On the Capacity of Capacitated Automata

Orna Kupferman[1] and Sarai Sheinvald[2]

[1] School of Computer Science and Engineering, The Hebrew University, Israel
[2] Dept. of Software Engineering, Ort Braude Academic College, Israel

Abstract. *Capacitated automata* (CAs) have been recently introduced in [8] as a variant of finite-state automata in which each transition is associated with a (possibly infinite) *capacity*. The capacity bounds the number of times the transition may be traversed in a single run.
The study in [8] includes preliminary results about the expressive power of CAs, their succinctness, and the complexity of basic decision problems for them. We continue the investigation of the theoretical properties of CAs and solve problems that have been left open in [8]. In particular, we show that union and intersection of CAs involve an exponential blow-up and that their determinization involves a doubly-exponential blow up. This blow-up is carried over to complementation and to the complexity of the universality and containment problems, which we show to be EXPSPACE-complete. On the positive side, capacities do not increase the complexity when used in the deterministic setting. Also, the containment problem for nondeterministic CAs is PSPACE-complete when capacities are used only in the left-hand side automaton. Our results suggest that while the succinctness of CAs leads to a corresponding increase in the complexity of some of their decision problems, there are also settings in which succinctness comes at no price.

1 Introduction

Finite automata (FAs) are used in the modeling and design of finite-state systems and their behaviors, with applications in engineering, databases, linguistics, biology, and many more. The traditional definition of an automaton does not refer to its transitions as consumable resources. Indeed, a run of an automaton is a sequence of successive transitions, and there is no bound whatsoever on the number of times that a transition may be traversed. In practice, the use of a transition may correspond to the use of some resource. For example, it may be associated with filling a buffer, consumption of bandwidth, or a usage of some energy-consuming machine. In [8], the authors introduced *capacitated automata* (CAs) – a variant of finite automata in which each transition is associated with a (possibly infinite) *capacity*, which limits the number of times the transition may be traversed in a single run.

The study in [8] examines CAs from two points of view. The first views CAs as recognizers of formal languages. The interesting questions that arise in this view are the classical questions about automata: their expressive power, succinctness, closure properties, determinization, etc. The second view, more related to traditional resource-allocation problems, views CAs as labeled flow networks. The interesting questions

then have to do with maximum utilization of the system modeled by CA and can be viewed as a generalization of the max-flow problem in networks [6].

Our work here continues the study of the first approach taken in [8]. There, the authors study the expressive power of CAs, their succinctness with respect to FA, and the basic decision problems of non-emptiness and membership. The study in [8] also relates CAs with extensions of FAs of similar flavor. For completeness, we briefly repeat the main findings here. For a full description, see [8]. Augmenting the transitions of automata by numerical values is common in the quantitative setting, for example in probabilistic automata and weighted automata [5, 10]. There, the values are used for modeling probabilities, costs, rewards, certainty, and many more. The semantics of these models is multi-valued. CAs, on the other hand, maintain the Boolean nature of regular languages and only augment the way in which acceptance is defined. In this family of extensions, we can find *Parikh automata* [7], whose semantics involves counting of the number of occurrences of each letter in the word, and their variants, in particular the *constrained automata* of [3]. The expressive power of Parikh automata and their variants goes beyond regular languages, and the type of questions studied for them is different than these studied for CAs. Additional strictly more expressive models include *multiple counters automata* [4], where transitions can be taken only if guards referring to traversals so far are satisfied, and *queue-content decision diagrams*, which are used to represent queue content of FIFO-channel systems [1, 2]. Finally, a model with the same name – *finite capacity automata* – is used in [9] for modeling the control of an automated manufacturing system. This model is different from the CAs studied here and is more related to Petri nets.

In order to describe the results in [8], we first need some definitions. Let Δ be the set of transitions of a CA $\mathcal{A}$ and let $c : \Delta \to \mathbb{N} \cup \{\infty\}$ be its capacity function. That is, for every transition $\tau \in \Delta$, a run of $\mathcal{A}$ can traverse τ at most $c(\tau)$ times. A naive translation of $\mathcal{A}$ to an FA involves a blow-up that is polynomial in the number γ of "capacity configurations". Formally, $\gamma = \Pi_{\tau \in \Delta : c(\tau) \neq \infty}(c(\tau) + 1)$. That is, for every transition τ with a bounded capacity, the FA has to remember how many more times τ can be traversed. It is not hard to see that by attributing the states of $\mathcal{A}$ by the current capacity configuration, we can obtain an equivalent FA. Note that γ is exponential in the number of transitions with finite capacities. The above implies that CAs are not more expressive than FAs, but may be exponentially more succinct. Indeed, as shown in [8], the blow-up in γ may not be avoided in some cases. Consider, for example, the language $L_{n,m}$ over the alphabet $\Sigma_n = \{1, \ldots, n\}$ that contains exactly all words in which each letter in Σ_n appears at most m times. It is not hard to see that a traditional, possibly nondeterministic, automaton for $L_{n,m}$ needs at least m^n states. On the other hand, a deterministic CA for $L_{n,m}$ consists of a single state with n self loops, each labeled with a different letter and having capacity m.

