Advanced topics in automata theory

Mikołaj Bojańczyk and Wojciech Czerwiński

Preface

These are lecture notes for a course on advanced automata theory, that we gave at the University of Warsaw in the years 2015-2018. The lectures were written down by the first author and the exercises by the second author, but we consulted each other extensively in the process of both teaching and writing.

Mikołaj Bojańczyk and Wojciech Czerwiński

Contents

1
Determinisation of ω-automata

In this chapter, we discuss automata for ω-words, i.e. infinite words of the form

$$a_1 a_2 a_3 \cdots$$

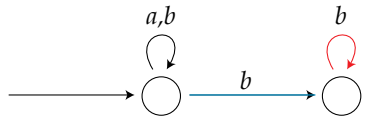
We write Σ^ω for the set of ω words over alphabet Σ. The topic of this chapter is McNaughton's Theorem, which shows that automata over ω-words can be determinised. A more in depth account of automata (and logic) for ω words can be found in [6].

1.1 Automata models for ω-words

A *nondeterministic Büchi automaton* is a type of automaton for ω-words. Its syntax is typically defined to be the same as that of an nondeterministic finite automaton: a set of states, an input alphabet, initial and accepting subsets of states, and a set of transitions. For our presentation it will be more convenient to use accepting transitions, i.e. the accepting set is a set of transitions, not a set of states. An infinite word is accepted by the automaton if there exists a run which begins in one of the initial states, and visits some accepting transition infinitely often.

Example 1. Consider the set of words over alphabet $\{a, b\}$ where the letter a appears finitely often. This language is recognised by a nondeterministic Büchi

automaton like this (we adopt the convention that accepting transitions are red edges):



□

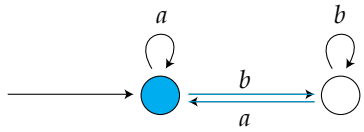
Fact 1.1. *Nondeterministic Büchi automata recognise strictly more languages than deterministic Büchi automata.*

Proof. Take the automaton from Example 1. Suppose that there is a deterministic Büchi automaton that is equivalent, i.e. recognised the same language. Let us view the set of all possible inputs as an infinite tree, where the vertices are prefixes $\{a, b\}^*$. Since the automaton is deterministic, to each edge of this tree one can uniquely assign a transition of the automaton. Every vertex $v \in \{a, b\}^*$ of this tree has an accepting transition in its subtree, because the word vb^ω should have an accepting run. Therefore, we can find an infinite path in this tree which has a infinitely often and uses accepting transitions infinitely often. ■

The above fact shows that if we want to determines automata for ω-words, we need something more powerful than the Büchi condition. One solution is called the Muller condition, and is described below. Later, we will see another, equivalent, solution, which is called the parity condition.

Muller automata. The syntax of a Muller automaton is the same as for a Büchi automaton, except that the accepting set is different. Suppose that Δ is the set of transitions. Instead of being a set $F \subseteq \Delta$ of transitions, the accepting set in a Muller automaton is a family $\mathcal{F} \subseteq \mathsf{P}\Delta$ of sets of transitions. A run is accepting if the set of transitions visited infinitely often belongs to the family $\mathcal{F}$.

Example 2. Consider this automaton



If we set $\mathcal{F}$ to be all subsets which contain at least one transition that enters the blue state, then the automaton will accept words which contain infinitely many a's. If we set $\mathcal{F}$ to be all subsets which contain only transitions that enter the blue state, then the automaton will accept words which contain infinitely many a's and finitely many b's. □

Deterministic Muller automata are clearly closed under complement – it suffices to replace the accepting family by $\mathsf{P}\Delta - \mathcal{F}$. This lecture is devoted to proving the following determinisation result.

Theorem 1.2 (McNaughton's Theorem)**.** *For every nondeterministic Büchi automaton there exists an equivalent (accepting the same ω-words) deterministic Muller automaton.*

The converse of the theorem, namely that deterministic Muller (even nondeterministic) automata can be transformed into equivalent nondeterministic Büchi automata is more straightforward, see Exercise ... It follows that from the above discussion that

- nondeterministic Büchi automata
- nondeterministic Muller automata
- deterministic Muller automata

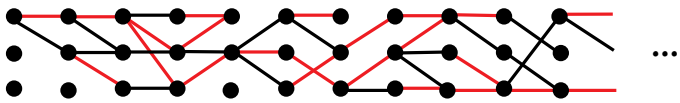
have the same expressive power, but deterministic Büchi automata are weaker. The theorem was first proved in [4]. The proof here is similar to one by Muller and Schupp [5]. An alternative proof method is the Safra Construction, see e.g. [6].

The proof strategy is as follows. We first define a family of languages, called universal Büchi languages, and show that the McNaughton's theorem boils down to recognising these languages. Then we show how the universal languages can be recognised by deterministic Muller automata.

The universal Büchi language. For $n \in \mathbb{N}$, define a width n dag to be a directed acyclic graph where the nodes are pairs $\{1,\ldots,n\} \times \{1,2,\ldots\}$ and every edge is of the form

$$(q,i) \to (p,i+1) \qquad \text{for some } p,q \in \{1,\ldots,n\} \text{ and } i \in \{1,2,\ldots\}.$$

Furthermore, every edge is either red or black, with red meaning "accepting". We assume that there are no parallel edges. Here is a picture of a width 3 dag:



In the pictures, we adopt the convention that the i-th column stands for the set of vertices $\{1,\ldots,n\} \times \{i\}$. The top left corner of the picture, namely the vertex $(1,1)$, will be called the *initial vertex*.
As we will show, the essence of McNaughton's theorem is showing that for every n, there is a deterministic Muller automaton which inputs a width n dag and says if it contains a path that begins in the initial vertex and visits infinitely many red (accepting) edges. In order to write such an automaton, we need to encode as width n dag as an ω-word over some finite alphabet. This is done using an alphabet, which we denote by $[n]$, where the letters look like this:

Formally speaking, $[n]$ is the set of functions

$$\{1,\ldots,n\} \times \{1,\ldots,n\} \to \{\text{no edge, non-accepting edge, accepting edge}\}.$$

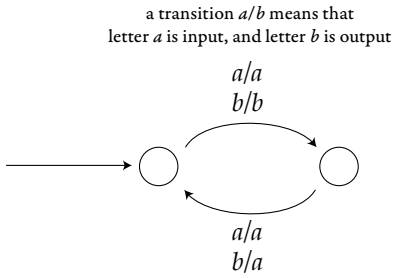
Define the universal n state Büchi language to be the set of words $w \in [n]^\omega$ which, when treated as a width n dag, contain a path that visits accepting edges infinitely often and starts in the initial vertex. The key to McNaughton's theorem is the following proposition.

