Weak $\omega$-Automata

Shaked Flur
Weak $\omega$-Automata

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Shaked Flur

Submitted to the Senate of
the Technion — Israel Institute of Technology
Adar 5773 Haifa February 2013
The research thesis was done under the supervision of Prof. Orna Grumberg in the Computer Science Department.

I would like to thank my advisors Prof. Orna Grumberg and Prof. Orna Kupferman for guiding and inspiring me. I would also like to thank Prof. Rom Pinchasi for helping me solve some problems in Combinatoric. Finally, I would like to thank my girlfriend for her encouragement and love. In times when I could not see the end she was there to support and believe in me.

The generous financial support of the Technion is gratefully acknowledged.
Contents

List of Figures ii

List of Tables iii

Abstract 1

Abbreviations and Notations 2

1 Introduction 3
   1.1 Background ......................................................... 3

2 Preliminaries 10
   2.1 Automata over infinite words ....................................... 10
   2.2 Observations ....................................................... 13
   2.3 Classes of automata ............................................... 15
   2.4 Duality ............................................................ 16

3 ABW to AWW 18
   3.1 Upper bound for the translation from ABW to AWW .................. 18
   3.2 Lower bound for the translation from ABW to AWW .................. 21
   3.3 Lower bound for AWWX ............................................ 26
   3.4 Lower bound for $L_n^k$ ........................................... 29
   3.5 Restricted AWW ................................................... 30

4 DBW to DWW 32
   4.1 The DWW hierarchy ................................................. 32
   4.2 DBW is DWW[1]-type ............................................. 35

5 Conclusion 40

References 41
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Drawing alternating automaton example. $A = \langle \Sigma, Q, q_{init}, \delta, \alpha \rangle$ where $\Sigma = [2], Q = [2], q_{init} = 0, \delta(0, 0) = 0 \land 1, \delta(0, 1) = 0, \delta(1, 0) = 1$ and $\delta(1, 1) = \top$.</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>The NBW automaton defining $L_n$. Unlabeled edges can be taken with any $\sigma \in \Sigma$. There is an unlabeled edge from $\langle 0, 0 \rangle$ to $\langle i, 1 \rangle$ for any $i \in [n]$. There is an edge labeled with $\langle i, j \rangle$ from $\langle i, 1 \rangle$ to $\langle j, 2 \rangle$ for any $i, j \in [n]$. There is an unlabeled edge from $\langle i, 1 \rangle$ to $\langle i, 1 \rangle$ and from $\langle i, 2 \rangle$ to $\langle i, 1 \rangle$ for any $i \in [n]$.</td>
<td>23</td>
</tr>
<tr>
<td>4.1</td>
<td>DWW[4] automaton from the proof of Theorem 9.</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>Cardinal example for Lemma 7.</td>
<td>37</td>
</tr>
</tbody>
</table>
List of Tables

1.1 Typeness for deterministic automata. .............................................. 5
1.2 Typeness for nondeterministic automata. ........................................ 5
Abstract

Automata over infinite words are a core concept in program verification. Many different types of automata exist and they can be classified by the type of their transition relation and by the type of their acceptance condition. In this thesis we explore weak automata over infinite words. In weak automata the state space is partitioned into partially ordered sets. The transition relation is then restricted so that a state from one set can only move to states in partitions lower in the order. More over, the states in each partition are either all accepting states, or all non-accepting states. These restriction give weak automata a structure which makes reasoning about easier. Hence the translation from different kinds of automata to weak automata is of interest. We show that when a translation from deterministic Büchi automata to deterministic weak automata with $k$ partitions is possible, it involves only very simple modifications to the original automaton. In particular, the state space and the transition function are untouched. More over, we show that restricting the number of partitions in deterministic weak automata also restricts their expression power. This comes as a sharp contrast to the translation of alternating Büchi automata to alternating weak automata where translation is always possible but involves a quadratic blow up in the state space. We present such quadratic translation and provide a proof that it can not be improved to less then $n \log n$ blow up. More over, we discuss some leads on how to show the quadratic blow up is necessary.
Abbreviations and Notations

\([n], [n]^{\text{even}}, [n]^{\text{odd}}\) — the set of natural numbers \(\{0, 1, \ldots, n - 1\}\), the subset of even numbers and the subset of odd numbers, respectively.

\(w = w_0w_1 \ldots w_n, w = w_0w_1 \ldots\) — a finite and infinite words, respectively, with letters \(w_0, w_1, \ldots \in \Sigma\).

\(|w|\) — the length of the finite word \(w\).

\(\epsilon\) — the empty word.

\(\Sigma^*\) — the set of all finite words over alphabet \(\Sigma\).

\(\Sigma^+\) — the set of all finite words over alphabet \(\Sigma\) except the empty word.

\(B^+(X)\) — the set of Boolean formulas over \(X\).

DBW, NBW, UBW, ABW — the set of det’/nondet’/universal/alternating Büchi automata.

DCW, NCW, UCW, ACW — the set of det’/nondet’/universal/alternating co-Büchi automata.

DWW, NWW, UWW, AWW — the set of det’/nondet’/universal/alternating weak automata.
Chapter 1

Introduction

1.1 Background

The definition of a run of finite state automata on finite words can be naturally extended to infinite words. A run on a finite word is a finite sequence of states and a run on an infinite word is an infinite sequence of states. Extending the definition of finite word acceptance to infinite word is more involved and several different definitions exist. In the early 1960's, Büchi [5] and Rabin [27] gave different definitions and used automata on infinite objects as a mean of solving decision problems in mathematics and logic. In particular, Büchi showed the set of languages recognized by Büchi automata are exactly the set of \( \omega \)-regular languages. Then, by showing that Büchi automata are closed under complementation, he managed to prove that \( \omega \)-regular languages are also closed under complementation. The construction Büchi gave for the complementing automaton is doubly exponential, i.e., asymptotically the number of states in the resulting automaton is doubly exponential in terms of the number of states of the original automaton.

Muller [24] used his own definition of acceptance to describe problems in asynchronous switching theory. McNaughton [20] showed that the set of languages accepted by Muller's automata is identical to the set of \( \omega \)-regular languages and therefore identical to the set of languages accepted by Büchi's automata. In later years, Streett [31] introduced an accepting condition that is the dual to Rabin's, and Mostowski [23] introduced the parity condition which can be viewed as a special case of Rabin and Streett.

Muller et al. [26] introduced the weak acceptance condition to characterize the weak monadic theory. The states of weak automata are given as partially ordered sets of states. Each of the sets is classified either as an accepting set or as a non-accepting set. From each state, the transition function of weak automata can only move to states in sets no higher in the order.

As automata on finite words, the transition function of automata on infinite words can be deterministic or nondeterministic. Given a state and a letter, a deterministic
automaton has one successor state for which it has to move to. More formally, the range of the transition function of a deterministic automaton is the set of the automaton states. Given a state and a letter, a nondeterministic automaton has one set of states, from which it has to choose one successor state and then move to it. That is, the range of the transition function of a nondeterministic automaton is the power set of the set of the automaton states. Different choices of successor states will produce different runs, as will be described shortly, and all the possible choices have to be considered. Universal automata are the dual of nondeterministic automata, given a state and a letter, a universal automaton has one set and it has to move simultaneously to all the states in that set.

For a given automaton, we associate with each infinite word a set of runs. The runs of deterministic and nondeterministic automata are sequences of states. The first state of a run is always the initial state of the automaton. The second state of the run is given by applying the transition function to the first state and the first letter of the word. For deterministic automata this gives exactly one possible state. For nondeterministic automata this gives a set of possible states (from which one is chosen). In general, the $i+1$ state of the run is given by applying the transition function to the $i$'th state and the $i$'th letter of the word (for deterministic automata this gives exactly one state and for nondeterministic automata this gives a set of possible states). For deterministic automata, since the initial state is fixed and each successor state has only a single possible state, there is only one run associated with each word. For nondeterministic automata, since each successor state has (possibly) multiple possible states, each word is associated with multiple runs, corresponding to different choices of successor states.

As mentioned before, universal automata move from a state to a set of successor states simultaneously. This means a run of universal automaton needs to track several threads at the same time, giving it a tree structure as opposed to the single thread/string structure of deterministic and nondeterministic automaton run. The nodes of a run-tree are labeled with states. The root node is always labeled with the initial state. A node in distance $i$ from the root node, has exactly one child for each state in the set given by applying the transition function to the label of the node and the $i$'th letter of the word. Each child node is labeled with a different state from that set. For a given universal automaton, each infinite word is associated with just one run (root is fixed and there are no choices for the child nodes).

Chandra et al. [6] introduced alternating Turing machines and showed they accept precisely the recursively enumerable sets. Moreover, they characterized the complexity classes of languages accepted by time- (space-) bounded deterministic Turing machines in terms of complexity classes of languages accepted by time- (space-) bounded alternating Turing machines. Next they defined alternating finite state automata on finite words and showed they accept only regular languages, although, in general, $2^{2k}$ states are necessary
Table 1.1: Typeness for deterministic automata.

<table>
<thead>
<tr>
<th></th>
<th>DWW</th>
<th>DBW</th>
<th>DCW</th>
<th>DRW</th>
<th>DSW</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWW</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>DBW</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>DCW</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>DRW</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>DSW</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

Table 1.2: Typeness for nondeterministic automata.

<table>
<thead>
<tr>
<th></th>
<th>NWW</th>
<th>NBW</th>
<th>NCW</th>
<th>NRW</th>
<th>NSW</th>
</tr>
</thead>
<tbody>
<tr>
<td>NWW</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>NBW</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>NCW</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>NRW</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>NSW</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

and sufficient to simulate a \( k \)-state alternating finite automaton deterministically.

Given a state and a letter, an alternating automaton has a set of sets of states, from
which it has to choose one set of states and then move simultaneously to all the states
in that set. The range of alternating transition function is the power set of the power
set of the set of states. Usually, positive Boolean formulas over the automaton states are
used to represent the transition function range symbolically. Like universal automata,
since each move involves several states, the run of an alternating automaton is a labeled
tree. Like nondeterministic automata, since each move involves choices, each word is
associated with multiple runs.

Krishnan et al. [10] introduced the notion of typeness. For two classes of automata
\( C \) and \( C' \) (e.g. deterministic Rabin and deterministic Büchi), \( C \) is called \( C' \)-type if
the translation of a \( C \) automaton to \( C' \) automaton can be done by no changes to the
structure of the automaton (i.e. by changing the acceptance condition only). Note that
typeness does not require that all \( C \) automata can be translated to \( C' \). But when such
translation is possible it requires that it is also possible to be done with no changes to
the structure of the automaton. In [10], Krishnan et al. showed deterministic Rabin and
Streett are deterministic Büchi-type. In [15], Kupferman et al. give a complete picture of
typeness for deterministic (nondeterministic) Büchi, co-Büchi, Rabin, Streett and weak
automata. The results from [15] are summarized in Tables 1.1 and 1.2 (YES in column
\( C \) and row \( C' \) means \( C \) is \( C' \)-type).

Vardi and Wolper [33] describe how to use \( \omega \)-automata for program verification. In
essence, when given a nondeterministic program \( Prog \) with respect to a finite set of
propositions \( P \), each of the program states can be associated with the set of propositions
from $P$ that hold in that state. That is, each state of $Prog$ is associated with a letter from the alphabet $2^P$. We can then associate each computation of $Prog$ (i.e. sequence of states) with an infinite word over $2^P$. Hence the program $Prog$ induces a language of infinite words over $2^P$ (the words associated with $Prog$’s computations). We can construct from $Prog$ and $P$ an automaton, $A_{Prog}$ that recognize the same language induced by $Prog$ with respect to $P$. In a similar way, given a specification $\psi$ with respect to $P$, which describes all the allowed computations (in terms of $P$), $\psi$ induces a language of infinite words over $2^P$ (the words associated with the allowed computations). We can construct an automaton, $A_\psi$, for $\psi$ that recognize the same language induced by $\psi$. We can now use the automaton $A_{Prog}$ and $A_\psi$ to verify that $Prog$ satisfies $\psi$ (i.e. every computation of $Prog$ is allowed by $\psi$) by checking if $L(A_{Prog}) \subseteq L(A_\psi)$ holds (where $L(A)$ is the language of the automaton $A$) which is equivalent to checking $L(A_{Prog}) \cap L(A_{\neg \psi}) = \emptyset$ where $A_{\neg \psi}$ is the complementing automaton of $A_\psi$ (i.e. for every word $w \in L(A_{\neg \psi})$ iff $w \notin L(A_\psi)$). It follows that complementation of automata and emptiness check are of great interest.

As stated before, Büchi gave a complementation construction for nondeterministic Büchi automata that is doubly exponential. In an unpublished work, Michel [21] proved the complementation of nondeterministic Büchi automata has a lower bound of $2^{O(n \log n)}$. In [30], Safra gives a constructions that meets the lower bound ($2^{O(n \log n)}$). The construction has three steps. In the first step, which is the main contribution of [30], a nondeterministic Büchi automaton is translated to a deterministic Rabin automaton with $2^{O(n \log n)}$ states blowup. In the next step, the deterministic Rabin automaton is converted to deterministic Streett automaton with complementing language. In the final step, a translation suggested by M. Y. Vardi [30] from deterministic Streett to nondeterministic Büchi is used.

In [12], Kupferman and Vardi give an alternative complementation construction for nondeterministic Büchi automata that meets the lower bound and does not go through determinization. The construction of Kupferman and Vardi also has three steps. Given a nondeterministic Büchi automaton we first convert it to universal co-Büchi automaton with complementing language. In the next step, which is the main contribution of [12], the co-Büchi automaton is translated to an alternating weak automaton with $O(n^2)$ states blowup. In the final step, the alternating weak automaton is translated to a nondeterministic Büchi automaton using the translation suggested by Miyano and Hayashi [22].

The work in this thesis is motivated by the second step in Kupferman and Vardi complementation construction. Michel’s lower bound for the nondeterministic Büchi complementation suggests a lower bound of $\Omega(n \log n)$ for the translation from nondeterministic Büchi automata to alternating weak automata. This leaves a gap between the best known upper bound and the lower bound. In the first half of this work we
investigate this gap. We start by giving a direct translation from alternating Büchi word automata to alternating weak word automata. The importance of that translation is in its proof (similar translation with the same complexity can already be achieved [12]). In the core of the proof is the pigeon hole principal, which explains why multiple copies of the original automaton structure need to be incorporated in the weak automaton.

We then give a direct proof for the $\Omega(n \log n)$ lower bound and explore different possibilities to improve it. Although we were not able to improve the lower bound, we provide some insights about it. Our lower bound proof exposes the explicit relationship between nondeterministic Büchi complementation and the translation from alternating Büchi to alternating weak automata. The language family used in the proof of the nondeterministic Büchi complementation lower bound, is also used in the proof of the alternating Büchi to alternating weak translation lower bound. Since the complementation is already tight, this relationship puts restriction on any other language family that can be instrumental in improving the lower bound for the translation. Such family must have properties rendering it unusable for the complementation lower bound proof.

