Automata on Infinite Words

Mahesh Viswanathan

Fall 2018

Recall that the weak monadic second order theory of one successor (WS1S) is the collection of all MSO sentences true in the structure $(\mathbb{N}, 0, S)$ under the restriction that all set quantifiers range over *finite* sets. We showed that WS1S is decidable. The proof of decidability relied on observing that for every MSO formula φ, we can construct a finite automaton M_φ such that the words accepted by M_φ are exactly the encodings of assignments under which φ is true in $(\mathbb{N}, 0, S)$. Our encoding of assignments as words crucially relied on assignments mapping each set variable to finite subset of $\mathbb{N}$ — we mapped each set into the binary word that had 1 at positions that belong to the set; if the set is infinite then this encoding would result in words of infinite length.

Büchi asked the question of whether the decidability of WS1S could be extended to the case where relational variables are not restructed to finite sets. That is, is the set of monadic second order sentences true in $(\mathbb{N}, 0, S)$ decidable? This collection of sentences is called the monadic second order theory of one successor (S1S). Büchi showed that S1S is indeed decidable. His proof followed the basic template of the decidability proof of WS1S of constructing automata corresponding to MSO formulas. However, as pointed out in the previous paragraph, to encode assignments that map relational variables to (possibly) infinite sets as binary strings, we need to consider strings of *infinite* length and automata that process them. Büchi developed this theory of machines working on infinite strings. We will introduce these machines and their properties before we present Büchi's theorem about the decidability of S1S.

1 Büchi Automata

We will begin by defining some preliminaries about infinite length strings. As always, we will use Σ to denote a finite set that will be the alphabet of the strings we consider.

Definition 1. A *sequence/word/string* of infinite length over Σ is $\alpha : \mathbb{N} \to \Sigma$. The ith symbol of string α will be denoted as $\alpha[i]$. The substring from i to j (inclusive) will be denoted as $\alpha[i, j] = \alpha[i] \cdots \alpha[j]$. The suffix starting from position i will be denoted as $\alpha[i, *]$. The collection of infinite length strings over Σ will be denoted as Σ^ω.

A Büchi automaton is a finite state machine (like classical finite automata). An example machine is shown in Figure 1. It has finitely many states, shown as vertices. Transitions (labeled edges in Figure 1) change the

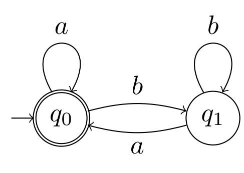

Figure 1: Transition diagram of Büchi automaton M_1 recognizing strings over $\{a, b\}$ that have infinitely many as.

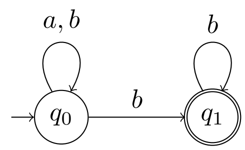

Figure 2: Büchi automaton M_2.

state of the automaton when it reads an input symbol. Intuitively, the automaton starts in the *start/initial* state (shown with an incoming edge with no source in Figure 1). At each step, the automaton reads an input symbol and changes its state according to the transition function. Unlikely a classical automaton, the input never ends, and so acceptance and rejection is defined differently. If a *final/accepting* state (shown with double circled vertices) is visited infinitely often then the input is said to accepted; otherwise it is said to be rejected.

Example 2. Consider the Büchi automaton M_1 shown in Figure 1. After reading a a, the automaton goes to state q_0, no matter what the current state is. Similarly, the automaton goes to state q_1 after reading a b. Since the only final state is q_0, the automaton will accept an input if and only if it has infinitely many as. Thus, $\mathbf{L}_{\exists\mathsf{B}}(M_1) = \{\alpha \in \{a, b\}^\omega \mid \alpha \text{ has infinitely many } a\text{s}\}$.

We define the automaton and its computation formally next.

Definition 3. A (nondeterministic) Büchi automaton is $M = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of (control) states
- Σ is finite (input) alphabet
- $\delta : Q \times \Sigma \to 2^Q$ is transition function
- $q_0 \in Q$ is initial state
- $F \subseteq Q$ the set of final/accepting states

If for every $a \in \Sigma$ and $q \in Q$, $|\delta(q, a)| = 1$ then M is said to be *deterministic*.

A *run* of M on input $\alpha \in \Sigma^\omega$ is an infinite sequence of states $\rho : \mathbb{N} \to Q$ such that $\rho[0] = q_0$ and for every i, $\rho[i + 1] \in \delta(\rho[i], \alpha[i])$. A run ρ is *accepting* if $\rho[i] \in F$ for infinitely many i.

An input α is said to be *accepted* by M if M has an accepting run on α.

Definition 4. The language *accepted/recognized* by Büchi automaton M is $\mathbf{L}_{\exists\mathsf{B}}(M) = \{\alpha \mid M \text{ accepts } \alpha\}$.

A language $A \subseteq \Sigma^\omega$ is said to be *Büchi recognizable* if there is a Büchi automaton M such that $A = \mathbf{L}_{\exists\mathsf{B}}(M)$.

Let us look at some more examples.

Example 5. Consider the automaton shown in Figure 2. On any input string α, the automaton M_2 has a run which stays in state q_0, i.e., the run $q_0^\omega = q_0 q_0 \cdots$. However, this run is not accepting because this run never visits an accepting state (q_1). Any accepting run must got to state q_1, but since q_1 has no transitions on a, it must be the case that once we reach q_1, we never see an a. Such a run is possible only on an input string α that has *finitely many* as. The converse is also true — if α has only finitely many as then there is an accepting run that transitions from q_0 to q_1 on the first b after the last a in α, and then it stays in q_1. Thus, we have

$$\mathbf{L}_{\exists\mathsf{B}}(M_2) = \{\alpha \in \{a, b\}^\omega \mid \alpha \text{ has finitely many } a\text{s}\} = \overline{\mathbf{L}_{\exists\mathsf{B}}(M_1)}$$

The automaton M_2 is nondeterministic. Is there a deterministic Büchi automaton recognizing this language?