A key question in the theory of CAs is the role that γ plays in constructions and decision problems. For example, it is shown in [8] that while determinization of nondeterministic CAs (NCAs) does involve a blow up in γ, the non-emptiness problem for NCAs can be solved in linear time and its complexity is independent of γ. The study of succinctness and complexity in [8] is preliminary. In particular, even for determinization, only an exponential lower bound has been shown, with no matching upper bound,

leaving open also the blow-up involved in other basic constructions like union, intersection, and complementation. Decision problems whose solution makes use of these constructions, most notably universality and containment, have been left open too.

We solve the problems left open in [8]. Our news is mainly bad: the exponential lower bound for determinization that is proven there is not tight, and determinization of NCAs actually involves a doubly-exponential blow-up. This is an interesting phenomenon, implying that the power of nondeterminism in the capacitated model goes beyond the subset construction. Even though the deterministic automaton we end up with is capacitated, it sometimes cannot make use of its capacities and has to maintain not only sets of states but also sets of capacity configurations. The doubly-exponential blow-up in determinization is carried over to complementation of NCAs. Also there, even though the complementing NCA is both nondeterministic and capacitated, it may be doubly-exponentially bigger, as an exponential blow up in γ cannot be avoided. Moreover, even the constructions of union and intersection involve a blow-up in γ. Essentially, it follows from the inability to merge the capacity functions of the underlying CAs to a single capacity function in the product CA. In fact, the blow-up applies even when one of the automata is not capacitated. An exception is union of NCAs, which can be defined by putting the NCAs "side by side", and thus involves no blow up.

We continue to the decision problems of universality (given a CA $\mathcal{A}$, decide whether $L(\mathcal{A}) = \Sigma^*$) and containment (given two CAs $\mathcal{A}$ and $\mathcal{A}'$, decide whether $L(\mathcal{A}) \subseteq L(\mathcal{A}')$) and show that the bad news is carried over also to their complexity, but only in the nondeterministic model: While the universality problem for DCAs is NLOGSPACE-complete, thus is not more complex than the problem for DFAs, it is EXPSPACE-complete for NCAs. The EXPSPACE complexity of universality immediately implies an EXPSPACE lower bound also for containment, in fact already containment of DFAs in NCAs. We study the various cases according to whether each of $\mathcal{A}$ and $\mathcal{A}'$ is capacitated and/or nondeterministic. Here, we are also able to come up good news: the containment problem for NCAs in NFAs is PSPACE-complete, thus is not harder than containment for NFAs. We conclude that while the succinctness of CAs often leads to a corresponding increase in the complexity of their decision problems, capacities come at no price when they model systems or when used in a deterministic automaton.

Due to the lack of space, some of the proofs are omitted from this version and can be found in the full version, in the authors' URLs.

2 Preliminaries

A *nondeterministic finite automaton* (NFA, for short) is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \Delta, F \rangle$, where Σ is a finite alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation, and $F \subseteq Q$ is a set of final states. Given a word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_l$, a *run* of $\mathcal{A}$ on w is a sequence r of successive transitions in Δ that reads w and starts in a transition from the set of initial states. Thus, $r = \langle q_0, \sigma_1, q_1 \rangle, \langle q_1, \sigma_2, q_2 \rangle, \ldots, \langle q_{l-1}, \sigma_l, q_l \rangle$, for $q_0 \in Q_0$. The run is accepting if $q_l \in F$. The NFA $\mathcal{A}$ *accepts* the word w iff it has an accepting run on it. Otherwise, $\mathcal{A}$ *rejects* w. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts. If $|Q_0| = 1$ and for all $q \in Q$ and $\sigma \in \Sigma$ there is at most one $q' \in Q$ with $\Delta(q, \sigma, q')$, then $\mathcal{A}$ is

deterministic. Note that a deterministic finite automaton (DFA) has at most one run on each word.

A *nondeterministic capacitated automaton* (NCA, for short) is an NFA in which each transition has a *capacity*, bounding the number of times it may be traversed. A transition may not be bounded, in which case its capacity is ∞. Let $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$ and $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. Formally, an NCA is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \Delta, F, c \rangle$, where Σ, Q, Q_0, Δ, and F are as in NFAs and $c : \Delta \to \mathbb{N}^\infty$ is a *capacity function* that maps each transition in Δ to its capacity. A legal run of $\mathcal{A}$ is defined as for NFA, with the additional condition that the number of occurrences of each transition $e \in \Delta$ is at most $c(e)$. When the underlying NFA is deterministic, then so is $\mathcal{A}$. The *width* of an NCA $\mathcal{A}$ is the maximal finite capacity that c assigns to a transition in Δ.