Proposition 1.3. *For every $n \in \mathbb{N}$ there is a deterministic Muller automaton recognizing the universal n state Büchi language.*

Before proving the proposition, let us show how it implies McNaughton's theorem. To make this and other proofs more transparent, it will be convenient to use transducers. Define a *sequential transducer* to be a deterministic finite automaton, without accepting states, where each transition is additionally labelled by a letter from some output alphabet. The name transducer refers to an automaton which outputs more than just yes/no; later in this book we will see many other (and more powerful) types of transducers, with names like bimachine transducer or register transducer. If the input alphabet is Σ and the output alphabet is Γ, then a sequential transducer defines a function

$$f : \Sigma^\omega \to \Gamma^\omega.$$

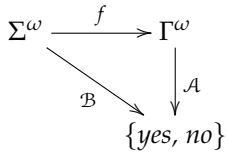
Example 3. Here is a picture of a sequential transducer which inputs a word over $\{a, b\}$ and replaces letters on even-numbered positions by a.



□

Lemma 1.4. *Languages recognised by deterministic Muller automata are closed under inverse images of sequential transducers, i.e. if $\mathcal{A}$ in the diagram below is a deterministic Muller automaton f is a sequential transducer, there is a deterministic*

Muller automaton $\mathcal{B}$ which makes the following diagram commute:



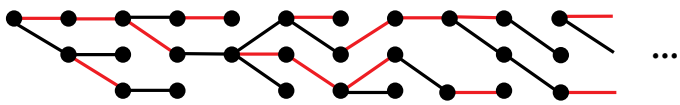
Proof. The states of automaton $\mathcal{B}$ are the product of the states for f and $\mathcal{A}$. ∎

Let us continue with the proof of McNaughton's theorem. We claim that every language recognised by a nondeterministic Büchi automaton reduces to a universal Büchi language via some transducer. Let $\mathcal{A}$ be a nondeterministic Büchi automaton with input alphabet Σ. We assume without loss of generality that the states are numbers $\{1, \ldots, n\}$ and the initial state is 1. By simply copying the transitions of the automaton, one obtains a sequential transducer

$$f : \Sigma^\omega \to [n]^\omega$$

such that a word $w \in A^\omega$ is accepted by $\mathcal{A}$ if and only if $f(w)$ contains a path fro the initial vertex with infinitely many accepting edges. By composing the transducer with the automaton from the proposition, we get a deterministic Muller automaton equivalent to $\mathcal{A}$. It now remains to show the proposition, i.e. that the n state universal Büchi language can be recognised by a Muller automaton. The proof has two steps.

The first step is stated in Lemma 1.5 and says that a when dealing with the universal Büchi language, one can write a deterministic transducer which replaces an arbitrary dag by an equivalent tree. Here we use the name tree for a width n dag where, every non-isolated node other than (1,1) has exactly one incoming edge. Here is a picture of such a tree, with the isolated nodes not drawn:



Lemma 1.5. *There exists a sequential transducer*

$$f : [n]^\omega \to [n]^\omega$$

which outputs only trees and is invariant with respect to the universal Büchi language, i.e. if the input contains a path with infinitely many accepting edges, then so does the output and vice versa.

The second step is showing that a deterministic Muller automaton can test if a tree contains an accepting path.

Lemma 1.6. *There exists a deterministic Muller automaton recognising*

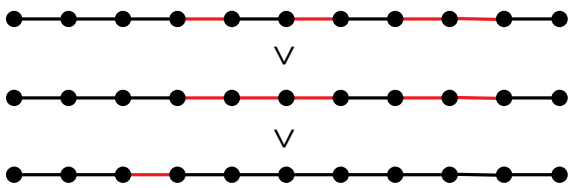
$$\{w \in [n]^\omega : w \text{ is a tree and contains a path with infinitely many accepting edges}\}.$$

Combining the two lemmas, we get Proposition 1.3, and thus finish the proof of McNaughton's theorem. Lemma 1.5 is proved in Section 1.2 and Lemma 1.6 is proved in Section 1.3.

1.2 Reduction to trees

We begin by proving Lemma 1.5, which says that a sequential transducer can convert a width n dag into a tree, while preserving the existence of a path from the initial vertex with infinitely many accepting edges.

Profiles. For a path π in a width n-dag, define its *profile* to be the word of same length over the alphabet "accepting" and "non-accepting" which is obtained by replacing each edge with its appropriate type. We consider order profiles lexicographically, with "accepting" is smaller than "non-accepting".



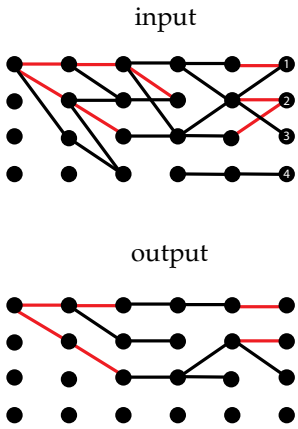
A finite path π in a width n dag is called *profile optimal* if it has lexicographically least profile among paths in w that have the same source and target.

Lemma 1.7. *There is a sequential transducer*

$$f : [n]^\omega \to [n]^\omega$$

such that if the input is w, then $f(w)$ is a tree with the same reachable (from tie initial vertex) vertices as in w, and such that every finite path in $f(w)$ that begins in the root is an optimal path in w.