The fact that alternating weak automata can be complemented very easily is the dominating factor in the lower bound proof. In fact, our proof does not make use of the special structure of weak automata (other then the easy complementation), and can be seen as a lower bound for the complementation of alternating Büchi automata. In order to extend our understanding of how to exploit the special structure of weak automata, we further restrict it to the point where we are able to provide a lower bound of $\Omega(n^2)$. In the proof of the $\Omega(n \log n)$ lower bound (translation to alternating weak) we show the number of subsets of states in the weak automaton is at least $n!$ (where $n$ is the number of states in the original automaton), and therefore the weak automaton has at least $\log n!$ states. In the proof of the $\Omega(n^2)$ lower bound (translation to restricted alternating weak) we are able to show that the $n!$ subsets have special inter-relations and therefore we must have at least $\binom{n^2}{2}$ states.

In [14], Kupferman et al. showed that the nonemptiness problem for weak alternating automata over a singleton alphabet can be solved in linear time. The best known upper bound for the nonemptiness problem for nondeterministic Büchi and co-Büchi automata is quadratic. Using the first two steps from Kupferman and Vardi complementation construction (translating nondeterministic Büchi automaton to alternating weak automaton) and then the linear time algorithm for nonemptiness from [14], the best known upper bound for the nonemptiness problem for nondeterministic Büchi (and co-Büchi) automata is matched.

Solving the nonemptiness problem for nondeterministic Büchi automata is usually done by reduction to the bad cycle problem. In the bad cycle problem we try to find a cycle in a graph that goes through a given set of nodes (i.e. fair cycle). In general finding a bad cycle is linear in the size of the graph (using depth first search that iden-
ties strongly connected components). Since the graphs we work on are very big, BDDs (binary decision diagrams) or other symbolic means are usually used to represent them. Implementing a DFS algorithm over such symbolic representation is not possible (that is, not possible without defeating the whole purpose of symbolic representation) since DFS requires iteration over the individual nodes. Hence, symbolic model checking algorithms use variants of the Emerson and Lei algorithm [8] to find bad cycles. This algorithm is quadratic in the graph size with best known lower bound of $n \log n$. If we view the graph as a nondeterministic Büchi automaton over a singleton alphabet, the quadratic complexity can be matched by first translating the automaton to alternating weak automaton (quadratic in size) and then using the linear nonemptiness algorithm from [14].

The nonemptiness problem for alternating Büchi automata over a singleton alphabet is another closely related problem. This problem is equivalent to the special case of parity games [7] with two colors. The best known algorithm for solving the nonemptiness problem for alternating Büchi automata over a singleton alphabet is quadratic [34]. Again, by translating the alternating automaton to alternating weak automaton and then using [14], the upper bound can be matched.

As suggested by the translation from nondeterministic Büchi word automata to alternating weak word automata, alternating weak and Büchi automata have the same expression power (both are $\omega$-regular). This is not the case when considering tree automata. Tree automata [27] are very similar to word automata but instead of running on infinite words, tree automata run on infinite binary trees. Alternating weak tree automata are strictly less expressive than alternating Büchi tree automata. A language of trees can be recognized by an alternating weak tree automaton iff the language and its complement can be recognized by nondeterministic Büchi tree automata (following Rabin [28] expressiveness results in second-order logic, and the equivalence of alternating weak tree automata and weak second-order logic). In [13], Kupferman and Vardi extended the ideas behind the translation to alternating weak word automata to handle tree automata. Given two nondeterministic Büchi tree automata, recognizing some language and its complement, Kupferman and Vardi construct an alternating weak tree automaton that recognize the language, with quadratic states blowup. The translation can be supplemented by translating the alternating weak tree automaton to formula in the alternating-free $\mu$-calculus [11]. Since this translation is linear [11], we get a quadratic translation from nondeterministic Büchi automata (when the complement is also given) to alternation-free $\mu$-calculus, which extends the scope of efficient symbolic model checking to highly expressive specification formalisms.

In [3], Boker and Kupferman closed the gap for the translation (when possible) from Emerson and Lei approach to the bad cycle problem is called SCC-hull. A different approach, called SCC-enumeration have been shown to be $\Theta(n \log n)$ [2]. Yet, in practice, the SCC-hull preforms better [29].
nondeterministic Büchi automata to nondeterministic co-Büchi automata. Boker and Kupferman improved the best known lower bound from $1.5n$ to $2^\Omega(n)$ and the best known upper bound from $2^{O(n \log n)}$ to $n2^n$ (i.e. $2^{\Theta(n)}$). For alternating automata the translation from Büchi to co-Büchi is always possible and has the same complexity as the complementation of alternating Büchi automata (and dually, complementing alternating co-Büchi automata). Complementing an alternating Büchi automaton and dualising the transition function (see chapter 2) result in an alternating co-Büchi automaton with the same language as the original (the dualisation complements the language a second time). Hence the gap for the translation of alternating Büchi to co-Büchi is the same as the gap for translating alternating Büchi to alternating weak.

In the second half of this work we also explore the translation from deterministic Büchi automata to deterministic $k$-weak automata. We first show that restricting the number of partitions in deterministic weak automata also restrict the expression power (this is not true in the case of nondeterministic, universal and alternating automata). We then show that deterministic Büchi automata are deterministic $k$-weak-type. That is, the translation of deterministic Büchi automata to deterministic weak automata with $k$ partitions involves only minor changes to the automaton (in particular, no changes to the transition function), when the translation is possible at all.
Chapter 2

Preliminaries

In this chapter we will formally lay down the basic concepts and notations. At the end of the chapter the reader will be familiar with finite state automata over infinite words, how they are expressed, their semantics and how we classify them into different groups.

2.1 Automata over infinite words

Before we define automata we need to define some basic notations. We will use subsets of natural numbers extensively throughout this thesis. For any \( n \in \mathbb{N} \) we denote by \([n]\) the set of \( n \) elements, \( \{0, 1, \ldots, n-1\} \). In particular, \([0]\) is the empty set, \( \emptyset \). The set of even number in \([n]\) is denoted \([n]^{\text{even}}\) and the set of odd number in \([n]\) is denoted \([n]^{\text{odd}}\). Automata over infinite words, as its name suggests, “reads” infinite words. We will formalize this notion when we define the semantics of automata. For now we will formally define what a word is. Given a set \( \Sigma \), called the alphabet, a function \( w : \mathbb{N} \rightarrow \Sigma \) is called an infinite word over \( \Sigma \). For notation ease we will write \( w = w_0w_1w_2\ldots \) to mean \( w \) is the word \( w : \mathbb{N} \rightarrow \Sigma \) where \( w(i) = w_i \) for every \( i \in \mathbb{N} \). The set of all infinite words over \( \Sigma \) is denoted \( \Sigma^\omega \). Although we do not discuss automata over finite words we will use finite words. Given an alphabet \( \Sigma \) and \( n \in \mathbb{N} \), a function \( w : [n] \rightarrow \Sigma \) is called a finite word over \( \Sigma \). \( n \) is called the length of the word and denoted by \( |w| \). The word of length 0, \( w : [0] \rightarrow \Sigma \), is called the empty word and we denote it by \( \epsilon \). The set of all finite words over \( \Sigma \) is denoted by \( \Sigma^* \). The set of all finite words over \( \Sigma \) except for the empty word (i.e. \( \Sigma^* \setminus \{\epsilon\} \)) is denoted \( \Sigma^+ \). Finally, for a finite word \( w \) and an infinite/finite word \( w' \) we say \( w \) is a prefix of \( w' \) if \( w \) is not longer then \( w' \) (i.e. \( w' \) is infinite or \(|w| \leq |w'|\)) and \( w_i = w'_i \) for every \( i \in [|w|] \).

We build language upon words. A set of words is called a language. Given a language \( L \) over alphabet \( \Sigma \) (i.e. \( L \subseteq \Sigma^\omega \)), we use over-line to indicate the complement language of \( L \) in \( \Sigma^\omega \), that is \( \overline{L} = \Sigma^\omega \setminus L \).

Transition relations of automata will be express using positive Boolean formulas.
Intuitively, this is the mechanism that defines how the automaton carries its steps. Given a set $X$, called the variables, we denote by $B^+(X)$ the set of positive Boolean formulas over $X$, i.e., all the formulas with variables from $X$, the atoms $\top, \bot$ and the operators $\land, \lor$ (in particular, ‘$\lnot$’ is not a valid operator). For example, let $X$ be the set $\{q_1,q_2,q_3\}$. Every element in $X$ is a positive Boolean formula and therefore ‘$q_1$’ is a positive Boolean formula. Any of the atoms is also a positive Boolean formula and therefore ‘$\bot$’ and ‘$\top$’ are positive Boolean formula. Using the operators $\land$ and $\lor$ we can create more complex formulas like $(q_1 \land q_2) \lor (q_1 \land q_3) \lor (q_2 \lor \bot)$.

The semantics of a positive Boolean formula over $X$ is a set of subsets of $X$. For $\psi \in B^+(X)$ and $S$, a subset of $X$, the relation $S \models \psi$ indicates $S$ satisfies $\psi$. We define the relation $\models$ inductively. Given a set $X$, $\psi, \psi_1, \psi_2 \in B^+(X)$, $x \in X$, and $S \subseteq X$,

- $S \models \top$
- $S \models x$ iff $x \in S$
- $S \models \psi_1 \land \psi_2$ iff $S \models \psi_1$ and $S \models \psi_2$
- $S \models \psi_1 \lor \psi_2$ iff $S \models \psi_1$ or $S \models \psi_2$

If $S \models \psi$ we say $S$ satisfies $\psi$, otherwise we say $S$ does not satisfy $\psi$ and we write $S \not\models \psi$.

The last thing we define before we give the definition of automata is a tree. Intuitively, we describe multiple consecutive steps an automaton takes, given a word, as a labeled tree. Properties of these trees will determine whether the word is accepted by the automaton. Formally, a tree is a subset $T \subseteq \mathbb{N}^*$ where $\epsilon \in T$ and for every $x \in T$, if $x' \in \mathbb{N}^*$ is a prefix of $x$ then $x' \in T$, i.e., $T$ is prefix-closed. The elements of $T$ are called nodes. $\epsilon$ is called the root of $T$. For every $x \in T$, the set $\{x' \in T \mid c \in \mathbb{N}, x' = x \cdot c\}$ is called the children of $x$. If $x \in T$ has no children (i.e., empty set), $x$ is called a leaf. For every $n \in \mathbb{N}$ we call the set $\{x \in T \mid |x| = n\}$ the $n$th level of $T$. A path in $T$ is a prefix-closed subset $\pi \subseteq T$ such that $\epsilon \in \pi$ and for any $x \in \pi$ either $x$ is a leaf or there exists a unique $c \in \mathbb{N}$ such that $x \cdot c \in \pi$. A path is called infinite if $\pi$ is infinite. Given a set $\Sigma$, a $\Sigma$-labeled tree is a pair $(T, r)$ where $T$ is a tree and $r : T \to \Sigma$ is a function. With each infinite path, $\pi$, in the $\Sigma$-labeled tree $(T, r)$, we associate a labeling function $r_\pi : \mathbb{N} \to \Sigma$ where $r_\pi(|x|) = r(x)$ for every $x \in \pi$ (note that $r_\pi$ is well defined since $|x| = |x'| \iff x = x'$ for any $x, x' \in \pi$ and for every $n \in \mathbb{N}$ there exists $x \in \pi$ such that $|x| = n$).

We are now ready to define automata over infinite words. All the automata in this thesis are finite state automata and so we will not explicitly state it anymore. Automata over infinite words are very similar to automata over finite words. Just like automata over finite words, automata over infinite words can be deterministic and non-deterministic. In fact, we classify automata over infinite words into two more classes, universal and alternating, which we will present shortly. The class of alternating automata is the most general from all these classes. In the same sense that deterministic automata over finite
words is a special case of non-deterministic automata over finite words, the classes of deterministic, non-deterministic and universal automata over infinite words are special cases of alternating automata over infinite words. In light of this, we start by defining alternating automata over infinite words and later on we will derive the other classes by adding restrictions.

**Definition 1** (Alternating automata over infinite words). *Alternating automaton* over infinite words is a tuple $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ where,

- $\Sigma$ is a finite set called the *alphabet* of $A$.
- $Q$ is a finite set called the *states* of $A$.
- $q_{\text{init}} \in Q$ is called the *initial state* of $A$.
- $\delta : Q \times \Sigma \rightarrow B^+ (Q)$ is a function called the *transition function* of $A$.
- $\alpha$ is called the *acceptance condition* of $A$ (the types of $\alpha$ is determined by the type of the automaton).

In its very core automaton is a mean of expressing a language. Given a word $w \in \Sigma^\omega$ we are interested to know whether $w$ is in the language an automaton expresses or not. Each word is associated with runs of the automaton. If one of these runs is an *accepting run* we will determine $w$ is in the language the automaton expresses. In the following we formally define run of automaton on a word.

**Definition 2** (Run & Accepting Run). Given an alternating automaton $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ and a word $w \in \Sigma^\omega$, any $Q$-labeled tree $\langle T, r \rangle$ that satisfies the following conditions is call a run of $A$ on $w$,

- $r(\epsilon) = q_{\text{init}}$.
- $\{ q \in Q \mid c \in \mathbb{N}, x \cdot c \in T, r(x \cdot c) = q \} \models \delta(r(x), w|_{\langle x \rangle})$ for every $x \in T$.

We say a run $\langle T, r \rangle$ of $A$ is *accepting* if every infinite path in $T$ satisfies $\alpha$.

As one can see the definition of the $\alpha$ part in automata is very loose. There are a few types of *acceptance condition* and each one has a different definition of $\alpha$\[1\]. In this thesis we deal almost exclusively with the *Büchi* \[3\] type. For this type $\alpha$ is defined to be a subset of $Q$. It remains to define how an infinite path satisfies a Büchi acceptance condition. Before we can do so we have to introduce the *inf operator*. Given a function $r : \mathbb{N} \rightarrow X$ we define $\inf (r)$ to be the set $\{ x \in X \mid \{ n \in \mathbb{N} \mid r(n) = x \} \text{ is infinite} \}$. In other words, $\inf (r)$ is the set of elements in $X$ that appear infinitely many times in $r$.

\[1\]Do not confuse the acceptance condition type with the automata class. We currently present the class of alternating automata and later on we will discuss the other classes, these classes differ one from the other by the restrictions on $\delta$. We now discuss the *type* of the acceptance condition which determines $\alpha$.
Definition 3 (Büchi acceptance condition). Let $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$. When interpreted as Büchi condition, $\alpha$ is a subset of $Q$. Given a $Q$-labeled tree $\langle T, r \rangle$ and a path $\pi$ in $T$, we say $\pi$ satisfies $\alpha$ if $\inf(r_\pi) \cap \alpha \neq \emptyset$.

A close relative of the Büchi type is the co-Büchi type.

Definition 4 (co-Büchi acceptance condition). Let $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$. When interpreted as co-Büchi condition $\alpha$ is a subset of $Q$. Given a $Q$-labeled tree $\langle T, r \rangle$ and a path $\pi$ in $T$, we say $\pi$ satisfies $\alpha$ if $\inf(r_\pi) \cap \alpha = \emptyset$.

Other types of acceptance condition are Rabin [27], Streett [31], Parity [23] and Muller [24]. In [17], Löding gives a formal definition of all these types.

At last, we are ready to define the language of automata.

Definition 5 (Language of automaton). Given an alternating automaton $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ and a word $w \in \Sigma^\omega$, we say $A$ accepts $w$ if there exists an accepting run of $A$ on $w$. The set $\{w \in \Sigma^\omega \mid A$ accepts $w\}$ is denoted by $L(A)$.