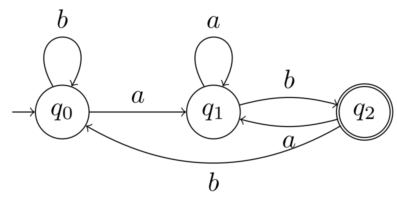

Figure 3: Automaton M_3.

Example 6. Consider automaton M_3 shown in Figure 3. This automaton is deterministic. Observe that the only way to go from q_0 to q_2 is to see both at least one a followed by a b. And to go from q_2 back to q_2 we also need to see at least one a followed by a b. Thus, the strings accepted by M_3 are those that have infinitely many as or finitely many bs. On a string that has finitely many as, the automaton will eventually stay in q_0, while on a string with finitely many bs, the M_3 will eventually stay in state q_1.

$$\mathbf{L}_{\exists\mathsf{B}}(M_3) = \{\alpha \in \{a, b\}^\omega \mid \alpha \text{ has infinitely many } a\text{s and } b\text{s}\}$$

1.1 Deterministic versus Nondeterministic Büchi Automata

Deterministic Büchi automata are weaker than nondeterministic Büchi automata — there are languages that can be recognized by nondeterministic Büchi automata that cannot be recognized by any deterministic machine. We will demonstrate this observatino by showing that the language recognized by M_2 in Example 5 cannot be recognized by a deterministic Büchi automaton. We will prove this result by first observing a "topological" property about languages that are recognized by Büchi automata.

Definition 7. For $U \subseteq \Sigma^*$, its *limit* is defined as

$$\lim(U) = \{\alpha \in \Sigma^\omega \mid \text{there are infinitely many } n \text{ such that } \alpha[0, n] \in U\}$$

Deterministic Büchi automata only recognize languages that are limits of regular languages.

Theorem 8. *$A \subseteq \Sigma^\omega$ is recognized by a deterministic Büchi automaton if and only if there is a regular language $U \subseteq \Sigma^*$ such that $A = \lim(U)$.*

Proof. ($\Rightarrow$) Let M be the deterministic Büchi automaton recognizing A. Take $U = \mathbf{L}(M)$ (i.e., the finite word language recognized by M). We can show that $A = \lim(U)$ as follows. Consider $\alpha \in A$. M has a unique run on α and since α is accepted by M, it visits a final state infinitely often. Each visit to the final state identifies a prefix which belongs to U. Thus, $\alpha \in \lim(U)$. Conversely, if $\alpha \in \lim(U)$, it has infinitely many prefixes in U. The state of M after reading any of these prefixes must be one of the final states. Since this happens for infinitely many prefixes and there are only finitely many final state, some final state must be reached on infinitely many prefixes. Thus $\alpha \in A$.

($\Leftarrow$) Let M be the DFA recognizing U. Then M (viewed as a Büchi automaton) accepts $A = \lim(U)$; the proof of this identical to the argument in the previous paragraph and so we skip it here. $\square$

Theorem 8 can be exploited to show that the language in Example 5 cannot be recognized by a deterministic automaton.

Proposition 9. *The language $A = \{\alpha \in \{a, b\}^\omega \mid \alpha \text{ has finitely many } a\text{s}\}$ is not recognized by any deterministic Büchi automaton.*

Proof. Suppose (for contradiction) A is recignized by a deterinistic Büchi automton. Then, by Theorem 8, there must be $U \subseteq \Sigma^*$ such that $A = \lim(U)$. Now since $b^\omega \in A$, there must be $n_1 \in \mathbb{N}$ such that $b^{n_1} \in U$. Next, since $b^{n_1}ab^\omega \in A$, there is a n_2 such that $b^{n_1}ab^{n_2} \in U$. Repeating this argument, we will get a sequence of natural numbers $n_1, n_2, n_3 \ldots$ such that $\alpha = b^{n_1}ab^{n_2}ab^{n_3}a\cdots \in \lim(U)$. But $\alpha \notin A$, which contradicts the fact that $A = \lim(U)$. $\square$

An immediate corollary of Proposition 9 is that languages recognized by deterministic Büchi automata are not closed under complementation.

Corollary 10. *There is a language A such that A is recognized by a deterministic Büchi automaton but $\overline{A}$ is not recognized by any deterministic Büchi automaton.*

Proof. Observe that $A = \{\alpha \in \{a,b\}^\omega \mid \alpha \text{ has infinitely many } a\text{s}\}$ is recognized by deterministic Büchi automaton M_1 shown in Figure 1. However, by Proposition 9, its complement $\overline{A} = \{\alpha \in \{a,b\}^\omega | \alpha \text{ has finitely many } a\text{s}\}$ cannot be recognized by any deterministic Büchi automaton. □

1.2 Closure Properties

Language recognizable by Büchi automata enjoy many of the closure properties that classical regular languages have.

Theorem 11. *If A_1 and A_2 are Büchi recognizable then so is $A_1 \cup A_2$.*

Proof. The easiest way to prove such a result (for regular languages or recusrively enumerable languages) is to exploit nondeterminism — given an input string α, nondeterministically guess whether $\alpha \in A_1$ or $\alpha \in A_2$, and then run the algorithm for the appropriate A_i to confirm that the guess was correct. Formally, the construction can proceed as follows.