For a capacity function $c : \Delta \to \mathbb{N}^\infty$, let $c_\downarrow$ be the set of capacity functions obtained by closing c downwards. Formally, a function $c' : \Delta \to \mathbb{N}^\infty$ is in $c_\downarrow$ if for all transitions $e \in \Delta$ with $c(e) = \infty$, we have $c'(e) = \infty$, and for all transitions $e \in \Delta$ with $c(e) \in \mathbb{N}$, we have $0 \leq c'(e) \leq c(e)$. It is easy to see that the size of $c_\downarrow$, denoted $|c_\downarrow|$, is $\Pi_{e:c(e)\in\mathbb{N}^+}(c(e)+1)$. Thus, $|c_\downarrow|$ is exponential in the number of transitions with finite capacities and is bounded by $w^{|\Delta|}$, where w is the width of $\mathcal{A}$ plus 1.

Example 1. For a word $w \in \{0,1\}^*$, let $\tilde{w}$ be the word obtained by flipping all the letters in w. For example, if $w = 0010$, then $\tilde{w} = 1101$. We refer to $\tilde{w}$ as the *negative* of w.

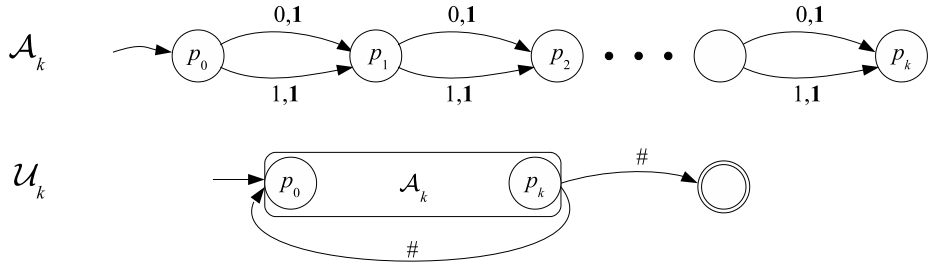

Fig. 1. Then NCAs $\mathcal{A}_k$ and $\mathcal{U}_k$, with $L(\mathcal{U}_k) = \{v\# : v \in (0+1)^k\} \cup \{v\#\tilde{v}\# : v \in (0+1)^k\}$.

We use the NCA $\mathcal{A}_k$ appearing in Figure 1 as a "box" in some of our constructions. As a warm-up example, consider the NCA $\mathcal{U}_k$ appearing in Figure 1. It is easy to see that $\mathcal{U}_k$ accepts only words in $(0+1)^k\#+(0+1)^k\#(0+1)^k\#$. Moreover, since the capacity of all the transitions labeled 0 or 1 is 1, the only way to traverse the sequence of states $p_0, \ldots, p_k$ twice is by reading a word $v \in (0+1)^k$ during the first traversal, then $\#$, and then the negative $\tilde{v}$ of v. Hence $L(\mathcal{U}_k) = \{v\# : v \in (0+1)^k\} \cup \{v\#\tilde{v}\# : v \in (0+1)^k\}$. Note that an NFA for $L(\mathcal{U}_k)$ requires at least 2^k states.

3 Closure Constructions

Since CAs are as expressive as FAs, the closure of regular languages under Boolean operations implies a similar closure for NCAs and DCAs. In this section we study the blow up that is involved in the corresponding constructions. We start with NCA determinization. In [8], the authors show an exponential blow-up for determinization. We tighten this result and show that determinization is tightly doubly exponential.

Theorem 1. *Determinization of NCAs is tightly doubly-exponential.*

Proof: For the upper bound, consider an NCA $\mathcal{A} = \langle \Sigma, Q, Q_0, \Delta, F, c\rangle$. By [8], $\mathcal{A}$ has an equivalent NFA $\mathcal{A}'$ with $|Q| \cdot |c_\downarrow|$ states, which is exponential in the size of $\mathcal{A}$. By determinizing $\mathcal{A}'$, we get an equivalent DFA that is exponential in $\mathcal{A}'$ and hence doubly-exponential in $\mathcal{A}$.

For the lower bound, we define a sequence of languages $L_1, L_2, \ldots$ over the alphabet $\Sigma = \{0, 1, \#, \$\}$ as follows. We define $L^1 = (0+1+\#)^*$, and for $n \geq 1$, we define $L^2_n = \{(0+1+\#)^*\#w\#(0+1+\#)^*\$(0+1+\#)^*\#\tilde{w}\#(0+1+\#)^* : w \in (0+1)^n\}$. That is, each word in L^2_n is of the form $u\$v$, for words $u, v \in (0+1+\#)^*$. Inside u, there should be some word $w \in (0+1)^n$ between two $\#$'s, and v includes $\tilde{w}$ between $\#$'s. Now, $L_n = L^1 \cup L^2_n$.

The language L_n can be recognized by an NCA $\mathcal{U}_n$ with $O(n)$ states. The NCA $\mathcal{U}_n$, described in Figure 3, guesses when the $\#$ before w starts, then reads w and traverses the NCA $\mathcal{A}_n$ from Example 1 once. Reading the $\#$ after w, it guesses whether the input word belongs to L^1, in which case it goes with $\#$ to a state that accepts all words in $(0+1+\#)^*$, or belongs to L^2_n, in which case it moves to a state that waits for a \$ and then waits for the $\#$ before $\tilde{w}$, with which it returns to the initial state of $\mathcal{A}_n$.

