Automata on Infinite Words and Trees

Course notes for the course
"Automata on Infinite Words and Trees"
given by Dr. Meghyn Bienvenu
at Universität Bremen
in the 2009-2010 winter semester

Last modified: January 13, 2010

Contents

Disclaimer:

These course notes were prepared for the benefit of students in the 2009-2010 winter semester automata course at Universität Bremen. These notes borrow heavily from the material found in the following resources:

- Automata Theory and its Applications.
  Khoussainov and Nerode, 2001.
- Automata, Logics, and Infinite Games.
  Grädel, Thomas, and Wilke (Eds.), 2002.
- "Languages, Automata, and Logic" in Handbook of Formal Languages.
  Thomas, 1997.
- Infinite Words: Automata, Semigroups, Logic, and Games.
  Perrin and Pin, 2004.
- Tree Automata Techniques and Applications.
  Available at `http://tata.gforge.inria.fr/`.

No originality of the content is implied.

Chapter 1

Review of Automata on Finite Words

1.1 Main Definitions

Definition 1. A *nondeterministic finite automaton*, or NFA, over an alphabet Σ is a quintuple $\mathcal{A} = (Q, \Sigma, T, I, F)$ where:

- Q is a finite non-empty *set of states*
- Σ is an *alphabet*, i.e. a finite non-empty set of symbols
- $T \subseteq Q \times \Sigma \times Q$ is the *transition relation*
- $I \subseteq Q$ is the *set of initial states*
- $F \subseteq Q$ is the *set of final states*

If for every $q \in Q$ and every $\sigma \in \Sigma$ there is a single tuple (q, σ, q') in T, and there is a single state in I, then $\mathcal{A}$ is said to be a *deterministic finite automaton* (DFA).

Often, it is convenient to represent finite automata using labelled directed graphs. For example, we can represent the finite automaton $\mathcal{A} = \{\{q_0, q_1\}, \{a, b\}, \{(q_0, a, q_0), (q_0, b, q_1), (q_1, b, q_1)\}, \{q_0\}, \{q_1\}\}$ by the graph pictured in Figure 1.1. Notice that the states are nodes, the directed edges give the transitions, initial states are indicated with incoming arrows, and final states are indicated by double edges.

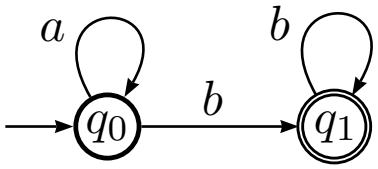

Figure 1.1. A pictorial representation of the DFA $\mathcal{A}$.

Definition 2. A *run* of $\mathcal{A}$ on $w = w_0 w_1 \dots w_n$ starting from q_0 is a sequence

$$q_0 q_1 q_2 \dots q_{n+1}$$

where for all $0 \leq i \leq n$: $(q_i, w_i, q_{i+1}) \in T$. In this case, we say that the word w *transforms* q_0 to q_{n+1}.

Definition 3. An automaton $\mathcal{A} = (Q, \Sigma, T, I, F)$ *accepts* the finite string

$$w = w_0 w_1 \dots w_n$$

if there exists a run

$$q_0 q_1 q_2 \dots q_{n+1}$$

of $\mathcal{A}$ on w such that $q_0 \in I$ and $q_{n+1} \in F$.

Definition 4. The language of $\mathcal{A}$, written $L(\mathcal{A})$, is defined to be the set $\{w \in \Sigma^* \mid \mathcal{A} \text{ accepts } w\}$.

Example 5. The automaton $\mathcal{A}$ from Figure 1.1 accepts the language of strings of the form $a^n b^m$ where $n \geq 0$ and $m \geq 1$.

Definition 6. A language $L \subseteq \Sigma^*$ is said to be *finite automaton recognizable* (or simply, FA recognizable) if there exists a NFA $\mathcal{A}$ such that $L = L(\mathcal{A})$.

1.2 Determinization

Theorem 7. *Let $\mathcal{A}$ be a (possibly nondeterminisitic) automaton. Then there exists a DFA $\mathcal{A}^d$ such that $L(\mathcal{A}) = L(\mathcal{A}^d)$.*

Proof sketch. Let $\mathcal{A} = (Q, \Sigma, T, I, F)$. We construct the desired automaton $\mathcal{A}^d = \{Q^d, \Sigma, T^d, I^d, F^d\}$ as follows:

- $Q^d = 2^Q$

- $(S, \sigma, S') \in T^d$ if and only if $S' = \{q' \mid \exists q \in S : (q, \sigma, q') \in T\}$
- $I^d = I$
- $F^d = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$

This method of determinization is known as the *power set construction.* □

1.3 Closure Properties

Theorem 8. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be finite automata over an alphabet Σ. Then there exists an automaton $\mathcal{A}_3$ such that $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.*

Proof sketch. Let $\mathcal{A}_1 = \{Q_1, \Sigma, T_1, I_1, F_1\}$ and $\mathcal{A}_2 = \{Q_2, \Sigma, T_2, I_2, F_2\}$. We suppose without loss of generality that the set of states of $\mathcal{A}_1$ and $\mathcal{A}_2$ are disjoint, i.e. $Q_1 \cap Q_2 = \emptyset$. An automaton $\mathcal{A}_3 = \{Q_3, \Sigma, T_3, I_3, F_3\}$ with the desired property can be constructed as follows:

- $Q_3 = Q_1 \cup Q_2$
- $T_3 = T_1 \cup T_2$
- $I_3 = I_1 \cup I_2$
- $F_3 = F_1 \cup F_2$ □

Theorem 9. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be finite automata over an alphabet Σ. Then there exists an automaton $\mathcal{A}_3$ such that $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.*

Proof sketch. Let $\mathcal{A}_1 = \{Q_1, \Sigma, T_1, I_1, F_1\}$ and $\mathcal{A}_2 = \{Q_2, \Sigma, T_2, I_2, F_2\}$. We construct an automaton $\mathcal{A}_3 = \{Q_3, \Sigma, T_3, I_3, F_3\}$ with the desired property as follows:

- $Q_3 = Q_1 \times Q_2$
- T is composed of all tuples $((q_1, q_2), \sigma, (q_3, q_4))$ such that $(q_1, \sigma, q_3) \in T_1$ and $(q_2, \sigma, q_4) \in T_2$
- $I_3 = I_1 \times I_2$
- $F_3 = F_1 \times F_2$ □

Theorem 10. *Let $\mathcal{A}$ be a finite automaton. Then there exists an automaton $\mathcal{A}^c$ such that $L(\mathcal{A}^c) = \overline{L(\mathcal{A})} = \Sigma^\star \setminus L(\mathcal{A})$.*

Proof. We can assume without loss of generality that $\mathcal{A} = (Q, \Sigma, T, I, F)$ is deterministic (this is because Theorem 7 shows how every recognizable language is accepted by some deterministic automaton). Then the automaton $\mathcal{A}^c = (Q, \Sigma, T, I, Q \setminus F)$ can be shown to satisfy the condition of the theorem. □

1.4 Myhill-Nerode Theorem

Definition 11. Let L be a language. The words u and v are L-equivalent, written $u \sim_L v$ if for all w, the word uw is in L if and only if the word vw is in L.

Definition 12. Let $\mathcal{A}$ be a finite automaton. The words u and v are $\mathcal{A}$-equivalent, written $u \sim_{\mathcal{A}} v$ if for all states q_1, q_2, u transforms q_1 to q_2 if and only if v transforms q_1 to q_2.

Lemma 13. *Let $\mathcal{A}$ be a finite automaton. The relation $\sim_{\mathcal{A}}$ is an equivalence relation on Σ^* and has finite index (i.e. the relation $\sim_{\mathcal{A}}$ induces a finite number of equivalence classes). Moreover, for all u, v, w: if $u \sim_{\mathcal{A}} v$, then $uw \in L(\mathcal{A})$ iff $vw \in L(\mathcal{A})$.*

Proof. Exercise. □

Theorem 14 (The Myhill-Nerode Theorem)**.** *A language L is finite automaton recognizable if and only if $\sim_L$ is of finite index.*

Proof sketch. Suppose $L = L(\mathcal{A})$ for some finite automaton $\mathcal{A}$. Now if $u \sim_{\mathcal{A}} v$, then by Lemma 13, $uw \in L(\mathcal{A})$ iff $vw \in L(\mathcal{A})$ for all w, and so $u \sim_L v$. So $\sim_L$ cannot have more equivalence classes than $\sim_{\mathcal{A}}$, which means $\sim_L$ has finite index.

For the other direction, suppose $\sim_L$ is of finite index. The basic idea is to use the equivalence classes in $\sim_L$ as the states in the automata. Specifically, we can construct a finite automaton $\mathcal{A} = (Q, \Sigma, T, I, F)$ which accepts L as follows:

- $Q = \{[u] \,|\, u \in \Sigma^*\}$ (where $[u]$ is the equivalence class in $\sim_L$ which contains u)
- T is composed of all tuples of the form $([u], \sigma, [u\sigma])$
- $I = [\lambda]$ (where λ denotes the empty string)
- $F = \{[w] \,|\, w \in L\}$

Note that because $\sim_L$ has finite index, Q has only finitely many states. □

Corollary 15. *If the index $\sim_L$ has n states, then there is a DFA with exactly n states which recognizes L.*

Example 16. Consider the language $L = \{a^m b^m \,|\, m \geq 0\}$. For every m, the word $a^m b^m \in L$, but $a^m b^k \notin L$ for $k \neq m$. It follows that $\sim_L$ has infinite index, and so L is not regular by Theorem 14.

1.5 Kleene's Theorem

Definition 17. Let L_1 and L_2 be languages of finite words. The *concatenation* of L_1 and L_2, denoted $L_1 \cdot L_2$ is defined as follows:

$$L_1 \cdot L_2 = \{uv \,|\, u \in L_1, v \in L_2\}$$

Definition 18. Let L_1 and L_2 be languages of finite words. The *union* of L_1 and L_2, denoted $L_1 + L_2$ is defined as follows:

$$L_1 + L_2 = L_1 \cup L_2$$

Definition 19. Let L be a language of finite words. The *application of the Kleene star ($\star$)* to L, denoted $L^\star$ is defined as follows:

$$L^\star = \bigcup_{i=0}^{\infty} L_i$$

where $L_0 = L$ and $L_{i+1} = L^n \cdot L$ for $i \geq 1$.

Definition 20. A language $L \subseteq \Sigma^\star$ is said to be *regular* if L can be obtained from the empty set, the set $\{\lambda\}$ (containing only the empty word), and the sets $\{a\}$ (for each $a \in \Sigma$) using a finite number of applications of the operators $\cdot$, $\cup$, and $\star$.

Definition 21. We define a regular expression r over the alphabet Σ and the associated language $L(r)$ inductively as follows:

- $r = \emptyset$ is a regular expression with $L(r) = \emptyset$
- $r = \lambda$ is a regular expression with $L(r) = \{\lambda\}$
- $r = a$, for $a \in \Sigma$, is a regular expression with $L(a) = \{a\}$

If r_1 and r_2 are regular expressions with languages $L(r_1)$ and $L(r_2)$, then we have:

- $r = (r_1 \cdot r_2)$ is a regular expression with $L(r) = L(r_1) \cdot L(r_2)$
- $r = (r_1 \cup r_2)$ is a regular expression with $L(r) = L(r_1) \cup L(r_2)$
- $r = r_1^\star$ is a regular expression with $L(r) = L(r_1)^\star$

Theorem 22. *A language L is regular if and only if $L = L(r)$ for some regular expression r.*

Theorem 23 (The Kleene Theorem)**.** *A language L is regular if and only if it is finite automaton recognizable.*

Proof. Suppose that L is regular. Then by Theorem 22, $L = L(r)$ for some regular expression r. We show by induction on the structure of r that L is finite automaton recognizable. The base case is when r is an atomic regular expression. In this case, $L(r)$ is either $\emptyset$, $\{\lambda\}$, or $\{a\}$ for some $a \in \Sigma$, all of which can be recognized by a finite automaton. For the induction step, suppose that r_1 and r_2 are regular expressions such that $L(r_1)$ and $L(r_2)$ are finite automaton recognizable, and let $\mathcal{A}_1 = \{Q_1, \Sigma, T_1, \{q_{I,1}\}, F_1\}$ and $\mathcal{A}_2 = \{Q_2, \Sigma, T_2, \{q_{I,2}\}, F_2\}$ be automata recognizing these languages (we assume w.l.o.g. that their sets of states are disjoint). We now show that $L(r_1 + r_2)$, $L(r_1 \cdot r_2)$, and $L(r_1^\star)$ are also FA recognizable. We consider only the latter two cases, since the first case, the result follows directly from the closure of FA recognizable languages under union.

We start by showing that $L(r_1 \cdot r_2)$ is FA recognizable. The idea is simple: our (nondeterministic) automata will simulate $\mathcal{A}_1$, and when a final state is reached, it will either continue the simulation of $\mathcal{A}_1$, or switch over to simulating $\mathcal{A}_2$ from $\mathcal{A}_2$'s start state. Formally, the desired automaton $\mathcal{A}_3 = \{Q_3, \Sigma, T_3, I, F)$ is as follows:

- $Q_3 = Q_1 \cup Q_2$
- T_3 contains T_1, T_2, plus all tuples (s, σ, s') such that $s \in F_1$ and $(q_{I,2}, \sigma, s') \in T_2$
- $I_3 = \{q_{I,1}\}$
- $F_3 = F_2$

To show that $L(r_1^\star)$ is recognizable, we create an automaton which simulates $\mathcal{A}_1$, and whenever a final state is reached, it either continues the simulation, or starts a new simulation of $\mathcal{A}_1$ from the initial state. Formally, the desired automaton is just like $\mathcal{A}_1$ (which we assume w.l.o.g. to have no transitions to its initial state), except we additionally have transitions of the form $(s, \sigma, s_{I,1})$ whenever s is such that $T(s, \sigma) \in F_1$.

Now we need to show the second part of the theorem, namely that every FA recognizable language is regular. So let us consider some FA recognizable language L, and let $\mathcal{A} = (Q, \Sigma, T, q_I, F)$ be a deterministic automaton with $L(\mathcal{A}) = L$. For $q_1, q_2 \in Q$ and $X \subseteq Q$, we use $G(q_1, q_2, X)$ to refer to the set of all words which transform q_1 to q_2, while passing only through intermediate states in X. We can show by induction on the cardinality of the set X that every set $G(q_1, q_2, X)$ is regular. The base case is when X is the empty set: in this case, there can be no intermediate states, so $G(q_1, q_2, \emptyset)$ consists of all $\sigma \in \Sigma$ such that $(q_1, \sigma, q_2) \in T$. For the induction hypothesis, we suppose the result holds for $|X| \leq n$. Then we take a set Y with $n + 1$ elements, and we use Y_q to refer to the set $Y \setminus \{q\}$. It can then be shown that the set $G(q_1, q_2, X)$ is equal to

$$(\bigcup_{q \in Y} G(q_1, q_2, Y_q)) \cup (\bigcup_{q \in Y} G(q_1, q, Y_q) \cdot [G(q, q, Y_q)]^* \cdot G(q, q_2, Y_q))$$

which is clearly regular, as it is obtained from the regular sets Y_q using $\cdot$, *, and $\cup$. This means that

$$L(\mathcal{A}) = \bigcup_{f \in F} G(q_I, f, Q)$$

must also be regular, completing the proof. □

1.6 Pumping Lemma

Theorem 24 (The Pumping Lemma)**.** *Let $\mathcal{A}$ be a finite automaton which has exactly n states, and let*

$$u = \sigma_0 \dots \sigma_m$$

be a word of length at least n. Then if u is accepted by $\mathcal{A}$, there must exist a non-empty substring $v = \sigma_i \dots \sigma_{i+j}$ of u such that for every $k \geq 0$, the automaton $\mathcal{A}$ accepts all strings of the form

$$\sigma_0 \dots \sigma_{i-1} v^k \sigma_{i+j+1} \dots \sigma_m$$

Proof sketch. Let $\mathcal{A} = (Q, \Sigma, T, I, F)$ be a finite automaton with n states, and let $u = \sigma_0 \dots \sigma_m$ be a word of length at least n which is accepted by $\mathcal{A}$. There must be a run

$$q_0 \dots q_{m+1}$$

of $\mathcal{A}$ on u such that $q_0 \in I$ and $q_{m+1} \in F$. Now since $m \geq n$ and $\mathcal{A}$ has only n states, there must be some state which occurs twice in the run $q_0 \dots q_{m+1}$, i.e. we can find positive integers i and j such that $q_i = q_{i+j}$. This means that

$$q_0 \dots q_i (q_{i+1} \dots q_{i+j})^k q_{i+j+1} \dots q_{m+1}$$

is a successful run of $\mathcal{A}$ on the word

$$\sigma_0 \dots \sigma_i (\sigma_{i+1} \dots \sigma_{i+j})^k \sigma_{i+j+1} \dots \sigma_m$$

for all $k \geq 0$. □

Example 25. Suppose that the language $L = \{a^m b^m \mid m \geq 0\}$ is recognizable, and let $\mathcal{A}$ be an automaton such that $L(\mathcal{A}) = L$. Let n be the number of states of $\mathcal{A}$. As $a^n b^n$ is a word in L with length greater than n, by Theorem 24, $a^n b^n$ can be decomposed into uvw such that v is non-empty, and for every $k \geq 0$, the word $uv^k w$ also belongs to L. But this is a contradiction, because $uv^2 w$ has either a different number of a's and b's, or has some b's which appear before a's, and in either case, it cannot belong to L.

1.7 Decision Problems

Theorem 26. *The emptiness problem (is $L(\mathcal{A}) = \emptyset$?) is decidable.*

Proof sketch. It follows from the pumping lemma that if $\mathcal{A}$ accepts some word, then $\mathcal{A}$ accepts some word of length at most n, where n is the number of states of $\mathcal{A}$. Since there are only finitely many strings of length at most n over a finite alphabet, we can simply test for each of these strings whether it is accepted by $\mathcal{A}$. If some such string is accepted, then $L(\mathcal{A}) \neq \emptyset$, and if all strings fail, then $L(\mathcal{A}) = \emptyset$. □

Theorem 27. *The equivalence problem (is $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?) is decidable.*

Proof sketch. We remark that $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ if and only if both $L(\mathcal{A}_1) \setminus L(\mathcal{A}_2) = \emptyset$ and $L(\mathcal{A}_2) \setminus L(\mathcal{A}_1) = \emptyset$. Using our closure properties, we can construct automata whose languages are precisely $L(\mathcal{A}_1) \setminus L(\mathcal{A}_2)$ and $L(\mathcal{A}_2) \setminus L(\mathcal{A}_1)$. We then use the previous theorem to decide emptiness. □

Theorem 28. *The universality problem (is $L(\mathcal{A}) = \Sigma^\star$?) is decidable.*

Proof sketch. We remark that $L(\mathcal{A}) = \Sigma^\star$ if and only if $\Sigma^\star \setminus L(\mathcal{A}) = \emptyset$. That means we can use closure under complementation to construct an automata with language $\Sigma^\star \setminus L(\mathcal{A})$ and then apply the above emptiness test to this automaton. □

Chapter 2

Automata on Infinite Words

2.1 Basic Notions

Definition 29. An *infinite word* (or infinite string) over an alphabet Σ is a function α from the set ω of natural numbers to Σ. We often write α as

$$\alpha_0\alpha_1\alpha_2\ldots$$

We use $\alpha[m,n]$ (where $m \leq n$) to refer to the finite string $\alpha_m\alpha_{m+1}\ldots\alpha_{n-1}\alpha_n$, and $\alpha(n)$ to refer to the symbol α_n.