For $i \in \{1,2\}$, let $M_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ be a Büchi automaton recognizing A_i. Without loss of generality, we may assume that $Q_1 \cap Q_2 = \emptyset$. The automaton recognizing $A_1 \cup A_2$ will be $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \cup Q_2 \cup \{q_0\}$, where $q_0 \notin Q_1 \cup Q_2$
- $F = F_1 \cup F_2$, and

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 \\ \delta_2(q,a) & \text{if } q \in Q_2 \\ \delta_1(q_1,a) \cup \delta_2(q_2,a) & \text{if } q = q_0 \end{cases}$$

□

Theorem 12. *If A_1 and A_2 are Büchi recognizable then so is $A_1 \cap A_2$.*

Proof. For $i \in \{1,2\}$, let $M_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ be a Büchi automaton recognizing A_i. Given an input α, the problem $A_1 \cap A_2$ requires us to determine if α belongs to both A_1 and A_2. The classical way to solve this (for either regular languages or recursively enumerable languages) is to run the algorithms M_1 and M_2 (for A_1 and A_2, respectively) *simultaneously* on α and accept if *both* accept. For automata, running two automata simultaneously, can be accomplished by the classical "cross-product" construction. So carrying this out here, we would get the automaton $A_1 \times A_2 = (Q_1 \times Q_2, \Sigma, \delta_1 \times \delta_2, (q_1, q_2), F_1 \times F_2)$ where

$$\delta_1 \times \delta_2((p_1,p_2),a) = \delta_1(p_1,a) \times \delta_2(p_2,a).$$

Unfortunately, this automaton fails to recognize $A_1 \cap A_2$.

Consider the Büchi automata M_1 (Figure 1) that recognizes the collection of all strings having infinitely many as. Let M_4 be the same Büchi automaton as M_1, except the final state is take to be q_1; this automaton recongizes strings that have infinitely many bs. Let us construct the standard cross product of M_1 and M_4; this is shown in Figure 4. Observe that since the unique final state $1B$ can be reached on any input from the initial state $1A$, the cross product automaton accepted the language $\emptyset$ and not $\mathbf{L}_{\exists\mathsf{B}}(M_1) \cap \mathbf{L}_{\exists\mathsf{B}}(M_4) = \{\alpha \in \{a,b\}^\omega \mid \alpha \text{ has infinitely many } a\text{s and } b\text{s}\}$.

The main reason the classical cross product construction fails is because it demands that both M_1 and M_2 meet their obligation to visit their final states *at the same time.* This is too strong a requirement — we need M_1 and M_2 to each visit their final states infinitely often, but not necessarily at the same time. Our Büchi automaton accepting $A_1 \cap A_2$ will run M_1 and M_2 simultaneously, but will proceed in phases to track

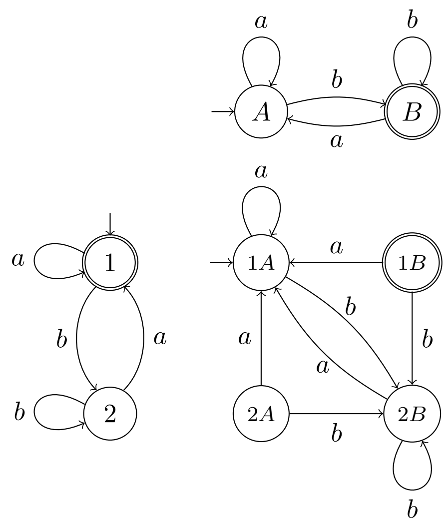

Figure 4: Example showing the failure of the standard cross product construction to recognize the intersection of two Büchi recognizable languages

if both of them visit their final states infinitely often. In every odd phase, it will keep track of whether M_1 has visited a final state *since the previous phase*. Once M_1 visits a final state, the automaton will move to the next phase. In even phases, it will track whether M_2 has visited a final state since the previous phase, and if it does it will advance the phase. Observe that if the obligations are met in *each* phase then both M_1 and M_2 visit their final states infinitely often. On the other hand, if M_1 and M_2 do visit their final states infinitely often, then the obligation in each phase will be met, because no matter how much of the input has been processed, both automata will visit their final states infinitely often in the future.

The formal construction of the "phased" automaton is as follows. The automaton is $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \times Q_2 \times \{1, 2\}$
- $q_0 = (q_1, q_2, 1)$
- $F = Q_1 \times F_2 \times \{2\}$, and

$$\delta((p_1, p_2, i), a) = \begin{cases} \delta_1(p_1, a) \times \delta(p_2, a) \times \{i\} & \text{if } p_i \notin F_i \\ \delta_1(p_1, a) \times \delta(p_2, a) \times \{3 - i\} & \text{if } p_i \in F_i \end{cases}$$

□

Example 13. Consider the automata M_1 from Figure 1 and M_4 described in the proof of Theorem 12 and shown in Figure 4. Their intersection according to the phased construction described in the proof Theorem 12, will result in the automaton M_3 shown in Figure 3; the figure only shows the relevant states that can reached from the initial state.

The most difficult result that Büchi had to prove was the fact that Büchi recognizable languages are closed under complementation. The reason why this proof is difficult is because, unlike classical automata, deterministic Büchi automata are not as powerful as nondeterministic Büchi automata (Proposition 9). Moreover, deterministic Büchi automata are not even closed under complementation (Corollary 10)! We will postpone the proof of this result to later, and we just state this observation for now.

Theorem 14. *If A is Büchi recognizable then $\Sigma^\omega \setminus A$ is also Büchi recognizable.*

We conclude this section by showing that Büchi recognizable labguages are closed under projections. But what are projections? We begin by defining this.

Definition 15. Let Σ_1 and Σ_2 be finite alphabets such that $|\Sigma_2| \leq |\Sigma_1$. A *projection* $\pi : \Sigma_1 \to \Sigma_2$ is an onto function.