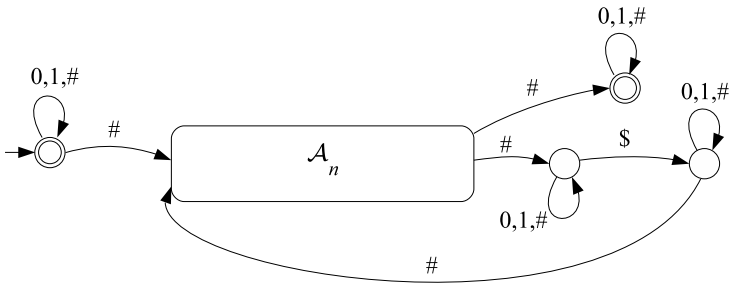

Fig. 2. the NCA $\mathcal{U}_n$.

A DFA for L_n must have at least 2^{2^n} states. Indeed, in order to recognize words in L^2_n, upon reaching the \$, the DFA must remember the subset of words in $(0+1)^n$ that have appeared between $\#$'s before the \$.

The same arguments imply that the language $S_n = \Sigma^* \cdot L_n$ can be recognized by an NCA with $O(n)$ states but requires a DFA with at least 2^{2^n} states. As shown in [8],

a minimal DCA for a language of the form $\Sigma^* \cdot L$ is not smaller than a minimal DFA for it. It follows that a minimal DCA for S_n requires 2^{2^n} states, hence the lower bound for determinization. □

We continue to study complementation. For DFAs, complementation is easy, as one only has to dualize the set of accepting states. In the case of DCAs, we also need to dualize the upper bound on the number of traversals of transitions that is imposed by capacities. Such a dualization amounts to adding lower bounds on the number of traversals, which is not supported in CAs. As we show, this makes complementation of DCAs exponential. When, however, we complement the DCA into an NCA, we can "implement" the lower bound with a polynomial blow-up. Starting, however, with an NCA, a doubly-exponential blow-up in complementation cannot be avoided.

Theorem 2. *Complementing a DCA to a DCA is tightly exponential. Complementing a DCA to an NCA is polynomial.*

Proof: Consider a DCA $\mathcal{A}$. By translating $\mathcal{A}$ to a DFA and complementing the latter, we obtain a complementing DCA (in fact, DFA) of exponential size. For the lower bound, consider the language $L_{m,n}$ described in Section 1. A DCA for $L_{n,m}$ consists of a single state with n self loops, one for each letter, with capacity m. Let $\tilde{L}_{n,m}$ be the complement of $L_{n,m}$. That is, $\tilde{L}_{n,m}$ contains all words in which at least one letter appears more than m times. Note that $\tilde{L}_{n,m} = \Sigma_n^* \cdot \tilde{L}_{n,m}$. Hence, by [8], a minimal DCA for $\tilde{L}_{n,m}$ is not smaller than a minimal DFA for it, which needs at least m^n states.

Now, let $\mathcal{A} = \langle \Sigma, Q, Q_0, \Delta, F, c \rangle$. Intuitively, the complementing NCA $\mathcal{A}'$ has $|\Delta|$ components, one for each transition $\tau \in \Delta$. The component $\mathcal{A}_\tau$ accepts exactly all words w such that the run of $\mathcal{A}$ on w traverses τ more than $c(\tau)$ times or ends in a rejecting state. Thus, $\mathcal{A}_\tau$ has $c(\tau)$ copies of $\mathcal{A}$, starts from the first copy and moves to the next copy whenever τ is traversed. When τ is traversed in the $c(\tau)$-th copy, the component moves to an accepting sink. In all copies, the states in $Q \setminus F$ are accepting. Note that we end up with an NFA (in fact, the only nondeterminism is in choosing the component) that consists of $\sum_{\tau \in \Delta} c(\tau)$ copies of $\mathcal{A}$. For the lower bound, observe that an NCA for the language $\tilde{L}_{n,m}$ above requires $O(mn)$ states. □

Theorem 3. *Complementation of NCAs is tightly doubly-exponential.*

Proof: The upper bound follows from the doubly-exponential DFA that the determinization in Theorem 1 results in, which can be complemented with no blow-up.

For the lower bound, consider the sequence of languages $L_1, L_2, \ldots$ defined in the proof of Theorem 1. An NCA for the complement $\tilde{L}_n = \Sigma^* \setminus L_n$ needs at least 2^{2^n} states. Upon reaching the \$, the NCA must remember the subset of words in $(0+1)^n$ that have appeared between #'s before the \$. Formally, let $\tilde{\mathcal{U}}_n$ be an NCA for $\tilde{L}_n$. For a word $u \in (0+1+\#)^*$, let S_u be the set of words $w \in (0+1)^n$ in which w appears in u between #'s, and let $C_u \in 2^{(0+1+\#)^*}$ be the set of words u' such that a word $x \in (0+1)^n$ is in $S_{u'}$ iff $\tilde{x}$ is not in S_u. For example, if $n = 3$ and $u = 0\#000\#10\#010\#110\#$, then $S_u = \{000, 010, 110\}$. Thus, $(0+1)^3 \setminus S_u = \{001, 011, 100, 101, 111\}$, so a possible $u' \in C_u$ is $\#110\#100\#011\#010\#000\#$.