Definition 30. We use Σ^ω to refer to the set of all infinite strings over Σ.

Definition 31. An *ω-language* (with respect to the alphabet Σ) is any subset of Σ^ω.

Definition 32. Let W be a language of finite words over Σ. The ω-language W^ω is defined as the set of all infinite strings of the form

$$w_0w_1w_2\ldots$$

such that each w_i is non-empty and belongs to W.

Definition 33. Let W be a language of finite words over Σ. The ω-language $\overrightarrow{W}$ is defined as the set of all infinite strings α such that $\alpha[0,n] \in W$ for infinitely many n.

Definition 34. Let $W \subseteq \Sigma^*$ and $L \subseteq \Sigma^\omega$. The ω-language WL is defined as the set of all infinite strings of the form $w\alpha$ where $w \in W$ and $\alpha \in L$.

2.2 Büchi Automata

Definition 35. A *nondeterministic Büchi automaton* is a quintuple $\mathcal{A} = (Q, \Sigma, T, I, F)$ where:

- Q is a finite non-empty *set of states*
- Σ is an *alphabet*, i.e. a finite non-empty set of symbols
- $T \subseteq Q \times \Sigma \times Q$ is the *transition relation*
- $I \subseteq Q$ is the *set of initial states*
- $F \subseteq Q$ is the *set of final states*

If for every $q \in Q$ and every $\sigma \in \Sigma$ there is at most one tuple of the form (q, σ, q') in T, and there is a single state in I, then $\mathcal{A}$ is said to be *deterministic*.

Definition 36. A *run* of $\mathcal{A}$ on $w = w_0 w_1 w_2 \ldots$ starting from q_0 is an infinite sequence

$$q_0 q_1 q_2 \ldots$$

of states such that for all $i \geq 0$: $(q_i, w_i, q_{i+1}) \in T$.

Definition 37. If $\mathbf{r} = q_0 q_1 q_2 \ldots$ is an infinite sequence of states, then the *infinity set* of $\mathbf{r}$, written $\mathsf{Inf}(\mathbf{r})$, is defined to be the set of states which appear infinitely often in $\mathbf{r}$.

Definition 38. A run $\mathbf{r} = q_0 q_1 q_2 \ldots$ of $\mathcal{A}$ on the string w is said to be *successful* if $q_0 \in I$ and $\mathsf{Inf}(\mathbf{r}) \cap F \neq \emptyset$.

Thus a run is successful if it starts from an initial state and visits some final state infinitely often.

Definition 39. An automaton $\mathcal{A}$ *accepts* (or recognizes) a word w if there is a successful run of $\mathcal{A}$ on w.

Definition 40. The language of $\mathcal{A}$, written $L(\mathcal{A})$, is defined to be the set $\{w \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } w\}$. We say that $\mathcal{A}$ accepts (or recognizes) the language $L(\mathcal{A})$.

Definition 41. A language $L \subseteq \Sigma^\omega$ is said to be *Büchi recognizable* if there exists a Büchi automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.

We conclude the section with a few simple examples of Büchi automata. In the examples, we present the automata using graphs, with nodes for states, directed labelled edges for transitions, an incoming arrow for initial states, and double lines to indicate final states.

Example 42. Consider the Büchi automata represented by the following graph: This automaton accepts all strings which contain a finite block of a's

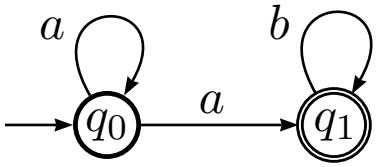

Figure 2.1

followed by an infinite string of only b's. This is because any successful run must contain the final state q_1, only a string of a's can lead to q_1, and once in the state q_1, only b's can be read.

Example 43. Consider the Büchi automata represented by the following graph: This automaton accepts all strings which contain only finitely

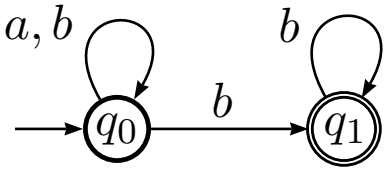

Figure 2.2

many a's. This is because any successful run must contain the final state q_1, any string can lead from the initial state q_0 to q_1, and from state q_1 no further a's can be read.

Example 44. Consider the Büchi automata represented in Figure 2.3. This automaton accepts all strings in which every occurrence of b is doubled, i.e. if there is a substring of the form ab^na, then n must be even.

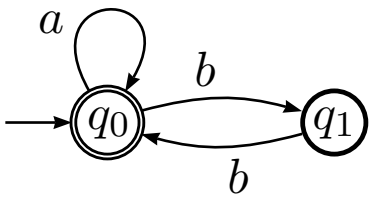

Figure 2.3

2.3 Closure Properties

Theorem 45. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be Büchi automata over an alphabet Σ. Then there exists a Büchi automaton $\mathcal{A}_3$ such that $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.*

Proof. Let $\mathcal{A}_1 = \{Q_1, \Sigma, T_1, I_1, F_1\}$ and $\mathcal{A}_2 = \{Q_2, \Sigma, T_2, I_2, F_2\}$ be Büchi automata. We suppose without loss of generality that $Q_1 \cap Q_2 \neq \emptyset$. The proof is essentially the same as for automata on finite words, but we give it here for completeness. We consider the automaton $\mathcal{A}_3 = \{Q_3, \Sigma, T_3, I_3, F_3\}$ defined as follows:

- $Q_3 = Q_1 \cup Q_2$
- $T_3 = T_1 \cup T_2$
- $I_3 = I_1 \cup I_2$
- $F_3 = F_1 \cup F_2$

We want to show that $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$. For the first direction, suppose that $\mathcal{A}_3$ accepts the infinite word $w = w_0 w_1 w_2 \ldots$. Then there is some run $\mathbf{r} = q_0 q_1 q_2 \ldots$ of $\mathcal{A}$ on w such that $q_0 \in I_3$ and $\mathsf{Inf}(\mathbf{r}) \cap F_3 \neq \emptyset$. If $q_0 \in Q_1$, then we must have $q_i \in Q_1$ for all $i \geq 0$ (since there are no transitions between states in Q_1 and Q_2), and so $\mathsf{Inf}(\mathbf{r}) \cap F_1 \neq \emptyset$. So $\mathbf{r}$ is a successful run of $\mathcal{A}_1$ on w. If instead $q_1 \notin Q_1$, then we must have $q_1 \in Q_2$, and we have that $\mathbf{r}$ is a successful run of $\mathcal{A}_2$ on w. Thus, $w \in L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

For the other direction, we suppose that $w \in L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$. That means that there is a successful run $\mathbf{r} = q_0 q_1 q_2 \ldots$ of either $\mathcal{A}_1$ or $\mathcal{A}_2$ on w. But then $\mathbf{r}$ must also be a successful run of $\mathcal{A}_3$ on w, since $I_1 \cup I_2 \subseteq I_3$, $F_1 \cup F_2 \subseteq F_3$, and $T_1 \cup T_2 \subseteq T_3$. So $w \in L(\mathcal{A}_3)$. □

Theorem 46. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be Büchi automata over an alphabet Σ. Then there exists a Büchi automaton $\mathcal{A}_3$ such that $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.*

Proof. Let $\mathcal{A}_1 = \{Q_1, \Sigma, T_1, I_1, F_1\}$ and $\mathcal{A}_2 = \{Q_2, \Sigma, T_2, I_2, F_2\}$ be Büchi automata. The idea of the proof is as follows: our automaton will simulate both $\mathcal{A}_1$ and $\mathcal{A}_2$. When it reaches a final state of $\mathcal{A}_1$, it notes this (using its states) and then waits until it sees a final state of $\mathcal{A}_2$, at which point it starts looking again for a final state for $\mathcal{A}_1$, and so on. If we switch back and forth infinitely often between these two "modes" (waiting for a final state of $\mathcal{A}_1$, waiting for final state of $\mathcal{A}_2$), then the word will be accepted by our automaton. Formally, we construct the Büchi automaton $\mathcal{A}_3 = \{Q_3, \Sigma, T_3, I_3, F_3\}$ where:

- $Q_3 = Q_1 \times Q_2 \times \{1, 2\}$
- T_3 is composed of the following tuples:
  - $((q_1, q_2, 1), \sigma, (q_3, q_4, 1))$, where $(q_1, \sigma, q_3) \in T_1$, $(q_2, \sigma, q_4) \in T_2$, and $q_1 \notin F_1$
  - $((q_1, q_2, 1), \sigma, (q_3, q_4, 2))$, where $(q_1, \sigma, q_3) \in T_1$, $(q_2, \sigma, q_4) \in T_2$, and $q_1 \in F_1$
  - $((q_1, q_2, 2), \sigma, (q_3, q_4, 2))$, where $(q_1, \sigma, q_3) \in T_1$, $(q_2, \sigma, q_4) \in T_2$, and $q_2 \notin F_2$
  - $((q_1, q_2, 2), \sigma, (q_3, q_4, 1))$, where $(q_1, \sigma, q_3) \in T_1$, $(q_2, \sigma, q_4) \in T_2$, and $q_2 \in F_2$
- $I_3 = I_1 \times I_2 \times \{1\}$
- $F_3 = Q_1 \times F_2 \times \{2\}$

We claim that $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. For the first direction, suppose that $w \in L(\mathcal{A}_3)$. Then there is a run $\mathbf{r} = q_0 q_1 q_2 \dots$ of $\mathcal{A}$ on w such that $q_0 \in I_3$ and $\mathsf{Inf}(\mathbf{r}) \cap F_3 \neq \emptyset$. Each q_i must be of the form (s_i, t_i, n_i) where $s_i \in Q_1$ and $t_i \in Q_2$. The sequence $\mathbf{s} = s_0 s_1 s_2 \dots$ is a run of $\mathcal{A}_1$ on w since $(s_i, w_i, s_{i+1}) \in T_1$ for all $i \geq 0$. It is an accepting run since $s_0 \in I_1$ (by definition of I_3) and $\mathsf{Inf}(\mathbf{r}) \cap F_3 \neq \emptyset$, which means that there are infinitely many states with last coordinate 2, and hence infinitely many i for which $s_i \in F_1$, since we must switch between 1-states and 2-states infinitely often, qnd the only way to switch from 1 to 2 is to pass by some state in F_1. Likewise, we can show that the run $\mathbf{t} = t_0 t_1 t_2 \dots$ is an accepting run of $\mathcal{A}_2$ on w. So $w \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

For the second direction, suppose $w \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. Then there is a accepting run $\mathbf{s} = s_0 s_1 s_2 \dots$ of $\mathcal{A}_1$ on w, and an accepting run $\mathbf{t} = t_0 t_1 t_2 \dots$ of $\mathcal{A}_2$ on w. Consider the following sequence of states

$$(s_0, t_0, n_0)(s_1, t_1, n_1)(s_2, t_2, n_2) \dots$$

where the n_i are defined inductively as follows: $n_0 = 1$; $n_i = 1$ if $n_{i-1} = 1$ and $s_{i-1} \notin F_1$ or if $n_{i-1} = 2$ and $t_{i-1} \in F_2$, and otherwise $n_i = 2$. It is easily verified that this sequence of states defines an accepting run of $\mathcal{A}_3$ on w, which means $w \in L(\mathcal{A}_3)$. □

Complementation will be discussed later in Section 2.9.

2.4 Büchi's Theorem

In this section, we provide a characterization of the set of Büchi recognizable languages in terms of FA recognizable languages. We start with two lemmas.

Lemma 47. *If $W \subseteq \Sigma^*$ is FA recognizable, then the ω-language W^ω is Büchi-recognizable.*

Proof. Let $\mathcal{A}_1 = \{Q_1, \Sigma, T_1, \{q_I\}, F_1\}$ be a DFA which recognizes W. The basic idea is to create a Büchi automaton which simulates $\mathcal{A}_1$ until a final state is reached, and then decides non-deterministically to either continue the simulation, or to start a new simulation of $\mathcal{A}_1$ from the initial state. Formally, we construct a Büchi automaton $\mathcal{A}_2 = \{Q_2, \Sigma, T_2, I_2, F_2\}$ defined as follows:

- $Q_2 = Q_1$
- T_2 contains all the transitions in T_1 plus:
  - all tuples (q, σ, q') such that $q \in F_1$ and $(q_I, \sigma, q') \in T_1$
- $I_2 = I_1 = \{q_I\}$
- $F_2 = F_1$

It can be verified that $\mathcal{A}_2$ indeed recognizes the language W^ω. □

Lemma 48. *If $V \subseteq \Sigma^*$ is FA recognizable and L is Büchi-recognizable, then the ω-language VL is Büchi-recognizable.*

Proof. Let $\mathcal{A}_1 = \{Q_1, \Sigma, T_1, \{q_I\}, F_1\}$ be a DFA which recognizes V, and let $\mathcal{A}_2 = \{Q_2, \Sigma, T_2, I_2, F_2\}$ be a Büchi automaton which recognizes L. We suppose without loss of generality that $Q_1 \cap Q_2 \neq \emptyset$. The idea is as follows: we simulate $\mathcal{A}_1$ until we reach a final state, at which point we either continue the simulation or switch to simulating $\mathcal{A}_2$ from its initial state. Formally, the desired automaton $\mathcal{A}_3 = \{Q_3, \Sigma, T_3, I_3, F_3\}$ can be constructed as follows:

- $Q_3 = Q_1 \cup Q_2$
- T_3 contains T_1, T_2, plus
  - all tuples (q, σ, q') such that $q \in F_1$ and there is some $(q'', \sigma, q') \in T_2$ such that $q'' \in I_2$
- $I_3 = I_1$
- $F_3 = F_2$

It can be checked that the language accepted by $\mathcal{A}_3$ is in fact VL. □

Theorem 49 (Büchi's Theorem)**.** *A language $L \subseteq \Sigma^\omega$ is Büchi recognizable if and only if L is a finite union of sets VW^ω, where V, W are FA recognizable languages.*

Proof. For the first direction, we need to show that we can represent every Büchi recognizable language as a union of languages of the VW^ω for FA recognizable V and W. The idea is quite simple: every word which is accepted by a Büchi automaton must be composed of a prefix which takes the automaton from the initial state to some repeated final state plus an infinite sequence of segments which take this final state back to itself. Since the language of all finite words which take a state q_1 to a state q_2 is FA recognizable, we obtain the desired result. Formally, given a Büchi automata $\mathcal{A} = (Q, \Sigma, T, I, F)$, we consider the languages

$$W_{q_1,q_2} = \{w \mid w \text{ is accepted by the finite automaton } (Q, \Sigma, T, \{q_1\}, \{q_2\})\}$$

By definition, each language W_{q_1,q_2} is FA recognizable. As

$$L(\mathcal{A}) = \bigcup_{q_I \in I, q_F \in F} W_{q_I,q_F} W^\omega_{q_F,q_F}$$

we obtain the desired result.

For the second direction, because we know Büchi recognizable languages to be closed under finite union, all we have to show is that the language VW^ω is Büchi recognizable whenever V, W are FA recognizable languages. So take two FA recognizable languages V and W. By Lemma 47, we know that the language W^ω is Büchi-recognizable because W is FA recognizable. Then by Lemma 48, we know that the language VW^ω must be Büchi-recognizable, as V is FA recognizable and W^ω is Büchi-recognizable. This completes the proof. □

2.5 Deterministic Büchi Automata

For finite strings, deterministic and nondeterministic finite automata have the same expressive power: every FA recognizable language is accepted by some deterministic finite automaton. It is natural to wonder whether the same is true for Büchi automata. We show in this section that this is not the case, i.e. there exist ω-languages that are accepted by a nondeterministic Büchi automaton but not by any deterministic Büchi automaton.

We begin by giving a characterization of the sets of ω-languages which are recognized by deterministic Büchi automata.

Theorem 50. *A language $L \subseteq \Sigma^\omega$ is recognizable by a deterministic Büchi automaton if and only if there exists a FA recognizable language $W \subseteq \Sigma^*$ such that $L = \overrightarrow{W}$.*

Proof. Consider some deterministic Büchi automaton $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, F)$, and let W be the language accepted by $\mathcal{A}$ if it is viewed as a deterministic finite automaton. We intend to show $L(\mathcal{A}) = \overrightarrow{W}$. For the first direction, suppose α is accepted by $\mathcal{A}$. Since $\mathcal{A}$ is deterministic, there is a unique run $\mathbf{r} = q_0 q_1 q_2 \ldots$ of $\mathcal{A}$ on α with $q_0 = q_I$, and this run $\mathbf{r}$ must be such that $\mathsf{Inf}(\mathbf{r}) \cap F \neq \emptyset$. This means that each of the infinitely many finite runs ending in a final state in F determine a prefix of α which belongs to W, so $\alpha \in \overrightarrow{W}$.

For the other direction, suppose that $\alpha \in \overrightarrow{W}$. That means that α has infinitely many prefixes in W, or in other words, the unique run $\mathbf{r}$ of $\mathcal{A}$ on α reaches a final state infinitely often, so $\alpha \in L(\alpha)$. □

We now use Theorem 50 to identify a language which is recognizable only by nondeterministic Büchi automata.

Theorem 51. *There exist Büchi recognizable languages which are recognized by no deterministic Büchi automaton.*

Proof. Consider the language $L = \{\alpha \in \{a, b\}^\omega \mid \alpha \text{ contains finitely many } b\text{'s}\}$. As $L = \Sigma^*\{a\}^\omega$ and Σ^* and $\{a\}$ are both FA recognizable, by Theorem 49, L is Büchi recognizable. Now suppose for a contradiction that L is accepted by a deterministic Büchi automaton. Then by Theorem 50, $L = \overrightarrow{W}$ for some FA recognizable language W. Now since $a^\omega \in L$ and $L = \overrightarrow{W}$, there must be some non-empty string $a^{n_1} \in W$. But since $a^{n_1} b a^\omega$ is also in L, we can find a string $a^{n_1} b a^{n_2} \in W$. Continuing in this way we construct an infinite string

$$a^{n_1} b a^{n_2} b a^{n_3} b a^{n_4} \ldots$$

which belongs to L since it has infinitely many prefixes in W. But this string has infinitely many b's, so it cannot belong to L. Thus, we have shown it to be impossible for a deterministic Büchi automaton to accept L. □

The proof of the previous theorem also shows that the set of languages recognized by deterministic Büchi automata is not closed under complementation. This is because the complement of the language L used in the proof is the language consisting of those strings having infinitely many b's, which is recognizable by a deterministic Büchi automata.

