A})(maxw(q_{rep}^{T_j}) \cdot z) < l$. Now, let $\mathcal{U}$ be an LDFA for L, and assume by way of contradiction that $maxw(q_{rep}^{T_i})$ and $maxw(q_{rep}^{T_j})$ are reaching the same state in $\mathcal{U}$. Let q be that state. Applying Lemma 5 on $\mathcal{U}$, we get that both $maxw(q_{rep}^{T_i})$ and $maxw(q_{rep}^{T_j})$ are reaching q with traversal value at least l. Now, let us examine the value v_z with which z is read from q. If $v_z \geq l$, then $L(\mathcal{U})(maxw(q_{rep}^{T_j}) \cdot z) \geq l$, which contradicts the fact that $L(\mathcal{A})(maxw(q_{rep}^{T_j}) \cdot z) < l$. On the other hand, if $v_z < l$, then $L(\mathcal{U})(maxw(q_{rep}^{T_i}) \cdot z) < l$, which contradicts the fact that $L(\mathcal{A})(maxw(q_{rep}^{T_i}) \cdot z) \geq l$.

Thus, we conclude that $|\mathcal{A}_{min}|$ is minimal.

It is left to prove that the time complexity of the algorithm is polynomial. Recall that the algorithm has three stages: constructing the automata $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m$ and minimizing them, generating $\mathcal{A}'$ from $\mathcal{A}$, and constructing $\mathcal{A}_{min}$ from $\mathcal{A}'$. We analyze the running time of each stage.

In the first stage we construct $O(|\mathcal{L}|)$ automata and minimize them. The complexity of constructing and minimizing $\mathcal{A}_i$ is $O(|Q| \log |Q| + |\delta|)$ for all i, as follows. To get Q_i, we have to get rid of all states that are not reachable from the initial state using transitions with value at least i, or have no state with acceptance value at least i that is reachable from them using zero or more transitions with value at least i. We start with $\mathcal{A}'_i$ in which $Q'_i = Q$, the transitions δ'_i are all transitions with value at least i, and F'_i are all states with acceptance value at least i. It takes $O(|Q| + |\delta|)$ to construct $\mathcal{A}'_i$, since defining δ'_i and F'_i requires one pass on the edges and the states, and Q'_i is already defined to be Q. We then apply BFS to $\mathcal{A}'_i$, starting in the initial state, and omit from Q'_i all unreachable states. This takes $O(|Q| + |\delta|)$. Then we apply the algorithm for minimizing DFA using the Myhill-Nerode theorem, and omit all those states that are equivalent to a rejecting sink. This can be done in $O(|Q| \log(|Q|))$ [9]. The states that remain are exactly the states of Q_i. Also, we get the partition of Q_i according to the Myhill-Nerode theorem, that is, we minimize $\mathcal{A}_i$. All in all, we get time complexity of $O(|\mathcal{L}|(|Q| \log |Q| + |\delta|))$ for this part.

The time complexity of the second stage is $O(|\mathcal{L}| \times (|Q| + |\delta|))$, as defining δ' and F' requires one pass over all edges and states, performing at most $|\mathcal{L}|$ checks for each edge and state.

As for the third stage, note that in order to construct Q_{min}, we defined, for all $0 \leq i \leq m$, a function $f_i : Q \to \{1...d_i\}$, such that $f_i(q) = j$ iff $q \in T_j^i$. Also, we defined $g_i : Q \to \{1...n_i\} \cup \{\#\}$, such that for $q \in Q_i$ we have $g_i(q) = k$ iff $q \in S_k^i$, and for $q \notin Q_i$ we have $g_i(q) = \#$. In the above notations, we can get Q_{min} by finding f_m. We show that for all $1 \leq i \leq m$, we can obtain f_i from f_{i-1} and g_i in time $O(|Q| \log |Q|)$. We first compute $f_{i-1} \cdot g_i : Q \to \{1...d_{i-1}\} \times (\{1...n_i\} \cup \{\#\})$, that is, $f_{i-1} \cdot g_i(q) = \langle f_{i-1}(q), g_i(q) \rangle$. Assuming random access to f_{i-1} and g_i, this can be done in $O(|Q|)$. For all $1 \leq j \leq d_{i-1}$, we then consider