Indeed, $S_{u'} = \{110, 100, 011, 010, 000\}$, which contains exactly all the negatives of words not in S_u.

It is easy to see that for all $u \in (0+1+\#)^*$, a word $u\$u'$ such that $u' \in C_u$ is not in L_n and thus should be accepted by $\tilde{\mathcal{U}}_n$. Let Q_u be the set of states that all accepting runs of $\mathcal{U}_n$ over $u\$u'$ reach after reading the \$. In the full version, we prove that if $u_1, u_2 \in (0+1+\#)^n$ are such that $S_{u_1} \neq S_{u_2}$, then it must be that $Q_{u_1} \cap Q_{u_2} = \emptyset$. □

We now turn to intersection of NCAs. We show that a blow-up in the capacity factor of the underlying automata cannot be avoided. Moreover, it applies even when we construct an NCA for the intersection of a DCA and a DFA.

Theorem 4. *Intersection of NCAs and DCAs is tightly exponential.*

Proof: For the upper bound, given two NCAs (or DCAs), we can construct their intersection by first removing capacities, obtaining exponentially bigger NFAs (or DFAs), and then constructing the product of the latter. Note that this leads to a DFA or an NFA with no capacities.

For the lower bound, we define two sequences of languages $L_1^1, L_2^1, L_3^1, \ldots$ and $L_1^2, L_2^2, L_3^2, \ldots$ such that for all $n \geq 1$, the language L_n^1 can be recognized by a DCA with one state, the language L_n^2 can be recognized by a DFA with $n+1$ states, and the minimal NCA for their intersection requires at least 2^n states.

Let $n \geq 1$, and let $\Sigma_n = \{1, \ldots, n\}$. We define L_n^1 as the set of all words over Σ_n in which every letter $i \in \Sigma_n$ occurs at most once. We define L_n^2 to be the set of all words over Σ_n whose length is n. It is easy to see that indeed L_n^1 can be recognized by a DCA with one state (it has a self-loop transition with capacity 1 for each letter in Σ_n), and L_n^2 can be recognized by a DFA with $n+1$ states. We prove that an NCA for the language $L_n = L_n^1 \cap L_n^2$ needs at least 2^n states. Note that L_n includes exactly all words that form a permutation of Σ_n.

For a set $S \subseteq \Sigma_n$, let w_S be the word obtained by concatenating the letters in S in ascending order, and let $\bar{S} = \Sigma_n \setminus S$. Note that $w_S \cdot w_{\bar{S}} \in L_n$

Consider an NCA $\mathcal{A}$ for L_n. Assume by way of contradiction that $\mathcal{A}$ has less than 2^n states. Then, there are two sets $S, T \subseteq \Sigma_n$ such that $S \neq T$ and there is a state q that is visited by both an accepting run r of $\mathcal{A}$ on $w_S \cdot w_{\bar{S}}$ after it reads w_S and by an accepting run r' of $\mathcal{A}$ on $w_T \cdot w_{\bar{T}}$ after it reads w_T. Recall that r and r' are sequences of transitions. Let $r_S \cdot r_{\bar{S}}$ be a partition of r to the parts that traverse w_S and $w_{\bar{S}}$, respectively, and similarly for $r_T \cdot r_{\bar{T}}$ and r'.

In the full version, we show that the existence of two sets S and T, and a state q as above leads to a contradiction. □

The union of two NFAs can be easily constructed in linear time by putting the two NFAs "side by side". The product construction can be used to construct a DFA for the union of two DFAs in polynomial time. We now show that while the union of two NCAs can be constructed linearly, the construction of a DCA for the union of two DCAs involves an exponential blow up.

Theorem 5. *Union of NCAs is linear. Union of DCAs is tightly exponential.*

Proof: The union of two NCAs can be constructed linearly, similarly to the union construction for NFAs.

We proceed to DCAs. For the upper bound, given two DCAs, we can construct their union by removing capacities, obtaining exponentially bigger DFAs, and then constructing the product of the latter. As in the case of intersection, this construction ends up with a DFA with no capacities. As we show in the lower bound below, however, a blow-up in the capacity factor of the underlying automata cannot be avoided. Moreover, it applies even when we construct a DCA for the union of a DCA and a DFA.

Consider the two sequences of languages $L_1^1, L_2^1, L_3^1, \ldots$ and $L_1^2, L_2^2, L_3^2, \ldots$ used in the proof of Theorem 4. We prove that a DCA for the language $L_n = L_n^1 \cup L_n^2$ needs at least 2^{n-2} states.