2.6 Müller Automata

In the last section, we saw that deterministic Büchi automata are strictly weaker than nondeterministic ones. This is a disappointing result since it is often proves more convenient to work with deterministic automata (e.g. for complementation). This is why in the following sections we investigate other ways of defining deterministic automata on infinite words. The real difference with these new types of automata will be in the way in which we define successful runs. For Büchi automata, we specified a set of final states F, and we said that a run $\mathbf{r}$ was successful if $\mathsf{Inf}(\mathbf{r}) \cap F \neq \emptyset$. But as we shall see, there are other interesting conditions that can be placed on the set $\mathsf{Inf}(\mathbf{r})$. For instance, for Müller automata, which we consider in this section, the condition is that the set $\mathsf{Inf}(\mathbf{r})$ belongs to a given set of distinguished sets of states $\mathcal{F}$.

Definition 52. A *Müller automaton* is a quintuple $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \mathcal{F})$ where:

- Q is a finite non-empty *set of states*
- Σ is an *alphabet*, i.e. a finite non-empty set of symbols
- $T : Q \times \Sigma \to Q$ is the *transition function*
- q_I is the *initial state*
- $\mathcal{F} \subseteq 2^Q$ is the *set of final states*

The definition of runs, acceptance, and $L(\mathcal{A})$ for Müller automata (and for the other automata we introduce in the following sections) is the same as for Büchi automata. However, the definition of a successful run is different.

Definition 53. A run $\mathbf{r} = q_0 q_1 q_2 \ldots$ of a Müller automaton $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \mathcal{F})$ on w is said to be *successful* if $q_0 = q_I$ and $\mathsf{Inf}(\mathbf{r}) \in \mathcal{F}$.

Thus, a Müller automaton accepts a word w if the infinity set of the unique run of $\mathcal{A}$ on w belongs to $\mathcal{F}$.

Definition 54. A language $L \subseteq \Sigma^\omega$ is said to be *Müller recognizable* if there exists a Müller automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.

We illustrate the above definitions with an example.

Example 55. Consider the Müller automata represented in Figure 2.4. If

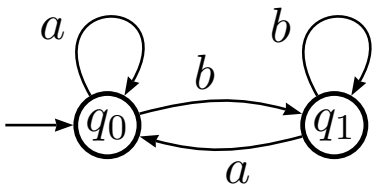

Figure 2.4

we have $\mathcal{F} = \{\{q_0\}\}$, then that means that successful runs visit q_0 infinitely often and q_1 finitely often. Since every occurrence of a leads to q_0, and every b leads to q_1, it follows that the language of this automaton is the set of strings with finitely many b's. If instead we have $\mathcal{F} = \{\{q_1\}\}$, we get the set of strings with finitely many a's. Finally, if $\mathcal{F} = \{\{q_0\}, \{q_1\}\}$, this describes the language of strings having either finitely many a's or finitely many b's.

We remark that the previous example shows that there are languages which are recognized by Müller automata but not by deterministic Büchi automata. The following two theorems further clarify the relationship between the sets of Büchi and Müller recognizable languages.

Theorem 56. *Every Müller recognizable language is Büchi recognizable.*

Proof. Let L be a Müller recognizable language, and let $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \mathcal{F})$ be a Müller automaton with $L = L(\mathcal{A})$. We remark that

$$L = \bigcup_{S \in \mathcal{F}} L_S$$

where L_S is the language recognized by $\mathcal{A}_S = (Q, \Sigma, T, \{q_I\}, \{S\})$. Since Büchi recognizable languages are closed under union (Theorem 45), it suffices to show that every L_S is Büchi recognizable. So consider some $\mathcal{A}_S = (Q, \Sigma, T, \{q_I\}, \{S\})$. The idea is as follows: our Büchi automaton will simulate $\mathcal{A}_S$, then at some point, it will nondeterministically switch into a new mode in which it simulates $\mathcal{A}_S$ *and* keeps track of the states in S which are being visited. Intuitively, the switch to a new mode marks the point at which all of the future states in the run must belong to S, and we need to keep track of the states visited in order to ensure that all of the states in S are visited infinitely often. Now whenever the set of states we are keeping track of is equal to S, we reset the set of states to $\emptyset$, and continue as before. If at any point we encounter a state that is not in S, there will be no possible transitions (so we reject the word), and if we reset the set of states infinintely often to $\emptyset$, then we accept. Formally, we construct a Büchi automaton $\mathcal{A}' = \{Q', \Sigma, T', \{q_I\}, F)$ as follows:

- $Q' = Q \cup \{(q, U) \mid q \in S, U \subseteq S\}$
- T' is composed of the tuples:
  - (q, σ, r), where $T(q, \sigma) = r$
  - $(q, \sigma, (r, \emptyset))$, where $T(q, \sigma) = r$
  - $((q, U), \sigma, (r, \emptyset))$, where $T(q, \sigma) = r$, $U = S$
  - $((q, U), \sigma, (r, V))$, where $T(q, \sigma) = r$, $U \neq S$, and $V = U \cup \{r\}$
- $F = \{(q, \emptyset) \mid q \in S\}$

We want to show $L(\mathcal{A}') = L_S$. For the first direction, suppose that $w \in L(\mathcal{A}')$. Then there is some run $\mathbf{r}$ of $\mathcal{A}'$ on w starting from q_I such that some state $(q, \emptyset)$ appears infinitely often in $\mathbf{r}$. Now consider the sequence of states $\mathbf{s}$ which is defined by $\mathbf{s}(i) = q$ when either $\mathbf{r}(i) = q$ or $\mathbf{r}(i) = (q, U)$ for some U. It is not hard to see that $\mathbf{s}$ is a run of $\mathcal{A}_S$ on w from q_I. Moreover, $\mathbf{s}$ can contain only finitely many occurrences of states outside S, otherwise $\mathbf{r}$ could not be a run of $\mathcal{A}'$. Finally, since the state $(q, \emptyset)$ occurs infinitely often in $\mathbf{r}$, it must be the case that every state in S occurs infinitely often in $\mathbf{s}$. So $\mathbf{s}$ is a successful run of $\mathcal{A}_S$ on w, so $w \in L(\mathcal{A}_S)$.

For the second direction, suppose that $w \in L(\mathcal{A}_S)$. Let $\mathbf{r} = q_0 q_1 q_2 \ldots$ be the successful run of $\mathcal{A}_S$ on w. Then $q_0 = q_I$ and $\mathsf{Inf}(\mathbf{r}) = S$. Let $j \geq 1$ be such that $q_i \in S$ for all $i \geq j$. Now define a sequence $\mathbf{s}$ of states of Q' inductively as follows: $\mathbf{s}(i) = q_i$ for $i < j$, $\mathbf{s}(j) = (q_j, \emptyset)$, and for $i > j$, $\mathbf{s}(i) = (q_i, U \cup \{q_i\})$ if $\mathbf{s}(i) = (q_{i-1}, U)$ and $U \neq S$, and $\mathbf{s}(i) = (q_i, \emptyset)$

otherwise. It is not hard to show that $\mathbf{s}$ is a run of $\mathcal{A}'$ on w from q_I. Moreover, since $\mathsf{Inf}(\mathbf{r}) = S$, each state in S appears infinitely often as the first component of a state in $\mathbf{s}$, and so there will be infinitely many states in $\mathbf{s}$ with $\emptyset$ as their second component, i.e. $\mathsf{Inf}(\mathbf{s}) \cap F \neq \emptyset$. This means w is accepted by $\mathcal{A}'$. □

Later in the chapter, we will see that the converse of Theorem 56 also holds, i.e. every Büchi recognizable language is also Müller recognizable. For now, we prove the following much easier result relating deterministic Büchi automata to Müller automata.

Theorem 57. *Every language which is recognized by a deterministic Büchi automaton is recognized by some Müller automaton.*

Proof. Consider some deterministic Büchi automaton $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, F)$. We construct the Müller automaton $\mathcal{A}' = (Q, \Sigma, T', \{q_I\}, \mathcal{F})$ as follows:

- $T'(q, \sigma) = q'$ if $(q, \sigma, q') \in T$
- $\mathcal{F} = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$

Note that the definition of T' is well-defined, since there can be at most one tuple of the form (q, σ, q') in T. It can be easily verified that $\mathcal{A}$ and $\mathcal{A}'$ accept the same language. □

We end this section with some closure properties of Müller automata.

Theorem 58. *The set of Müller recognizable languages is closed under complementation, union, and intersection.*

Proof. For complementation, consider some Müller recognizable language L, and let $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \mathcal{F})$ be a Müller automaton with $L = L(\mathcal{A})$. Now consider the Müller automaton $\mathcal{A}' = (Q, \Sigma, T, \{q_I\}, 2^Q \setminus \mathcal{F})$. A word α is in L if and only if $\mathsf{Inf}(\mathbf{r}) \in \mathcal{F}$ for the unique run of $\mathcal{A}$ on α from q_I if and only if $\mathsf{Inf}(\mathbf{r}) \notin 2^Q \setminus \mathcal{F}$ if and only if $\mathcal{A}'$ does not accept α. Thus, $L(\mathcal{A}') = \Sigma^\omega \setminus L$.

For union, consider Müller recognizable languages L_1 and L_2, and Müller automata $\mathcal{A} = \{Q_1, \Sigma, T_1, \{q_{I,1}\}, \mathcal{F}_1\}$ and $\mathcal{A} = \{Q_2, \Sigma, T_2, \{q_{I,2}\}, \mathcal{F}_2\}$ such that $L_1 = L(\mathcal{A}_1)$ and $L_2 = L(\mathcal{A}_2)$. We construct our new Müller automaton $\mathcal{A} = \{Q_3, \Sigma, T_3, \{q_{I,3}\}, \mathcal{F}_3\}$ as follows:

- $Q_3 = Q_1 \times Q_2$
- $T_3((q_1, q_2), \sigma) = (q_3, q_4)$ where $T(q_1, \sigma) = q_3$ and $T(q_2, \sigma) = q_4$

- $q_{I_3} = (q_{I,1}, q_{I,2})$
- $\mathcal{F}$ contains all sets $\{(q_1, q'_1), \ldots, (q_n, q'_n)\}$ such that $\{q_1, \ldots, q_n\} \in \mathcal{F}_1$ or $\{q'_1, \ldots, q'_n\} \in \mathcal{F}_2$

It is straightforward to show that either $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2) = L_1 \cup L_2$.

For intersection, we simply note that

$$L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$$

so closure under intersection follows directly from closure under union and complementation. □

2.7 Rabin Automata

In this section, we introduce Rabin automata, which are another type of deterministic automata on infinite words.

Definition 59. A *Rabin automaton* is a quintuple $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \Omega)$ where:

- Q is a finite non-empty *set of states*
- Σ is an *alphabet*, i.e. a finite non-empty set of symbols
- $T : Q \times \Sigma \to Q$ is the *transition function*
- q_I is the *initial state*
- $\Omega = \{(E_1, F_1), \ldots, (E_k, F_k)\}$ is the *set of accepting pairs*

Intuitively, each accepting pair (E_i, F_i) in a Rabin automaton specifies a set E_i of states we want to avoid visiting infinitely often and a set F_i of states we would like to visit infinitely often. This yields the following notion of a successful run.

Definition 60. A run $\mathbf{r} = q_0 q_1 q_2 \ldots$ of a Rabin automaton $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \Omega)$ on w is said to be *successful* if $q_0 = q_I$ and for some $(E_i, F_i) \in \Omega$ we have $\mathsf{Inf}(\mathbf{r}) \cap F_i \neq \emptyset$ and $\mathsf{Inf}(\mathbf{r}) \cap E_i = \emptyset$.

Thus, a Rabin automaton accepts a word w if for some accepting pair (E_i, F_i) the unique run of $\mathcal{A}$ visits infinitely often some state in F_i and finitely often every state in E_i.

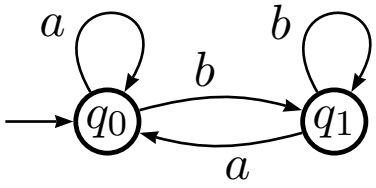

Figure 2.5

Example 61. Consider the Rabin automata presented in Figure 2.5. If the set Ω contains a single accepting pair $(\{q_1\}, \{q_0\})$, then every successful run visits q_0 infinitely often and q_1 only finitely often. It follows that the language accepted by the automaton is the set of strings with finitely many b's. If instead Ω has a single pair $(\{q_0\}, \{q_1\})$, we get the set of strings with finitely many a's. Finally, if $\mathcal{F} = \{\emptyset, \{q_1\}\}$, we obtain the language of strings having an infinite number of b's.

We can show that Müller and Rabin automata recognize the same set of ω-languages.

Theorem 62. *Every language which is recognized by a Rabin automaton is recognized by some Müller automaton.*

Proof. Let $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \Omega)$ be some Rabin automaton. Then it is easy to see that the Müller automaton $\mathcal{A}' = (Q, \Sigma, T, \{q_I\}, \mathcal{F})$ with

$$\mathcal{F} = \{S \subseteq Q \mid S \cap E_i = \emptyset \text{ and } S \cap F_i \neq \emptyset \text{ for some } (E_i, F_i) \in \Omega)$$

accepts exactly the same inputs as $\mathcal{A}$. □

Corollary 63. *Every language which is recognized by a Rabin automaton is recognized by some Büchi automaton.*

Proof. Direct consequence of Theorems 56 and 62. □

Theorem 64. *Every language which is recognized by some Müller automaton is recognized by some Rabin automaton.*

Proof. Let $L = L(\mathcal{A})$ for some Müller automaton $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \mathcal{F})$, where $\mathcal{F} = \{S_1, \ldots, S_n\}$. The basic idea is to simulate this automaton, while keeping track of the subset of each S_i which has been visited. When we have seen all elements in S_i, we "empty" that set of states, and start

again from $\emptyset$. We accept an input if for some i, the ith set of states is reset infinitely often *and* the elements outside S_i are visited only finitely often. Formally, we create the Rabin automaton $\mathcal{A}' = \{Q', \Sigma, T', \{q'_I\}, \Omega)$ where:

- $Q' = \{(U_1, \ldots, U_n, q) \,|\, U_i \subseteq S_i, q \in Q\}$
- $T'((U_1, \ldots, U_n, q), \sigma) = (U'_1, \ldots, U'_n, q')$ where
  - $T(q, \sigma) = q'$
  - $U'_i = \emptyset$ if $U_i = S_i$, and $U'_i = (U_i \cup \{q'\}) \cap S_i$ otherwise
- $q'_I = (\emptyset, \ldots, \emptyset, q_I)$
- $\Omega = \{(E_1, F_1), \ldots, (E_n, F_n)\}$ where:
  - $E_i = \{(U_1, \ldots, U_n, q) \,|\, q \notin S_i\}$
  - $F_i = \{(U_1, \ldots, U_n, q) \,|\, U_i = S_i\}$

For the first direction, suppose $w \in L(\mathcal{A})$, and let $\mathbf{r}$ be the successful run of $\mathcal{A}$ on w. Then we must have $\mathsf{Inf}(\mathbf{r}) = S_i$ for some $1 \leq i \leq n$. Now let $\mathbf{r}'$ be the run of $\mathcal{A}'$ on w from q'_I. Now since $\mathsf{Inf}(\mathbf{r}) = S_i$, it follows that $U_i = \emptyset$ infinitely often during the run $\mathbf{r}'$, so $\mathsf{Inf}(\mathbf{r}') \cap F_i \neq \emptyset$. Also, we know that the states outside S_i appear only finitely often in $\mathbf{r}$, and hence only finitely often as a last component of states in $\mathbf{r}'$, so $\mathsf{Inf}(\mathbf{r}') \cap E_i = \emptyset$. It follows that w is accepted by $\mathcal{A}'$. The second direction of the proof is left as an exercise. □

Using Theorems 62 and 64, we obtain the following closure properties.

Corollary 65. *The set of languages recognized by Rabin automata is closed under union, intersection, and complementation.*

Proof. Theorems 62 and 64 tell us that the set of languages recognizable by Rabin automata is precisely the set of Müller recognizable languages. Since the set of Müller recognizable languages is known to be closed under union, intersection, and complementation (Theorem 58), it follows the same must be true for the set of languages recognized by Rabin automata. □

2.8 Streett Automata

We introduce another variety of deterministic automata on infinite words.

Definition 66. A *Streett automaton* is a quintuple $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \Omega)$ where:

- Q is a finite non-empty *set of states*
- Σ is an *alphabet*, i.e. a finite non-empty set of symbols
- $T : Q \times \Sigma \to Q$ is the *transition function*
- q_I is the *initial state*
- $\Omega = \{(E_1, F_1), \ldots, (E_k, F_k)\}$ is the *set of fairness conditions*

Formally, Streett automata are defined the same as Rabin automata, but we will use a different definition of successful run:

Definition 67. A run $\mathbf{r} = q_0 q_1 q_2 \ldots$ of a Streett automaton $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \Omega)$ on w is said to be *successful* if $q_0 = q_I$ and for every $(E_i, F_i) \in \Omega$ we have either $\mathsf{Inf}(\mathbf{r}) \cap E_i \neq \emptyset$ or $\mathsf{Inf}(\mathbf{r}) \cap F_i = \emptyset$.

Thus, a word w is accepted by a Streett automaton if for every fairness condition (E_i, F_i) the unique run of $\mathcal{A}$ visits infinitely often some state in E_i whenever it visits infinitely often some state in F_i.

Example 68. Consider the Streett automata presented in Figure 2.6. If

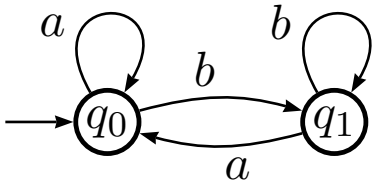

Figure 2.6

the set Ω consists of the pair $(\{q_1\}, \{q_0\})$, then every successful run visits q_1 infinitely often or visits q_0 finitely often. Now the strings which visit q_1 infinitely often are those which contain infinitely many b's, and the strings which visit q_0 finitely often are those which contain finitely many a's (hence infinitely many b's). So the automaton accepts all strings with infinitely many b's. If Ω consists instead of the pair $(\{q_0\}, \{q_1\})$, we obtain the set of strings with infinitely many a's. Finally, if $\Omega = \{(\emptyset, \{q_1\})\}$, then the infinity set of each successful run must either have a non-empty intersection with $\emptyset$ (impossible) or an empty intersection with $\{q_1\}$, i.e. q_1 must be visited finitely often. So the language accepted is the set of infinite words with finitely many b's.

We now consider the relationship between Streett automata and Müller and Rabin automata.