A projection $\pi : \Sigma_1 \to \Sigma_2$ naturally extends to (infinite) words and languages as follows: $\beta = \pi(\alpha)$ iff for all i, $\beta(i) = \pi(\alpha(i))$ and $\pi(A) = \{\pi(\alpha) \mid \alpha \in A\}$.

Theorem 16. *If $A \subseteq \Sigma_1^\omega$ is Büchi recognizable and $\pi : \Sigma_1 \to \Sigma_2$ is a projection then $\pi(A)$ is Büchi recognizable.*

Proof. The proof is similar to the showing that regular languages are closed under homomorphisms. Let $M = (Q, \Sigma_1, \delta, q_0, F)$ be an Büchi automaton recognizing A. Then $\pi(M) = (Q, \Sigma_2, \pi(\delta), q_0, F)$ recognizes $\pi(A)$, where

$$\pi(\delta)(q, b) = \cup_{a:\, \pi(a)=b}\delta(q, a)$$

□

1.3 ω-Regularity

Regular languages are a robust class of decision problems that characterize problems solvable with finite memory that enjoy a number of nice properties. There is an analogous class of problems over infinite strings that is "regular" and enjoys many similar properties as classical regular languages. These are called ω-regular languages (to distinguish them from the notion of regularity over finite words), and the notion coincides with Büchi recognizability. Before defining ω-regularity, we need to introduce the operation analogous to Kleene-closure, that constructs languages of infinite strings.

Definition 17. For a *nonempty* subset $U \subseteq \Sigma^*$ with $U \neq \{\varepsilon\}$, the *ω-iteration* of U is the set $U^\omega = \{u_1 u_2 \cdots \in \Sigma^\omega \mid u_i \in U \setminus \{\varepsilon\}\}$.

For $U \subseteq \Sigma^*$ and $A \subseteq \Sigma^\omega$, $UA = \{\beta = u\alpha \mid u \in U \text{ and } \alpha \in A\}$.

Like regular languages, ω-regular languages are generated using union, concatenation and ω-iteration (as opposed to Kleene closure). But in the context of languages of infinite length strings, these operations can be combined in a restrictive manner.

Definition 18. $A \subseteq \Sigma^\omega$ is said to be *ω-regular* if $A = \cup_{i=1}^n U_i V_i^\omega$, where U_i, V_i are regular (finite word) languages.

The notion of ω-regularity coincides with Büchi recognizability.

Theorem 19. *A is ω-regular if and only if A is Büchi recognizable.*

Proof. ($\Rightarrow$) Consider $A = \cup_{i=1}^n U_i V_i^\omega$, where U_i, V_i are regular languages. Since Büchi recognizable languages are closed under union (Theorem 11), our proof that A is Büchi recognizable, follows if we establish the following two observations.

1. If U is regular then U^ω is Büchi recognizable.
2. If U is regular and B is Büchi recognizable then UB is Büchi recognizable.

Let us prove these claims in order.

For 1, let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA recognizing U. The Büchi automaton for U^ω will essentially run M on the input string. Whenever it reads a prefix that is accepted by M, it can either continue to the simulation or "reset" M to identify a new substring of the input that belongs to U. We need to make sure that the automaton is "reset" infinitely many times so that it means that we have found infinitely many substrings of the input that belong to U. This is captured by the following formal definition of automaton $N = (Q', \Sigma, \delta', q_0, \{q_*\})$, where

- $Q' = Q \cup \{q_*\}$ where $q_* \notin Q$, and

$$\delta'(q,a) = \begin{cases} \delta(q_0,a) & \text{if } q = q_* \\ \delta(q,a) & \text{if } q \in Q \text{ and } \delta(q,a) \cap F = \emptyset \\ \delta(q,a) \cup \{q_*\} & \text{otherwise} \end{cases}$$

The proof to establish 2 is similar to the proof that regular languages are closed under concatenation. Let $M_U = (Q_U, \Sigma, \delta_U, q_U, F_U)$ be NFA recognizing U and let $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ be the Büchi automaton recognizing B. Without loss of genrality we will assume that $Q_U \cap Q_B = \emptyset$. The automaton for UA will run M_U, and if a prefix of the input is accepted by M_U, it may choose to process the rest of the input on M_B. The automaton for UB will accept if M_B accepts an appropriate suffix. These ideas result in the following automaton for UB: $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_U \cup Q_B$,
- $q_0 = q_U$,
- $F = F_B$, and

$$\delta(q,a) = \begin{cases} \delta_A(q,a) & \text{if } q \in Q_B \\ \delta_U(q,a) & \text{if } q \in Q_U \text{ and } \delta_U(q,a) \cap F_U = \emptyset \\ \delta_U(q,a) \cup \{q_B\} & \text{otherwise} \end{cases}$$

($\Leftarrow$) Let A be recognized by Büchi automaton $M = (Q, \Sigma, \delta, q_0, F)$. For $s, s' \in Q$, let $V_{ss'} = \{w \in \Sigma^* \,|\, s' \in \hat{\delta}(s,w)\}$; clearly $V_{ss'}$ is regular. Then,

$$A = \cup_{s \in F} V_{q_0 s} V_{ss}^{\omega}$$

□

1.4 Decidability results

ω-regular languages define a tractable class of languages — classical decision problems of emptiness and universality are decidable. We present the result here only for emptiness, but the decidability of universality can be established by reducing it to emptiness, given that ω-regular languages are effectively closed under complementation (see Section 1.5).