Proof. We use the following congruence property of optimal paths: if π and σ are optimal paths such that the target of π is equal to the source of σ, then also their composition $\pi\sigma$ is optimal. A corollary is that if we choose for every vertex in the input graph w an outgoing edge that participates in some optimal path, then the putting all of these edges together will yield a tree as in the statement of the claim. To produce such edges, after reading the first n letters, the automaton keeps in its memory the lexicographic ordering on the profiles of optimal paths lead from the root to nodes at depth n. Here is a picture of the construction:



The state of the tranducer is this information:

The reachable vertices are

❶ ❷ ❸

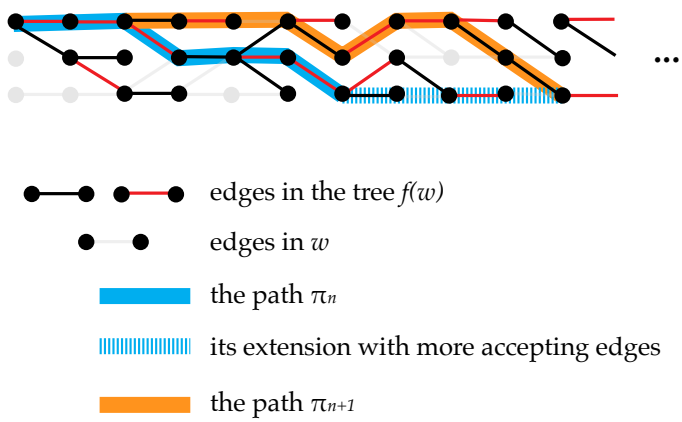
and the least profiles for reaching them are ordered as

❶ = ❷ < ❸

■

Lemma 1.8. *Let f be the sequential transducer from Lemma 1.7. If the input to f contains a path with infinitely many accepting edges, then so does the output.*

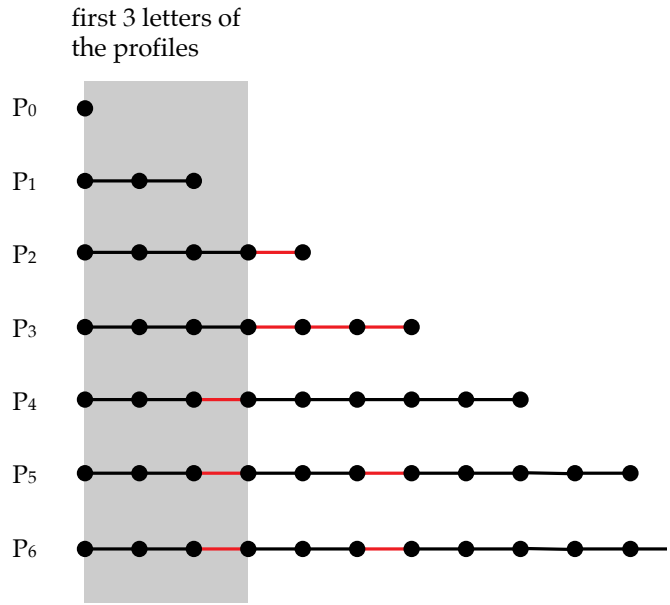
Proof. Assume that the input w to f contains a path with infinitely many accepting edges. Define a sequence $\pi_0, \pi_1, \ldots$ of finite paths in $f(w)$ as follows by induction. In the definition, we preserve the invariant that each path in the sequence $\pi_0, \pi_1, \ldots$ can be extended to an an accepting path in the graph w. We begin with π_0 being the edgeless path that begins and ends in the root of the tree $f(w)$. This path π_0 satisfies the invariant, by the assumption that the input w contains a path with infinitely many accepting edges. Suppose that π_n has been defined. By the invariant, we can extend π_n to an infinite path in the graph w, and therefore we can extend π_n to a finite path in w that contains at least one more accepting edge. Define π_{n+1} to be the unique path in the tree $f(w)$ which has the same source and target as the new path that extends π_n with at at leat one accepting edge.



Define P_n to be the profile of the path π_n. We claim that the sequence of profile $P_0, P_1, P_2, \ldots$ has a well defined limit

$$\lim_{n\to\infty} P_n = P \in \{\text{accepting, non-accepting}\}^\omega.$$

More precisely, we claim that for every position i, the i-th letter of the profiles $P_1, P_2, \ldots$ eventually stabilises. The limit P is defined to be the sequence of these stable values. The limit exists because for every i, if we look at the prefixes of $P_0, P_1, \ldots$ of length i, then they get lexicographically smaller and smaller; and therefore they must eventually stabilise, as in the following picture:



Claim 1.9. *The limit P contains the letter "accepting" infinitely often.*

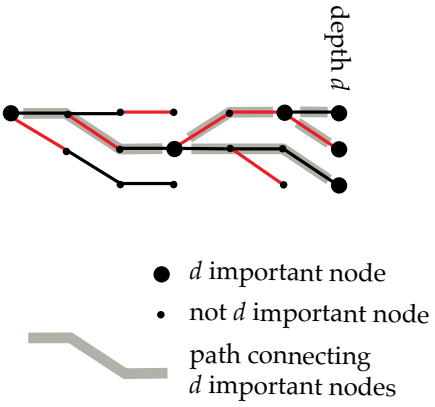
Proof. Toward a contradiction, suppose that P has the letter "accepting" finitely often, i.e. there is some i such that after i, only the letter "non-accepting" appears in P. Choose n so that $\pi_n, \pi_{n+1}, \ldots$ have profile consistent with P on the first i letters. By construction, the profile P_{n+1} has an accepting letter on some position after i, and this property remains true for all subsequent profiles

$P_{n+2}, P_{n+3} \ldots$ and therefore is also true in the limit, contradicting our assumption that P has only "non-accepting" letters after position i. ■

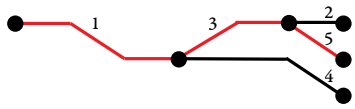
Consider the set of finite paths in the tree $f(w)$ which have profile that is a prefix of P. This set of paths forms a tree (because it is prefix-closed). This tree has bounded degree (assuming the parent of a path is obtained by removing the last edge) and it contain paths of arbitrary finite length (suitable prefixes of the paths $\pi_1, \pi_2, \ldots$). The Köning lemma says that every finitely branching tree with arbitrarily long paths contains an infinite path. Applying the König lemma to the paths in $f(w)$ with profile P, we get an infinite path with profile P. By Claim 1.9 this path has infinitely many accepting edges. ■

1.3 *Finding an accepting path in a tree graph*

We now show Lemma 1.6, which says that a deterministic Muller automaton can check if a width n tree contains a path with infinitely many accepting edges. Consider a tree $t \in [n]^\omega$, and let $d \in \mathbb{N}$ be some depth. Define a node to be important for depth d if it is either: the root, a node at depth d, or a node which is a closest common ancestor of two nodes at depth d. This definition is illustrated below (with solid lines representing accepting edges, and dotted lines representing non-accepting edges):