Consider a DCA $\mathcal{A}$ for L_n. Assume by way of contradiction that $\mathcal{A}$ has less than 2^{n-2} states. Then, there are two sets $S, T \subseteq \Sigma_n$ of size at most $n-2$ such that $S \neq T$, and there is a state q such that both the run of $\mathcal{A}$ on w_S and the run of $\mathcal{A}$ on w_T reach q. Assume w.l.o.g. that there is a letter $i \in S \backslash T$. Consider the transition $\tau = \langle q, i, q' \rangle$. That is, when $\mathcal{A}$ is in state q and reads the letter i, it moves to state q'. Clearly, $w_T \cdot i \in L_n$, so q' is accepting. Also, since $|S| \neq n-1$, then $w_S \cdot i \notin L_n$. Hence, either q' is not accepting, and we have reached a contradiction, or τ was taken $c(\tau)$ times while w_S is read. Then, however, as $|S| \leq n-2$, the run of $\mathcal{A}$ on $w_S \cdot i^{n-|S|}$ has to traverse τ more than $c(\tau)$ times, and is thus rejecting. But $w_S \cdot i^{n-|S|}$ is of length n, a contradiction. $\square$

Note that in all lower-bound proofs we have used NCAs and DCAs of width 1.

4 Decision procedures

The *universality problem* is to decide, given an automaton $\mathcal{A}$ and an alphabet Σ, whether $L(\mathcal{A}) = \Sigma^*$. The problem is known to be NLOGSPACE-complete for DFAs, and PSPACE-complete for NFAs. The *containment problem* is to decide, given two automata $\mathcal{A}_1$ and $\mathcal{A}_2$, whether $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. The problem is known to be PSPACE-complete when $\mathcal{A}_2$ is an NFA and NLOGSPACE-complete when $\mathcal{A}_2$ is a DFA.

In this section we study the universality and containment problems for the different classes of CA. We start with the universality problem. In the case of FA, the solution involves a construction of a complementing automaton and checking its emptiness. By [8], the emptiness problem for NCAs is NLOGSPACE complete, as it is for NFAs. Complementation of CAs, however, is expensive (and in fact results in FAs). We show that the blow-up in complementation is carried over to the complexity of the universality problem. Still, in the deterministic setting it is possible to reason directly on the structure of the DCA and circumvent the construction of a complementing automaton.

Theorem 6. *The universality problem for DCA is NLOGSPACE-complete.*

Proof: Since DFAs, for which universality is known to be NLOGSPACE-hard, are a special case of DCAs, the lower bound is immediate.

For the upper bound, we claim that a DCA $\mathcal{D} = \langle \Sigma, Q, q_0, \Delta, F, c \rangle$ is not universal iff there exists a path from q_0 to a state $q \notin F$, or to a state q from which there is no transition labeled σ for some $\sigma \in \Sigma$, or a path from q_0 to a transition $\langle q, a, q' \rangle$ that is

part of a cycle and such that $c(\langle q, a, q' \rangle) = k$ for some $k \in \mathbb{N}$. In the first two cases, a path to a rejecting state or to a state from which some letter cannot be read induces a word that is rejected by $\mathcal{D}$. In the third case, a path to a cycle on which $\langle q, a, q' \rangle$ occurs induces a word that gets stuck after at most k traversals through the cycle. Checking all cases can be performed in NLOGSPACE by guessing the path to a rejecting state in the former case, or a path to a finitely capacitated transition $\langle q, a, q' \rangle$ followed by a path from q' to q in the latter case. □

Theorem 7. *The universality problem for NCAs is EXPSPACE-complete.*

Proof: An NCA can be translated to a DFA with a doubly-exponential blow-up. The upper bound then follows from the NLOGSPACE complexity for universality of DFAs.

For the lower bound, we show a reduction from an exponent version of the *tiling problem*, defined as follows. We are given a finite set T of tiles, two relations $V \subseteq T \times T$ and $H \subseteq T \times T$, an initial tile t_0, a final tile t_f, and a bound $n > 0$. We have to decide whether there is some $m > 0$ and a tiling of a $2^n \times m$-grid such that (1) The tile t_0 is in the bottom left corner and the tile t_f is in the top left corner, (2) A horizontal condition: every pair of horizontal neighbors is in H, and (3) A vertical condition: every pair of vertical neighbors is in V. Formally, we have to decide whether there exist $m \in \mathbb{N}$ and a function $f : \{0, \ldots, 2^n - 1\} \times \{0, \ldots, m - 1\} \to T$ such that (1) $f(0, 0) = t_0$ and $f(0, m - 1) = t_f$, (2) For every $0 \leq i \leq 2^n - 2$ and $0 \leq j \leq m - 1$, we have that $(f(i, j), f(i + 1, j)) \in H$, and (3) For every $0 \leq i \leq 2^n - 1$ and $0 \leq j \leq m - 2$, we have that $(f(i, j), f(i, j + 1)) \in V$. When n is given in unary, the problem is known to be EXPSPACE-complete.

We encode a tiling as a word over $T \cup \{0, 1, \#, \$\}$, consisting of a sequence of rows (each row is of length 2^n). Each row is a sequence of tile blocks, where each tile block contains the tile and its index in the row, as an n-bit counter. Each row starts with $\#$, and the entire grid ends with $\#\$$. Such a word represents a proper tiling if it starts with t_0, has a last row that starts with t_f, every pair of adjacent tiles in a row are in H, and every pair of tiles that are 2^n tiles apart are in V.