To sum up, a word $w$ is accepted by automaton $A$ if there exists $\langle T, r \rangle$, a run of $A$ on $w$, such that all infinite paths in $\langle T, r \rangle$ satisfy the acceptance condition.

When we draw alternating automata we will draw a circle for each state with the state name inside the circle. Double circles indicate states in $\alpha$. The initial state will be indicated by an incoming arrow with no source. Drawing the transition function is a bit tricky. Consider $\delta(q, \sigma)$ and let $S$ be the set of all minimal sets that satisfy it. For each $S \in S$ we draw a directed multi-edge labeled with $\sigma$, the source of the edge is $q$ and the targets are $S$ (i.e. the edge has a single source and multiple targets, see the edge outgoing from state 0 and labeled with 0 in Figure 2.1). If $S = \{\emptyset\}$ (i.e. $\delta(q, \sigma)$ is equivalent to $\top$) we will draw an edge labeled with $\sigma$, the source of the edge is $q$ and the target is $\top$ (see the edge outgoing from state 1 and labeled with 1 in Figure 2.1). If $S$ is empty (i.e. $\delta(q, \sigma)$ is unsatisfiable) we will not draw any edge for it. Sometimes we want to emphasize that $\delta(q, \sigma)$ is unsatisfiable, in that case we will draw an edge labeled with $\sigma$, the source of the edge is $q$ and the target is $\bot$. Figure 2.1 illustrates the drawing of alternating automaton.

2.2 Observations

Consider an accepting run $\langle T, r \rangle$ of some alternating Büchi/co-Büchi automaton $A$ on a word $w$. We say that two nodes $x_1, x_2 \in T$ are similar if $|x_1| = |x_2|$ and $r(x_1) = r(x_2)$. We say the run $\langle T, r \rangle$ is memoryless if for every similar nodes $x_1, x_2 \in T$ and $c \in \mathbb{N}$ we have that $x_1 \cdot c \in T \iff x_2 \cdot c \in T$ and (if they are in $T$) $x_1 \cdot c, x_2 \cdot c$ are similar (i.e. $r(x_1 \cdot c) = r(x_2 \cdot c)$). Intuitively, if the run reaches some state $q$ more than once on the same level, it will continue the subruns rooted at each of these nodes in a similar manner.
Figure 2.1: Drawing alternating automaton example. \( A = (\Sigma, Q, q_{\text{init}}, \delta, \alpha) \) where \( \Sigma = \{2\}, Q = \{2\}, q_{\text{init}} = 0, \delta(0, 0) = 0 \land 1, \delta(0, 1) = 0, \delta(1, 0) = 1 \) and \( \delta(1, 1) = \top \).

In [7], Emerson and Jutla prove for alternating parity tree automata, that if there exists an accepting run then there also exists a memoryless run. Since alternating Büchi and co-Büchi word automata are a special case of alternating parity tree automata we have the following theorem.

**Theorem 1** ([7]). If an alternating Büchi/co-Büchi automaton \( A \) accepts a word \( w \) then there exists a memoryless accepting run of \( A \) on \( w \).

Note that the converse is trivial, i.e., if there exists a memoryless accepting run then \( A \) accepts \( w \). This property allows us to represent runs in a more structured way. Namely, we will represent a run as an infinite DAG (directed acyclic graph). Similar nodes in the run tree will be mapped to the same node in the DAG. Given alternating automaton \( A = (\Sigma, Q, q_{\text{init}}, \delta, \alpha) \) and a memoryless run \( \langle T, r \rangle \), we define the DAG, \( G = (V, E) \), corresponding to the run \( \langle T, r \rangle \) as follows,

- \( V = \{(r(x), |x|) \mid x \in T\} \subseteq Q \times \mathbb{N} \)
- \( E = \{(\langle r(x), |x| \rangle, \langle r(x \cdot c), |x \cdot c| \rangle) \mid c \in \mathbb{N} \text{ and } x, x \cdot c \in T\} \)

For every \( n \in \mathbb{N} \) we call the set \( V \cap (Q \times \{n\}) \) the \( n \)’th level of \( G \). If a level of \( \langle T, r \rangle \) has more than \( |Q| \) nodes, at least two of them have to be similar. Therefore, each level of \( G \) has at most \( |Q| \) nodes.

Given an alternating Büchi/co-Büchi automaton \( A = (\Sigma, Q, q_{\text{init}}, \delta, \alpha) \), by adding two new states to \( Q \) we can eliminate the use of \( \bot \) and \( \top \) in \( \delta \). Let \( q_{\bot} \) and \( q_{\top} \) be two unique states not in \( Q \) and let \( A' = (\Sigma, Q \cup \{q_{\bot}, q_{\top}\}, q_{\text{init}}, \delta', \alpha') \) where \( \alpha' = \alpha \cup \{q_{\top}\} \) if \( A \) is Büchi and \( \alpha' = \alpha \cup \{q_{\bot}\} \) if \( A \) is co-Büchi, and \( \delta' \) is the result of replacing in \( \delta \) every \( \bot \) with \( q_{\bot} \), every \( \top \) with \( q_{\top} \) and adding \( \delta'(q_{\bot}, \sigma) = q_{\bot} \) and \( \delta'(q_{\top}, \sigma) = q_{\top} \). We have that \( \mathcal{L}(A) = \mathcal{L}(A') \) and the transition function of \( A' \) uses neither \( \bot \) nor \( \top \). Hence the atoms \( \bot \) and \( \top \) do not add to the expression power of alternating Büchi/co-Büchi automata. Moreover, removing them from a given automaton can be done in the cost of no more then two extra states.
2.3 Classes of automata

As we saw before, automata can be classified by the type of their acceptance condition. We now introduce an orthogonal classification. This classification is by properties of the transition function, similar to what one might already be familiar with from automata over finite words.

Definition 6 (Automata classes). Let $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ be an alternating automaton.

- We say $A$ is deterministic if $\delta(q, \sigma)$ has no operators (i.e. no $\land$ or $\lor$ in the formula). In other words, $\delta(q, \sigma)$ is a variable or $\bot$ or $\top$.
- We say $A$ is nondeterministic if the operators appearing in $\delta(q, \sigma)$ are restricted to $\lor$.
- We say $A$ is universal if the operators appearing in $\delta(q, \sigma)$ are restricted to $\land$.

Given an alternating automaton $A$ and a word $w$. If $A$ is deterministic or universal then $A$ has exactly one run on $w$. If $A$ is deterministic or nondeterministic then every run of $A$ on $w$ has exactly one path (i.e. the tree is a string).

Throughout the thesis we will use a 3 letters abbreviation to denote automata classes. The first letter denotes the transition function type, ‘A’ for alternating, ‘D’ for deterministic, ‘N’ for nondeterministic and ‘U’ for universal. The second letter denotes the acceptance condition type, ‘B’ for Büchi and ‘C’ for co-Büchi. The last letter will always be ‘W’ to denoting word automata (in the literature this letter can also be ‘T’ to denote tree automata which is not in our scope). For example, the class of nondeterministic Büchi automata is denoted by NBW and the class of alternating co-Büchi automata is denoted by ACW.

In [26], Muller et al. introduce weak automata. Weak can be seen as another restriction on the transition function for Büchi automata and we define it as follows.

Definition 7 (Weak automaton). For any $m \in \mathbb{N}$, an alternating automaton $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ with Büchi acceptance condition and no $\bot$ or $\top$ in the transition function, is said to be $m$-weak if there exists a partition of $Q$ into $Q_1, Q_2, \ldots, Q_m$ such that

1. $Q_i \subseteq \alpha$ or $Q_i \cap \alpha = \emptyset$ for any $1 \leq i \leq m$, and
2. If $q_i$ occurs in $\delta(q_j, \sigma)$ and $q_i \in Q_{i'}$ and $q_j \in Q_{j'}$ then $i' \leq j'$ for any $q_i, q_j \in Q$ and $\sigma \in \Sigma$.

\footnote{In fact, in some places universal automata are also said to be deterministic. We avoid doing so to prevent confusion.}

\footnote{In other papers where other types are used, ‘R’ stands for Rabin, ‘S’ for Street, ‘P’ for parity and ‘M’ for Muller.

15
Automaton $A$ is said to be weak if it is $m$-weak for some $m \in \mathbb{N}$.

Note that the acceptance condition type of weak automata is always Büchi. The use of $\bot$ and $\top$ in the transition function can affect the minimal $m$ an automaton can satisfy. For this reason we define $m$-weak for automata that do not use $\bot$ or $\top$. For automaton $A$ that does use $\bot$ or $\top$ we say $A$ is $m$-weak if $A'$, the result of eliminating $\bot$ and $\top$ from $A$ (as described at the end of the previous section), is $m$-weak. In the literature, sometimes a different approach is taken. No restrictions on the presence of $\bot$ or $\top$ is given, instead if the states can be partitioned to $m$ partitions (satisfying the two conditions above) the automaton is classified as $(m+1)$-weak.

In the class abbreviation, the weak class will have ‘W’ as the second letter. Furthermore, we add the suffix $[m]$ for the class of $m$-weak automata. For example, the class of deterministic weak automata is denoted by $DWW$ and the class of nondeterministic 5-weak automata is denoted $NWW[5]$.

For a class of automata $C$ and a language $L$, we say $L$ is $C$ recognizable if there exists automaton $A \in C$ such that $A$ accepts $L$ (i.e. $L(A) = L$). We denote by $\mathcal{L}(C)$ the set of all $C$ recognizable languages. For two classes of automata $C$ and $C'$, we say $C$ is $C'$-type if for every automaton $A = (\Sigma, Q, q_{\text{init}}, \delta, \alpha) \in C$ such that $\mathcal{L}(A)$ is $C'$ recognizable, there exists some $C'$-type $\alpha'$ such that $A' = (\Sigma, Q, q_{\text{init}}, \delta, \alpha') \in C'$ and $\mathcal{L}(A) = \mathcal{L}(A')$.

In this thesis we will explore the translation of automata from class $C$ to class $C'$. Translation is an algorithm that takes as input automaton $A \in C$ and returns automaton $A' \in C'$ such that $\mathcal{L}(A) = \mathcal{L}(A')$. Complementation of automata class $C$ is an algorithm that takes as input automaton $A \in C$ and returns automaton $A' \in C$ such that $\mathcal{L}(A') = \overline{\mathcal{L}(A)}$.

### 2.4 Duality

In [25], Muller and Schupp give an easy complementation procedure. In fact the procedure Muller and Schupp give is more general for tree automata which is outside of this thesis scope. The following theorem is a special case of Muller and Schupp result.

**Theorem 2.** Let $A = (\Sigma, Q, q_{\text{init}}, \delta, \alpha)$ be an alternating Büchi or co-Büchi automaton, let $\overline{\delta}$ be the same as $\delta$ with every $\bot$ replaced with $\top$, every $\top$ replaced with $\bot$, every $\land$ replaced with $\lor$ and every $\lor$ replaced with $\land$ (all replacements are done simultaneously) and let $\overline{A}$ be the automaton $\langle \Sigma, Q, q_{\text{init}}, \overline{\delta}, \alpha \rangle$. If $A \in ABW$ then the automaton $\overline{A}$ is an ACW automaton and $\mathcal{L}(\overline{A}) = \overline{\mathcal{L}(A)}$. Moreover, if $A \in ACW$ then the automaton $\overline{A}$ is an ABW automaton and $\mathcal{L}(\overline{A}) = \overline{\mathcal{L}(A)}$.

We refer the reader to Löding work [17] for a complete proof in the context of automata over infinite words.
It can be easily seen that if $A$ is deterministic then so is $\overline{A}$, if $A$ is nondeterministic then $\overline{A}$ is universal, if $A$ is universal then $\overline{A}$ is nondeterministic and if $A$ is weak then $\overline{A}$ is also weak.

This theorem has a particular interesting effect on AWW automata. Observe that for any $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle \in \text{AWW}$ we have that $A' = \langle \Sigma, Q, q_{\text{init}}, \delta, Q \setminus \alpha \rangle \in \text{ACW}$ has the same language as $A$ (i.e. $L(A') = L(A)$). We conclude that for any $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle \in \text{AWW}$ we have that $A' = \langle \Sigma, Q, q_{\text{init}}, \overline{\delta}, Q \setminus \alpha \rangle \in \text{AWW}$ and $L(A') = \overline{L(A)}$. That is, complementation of AWW is simple, all we need to do is dualize the transition function and complement $\alpha$. 
Chapter 3

ABW to AWW

In this chapter we explore the translation of ABW automata to AWW automata. This translation is of particular interest as it can be used in the complementation of NBW automata. In the following we will show the best known upper and lower bounds for the state complexity in the translation of ABW to AWW. Previous results achieve the same bounds we will show, but in an indirect way. Since the bounds do not meet (i.e. not tight) we believe giving direct proofs for both lower and upper bound can help us to find insights that will be essential for improving the bounds. After we present our proofs for the best known upper and lower bounds we show what we think will be the best way to continue the search for the tight bound. At the end of this chapter (and thesis), the tight bound for the ABW to AWW translation remains an open problem, with $\Omega(n \log n)$ being the best known lower bound and $O(n^2)$ being the best known upper bound.

3.1 Upper bound for the translation from ABW to AWW

Obtaining an upper bound for the state complexity of the translation from ABW to AWW involves an algorithm for the translation and analyzing the number of states in the resulting automata. One can find a linear translation from deterministic Muller automata to AWW automata in [26,16]. However, translating ABW automata to deterministic Muller automata involves an exponential blow-up [30]. For a long time the composition of the above two algorithms was the best known translation from ABW to AWW. In [12], Kupferman and Vardi obtain a quadratic algorithm for the translation from ABW to AWW. The core of the algorithm in [12] is the translation from ACW to AWW. By applying the duality principal to ABW and complementation to AWW (both are linear) we obtain a quadratic translation from ABW to AWW. Namely, given $A \in \text{ABW}$ we first compute the dual $\overline{A} \in \text{ACW}$, we then translate $\overline{A}$ to $A' \in \text{AWW}$ and finally we compute $A'' \in \text{AWW}$, the complementation of $A'$. We have $\mathcal{L}(A'') = \overline{\mathcal{L}(A')} = \overline{\overline{\mathcal{L}(A)}} = \mathcal{L}(A)$. Note that we actually applied the duality principal twice, first in the translation of ABW
to ACW and second in the complementation of AWW. In the following we obtain a
direct translation from ABW to AWW. In fact, the proof can be viewed as the “dual” to
the proof obtained in [12]. Instead of using ranking (progress-measure) as in [12] we
use infinite applications of the pigeon hole principal. For simplicity, we first give a translation
from NBW to AWW which is very similar to the ABW to AWW translation. After the
proof we discuss the differences.

**Theorem 3.** Let A be an NBW. There is an A′ ∈ AWW such that L(A′) = L(A) and
the number of states in A′ is quadratic in that of A.