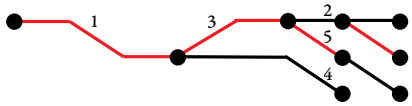
Definition of the Muller automaton. We now describe the Muller automaton for Lemma 1.6. After reading the first d letters of an input tree (i.e. after reading the input tree up to depth d), the automaton keeps in its state a tree, where the nodes correspond to nodes of the input tree that are important for depth d, and the edges corresponds to paths in the input tree that connect these nodes. This tree stored by the automaton is a tree with at most n leaves, and therefore it has less than $2n$ edges. The automaton also keeps track of a colouring of the edges, with each edge being marked as accepting or not, where "accepting" means that the corresponding path in the input tree contains at least one accepting edge. Finally, the automaton remembers for each edge an identifiers from the set $\{1, \ldots, 2n-1\}$, with the identifier policy being described below. A typical memory state looks like this:



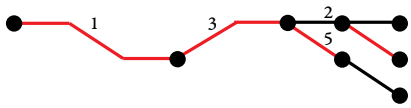
The big black dots correspond to important nodes for the current depth, red edges are accepting, black edges are non-accepting, while the numbers are the identifiers. All identifiers are distinct, i.e. different edges get different identifiers. It might be the case (which is not true for the picture above), that the identifiers used at a given moment have gaps, e.g. identifier 4 is used but not 3. The initial state of the automaton is a tree which has one node, which is the root and a leaf at the same time, and no edges. We now explain how the state is updated. Suppose the automaton reads a new letter, which looks something like this:

To define the new state, perform the following steps.

1. Append the new letter to the tree in the state of the automaton. In the example of the tree and letter illustrated above, the result looks like this:



2. Eliminate paths that do die out before reaching the new maximal depth. In the above picture, this means eliminating the path with identifier 4:



3. Eliminate unary nodes, thus joining several edges into a single edge. This means that a path which only passes through nodes of degree one gets collapsed to a single edge, the identifier for such a path is inherited from the first edge on the path. In the above picture, this means eliminating the unary nodes that are the targets of edges with identifiers 1 and 5:

4. Finally, if there are edges that do not have identifiers, these edges get assigned arbitrary identifiers that are not currently used. In the above picture, there are two such edges, and the final result looks like this:

This completes the definition of the state update function. We now define the acceptance condition.

The acceptance condition. When executing a transition, the automaton described above goes from one tree with edges labelled by identifiers to another tree with edges labelled by identifiers. For each identifier, a transition can have three possible effects, described below:

1. **Delete.** An edge can be deleted in step 2 or in step 3 (by being merged with an edge closer to the root). The identifier of such an edge is said to be deleted in the transition. Since we reuse identifiers, an identifier can still be present after a transition that deletes it, because it has been added again in step 4, e.g. this happens to identifier 4 in the above example.

2. **Refresh.** In step 3, a whole path $e_1 e_2 \cdots e_n$ can be folded into its first edge e_1. If the part $e_2 \cdots e_n$ contains at least one accepting edge, then we say that the identifier of edge e_1 is refreshed.

3. **Nothing.** An identifier might be neither deleted nor refreshed, e.g. this is the case for identifier 2 in the example.

The following lemma describes the key property of the above data structure.

Lemma 1.10. *For every tree in $[n]^\omega$, the following are equivalent:*

(a) the tree contains a path from the root with infinitely many accepting edges;

(b) some identifier is deleted finitely often but refreshed infinitely often.

Before proving the above fact, we show how it completes the proof of Lemma 1.6. We claim that condition (a) can be expressed as a Muller condition on transitions. The accepting family of subsets of transitions is

$$\bigcup_i \mathcal{F}_i$$

where i ranges over possible identifiers, and the family $\mathcal{F}_i$ contains a set X of transitions if

- some transition in X refreshes identifier i; and

- none of the transitions in X delete identifier i.

Identifier i is deleted finitely often but refreshed infinitely often if and only if the set of transitions seen infinitely often belongs to $\mathcal{F}_i$, and therefore, thanks to the fact above, the automaton defined above recognises the language in the statement of Lemma 1.6.

Proof of Lemma 1.10. The implication from (b) to (a) is straightforward. An identifier in the state of the automaton corresponds to a finite path in the input tree. If the identifier is not deleted, then this path stays the same or grows to the right (i.e. something is appended to the path). When the identifier is refreshed, the path grows by at least one accepting state. Therefore, if the identifier is deleted finitely often and refreshed infinitely often, there is some path that keeps on growing with more and more accepting states, and its limit is a path with infinitely many accepting edges.
Let us now focus on the implication from (a) to (b). Suppose that the tree t contains some infinite path π that begins in the root and has infinitely many accepting edges. Call an identifier *active* in step d if the path described by this identifier in the d-th state of the run corresponds to a infix of the path π. Let I be the set of identifiers that are active in all but finitely many steps, and which are deleted finitely often. This set is nonempty, e.g. the first edge of the path π always has the same identifier. In particular, there is some step d, such that identifiers from I are not deleted after step n. Let $i \in I$ be the identifier that is last on the path π, i.e. all other identifiers in I describe finite paths that are earlier on π. It is not difficult to see that the identifier i must be refreshed infinitely often by prefixes of the path π. ∎

Problem 1. Are the following languages regular:

1. prefix of v belongs infinitely often to the fixed regular language of finite words $L \subseteq \Sigma^*$;

2. word v contains infinitely many infixes of the form $ab^p a$, where p is prime;

3. word v contains infinitely many infixes of the form $ab^p a$, where p is even;
4. word v contains arbitrary long infixes in the fixed regular language of finite words L;
5. prefix of v belongs infinitely often to the fixed language of finite words $L \subseteq \Sigma^*$ (not necessarily regular).

Problem 2. Show that language of words "there exists a letter b" cannot be accepted by a nondeterministic automaton with Büchi acceptance condition, where all the states are accepting (but possibly transitions over some letters missing in some states).

Problem 3. Show that language "finitely many occurrences of letter a" cannot be accepted by a deterministic automaton with Büchi acceptance condition.

Problem 4. Show that every language accepted by some nondeterministic automaton with Muller acceptance condition is also accepted by some nondeterministic automaton with Büchi acceptance condition.

Problem 5. Assume that we have changed the acceptance condition into such which investigates which sets of transitions are visited infinitely often. Does it affect the expressivity of automata? How it is for Büchi acceptance condition? And how for Muller acceptance condition?

Problem 6. Show that nonemptiness is decidable for automata with Muller acceptance condition.

2
Games with ω-regular winning conditions

In this chapter, we prove the Büchi-Landweber Theorem [1, Theorem 1], see also [6, Theorem 6.5], about games with ω-regular winning conditions. These are games where two players move a token around a graph, yielding an infinite path, and the winner is decided based on some property of this path that is recognised by an automaton on ω-words. The Büchi-Landweber Theorem gives an algorithm for deciding the winner in such games, thus solving "Church's Problem"[2].