Theorem 20. *Given a Büchi automaton $M = (Q, \Sigma, \delta, q_0, F)$, whether $\mathbf{L}_{\exists\mathsf{B}}(M) = \emptyset$ can be determined in linear time and in NL.*

Proof. Let us first give the linear time algorithm. Let $G_M = (V, E)$ be the directed graph where $V = Q$ and $E = \{(s, s') \mid \exists a \in \Sigma.\ s' \in \delta(s,a)\}$. Observe $\mathbf{L}_{\exists\mathsf{B}}(M) \neq \emptyset$ iff there is a strongly connected component C [1] of G_M that is reachable from q_0 that contains a state of F. The algorithm, therefore, is to compute maximal strongly connected components of G_M (linear time), check if one of the reachable ones contains a final state (linear time).

The NL algorithm is the same. Just as in the previous paragraph, we can show that $\mathbf{L}_{\exists\mathsf{B}}(M) \neq \emptyset$ iff there is a vertex $s \in F$ reachable from q_0 in G_M such that s is on a directed cycle. The nondeterministic algorithm has two phases. In the first phase it guess a path from q_0 vertex-by-vertex until it reaches a final state that it guesses to be on a cycle. In the second phase it guesses the cycle. Notice that to carry this clagorithm out, we only need to keep track of the current vertex on the path/cycle and the vertex s that we guess to be on the cycle. This takes space $O(\log n)$, where M has n states. □

The fact that emptiness can be decided allows us to get decision procedures for fragments of MSO on restricted structures. We can get analogues of the decidability results for MSO on words (the Büch-Elgot-Trakhtenbrot theorem) and WS1S.

[1] C is strongly connected if for every pair $u, v \in C$ there is a directed path from u to v and one from v to u.

Recall that a word $w \in \Sigma^\omega$ can be represented as a structure $\mathcal{W} = (\mathbb{N}, <^{\mathcal{W}}, S^{\mathcal{W}}, (Q_a^{\mathcal{W}})_{a\in\Sigma})$ over the signature $\tau_W = \{<, S, (Q_a)_{a\in\Sigma}\}$, where $<^{\mathcal{W}}$ and $S^{\mathcal{W}}$ are the standard ordering and successor relations on $\mathbb{N}$, and $Q_a = \{p \in \mathbb{N} \mid w(p) = a\}$. As always this correspondence between infinite sequences over Σ and word structures is bijective, and so we will represent these by the same object.

We can extend the notions of MSO definability to the case of infinite words. As always, the *models* of an MSO sentence φ (denoted $[\![\varphi]\!]$) is the set of all word structures that satisfy φ. A language $A \subseteq \Sigma^\omega$ is said to be *definable in monadic second-order logic* if there is a sentence φ such that $w \in A$ iff w (viewed as a structure) satisfies φ. In such cases we will abuse notation, and say $A = [\![\varphi]\!]$. The Büchi-Elgot-Trakhtenbrot result extends to the case of ω-regular languages. That is, ω-regular languages are exactly those that are definable in MSO.

Theorem 21 (Büchi)**.** *$A \subseteq \Sigma^\omega$ is definable in MSO iff A is ω-regular.*

Proof. The proof is identical to that of the Büchi-Elgot-Trakhtenbrot theorem. To show the implication from left to right, for each MSO formula φ, we can inductively construct a Büchi automaton M_φ such that $\mathcal{W} \models \varphi[\alpha]$ iff $\mathrm{enc}(\mathcal{W}, \alpha) \in \mathbf{L}_{\exists\mathsf{B}}(M_\varphi)$, where $\mathrm{enc}(\mathcal{W}, \alpha)$ is the encoding of the structure $\mathcal{W}$ and assignment α as an infinite word. Here we use the fact that ω-regular languages are closed under union, intersection, complementation, and projection.

Conversely, given a Büchi automaton M, the fact that M has an accepting run on word w can encoded in MSO. The resulting formula will have the same special structure as in the Büchi-Elgot-Trakhtenbrot theorem. $\square$

Theorems 20 and 21 together imply that the satisfiability and validity problems for MSO on words is decidable.

Corollary 22. *Given an MSO sentence φ, checking wether φ is satisfiable or valid on word structures is decidable.*

We conclude this section by establishing the initial reason behind why the theory of automata on infinite words was developed by Büchi — the decidability of the monadic second order theory of one successor (S1S).

Definition 23. The monadic second order theory of one successor (S1S) is the collection of MSO sentences true in the structure $(\mathbb{N}, 0, S)$. That is, $\mathsf{S1S} = \{\varphi \text{ sentence in MSO} \mid (\mathbb{N}, 0, S) \models \varphi\}$.

Theorem 24 (Büchi)**.** $\mathsf{S1S}$ *is decidable.*

Proof. The proof is similary to the proof for the decidability of WS1S. For any MSO formula φ, we can construct a Büchi automaton M_φ such that $(\mathbb{N}, 0, S) \models \varphi[\alpha]$ iff $\mathrm{enc}(\alpha) \in \mathbf{L}_{\exists\mathsf{B}}(M_\varphi)$, where $\mathrm{enc}(\alpha)$ is an encoding of assignment α as in infinite binary word. The construction of M_φ is inductive on φ is similar to the construction for WS1S. The decidability of S1S then follows from Theorem 20. $\square$

1.5 Complementation

Theorems 21 and 24 rely on closure properties of ω-regular languages. In particular, they rely on their closure under complementation (Theorem 14). The proof of Theorem 14 is difficult and was Büchi's main technical challenge in establishing the decidability of S1S (Theorem 24). His original proof constructed an automaton of doubly exponential size in a logical form. The closure under complementation is an important result with practical applications, and therefore, since Büchi's orginal attempt, there have been many modern attempts to reprove this result. These new proofs have made the argument simpler and improved the assymptotic size of the constructed automaton. There are broadly four approaches to establishing this result: Ramsey-based approach, determinization-based approach, rank-based approach, and slice-based approach. We will present a Ramsey-based proof that is the easiest of these proofs and is close to optimal in terms of the resulting automaton. It is broadly based on ideas in Büchi's original proof and subsequent improvements proposed by Vardi, Wolper, and Sistla.