**Proof.** Let A = ⟨Σ, Q, q_init, δ, α⟩ ∈ NBW, and let n = |Q|. For simplicity we assume δ
uses neither ⊥ nor ⊤. In the following we write q′ ∈ δ(q, σ) iff q′ appears in the formula
δ(q, σ). We define A′ = ⟨Σ, Q′, q_init′, δ′, α′⟩, where

- Q′ = Q × [2n + 1]
- q_init′ = ⟨q_init, 2n⟩
- δ′(⟨q, i⟩, σ) = \[\begin{cases} \bigvee_{q′ \in \delta(q, \sigma)} (⟨q′, i⟩ \land ⟨q′, i - 1⟩) & \text{if } i \neq 0 \text{ is even and } q \notin \alpha \\
\bigvee_{q′ \in \delta(q, \sigma)} ⟨q′, i - 1⟩ & \text{if } i \text{ is odd and } q \in \alpha \\
\bigvee_{q′ \in \delta(q, \sigma)} ⟨q′, i⟩ & \text{otherwise} \end{cases}\]
- α′ = Q × [2n + 1]^{even}

Let Q_i = Q × {i}. The partition of Q′ to Q_0, Q_1, ..., Q_{2n} satisfies the week condition
and therefore A′ ∈ AWW.

We now prove the correctness of the construction. We first prove that L(A′) ⊆ L(A).
Consider a word w accepted by A′ and let ⟨T_r, r⟩ be an accepting run of A′ on w. We
distinguish two cases. First we assume there exists some non-zero even partition Q_{2k}
and node x ∈ T_r such that r(x · x′) ∈ Q_{2k} for any x′ such that x · x′ ∈ T_r, i.e., there
exists a subtree of T_r for which all nodes are labeled with Q_{2k} states (we say the run
gets trapped in Q_{2k}). We note that in this case, by the definition of δ′, for any x′ and
r(x · x′) = ⟨q, 2k⟩ we have that q ∈ α. Moreover, the subtree rooted in x is actually a path.
Let x_0, x_1, ..., be the (unique) path in T_r that starts from x_0 = ε and passes through
x and let r(x_i) = ⟨q_i, j_i⟩ for any i ≥ 0. By the above observation we have that
q_i ∈ α for any i ≥ m where x_m = x and therefore q_0, q_1, ..., is an accepting run of A on
w.

We now assume the run does not get trapped in any non-zero even partition. In this
case we are sure there exists a level in the run such that for every even partition, Q_{2k},
there exists x ∈ T_r from the level for which r(x) ∈ Q_{2k}. Since there are n + 1 even partitions and only n states in each, by the pigeon hole principal there has to be two
nodes x, x′ ∈ T_r from the level such that r(x) = ⟨q, 2k⟩ and r(x′) = ⟨q, 2k′⟩ and k < k′.
(i.e. the run reaches two copies of the same node at the same time). To construct the accepting path of $A$ we will find a path in $T_r$ and use its projection on $Q$ (i.e. remove the tags). A problem arise when the path reach $Q_0$, the projection on $Q$ cannot be guaranteed to be accepting. We will use these copies of the same states to “jump” to a higher partition and avoid getting trapped in $Q_0$.

We describe the second case (the run does not get trapped in any non-zero even partition) more formally. We will construct a sequence $(x_0,y_0),(x_1,y_1),\ldots$ such that $y_i$ is a prefix of $x_{i+1}$ and $y_{i+1}$ are nodes in the subtree rooted in $y_i$, and $r(x_{i+1}),r(y_{i+1})$ are copies of the same state, both are in even partitions and $r(y_{i+1})$ is in a higher partition then $r(x_{i+1})$ (i.e. $r(x_{i+1}) = \langle q,2k \rangle$, $r(y_{i+1}) = \langle q,2k' \rangle$, $k < k'$), for any $i \geq 0$. Observe the sub-paths from $y_0$ to $x_1$, from $y_1$ to $x_2$, from $y_2$ to $x_3$ and so on. If we project them on $Q$ using $r$ each sub-path starts with the same state the previous sub-path ended with. Concatenating the projections gives us an accepting run of $A$ on $w$. Note that $r(y_i)$ is in the same partition or higher then $r(y_{i+1})$ and therefore (strictly) higher then the partition of $r(x_{i+1})$ which insures the projection of the sub-path from $y_i$ to $x_{i+1}$ on $Q$ has at least one accepting state.

We set $x_0 = \epsilon$ and $y_0 = \epsilon$. Given $(x_i,y_i)$ we observe the subtree rooted at $y_i$. By the pigeon hole principal, there exist nodes $z,z'$ in the subtree rooted in $y_i$, integers $0 \leq k < k' \leq n$ and state $q \in Q$ such that $|z| = |z'|$ and $r(z) = \langle q,Q_{2k} \rangle$ and $r(z') = \langle q,Q_{2k'} \rangle$. We set $x_{i+1} = z$ and $y_{i+1} = z'$ for such $z,z'$ with the greatest $k'$. The selection of $z,z'$ with the greatest $k'$ insures the pigeon hole principal will still be valid in the next iteration (in other wards, for any partition $Q_{2k''}$ where $k' < k''$ we have that for the next run levels the states in $Q_{2k''}$ are unique to this partition).

Finally we show $L(A) \subseteq L(A')$. Consider a word $w$ accepted by $A$. Let $\pi = \pi_0\pi_1\ldots$ be an accepting run of $A$ on $w$. We define a run of $A'$ on $w$, $\langle T_r,r \rangle$, recursively.

**Base:** $\{\epsilon\} \subseteq T_r$ and $r(\epsilon) = \langle \pi_0,2n \rangle$ (note that $\pi_0 = q_{init}$ and $\langle \pi_0,2n \rangle = q'_{init}$).

**Closure:** For any $x \in T_r$ such that $r(x) = \langle \pi_i,k \rangle$,

1. If $k \neq 0$ is even and $\pi_i \notin \alpha$ then $x \cdot 0,x \cdot 1 \in T_r$ and $r(x \cdot 0) = \langle \pi_{i+1},k \rangle$ and $r(x \cdot 1) = \langle \pi_{i+1},k - 1 \rangle$.
2. If $k$ is odd and $\pi_i \in \alpha$ then $x \cdot 0 \in T_r$ and $r(x \cdot 0) = \langle \pi_{i+1},k - 1 \rangle$.
3. Otherwise $x \cdot 0 \in T_r$ and $r(x \cdot 0) = \langle \pi_{i+1},k \rangle$.

(note the similarity between the cases in the definition of $\delta'$ and the cases of the closure).

To see that $\langle T_r,r \rangle$ is an accepting run consider a sub-path in $T_r x_0,x_1,\ldots$ such that $r(x_0) = \langle \pi_i,k \rangle \notin \alpha'$. Since $\pi$ is accepting, there exists minimal $i' \geq 0$ such that $\pi_{i+i'} \in \alpha$. By the definition of $\alpha'$ we have that $k$ is odd. Therefore $r(x_j) = \langle \pi_{i+j},k \rangle$ for any $0 < j \leq i'$ (see third case in the closer) and $r(x_{i'+1}) = \langle \pi_{i+i'+1},k - 1 \rangle \in \alpha'$ (see second case in the closure).
Note that the resulting automaton is slightly different from the automaton one would get by applying the construction from [12]. The difference is in $\delta'$, the automaton in [12] would have

$$\delta'(\langle q, i \rangle, \sigma) = \begin{cases} 
\bigvee_{q' \in S} \delta(q, \sigma) \left( \bigwedge_{q \in S} \left( \langle q', i \rangle \land \langle q', i - 1 \rangle \right) \right) & \text{if } q \neq \alpha \text{ or } i \text{ is even} \\
\top & \text{otherwise}
\end{cases}$$

**Theorem 4.** Let $A$ be an ABW. There is an $A' \in AWW$ such that $L(A') = L(A)$ and the number of states in $A'$ is quadratic in that of $A$.

The construction of $A'$ is very similar to the one in the previous proof. In this case we will use the following $\delta'$,

$$\delta'(\langle q, i \rangle, \sigma) = \begin{cases} 
\bigvee_{S \models \delta(q, \sigma)} \left( \bigwedge_{q' \in S} \left( \langle q', i \rangle \land \langle q', i - 1 \rangle \right) \right) & \text{if } i \neq 0 \text{ is even and } q \neq \alpha \\
\bigvee_{S \models \delta(q, \sigma)} \left( \bigwedge_{q' \in S} \langle q', i \rangle \right) & \text{if } i \text{ is odd and } q \in \alpha \\
\bigvee_{S \models \delta(q, \sigma)} \left( \bigwedge_{q' \in S} \langle q', i \rangle \right) & \text{otherwise}
\end{cases}$$

It is easy to see the proof with minor changes still holds. The observant reader will notice the definition of $\delta'$ uses the semantics of $\delta$, i.e., $S \models \delta(q, \sigma)$. In the NBW case $\{q'\} \models \delta(q, \sigma)$ is replaced by the syntactic condition “$q'$ appears in the disjunction $\delta(q, \sigma)$”.

For the ABW case we first define two functions, $\text{tag} : B^+(Q) \times [2n+1] \rightarrow B^+(Q')$ and $\text{tag}' : B^+(Q) \times [2n+1] \rightarrow B^+(Q')$ where $\text{tag}(\psi, i)$ is obtained from $\psi$ by replacing any variable $q$ by $\langle q, i \rangle$, and $\text{tag}'(\psi, i)$ is obtained from $\psi$ by replacing any variable $q$ by the conjunction $\langle q, i \rangle \land \langle q, i - 1 \rangle$. We can now define

$$\delta'(\langle q, i \rangle, \sigma) = \begin{cases} 
\text{tag}'(\delta(q, \sigma), i) & \text{if } i \neq 0 \text{ is even and } q \neq \alpha \\
\text{tag}(\delta(q, \sigma), i - 1) & \text{if } i \text{ is odd and } q \in \alpha \\
\text{tag}(\delta(q, \sigma), i) & \text{otherwise}
\end{cases}$$

Note the two different definitions of $\delta'$ have the same semantics, i.e., if we denote the first definition by $\delta'_1$ and the second definition by $\delta'_2$ then $S \models \delta'_1(q, \sigma)$ iff $S \models \delta'_2(q, \sigma)$ for any $S, q$ and $\sigma$.

### 3.2 Lower bound for the translation from ABW to AWW

To show a $\Omega(f(n))$ lower bound for the state complexity of translation from class $C$ to class $C'$ one needs to show an example for $C$ automaton with $O(n)$ states, such that any $C'$ automaton with the same language will have at least $\Omega(f(n))$ states. Since we are interested in an asymptotic bound we have to show such example for every $n$ greater then some $m$. To do so we parametrize the example with $n$, i.e., we describe the $C$ automaton
in terms of \( n \). The resulting set of automata (for \( n = 0, n = 1, n = 2, \ldots \)) is called a family of automata, and the set of their languages is called a family of languages.

Sometimes we can take a shortcut and show lower bound by reduction. If we know the translation from \( C_1 \) to \( C_2 \) has an \( \Omega(f(n)) \) lower bound and the translation from \( C_3 \) to \( C_2 \) has an \( O(g(n)) \) upper bound then the translation from \( C_1 \) to \( C_3 \) has an \( \Omega(h(n)) \) lower bound for any function \( h \) such that \( \Omega(f(n)) \geq \Omega(g(h(n))) \).

In [12], Kupferman and Vardi use reductions to show the translation from NBW to AWW cannot be improved to linear translation due to the lower bound of NBW complementation, as follows. The complementation of NBW is known to have a \( \Omega(2^n \log n) \) lower bound (unpublished work [21] by Michel, see [32, 18]). The complementation of AWW is known to be \( O(n) \) (duality) and the translation from AWW to NBW is known to be \( O(3^n) \) (Miyano and Hayashi breakpoint construction [22, 4]). Hence the translation from NBW to AWW has an \( \Omega(n \log n) \) lower bound. Since NBW is a special case of ABW we conclude the translation from ABW to AWW also has an \( \Omega(n \log n) \) lower bound.

By its nature, the reduction technique is very opaque and makes it hard to improve the lower bound. To do so one will have to improve the other bounds involved (e.g. improving \( f \) or \( g \) above) or find a different reduction that goes through different classes. All the other bounds used by Kupferman and Vardi reduction are already tight, meaning we cannot improve the lower bound of ABW to AWW translation using this reduction.

In the following we present a proof for the \( \Omega(n \log n) \) lower bound of the ABW to AWW translation that takes its inspiration from Michel’s unpublished work, but applies it directly to the translation in hand (i.e. no reductions). We hope that this proof will give us insights on how to improve the lower bound.

**Definition 8** (The language \( L_n \)). \( L_n \) is the language defined by the NBW automaton in Figure 3.1. Formally, \( L_n = \mathcal{L}(A) \) where \( A = (\Sigma, Q, q_{\text{init}}, \delta, \alpha) \) and,

- \( \Sigma = [n] \times [n] \)
- \( Q = \{(0, 0)\} \cup [n] \times \{1, 2\} \) (note that \( |Q| = 2n + 1 \))
- \( q_{\text{init}} = (0, 0) \)
- For any \( \sigma \in \Sigma \), \( \delta(q_{\text{init}}, \sigma) = \bigvee_{i \in [n]} \langle i, 1 \rangle \)
  - For any \( \langle q, 1 \rangle \in Q \) and \( \langle \sigma_1, \sigma_2 \rangle \in \Sigma \),
    \[
    \delta(\langle q, 1 \rangle, \langle \sigma_1, \sigma_2 \rangle) = \begin{cases} 
        \langle q, 1 \rangle \lor \langle \sigma_2, 2 \rangle & q = \sigma_1 \\
        \langle q, 1 \rangle & \text{otherwise}
    \end{cases}
    \]
  - For any \( \langle q, 2 \rangle \in Q \) and \( \sigma \in \Sigma \), \( \delta(\langle q, 2 \rangle, \sigma) = \langle q, 1 \rangle \)
- \( \alpha = [n] \times \{2\} \)
Figure 3.1: The NBW automaton defining $L_n$. Unlabeled edges can be taken with any $\sigma \in \Sigma$. There is an unlabeled edge from $(0, 0)$ to $(i, 1)$ for any $i \in [n]$. There is an edge labeled with $(i, j)$ from $(i, 1)$ to $(j, 2)$ for any $i, j \in [n]$. There is an unlabeled edge from $(i, 1)$ to $(i, 1)$ and from $(i, 2)$ to $(i, 1)$ for any $i \in [n]$.

We can associate a word $w \in \Sigma^\omega$ with a directed graph where the graph edges are the letters in $\text{inf}(w)$. In this sense, the language $L_n$ is the set of all words for which the associated graph has a directed cycle. The automaton tracks the cycle in the word. The states right-tagged with 1 (i.e. $[n] \times \{1\}$) “remember” the current vertex in the cycle. The self loop of these states allows the automaton to wait for the next edge in the cycle. The automaton cannot wait forever since these states are not in $\alpha$. When the next edge is finally read from the word, the automaton moves to the state, right-tagged with 2, that corresponds to the edge endpoint, from which it has to move to the same state only right-tagged with 1. This two steps move goes through an $\alpha$ state and therefore allows the automaton to accept the word. In the other direction, any accepting run of the automaton goes through this kind of two steps moves infinitely many times. Sequential moves, correspond to edges with common vertex and therefore some vertex has to be visited infinitely many times. Since there is finite number of (simple) cycles, at least one of them has to appear infinitely many times. The following lemma formalize this characterization of $L_n$.

**Lemma 1.** Word $w \in \Sigma^\omega$ is in $L_n$ iff the (directed) graph $G = \langle V,E \rangle$, where $V = [n]$ and $E = \text{inf}(w)$, has a cycle.