2.1 Games

In this chapter, we consider games played by two players (called 0 and 1), which are zero-sum, perfect information, and most importantly, of potentially infinite duration.

Definition 2.1 (Game)**.** *Suppose that $W \subseteq \Sigma^\omega$ is a set of ω-words. Define a game with winning condition W to be:*

- *a directed graph, not necessarily finite, whose vertices are called positions of the game;*
- *a distinguished initial position;*

- *a partition of the positions into positions controlled by player 0 and positions controlled by player 1;*
- *a labelling function that maps each position to a label from* Σ.

The game is played as follows. The game begins in the initial position. The player who controls the initial position chooses an outgoing edge, leading to a new position. The player who controls the new position chooses an outgoing edge, leading to a new position, and so on. If the play reaches a position with no outgoing edges (called a dead end), then the player who controls the dead end loses immediately. Otherwise, the play continues forever, and yields an infinite path. By applying the labelling function to the path, we get a word in Σ^ω; if this word belongs to W then player 0 wins, otherwise player 1 wins.
To formalise the notions in the above paragraph, one uses the concept of a strategy. A *strategy* for player $i \in \{0,1\}$ is a function which inputs a history of the play so far (a path from the initial position to some position controlled by player i), and outputs the new position (consistent with the edge relation in the graph). Given strategies for both players, call these σ_0 and σ_1, a unique play is determined, which is either a finite path ending in a dead end, or an infinite path. This play is called winning for player 0 if it is finite and ends in a dead end controlled by the opposing player 1; or if it is infinite and satisfies the winning condition after applying the labelling function. Otherwise, the play is winning for player 1. A winning strategy for player i is defined to be a strategy σ_i such that for every possible strategy σ_{1-i} of the opponent, the resulting play is winning for player i.

Determinacy. A game is called *determined* if one of the players has a winning strategy. Clearly it cannot be the case that both players have winning strategies. One could be tempted to think that, because of the perfect information, one of the players must have a winning strategy. However, because of the infinite duration, one can come up with strange games (e.g. using the axiom of choice) which are not determined because none of the players has a winning strategy. The goal of this lecture is to show a theorem by Büchi and Landweber: if the winning condition of the game is recognised by an automaton, then the game is

determined, and furthermore the winning player has a finite memory winning strategy, in the following sense.

Definition 2.2 (Finite memory strategy)**.** *Consider a game where the positions are V. Let i be one of the players. A strategy for player i with memory M is given by:*

- *a deterministic automaton with states M and input alphabet V; and*
- *for every position v controlled by i, a function f_v from M to the neighbors of v.*

The two ingredients above define a strategy for player i in the following way: the next move chosen by player i in a position v is obtained by applying the function f_v to the state of the automaton after reading the history of the play, including v. We will apply this definition also to games with infinitely many positions, but we will only care about finite memory sets M (which leads us to consider finite state automata over infinite alphabets).

An important special case is when the set M has only one element, in which case the strategy is called *memoryless*. In this case, the new position chosen by the player only depends on the current position, and not on the history of the game before that.

Theorem 2.3 (Büchi-Landweber Theorem)**.** *For every ω-regular language W there exists a finite set M such that for every game with winning condition W, one of the players has a winning strategy that uses memory M.*

The proof of the above theorem has two parts. The first part is to identify a special case of games with ω-regular winning conditions, called parity conditions.

Definition 2.4 (Parity condition)**.** *Define the n-rank parity condition to be the set of ω-words over the alphabet $\{1, \ldots, n\}$ where the smallest number appearing infinitely often is even.*

A parity game is a game where the winning condition is the n-rank parity language for some n. Parity games are important because not only can they be won using finite memory strategies, but even memoryless strategies are enough.

Theorem 2.5 (Memoryless determinacy of parity games)**.** *For every parity game, one of the players has a memoryless winning strategy.*

The above theorem is proved in Section 2.2. The second step of the Büchi-Landweber theorem is the reduction to parity games. This essentially boils down to transforming deterministic Muller automata into called deterministic parity automata. In a parity automaton, there is a ranking function from states to numbers, and a run is considered accepting if the minimal rank appearing infinitely often is even. This is a special case of the Muller condition, but it turns out to be expressively complete in the following sense:

Lemma 2.6. *For every deterministic Muller automaton, there is an equivalent deterministic parity automaton.*

Proof. The lemma can be proved in two ways. One of them is to show that, by taking more care in the determinisation construction in McNaughton's Theorem, we can actually produce a parity automaton. Here we use an alternative construction, based on a data structure called the later appearance record [3]. The construction is presented in the following claim.

Claim 2.7. *For every finite alphabet Σ, there exists a deterministic automaton with input alphabet Σ, a totally ordered state space Q, and a function*

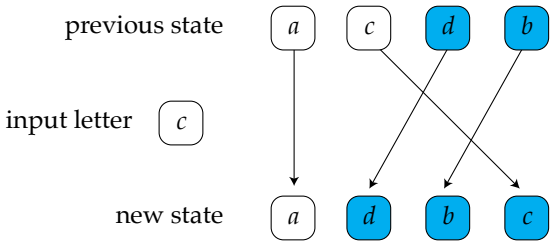
$$g : Q \to \mathcal{P}(\Sigma)$$

with the following property. For every input word, the set of letters appearing infinitely often in the input is obtained by applying g to the biggest state that appears infinitely often in the run.

Proof. The state space Q consists of data structures that look like this:

More precisely, a state is a (possibly empty) sequence of distinct letters from Σ, with distinguished blue suffix. The initial state is the empty sequence. After reading the first letter a, the state of the automaton is

When that automaton reads an input letter, it moves the input letter to the end of the sequence (if it was not previously in the sequence, then it is added), and marks as blue all those positions in the sequence which were changed, as in the following picture:



Consider a run of this automaton over some infinite input $w \in \Sigma^\omega$. Take some blue suffix of maximal size that appears infinitely often in the run. Then the letters in this suffix are exactly those that appear in w infinitely often. Therefore, to get the statement of the claim, we order Q first by the number of blue positions, and in case of the same number of blue positions, we use some arbitrary total ordering. The function g returns the set of blue positions. This completes the proof of the claim. ∎

The conversion of Muller to parity is a straightforward corollary of the above lemma: one applies the above lemma to the state space of the Muller automaton, and defines the ranks according to the Muller condition. This completes the proof of the lemma on conversion of Muller automata into parity automata. ∎