Theorem 69. *Every language which is recognized by a Streett automaton is recognized by a Müller automaton.*

Proof. Suppose we are given a Streett automaton $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \Omega)$. Then it can be easily verified that the Müller automaton $\mathcal{A}' = (Q, \Sigma, T, \{q_I\}, \mathcal{F})$ with

$$\mathcal{F} = \{S \subseteq Q \mid S \cap E_i \neq \emptyset \text{ or } S \cap F_i = \emptyset \text{ for every } (E_i, F_i) \in \Omega\}$$

gives the desired result. □

Theorem 70. *Every language L whose complement is recognized by a Rabin automaton is recognized by some Streett automaton, and vice-versa.*

Proof. Suppose that $L = L(\mathcal{A})$ for a Rabin automaton $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \Omega)$. Then it can be straightforwardly verified that the Streett automaton $\mathcal{A}' = (Q, \Sigma, T, \{q_I\}, \Omega)$ accepts the language $\Sigma^\omega \setminus L$. Conversely, if $L = L(\mathcal{A})$ for a Streett automaton $\mathcal{A} = (Q, \Sigma, T, \{q_I\}, \Omega)$, then the Rabin automaton $\mathcal{A}' = (Q, \Sigma, T, \{q_I\}, \Omega)$ accepts the complement of L. □

Theorem 71. *Müller, Rabin, and Streett automata recognize the same set of ω-languages.*

Proof. We have already shown in Theorems 62 and 64 that Müller and Rabin automata recognize the same set of ω-languages. We also know from Theorem 69 that every Streett recognizable language is Müller recognizable. Thus all that remains to be shown is that Müller recognizable languages are all Streett recognizable. So let us consider some language $L = L(\mathcal{A})$ where $\mathcal{A}$ is a Müller automaton. By Theorem 58, there exists a Müller automaton $\mathcal{A}'$ which recognizes $\Sigma^\omega \setminus L$. Then by Theorem 64 we can find a Rabin automaton $\mathcal{A}''$ such that $L(\mathcal{A}'') = L(\mathcal{A}') = \Sigma^\omega \setminus L$. Finally, by Theorem 70, we can construct a Streett automaton which recognizes $\Sigma^\omega \setminus L(\mathcal{A}'') = L$, which completes the proof. □

Corollary 72. *The set of ω-languages which are recognized by Streett automata is closed under union, intersection, and complementation.*

2.9 Determinization of Büchi Automata

In this section, we prove an important result due to McNaughton which shows that every Büchi recognizable language is recognized by some Rabin automaton (and hence also by Müller and Streett automata). This result shows that in some sense nondeterministic Büchi automata can be determinized.

For automata on finite words, we used the power set construction for constructing deterministic automata from nondeterministic ones. However, such a construction does not give the desired result for Büchi automata. For instance, consider the automaton $\mathcal{A}$ from Figure 2.2 which recognizes the set of infinite strings with only finitely many a's. Applying the power set construction to $\mathcal{A}$ yields the automaton $\mathcal{A}'$ shown in Figure 2.7. We remark

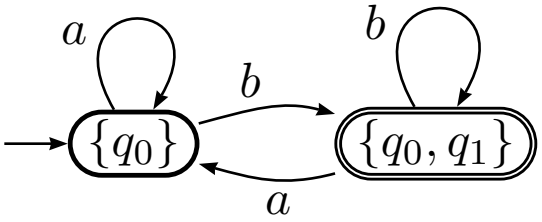

Figure 2.7. The automaton resulting from applying the power set construction to $\mathcal{A}$ from Figure 2.7.

that $\mathcal{A}'$ accepts the string $w = bababa\ldots$ since

$$\{q_0\}\{q_0,q_1\}\{q_0\}\{q_0,q_1\}\{q_0\}\{q_0,q_1\}\ldots$$

is a successful run of $\mathcal{A}$ on w. However, the only run of $\mathcal{A}$ on w is $q_0q_0q_0\ldots$, which is not successful as it does not visit the final state q_1. It follows that $L(\mathcal{A}') \neq L(\mathcal{A})$, which is not very surprising since we showed earlier in the chapter that no deterministic Büchi automata accepts $L(\mathcal{A})$. Notice however that if we consider $\mathcal{A}'$ as a Rabin automaton with a single accepting pair $(\{\{q_0\}\},\{\{q_0,q_1\}\})$ then we would obtain automata which accept the desired language.

One might then wonder whether the power set construction can always be used to construct an appropriate Rabin automaton. We will show that this is not the case. Consider the Büchi automaton in Figure 2.8. Then it can be shown that if the automaton resulting from the power set construction is viewed as a Rabin automaton, then no matter what set of accepting pairs we choose, the accepted language will not be the same as the language of the

original automaton. This means the power set construction can not always be used to transform Büchi automata into equivalent Rabin automata.

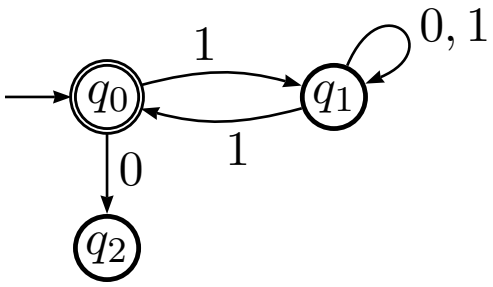

Figure 2.8

In the remainder of this section, we present a proof of McNaughton's theorem which is due to Safra. It provides a procedure for constructing a Rabin automaton which accepts the same language as a given Büchi automaton, which can be seen as a more sophisticated version of the power set construction. In Safra's construction, states in the new automaton are not sets of states (as in the power set construction), but rather trees of sets of states, known as *Safra trees*. Formally, a Safra tree over a set V of node names and a set Q of states is defined as an ordered tree in which the nodes belong to V, are each labelled with a non-empty subset of Q, and can possibly be marked with a special symbol '!'. Additionally, we require that Safra trees satisfy the following conditions:

1. If a node is labelled by S has children labelled with states $S_1, \ldots, S_k$, then $\bigcup_{i=1}^{n} S_i \subsetneq S$.

2. Brother nodes (i.e. those nodes having the same parent) are labelled with disjoint sets of states.

The first condition ensures that the states appearing in the label of a node appear as well in the label of the node's parent and also that at least one state in the node's label appears in none of the labels of its children. The second condition ensures that each state appearing in a node's label appears in the label of at most one of the node's children. We remark that these two conditions together imply that every Safra tree which is labelled with states from the set Q has at most $|Q|$ nodes (this is easily proven by induction of tree height).

We now present the details of Safra's construction. Consider some Büchi automaton $\mathcal{A} = (Q, \Sigma, T, I, F)$, and let $V = \{1, \ldots, 2|Q|\}$. We construct a

Rabin automaton $\mathcal{A}' = (Q', \Sigma, T', \{q_I\}, \Omega)$. We start by defining the transition function T'. The result of applying T' to a symbol σ and a Safra tree with nodes N is computed according to the following steps:

Step 1 Remove mark '!' from all marked nodes.

Step 2 For every node n with label S such that $S \cap F \neq \emptyset$, we create a child $n' \in V \setminus N$ with label $S \cap F$ which is placed to the right of n's other children.

Step 3 Apply the powerset construction to all node labels, i.e. replace a label S with the label $\{s' \mid \exists s \in S \text{ such that } (s, \sigma, s') \in T\}$.

Step 4 (horizontal merge) For every node n with label S, if $s \in S$ belongs to some brother node to the left of n, delete s from n's label.

Step 5 Remove all nodes with empty labels.

Step 6 (vertical merge) For every node n whose label is contained in the union of its children's labels, remove all children of n (and their descendants) and mark n with '!'.

The initial state q_I of $\mathcal{A}'$ is the Safra tree with an unmarked root node 1 with label I. The set Q' of states consists of all Safra trees over V and Q which are reachable from q_I. To complete our description of $\mathcal{A}'$, we need to specify the set of accepting pairs Ω. We set $\Omega = \{(E_v, F_v) \mid v \in V\}$ where:

- E_v consists of all Safra trees which do not contain node v.
- F_v consists of all Safra trees in which node v is marked by '!'.

In order for the above definition of T' to be well-defined, we must check that T' actually maps every Safra tree and input symbol to another Safra tree. We remark that because of Step 5, every node has a non-empty label. Since any new child of a node n starts off with a subset of the n's label, and all other children are known to have labels contained in n's label, and we apply the same power set construction to all nodes in the tree, clearly any state present in a node's label at the end of Step 6 must appear in its parent node. Moreover, because of Step 6, we know that the states appearing in the labels of a node's children must be a proper subset of the node's own label. Finally, because of Step 4, labels of brother nodes must be disjoint. Thus, T' always yields a new Safra tree.

Before proving the correctness of Safra's construction, we illustrate the procedure step-by-step on an example.

Example 73. We consider the Büchi automaton

$$\mathcal{A} = (\{q_0, q_1\}, \{a, b\}, \{(q_0, a, q_0), (q_0, b, q_0), (q_0, b, q_1), (q_1, b, q_1)\}, \{q_0\}, \{q_1\})$$

which was introduced in Figure 2.2. The initial state S_0 of our Rabin automaton will be the Safra tree pictured in Figure 2.9. We first compute

Figure 2.9. The state S_0.

$T'(S_0, a)$. Steps 1 and 2 are inapplicable since S_0 has no marked nodes and does not contain any final states. Step 3 also leaves the tree unchanged, since the only transition for q_0 and a is q_0. Steps 4, 5, and 6 are also inapplicable since there is a single node with a non-empty label. Thus, $T'(S_0, a) = S_0$.

We now compute $T'(S_0, b)$. Again, steps 1 and 2 do not apply. In step 3, the label of the root node changes to $\{q_0, q_1\}$ since both (q_0, b, q_0) and (q_0, b, q_1) belong to T. Steps 4, 5, and 6 do not apply. So we have $T'(S_0, b) = S_1$ where the Safra tree S_1 is pictured in Figure 2.10.

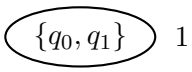

Figure 2.10. The state S_1.

Now we need to compute the transitions for state S_1. First we compute $T'(S_1, a)$. Since there are no marked nodes in S_1, we proceed to step 2, where we add a new child node 2 with label $\{q_1\}$ since $q_1 \in F$. We obtain the tree in Figure 2.11. In Step 3, we apply the power set construction to

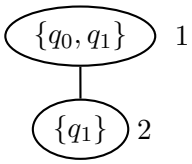

Figure 2.11. After step 2 in computation of $T'(S_1, a)$.

each of the labels. The label $\{q_0, q_1\}$ changes to $\{q_0\}$ and $\{q_1\}$ goes to $\emptyset$ since there is a transition (q_0, a, q_0) but no transition from q_1 via a. We obtain the tree pictured in Figure 2.12. Step 4 is inapplicable since there is

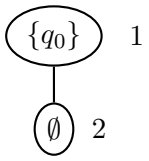

Figure 2.12. After step 3 in computation of $T'(S_1, a)$.

a single child node, and in step 5, we remove the node 2 since its label is empty. Thus, we obtain $T'(S_1, a) = S_0$.

We now compute $T'(S_1, b)$. The first two steps are just as for $T'(S_1, a)$, so at the end of step 2, we have the tree in 2.11. In step 3, we change the label $\{q_0, q_1\}$ to $\{q_0, q_1\}$ and the label $\{q_1\}$ to $\{q_1\}$ since there are transitions (q_0, b, q_0), (q_0, b, q_1), and (q_1, b, q_1) in T. This gives the tree S_2 in Figure 2.13. Steps 4, 5, and 6 are inapplicable, so we end up with $T'(S_1, b) = S_2$.

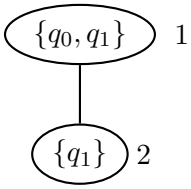

Figure 2.13. The state S_2.

We now calculate $T'(S_2, a)$. Step 1 is still inapplicable since there are no marked nodes, but in Step 2, we add two new nodes, since both nodes 1 and 2 have final states in their labels. We get the tree in Figure 2.14. After step

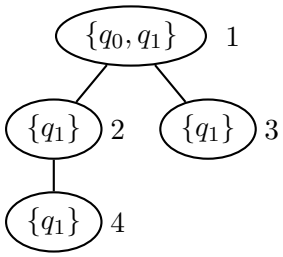

Figure 2.14. After step 2 in computation of $T'(S_2, a)$.

3, the label $\{q_0, q_1\}$ of node 1 is replaced by $\{q_0\}$, and the three node labels $\{q_1\}$ are replaced by $\emptyset$, since there is no transition from q_1 with the symbol a. We thus obtain the tree in Figure 2.15. In Step 5, the three nodes with empty labels are removed, and so the final result is $T'(S_2, a) = S_0$.

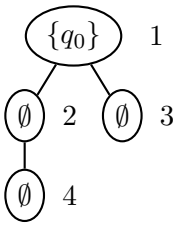

Figure 2.15. After step 2 in computation of $T'(S_2, a)$.

The first two steps of the computation of $T'(S_2, b)$ are the same as for $T'(S_2, a)$. In step 3, we change the labels do not change since T maps $\{q_0, q_1\}$ and b to $\{q_0, q_1\}$ and $\{q_1\}$ and b to $\{q_1\}$. In step 4, we remove q_1 from the label of 3 since q_1 already appears in the node 2 to the left of 3. We thus have the tree in Figure 2.16. The node 3 is removed in step 5 since it has

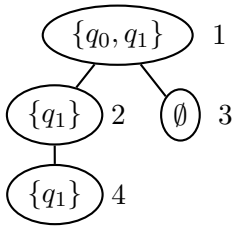

Figure 2.16. After step 4 in computation of $T'(S_2, b)$.

an empty label, and in step 6, we mark node 2 since the unique state in its label is shared by its child node, and we remove the child node 4. We obtain the tree S_3 pictured in Figure 2.17.

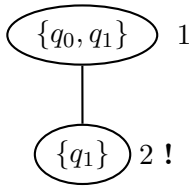

Figure 2.17. The state S_3.

It can be easily verified that $T'(S_3, a) = S_0$ and $T'(S_3, b) = S_3$, since S_2 and S_3 only differ in the marking of the node 2. There are no more transitions to calculate, so we obtain the Rabin automaton $\mathcal{A}'$ pictured in Figure 2.18. The set of accepting pairs of this automaton is composed of $(E_1, F_1) = (\emptyset, \emptyset)$ (since the node 1 appears in all states but is never marked)

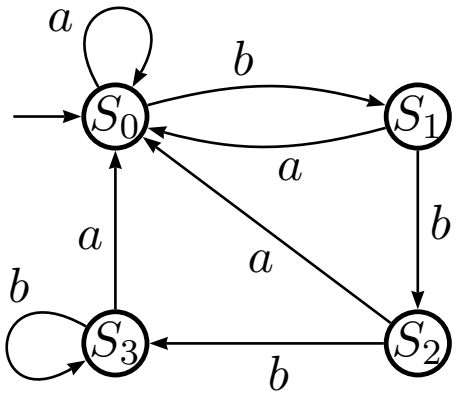

Figure 2.18. The automaton resulting from applying Safra's construction to $\mathcal{A}$ from Figure 2.2.

and $(E_2, F_2) = (\{S_0, S_1\}, \{S_3\})$ (since the node 2 does not appear in S_0 or S_1, and it is marked only in S_3). So the successful runs of our automaton are those in which S_0 and S_1 are visited finitely often and S_3 is visited infinitely often. Now since every occurrence of a leads back to S_0, all accepted words have only finitely many a's. Conversely, every infinite string with finitely many a's is accepted by our automaton since the unique run will eventually get stuck in S_3. Thus, the Rabin automaton we have constructed recognizes the set of strings with finitely many a's, which is precisely $L(\mathcal{A})$.

We now show the correctness of Safra's determinization procedure. We break the proof down into two pieces (Propositions 74 and 75). The first proposition shows that every word accepted by the original Büchi automaton must also be accepted by the Rabin automaton we construct via Safra's procedure.

Proposition 74 (Completeness)**.** *Let $\mathcal{A} = (Q, \Sigma, T, I, F)$ be a Büchi automaton, and let $\mathcal{A}' = (Q', \Sigma, T', \{q_I\}, \Omega)$ be the Rabin automaton obtained by applying Safra's construction to $\mathcal{A}$. Then $L(\mathcal{A}) \subseteq L(\mathcal{A}')$.*

Proof. Suppose that $w \in L(\mathcal{A})$, and let $\mathbf{r} = q_0 q_1 q_2 \ldots$ be an accepting run of $\mathcal{A}$ on w. As $\mathcal{A}'$ is deterministic, there is a unique run of $\mathbf{s} = S_0 S_1 S_2 \ldots$ of $\mathcal{A}'$ on w from q_I. We intend to show that there is some node n such that:

(a) there exists some integer m such that n belongs to S_i for all $i \geq m$

(b) n is marked by '!' in infinitely many S_i

Note that if a node n satisfies (a), then $\mathsf{Inf}(\mathbf{s}) \cap E_n = \emptyset$; if n satisfies (b), then $\mathsf{Inf}(\mathbf{s}) \cap F_n \neq \emptyset$. Thus if we find some node n satisfying both (a) and (b), then we have shown that $\mathbf{s}$ is a successful run of $\mathcal{A}'$, and hence that $w \in L(\mathcal{A}')$.

We remark that by construction the label of the node 1 in the Safra tree S_i contains all states which can be reached from some initial state by reading the first i symbols in w. In particular, this means that S_i must contain the state q_i from the run $\mathbf{r}$. This means the node 1 always has a non-empty label, and so is never removed in step 5 of Safra's procedure. So the root node 1 satisfies condition (a). Now if the node 1 is marked infinitely often with '!', then condition (b) is also satisfied, and we are done. Otherwise, there is some point m such that 1 is unmarked in every S_i with $i > m$.

Since $\mathbf{r}$ is a successful run of $\mathcal{A}$, there must be some state $f \in \mathsf{Inf}(\mathbf{r}) \cap F$. Let p be the index of the first occurrence of f in $\mathbf{r}$ after point m (i.e. we have $q_p = f$ and $q_i \neq f$ for all $m < i < p$). Since $q_p = f$, the state f must appear in the label of node 1 in S_p. That means that during step 2 of the computation of S_{p+1}, a new right-most child will be added to the node 1 and its label will contain f. As the root node 1 remains unmarked for the remainder of the run $\mathbf{s}$, the state q_i will appear in some child of 1 for all $i \geq p + 1$. Note however that because of horizontal merges in step 4, the state q_i might move to the label of another child of 1 which is further to the left. But since all new children are added to the right, there can be only finitely many moves to the left, and so at some point, the state q_i will remain in a fixed child c of the root node 1. Thus, the node c will satisfy condition (a). If c is marked infinitely often, then we have found a suitable node. Otherwise, there is some point m' after which c is permanently unmarked. We can then apply the same argument to c as we did for the node 1. But as the Safra trees S_i have a depth of at most $|Q|$, this procedure must stop at some point, and we will have found a node with the desired properties, which means $w \in L(\mathcal{A}')$. □

We now prove the second proposition, which states that every word which is accepted by the Rabin automaton we constructed must also belong to the language of the Büchi automaton.