The proof exploits *equivalences* on words. Let us illustrate the broad principles of the proof in the context of regular languages (on finite words). Consider a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$. We would like to define a notion of equivalence on Σ^* such that if v_1 and v_2 are equivalent, then replacing the substring v_1 by v_2 in any input does not "affect" the behavior of M. That is, for any u, w, if uv_1w is accepted then so is uv_2w, and if uv_1w is not accepted then neither is uv_2w. This is easy to achieve in the context of classical automata — say $v_1 \equiv_M v_2$ if $\hat{\delta}(q, v_1) = \hat{\delta}(q, v_2)$ for every $q \in Q$. The resulting equivalence satisfies the desired properties, namely, if $v_1 \equiv_M v_2$ then for any u, w, $uv_1w \in \mathbf{L}(M)$ iff $uv_2w \in \mathbf{L}(M)$. Moreover, we have the property that $\equiv_M$ *saturates* the language $\mathbf{L}(M)$, i.e., any equivalence class of $\equiv_M$ is either contained in $\mathbf{L}(M)$ or is disjoint from it. Saturation ensures that $\mathbf{L}(M)$ is a union of equivalence classes of $\equiv_M$ and $\overline{\mathbf{L}(M)}$ is also a union of equivalence classes. Thus, if each equivalence class of $\equiv_M$ can be recognized by a finite automaton, and since regular languages are closed under union, these observations can be used to prove that $\overline{\mathbf{L}(M)}$ is also regular.

To generalize the above argument to the case Büchi automata, the first step is to identify an appropriate notion of equivalence on words. The idea is once to define an equivalence on words in Σ^* with the property that replacing equivalent substrings in an input string does not change the behavior of the automaton. In the context of infinite length strings, we need to consider two types of replacements. One is replacing a substring v_1 once (or finitely many times) in an infinite word α. The other is to replace v_1 in *infinitely many* places inside α. While invariance under finite replacements can be ensured if the words v_1 and v_2 lead the automaton to the same states as in the finite word case, to ensure invariance under infinitely many replacements, we need to ensure that if v_1 goes through a final state during the computation, then so does v_2. We will define this informal intuition precisely. In order to do that, we need to introduce some new notation.

For the rest of this section let us fix a Büchi automaton $M = (Q, \Sigma, \delta, q_0, F)$. For $v \in \Sigma^*$, $\hat{\delta}(q, v)$ is the set of states M could be in after reading v when started in state q. We defined this inductively as

$$\hat{\delta}(q, v) = \begin{cases} \{q\} & \text{if } v = \varepsilon \\ \bigcup_{p \in \hat{\delta}(q,u)} \delta(p, a) & \text{if } v = ua \end{cases}$$

We will also define the states of M on input v reached by *going through* a final state. We denote this by the set $\hat{\delta}_F(q, w)$. This can be defined inductively as follows.

$$\hat{\delta}_F(q, v) = \begin{cases} \emptyset & \text{if } v = \varepsilon \text{ and } q \notin F \\ \{q\} & \text{if } v = \varepsilon \text{ and } q \in F \\ \bigcup_{p \in \hat{\delta}_F(q,u)} \delta(p, a) \cup \bigcup_{p \in \hat{\delta}(q,u)} (\delta(p, a) \cap F) & \text{if } v = ua \end{cases}$$

Having defined the sets of states reached on input v, and the set of states reached on input v by passing through a final state, we are ready to define the equivalence on strings that we will use in the complementation construction.

Definition 25. Two strings $v_1, v_2 \in \Sigma^*$ are said to *equivalent* (w.r.t. Büchi automaton M), denoted $v_1 \equiv_M v_2$, if for every state q, $\hat{\delta}(q, v_1) = \hat{\delta}(q, v_2)$ and $\hat{\delta}_F(q, v_1) = \hat{\delta}_F(q, v_2)$.

Let us look at an example to understand this definition of equivalence.

Example 26. Consider the Büchi automaton M_3 should in Figure 3. Let us consider the string $v = b$. Let us first identify what the functions $\hat{\delta}$ and $\hat{\delta}_F$ are for this input v.

$$\begin{array}{ll} \hat{\delta}(q_0, v) = \{q_0\} & \hat{\delta}_F(q_0, v) = \emptyset \\ \hat{\delta}(q_1, v) = \{q_2\} & \hat{\delta}_F(q_1, v) = \{q_2\} \\ \hat{\delta}(q_2, v) = \{q_0\} & \hat{\delta}_F(q_2, v) = \{q_0\} \end{array}$$

Consider $u = bab$. u is not equivalent to v; one way to see this is because $\hat{\delta}(q_0, u) = \{q_2\} \neq \{q_0\} = \hat{\delta}(q_0, v)$. We can also see that bb is also not equivalent to b because $\hat{\delta}(q_1, bb) = \{q_0\} \neq \{q_2\} = \hat{\delta}(q_1, v)$. In fact, the only string equivalent to v is v itself. We can reason about this as follows. $\varepsilon \not\equiv_{M_3} v$ because $\hat{\delta}(q_1, \varepsilon) = \{q_1\} \neq \hat{\delta}(q_1, v)$. If we consider any string of the form ua (i.e., ends in a a), $\hat{\delta}(q_0, ua) = \{q_1\} \neq \hat{\delta}(q_0, v)$. If the strings ends in a b, there are two possibilities. If it is of the form ubb then $\hat{\delta}(q_1, ubb) = \{q_0\} \neq \hat{\delta}(q_1, b)$. On the other hand, for any string of the form uab, $\hat{\delta}_F(q_0, uab) = \{q_2\} \neq \hat{\delta}_F(q_0, v)$.