Let us now finish the proof of the Büchi-Landweber theorem. Consider a game with an ω-regular winning condition $L \subseteq \Sigma^\omega$. By Lemma 2.6, there is a

deterministic parity automaton which recognises the language L. Consider a new game, call it the product game, where the positions are pairs (position of the original game, state of the deterministic parity automaton). This is a parity game, with the ranks inherited from the automaton. In a position (v, q), the player controlling position v chooses an edge in the original game, and the state is updated deterministically according to the transition function of the automaton. It is not difficult to see that the following conditions are equivalent for every position v of the original game and every player $i \in \{0, 1\}$:

1. player i wins from position v in the original game;

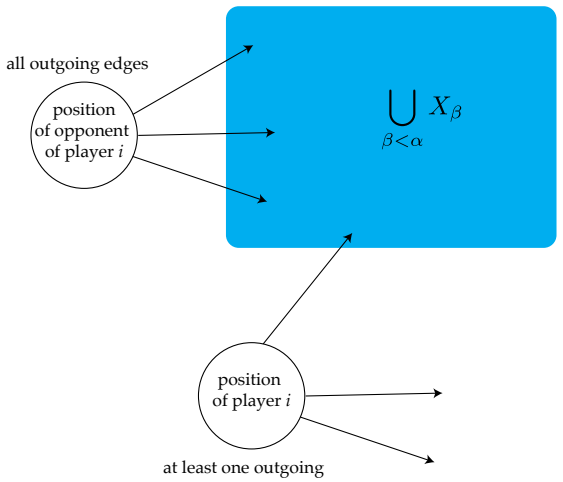
2. player i wins from position (v, q) in the product game, where q is the initial state of the deterministic parity automaton recognising L.

The implication from 1 to 2 crucially uses determinism of the automaton and would fail if a nondeterministic automaton were used (under an appropriate definition of a product game). Since the product game is a parity game, for every position v, condition 2 must hold for either player 0 or 1; furthermore, a positional strategy in the product game corresponds to a finite memory strategy in the original game, where the memory is the states of the deterministic parity automaton.
This completes the proof of the Büchi-Landweber Theorem. It remains to show memoryless determinacy of parity games, which is done below.

2.2 *Memoryless determinacy of parity games*

In this section, we prove Theorem 2.5 on memoryless determinacy of parity games. Recall that in a parity game, the positions are assigned ranks from a set $\{0, \ldots, n\}$, and the goal of player 0 is to ensure that for infinite plays, the minimal number appearing infinitely often is even. Our goal is to show that one of the players has a winning strategy, and furthermore this strategy is memoryless. The proof of the theorem is by induction on the number ranks from $\{0, \ldots, n\}$ that are used in the game.

Attractors. Consider a set of positions X in a parity game (actually the winning condition is irrelevant for the definition). For a player $i \in \{0,1\}$, we define below the i-attractor of X, which intuitively represents positions where player i can force a visit to the set X. The attractor is approximated using ordinal numbers. Define X_0 to be X, and for an ordinal number $\alpha > 0$, define X_α to be $X_{<\alpha}$, i.e. the union of X_β ranging over ordinals $\beta < \alpha$, plus the following two sets of positions: positions owned by player i where some outgoing edge leads to $X_{<\alpha}$; and positions owned by the opponent of i where all outgoing edges lead to $X_{<\alpha}$. Here is a picture:



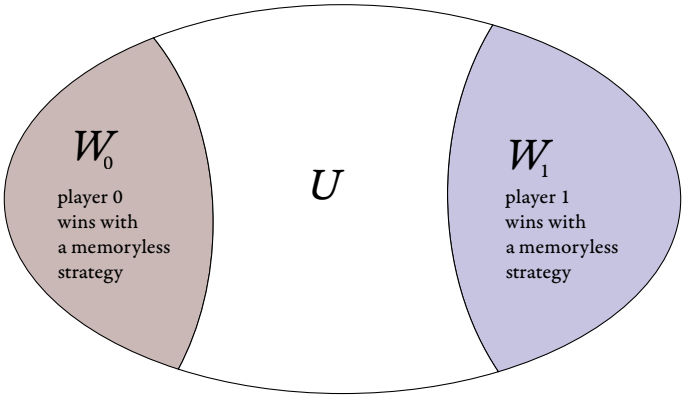
The set X_α grows as α grows, and therefore at some point it stabilises. This stable set is called the i-attractor of X. Over positions in the i-attractor, player i has a memoryless strategy which guarantees that after a finite number of steps, the game will end up in X or in a dead end owned by the opponent of player i. This strategy, called the attractor strategy, is to choose the neighbour that belongs to X_α with the smallest possible index α.

Induction base. Recall that we prove memoryless determinacy by induction on the number of ranks that are used in the game. The induction base is when only one rank is used. Let i be the parity of the only rank, without loss of generality we assume $i \in \{0, 1\}$. This means that every infinite play is won by player i. This does not necessarily mean that player i wins the game, because the game might end up in a dead end owned by player i. Let X be the $(1 - i)$-attractor of the empty set (i.e. the attractor from the point of view of the opponent of player i). We claim that on positions from X, the opponent of player i has a memoryless winning strategy, and on positions outside X, player i has a memoryless winning strategy. For positions in X, the opponent of player i plays the attractor strategy, which guarantees reaching a dead end position owned by player i in a finite number of steps. For positions outside X, we make the following observation, which follows immediately from the definition of X:

- if player i owns a position outside X, then some outgoing edge leads to a position outside X; and
- if the opponent of player i owns a position outside X, then all outgoing edges lead to a position outside X.

It follows that if the play begins outside X, then player i has a memoryless strategy that guarantees avoiding X forever, in particular this strategy is winning.

Induction step. Consider a parity game. For $i \in \{0, 1\}$ define W_i to be the set of positions v such that if the initial position is replaced by v, then player i has a memoryless winning strategy. Define U to be the vertices that are in neither W_0 nor in W_1. Our goal is to prove that W is empty. Here is the picture:

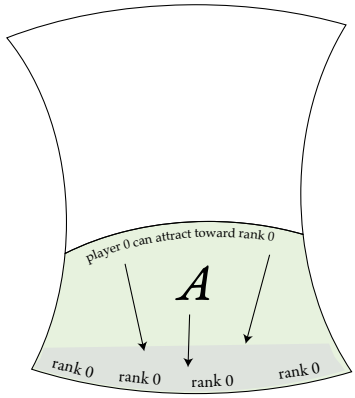


By definition, for every position in $w \in W_i$, player i has a memoryless winning strategy that wins when starting in position w. In principle, the memoryless strategy might depend on the choice of w, but the following lemma shows that this is not the case.