Proposition 75 (Soundness)**.** *Let $\mathcal{A} = (Q, \Sigma, T, I, F)$ be a Büchi automaton, and let $\mathcal{A}' = (Q', \Sigma, T', \{q_I\}, \Omega)$ be the Rabin automaton obtained by applying Safra's construction to $\mathcal{A}$. Then $L(\mathcal{A}') \subseteq L(\mathcal{A})$.*

For our proof of Proposition 75, we will require the following result from graph theory.

Theorem 76 (König's Lemma)**.** *Every finitely branching infinite tree has an infinite path.*

In order to make the proof of Proposition 75 a bit easier to read, we introduce the abbreviation $T^*(S, u)$ which is defined recursively as follows: $T^*(S, a) = \{q' \mid \exists q \in S : (q, a, q') \in T\}$ for $a \in \Sigma$, and $T^*(S, wa) = T^*(T^*(S, w), a)$. Intuitively, $T^*(S, u)$ gives the set of states which can be reached from states in S with the finite word u according to the transition relation T. We now proceed with the proof of Proposition 75.

Proof of Proposition 75. Suppose $w \in L(\mathcal{A}')$. Then the unique run $\mathbf{s} = S_0 S_1 S_2 \ldots$ of $\mathcal{A}'$ on w from $q_I = S_0$ is successful. This means that for some node n, both $\mathsf{Inf}(\mathbf{s}) \cap E_n = \emptyset$ and $\mathsf{Inf}(\mathbf{s}) \cap F_n \neq \emptyset$. In other words, there is a point p such that the node n appears in every Safra tree S_i for $i \geq p$, and the node n is marked infinitely often. We let $m_1, m_2, \ldots$ be the increasing infinite sequence composed of all indices $i \geq p$ such that node n is marked in the tree S_i, and we will use L_i to refer to the label of node n in the tree S_i. We use the sequence $m_1 m_2 \ldots$ to split w into a sequence of finite words: $w_1 = w[0, m_1 - 1]$, and $w_{i+1} = w[m_i, m_{i+1} - 1]$ for $i \geq 1$. Thus, we have $w = w_1 w_2 \ldots$ We claim the following:

> For every $i > 0$ and every $q \in L_{m_{i+1}}$, there is a finite run[1] of $\mathcal{A}$ on the word w_{i+1} which starts from some state in L_{m_i}, terminates in the state q, and contains at least one final state which appears before the terminating state.

In order to show this claim, we begin by proving by induction on j that for every $m_i < j \leq m_{i+1} - 1$ and for every state q_j appearing in the label of a descendant of n in S_j, there exists a run of $\mathcal{A}$ on $w[m_i, j-1]$ starting from a state in L_{m_i}, ending with q_j, and containing at least one final state. The base case is when $j = m_i + 1$. Since n is marked in S_{m_i}, there are no children of n in S_{m_i}. Thus, if a state q_j appears in a descendant of n in $S_j = S_{m_i+1}$, then there must have been a child c of n created in step 2 with label $L_{m_i} \cap F$. In step 3, the label of c will be updated to $T^*(L_{m_i} \cap F, w(m_i))$, and this will be the label of c in S_j. So we must have $q_j \in T^*(L_{m_i} \cap F, w(m_i))$, which means that there is a state $s \in L_{m_i} \cap F$ such that $(s, w(m_i), q_j) \in T$. But then sq_j is a run of $\mathcal{A}$ on $w[m_i, m_i]$ starting from a state in L_{m_i}, terminating

[1] Here we treat the Büchi automaton $\mathcal{A}$ as an automaton on finite words.

in q_j, and passing by a final state (namely s). For the induction step, assume that we have the result for all $m_i < j \leq k < m_{i+1} - 2$, and consider the case where q_{k+1} appears in a descendant c of node n in S_{k+1}. Then there are two possibilities: either the node c was already present in S_k, or the node c was created in step 2 of Safra's procedure. In the first case, there is a state q_k in the label of c in S_k such that $(q_k, w(k), q_{k+1}) \in T$. By the induction hypothesis, there is a run of $\mathcal{A}$ on $w[m_i, k-1]$ beginning in L_{m_i}, terminating in q_k, and passing by some final state. If we simply add q_{k+1} to the end of this run, then we get a run satisfying the desired conditions for q_{k+1}. The other possibility is that we created the node c in step 2. That means that c has label $T^*(L_k \cap F, w(k))$, and so there must be some $q_k \in L_k \cap F$ with $(q_k, w(k), q_{k+1}) \in T$. Then any run of $\mathcal{A}$ on $w[m_j, k]$ from a state in L_{m_i} which terminates with $q_k q_{k+1}$ will work. Note that there must exist such a run since q_k appears in L_k and $(q_k, w(k), q_{k+1}) \in T$.

To finish the proof of the above claim, suppose $q \in L_{m_{i+1}}$. Since n is marked in $S_{m_{i+1}}$, we know that in step 6, the children of n were removed, and the union of their labels was equal to the label of n. Thus, every state $q \in L_{m_{i+1}}$ appeared in the label of a child c of n at the beginning of step 6. There are two possibilities: either the child c was already present in $S_{m_{i+1}-1}$, or it was created during step 2. In the first case, we can find a state q' in the label of c in $S_{m_{i+1}-1}$ such that $(q', w(m_{i+1} - 1), q) \in T$, and by using what we showed above, we can find a finite run of $\mathcal{A}$ on $w[m_i, m_{i+1} - 1]$ leading from L_{m_i} to q', and passing by a final state. By adding q to the end of this run, we obtain the desired run for q. Now in the second case, there must be some state $q' \in L_{m_{i+1}-1} \cap F$ such that $(q', w(m_{i+1}), q) \in T$. Then any run of $\mathcal{A}$ on $w[m_i, m_{i+1} - 1]$ from a state in L_{m_i} which terminates with $q'q$ will work since we have $q' \in F$. Note that such a run must exist, since every state in $L_{m_{i+1}-1}$ must have been obtained from a state L_{m_i} by applying the transition function with respect to the word $w[m_i, m_{i+1} - 2]$.

We now use the above claim to complete the proof. We start by constructing an infinite tree whose set of nodes is $\{r\} \cup \{(q, 0) \mid q \in I\} \cup \{(q, i) \mid q \in L_{m_i}, i \geq 1\}$. Note that there must be at least one node (q, i) for each i since the labels L_{m_i} are always non-empty. The nodes of the form $(q, 0)$ all have r as parent, and have as children nodes $(q', 1)$ such that $q' \in T^*(q, w_1)$. For each node of the form (q', m_{i+1}), we choose a parent node (q, m_i) such that there is a finite run of $\mathcal{A}$ on the word w_{i+1} which starts from $q \in L_{m_i}$, terminates in the state q', and contains at least one final state. Because of our claim, we know that it is always possible to find at least one such node (q, m_i). Thus, we have an infinite tree which is finitely branching, so by König's Lemma (Theorem 76), there must be an infinite

path in this tree. Choose some such infinite path, and define q_i to be the unique state which appears at level i in the infinite path (i.e. the node (q_i, i) belongs to the path). We now use these states to construct an infinite run of $\mathcal{A}$ on w. First for the initial segment w_1, we take any run r_1 of $\mathcal{A}$ which leads from the state q_0 to the state q_1 on the word w_1; because of the way we have constructed the tree, we know that such a run must exist. Then for any word w_{i+1}, we select a run r_{i+1} which leads from the states q_i and q_{i+1} while reading w_{i+1} and which contains a final state. We know that such a run must exist, since otherwise the nodes (q_i, i) and $(q_{i+1}, i+1)$ could not be connected in the tree. Now by putting these runs altogether, we obtain a run $r = r_1 r_2 r_3 \ldots$ of $\mathcal{A}$ on w which starts from some initial state (since q_0 must be in I) and which passes infinitely often by some final state (since the run r_i has at least one final state, for all $i > 1$). Thus, we have shown $w \in L(\mathcal{A})$. □

Putting Propositions 74 and 75 together, we obtain:

Theorem 77 (McNaughton's Theorem)**.** *Let $\mathcal{A}$ be a Büchi automaton. Then there exists a Rabin automaton $\mathcal{A}'$ with $L(\mathcal{A}') = L(\mathcal{A})$.*

As a corollary to Theorem 77, we are able to show the set of Büchi recognizable languages to be closed under complementation.

Corollary 78. *Let $\mathcal{A}$ be a Büchi automaton. Then there exists a Büchi automaton $\mathcal{A}^c$ such that $L(\mathcal{A}^c) = \Sigma^\omega \setminus L(\mathcal{A})$.*

Proof. Consider some Büchi automaton $\mathcal{A}$. By McNaughton's Theorem, there is a Rabin automaton $\mathcal{A}^d$ such that $L(\mathcal{A}) = L(\mathcal{A}^d)$. Now by Corollary 65, we can find a Rabin automaton $\bar{\mathcal{A}}^d$ which accepts the language $\Sigma^\omega \setminus L(\mathcal{A}^d)$. But then by Corollary 63 we can transform $\bar{\mathcal{A}}^d$ into an equivalent Büchi automaton $\mathcal{A}'$. □

2.10 Decision Problems

We consider the standard decision problems: emptiness, equivalence, and universality. We focus on Büchi automata in this section, but the decidability results immediately transfer to Müller, Rabin, and Streett automata.

Theorem 79. *The emptiness problem (is $L(\mathcal{A}) = \emptyset$?) is decidable for Büchi automata.*

Proof. Let $\mathcal{A} = (Q, \Sigma, T, I, F)$ be a Büchi automata. From the proof of the Büchi theorem (Theorem 49), we know that

$$L(\mathcal{A}) = \bigcup_{q_I \in I, q_F \in F} W_{q_I,q_F} W^{\omega}_{q_F,q_F}$$

where $W_{q,q'}$ denotes the set of strings which lead from q to q'. Thus, $L(\mathcal{A}) \neq \emptyset$ if and only if there is some state $q_F \in F$ such that there is a finite string v taking q_0 to q_F and some finite string w taking q_F to q_F. If we represent $\mathcal{A}$ by a graph, then these conditions correspond to finding a path from q_0 to q_F and a path from q_F to q_F. Thus, emptiness reduces to graph reachability, which is a decidable (in fact linear-time) problem. □

Theorem 80. *The equivalence problem (is $L(\mathcal{A}_1) = L(\mathcal{A}_2)$?) is decidable for Büchi automata.*

Proof. Consider Büchi automata $\mathcal{A}_1$ and $\mathcal{A}_2$. We remark that $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ if and only if both $L(\mathcal{A}_1) \cap \overline{L(\mathcal{A}_2)} = \emptyset$ and $L(\mathcal{A}_2) \cap \overline{L(\mathcal{A}_1)} = \emptyset$. As Büchi automata are closed under complementation (Corollary 78) and intersection (Theorem 46), we can construct Büchi automata whose languages are precisely $L(\mathcal{A}_1) \cap \overline{L(\mathcal{A}_2)}$ and $L(\mathcal{A}_2) \cap \overline{L(\mathcal{A}_1)}$. We then apply Theorem 79 to decide emptiness of these languages. □

Theorem 81. *The universality problem (is $L(\mathcal{A}) = \Sigma^\star$?) is decidable for Büchi automata.*

Proof sketch. We remark that $L(\mathcal{A}) = \Sigma^\star$ if and only if $\Sigma^\star \setminus L(\mathcal{A}) = \emptyset$. We can thus use closure under complementation (Corollary 78) to construct a Büchi automaton with language $\Sigma^\star \setminus L(\mathcal{A})$ and then apply the above emptiness test (Theorem 79) to this automaton. □

Chapter 3

Automata on Finite Trees

3.1 Basic Notions

We denote by $\mathbb{N}$ the set of *positive integers*, and by $\mathbb{N}^\star$ the set of all finite strings over $\mathbb{N}$. We use λ to denote the empty string.

A *finite ordered tree* t over an (unranked) alphabet Γ is a function from a non-empty prefix-closed[1] subset P of $\mathbb{N}^\star$ to Γ.

A *ranked alphabet* Σ is defined to be a non-empty set of symbols each with an associated arity. We will use Σ_m to denote the set of symbols in the ranked alphabet Σ with arity m. For convenience, when presenting a ranked alphabet Σ with symbols $s_1, \ldots, s_n$ of arities $a_1, \ldots, a_n$, we write $\Sigma = \{s_1/a_1, \ldots, s_n/a_n\}$. For example, $\Sigma = \{f/1, g/3, a/0\}$ refers to the ranked alphabet composed of a symbol f of arity 1, a symbol g of arity 3, and a symbol a of arity 0.

A *finite ordered ranked tree* t over the ranked alphabet Σ (or simply a Σ-tree) is a function $t : P \to \Sigma$ which satisfies the following conditions:

- $P \subseteq \mathbb{N}^\star$ is a non-empty prefix-closed set
- if $t(p) \in \Sigma_m$ where $m \geq 1$, then $\{j \mid pj \in P\} = \{1, \ldots, m\}$
- if $t(p) \in \Sigma_0$, then $\{j \mid pj \in P\} = \emptyset$

We will call the set P the *domain* of t, and we write $\mathsf{Pos}(t) = P$. The elements of $\mathsf{Pos}(t)$ are called the *positions* (or nodes) of the tree t. If $p \in \mathsf{Pos}(t)$ and $pi \in \mathsf{Pos}(t)$, then pi is said to be a *child* of the node p. If $p, p' \in \mathsf{Pos}(t)$ and p is a prefix of p', then p' is a descendant of p, and we

[1] We recall that a set S of finite words is *prefix-closed* if $w \in S$ whenever $wv \in S$.

write $p \sqsubseteq p'$. A position p is said to be a *frontier position* (sometimes called a *terminal position* or *leaf node*) if it has no descendants. The set of *frontier positions* in t will be denoted by $\mathsf{FPos}(t)$. The *height* of a tree t is the longest position in $\mathsf{Pos}(t)$. We denote by t_p the *subtree* of t which is based at p; formally, $\mathsf{Pos}(t_p) = \{w \in \mathsf{Pos}(t) \mid p \text{ is a prefix of} w\}$ and $t_p(q) = t(pq)$.

Example 82. Consider the ranked alphabet $\Sigma = \{f/1, g/3, a/0\}$, and let t be a tree with domain $\{\lambda, 1, 11, 12, 13, 131\}$ such that $t(\lambda) = f$, $t(1) = g$, $t(11) = a$, $t(12) = a$, $t(13) = f$, and $t(131) = a$. The tree t has height 3, its root symbol is f, and its set of terminal positions is $\{11, 12, 131\}$. A graphical representation of this tree is given in Figure 3.1.

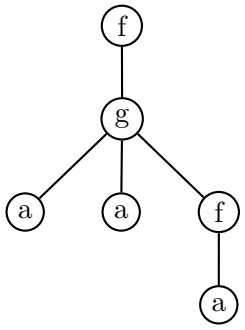

Figure 3.1. An example tree.

3.2 Bottom-up Automata on Finite Trees

Definition 83. A *nondeterministic bottom-up automaton on finite ordered ranked trees* (or NFTA for short) is a quadruple $\mathcal{A} = (Q, \Sigma, T, F)$ where:

- Q is a non-empty *set of states*
- Σ is a *ranked alphabet*
- T is a set of *transition rules* of the form:

  $$f(q_1, \ldots, q_m) \to q$$

  where $m \geq 0$, $f \in \Sigma_m$, $q, q_1, \ldots, q_m \in Q$.
- $F \subseteq Q$ is a set of *final states*

If there do not exist two distinct rules in T with the same left hand side, then the automaton is said to be a *deterministic* finite tree automaton (DFTA).

NFTAs with alphabet Σ operate on Σ-trees. A NFTA can be seen as assigning states to each of the nodes in the tree, starting with the terminal positions and working upwards to the root. The states that are assigned must conform to the set of transition rules T. We now give a formal definition of the behaviour of NFTAs.

Definition 84. A *run* of a NFTA $\mathcal{A} = (Q, \Sigma, T, F)$ on a Σ-tree t is a function $r : \mathsf{Pos}(t) \to Q$ which satisfies the following conditions:

- if $t(p) = a \in \Sigma_0$ and $r(p) = q$, then there is a rule $a \to q$ in T
- if $t(p) = f \in \Sigma_m$ $(m \geq 1)$, $r(p) = q$, and $r(p1) = q_1, \ldots, r(pm) = q_m$, then there is a rule $f(q_1, \ldots, q_m) \to q$ in T

Thus, a frontier position with symbol a can only be assigned a state q for which $a \to q \in T$. For a position p with symbol f whose children have been assigned states $q_1, \ldots, q_m$, we must assign a state q to p such that the transition $f(q_1, \ldots, q_m) \to q$ belongs to T.

Definition 85. A run r of a NFTA $\mathcal{A}$ on a tree t is *successful* if $r(\lambda) \in F$. A tree t is said to be *accepted* by a NFTA $\mathcal{A}$ if there exists a successful run r of $\mathcal{A}$ on t.

Thus, a tree is accepted by a NFTA if there is some way of assigning states to the tree's nodes according to the transition rules such that the root node is labelled with a final state.

Definition 86. The language $L(\mathcal{A})$ of a NFTA $\mathcal{A}$ is defined to be the set of all trees which are accepted by $\mathcal{A}$. We say that $\mathcal{A}$ *recognizes* (or accepts) the language $L(\mathcal{A})$.

Definition 87. A set L of (finite ordered ranked) trees is said to be a *recognizable tree language* if $L = L(\mathcal{A})$ for some NFTA $\mathcal{A}$.

Example 88. Consider the NFTA $\mathcal{A} = (Q, \Sigma, T, F)$ where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{f/2, a/0, b/0\}$, $T = \{a \to q_0, b \to q_0, f(q_0, q_0) \to q_1, f(q_1, q_1) \to q_2, f(q_1, q_1) \to q_3\}$, and $F = \{q_2\}$. In Figure 3.2, we present graphically a tree t, and in Figures 3.3 and 3.4 we give the two runs r and r' of $\mathcal{A}$ on t. As $r(\lambda) = q_2 \in F$, the tree t is accepted. The language of $\mathcal{A}$ is the set of Σ-trees all of whose paths have height 2.

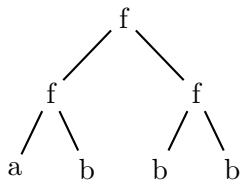

Figure 3.2. Tree t

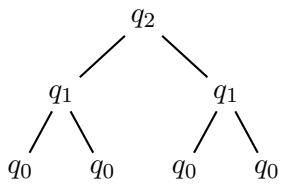

Figure 3.3. Run r

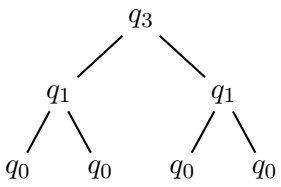

Figure 3.4. Run r'

3.3 Determinization

A natural question is whether deterministic tree automata are as powerful as nondeterministic tree automata. In this section, we show that this is indeed the case. The proof employs the power set construction and is quite similar to the proof of the analogous result for finite words.