On the other hand, observe that $a^i \equiv_{M_3} a^j$ for any $i, j \geq 1$. This is because for any $i \geq 1$, we have the following observations.

$$\begin{array}{ll} \hat{\delta}(q_0, a^i) = \{q_1\} & \hat{\delta}_F(q_0, a^i) = \emptyset \\ \hat{\delta}(q_1, a^i) = \{q_1\} & \hat{\delta}_F(q_1, a^i) = \emptyset \\ \hat{\delta}(q_2, a^i) = \{q_1\} & \hat{\delta}_F(q_2, a^i) = \{q_1\} \end{array}$$

The *index* of an equivalence relation is the number of equivalence classes that it defines. Eventhough $\equiv_M$ is a relation on an infinite set (Σ^*) it has only finitely many equivalence classes.

Proposition 27. *For any Büchi automaton M, $\equiv_M$ has finite index.*

Proof. Based on the definition of $\equiv_M$, we can see that the number of equivalence classes of $\equiv_M$ is bounded by the number of functions $f : Q \to (2^Q \times 2^Q)$, where Q is the set of states of M. The number of such functions is $\leq 2^{O(|Q|^2)}$, which is finite. $\square$

For any string $v \in \Sigma^*$, let us denote by $[v]_M$ the equivalence class of v with respect to $\equiv_M$. In other words,

$$[v]_M = \{u \in \Sigma^* \mid u \equiv_M v\}.$$

The equivalence class of any string turns out to form a regular language.

Theorem 28. *For any string $v \in \Sigma^*$, $[v]_M$ is regular.*

Proof. We will construct a DFA that recognizes $[v]_M$. Let the Büchi automaton M be $(Q, \Sigma, \delta, q_0, F)$. Let R be the set of all functions of type $Q \to (2^Q \times 2^Q)$; as observed in Proposition 27 every equivalence class of $\equiv_M$ is identified by such a function and R is a finite set. Let $f_* \in \mathsf{R}$ be the function such that $f_*(q) = (\hat{\delta}(q, v), \hat{\delta}_F(q, v))$, i.e., f_* is the equivalence class of v.

Our DFA D will track the equivalence class (which is a member of R) of the input we have read so far. The invariant we will maintain is that the state reached after reading a prefix u of the input is the function $f \in \mathsf{R}$ where $f(q) = (\hat{\delta}(q, u), \hat{\delta}_F(q, u))$. The formal definition of D is as follows. $D = (\mathsf{R}, \Sigma, \delta_D, f_\varepsilon, \{f_*\})$ where

- f_ε is the function such that for any $q \in Q$, $f_\varepsilon(q) = (\hat{\delta}(q, \varepsilon), \hat{\delta}_F(q, \varepsilon))$, and
- $\delta(f, a) = g$ where $f(q) = (Q_1, Q_2)$, and

$$g(q) = \left(\bigcup_{p \in Q_1} \delta(p, a), \bigcup_{p \in Q_2} \delta(p, a) \cup \bigcup_{p \in Q_1} (\delta(p, a) \cap F) \right)$$

$\square$

Consider a pair of strings $u, v \in \Sigma^*$. Let us look at the set $B = [u]_M [v]_M^\omega \subseteq \Sigma^\omega$. We can show that such sets *saturate* $\mathbf{L}_{\exists\mathsf{B}}(M)$. That is, either $B \subseteq \mathbf{L}_{\exists\mathsf{B}}(M)$ or $B \cap \mathbf{L}_{\exists\mathsf{B}}(M) = \emptyset$. The reason is because of our careful definition of $\equiv_M$. Suppose $uv^\omega \in \mathbf{L}_{\exists\mathsf{B}}(M)$. Our definition of $\equiv_M$ ensures that "essentially" the accepting run of M on uv^ω is also an accepting run for any string in B.

Theorem 29. *For any strings $u, v \in \Sigma^*$, either $[u]_M [v]_M^\omega \subseteq \mathbf{L}_{\exists\mathsf{B}}(M)$ or $[u]_M [v]_M^\omega \cap \mathbf{L}_{\exists\mathsf{B}}(M) = \emptyset$.*

Proof. Let $B = [u]_M[v]_M^\omega$ and $A = \mathbf{L}_{\exists \mathsf{B}}(M)$. Suppose $B \cap A \neq \emptyset$; we need to argue that then $B \subseteq A$. This means that there are strings $u_1 \in [u]_M$ and $v_i \in [v]_M$ for $i \in \mathbb{N}$ such that $\alpha = u_1 v_1 v_2 \cdots \in A$. Let ρ be an accepting run of M on α. It will be useful to identify some of the states in this run ρ. ρ begins in the initial state q_0 of M. Let p_0 be the state of M in ρ after reading u_1, and p_i be the state of M in ρ after reading the prefix $u_1 v_1 v_2 \cdots v_i$, for $i \geq 1$. Using $p \xrightarrow{w} q$ to indicate a run of M from p to q on input w, we can think of ρ as the following run.

$$\rho = q_0 \xrightarrow{u_1} p_0 \xrightarrow{v_1} p_1 \xrightarrow{v_2} p_2 \cdots$$

In addition, since ρ is accepting, for infinitely many i, the subrun $p_{i-1} \xrightarrow{v_i} p_i$ goes through a final state.

