ω-Automata

Ming Gong

December 2023

This semester, we focused on languages defined by finite strings, where the set of possible strings could be infinite, yet each individual string remained finite. In this project, I will explore the realm of infinite strings, commonly known as ω-strings. I will explore the language of infinite strings, their properties, and the machines that work with them.

1 ω-Language

An ω-string is a sequence of alphabets with length ω, where ω refers to the first ordinal number (the set of natural numbers).

Definition. An ω-string over an alphabet Σ is a function $\alpha : \mathbb{N} \to \Sigma$.

$\alpha(n)$ denotes the character at the nth place of the string. An ω-string must be infinite.

Example. Let

$$\alpha(n) = \begin{cases} 1 & \text{if } 3 \mid n, \\ 0 & \text{else.} \end{cases}$$

This string is 001001001..., an infinite repeat of 001.

A set of ω-strings is an ω-language.

Definition. An ω-language is a subset of Σ^ω, where Σ^ω is the set of all ω-strings.

Definition. Let L be a language on finite strings over alphabet Σ. The ω-iteration of L, L^ω, is the set of all infinite strings of the form

$$s_1 s_2 s_3 \ldots$$

such that each $s_i \in L \; \forall i \in \mathbb{N}$.

Example. $\{\epsilon\}^\omega = \emptyset$.

Any continuation of ϵ will be the empty string. Because ω-strings are defined to have infinite length, $\{\epsilon\}^\omega$ has to be the empty language.

Later in this paper, unless otherwise specified, the word "string" refers to ω-string, and "language" refers to ω-language.

2 ω-Automata

An ω-automaton is a variation of a finite automaton that runs on infinite strings as an input.

Finite automata have a set of final states, but since ω-automata never stops on an infinite input, we need different acceptance conditions.

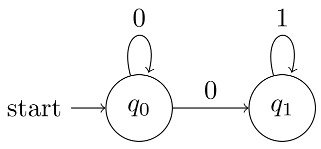
Definition. Let M be a nondeterministic automaton. A *run* of M on an input $s = a_0 a_1 a_2 \ldots$ is an infinite sequence

$$q_0 q_1 q_2 \ldots$$

of states that begins in the start state q_0 and $\forall i \geq 0$, $q_{i+1} \in \delta(q_i, a_i)$.

Definition. Let $\mathbf{r}$ be a run. The *infinite set* of $\mathbf{r}$, $\text{Inf}(\mathbf{r})$, is the set of states that appear infinitely often in $\mathbf{r}$.

Example. On input 001111...



Here is a valid run: $\mathbf{r} = q_0q_0q_1q_1q_1q_1q_1...$ $\mathbf{r}$ starts in q_0 and stays in q_0 after reading the first 0. Then, it transitions and loops in q_1 on the rest of the input.

$\text{Inf}(\mathbf{r}) = \{q_1\}$.

Definition. A deterministic ω-automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, Acc)$ such that:

- Q is a finite set of states.
- Σ is a finite set of alphabets.
- $\delta : Q \times \Sigma \to Q$ is the transition function.
- q_0 is the starting state.
- Acc is the *acceptance condition*, which is a subset of Q^ω.

An ω-automaton can also be nondeterministic. In this case, the transition function will be:

- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$.

In the formal definition, the acceptance condition Acc is a set of accepted runs, a subset of all possible infinite sequences of states (runs). This is cumbersome to enumerate. Typically, we are only interested in certain aspects of a run. For example, in DFAs, Acc will be the set of runs whose last state is a final state. In ω-automata, we also seek to abstract and encode the accepting runs, leading to different types of ω-automata.

After defining the terminologies, we can look at different acceptance conditions. Let $\mathbf{r}$ be the machine M's run on an input.

- Büchi condition: Let $F \subseteq Q$ be a set of *final states.* $\mathbf{r} \in Acc$ if and only if $\text{Inf}(\mathbf{r}) \cap F \neq \emptyset$.

  In other words, a final state is visited infinitely often.

- Muller condition: Let $\mathcal{F} \subseteq \mathcal{P}(Q)$ be the *acceptance set.* $\mathbf{r} \in Acc$ iff $\text{Inf}(\mathbf{r}) \in \mathcal{F}$.

  In other words, a specified set of states is visited infinitely often.

- Rabin condition: Let $\Omega \subseteq \mathcal{P}(Q \times Q)$ be the set of *accepting pairs.* $\Omega = \{(E_1, F_1)...(E_k, F_k)\}$. $\mathbf{r} \in Acc$ iff $\exists (E_i, F_i) \in \Omega$ such that $\text{Inf}(\mathbf{r}) \cap E_i = \emptyset$ and $\text{Inf}(\mathbf{r}) \cap F_i \neq \emptyset$.