Theorem 89. *Let L be a recognizable tree language. Then there exists a DFTA which recognizes L.*

Proof. Consider a recognizable tree language L, and let $\mathcal{A} = (Q, \Sigma, T, F)$ be a NFTA with $L = L(\mathcal{A})$. Consider the DFTA $\mathcal{A}' = (Q', \Sigma, T', F')$ defined as follows:

- $Q' = 2^Q$
- T' is composed of all rules of the form

$$f(S_1, \ldots, S_m) \to S$$

  where $S = \{q \in Q \mid \exists q_1 \in S_1, \ldots, q_m \in S_m : f(q_1, \ldots, q_m) \to q \in T\}$
- $F' = \{S \in Q' \mid S \cap F \neq \emptyset\}$

We will now show that the unique run r' on a tree t satisfies the following property:

$$\text{for all } p \in \mathsf{Pos}(t),\ r'(p) = \{r(\lambda) \mid r \text{ is a run of } \mathcal{A} \text{ on } t_p\}$$

The proof will be by induction on the distance of p from a frontier position. The base case is when p is a frontier position. In this case, we have $r'(p) = \{q \in Q \mid t(p) \to q \in T\}$. By definition, each $q \in Q$ with $t(p) \to q \in T$ gives rise to a run r of $\mathcal{A}$ on t_p where $r(\lambda) = q$. So we have $r'(p) \subseteq \{r(\lambda) \mid r \text{ is a run of } \mathcal{A} \text{ on } t_p\}$ Conversely, if r is a run of $\mathcal{A}$ on t_p, then we must have $r(\lambda) \in \{q \in Q \mid t(p) \to q \in T\}$, so $\{r(\lambda) \mid r \text{ is a run of } \mathcal{A} \text{ on } t_p\} \subseteq t'(p)$. Now for the induction step, let us suppose that the statement holds

for positions in t which are at most k from a frontier position, and let p be a position which is a distance of $k+1$ from a frontier position. By definition of $\mathcal{A}'$, if $t(p) = f \in \Sigma_m$, then

$$r'(p) = \{q \in Q \mid \exists q_1 \in r'(p1), \ldots, q_m \in r'(pm) \text{ with } f(q_1, \ldots, q_m) \to q \in T\}$$

Consider some $q \in r'(p)$, and let $q_1 \in r'(p1), \ldots, q_m \in r'(pm)$ be such that $f(q_1, \ldots, q_m) \to q \in T$. Then by the induction hypothesis, there exist runs $r_1, \ldots, r_m$ of $\mathcal{A}$ on $t_{p1}, \ldots, t_{pm}$ such that $r_{pi}(\lambda) = q_i$ for all $1 \leq i \leq m$. But then if we set $r(\lambda) = q$ and $r(iw) = r_i(w)$ for $1 \leq i \leq m$, we obtain a run of $\mathcal{A}$ on t_p satisfying $r(\lambda) = q$. So $r'(p) \subseteq \{r(\lambda) \mid r \text{ is a run of } \mathcal{A} \text{ on } t_p\}$. For the other half of the equality, suppose that r is a run of $\mathcal{A}$ on t_p with $r(\lambda) = q$, and let $q_i = r(i)$ for $1 \leq i \leq m$. We must have a rule $f(q_1, \ldots, q_m) \to q$. Then for each $1 \leq i \leq m$ we can define a run r_i of $\mathcal{A}$ on t_{pi} simply by setting $r_i(w) = r(iw)$. By the induction hypothesis, we have $q_i = r_i(\lambda) \in r'(pi)$. It follows that $q \in r'(p)$, completing our proof of the statement.

To complete the proof, suppose that $t \in L(\mathcal{A}')$. Then the unique run r' of $\mathcal{A}'$ on t is such that $r'(\lambda)$ contains some $q \in F$. But by above, $r'(\lambda) = \{r(\lambda) \mid r \text{ is a run of } \mathcal{A} \text{ on } t_\lambda\}$, so there must be some run r of $\mathcal{A}$ on $t_\lambda = t$ with $r(\lambda) = q$, which means $t \in L(\mathcal{A})$. For the other direction, suppose $t \in L(\mathcal{A})$. Then there is a run r of $\mathcal{A}$ on t such that $r(\lambda) \in F$. By the previous paragraph, $r(\lambda) \in r'(\lambda)$, so t is accepted by $\mathcal{A}'$. □

3.4 Pumping Lemma for Tree Languages

In this section, we show how the pumping lemma for finite words can be generalized to finite trees. For the statement of the result, we will require some additional terminology and notation, which we introduce now.

Let Σ be a ranked alphabet, and $\mathcal{X}$ a set of symbols (called *variables*) which is disjoint from Σ. A $\Sigma, \mathcal{X}$-*tree* is defined just like a Σ-tree except that some of the frontier positions may be labelled with symbols from $\mathcal{X}$. A $\Sigma, \mathcal{X}$-tree is called a *context* if there is at most one occurrence of each symbol from $\mathcal{X}$; it is a *trivial context* if it consists of a single position labelled by a variable. If C is a context with n variables $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $t^1, \ldots, t^n$ are $\Sigma, \mathcal{X}$-trees, then $C[t^1, \ldots, t^n]$ is the $\Sigma, \mathcal{X}$-tree obtained by replacing x_i with the tree t^i. We define C^n inductively as follows: C^0 is the trivial context, and $C^{n+1} = C^n[C]$.

We are now ready to state the tree analogue of the pumping lemma.

Theorem 90. *Let L be a recognizable tree language over the ranked alphabet Σ. Then there exists a constant $k > 0$ satisfying: for every tree $t \in L$ with*

height greater than k, there exist a (one-variable) context C, a non trivial (one-variable) context D, and a Σ-tree v such that $t = C[D[v]]$ and for all $n \geq 0$, $C[D^n[v]] \in L$.

Proof. Let L be a recognizable tree language, and let $\mathcal{A} = (Q, \Sigma, T, F)$ be a NFTA such that $L = L(\mathcal{A})$. We set k equal to the number of states in Q. Consider some tree $t \in L$ with height greater than k, and let r be a successful run of $\mathcal{A}$ on t. As t (hence r) has height greater than k, there must be some path from the root to a frontier position which contains twice the same state. So let p_1 and p_2 be positions such that $p_2 = p_1 p_3$ (for $p_3 \neq \lambda$) and $r(p_1) = r(p_2) = q$. We let u be the tree t_{p_1}, and we let C be some context such that $t = C[u]$. Likewise, we let v be the tree t_{p_2}, and D be a non-trivial context such that $u = D[v]$. Then we have $t = C[D[v]]$.

Now consider the tree $t^0 = C[D^0[v]] = C[v]$. Define r_0 as follows: $r_0(p) = r(p)$ if $p_1 \not\sqsubseteq p$, and $r_0(p) = r(p_2 p')$ if $p = p_1 p'$. It is easily verified that r_0 is a successful run of $\mathcal{A}$ on t^0, which means that $t^0 \in L$. We now consider the case of $t^n = C[D^n[v]]$ where $n \geq 1$. We define r_n in the following manner: $r_n(p) = r(p)$ if $p_1 \not\sqsubseteq p$, $r_n(p) = r(p_1 p')$ if $p = p_1 p_3^j p'$, where $p_3 \not\sqsubseteq p'$ and $p_1 p_3^n \not\sqsubseteq p$, and $r_n(p) = r(p_2 p')$ if $p = p_1 p_3^n p'$. Again, it can be checked that r_n indeed defines a successful run of $\mathcal{A}$ on t^n, which implies $t^n \in L$. □

Corollary 91. *Let $\mathcal{A} = (Q, \Sigma, T, F)$ be a NFTA. Then $L(\mathcal{A})$ is non-empty if and only if there exists a tree t in $L(\mathcal{A})$ with height at most $|Q|$.*

3.5 Closure Properties for Tree Languages

In this section, we investigate the closure properties of the set of recognizable tree languages.

Theorem 92. *The class of recognizable tree languages is closed under union.*

Proof. Let L_1, L_2 be a recognizable tree languages, and let $\mathcal{A}_1 = (Q_1, \Sigma, T_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, T_2, F_2)$ be such that $L_1 = L(\mathcal{A}_1)$ and $L_2 = L(\mathcal{A}_2)$. Then the desired automaton $\mathcal{A} = (Q, \Sigma, T, F)$ is as follows:

- $Q = Q_1 \cup Q_2$
- $T = T_1 \cup T_2$
- $F = F_1 \cup F_2$

It is easily verified that this automaton accepts the language $L_1 \cup L_2$. □

Theorem 93. *The class of recognizable tree languages is closed under complementation.*

Proof. Let L be a recognizable tree language, and let $\mathcal{A} = (Q, \Sigma, T, F)$ be a DFTA with $L = L(\mathcal{A})$. Then the DFTA $\mathcal{A}' = (Q, \Sigma, T, Q \setminus F)$ accepts precisely those trees which are rejected by $\mathcal{A}$, so $L(\mathcal{A}')$ is the complement of L. □

Theorem 94. *The class of recognizable tree languages is closer under intersection.*

Proof. Follows directly from closure under union and complementation. □

3.6 Top-down Tree Automata

The tree automata that we have investigated so far are called bottom-up tree automata since their runs correspond to assigning states to the leaf nodes and then continuing up the tree. In this section, we introduce top-down tree automata, which start by assigning a state to the root node, and then work downwards to the leaf nodes.

Definition 95. A *nondeterministic top-down automaton on finite ordered ranked trees* (or NTDTA for short) is a quadruple $\mathcal{A} = (Q, \Sigma, T, I)$ where:

- Q is a non-empty *set of states*
- Σ is a *ranked alphabet*
- T is a set of *transition rules* of the form:

  $$(f, q) \to (q_1, \ldots, q_m)$$

  where $m \geq 0$, $f \in \Sigma_m$, $q, q_1, \ldots, q_m \in Q$.
- $I \subseteq Q$ is a set of *initial states*

If I contains a single state and there do not exist two distinct rules in T with the same left hand side, then the automaton is said to be *deterministic*.

We give an informal description of the behaviour of top-down tree automata. The automaton starts by assigning an initial state to the root node of the tree. Then it selects a rule which matches the state of the root node in order to define the states of its children. It continues down the tree in this manner until the tree is fully labelled, or there are no transitions which match one of the already labelled nodes.

Definition 96. A *run* of a NTDTA $\mathcal{A} = (Q, \Sigma, T, I)$ on a Σ-tree t is a function $r : \mathsf{Pos}(t) \to Q$ which satisfies the following conditions:

- $r(\lambda) \in I$
- if $t(p) = f \in \Sigma_m$ $(m \geq 1)$, $r(p) = q$, and $r(p1) = q_1, \ldots, r(pm) = q_m$, then there is a rule $(f, q) \to (q_1, \ldots, q_m)$ in T

Definition 97. A tree t is said to be *accepted* by a NTDTA $\mathcal{A}$ if there exists a run r of $\mathcal{A}$ on t.

Thus, the only requirement for a tree to be accepted by a NTDTA is that there exists at least one run of the automaton on the tree. No special conditions are placed on the type of runs that a tree has.

We now show that nondeterministic top-down tree automata recognize the same languages as NFTAs.

Theorem 98. *The set of languages which are recognized by nondeterministic top-down tree automata is precisely the set of recognizable tree languages.*

Proof. Suppose $L = L(\mathcal{A})$ for some bottom-up finite tree automaton $\mathcal{A} = (Q.\Sigma, T, F)$. Consider the top-down automaton $\mathcal{A}' = (Q, \Sigma, T', F)$ where

$$(f, q) \to (q_1, \ldots, q_m) \in T' \quad \text{iff} \quad f(q_1, \ldots, q_m) \to q \in T$$

It should be clear that the successful runs of $\mathcal{A}$ on a tree t are precisely the runs of $\mathcal{A}'$ on t. So $L(\mathcal{A}') = L$. Thus, every recognizable tree language is recognized by some NTDTA.

Conversely, let $L = L(\mathcal{A})$ for some top-down automaton $\mathcal{A} = (Q.\Sigma, T, I)$. Consider the bottom-up automaton $\mathcal{A}' = (Q, \Sigma, T', I)$ where

$$f(q_1, \ldots, q_m) \to q \in T' \quad \text{iff} \quad (f, q) \to (q_1, \ldots, q_m) \in T$$

Again, we have that the runs of $\mathcal{A}$ on an input tree t correspond precisely to the successful runs of $\mathcal{A}'$ on t, so $L(\mathcal{A}') = L$. It follows that every language accepted by a NTDTA is a recognizable tree language. □

We saw earlier that nondeterministic and deterministic bottom-up tree automata are equivalent in expressive power. The same is not true for top-down tree automata, as the following theorem attests.

Theorem 99. *There exist recognizable tree languages which are recognized by no deterministic top-down tree automaton.*

Proof. Let $\Sigma = \{f/2, a/0, b/0\}$, and consider the recognizable tree language $L = \{t_1, t_2\}$, where $\mathsf{Pos}(t_1) = \mathsf{Pos}(t_2) = \{\lambda, 1, 2\}$, $t_1(\lambda) = t_2(\lambda) = f$, $t_1(1) = t_2(2) = a$, and $t_1(2) = t_2(1) = b$. Now let us suppose there exists a deterministic top-down tree automaton $\mathcal{A} = (Q, \Sigma, T, I)$ that accepts L. Then there is a single initial state q_I and a single rule $(f, q_I) \to (q_1, q_2)$ with (f, q_I) on the left hand side. As $t_1 \in L$ and $t_2 \in L$, the rules $(a, q_1) \to ()$, $(b, q_2) \to ()$, $(b, q_1) \to ()$, and $(a, q_2) \to ()$ must all belong to T. But then the tree t_3 with $\mathsf{Pos}(t_3)$, $t_3(\lambda) = f$, and $t_3(1) = t_3(2) = a$ will also be accepted by $\mathcal{A}$, which is a contradiction. □

3.7 Decision Problems

We consider the usual decision problems: emptiness, equivalence, and universality.

Theorem 100. *The emptiness problem for NFTAs is decidable in polynomial time.*

Proof. The basic idea is to compute the set of reachable states, and then to check whether some final state is reachable. More specifically, we start by adding to the set of reachable states every state q such that there is a rule $a \to q$. Then if $q_1, \ldots, q_n$ is reachable and there is a rule $f(q_1, \ldots, q_n) \to q$, we add q to the set of reachable states. Clearly, this computation of reachable states must terminate in polynomial time since at each stage we add a new state from Q, and so there can be at most $|Q|$ stages in the process. It remains to be shown that the language is non-empty if and only if some final state belongs to the set of reachable states we compute.

First suppose that $L(\mathcal{A})$ is non-empty, and let t be a tree in L and r a run of $\mathcal{A}$ on t. Then clearly each of the states in r, including the final state at the root, will belong to the set of reachable states. Conversely, if a final state q belongs to the set of reachable states, then we can construct a run which has q at its root. We place q at the root position and find the (unique) rule $f(q_1, \ldots, q_n) \to q$ which was used to add q to the set of reachable states. We set $t(\lambda) = f$, and then we iterate the process with states $q_1, \ldots, q_n$ in order to find the symbols at positions $1, \ldots, m$ of t, and so on. This process must terminate with a tree (and run) of height at most $|Q|$ since by construction a state cannot appear twice on a path in the run we construct. Thus, the constructed run r will be a successful run of the tree t, so $L(\mathcal{A})$ is non-empty. □

Theorem 101. *The equivalence problem for NFTAs is decidable.*

Proof. If we want to test that $\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent, we use closure under union, intersection, and complementation to build an automaton which recognizes $(L(\mathcal{A}_1) \cap \overline{L(\mathcal{A}_2)}) \cup (L(\mathcal{A}_2) \cap \overline{L(\mathcal{A}_1)})$. We then use Theorem 100 to decide emptiness for this automaton. □

Theorem 102. *The univerality problem for NFTAs is decidable.*

Proof. To test whether $\mathcal{A}$ accepts all Σ-trees, we use closure under complementation to build an automaton which accepts $\overline{L(\mathcal{A})}$, and we perform the emptiness test on this automaton. □

Note however that while the emptiness problem is decidable in polynomial time, this is not the case for the equivalence and universality problems. Indeed, these problems can be shown to be EXPTIME-complete, which means that there cannot exist any polynomial algorithm which solves them. The reason essentially is that we need to perform complementation, and complementation may lead to an exponential blow-up in automata size.

Chapter 4

Automata on Infinite Trees

4.1 Basic Definitions and Notations

We will be working with the infinite binary tree, i.e. the tree in which every node has exactly two children. Formally, we define the infinite binary tree, denoted $\mathcal{T}$ as the set $\{0,1\}^\star$. Each of the elements of $\{0,1\}^\star$ corresponds to node: the empty string λ corresponds to the root of the tree, a string of the form $w0$ corresponds to the left child of the node w, and a string of the form $w1$ corresponds to the right child of the node w.

The *level* of a node w is simply the length of the word w. A node u is called a *predecessor* of a node v, written $u \sqsubseteq v$, if there exists a string w such that $u = vw$, or in other words, if u is a prefix of v.

Definition 103. A subset π of $\mathcal{T}$ is a *path* if the following conditions hold:

1. The root λ belongs to π
2. If $u \in \pi$, then exactly one of $u0$ and $u1$ is in π

It follows from the above definition, that for every pair of nodes u, v in a path π, we must either have $u \sqsubseteq v$ or $v \sqsubseteq u$.

Definition 104. Let Σ be an alphabet[1]. Then a Σ-*tree* τ is a function from $\mathcal{T}$ to Σ.

Thus, a Σ-tree is a labelling of the nodes in $\mathcal{T}$ by the symbols in Σ. Given a Σ-tree τ and a node u, we denote by τ_u the Σ-tree defined by $\tau_u(v) = \tau(uv)$. The Σ-tree τ_u corresponds to the subtree of τ rooted at the

[1]Here we will again be working with standard, i.e. unranked, alphabets.

node u. Given a Σ-tree τ and a path π, we denote by $\tau|\pi$ the infinite string of symbols from Σ visited along the path π. Given two Σ-trees τ and ρ, we denote by $\tau[u \to \rho]$ the Σ-tree which assigns the value $\tau(v)$ to every v with $u \not\sqsubseteq v$, and the value $\rho(w)$ to every node $v = uw$. Thus, $\tau[u \to \rho]$ is the Σ-tree obtained by replacing the subtree rooted at u by the tree ρ.

In this chapter, by *language* we will mean a subset of the set of Σ-trees.

4.2 Definition of Automata on Infinite Trees

In this section, we show how different types of automata on infinite words can be extended to automata on infinite trees. Note that all of the automata we consider in this chapter will be non-deterministic.

Definition 105. A *Büchi (tree) automaton* is a quintuple $\mathcal{A} = (Q, \Sigma, T, I, F)$ where:

- Q is a finite set of states
- Σ is an alphabet
- T is a subset of $Q \times \Sigma \times Q \times Q$ called the *transition table*
- $I \subseteq Q$ is the set of initial states
- $F \subseteq Q$ is a set of final states

To define Müller tree automata, we simply replace the set of final states F by a set $\mathcal{F} \subseteq 2^Q$ of distinguished subsets.