**Proof.** Assume $w \in L_n$ and let $A$ be the NBW in definition 8. We will show $G$ has a cycle. Since $w \in L(A)$ there exists an accepting run of $A$ on $w$, $\pi = \pi_0, \pi_1, \ldots$ (note
that since $A$ is nondeterministic the run is given as a sequence of states instead of a tree). Let $\beta \subseteq \alpha$ be the set of accepting states that appear infinitely many times in $\pi$ and denote $\beta = \{\langle \beta_1, 2 \rangle, \ldots, \langle \beta_k, 2 \rangle\}$. By the structure of $A$, for each $\beta_i$ there exists $\beta_j$ and $\beta_l$ such that $\langle \beta_j, \beta_i \rangle, \langle \beta_i, \beta_j \rangle \in \text{inf}(w)$. Observe the subgraph of $G$, $G' = \langle V', E' \rangle$, where $V' = \{\beta_1, \ldots, \beta_k\}$ and $E' = \text{inf}(w)$. Each vertex of this graph has an in-degree greater then zero and an out-degree greater then zero, therefore this graph has a cycle and therefore $G$ has a cycle.

We now assume the graph $G$ has a cycle and we will show $w \in L_n$. Let $\langle \beta_1, \beta_2 \rangle, \langle \beta_2, \beta_3 \rangle, \ldots, \langle \beta_{k-1}, \beta_k \rangle, \langle \beta_k, \beta_1 \rangle \in \text{inf}(w)$ be some edges that form a cycle in $G$. An accepting run of $A$ on $w$ can be constructed in the following way. $\pi = \pi_0, \pi_1^+, \pi_2, \pi_3^+, \ldots$ where,

- $\pi_0 = (0, 0)$ and $\pi_1 = \langle \beta_1, 1 \rangle$
- $\pi_{2i} = \langle \beta_j, 2 \rangle$ and $\pi_{2i+1} = \langle \beta_j, 1 \rangle$ for any $i > 0$ and $j = (i \mod k) + 1$.
- $\pi_i^+$ means the state $\pi_i$ repeats a few times (at least once).

\[\square\]

As Lemma 1 characterizes the language $L_n$ it is easy to see that the language $\overline{L_n}$ is the set of all words for which the associated graph has no cycles. We use this characterization to prove the following lemma.

**Lemma 2.** Any automaton $A \in \text{ABW}$ such that $L(A) = \overline{L_n}$ has at least $\Omega(n \log n)$ states.

Before we prove Lemma 2, let us note the key observation behind it. There is a set of $n!$ words in $\overline{L_n}$ for which “interlacing” any pair in the set results in a word not in $\overline{L_n}$. By interlacing we mean, constructing a new infinite word by taking alternating fragments from the two words. This special set of words is induced from the set of strict total order graphs (i.e., transitive, asymmetric, total and no self-loops) over vertices $[n]$. Each of these graphs can be associated with some finite word $w \in \Sigma^*$ which is an arbitrary enumeration of the graph edges. We then have that the infinite word $w^\omega$ corresponds to the graph from which $w$ was constructed. For any two such graphs and the finite words $w, w'$ associated with them, it is easy to see that the graph corresponding to the “interlaced” infinite word $(ww')^\omega$ is exactly the union of the two strict total order graphs. By the totality of the two graphs, the union of them has two edges of the form $(u, v)$ and $(v, u)$, i.e., the union has a cycle and therefore $(ww')^\omega \not\in \overline{L_n}$ or in other words $(ww')^\omega \in L_n$.

**Proof.** Let $p_1, p_2, \ldots, p_n$ and $s_1, s_2, \ldots, s_n$ be two different permutations of $[n]$. We define $\Sigma_p = \{\langle p_i, p_j \rangle \mid i < j\}$ and $\Sigma_s = \{\langle s_i, s_j \rangle \mid i < j\}$. We define $w^p = (e_1 e_2 \ldots e_k)^\omega$
where \( e_1, e_2, \ldots, e_k \) is some (arbitrary) enumeration of the elements of \( \Sigma_p \), and \( w^s = (e'_1 e'_2 \ldots e'_k)^\omega \) where \( e'_1, e'_2, \ldots, e'_k \) is some (arbitrary) enumeration of the elements of \( \Sigma_s \).

By Lemma 1 we have \( w^p, w^s \in L_n \). Assume \( A = (\Sigma, Q_{\text{init}}, \delta, \alpha) \in \text{ABW} \) such that \( L(A) = L_n \). Let \( \pi^p \) and \( \pi^s \) be accepting memoryless runs of \( A \) on \( w^p \) and \( w^s \), respectively. Observe the subsets of states visited by \( \pi^p \). That is, let \( \Pi^p \) be the states in the \( i \)th level of the run DAG of \( \pi^p \) and let \( \Pi^s \) be the states in the \( i \)th level of the run DAG of \( \pi^s \).

Since the runs are infinite and there are only finite many states, each run has a subset that appears infinitely many times. Let \( \Pi^p \) be such subset for \( \pi^p \) and \( \Pi^s \) such subset for \( \pi^s \).

Assume by way of negation that \( \Pi^p = \Pi^s \). We will show a word \( w \) such that \( w \in L_n \) and \( w \in L(A) \). We will construct \( w \) and the accepting run of \( A \) on \( w \) at the same time by picking alternating segments from \( w^p \) and \( w^s \) with their corresponding segments from \( \pi^p \) and \( \pi^s \).

Let \( i,j \in \mathbb{N} \) such that \( \Pi^p_i = \Pi^p_j = \Pi^p \) and \( i + \binom{n}{2} < j \) and any path in \( \pi^p \) has an accepting state somewhere between the \( i \)th and \( j \)th nodes (note that the last condition can be satisfied by taking big enough \( j \)). Let \( i', j' \in \mathbb{N} \) such that \( \Pi^s_{i'} = \Pi^s_{j'} = \Pi^s \) and \( i' + \binom{n}{2} < j' \). Observe the word \( w = w^p_0 \ldots w^p_{i-1}(w^p_i \ldots w^p_{j-1})^{\omega} \ldots w^s_{j'-1})^{\omega} \). Since \( \inf(w) = \inf(w^p) \cup \inf(w^s) \) the graph associated with \( w \) has a cycle and therefore \( w \in L_n \).

We can construct an accepting run of \( A \) on \( w \) by taking the corresponding steps from \( \pi^p \) and \( \pi^s \), i.e., when reading a letter \( w^p_i \) from \( w \) we take the \( i \)th step from \( \pi^p \) and when reading a letter \( w^s_i \) from \( w \) we take the \( i \)th step from \( \pi^s \).

Since \( L_n \cap L(A) = \emptyset \) the existence of \( w \) is a contradiction and therefore \( \Pi^p \neq \Pi^s \). There are \( n! \) different permutations of \( [n] \), and we can associate a different subset of \( Q \) with each one of them. Therefore \( 2^{|Q|} \geq n! \) and therefore \( |Q| \geq \log n! = \Omega(n \log n) \)

Since we have an ABW (as a super case of NBW) automata family with \( O(n) \) states accepting \( L_n \), it follows immediately from Lemma 2 that ABW complementation has state complexity \( \Omega(n \log n) \).

Since AWW is a special case of ABW and complementing AWW does not change the number of states, we have the following corollary.

**Corollary 1.** Any automaton \( A \in \text{AWW} \) such that \( L(A) = L_n \) has at least \( \Omega(n \log n) \) states.

Finally we can prove the following theorem.

**Theorem 5.** The state complexity of the translation from ABW to AWW is \( \Omega(n \log n) \).

**Proof.** The family of languages given by \( L_n \) is accepted by ABW automata with \( 2n + 1 \) states (by definition 8), yet by corollary 1 any AWW automaton accepting \( L_n \) has at least \( \Omega(n \log n) \) states. \( \square \)
By using the duality principal, any algorithm for translating ABW to AWW can be trivially extended (i.e. without changing the state complexity) to an ABW complementation algorithm. It immediately follows that any upper bound on ABW to AWW translation is also an upper bound for ABW complementation, and any lower bound on ABW complementation is also a lower bound on ABW to AWW translation. Our current state is that we have an $O(n^2)$ algorithm for the ABW to AWW translation and an $\Omega(n \log n)$ bound on ABW complementation.

The proof of Theorem 5 can be generalized in the following way. Let $L'_n$ be a language family that can be accepted by NBW automata with $O(n)$ states and $L'_n$ has a set of $m$ words for which “interlacing” any pair in the set results in a word not in $L'_n$. By replacing $L_n$ in the proof above with $L'_n$ we have that NBW to AWW translation is $\Omega(\log m)$. The exact same generalization works for NBW complementation. A Language family, $L'_n$ as above implies NBW complementation is $\Omega(m)$ [18]. We would like to find $L'_n$ with $m = 2^{n^2}$, but that would imply NBW complementation is $\Omega(2^{n^2})$ and that is a contradiction since we already know NBW complementation to be $O(2^{n \log n})$ by Safra’s construction [30]. Note that if we ask $L'_n$ to be accepted by ABW automata with $O(n)$ states instead of NBW automata, we can still use this technique to prove an $\Omega(n^2)$ bound on ABW to AWW translation.

The part of the proof we can still improve is when we move from lower bound on the number of subsets of states to lower bound on the number of states. In the proof of Lemma 2 we find the number of subsets of states is bounded by $n!$. We then conclude the number of state is bounded by $\log n!$. This conclusion uses the most permissive assumption on subsets, i.e., $|\mathcal{P}(Q)| = 2^{|Q|}$. If we can show more restrictions on the subsets we might be able to improve the lower bound. These restrictions should stem from the special structure of the target automata class. Since AWW has a more constraint structure then ABW, it should be “easier” finding restrictions on the subsets for that class.

### 3.3 Lower bound for AWWX

In this section we present a subclass of AWW for which we can prove that any automaton accepting $L_n$ has $\Omega(n^2)$ states. We hope this exercise will give insights on how we can do the same for the full class AWW and even ABW.

**Definition 9.** We say AWW automaton $A = (\Sigma, Q, q_{init}, \delta, \alpha)$ is a AWWX if $\delta(q, \sigma)$ is either $q$ or $\perp$, for every $q \in \alpha$ and $\sigma \in \Sigma$.

First let us show an AWWX automaton with $O(n^2)$ states that accepts $L_n$. We define $A_n = (\Sigma, Q, q_{init}, \delta, \alpha)$ where.

- $\Sigma = [n] \times [n]$
• $Q = \{0\} \cup [n] \times [n]$
• $q_{init} = 0$
• For any $\sigma \in \Sigma$,
  $$\delta(0, \sigma) = 0 \lor \bigvee_{(\{n\}, E) \in ST} \bigwedge_{e \in [n] \times [n] \setminus E} e$$

where $ST$ is the set of all strict total order graphs\(^1\) over vertices $[n]$. In other words, $\delta(0, \sigma)$ can choose to stay in the initial state (0), or pick a strict total order graph and then move to all the states encoded by edges not in the graph.

For any $q \in [n] \times [n]$ and $\sigma \in \Sigma$,
  $$\delta(q, \sigma) = \begin{cases} 
\bot & q = \sigma \\
q & \text{otherwise}
\end{cases}$$

• $\alpha = [n] \times [n]$

The partition of $Q$ to $Q \setminus \{0\}, \{0\}$ satisfies the weak condition and therefore $A_n \in AWW$. It is also easy to see that for every $q \in \alpha$ and $\sigma \in \Sigma$, $\delta(q, \sigma)$ is either $q$ or $\bot$ and therefore $A_n \in AWWX$. To see that $L(A_n) = \Gamma_n$, consider some $w \in \Gamma_n$. By definition, the graph associated with $w$ has no cycles and therefore is a subgraph of some strict total order graph. Moreover, there exists some $m \in \mathbb{N}$ such that $w_i \in \inf(w)$ for every $i \geq m$. It is now easy to see $A_n$ can accept $w$ by staying in $q_{init}$ while reading the first $m$ letters and then using the conjunct associated with the strict total order graph subsuming $\inf(w)$ to move to the $\alpha$ partition, which it will never leave from that point. In the other direction, if $A_n$ accepts some word $w$, then it must have reached the $\alpha$ partition by using one of the conjuncts associated with a strict total order graph and therefore the graph associated with $w$ is a subgraph of that strict total order and therefore has no cycles and therefore $w \in \Gamma_n$.

To show that any automaton $A \in AWWX$ such that $L(A) = \Gamma_n$ has $\Omega(n^2)$ states we first need to observe some results in combinatorics. We will denote the set of permutation of $[n]$ by $[n]!$. For permutation $p = \langle p_1, p_2, \ldots, p_n \rangle \in [n]!$ we define $\Sigma_p = \{\langle p_i, p_j \rangle \mid i < j\}$. For every $p \in [n]!$ we define $w_p = e_1 e_2 \ldots e_k \in \Sigma^+$ where $e_1, e_2, \ldots, e_k$ is some (arbitrary) enumeration of the elements of $\Sigma_p$. The function $f_n : \mathcal{P}(\Sigma) \rightarrow \mathbb{N}$ is defined $f_n(E) = |\{p \in [n]! \mid E \subseteq \Sigma_p\}|$ for $E \in \mathcal{P}(\Sigma)$.

**Lemma 3.** $f_n(E) \leq \frac{n!}{|E|+1}$ for every $E \in \mathcal{P}(\Sigma)$.

**Proof.** We prove the lemma by induction on $n + |E|$. Base: assume $n + |E| = 0$. Since $E = \emptyset$ we have that $E \subseteq \Sigma_p$ for any $p \in [n]!$ and therefore $f_n(E) = n!$. Moreover,

\[\text{Technion - Computer Science Department - M.Sc. Thesis MSC-2013-13 - 2013}\]

\[\text{\textsuperscript{1}A strict total order graph is a transitive, antisymmetric, and total graph with no self loops.}\]
If \( n! = \frac{n!}{|E|+1} \), then \( f_n(E) \leq \frac{n!}{|E|+1} \). Step: Let \( E \) and \( n \), and assume (I.H.) for any \( E' \) and \( n' \) such that \( |E'| \leq |E| \) and \( n' < n \) we have \( f_{n'}(E') \leq \frac{n!}{|E'|+1} \). We observe two cases, \( n-1 > |E| \) and \( n-1 \leq E \). Assume \( n-1 > |E| \): the graph \( G = ([n], E) \) is not connected. Let \( E' \subseteq E \) be a non-empty connected component of \( E \) and \( V' \) the set of \([n]\) vertices induced by \( E' \). Let \( E'' = E \setminus E' \) and \( V'' \) be the complements of \( E' \) and \( V' \), i.e. \( E'' = E \setminus E' \) and \( V'' = [n] \setminus V' \). By the I.H. we have \( f_{|V'|}(E') \leq \frac{|V''|!}{|E'|+1} \) and \( f_{|V''|}(E'') \leq \frac{|V'|!}{|E'|+1} \). We calculate \( n! = \frac{n!}{|E|+1} \cdot \frac{1}{|E'|+1} \cdot n! = \frac{|V'|!}{|E'|+1} \cdot \frac{|V''|!}{|E'|+1} \cdot (|V'|) \geq f_{|V'|}(E') \cdot f_{|V''|}(E'') \cdot (|V'|) \geq f_n(E). \) Assume \( n-1 \leq E \): let \( v \in [n] \) such that \((u,v) \notin E \) for any \( u \in [n] \) (if no such \( v \) exists then \( E \) cannot be a subset of any \( \Sigma \) and therefore \( f_n(E) = 0 \)). Let \( E' = \{(v,u) \in \Sigma \mid (v,u) \in E\} \). \( n-1 \leq E \Rightarrow n \leq |E| + 1 \Rightarrow -|E'| \leq -|E'| + |E'|- |E'| \Rightarrow n|E'| + |E'| + n = n|E'| + n - |E'||E'|- |E'| \Rightarrow n(|E'| + 1) \geq (|E'|-1)(|E'|) + 1 \Rightarrow \frac{n!}{|E'|+1} \geq \frac{1}{|E'|+1} \cdot (n-|E'|) \Rightarrow \frac{n!}{|E'|+1} \geq \frac{(n-1)!}{|E'|+1} \cdot (n-|E'|). \) Therefore \( \frac{n!}{|E'|+1} \geq \frac{(n-1)!}{|E'|+1} \cdot (n-|E'|) \geq f_{n-1}(E \setminus E') \cdot (n-|E'|) \geq f_n(E). \) \( \square \)

We say that \( E_0, E_1, \ldots, E_{k-1} \in \mathcal{P}(\Sigma) \) are \( n \)-cover of size \( k \) if for any permutation \( p \in [n]! \) there exists \( i_1, i_2, \ldots, i_m \in [k] \) such that \( \Sigma_p = E_{i_1} \cup E_{i_2} \cdots \cup E_{i_m} \).