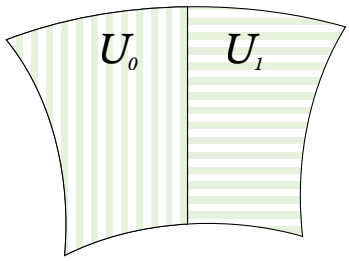
Lemma 2.8. *Let $i \in \{0, 1\}$ be one of the players. There is a memoryless strategy σ_i for player i, such that if the game starts in W_i, then player i wins by playing σ_i.*

Proof. By definition, for every position $w \in W_i$ there is a memoryless winning strategy, which we call the strategy of w. We want to consolidate these strategies into a single one that does not depend on w. Choose some well-ordering of the vertices from W_i, i.e. a total ordering which is well-founded. For a position $w \in W_i$, define its *companion* to be the least position v such that the strategy of v wins when starting in w. The companion is well defined because we take the least element, under a well-founded ordering, of some set that is nonempty (because it contains w). The consolidated strategy is defined as follows: when in position w, play according to the strategy of the companion of w. The key observation is that for every play using this consolidated strategy, the sequence of companions is non-decreasing in the well-ordering, and therefore it must stabilise at some companion v; and therefore the play must be winning for player i, since from some point on it is consistent with the strategy of v. ∎

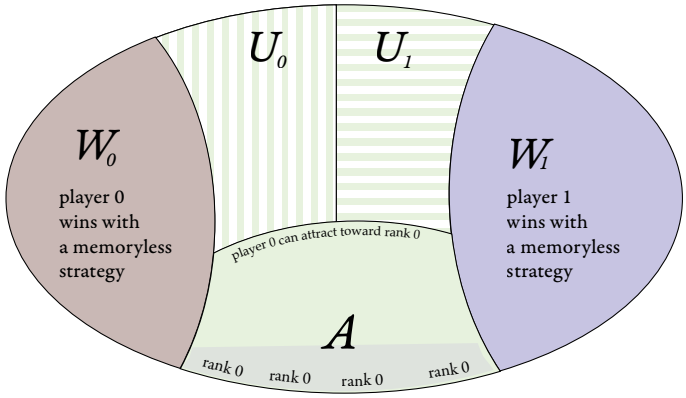
Consider the minimal rank that appears in the entire game, let it be n. By symmetry, we assume that this minimal rank is 0. (The symmetric case is when the minimal rank is 1.) Define A to be the 0-attractor, inside the game limited to U, of positions that are in U and have minimal rank 0. Here is the picture of the game restricted to U:



In the original game, if the play begins in a position from A and player 0 plays the attractor strategy on the set A, then the play is bound to either end up in a position in U that has rank 0, or in the set W_0. Let us consider the game restricted to set $U - A$. Since this game does not use rank 0, the induction assumption can be applied to get a partition of $U - A$ into two sets of positions U_0 and U_1, such that on each U_i player U_i has a memoryless winning strategy, assuming that the game is limited to $U - A$:



Here is how the sets U_0, U_1 can be interpreted in terms of the bigger original game. For every $i \in \{0,1\}$, if in the original game, if the play begins in a position from U_i and player i uses the memoryless winning strategy corresponding to U_i, then either the play stays forever in $U - A$ and player i wins, or it eventually leaves $U - A$.
Here is a picture of the original game with all sets:



Lemma 2.9. *U_1 is empty.*

Proof. Consider this memoryless strategy for player 1 in the original game:

- in U_1, use the winning memoryless strategy inherited from the game restricted to $U - A$;

- in W_1, use the winning memoryless strategy from Lemma 2.8;
- in other positions do whatever.

We claim that the above memoryless strategy is winning for all positions from U_1, and therefore U_1 must be empty by assumption on W_1 being all positions where player 1 can win in a memoryless way. Suppose player 1 plays the above strategy, and the play begins in U_1. If the play never leaves $U - A$, then player 1 wins by assumption on the strategy. Suppose that the play does leave $U - A$. If it enters W_0 or A, this would have to be a choice of player 0, but positions with such a choice already belong to W_0 or A. Therefore, if the play leaves $U - A$, then it enters W_1, where player 1 wins as well. ■

In the entire game, consider the following memoryless strategy for player 0:

- in U_0, use the winning memoryless strategy inherited from the game restricted to $U - A$;
- in W_0, use the winning memoryless strategy from Lemma 2.8;
- in A use the attractor strategy to reach rank 0 inside U;
- on other positions, i.e. on W_1, do whatever.

We claim that the above strategy wins on all positions except for W_1, and therefore the theorem is proved. We first observe that the play can never enter W_1, because this would have to be a choice of player 1, and such choices are only possible in W_1. Next we observe that if the play enters W_0, then player 0 wins by assumption on W_0. Other plays will reach positions of rank 0 infinitely often, by using the attractor, or will stay in U_0 from some point on. In the first case, player 0 will win by the assumption on 0 being the minimal rank. In the second case, player 0 will win by the assumption on U_0 being winning for the game restricted to $U - A$.
This completes the proof of memoryless determinacy for parity games, and also of the Büchi-Landweber Theorem.

Exercise solutions

1
Determinisation of ω-automata

Problem 1. *Are the following languages regular:*

1. *prefix of v belongs infinitely often to the fixed regular language of finite words $L \subseteq \Sigma^*$;*
2. *word v contains infinitely many infixes of the form $ab^p a$, where p is prime;*
3. *word v contains infinitely many infixes of the form $ab^p a$, where p is even;*
4. *word v contains arbitrary long infixes in the fixed regular language of finite words L;*
5. *prefix of v belongs infinitely often to the fixed language of finite words $L \subseteq \Sigma^*$ (not necessarily regular).*

Solution. Solutions of the points:

1. YES. Let $\mathcal{A}$ be an automaton for L. We make the same automaton with the same final states and Büchi acceptance condition.
2. NO. Assume that yes and $\mathcal{A}$ is an automaton for this language. Let $\mathcal{A}$ has n states and let $p > n$ be some prime number. The word $(b^p a)^\omega$ belongs to L, so $\mathcal{A}$ has an accepting run on it. Note that in every block b^p some two states are the same. We pump the part b^k between them p times, so

that the block has now the length b^{p+kp}, which is not prime. The now word is accepted by $\mathcal{A}$, because it has an accepting run (which came out from the pumping of an old run), but no block has prime length.