Definition 106. A *Müller (tree) automaton* is a quintuple $\mathcal{A} = (Q, \Sigma, T, I, \mathcal{F})$ where Q, Σ, T, I are as above and $\mathcal{F} \subseteq 2^Q$ is a set of distinguished subsets of Q.

For parity tree automata[2], we use instead a function $c : Q \to \mathbb{N}$.

Definition 107. A *parity (tree) automaton* is a quintuple $\mathcal{A} = (Q, \Sigma, T, I, c)$ where Q, Σ, T, I are as above and c is a function from Q to $\mathbb{N}$.

We will not consider Rabin and Street tree automata in this chapter, but obviously they can be defined in a similar manner by using a set of accepting pairs or a set of fairness conditions as the last component of the automaton.

[2]Parity word automata were introduced in problem set 3.

All of the above automata take (infinite) Σ-trees as input. They will work in a top-down manner, assigning first a state from I to the root, then assigning states to the other nodes in $\mathcal{T}$ according to the transition table T. Formally, runs of tree automata on Σ-trees are defined as follows:

Definition 108. A *run* of a Büchi (or Müller or parity) automaton $\mathcal{A}$ on a Σ-tree τ is a mapping $\mathbf{r} : \mathcal{T} \to Q$ such that (i) $\mathbf{r}(\lambda) \in I$ and (ii) for all $u \in \mathcal{T}$, we have $(\mathbf{r}(u), \tau(u), \mathbf{r}(u0), \mathbf{r}(u1)) \in T$.

Thus, a run of an automaton on a Σ-tree is simply a labelling of the nodes in the tree by states from Q which satisfies the automaton's initial states and transition table.

We now define acceptance for our three different types of infinite tree automata.

Definition 109. A run $\mathbf{r}$ of an automaton $\mathcal{A}$ on a Σ-tree τ is said to be *successful* if one of the following holds:

- $\mathcal{A}$ is a Büchi automaton with final state set F, and for every path π of $\mathcal{T}$, we have

$$\{q \in Q \mid \mathbf{r}(u) = q \text{ for infinitely many } u \in \pi\} \cap F \neq \emptyset$$

- $\mathcal{A}$ is a Müller automaton with $\mathcal{F}$ as its set of distinguished subsets, and for every path π of $\mathcal{T}$, we have

$$\{q \in Q \mid \mathbf{r}(u) = q \text{ for infinitely many } u \in \pi\} \in \mathcal{F}$$

- $\mathcal{A}$ is a parity automaton with coloring function c, and for every path π of $\mathcal{T}$, we have

$$\mathsf{min}\{c(q) \mid \mathbf{r}(u) = q \text{ for infinitely many } u \in \pi\} \text{ is even}$$

A Σ-tree τ is *accepted* by an automaton $\mathcal{A}$ if there is a successful run of $\mathcal{A}$ on τ.

Thus, in order to decide whether a run is successful, we consider each of the restrictions of the run to a single path of the tree, and we check whether each of these infinite sequences of states satisfies the acceptance condition of the automaton. So for instance if we are working with a Büchi tree automaton, then we require that every infinite path has infinitely many nodes labelled by a final state.

Definition 110. The *language* of an automaton $\mathcal{A}$ with alphabet Σ, denoted $L(\mathcal{A})$, is the set of Σ-trees which are accepted by $\mathcal{A}$. A language is said to be Büchi- (resp. Müller-, parity-) *recognizable* if it is accepted by some Büchi (resp. Müller, parity) automaton.

We illustrate the above definitions with some examples.

Example 111. Consider the Büchi automaton $\mathcal{A} = (Q, \Sigma, T, I, F)$ where $Q = \{q_a, q_b\}$, $\Sigma = \{a, b\}$, $T = \{(q_a, a, q_a, q_a), (q_b, a, q_a, q_a), (q_a, b, q_b, q_b), (q_b, b, q_b, q_b)\}$, $I = \{q_a\}$, and $F = \{q_a\}$. We remark that the transition table T assigns to a node the state q_a whenever the node's parent is labelled by the symbol a, and it assigns the state q_b to nodes with parents labelled by b. It follows that if a tree has an accepting run, then each of its branches must contain an infinite number of a's. Likewise, if a tree has infinitely many a's on each of its branches, then the unique run of $\mathcal{A}$ on the tree will be accepting. Thus, $L(\mathcal{A})$ is the set of all Σ-trees having infinitely many a's on every branch.

Example 112. Consider the Büchi automaton $\mathcal{A} = (Q, \Sigma, T, I, F)$ where $Q = \{q_a, q_b, q_c\}$, $\Sigma = \{a, b\}$, $T = \{(q_a, a, q_a, q_c), (q_a, a, q_c, q_a), (q_b, a, q_a, q_c), (q_b, a, q_c, q_a), (q_a, b, q_b, q_c), (q_a, b, q_c, q_b), (q_b, b, q_b, q_c), (q_b, b, q_c, q_b), (q_c, a, q_c, q_c), (q_c, b, q_c, q_c)\}$, $I = \{q_a\}$, and $F = \{q_a, q_c\}$. We remark that when we are in a state q_a or q_b and we read a symbol $x \in \{a, b\}$, then one of the children is assigned the state q_c and the other is assigned q_x. Since we must start with the state q_a at the root, it follows that every run must contain a single branch with only q_a and q_b, and all other branches will contain finitely many q_a and q_b states, and hence infinitely many q_c states. The latter branches will all satisfy the acceptance condition, and the "chosen" branch which contains only q_a and q_b will satisfy the acceptance condition if and only if it has infinitely many q_a states. It follows that the language accepted by $\mathcal{A}$ is the set of all Σ-trees which have at least one branch with infinitely many a's.

Example 113. Consider the Müller automaton $\mathcal{A} = (Q, \Sigma, T, I, \mathcal{F})$ where $Q = \{q_a, q_b\}$, $\Sigma = \{a, b\}$, $T = \{(q_a, a, q_a, q_a), (q_b, a, q_a, q_a), (q_a, b, q_b, q_b), (q_b, b, q_b, q_b)\}$, $I = \{q_a\}$, and $\mathcal{F} = \{\{q_b\}\}$. Notice that as in Example 111, T assigns to children of a node labelled a the state q_a and to children of a node b the state q_b. Moreover, accepting runs have finitely many a's on every branch. It follows that the language of $\mathcal{A}$ is the set of Σ-trees in which every branch contains only finitely many a's.

Example 114. Consider the parity automaton $\mathcal{A} = (Q, \Sigma, T, I, c)$ where $Q = \{q_a, q_b\}$, $\Sigma = \{a, b\}$, $T = \{(q_a, a, q_a, q_a), (q_b, a, q_a, q_a), (q_a, b, q_b, q_b),$

$(q_b, b, q_b, q_b)\}$, $I = \{q_a\}$, $c(q_a) = 1$ and $c(q_b) = 2$. Then a branch of a run satisfies the acceptance condition if the minimum value of the infinity set of the branch is even, i.e. the infinity set is equal to $\{q_2\}$. It follows that $\mathcal{A}$ recognizes the same language as the Müller automaton in Example 113, which is the set of all Σ-trees having finitely many a's per branch.

4.3 Relation between Büchi and Müller Tree Automata

For infinite words, we proved in Chapter 2 that Büchi and Müller automata recognize the same ω-languages. In this section, we show that for infinite trees, this is no longer the case: Büchi tree automata are strictly weaker than Müller tree automata.

We first show that Müller tree automata can recognize all Büchi-recognizable languages.

Theorem 115. *Every Büchi-recognizable language is Müller-recognizable.*

Proof. The proof is essentially the same as for infinite words. Given a Büchi-automata $\mathcal{A} = (Q, \Sigma, T, I, F)$, we construct the Müller automaton $\mathcal{A}' = (Q, \Sigma, T, I, \mathcal{F})$ where $\mathcal{F} = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$. It can be easily verified that $L(\mathcal{A}) = L(\mathcal{A}')$. □

We now show that there are tree languages which are Müller-recognizable but not Büchi-recognizable.

Theorem 116. *There exist Müller-recognizable languages which are not Büchi-recognizable.*

Proof. Let $\Sigma = \{a, b\}$, and let L be the language of Σ-trees in which every branch contains only finitely many a's. We know from example 113 that L is Müller-recognizable, and we aim to show that L is not Büchi-recognizable.

Suppose then for a contradiction that there is a Büchi tree automaton $\mathcal{A} = (Q, \Sigma, T, I, F)$ which recognizes L. Let n be the number of states in Q. Now define the sets of nodes U_i (for $i \geq 0$) as follows: $U_i = \cup_{k=0}^{i} (1^+0)^k$. Define for each $i \geq 0$ the Σ-tree τ_i so that $\tau_i(w) = a$ for $w \in U_i$ and $\tau_i = b$ otherwise. By construction, we have the following properties:

1. If $\tau_i(w) = a$, then $\tau_{i+1}(w) = a$ and $\tau_{i+1}(1^k 0w) = a$ for every $k \geq 1$.

2. If $\tau_{i+1}(w) = a$ and $w = 1^k 0v$ for some $k \geq 1$, then $\tau_i(v) = a$.

We also remark that each tree τ_i belongs to L since each branch of τ_i has only finitely many a's. In particular, this means that there must be some accepting run $\mathbf{r}$ of the Büchi automaton $\mathcal{A}$ on τ_n.

We now use the run $\mathbf{r}$ to construct a sequence of indices $m_1, \ldots, m_n$. We let $m_1 > 0$ be such that $\mathbf{r}(1^{m_1}) = f_1 \in F$ and for every $0 < k < m_1$ we have $\mathbf{r}(1^k) \notin F$. Note that we must be able to find such a m_1 since $\mathbf{r}$ is an accepting run, and so every path in $\mathbf{r}$ contains infinitely many final states. If we have already defined $m_1, \ldots, m_i$, and $i < n$, then we define m_{i+1} so that $\mathbf{r}(1^{m_1}01^{m_2}0\ldots1^{m_i}01^{m_{i+1}}) = f_{i+1} \in F$ and no smaller value has this property.

Now because $|Q| < n$, it follows that there exist $j < k$ such that $f_j = f_k$. Let $u = 1^{m_1}01^{m_2}0\ldots1^{m_j}$ and $v = 01^{m_{j+1}}01^{m_{j+2}}0\ldots1^{m_k}$. Then we have $\mathbf{r}(u) = \mathbf{r}(uv) = f_j = f_k \in F$.

We now replace the subtree rooted at node uv by the tree τ_{n_u}, i.e. we replace τ_n by $\tau_n[uv \to \tau_{n_u}]$. It is not hard to show that the resulting tree still has an accepting run, namely $\mathbf{r}[uv \to \mathbf{r}_u]$. We can apply this procedure a second time in the subtree rooted at uv, again yielding an accepting run. If we then iterate this procedure an infinite number of times, then the resulting tree will still have an accepting run. However, this tree will contain a path with an infinite number of a's since by construction there must be at least one a between u and uv (and hence between uv and uvv, uvv and $uvvv$, ...). This means that the automaton $\mathcal{A}$ accepts a tree which contains a path with infinitely many a's, which contradicts our assumption that $L(\mathcal{A}) = L$. □

Note that the proof of Theorem 116 also shows that the set of Büchi-recognizable tree languages is not closed under complement. This is because the language L from the proof is the complement of the language from Example 112, which we showed to be Büchi-recognizable.

It is also interesting to note that the language we used to show that Müller tree automata are more powerful than Büchi tree automata is the tree version of the language that we used to show nondeterministic Büchi are strictly stronger than deterministic Büchi. The following result, which we will not prove here, states that all tree languages which are Müller-recognizable but not Büchi-recognizable can be obtained from some Büchi-recognizable language of infinite words which is not recognized by any deterministic Büchi automaton. In the statement of the theorem, we use L_Δ to denote the set of trees τ such that every path of τ belongs to the infinite word language L.

Theorem 117. *Every Σ-tree language which is Müller-recognizable but not Büchi-recognizable is of the form L_Δ for some infinite word language L*

over Σ which is Büchi-recognizable but not Müller-recognizable. Conversely, if L is a Büchi-recognizable language of infinite words which is not accepted by any deterministic Büchi automaton, then L_Δ is Müller-recognizable but not Büchi-recognizable.

4.4 Relation between Müller and Parity Tree Automata

In this section, we show that Müller and parity automata have the same expressive power. We first show the easier direction, which is that parity-recognizable languages can also be recognized by Müller automata.

Theorem 118. *Every parity-recognizable language is Müller-recognizable.*

Proof. Given a parity tree automaton $\mathcal{A} = (Q, \Sigma, T, I, c)$, we construct a Müller automaton $\mathcal{A}' = (Q, \Sigma, T, I, \mathcal{F})$, by setting

$$\mathcal{F} = \{S \subseteq Q \mid \mathsf{min}\{c(q) \mid q \in S\} \text{ is even}\}$$

It is easily verified that $\mathcal{A}'$ and $\mathcal{A}$ accept the same tree language. □

The other direction of the equivalence (Müller-recognizable ⇒ parity-recognizable) is more interesting.

Theorem 119. *Every Müller-recognizable language is parity-recognizable.*

Proof. Consider some Müller tree automaton $\mathcal{A} = (Q, \Sigma, T, I, \mathcal{F})$. For simplicity, we suppose that the states of Q are $\{1, \ldots, n\}$. For the state set of the parity tree automaton, we will use all pairs $((q_1 \ldots q_n), p)$ where $q_1 \ldots q_n$ is a permutation of $1 \ldots n$ (i.e. contains exactly once each state in Q) and $p \in \{1, \ldots, n\}$. The idea is that $q_1 \ldots q_n$ represents the order of the last visits of the states on the run of the Müller automaton on some path: q_n is the most recently visited state on the path, then the state q_{n-1}, and so on. We use p to indicate the previous position of the most recently visited state s_q. For example, if we are at a state $((2\,1\,3), 1)$ in the parity automaton and the next visited state of the Müller automaton on the path is 1, we transition to $((2\,3\,1), 2)$, since 1 is now the most recently visited state of Q, 3 is the second most recently visited, state 2 is the least recently visted, and the previous position of the state 1 in the sequence was 2. If the next state visited by the Müller automaton on the path is 2, then we would get the state $((3\,1\,2), 1)$, since 2 is the most recently visited state (then state 1, and then state 3), and 2 was previously at position 1.

Before describing the rest of the parity automaton, we prove some properties of its set of states. Let us suppose that $q_1q_2q_3\ldots$ is a sequence of states in Q, and let $s_1s_2s_3\ldots$ be the corresponding sequence of states in the parity automaton starting from some state $((t_1\ldots t_{n-1}q_1),1)$ where $t_1\ldots t_{n-1}$ is any permutation of $\{1,\ldots,n\}\setminus\{q_1\}$. Then we claim the following:

$\mathsf{Inf}(q_1q_2q_3\ldots)=S$ (where $|S|=k$) iff the sequence $s_1s_2s_3\ldots$ satisfies:

1. the position p_i of state s_i is less than $n-k+1$ for finitely many i
2. there are infinitely many i such that $p_i=n-k+1$ and the set of states appearing in positions $n-k+1$ to n is precisely S

To prove the first direction of the claim, let l_1 be such that $q_i\in S$ for all $i\geq l_1$. Then let $l_2>l_1$ be such that every state in S appears at some position between l_1 and l_2. This means that starting from timepoint l_1, only states from S are moved to the final position in the sequence. Moreover, by timepoint l_2, we must have moved all of the states in S to the final position at least once, and so the last k positions of the sequence in state s_{l_2} must contain precisely the states in S (since the other states are pushed further and further to the left). In future states s_i, we never move a state outside of S (which are now in positions 1 to $n-k$) to the final position, so p_i must always be at least $n-k+1$, and the last k positions must remain in S. We also remark that if there are only finitely many states with position p_i equal to $n-k+1$, then there must be some state in S which is no longer visited, contradicting the fact that $\mathsf{Inf}(q_1q_2q_3\ldots)=S$. Thus, both parts of the claim must hold. The second direction of the claim is shown using a similar analysis of the construction.

Now we construct our new parity automaton and use the above claim to prove its correctness. Our parity automaton $\mathcal{A}'$ will have the state set described above. For the initial set of states I', we can use any state $((t_1\ldots t_{n-1}q),1)$ where $q\in I$ and $t_1\ldots t_{n-1}$ is any permutation of $\{1,\ldots,n\}\setminus\{q\}$. Our transition table T' is composed of all tuples

$$(((i_1\ldots i_{n-1}i),p),a,((i'_1\ldots i'_{n-1}i'),p'),((i''_1\ldots i''_{n-1}i''),p''))$$

such that (i,a,i',i'') belongs to the transition table of $\mathcal{A}$, the state sequence $i'_1\ldots i'_{n-1}$ (resp. $i''_1\ldots i''_{n-1}$) is obtained by removing i' (resp. i'') from $i_1\ldots i_{n-1}i$, and p' (resp. p'') denotes the position of i' (resp. i'') in $i_1\ldots i_{n-1}i$. Finally, for our acceptance condition, we define the function c as follows:

- if $s = ((q_1 \dots q_n), p)$ and $\{q_p \dots q_n\} \in \mathcal{F}$, then $c(s) = 2p$
- if $s = ((q_1 \dots q_n), p)$ and $\{q_p \dots q_n\} \notin \mathcal{F}$, then $c(s) = 2p + 1$

In order to show correctness of the construction, suppose first that $\tau \in L(\mathcal{A})$. Then we have an infinite run $\mathbf{r} = q_1 q_2 q_3 \dots$ of $\mathcal{A}$ on τ such that the infinity set of each branch of $\mathbf{r}$ belongs to $\mathcal{F}$. Choose some $((t_1 \dots t_{n-1} q_1), 1) \in I'$, and let $\mathbf{s}$ be the unique run of $\mathcal{A}'$ on τ starting from $((t_1 \dots t_{n-1} q_1), 1)$ such that the last component of the sequence in $\mathbf{s}(w)$ equals $\mathbf{r}(w)$. In other words, we use the transitions from the run $\mathbf{r}$ to determine our transitions in $\mathbf{s}$. Now consider some path π. We know that $\mathsf{Inf}(\mathbf{r}|\pi) = S$ for some $S \in \mathcal{F}$ since $\mathbf{r}$ is a successful run. It follows from the above claim that $\mathbf{s}|\pi$ is such that (i) the position p_i of state s_i is less than $n - |S| + 1$ for finitely many i, and (ii) there are infinitely many i such that $p_i = n - |S| + 1$ and the set of states appearing in positions $n - |S| + 1$ to n belongs to S. But it follows then that $\mathsf{Inf}(\mathbf{s}|\pi)$ contains some element of the form $((q_1 \dots q_n), n - |S| + 1)$ where $\{q_{n-|S|+1}, \dots, q_n\} = S$, and $\mathsf{Inf}(\mathbf{s}|\pi)$ contains no elements of the form $((q_1 \dots q_n), p)$ for $p \leq n - |S|$. It follows that the path $\mathbf{s}|\pi$ satisfies the parity acceptance condition. Since the same holds for all paths, we get that $\tau \in L(\mathcal{A}')$.