**Theorem 6.** If \( E_0, E_1, \ldots, E_{k-1} \in \mathcal{P}(\Sigma) \) are \( n \)-cover then \( k \geq \left( \begin{array}{c} n \end{array} \right) \).

**Proof.** We calculate \( k \cdot n! \geq \sum_{i=0}^{k-1} (|E_i| \cdot \frac{n!}{|E|+1}) \geq \sum_{i=0}^{k-1} (|E_i| \cdot f_n(E_i)) \geq \left( \begin{array}{c} n \end{array} \right) \cdot n! \). And therefore \( k \geq \left( \begin{array}{c} n \end{array} \right) \). \( \square \)

For an AWXX automaton, each state \( q \in \alpha \) can be associated with the set of letters that lead to \( \bot \) from that state, \( \Sigma_q = \{ \sigma \in \Sigma \mid \delta(q, \sigma) = \bot \} \). If the automaton accepts \( L_n \) the set of \( \Sigma_q \) sets has to be \( n \)-cover. By Theorem 6 we conclude the automaton has \( \Omega(n^2) \) states.

**Theorem 7.** Any automaton \( A \in \mathcal{AWX}X \) such that \( \mathcal{L}(A) = \underline{L_n} \) has \( \Omega(n^2) \) states.

**Proof.** Let \( A = (\Sigma, Q, q_{\text{init}}, \delta, \alpha) \) be an AWXX automaton such that \( \mathcal{L}(A) = \underline{L_n} \). Let \( q_1, \ldots, q_k \) be some enumeration of the elements of \( \alpha \). It is enough to show that \( \Sigma_{q_1}, \ldots, \Sigma_{q_k} \) are \( n \)-cover and then by Theorem 6 we have that \( |\alpha| = k \geq \left( \begin{array}{c} n \end{array} \right) \) and therefore \( |Q| = \Omega(n^2) \). For every permutation \( p \in [n]! \) let \( \overline{p} \) be the reverse permutation of \( p \) in \([n]! \) (note that \( \Sigma_p = \Sigma \setminus \Sigma_p \)). Let \( p \in [n]! \). By the definition of \( L_n \) we have \((w_p)^n \in \underline{L_n} \). Consider an accepting run of \( A \) on \((w_p)^n \). \( \Pi_0, \Pi_1, \ldots \) be the subsets of states the run visits. By the structure of AWXX automaton, exists \( i \) such that \( \Pi_i = \Pi_j \) for every \( j > i \) and \( \Pi_i \subseteq \alpha \). Let \( \pi_1, \ldots, \pi_t \) be some enumeration of the states in \( \Pi_i \). We have that \( \bigcup_{q \in \Pi_i} \Sigma_q = \Sigma \setminus \Sigma_p = \Sigma_p \). Therefore \( \Sigma_{q_1}, \ldots, \Sigma_{q_k} \) are \( n \)-cover and \( |Q| = \Omega(n^2) \). \( \square \)
3.4 Lower bound for $\overline{L}_n^k$

$L_n$ is the language of all words for which the associated graph $(G = \langle [n], \inf(w) \rangle)$ has a cycle. Let $L_n^k$ be the language of all words for which the associated graph has a cycle of length $k$ or shorter (e.g., $L_n^2 = L_n$). The language $L_n^2$ is of special interest since we can prove any ABW automaton accepting $L_n^2$ has at least $\Omega(n^2)$ states.

We say a graph is $k$-maximal if it has no cycle of length $k$ or shorter and adding to it any edge will create a cycle of length $k$ or shorter. We can associate a word in $\Sigma^+$ with each 2-maximal graph like we did with permutations (i.e., a word $w \in \Sigma^+$ such that $\inf(w^\omega) = E$). And just like with permutations, for two 2-maximal graphs $G_1, G_2$ and their associated words $w_1, w_2$, we can show the accepting runs of an ABW automaton accepting $L_n^2$ on $(w_1)^\omega$ and $(w_2)^\omega$ are associated with different subsets of states. Since there are $2^{\binom{n}{2}}$ 2-maximal graphs over $n$ vertices we conclude any ABW automaton accepting $L_n^2$ has at least $\Omega(n^2)$ states.

Since $\overline{L}_n^2$ has the $\Omega(n^2)$ bound, if we think $\overline{L}_n$ (i.e., $\overline{L}_n^0$) also has the $\Omega(n^2)$ bound it is tempting to believe the same will be true for any $\overline{L}_n^k$ where $2 < k < n$. For each $2 < k \leq n$ we can give a proof, similar to that given above for $k = 2$, which will show any ABW automaton accepting $\overline{L}_n^k$ has at least $\log m_k$ states where $m_k$ is the number of $k$-maximal graphs (note that $m_n = n!$). If we believe $\Omega(n^2)$ is a bound for all the $\overline{L}_n^k$ then maybe there is a gradual way to strengthen the $\log m_k$ bound.

To complete the picture we describe an AWW automaton that accepts $\overline{L}_n^k$. The AWW automaton $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ is defined as follows,

1. $\Sigma = [n] \times [n]$
2. $Q = \{0\} \cup \Sigma$
3. $q_{\text{init}} = 0$
4. for every $\sigma \in \Sigma$, $\delta(q_{\text{init}}, \sigma) = 0 \lor \bigvee_{G = \langle V, E \rangle \in G_n^k} (\bigwedge_{e \in \Sigma \setminus E} e)$ where $G_n^k$ is the set of all graphs over vertices $V = [n]$ and no cycles of length $k$ or shorter.
5. for every $q \in \Sigma$ and $\sigma \in \Sigma$, $\delta(q, \sigma) = \begin{cases} 1 & q = \sigma \\ q & \text{otherwise} \end{cases}$
6. $\alpha = \Sigma$

It is easy to see the partitioning of $Q$ to $\Sigma$, $\{0\}$ satisfies the weak condition and therefore $A \in \text{AWW}$. Assume a word $w \in \Sigma^\omega$ is accepted by $A$ and $(T, r)$ is an accepting run. Since $q_{\text{init}} \notin \alpha$ the run has to leave $q_{\text{init}}$ at some point. Therefore there exists some $G = \langle V, E \rangle \in G_n^k$ and $i \in \mathbb{N}$ such that $\{w_1, w_{i+1}, \ldots\} \cap (\Sigma \setminus E) = \emptyset$ which implies $\inf(w) \subseteq E$ and therefore the graph associated with $w$ is a subgraph of $G$ and therefore has no cycles of length $k$ or shorter.
Let \( w \in \overline{L^k_n} \). By definition of \( L^k_n \), the graph \( G = (\{v\}, \inf(w)) \) has no cycles of length \( k \) or shorter. Since \( \Sigma \) is finite there exists \( i \in \mathbb{N} \) such that \( w_j \in \inf(w) \) for all \( j \leq i \). We can construct an accepting run of \( A \) on \( w \) by staying at \( q_{\text{init}} \) for the first \( i \) steps and then moving to the subset of states associated with \( G \).

We note that the transition function of the automaton can be optimized a bit by considering only \( k \)-maximal graphs (since any graph with no cycles of length \( k \) or shorter is a subgraph of some \( k \)-maximal graph).

### 3.5 Restricted AWW

The resulting AWW automaton of the construction in the proof of Theorem 3 has an interesting property. The transition function can use \( \land \) only when applied to accepting states. That is, the full power of alternation (using both \( \land \) and \( \lor \)) is restricted to accepting states, while the transition from non-accepting states is restricted to non-deterministic (i.e. uses only \( \lor \)). Since NBW is \( \omega \)-regular, the construction entails this restricted version of AWW is also \( \omega \)-regular. In this section we show that if we restrict AWW to use alternation only for the transition from non-accepting states and non-deterministic for accepting states the expression power reduces to that of NWW.

Formally, we say an AWW automaton \( A = (\Sigma, Q, q_{\text{init}}, \delta, \alpha) \) is in \( N^\land \text{WW} \) if for every \( q \in Q \setminus \alpha \) and \( \sigma \in \Sigma \), \( \delta(q,\sigma) \) is restricted to variables and \( \lor \). We say \( A \) is in \( N^\land \text{WW} \) if for every \( q \in \alpha \) and \( \sigma \in \Sigma \), \( \delta(q,\sigma) \) is restricted to variables and \( \lor \).

As we already mentioned, \( N^\land \text{WW} \) is \( \omega \)-regular, yet \( N^\land \text{WW} \) has the same expression power as NWW (which is the same as NCW). Before we state it formally, let us observe an important property of \( N^\land \text{WW} \) accepting run.

**Lemma 4.** Let \( A = (\Sigma, Q, q_{\text{init}}, \delta, \alpha) \) be an \( N^\land \text{WW} \) automaton and let \( \langle T, r \rangle \) a run of \( A \). \( \langle T, r \rangle \) is an accepting run of \( A \) iff for some \( k \in \mathbb{N} \), \( r(x) \in \alpha \) for all \( x \in T \) such that \( |x| \geq k \).

**Proof.** The \( \Leftarrow \) direction is trivial. We bring the outline of the \( \Rightarrow \) direction. Assume for some accepting run, \( \langle T, r \rangle \), no such \( k \) exists, therefore there are infinitely many \( x \in T \) such that \( r(x) \notin \alpha \). We transform \( T \) by removing sub-paths of \( \alpha \) states. For example, a path \( \ldots, x_1, x_2, x_3, x_4, \ldots \) where \( r(x_1), r(x_4) \notin \alpha \) and \( r(x_2), r(x_3) \in \alpha \) is reduced to \( \ldots, x_1, x_4, \ldots \). This transformation preserves the property of finite non-\( \alpha \) states in every path, i.e., if every path in \( T \) has finite many nodes, \( x \), such that \( r(x) \notin \alpha \) then the same is true after the transformation. Moreover, due to the restriction on \( \delta(q,\sigma) \) for \( q \in \alpha \) we also have that the transformation preserves the finite degree, i.e., if each node in \( T \) has a finite degree then the same is true after the transformation. We can now use König's lemma to conclude the transformed \( T \) has an infinite path of non-accepting states. By the first preserved property we can conclude \( T \) has a path with infinite many non-accepting states. That is a contradiction to \( \langle T, r \rangle \) being an accepting run and \( A \) being weak. \( \square \)
Theorem 8. The expressive power of $\mathcal{N}_A \mathcal{W} \mathcal{W}$ is the same as $\mathcal{N}_A \mathcal{W} \mathcal{W}$.

Proof. Since $\mathcal{N}_A \mathcal{W} \mathcal{W}$ is a special case of $\mathcal{N}_A \mathcal{W} \mathcal{W}$ we have that $L(\mathcal{N}_A \mathcal{W} \mathcal{W}) \subseteq L\left(\mathcal{N}_A \mathcal{W} \mathcal{W}\right)$. Given an $\mathcal{N}_A \mathcal{W} \mathcal{W}$ automaton $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ we will construct a $\mathcal{N}_A \mathcal{W} \mathcal{W}$ automaton $A' = \langle \Sigma, Q', q'_{\text{init}}, \delta', \alpha' \rangle$ where,

- $Q' = \mathcal{P}(Q) \times \{1\} \cup \mathcal{P}(\alpha) \times \{2\}$
- $q'_{\text{init}} = \langle \{q_{\text{init}}\}, 1 \rangle$
- For every $S \subseteq Q$ and $\sigma \in \Sigma$,
  
  \[
  \delta'(\langle S, 1 \rangle, \sigma) = \bigvee_{S' \models \bigwedge_{q \in S} \delta(q, \sigma)} \langle S', 1 \rangle \lor \bigvee_{S' \models \bigwedge_{q \in S} \delta(q, \sigma), S' \subseteq \alpha} \langle S', 2 \rangle \lor \bot
  \]

- For every $S \subseteq \alpha$ and $\sigma \in \Sigma$,
  
  \[
  \delta'(\langle S, 2 \rangle, \sigma) = \bigvee_{S' \models \bigwedge_{q \in S} \delta(q, \sigma), S' \subseteq \alpha} \langle S', 2 \rangle \lor \bot
  \]

- $\alpha' = \mathcal{P}(\alpha) \times \{2\}$

(Note that $\delta(\langle \emptyset, 1 \rangle, \sigma) = \langle \emptyset, 2 \rangle \lor \ldots$ and $\delta(\langle \emptyset, 2 \rangle, \sigma) = \langle \emptyset, 2 \rangle \lor \ldots$. Therefore, any appearance of $\langle \emptyset, 1 \rangle$ and $\langle \emptyset, 2 \rangle$ in $\delta$ can be replaced with $\top$)

The partitioning of $Q'$ to $\mathcal{P}(\alpha) \times \{2\},\mathcal{P}(Q) \times \{1\}$ satisfies the weak condition and therefore $A' \in \mathcal{N}_A \mathcal{W} \mathcal{W}$. In fact it is easy to see $A' \in \mathcal{N}_A \mathcal{W} \mathcal{W}[3]$ (note that $\delta'$ uses $\bot$ and therefore we have to increase the number of partition by 1).

The construction is a simple subset construction and $L(A') \subseteq L(A)$ is immediate. By Lemma 4, it is easy to see that any accepting run of $A$, after some $k$ steps, reach a subset of states which are all accepting and all following subsets of states are also all accepting. Therefore $L(A) \subseteq L(A')$. \qed
Chapter 4

DBW to DWW

In chapter 3 we discussed the translation from ABW to AWW. In this chapter we discuss the deterministic analog, the translation from DBW to DWW. ABW can always be translated to AWW, both classes have the same expression power, both of them are \( \omega \)-regular. In fact every ABW automaton can be translated to AWW[3] automaton. The deterministic case is very different in that sense. Not every DBW automaton can be translated to DWW automaton. Moreover, the expression power of DWW\([k]\) and DWW\([k+1]\) is not the same. The translation from ABW to AWW involves \( \Omega(n \log n) \) state complexity. In the deterministic case, we prove DBW is DWW\([k]\)-typed, i.e., when possible the translation involves nothing but a change to the \( \alpha \) part of the automaton.