We reduce the tiling problem to the universality problem for NCAs. Given a tiling problem $\rho = \langle T, H, V, t_0, t_f, n \rangle$, we construct an NCA $\mathcal{A}_\rho$ that accepts a word w iff w is not an encoding of a legal tiling for ρ. Thus, $\mathcal{A}_\rho$ rejects a word w iff w encodes a legal tiling ρ, and is universal iff no such tiling exists.

The difficulty lies in relating vertical neighbors – tiles that are 2^n far apart. Detecting a violation of the vertical condition amounts to the existence of two tiles t and t' such that $(t, t') \notin V$, and there is a row with tile t in position i, such that the tile in position i in the successive row is t'. Recall the NCA $\mathcal{A}_k$ from Example 1. In order to use it for detecting when the position i repeats in the successive row, we maintain both the counter and its negative. That is, each tile block is of length $1 + 2n$, and consists of the tile, its index i in the row, and the negative $\tilde{i}$. Then, the property we need in order to find a violation of the condition V becomes "there is a row with tile t in position i, such that the tile whose negative position is $\tilde{i}$ in the successive row is t'".

We define $\mathcal{A}_\rho$ as the union of several NCAs, each guessing a different type of violation of an encoding of a legal tiling for ρ. We begin with a violation of the pattern and counters.

– The pattern is not of the form $(\# \cdot (T \cdot (0+1)^{2n})^*)^*\#\$$.
– It is not the case that every $\#$ precedes $\$$ or a block with counter value 0
– It is not the case that $\#$ follows all blocks with counter value $2^n - 1$.
– It is not the case that in every tile block the last n bits are the negation of the n bits before them.
– The counters are not increased properly. We check this separately for even and odd counter values. For even values, there exists a block whose counter value i ends with 0 and either the first $n-1$ bits of the counter i' in the next block do not agree with the first $n-1$ bits of i, or the last bit of i' is not 1. For odd counters, there exists a block with counter value i that ends with $0(1^k)$ for some $0 < k < n$ such that the first $n-k-1$ bits of the counter i' in the next block do not agree with the first $n-k-1$ bits of i, or the last $k+1$ bits in i' are not $1(0^k)$.

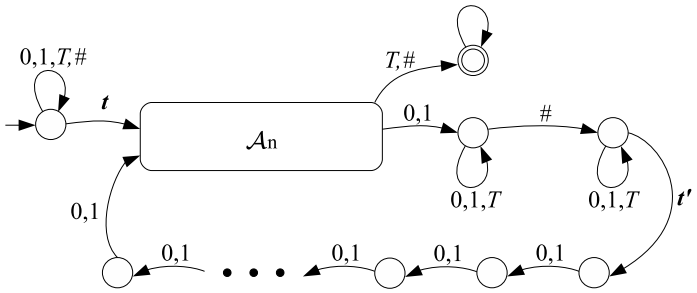

Fig. 3. the NCA $\mathcal{A}^V_{(t,t')}$.

Secondly, we have violations of the tiling conditions:

– The tiling does not start with t_0.
– The first tile in the last row is not t_f. That is, there is a row that starts with a tile that is different from t_f and the row ends with $\#\$$.
– For a violation of the horizontal condition, namely that there is $0 \leq i \leq 2^n - 2$ and $0 \leq j \leq m-1$ for which $(f(i,j), f(i+1,j)) \notin H$, we use the union of at most $|T|^2$ NFAs, one for each pair $(t,t') \notin H$. The NFA $\mathcal{A}^H_{(t,t')}$ accepts a word if it has a block with tile t such that the next block in the row (that is, the next letter in T in the word) is t'. Formally, $L(\mathcal{A}^H_{(t,t')}) = \Sigma^* t (0+1)^{2n} t' \Sigma^*$.
– For a violation of the vertical condition, namely that there is $0 \leq i \leq 2^n - 1$ and $0 \leq j \leq m-2$ for which $(f(i,j), f(i,j+1)) \notin V$, we use a union of NCAs, one for each pair $(t,t') \notin V$.
The NCA $\mathcal{A}^V_{(t,t')}$ accepts a word if it has a tile block with t and the tile block with the same counter value in the next row is t'. To check that the counters are identical, we check that the negation in the next counter is the negation of the current counter. This can be achieved by using a component similar to $\mathcal{A}_k$ of Example 1, with $k = n$. As described in Figure 4, the NCA $\mathcal{A}^V_{(t,t')}$ guesses a tile block with t and

the counter traverses $\mathcal{A}_n$. After a single occurence of $\#$, it guesses a tile block with t' and the negation of the counter in the tile block traverses $\mathcal{A}_n$ again. If the counter of the two tile blocks is identical, the second traversal of $\mathcal{A}_n$ is successful, and we have indeed identified a violation. Thus, $\mathcal{A}^V_{(t,t')}$ now accepts the rest of the word. Notice that $\mathcal{A}_n$ must be traversed twice in order to accept, as the first letter after the first traversal is the first bit of the negative, causing the run to take the cycle that leads back to $\mathcal{A}_n$.