3. YES. It suffices to count length of the block b^k modulo 2 and go into an accepting state on a which finishes such a block.

4. NO. Consider $L = ab^*a$. This language contains no ultimately periodic word, so it would have to be empty to be regular.

5. NO. Let us fix some infinite word u, which is not ultimately periodic. Let L be all its prefixes. When prefix of v belongs infinitely often to L if and only if $v = u$. However language $\{u\}$ is not regular. By the way note that the language $\{w\}$ is regular iff w is ultimately periodic.

■

Problem 2. *Show that language of words "there exists a letter b" cannot be accepted by a nondeterministic automaton with Büchi acceptance condition, where all the states are accepting (but possibly transitions over some letters missing in some states).*

Solution. By reading a^k for any $k \in \mathbb{N}$ we cannot get blocked. Therefore word a^ω also cannot get blocked, which means that it is accepted by the considered automaton, contradiction. ■

Problem 3. *Show that language "finitely many occurrences of letter a" cannot be accepted by a deterministic automaton with Büchi acceptance condition.*

Solution. Assume towards a contradiction that it is accepted by some automaton with n states. Let $w = (ab^n)^\omega$. For any prefix of it of the form $(ab^n)^k$ there should be an accepting state among the last $n+1$ states. Indeed, otherwise a word $(ab^n)^k b^\omega$ would not be accepted. Therefore a run over w visits infinitely many times an accepting state, which means that w is accepted. On the other hand it does not belong to the language, as it has infinitely many letters a. Contradiction. ■

Problem 4. *Show that every language accepted by some nondeterministic automaton with Muller acceptance condition is also accepted by some nondeterministic automaton with Büchi acceptance condition.*

Solution. We have an automaton $\mathcal{A}$ with Muller condition, we will be trying to make an automaton $\mathcal{A}'$ with Büchi condition such that $L(\mathcal{A}') = L(\mathcal{A})$. For every $S \in \mathcal{F}$ we will do a separate gadget in automaton $\mathcal{A}'$ such that acceptance in this gadget is if and only if $\inf(\rho) = S$. At the beginning we just make a copy of $\mathcal{A}$, but such that no states are accepting. Beside that for every $S \in \mathcal{F}$ we add a gadget $\mathcal{A}_S$. The idea is that automaton $\mathcal{A}'$ will jump into the gadget $\mathcal{A}_S$ if it wants to choose that $\inf(\rho) = S$ and now is exactly this moment from which on only states from S will occur. Let $S = \{q_1, \ldots, q_k\}$.
Observe now that if $\inf(\rho) = S$ and there are no states outside of S in ρ then states occurring in ρ have also an infinite subsequence of the form $(q_1 q_2 \cdots q_k)^\omega$. Thus we can just investigate whether there exist such a subsequence. Gadget A_S will be the following. It contains $|S|$ copies of $\mathcal{A}$: $\mathcal{A}_{S,1}, \ldots, \mathcal{A}_{S,|S|}$. The copy $\mathcal{A}_{S,i}$ has only one accepting state: q_i. In the copy $\mathcal{A}_{S,i}$ transitions are like in $\mathcal{A}$ with the only exception that from state q_i we go to the next copy: $\mathcal{A}_{S,(i+1) \mod |S|}$. Now we can easily observe that ρ visits infinitely many times accepting state iff it infinitely many times changes a copy. Therefore it has a subsequence of states of the form $(q_1 q_2 \cdots q_k)^\omega$, so indeed $\inf(\rho) = S$. ∎

Problem 5. *Assume that we have changed the acceptance condition into such which investigates which sets of transitions are visited infinitely often. Does it affect the expressivity of automata? How it is for Büchi acceptance condition? And how for Muller acceptance condition?*

Solution. In both cases (Büchi and Muller) expressivity does not change. Therefore we have to prove four facts: (1) condition with states can be implemented on transitions, (2) condition with transitions can be implemented on states, both points for both Büchi and Muller acceptance conditions.
Let us start

1. We first implement states on transitions for Büchi condition. It is very easy, simply these transitions are final which finish into previously final

states.

2. Now we implement transitions on states for Büchi condition. Let language L be accepted by automaton $\mathcal{A}$ with Büchi acceptance condition on transitions. We make an automaton $\mathcal{A}'$, which has two copies of every state in $\mathcal{A}$. To one copy go all the accepting transitions, while to another one go all the non accepting ones. The outgoing transitions are identical in both copies, the same as in $\mathcal{A}$. All the copies with accepting incoming transitions are final, while the other not (some of the final states may not be reachable, but this is not the problem).

3. Now we implement states on transitions for Muller acceptance condition. This is also easy, set of transitions is accepting iff the set of states into which they go is accepting.

4. Now we implement transitions on states for Muller acceptance conditions. Let automaton $\mathcal{A}$ with Muller condition on transitions accept $L = L(\mathcal{A})$. We make $\mathcal{A}'$ as follows. Every state of $\mathcal{A}$ is split into as many copies in $\mathcal{A}'$ as it has incoming transitions in $\mathcal{A}$. The state of $\mathcal{A}'$ is accepting if the incoming transition in $\mathcal{A}$ was accepting.

■

Problem 6. *Show that nonemptiness is decidable for automata with Muller acceptance condition.*

Solution. It is enough to check whether for some $S \in \mathcal{F}$ there exist a run ρ of an automaton in which $\inf(\rho) = S$, where $\inf(\rho)$ is the set of states which occur infinitely often in ρ. We do this separately for every $S \in \mathcal{F}$. We check whether we graph with only states from S is strongly connected and whether some state from S is reachable from some initial state (now in the situation where we have all the states). ■

2
Games with ω-regular winning conditions

Bibliography

[1] J Richard Buchi and Lawrence H Landweber. Solving Sequential Conditions by Finite-State Strategies. *Transactions of the American Mathematical Society*, 138:295, April 1969.

[2] Alonzo Church. Logic, Arithmetic, and Automata. pages 21–35, 1962.

[3] Yuri Gurevich and Leo Harrington. *Trees, automata, and games*. ACM, May 1982.

[4] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966.

[5] David E Muller and Paul E Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(2-3):267–276, 1987.

[6] Wolfgang Thomas. Languages, Automata, and Logic. In *Handbook of Formal Languages*, pages 389–455. Springer, Berlin, Heidelberg, Berlin, Heidelberg, 1997.