4.1 The DWW hierarchy

The deterministic nature of deterministic automata means after reading some prefix of a word, the automata can only reach one state. We use this observation to extend the transition function of DWW automata to finite words (i.e. prefix of infinite word). Let \( A = \langle \Sigma, Q, q_{init}, \delta, \alpha \rangle \) be a DBW automaton. Given a state \( q \in Q \) and a word \( w = w_0w_1 \ldots w_n \in \Sigma^+ \) we define \( \delta(q, w) \) (extension of the transition function) by recursion on the length of \( w \),

**Base:** for \( w = w_0 \), \( \delta(q, w) = \delta(q, w_0) \) (note that by definition of DBW \( \delta(q, w_0) \in Q \)).

**Closure:** for \( w = w_0w_1 \ldots w_{n+1} \), assume \( \delta(q, w_0w_1 \ldots w_n) = q' \), \( \delta(q, w) = \delta(q', w_{n+1}) \).

For two states \( q, p \in Q \) we say \( p \) is reachable from \( q \) if there exists a word \( w \in \Sigma^+ \) such that \( \delta(q, w) = p \).

The states of deterministic B"uchi automaton can be divided into two categories. States that can reach themselves and states that cannot. States of the later kind can appear at most once in a run and therefore, whether they are members of \( \alpha \) or not, do
not affect the language of the automaton. In [19], Löding defines these categories as follows.

**Definition 10 ([19])**. Let $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ be a DWW automaton. A state $q \in Q$ is called **recurrent** (in $A$) iff there exists a word $w \in \Sigma^+$ such that $\delta(q, w) = q$. Otherwise $q$ is called **transient**.

Given a DWW automaton and a sequence of recurrent states which are reachable from one another and are alternating between accepting and non-accepting, we can construct a word for every DWW automaton with the same language, that will force it to visit a similar sequence of alternating states.

**Lemma 5.** Let $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ be a DWW automaton. If there exists a sequence of recurrent states $q_0, q_1, \ldots, q_n$ in $Q$ such that $q_0$ is reachable from $q_{\text{init}}$, $q_{i+1}$ is reachable from $q_i$ and $q_{i+1} \in \alpha \iff q_i \notin \alpha$ for every $i \in [n]$, then for any DWW automaton $A' = \langle \Sigma, Q', q'_{\text{init}}, \delta', \alpha' \rangle$ such that $L(A) = L(A')$ there exists a sequence of states $q'_0, q'_1, \ldots, q'_n$ in $Q'$ such that $q'_{i+1}$ is reachable from $q'_i$ for every $i \in [n]$ and $q'_i \in \alpha' \iff q_i \in \alpha$ for every $i \in [n+1]$.

The word we construct to force $A'$ to visit a similar sequence of states has the form $u_0v_0u_1v_1 \ldots u_nv_n$ where $u_i, v_i \in \Sigma^*$. The definition of the $u_i, v_i$ words is done by observing both $A$ and $A'$. The $u_i$ words are used to make the run of $A$ move from $q_{i-1}$ to $q_i$. Since $q_i$ is a recurrent state we can make the run of $A$ loop and come back to $q_i$. If we loop long enough, the run of $A'$ has to reach a state similar to $q_i$ (i.e., accepting iff $q_i$ is accepting). $v_i$ is the looping part that forces $A'$ to reach that state.

**Proof.** Let $A$ and $q_0, q_1, \ldots, q_n$ be as stated in the lemma and assume $A' = \langle \Sigma, Q', q'_{\text{init}}, \delta', \alpha' \rangle$ is a DWW automaton such that $L(A) = L(A')$. We define a word $u_0v_0u_1v_1 \ldots u_nv_n$ where $u_i, v_i \in \Sigma^*$ for every $i \in [n+1]$, by induction on $n$, such that $\delta(q_{\text{init}}, u_0v_0u_1v_1 \ldots u_nv_n) = q_m$ and $q_m$ is reachable from $\delta(q_{\text{init}}, u_0v_0u_1v_1 \ldots u_nv_n)$ and $\delta'(q'_{\text{init}}, u_0v_0u_1v_1 \ldots u_nv_n) \in \alpha' \iff q_m \in \alpha$ for every $m \in [n+1]$.

**Base:** We set $u_0$ such that $\delta(q_{\text{init}}, u_0) = q_0$. By definition of recurrent state, there exists some $w \in \Sigma^+$ such that $\delta(q_0, w) = q_0$. If $q_0 \in \alpha$, we have that $u_0(w^\omega) \in L(A)$ and therefore there exists a prefix of $w^\omega$, denoted $v_0$, such that $\delta'(q'_{\text{init}}, u_0v_0) \notin \alpha'$. Otherwise, $q_0 \notin \alpha$, all the states in the run of $\delta(q_0, w)$ are in the same partition as $q_0$ and therefore none of them is in $\alpha$. Therefore $u_0(w^\omega) \notin L(A)$ and therefore there exists a prefix of $w^\omega$, denoted $v_0$, such that $\delta'(q'_{\text{init}}, u_0v_0) \notin \alpha'$.

**Step:** By the IH, for the sequence $q_0, \ldots, q_{n-1}$, we have $u_0v_0u_1v_1 \ldots u_{n-1}v_{n-1}$ such that $\delta(q_{\text{init}}, u_0v_0u_1v_1 \ldots u_nv_n) = q_m$ and $q_m$ is reachable from $\delta(q_{\text{init}}, u_0v_0u_1v_1 \ldots u_nv_n)$ and $\delta'(q'_{\text{init}}, u_0v_0u_1v_1 \ldots u_nv_n) \in \alpha' \iff q_m \in \alpha$ for every $m \in [n]$. We will show $u_n, v_n$ such that $u_0v_0 \ldots u_{n-1}v_{n-1}u_nv_n$ is as stated in the lemma. Since $q_{n-1}$
Figure 4.1: DWW[4] automaton from the proof of Theorem 9

is reachable from $\delta(q_{\text{init}}, u_0v_0 \ldots u_{n-1}v_{n-1})$ and $q_n$ is reachable from $q_{n-1}$, there exists a word $u_n \in \Sigma^*$ such that $\delta(q_{\text{init}}, u_0v_0 \ldots u_{n-1}v_{n-1}u_n) = q_n$. By definition of recurrent state, there exists some $w \in \Sigma^+$ such that $\delta(q_n, w) = q_n$. If $q_n \in \alpha$, we have that $u_0v_0 \ldots u_{n-1}v_{n-1}u_n(w^\omega) \in \mathcal{L}(A)$ and therefore there exists a prefix of $w^\omega$, denoted $v_n$, such that $\delta'(q'_{\text{init}}, u_0v_0 \ldots u_{n-1}v_{n-1}u_nv_n) \in \alpha'$. Otherwise, $q_n \notin \alpha$, all the states in the run of $\delta(q_n, w)$ are in the same partition as $q_n$ and therefore non of them is in $\alpha$. Therefore $u_0v_0 \ldots u_{n-1}v_{n-1}u_n(w^\omega) \notin \mathcal{L}(A)$ and therefore there exists a prefix of $w^\omega$, denoted $v_n$, such that $\delta'(q'_{\text{init}}, u_0v_0 \ldots u_{n-1}v_{n-1}u_nv_n) \notin \alpha'$.

Observe the sequence of $Q'$ states $q'_0 = \delta'(q'_{\text{init}}, u_0v_0), q'_1 = \delta'(q'_{\text{init}}, u_0v_0u_1v_1), \ldots, q'_n = \delta'(q'_{\text{init}}, u_0v_0 \ldots u_nv_n)$. Clearly $q'_{i+1}$ is reachable from $q'_i$ for every $i \in [n]$ and by the induction we have that $q'_i \in \alpha' \iff q_i \in \alpha$ for every $i \in [n+1]$.

Lemma 5 implies the different DWW[k] classes form a strict hierarchy. That is, a class with greater $k$ has more expression power then other classes with lesser $k$.

**Theorem 9.** $\mathcal{L}(\text{DWW}[n]) \subsetneq \mathcal{L}(\text{DWW}[n+1])$ for any $n \in \mathbb{N}$.

**Proof.** Let $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ be DWW[k + 1] automaton where,

- $\Sigma = \{0, 1\}$
- $Q = [k+1]$  
- $q_{\text{init}} = 0$
- For any $q \in Q$ and $\sigma \in \Sigma$: $\delta(q, \sigma) = \begin{cases} q + 1 & q \not\equiv \sigma \pmod{2} \text{ and } q < k \\ q & \text{otherwise} \end{cases}$
- $\alpha = \{q \in Q \mid q \text{ is odd}\}$

Figure 4.1 illustrates the automaton $A$ for $k = 3$.

The partitioning of $Q$ to $\{k\}, \{k-1\}, \ldots, \{0\}$ satisfies the weak conditions and therefore $A \in \text{DWW}[k + 1]$. 

---

Technion - Computer Science Department - M.Sc. Thesis MSC-2013-13 - 2013
We now prove that any DWW automaton accepting $L(A)$ has at least $k+1$ partitions and therefore cannot be in $\text{DWW}[k]$. Let $A'$ be a DWW automaton such that $L(A') = L(A)$. Denote the sequence of $A$ states $0, 1, \ldots, k$ by $q_0, q_1, \ldots, q_k$ (i.e. $q_0 = 0, q_1 = 1, \ldots, q_k = k$). Clearly the sequence satisfy the condition of Lemma 5 and therefore there exists a sequence of states, $q'_0, q'_1, \ldots, q'_k$ in $Q'$ such that $q'_{i+1}$ is reachable from $q'_i$ for every $i \in [k]$ and $q'_i \in \alpha'$ for every $i \in [k+1]$. Since the states in the sequence $q'_0, q'_1, \ldots, q'_k$ alternate between accepting and non-accepting, each one of them has to be in a different partition. Therefore $A'$ has at least $k+1$ partitions and $A' \notin \text{DWW}[k]$. \hfill \square

4.2 DBW is DWW$[k]$-type

In this section we show that if DBW automaton can be translated to DWW$[k]$ automaton then the translation involves only the modification of the $\alpha$ part.

Let $A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle$ be a DWW automaton. The following definition and two remarks are given by Löding.

**Definition 11** ([19]). A mapping $c : Q \rightarrow \mathbb{N}$ is called $A$-coloring iff,

1. $c(q)$ is even for every recurrent state $q \in \alpha$, and
2. $c(q)$ is odd for every recurrent state $q \notin \alpha$, and
3. $c(p) \leq c(q)$ for every $p, q \in Q$ with $\delta(p, a) = q$ for some $a \in \Sigma$.

For $k \in \mathbb{N}$, $c$ is called $k$-maximal iff,

4. $c(q) \leq k$ for every $q \in Q$, and
5. $c'(q) \leq c(q)$ for every $A$-coloring $c' : Q \rightarrow [k+1]$ and $q \in Q$.

**Remark 1** ([19]). Let $c$ be a $k$-maximal $A$-coloring for some $k \in \mathbb{N}$. For every $p \in Q$ with $c(p) \leq k - 2$ there exists $q \in Q$ such that $q$ is reachable from $p$ and $c(q) = c(p) + 1$.

**Remark 2** ([19]). Let $c$ be a $k$-maximal $A$-coloring for some $k \in \mathbb{N}$. For each $q \in Q$ there exists $w \in \Sigma^*$ such that the run of $A$ on $w$ starting from $q$ has only states with the color $c(q)$.

$A$-coloring induces partitioning of the states of $A$ where each color defines a different partition. This partitioning does not satisfies the weak condition since a single partition might have both accepting and non-accepting states. But if we look only on recurrent states, conditions 1 and 2 of coloring insures partitions arising from even colors have only accepting recurrent states, and partitions arising from odd colors have only non-accepting recurrent states. Recall that changing the acceptance of transient states does
not affect the language of the automaton. This observation allows us to simply change the acceptance of some transient states in order to make the partitioning induced by the $A$-coloring satisfy the weak condition.

**Lemma 6.** Let $A = \langle \Sigma, Q, q_{init}, \delta, \alpha \rangle$ be a DWW automaton and let $c : Q \rightarrow \mathbb{N}$ be an $A$-coloring. The automaton $A' = \langle \Sigma, Q, q_{init}, \delta, \alpha' \rangle$ where $\alpha' = \{ q \in Q \mid c(q) \text{ is even} \}$, is DWW and $L(A') = L(A)$.

**Proof.** Let $A$, $c$ and $A'$ be as stated in the lemma. We will first prove $L(A') = L(A)$. Let $w \in \Sigma^\omega$ and let $\pi$ be the run of $A$ (and $A'$) on $w$. By definition of transient, we have that $\inf (\pi)$ has no transient states. By the definition of $A$-coloring we have that $q \in \alpha$ iff $q \in \alpha'$ for every recurrent state $q \in Q$. Therefore $\inf (\pi) \cap \alpha = \inf (\pi) \cap \alpha'$, and therefore $w \in L(A)$ iff $w \in L(A')$.

We now show $A'$ satisfies the weak condition. Consider the following partitioning, 
\[ \{ q \in Q \mid c(q) = k \}, \{ q \in Q \mid c(q) = k - 1 \}, \ldots, \{ q \in Q \mid c(q) = c(q_{init}) \} \] where $k$ is the maximal color used by $c$. Condition 2 of the weak condition (definition 7) are satisfied by the definition of $A$-coloring. Condition 1 is satisfied by the definition of $\alpha'$.

As we saw, $A$-coloring induces partitioning for a given automaton. In the following lemma we show the partitioning induced by the $k$-maximal $A$-coloring with minimal $k$ has the smallest number of partitions possible for any DWW automaton accepting the same language.

**Lemma 7.** Let $A = \langle \Sigma, Q, q_{init}, \delta, \alpha \rangle$ be a DWW automaton. Let $k \in \mathbb{N}$ be the minimal number for which there exists a $k$-maximal $A$-coloring, $c : Q \rightarrow \mathbb{N}$. There exists $\alpha'$ such that $A' = \langle \Sigma, Q, q_{init}, \delta, \alpha' \rangle \in \text{DWW}[k - c(q_{init}) + 1]$ and $L(A') = L(A)$. Moreover, $L(A) \not\in L(\text{DWW}[k - c(q_{init})])$.

Before we present the formal proof, let us observe a cardinal example. The DWW[4] automaton in Figure 4.2 demonstrates the difficulties in Lemma 7. We first observe state 0. Since it is a transient state, whether it is an accepting state or not does not effect the language of the automaton (hence the dashed circle). But when we consider the minimal $k$-maximal $A$-coloring we want to have $c(0) = 0$. Since even colors are associated with accepting states it is beneficial to have 0 as accepting state. We would now want to find a sequence of 4 states as in Lemma 5. Such a sequence will immediately imply any DWW automaton accepting the same language has at least 4 partitions. clearly no such sequence exists in this example. When no such sequence exists, as in this example, we will show two sequences of length $k - c(q_{init})$ (3 in our example) such that one of them start with an accepting state and the second with a non-accepting state. In our example these sequences are $2, 4, 6$ and $1, 3, 5$. Note that such two sequences imply that the automaton has at least $k - c(q_{init}) + 1$ partitions. Finally, by Lemma 5, any DWW
automaton accepting the same language also has two such sequences and therefore has at least \( k - c(q_{\text{init}}) + 1 \) partitions.