Consider an arbitrary string $\beta \in B$. We can write β as $u_1' v_1' v_2' \cdots$, where $u_1' \equiv_M u_1$ and $v_i' \equiv_M v_i$ for all i. From the definition of $\equiv_M$, we know that there is a run of the form $q_0 \xrightarrow{u_1'} p_0$. In addition, for every i, there is a run $p_{i-1} \xrightarrow{v_i'} p_i$ and if the subrun $p_{i-1} \xrightarrow{v_i} p_i$ vists a final state, then there is a run $p_{i-1} \xrightarrow{v_i'} p_i$ that also visits a final state. Putting these subruns together, we get the run

$$\rho' = q_0 \xrightarrow{u_1'} p_0 \xrightarrow{v_1'} p_1 \xrightarrow{v_2'} p_2 \cdots$$

which is also accepting. Thus, $\beta \in A$, which establishes the fact that $B \subseteq A$. $\square$

Restating Theorem 29, we have the observation. For any pair $u, v \in \Sigma^*$, either $[u]_M[v]_M^\omega$ is either contained in $\mathbf{L}_{\exists \mathsf{B}}(M)$ or $\overline{\mathbf{L}_{\exists \mathsf{B}}(M)}$. Since $\equiv_M$ is finite index (Proposition 27), there are only finitely many sets of the form $[u]_M[v]_M^\omega$, for $u, v \in \Sigma^*$. Based in these observations, we could say

$$\overline{\mathbf{L}_{\exists \mathsf{B}}(M)} \supseteq \bigcup_{u,v : uv^\omega \notin \mathbf{L}_{\exists \mathsf{B}}(M)} [u]_M[v]_M^\omega. \tag{1}$$

Observe that the right hand side of the above equation is an ω-regular language because the union is a finite union and each equivalence class of $\equiv_M$ is regular (Theorem 28). If we can prove that the two sets in the above equation are actually equal, then we can conclude Theorem 14 because of the equivalence of ω-regular languages and Büchi recognizable languages (Theorem 19).

We will establish that the two sets in Equation (1) are equal by observing that for every $\alpha \in \Sigma^\omega$, there are $u, v \in \Sigma^*$ such that $\alpha \in [u]_M[v]_M^\omega$. This is because consider an arbitrary string $\alpha \in \overline{\mathbf{L}_{\exists \mathsf{B}}(M)}$. Then (by the still to be proven technical observation), there are $u, v \in \Sigma^*$ such that $\alpha \in [u]_M[v]_M^\omega$. Now, by Theorem 29, it must be the case that since $uv^\omega \in [u]_M[v]_M^\omega$, $uv^\omega \notin \mathbf{L}_{\exists \mathsf{B}}(M)$. Therefore, α belongs to the set on the right hand side of Equation (1), which establishes that they sets are equal.

To complete the proof of Theorem 14, we need to show that for every $\alpha \in \Sigma^\omega$ there are $u, v \in \Sigma^*$ such that $\alpha \in [u]_M[v]_M^\omega$. To prove this result we will use the infinite Ramsey theorem.

Theorem 30 (Infinite Ramsey Theorem)**.** *Let X be any infinite set, and let $X^{(n)}$ denote the set of subsets X of size n. Let $c : X^{(n)} \to \{1, 2, \ldots k\}$ be any assignment of colors (from a finite set) to elements of $X^{(n)}$. Then there is a infinite subset $S \subseteq X$ such that every set in $S^{(n)}$ (subsets of S of size n) has the same color.*

We postpone the proof of Theorem 30 to the end of this section for the interested reader. We will first state and prove the technical lemma that completes the proof of Theorem 14.

Theorem 31. *For any $\alpha \in \Sigma^\omega$, there are $u, v \in \Sigma^*$ such that $\alpha \in [u]_M[v]_M^\omega$.*

Proof. For $i \leq j \in \mathbb{N}$, recall that $\alpha[i, j]$ is the substring of α from i to j. Consider the coloring that maps a pair (i, j) to the equivalence class of $\alpha[i, j-1]$ with respect to $\equiv_M$. Since $\equiv_M$ has finite index (Proposition 27), using Theorem 30, we can conclude that there is a infinite subset $X \subseteq \mathbb{N}$ such that every pair from X gets the same "color". That is, taking $X = \{i_j \mid j \in \mathbb{N}\}$ such that $i_j < i_{j+1}$, we have $\alpha[i_j, i_{j+1} - 1] \equiv_M \alpha[i_k, i_{k+1} - 1]$ for any j, k. Taking, $u = \alpha[0, i_0 - 1]$, and $v = \alpha[i_0, i_1 - 1]$, we have $\alpha \in [u]_M[v]_M^\omega$. $\square$

Analysis. Let us assume that the Büchi automaton M has n states. Observe that the automaton constructed in the proof of Theorem 28 has size $2^{O(n^2)}$, which is the same as the number of equivalence classes of $\equiv_M$. Based on the constructions in Theorem 19, we know that the size of the Büchi automaton recognizing UV^ω (for any regular languages U and V) is around the sum of the sizes of automata recognizing U and V. This means that the size of Büchi automaton recognizing $[u]_M[v]_M^\omega$ (for $u, v \in \Sigma^*$) is also $2^{O(n^2)}$. Finally, the Büchi automaton recognizing $A_1 \cup A_2$, for Büchi recognizable A_1 and A_2, is the sum of the sizes of automata recognizing A_1 and A_2 (Theorem 11). Therefore, since the union on the right in Equation (1) has at most $2^{O(n^2)}$ languages, and each language can be recognized by automata of size $2^{O(n^2)}$, we get the automaton recognizing $\overline{\mathbf{L}_{\exists\mathsf{B}}(M)}$ is of size $2^{O(n^2)}$. This is not the best construction. Many alternate constructions produce an automaton of size $2^{O(n \log n)}$. The best construction known today, by Schewe, produces an automaton for the complement with $O((0.76n)^n)$ states. Schewe's construction matches the best known lower bound due to Yan of $O((0.76n)^n)$ within a factor of $O(n^2)$.

Proof of Theorem 30. To be written. See wikipedia page on the Infinite Ramsey Theorem. □