For the second direction, suppose that $\tau \in L(\mathcal{A}')$. That means that there is an accepting run $\mathbf{s}$ of $\mathcal{A}'$ on τ. Define $\mathbf{r}$ such that $\mathbf{r}(w) = q$ if q is the final state in the sequence of states in $\mathbf{s}(w)$. By definition, $\mathbf{r}(\lambda) \in I$, and for every position w, we must have $(\mathbf{r}(w), \tau(w), \mathbf{r}(w0), \mathbf{r}(w1)) \in T$. It follows that $\mathbf{r}$ is a run of $\mathcal{A}$ on τ. Now because $\mathbf{s}$ is accepting, it follows that for each path π, there is some state $((q_1 \dots q_n), p)$ such that $\{q_p \dots q_n\} \in \mathcal{F}$ which appears infinitely often on $\mathbf{s}|\pi$, and that no state with last component less than p appears infinitely often. But according to the above claim, and the way we have defined the run $\mathbf{r}$, it must be the case that $\mathsf{Inf}(\mathbf{r}|\pi) = \{q_p \dots q_n\} \in \mathcal{F}$. Thus, all paths of $\mathbf{r}$ satisfy the Müller acceptance condition, so $\mathbf{r}$ is an accepting run, and $w \in L(\mathcal{A})$. □

4.5 Complementation: Reduction to Parity Games

The class of languages recognized by Müller (or equivalently parity) automata can be shown to satisfy the standard closure properties: union, intersection, complementation, and projection. In this section, we will focus on complementation since the other closure properties are proved similarly to the case for infinite word automata. On the other hand, closure under complementation is a much more difficult and interesting result.

The idea behind our proof will be to reduce the complementation problem to a known result concerning strategies in certain types of (abstract) games. We associate to a tree automaton $\mathcal{A}$ (with alphabet Σ) and a Σ-tree τ an infinite two-person game denoted $G_{\mathcal{A},\tau}$. We will assume without loss of generality that $\mathcal{A}$ has a single initial state q_I. The two players in this game are called "Automaton" and "Pathfinder". The players alternate turns, and after each round of turns, they move to a subsequent position in the tree τ, starting from the root node. The game begins with Automaton who must select a transition from T which can be applied at the root node. Then the player Pathfinder chooses either the right or left child of the node, and moves the marker to this node. It is then Automaton's turn again to choose a new transition from T which must agree with the label of the current node of the tree and the state associated with this node. Pathfinder then chooses again whether to move to the marker to the left child or right child. The game continues indefinitely in this manner and results in the construction of an infinite sequence of states from Q. We say that Automaton wins the game if this sequence of states satisfies the acceptance condition of $\mathcal{A}$, and otherwise it is Pathfinder who wins. Thus, the player Automaton wants to choose transitions which will satisfy the acceptance condition, while Pathfinder aims to select a path for which Automaton will not be able to fulfill the acceptance condition.

Formally, the positions of the game $G_{\mathcal{A},\tau}$ are of two types, those for Automaton, and those for Pathfinder. Automaton's game positions are of the form (w, q) where w is a node and q is a state from Q. From the position (w, q), the possible moves of Automaton are precisely the transitions from T of the form $(q, \tau(w), q_1, q_2)$. If Automaton selects a transition $(q, \tau(w), q_1, q_2)$, then this yields the position $(w, (q, \tau(w), q_1, q_2))$ for Pathfinder. Now Pathfinder may select either a move to the left child of w, yielding the position $(w0, q_1)$, or a move to the right child node, which gives the game position $(w1, q_2)$. The usual starting position for the game is (λ, q_I).

A natural way of thinking of the game $G_{\mathcal{A},\tau}$ is as an infinite graph, in which the vertices are game positions, and there is an edge from one game position to another if there is a move which is possible in the first position and leads to the second position. Then each possible infinite sequence of the players' moves in the game corresponds to an infinite path in this game graph. We will call a game position p' *reachable* from a game position p if there exists a finite sequence of moves which leads from p to p'.

A *strategy* from position p for the player Automaton (resp. Pathfinder) is a function mapping each sequence of moves which begin in p and end in some

game position p' of Automaton (resp. Pathfinder) to a move which is possible in the position p'. Thus, a strategy for p tells the player exactly which moves he should select at each possible game position which is reachable from p. Strategies which depend only on the current position p', and not on the moves which lead to this position, are called *memoryless*. A *winning strategy* for a player is a strategy which guarantees that the player will win if he follows the strategy *no matter what moves the other player makes*.

We now show how acceptance of a tree by an automaton can be rephrased in terms of winning strategies.

Lemma 120. *The tree automaton $\mathcal{A}$ over alphabet Σ accepts a Σ-tree τ if and only if there is a winning strategy for the player Automaton in the game $G_{\mathcal{A},\tau}$ starting from the position (λ, q_I).*

Proof. Suppose that $\mathcal{A}$ accepts τ, and consider some successful run $\mathbf{r}$ of $\mathcal{A}$ on τ. We use $\mathbf{r}$ to construct a winning strategy from (λ, q_I) for Automaton in the game $G_{\mathcal{A},\tau}$. The strategy is defined as follows: at the root, Automaton should pick the transition $(\mathbf{r}(\lambda), \tau(\lambda), \mathbf{r}(0), \mathbf{r}(1))$, and at all subsequent positions (w, q), he should select the transition $(q, \tau(w), \mathbf{r}(q0), \mathbf{r}(q1))$. If Automaton follows this strategy, then the sequence of states generated by the players' moves must correspond to a path in the run $\mathbf{r}$. Since we know that all paths in $\mathbf{r}$ satisfy the acceptance condition of $\mathcal{A}$, it follows that Automaton must win if he follows this strategy.

Conversely, suppose that Automaton has a winning strategy from (λ, q_I) in the game $G_{\mathcal{A},\tau}$. Then we construct a successful run of $\mathcal{A}$ on τ as follows. At the root node, we assign the unique initial state of $\mathcal{A}$: $\mathbf{r}(\lambda) = q_I$. For the remaining nodes, we proceed level by level. If at some stage, $\mathbf{r}(w)$ is defined, but w's children have not been assigned states, then we set $\mathbf{r}(w0) = q_1$ and $\mathbf{r}(w1) = q_2$, where $(\mathbf{r}(w), \tau(w), q_1, q_2)$ is the transition which is assigned to the game position $(w, \mathbf{r}(w))$ by the winning strategy. Now by definition of the moves of Automaton, $\mathbf{r}$ must be a run of $\mathcal{A}$ on τ. Moreover, it is a successful run because if there were a path in $\mathbf{r}$ which does not satisfy the acceptance condition of $\mathcal{A}$, the Pathfinder could choose this path, and Automaton would not win the game, contradicting the fact that Automaton is using a winning strategy to select his moves. □

According to this lemma, a tree τ is not accepted by $\mathcal{A}$ if and only if Automaton does not have a winning strategy in the game $G_{\mathcal{A},\tau}$. The next step in the proof is to make use of the following important result[3] (which we very unfortunately will not have time to prove in the course):

[3]Actually, Theorem 121 is just a special case of a more general theorem which says

Theorem 121. *Let $\mathcal{A}$ be a parity tree automaton over Σ, and let τ be a Σ-tree. Then for every game position p in the game $G_{\mathcal{A},\tau}$, there exists a memoryless winning strategy for p for either Automaton or Pathfinder. In particular, from the starting position (λ, q_I), either Automaton or Pathfinder has a winning strategy.*

This theorem tells us that a tree τ is not accepted by $\mathcal{A}$ if and only if Pathfinder has a memoryless winning strategy from (λ, q_I) in the game $G_{\mathcal{A},\tau}$. To prove our complementation result, we will show how to use Pathfinder's memoryless winning strategy in $G_{\mathcal{A},\tau}$ to construct an automaton which accepts the complement of $L(\mathcal{A})$.

Theorem 122. *For any parity tree automaton $\mathcal{A}$ over Σ, there exists a parity tree automaton which recognizes exactly those Σ-trees which do not belong to $L(\mathcal{A})$.*

Proof. Consider some parity automaton $\mathcal{A} = (Q, \Sigma, T, I, c)$. We will construct a Müller automaton which accepts the complement of $L(\mathcal{A})$. By Theorem 119, the Müller automaton we construct can then be transformed into an equivalent parity automaton, yielding the desired result.

Now we know from Lemma 120 and Theorem 121 that a Σ-tree τ is not accepted by $\mathcal{A}$ if and only if

($\star$) Pathfinder has a memoryless winning strategy from (λ, q_I) in $G_{\mathcal{A},\tau}$

We wish to rephrase the latter condition in terms of acceptance of the tree τ by some automaton $\mathcal{B}$. First, we remark that Pathfinder's memoryless strategy can be broken down into a set of functions, one for each node w, which state how to compute the direction for any given transition. In other words, we can see Pathfinder's strategy as associating a function f_w to each node w which maps each transition from T to a direction in $\{0, 1\}$. We will let I denote the set of all such functions. Note that this set is finite since T is finite. This means that we can view Pathfinder's strategy as an I-tree σ, where at each node w, we have $\sigma(w) = f_w \in I$. We can thus reformulate the statement ($\star$) as follows:

> There exists an I-tree σ such that for every infinite sequence $t_0 t_1 t_2 \ldots$ of transitions from T (chosen by Automaton), the unique path $\pi \in \{0, 1\}^\omega$ determined by $t_0 t_1 t_2 \ldots$ and the strategy encoded in σ gives rise to an infinite sequence of states which violates the parity condition.

---

that all *parity games* (of which the game $G_{\mathcal{A},\tau}$ is just one example) are determined, i.e. one of the players must have a winning strategy.

We will call I-trees which satisfy the conditions of this statement *winning trees*. Our first objective will be to characterize using infinite word automata when a given I-tree σ is a winning tree. We will work with infinite words over the alphabet Σ' composed of triples (f, a, i) where f is a function from T to $\{0, 1\}$, a is a symbol from Σ, and $i \in \{0, 1\}$ is a direction. Given a strategy tree σ and a Σ-tree τ the language $L_{\sigma,\tau}$ will be composed of all Σ'-words

$$(\sigma(\lambda), \tau(\lambda), \pi(0))\ (\sigma(\pi[0,0]), \tau(\pi[0,0]), \pi(1))\ (\sigma(\pi[0,1]), \tau(\pi[0,1]), \pi(2)) \dots$$

where π is a path (which we view as an infinite string over $\{0, 1\}$). We notice that the last coordinate specifies the path π, and the first two coordinates specify the strategy function and symbol associated with each position in the path by the trees σ and τ. Thus, such a sequence encodes the path π and the restrictions of σ and τ to π, i.e. the words $\sigma|\pi$ and $\tau|\pi$.

Now we create a parity word automaton $\mathcal{B} = (Q, \Sigma', T', q_I, c)$ where Q, q_I, and c are as in the original tree automaton $\mathcal{A}$. For the transition relation T', we include all transitions of the form $(q, (f, a, i), q'_i)$ such that there exists $t = (q, a, q'_0, q'_1) \in T$ such that $f(t) = i$. Here the idea is that we allow the other player Automaton to choose a transition from T. However, we only allow those transitions that would give rise to the path encoded in the last coordinate of the word we are reading (since we want π to be the path resulting from the moves of Automaton and Pathfinder). Now we can give our characterization of winning trees:

Claim: The tree σ is a winning tree iff $L_{\sigma,\tau} \cap L(\mathcal{B}) = \emptyset$.

Proof. For the first direction, let σ be a winning tree, and suppose for a contradiction that $L_{\sigma,\tau} \cap L(\mathcal{B}) \neq \emptyset$. Then there exists a path π such that the Σ'-word

$$\mu = (\sigma(\lambda), \tau(\lambda), \pi(0))\ (\sigma(\pi[0,0]), \tau(\pi[0,0]), \pi(1))\ (\sigma(\pi[0,1]), \tau(\pi[0,1]), \pi(2)) \dots$$

is accepted by $\mathcal{B}$. It follows that there is a successful run $q_0 q_1 q_2 \dots$ of $\mathcal{B}$ on μ with $q_0 = q_I$. This means that for each $j \geq 0$ there is a transition

$$(q_j, (\sigma(\pi[0, j-1]), \tau(\pi[0, j-1]), \pi(j)), q_{j+1})$$

in T', and hence we can find a transition $t_j = (q_j, \tau(\pi[0, j-1]), q'_0, q'_1)$ of $\mathcal{A}$ satisfying $\sigma(\pi[0, j-1])(t_j) = \pi(j)$. Note that if $\pi(j) = 0$, we have $q_{j+1} = q'_0$, otherwise $q_{j+1} = q'_1$. Let us then suppose that the transitions t_j are those

chosen by the player Automaton during the game, and suppose Pathfinder responds following the strategy σ, i.e. plays the moves $\sigma(\pi[0,j])$. Then the induced sequence of states is $q_I q_1 q_2 \ldots$, which satisfies the acceptance condition of $\mathcal{B}$, and hence of $\mathcal{A}$. This contradicts our assumption that σ is a winning tree.

For the second direction, suppose $L_{\sigma,\tau} \cap L(\mathcal{B}) = \emptyset$. Consider some play of the game $G_{\mathcal{A},\tau}$ from the starting position (λ, q_I) such that Pathfinder follows the strategy encoded in σ. Let π be the path created resulting from the play, and let $(q_j, \tau(\pi[0,j-1]), q_{j0}, q_{j1}) \in T$ be the transition chosen by Automaton when at position $(\pi[0,j-1], q_j)$. Finally let $q_I q_1 q_2 \ldots$ be the sequence of states resulting from the play. It is not hard to verify that this sequence of states is a run of the word $\mu \in L_{\sigma,\tau}$ defined above. Since we have assumed that $L_{\sigma,\tau} \cap L(\mathcal{B}) = \emptyset$, we get $\mu \notin L(\mathcal{B})$. It follows that $q_I q_1 q_2 \ldots$ does not verify the acceptance condition of $\mathcal{B}$, hence the acceptance condition of $\mathcal{A}$. Now since the above argument holds for every possible sequence of transitions chosen by Automaton, we have shown that σ satisfies the conditions of winning trees. (end of proof of claim)

We now use the word automaton $\mathcal{B}$ in order to construct a parity tree automaton which recognizes the complement of $L(\mathcal{A})$. The first step is to complement the automaton $\mathcal{B}$, since according to the claim, σ is a winning tree if every word in $L_{\sigma,\tau}$ is *not* in $L(\mathcal{B})$. This can be done e.g. by first translating $\mathcal{B}$ into an equivalent Müller word automaton, then performing complementation, and then translating back to a parity automaton. Let $\mathcal{B}' = (Q', \Sigma', T'', q_I', c')$ be the resulting parity word automaton satisfying $L(\mathcal{B}') = \overline{L(\mathcal{B})}$.

We then use $\mathcal{B}'$ to define the desired parity tree automaton. The main idea is to run $\mathcal{B}'$ in parallel on each path of the input tree τ, while "guessing" the strategy σ. Formally, we define the automaton $\mathcal{A}' = (Q', \Sigma, T'''', q_I', c')$, where Q', q_I', c' are as in the automaton $\mathcal{B}'$, the alphabet Σ is the same as for $\mathcal{A}$, and the transition table T'''' contains a tuple (q, a, q_1, q_2) if and only if $(q, (f, a, 0), q_1) \in T'''$ and $(q, (f, a, 1), q_2) \in T'''$ for some function $f : T \to \{0,1\}$. In order to complete the proof, we must show that $L(\mathcal{A}') = \overline{\mathcal{A}}$.

For the first direction, suppose $\tau \in L(\mathcal{A}')$. Then there is an accepting run $\mathbf{r}$ of $\mathcal{A}'$ on τ. Hence $\mathbf{r}|\pi$ satisfies the parity condition c' for every path π. Moreover, for each position w, there must be a transition $(\mathbf{r}(w), \tau(w), \mathbf{r}(w0), \mathbf{r}(w1))$ in T''''. From the way we have defined T'''', we know that there must exist some function $f_w : T \to \{0,1\}$ such that both $(\mathbf{r}(w), (f, \tau(w), 0), \mathbf{r}(w0))$ and $(\mathbf{r}(w), (f_w, \tau(w), 1), \mathbf{r}(w1))$ belong to the tran-

sition set T''' of the word automaton $\mathcal{B}'$. Define an I-tree σ by setting $\sigma(w) = f_w$. Consider some path π and the corresponding word

$$\mu = (\sigma(\lambda), \tau(\lambda), \pi(0))\ (\sigma(\pi[0,0]), \tau(\pi[0,0]), \pi(1))\ (\sigma(\pi[0,1]), \tau(\pi[0,1]), \pi(2)) \dots$$

in $L_{\sigma,\tau}$. Now given the above, we know that $\mathbf{r}|\pi$ is a run of $\mathcal{B}'$ on μ, and moreover, it must be an accepting run since $\mathbf{r}$ is accepting, and the parity condition is the same for $\mathcal{B}'$ and $\mathcal{A}'$. So we get $\mu \in L(\mathcal{B}')$, and hence $\mu \notin L(\mathcal{B})$. It follows that $L_{\sigma,\tau} \cap L(\mathcal{B}) = \emptyset$, so by the above claim, σ is a winning tree. By applying $(\star)$ and its reformulation and Lemma 120, we obtain $\tau \notin L(\mathcal{A})$.

For the second direction, suppose $\tau \notin L(\mathcal{A})$. Then by Lemma 120 and the statement $(\star)$ and its reformulation, there must exist a winning tree σ. By the above claim, σ is such that $L_{\sigma,\tau} \cap L(\mathcal{B}) = \emptyset$, or equivalently, $L_{\sigma,\tau} \subseteq L(\mathcal{B}')$. So for every path π there exists an accepting run of $\mathcal{B}'$ on the word

$$(\sigma(\lambda), \tau(\lambda), \pi(0))\ (\sigma(\pi[0,0]), \tau(\pi[0,0]), \pi(1))\ (\sigma(\pi[0,1]), \tau(\pi[0,1]), \pi(2)) \dots$$

We note that since $\mathcal{B}'$ is a deterministic parity automaton with a single initial state, if two paths π and π' coincide on an initial segment, then the unique runs of $\mathcal{B}'$ on the corresponding words in $L_{\sigma,\tau}$ will assign the same states to this initial segment. That means that for every position w there is a unique state q such that if π is such that $w = \pi[0,k]$, then the run of $\mathcal{B}'$ on π assigns q to position k. Now define a run $\mathbf{r}$ of $\mathcal{A}'$ on τ by setting $\mathbf{r}(w)$ equal to the unique state q defined as described. Then each path $\mathbf{r}|\pi$ in $\mathbf{r}$ is precisely the run of $\mathcal{B}'$ on the word in $L_{\sigma,\tau}$ induced by the path π. It follows that $\mathbf{r}|\pi$ satisfies the parity condition c'. Thus, $\mathbf{r}$ is an accepting run of $\mathcal{A}'$ on τ, i.e. $\tau \in L(\mathcal{A}')$. □