□

Theorem 8. *The containment problem of a DFA or an NFA in a DCA is NLOGSPACE-complete*

Proof: The lower bound directly follows from the NLOGSPACE-hardness of the containment problem for DFAs.

For the upper bound, intuitively, we can guess a run on a word w such that $w \in L(A_1)$ and $w \notin L(A_2)$. We do so by running simultaneously, one transition at a time, on $\mathcal{A}_1$ and on the complement of $\mathcal{A}_2$, on the fly. It holds that $w \notin L(A_2)$ iff we reach a rejecting state, or traverse some finitely capacitated transition too many times. For the latter case, we can guess this transition in advance, and track the number of times it is traversed. The full details are in the full version. □

Theorem 9. *The containment problem of a DCA or an NCA in a DCA or a DFA is co-NP-complete.*

Proof: For the lower bound, we reduce from the Hamiltonian cycle problem for directed graphs. Given a directed graph $G = \langle V, E\rangle$, we construct a DCA $\mathcal{A}_1$ and a DFA $\mathcal{A}_2$ as follows. The DFA $\mathcal{A}_2$ accepts all words over V whose length is at most $2n-1$. The DCA $\mathcal{A}_1$ is the DCA described in Theorem 6 in [8], which accepts a word of length $2n$ iff G has a Hamiltonian cycle, and in any case, accepts only words whose length is at most $2n$. Therefore, we have that $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ iff G does not have a Hamiltonian cycle.

For the upper bound, it can be shown that if $L(\mathcal{A}_1) \not\subseteq L(\mathcal{A}_2)$, then there exists a witness word w such that $w \in L(\mathcal{A}_1)$ and $w \notin L(\mathcal{A}_2)$, and whose length is polynomial in the sizes of $\mathcal{A}_1, \mathcal{A}_2$, and the maximal finite capacity in $\mathcal{A}_2$.

Indeed, let $\mathcal{A}$ be the product of $\mathcal{A}_1$ and $\mathcal{A}_2$ which ignores the capacities of $\mathcal{A}_1$. A witness word w is either induced by a path to a state in $\mathcal{A}$ that is rejecting in $\mathcal{A}_1$ and accepting in $\mathcal{A}_2$, or by a path to a state in $\mathcal{A}$ that is accepting in $\mathcal{A}_1$, and traverses some transition in $\mathcal{A}_2$ too many times. In both cases, the length of the path can be polynomially bounded. The full details are in the full version. □

Theorem 10. *The containment problem of a DCA or an NCA in an NFA is PSPACE-complete.*

Proof: The lower bound follows from the containment problem of a DFA in an NFA.

For the upper bound, intuitively, we can guess a run on a word w such that $w \in L(A_1)$ and $w \notin L(A_2)$. We do so by running simultaneously, one transition at a time,

on $\mathcal{A}_1$ and on the complement of $\mathcal{A}_2$. To do so in PSPACE, we run on the latter on-the-fly. The full details are in the full version. □

Theorem 11. *The containment problem of a DFA, an NFA, a DCA, or an NCA in an NCA is EXPSPACE-complete.*

Proof: The upper bound follows from the doubly-exponential translation of NCA to DFA. Since Σ^* can be recognized by a one-state DFA, the lower bound follows from Theorem 7. □

5 Conclusion and Future Work

We have shown that while the succinctness that capacities offer often comes with a corresponding increase in the complexity of decision problems for them, there are some cases where it comes for free. In the future, we plan to study the idea of capacities in the formalism of temporal logic. The analogue of a capacitated transition would be *bounding operators* in the logic, which bound the number of prefixes along which a sub-specification may hold. Such operators can conveniently and succinctly bound the number of occurrences of events in a computation. We believe that the classical translation of LTL into automata [11] can be generalized so that LTL with bounded operators are translated to CAs.

References

1. B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using qdds. In *CAV 1996*, volume 1102 of *LNCS*, pages 1–12.
2. A. Bouajjani, P. Habermehl, and T. Vojnar. Verification of parametric concurrent systems with prioritised FIFO resource management. *Formal Methods in System Design*, 32(2):129–172, 2008.
3. M. Cadilhac, A. Finkel, and P. McKenzie. On the expressiveness of parikh automata and related models. In *NCMA 2011*, pages 103–119.
4. H. Comon and Y. Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In *CAV 1998*, volume 1427 of *LNCS*, pages 268–279.
5. M. Droste, W. Kuich, and H. Vogler (eds.). *Handbook of Weighted Automata*. Springer, 2009.
6. L.R. Ford and D.R. Fulkerson. *Flows in networks*. Princeton Univ. Press, Princeton, 1962.
7. F. Klaedtke and H. Rueß. Monadic second-order logics with cardinalities. In *ICALP 2003*, volume 2719 of *LNCS*, pages 681–696.
8. O. Kupferman and T. Tamir. Properties and utilization of capacitated automata. In *FSTTCS 2014*, volume 29 of *LIPIcs*, pages 33–44.
9. R.G. Qiu and S.B. Joshi. Deterministic finite capacity automata: a solution to reduce the complexity of modeling and control of automated manufacturing systems. In *CACSD 1996*, pages 218 –223.
10. M. O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.
11. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.