the states that are mapped by $f_{i-1} \cdot g_i$ to $\langle j, \# \rangle$. We arbitrarily choose some value k_j for which $\langle j, k_j \rangle \in image(f_{i-1} \cdot g_i)$, and change the mapping of these states to be $\langle j, k_j \rangle$. If no such k_j exists, we leave their mapping as is. This can be done in time $O(|Q| \log |Q|)$, by first sorting the states according to their values in $f_{i-1} \cdot g_i$. We denote the new mapping by $f_{i-1} \cdot g'_i$. Finally, we order the finite set $image(f_{i-1} \cdot g'_i)$, and define $f_i(q)$ to be the index of $f_{i-1} \cdot g'_i(q)$ in that order. Note that $d_i = |image(f_{i-1} \cdot g'_i)|$. It also takes time $O(|Q|)$. Altogether, constructing Q_{min} takes time $O(|\mathcal{L}| \times |Q| \log |Q|)$. Defining δ_{min} and F_{min} requires to find q^T_{rep} for all $T \in Q_{min}$. We can define in time $O(|\mathcal{L}| \times |Q|)$ a mapping $f : Q \to \{1...m\}$ such that $f(q) = i$ iff i is the maximal index for which $q \in \mathcal{A}_i$. Finding q^T_{rep} now takes time $O(|T|)$, so finding q^T_{rep} for all $T \in Q_{min}$ takes time $O(|Q|)$. After q^T_{rep} is found, it takes time $O(|\delta|)$ to define δ_{min}, and time $O(|Q|)$ to define F_{min}. Thus, the time complexity of this stage is $O(|\mathcal{L}| \times |Q| \log |Q| + |\delta|)$.

Altogether, we end up with a polynomial time complexity of $O(|\mathcal{L}|(|Q| \log |Q| + |\delta|))$.

We can now conclude with the following:

Theorem 3. *An LDFA over a fully ordered lattice can be minimized in polynomial time.*

References

1. R. Alur, A. Kanade, and G. Weiss. Ranking automata and games for prioritized requirements. In *Proc. 20th CAV*, LNCS 5123, pages 240-253, 2008.
2. ESF Network programme. Automata: from mathematics to applications (AutoMathA). http://www.esf.org/index.php?id=1789, 2010.
3. G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *Proc. 11th CAV*, pages 274–287, 1999.
4. G. Bruns and P. Godefroid. Temporal logic query checking. In *Proc. 16th LICS*, pages 409–420. IEEE Computer Society, 2001.
5. W. Chan. Temporal-logic queries. In *Proc. 12th CAV*, LNCS 1855, pages 450–463, Springer, 2000.
6. M. Chechik, B. Devereux, and A. Gurfinkel. Model-checking infinite state-space systems with fine-grained abstractions using SPIN. In *Proc. 8th SPIN*, LNCS 2057, pages 16–36. Springer, 2001.
7. M. Droste, W. Kuich, and H. Vogler (eds.). *Handbook of Weighted Automata*. Springer, 2009.
8. 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.
9. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. In Addison-Wesley, 1979.
10. S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In *Proc. 9th CAV*, LNCS 1254, pages 72–83, 1997.
11. T.A. Henzinger. From boolean to quantitative notions of correctness. In *Proc. 37th POPL*, pages 157–158, 2010.
12. A. Hussain and M. Huth. On model checking multiple hybrid views. Technical Report TR-2004-6, University of Cyprus, 2004.
13. R.M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.
14. O. Kupferman and Y. Lustig. Lattice automata. In *Proc. 8th VMCAI*, LNCS 4349 of *Lecture Notes in Computer Science*, pages 199 – 213, 2007.
15. O. Kupferman and Y. Lustig. Latticed simulation relations and games. *International Journal on the Foundations of Computer Science*, 21(2):167–189, 2010.
16. D. Kirsten and I. Mäurer. On the determinization of weighted automata. *Journal of Automata, Languages and Combinatorics*, 10(2/3):287–312, 2005.
17. D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Journal of Algebra and Computation*, 4:405–425, 1994.
18. Y. Li and W. Pedrycz. Minimization of lattice finite automata and its application to the decomposition of lattice languages. *Fuzzy Sets and Systems*, 158(13):1423–1436, 2007.

19. M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
20. D.S. Malik, J.N. Mordeson, and M.K. Sen. Minimization of fuzzy finite automata. *Information Sciences* 113: 323–330. Elsevier, 1999.
21. J. Myhill. Finite automata and the representation of events. Technical Report WADD TR-57-624, pages 112–137, Wright Patterson AFB, Ohio, 1957.
22. A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.
23. L. Zekai and S. Lan. Minimization of lattice automata. In *Proc. 2nd ICFIE*, pages 194–205, 2007.
24. J. Eisner. Simpler and More General Minimization for Weighted Finite-State Automata. *HLT-NAACL*, 2003.