**Proof.** Let \( A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle \) be a DWW automaton and \( c : Q \to \mathbb{N} \) a \( k \)-maximal \( A \)-coloring for minimal \( k \in \mathbb{N} \). We define \( \alpha' = \{ q \in Q \mid c(q) \text{ is even} \} \) and \( A' = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha' \rangle \). By Lemma 6 \( \mathcal{L}(A) = \mathcal{L}(A') \). Moreover, the partitioning \( \{ q \in Q \mid c(q) = k \}, \{ q \in Q \mid c(q) = k - 1 \}, \ldots, \{ q \in Q \mid c(q) = c(q_{\text{init}}) \} \) satisfies the weak conditions for \( A' \) and therefore \( A' \in \text{DWW}[k - c(q_{\text{init}}) + 1] \). Note that \( c \) is a \( k \)-maximal \( A' \)-coloring and \( k \) is minimal with respect to \( A' \) (originally \( c \) and \( k \) were defined with respect to \( A \)).

In the following we say a set of states \( U \) is reachable from a state \( q \) if any of the states in \( U \) is reachable from \( q \) or \( q \in U \). We define \( R_i \) for any \( i \in \mathbb{N} \) by induction on \( i \).

**Base:** \( R_0 = \{ q \in Q \mid c(q) = k \text{ and } q \text{ is recurrent} \} \).

**Step:** \( R_{i+1} = \{ q \in Q \mid c(q) = k - (i + 1), q \text{ is recurrent and } R_i \text{ is reachable from } q \} \).

\( R_0, R_1, \ldots, R_{k-1} \) is a sequence of sets of recurrent states. Each set is from a different partition of \( A' \) and every state in the set can reach the set preceding it. These are the states that have the potential to form one of the recurrent state sequences as in Lemma 5. If none of the sets \( R_0, R_1, \ldots, R_{k-1} \) is empty, we can construct the sequence of recurrent states \( q_0, q_1, \ldots, q_{k-1} \) as follows. We set \( q_0 \) to an arbitrary states from \( R_{k-1} \). By definition of \( R_{k-1} \), there exists a state, denoted \( q_1 \), in \( R_{k-2} \) that is reachable from \( q_0 \). By definition of \( R_{k-2} \), there exists a state, denoted \( q_2 \), in \( R_{k-3} \) that is reachable from \( q_1 \), etc. In general, we pick \( q_{i+1} \) to be a state in \( R_{k-1-(i+1)} \) that is reachable from \( q_i \).

We will show if one of the sets \( R_0, R_1, \ldots, R_{k-1} \) is empty \( k \) is not minimal. If \( R_0 \) is empty then there are no recurrent states with color \( k \). Since \( \delta \) does not use \( \perp, \top \)
there could also be no transient state with color \( k \). Hence \( c \) is a \((k-1)\)-maximal \( A' \)

coloring and that is a contradiction to \( k \) being minimal. We define \( R'_i = \{ q \in Q \mid c(q) = k - i \text{ and } R_i \text{ is reachable from } q \} \) for any \( i \in \mathbb{N} \) (note that \( R_i \subseteq R'_i \)). The states in \( R'_i \) are states for which the color has to be smaller than the color of the states in \( R_{i-1} \) in any \( A \)-coloring. We define \( T_0 = \emptyset \) and \( T_i = \{ q \in Q \mid c(q) = k - i \text{ and } q \text{ is transient and } R_{i-1} \text{ is reachable from } q \} \setminus R'_i \) for any \( i > 0 \). The states in \( T_i \) are states for which the color has to be not greater then the color of the states in \( R_{i-1} \) in any \( A \)-coloring.

If \( R'_{i+1} \) is empty we can modify \( c \) by decreasing the color of the states in \( R'_j \) and \( T_j \)

every \( j \leq i \). Since \( R'_0 = \{ q \in Q \mid c(q) = k \} \), the resulting \( A' \)-coloring will be \((k-1)\)-
maximal hence contradicting the minimality of \( k \). Since \( R_{i+1} = \emptyset \) implies \( R'_{i+1} = \emptyset \),
\( R_{i+1} \) is not empty. Formally, for any \( i \in [k-1] \), if \( R_{i+1} = \emptyset \) then the following is a
\((k-1)\)-maximal \( A \)-coloring,

\[
c'(q) = \begin{cases} 
c(q) - 2 & q \in \bigcup_{0 \leq j \leq i} R'_j \\
c(q) - 1 & q \in \bigcup_{0 \leq j \leq i+1} T_j \\
c(q) & \text{otherwise}
\end{cases}
\]

That is a contradiction to the minimality of \( k \). Therefore \( R_i \neq \emptyset \) for any \( i \in [k] \).

Now that we know we can construct the sequence \( q_0, q_1, \ldots, q_{k-1} \), observe that \( c(q_i) = i + 1 \). By definition of \( A \)-coloring (conditions 1 and 2 in definition 11), \( q_{i+1} \in \alpha \iff q_i \notin \alpha \) for any \( i \in [k-1] \). Moreover, \( c(q_0) = 1 \).

By the minimality of \( k \) we have that \( c(q_{\text{init}}) \) is either 0 or 1. If \( c(q_{\text{init}}) = 1 \) then
\( k - c(q_{\text{init}}) + 1 = k \) and by Lemma 5 and the sequence of \( k \) states, \( q_0, q_1, \ldots, q_{k-1} \), we
have that any automaton accepting \( \mathcal{L}(A) \) has at least \( k - c(q_{\text{init}}) + 1 \) partitions, and
therefore \( \mathcal{L}(A) \neq \mathcal{L}(\text{DWW}[k - c(q_{\text{init}})]) \).

Otherwise, \( c(q_{\text{init}}) = 0 \), we construct another sequence of \( k \) recurrent states \( p_0, p_1, \ldots, p_{k-1} \) in \( Q \). We define \( p_m \in Q \) for \( m \in [k] \) recursively such that \( p_m \) is recurrent and
\( c(p_m) = m \).

**Base**: By Remark 2, there exists a word \( w \in \Sigma^* \) such that the run of \( A' \) on \( w \) (starting
from \( q_{\text{init}} \)) has only states with color \( c(q_{\text{init}}) \). Therefore there exists a recurrent
state \( p_0 \) such that \( c(p_0) = 0 \).

**Closure**: Assume we already defined \( p_i \). By Remark 1 there exists a word \( w \in \Sigma^* \)
such that \( c(\delta(p_m, w)) = m + 1 \). By Remark 2 there exists a word \( w' \in \Sigma^* \) such that the run of \( A' \) on \( w' \), starting from state \( \delta(p_m, w) \) has only states with color
\( m + 1 \). Therefore there exists a recurrent state \( p_{m+1} \), reachable from \( p_m \), such that
\( c(p_{m+1}) = m + 1 \).

Let \( A'' = (\Sigma, Q'', q''_{\text{init}}, \delta'', \alpha'') \) be a DWW automaton such that \( \mathcal{L}(A'') = \mathcal{L}(A) \).
By Lemma 5, there exists a sequence of states \( q''_0, q''_1, \ldots, q''_{k-1} \) in \( Q'' \) such that \( q''_{i+1} \) is
reachable from \( q_i'' \) and \( q_i'' \in \alpha'' \iff q_i \in \alpha' \). And also there exists a sequence of states \( p_{i}'', p_{1}'', \ldots, p_{k-1}'' \) in \( Q'' \) such that \( p_i'' \) is reachable from \( p_i'' \) \( p_i'' \in \alpha'' \iff p_i \in \alpha' \). Since \( c(q_0) = 1 \) and \( c(p_0) = 0 \) we have that \( q_0'' \) and \( p_0'' \) are in different partitions. Therefore \( A'' \) has at least \( k + 1 = k - c(q_{\text{init}}) + 1 \) partitions.

In [15] Kupferman et al. show that if a DBW automaton can be translated to DWW, then the translation involves only the change of the \( \alpha \) part. We use this observation combined with Lemma 7 to show that if a DBW automaton can be translated to DWW\([k]\), then the translation involves only the change of the \( \alpha \) part.

**Theorem 10.** DBW is DWW\([k]\)-type for any \( k \in \mathbb{N} \).

**Proof.** Let \( A = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha \rangle \in \text{DBW} \) and assume \( L(A) \in L(\text{DWW}[k]) \) for some \( k \in \mathbb{N} \). By DBW is DWW-type (see [15]), there exists some \( \alpha' \) such that \( A' = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha' \rangle \) is a DWW automaton and \( L(A') = L(A) \). Let \( k' \in \mathbb{N} \) be the minimal number such that there exists a \( k' \)-maximal \( A' \)-coloring \( c : Q \to \mathbb{N} \). By Lemma 7, there exists some \( \alpha'' \) such that \( A'' = \langle \Sigma, Q, q_{\text{init}}, \delta, \alpha'' \rangle \in \text{DWW}[k' - c(q_{\text{init}}) + 1] \) and \( L(A'') = L(A') \) and \( L(A') \notin L(\text{DWW}[k' - c(q_{\text{init}})]) \). Therefore \( k \geq k' - c(q_{\text{init}}) + 1 \) and therefore \( A'' \in \text{DWW}[k] \). \( \square \)
Chapter 5

Conclusion

We showed a direct translation from ABW automata to AWW automata with state complexity $O(n^2)$. We complemented this upper bound with a proof that $\Omega(n \log n)$ is a lower bound for the translation. We showed which parts of the lower bound proof cannot be improved and which parts have a potential to give a better lower bound. In particular, we showed an example of a special case with $\Omega(n^2)$ lower bound. The tight bound for the ABW to AWW translation remains an open problem, with $\Omega(n \log n)$ being the best known lower bound and $O(n^2)$ being the best known upper bound.

We showed that the deterministic case is quite different. First we showed the DWW hierarchy is strict, that is, allowing the automaton to have more partitions gives it more expressive power. And finally we showed that if a DBW automaton can be translated to a DWW[k] automaton, then it is also possible to do the translation by only moving states in and out of the accepting set.
References


42


אוטומטי אומגה הולשימ

שקד פלור
אוטומטי אומגה חולשים

תינור על מחקר

ليس מאלי חלקי של הדרישה לקבילה והנורא
ממדים ממוצעים מבצעי המישור

שקד פלור

רות מזג גנטים - מוסק טכנולוגי לישראל
אדר בתשע"ג חיפה
פברואר 2013
המתקד מעשיה בדנה יותר ורובה אגרה ויררנברג מפקלי למודיעים המחשב.

אסי מודר לטכנאי על התמימה הבכיפה ורידגوحدة והשכלה התוכנית.
תקציר

הатומטים סיפסו Quân מאלי יאנשנפואט מסוריס ומתארים את המא웠 את האות אוטומט. קיימן

ם מתפר סוני שון של יאנשנפואט ספימו פדריטה חקוקה על במאייפן שוני, המחק

והרורו פנה. דובرمز אוטוטי מסיף על מאלי ספימו, ויה לע מי חון של פקניק

יפ המשכיות פקטקטי המ tysוית ב展演ים אוטוטי פלקטעי. דוהי מעב

ዊ רוחון. פקטקטי עברים תורמאניסיאת רדיס ינפל בשיט עלי יאנשנפואט לבחר. פקטקטי עבורי אוטוטי-דרפאנט. דוהי ערב עוהי, משך קצבי ומיסים רכיש מאוזן פקטקטי עבורי וטנייס. בהכית מס פוח, מנטור

הכרת מבכות את מחם חיבי יאנשנפואט עבון ומגאינו. רפה של יאנשנפואט תורמאניסיאית-א-י

dדררמייטס היון שזרתם של מבכות, בינクリック של יאנשנפואט ומיסים יאנשנפואט קובעל כדי שנוי,

שיך מעל הגוז עוזарат כאקת למאלי ומכת חרד מתיל. פקטקטי עבורי וטנייס

יסטא יאנשנפואט-א-ינגרסיל יד בזק רפה רוחש אפרתה. בער יאנשנפואט-א-דרפאנטיסיאת חסן

מסPager (ייל יאנשנפואט) של רפה, שיזוף התוספות והי' פקטקטי עבוריים ממלד על ד

שוכ סוס על יאנשנפואט הקצב את פקטקטי מדירס מחרת. פקטקטי עבוריים מחרת

לגבא את המחם של פקטקטי עבוריים-א-דרפאנטיסיאת לש פקטקטי עבוריים יאנשנפואט-א-ינגרסיל.

מדוהי מעל עבור, פקטקטי עבוריים מחרת פוניקי יאטח כבוטח למאלי מתויה.

טומט גיל לולא טווח במחם (באמפי-א-דרפאנטיס) און טרומל לחמי יאנשנפואט-א-ינגרסיל, שנה

בחקג מגר אוח במחם. פקטקטי עבוריים-א-ינגרסיל פקטקטי מדירס והקצב את יאנשנפואט-א-ינגרסיל.

ישיטי אלפ מדיטי היקחב את פקטקטי מדירס פקטקטי עבוריים עלי סוליקט חליפה יאנשנפואט מוהו

כבר המזון דובים מבוא. פקטקטי עבוריים-א-ינגרסיל פקטקטי מדירס והקצב את יאנשנפואט-א-ינגרסיל.

ראחחלת עבוריים במקלה. פקטקטי עבוריים-א-ינגרסיל רפה של יאנשנפואט-א-דרפאנטיסיאת יאנשנפואט-א-ינגרסיל, שנה

שיך ובצק מגר אוח במחם. פקטקטי עבוריים-א-ינגרסיל פקטקטי מדירס והקצב את יאנשנפואט-א-ינגרסיל.

בחקג מגר אוח במחם. פקטקטי עבוריים-א-ינגרסיל פקטקטי מדירס והקצב את יאנשנפואט-א-ינגרסיל.

לגבא את המחם של פקטקטי עבוריים-א-דרפאנטיסיאת לש פקטקטי עבוריים יאנשנפואט-א-ינגרסיל.

לגבא את המחם של פקטקטי עבוריים-א-דרפאנטיסיאת לש פקטקטי עבוריים יאנשנפואט-א-ינgré

שיך ובצק מגר אוח במחם. פקטקטי עבוריים-א-ינגרסיל פקטקטי מדירס והקצב את יאנשנפואט-א-ינgré.
The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.

The open problem of constructing a deterministic auxilliary memoryless automaton that tests if the language of some deterministic bounded automaton is empty.
מ נינת להראות כל השם והז緩ים רכושי

פרק 4: עיסק בדרכנים כדי לעצם את האוטומטים דטרמיניסטיים. Düה ככ כל שפה אוטומטית

מצטבלת על ידי האוטומטים של מרחוק בין 4 תת-דרכנים פריטים.لاحיש, הרספ התכניות

מעבר ל-4 האוטומטים את יישול הביאור של האוטומטים של מרחוק בין 4 מרחוק והם

ל Jwt האוטומטים יישום בדרכנים. האוטומטים יישום בדרכנים בין 4 מרחוק התכניות

כבר מתבצעים בחוזה שערת האוטומטים של מרחוק והם

שתכניות 마련ים את כל הלקטנים לאתרים של מרחוק שאם האוטומטים בדרכנים הם

יתרIKEA, המילא את האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

ל Jwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

ל Jwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

ל Jwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בין 4

לJwt האוטומטים של מרחוק בין 4 מרחוק התכניות בינ