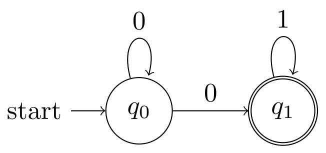
  In other words, a string is accepted if specific states E_i is visited finitely often, but F_i is visited infinitely often.

3 Büchi Automata

A Büchi Automaton (NBA) is a nondeterministic ω-automaton with the Büchi acceptance condition. It can be formalized as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$: the states, symbols, transition function, initial state, and final states.

Definition. A language $L \subseteq \Sigma^\omega$ is *Büchi-recognizable* if there exists a Büchi automaton M such that $L = L(M)$.

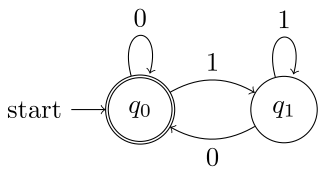
Example.



An accepting run must visit a final state (q_1 in this case) infinitely often. In q_1, it can only read 1s. If any 0 is read at q_1, this nondeterministic branch will die off.

Starting from q_0, an accepting run begins by reading a non-empty block of 0s and transitioning to q_1. Then, it loops in q_1 infinitely often by reading 1s. This automaton accepts all ω-strings in the form 0^+1^ω.

Example.



An accepting run must visit q_0 infinitely often. This can be done by either reading 0s or a block of 1s followed by a 0. As long as we have an infinite "supply" of 0s, q_0 can be visited infinitely often.

Essentially, this automaton accepts all strings with an infinite number of 0s.

4 Closure Properties

We are interested in whether the languages of Büchi automata are closed under different operations, such as concatenation, iteration, union, intersection, and complement.

In this section, suppose the languages use the same alphabet set Σ. For a language L_i, let $L(M_i) = L_i$, where $M_i = (Q_i, \Sigma, \delta_i, q_{0i}, F_i)$.

Theorem. Büchi-recognizable languages are closed under union.

Proof. The construction is very similar to taking the union of two NFA languages. Combine the states and transitions and nondeterministically simulate both machines in parallel. If one branch has a successful run, accept.

Let the two languages be L_1, L_2. Construct $M = (Q, \Sigma, \delta, q_0, F)$, where

- $Q = Q_1 \cup Q_2 \cup \{q_0\}$.
- $\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1, \\ \delta_2(q,a) & \text{if } q \in Q_2, \\ \delta_1(q_{01},a) \cup \delta_2(q_{02},a) & \text{if } q = q_0. \end{cases}$
- $F = F_1 \cup F_2$.

□

Theorem. Büchi-recognizable languages are closed under intersection.

Proof. We can use the construction for DFA language intersection to keep track of the states from both M_1 and M_2 and make $F = F_1 \times F_2$. However, the Büchi condition is less restrictive. For a string to be in the intersection, its run only needs to visit some final states in F_1 and F_2 infinitely often, but not necessarily visit them simultaneously infinitely often.

On an input s, M keeps track of both the states in M_1 and M_2, but instead of searching for final states in both, it first searches a final state of M_1. Once it visits a state of M_1, it searches for a final state of

M_2. Again, once it visits a final state in M_2, it searches for a final state of M_1, and this process repeats. If both M_1 and M_2 accept s, the final states in M_1 and M_2 can be found infinitely often, so this alternating algorithm can repeat forever.

M has two modes: "F_1-seeking" and "F_2-seeking". This can be done by adding additional states specifying the mode.

We want M to accept if this alternate checking process can repeat forever. Whether the "mode switching" states can be visited infinitely often is an indicator.

Formally, construct a Büchi automaton $M = (Q, \Sigma, \delta, q_0, F)$, where

- $Q = Q_1 \times Q_2 \times \{1, 2\}$.

  $\{1, 2\}$ specifies the mode.

- $\delta((q_1, q_2, m), a) = (\delta_1(q_1), \delta_2(q_2), t(m))$, where $q_1 \in Q_1, q_2 \in Q_2, m \in \{1, 2\}, a \in \Sigma$,

  $$t(m) = \begin{cases} 1 & (m = 1 \text{ and } q_1 \notin F_1) \text{ or } (m = 2 \text{ and } q_2 \in F_2), \\ 2 & (m = 2 \text{ and } q_2 \notin F_2) \text{ or } (m = 1 \text{ and } q_1 \in F_1). \end{cases}$$

  In other words, $t(m)$ "switches" the "mode." Once a final state is visited, M starts seeking a final state in the other machine. If a final state is not visited, M stays in the current mode.

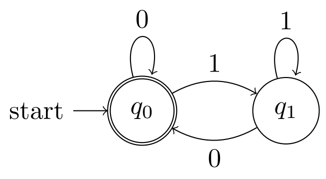
- $q_0 = (q_{01}, q_{02}, 1)$.

- $F = F_1 \times Q_2 \times \{1\} \cup Q_1 \times F_2 \times \{2\}$.

  In other words, M can successfully visit the "mode-switching" states (a state in F_1 in the F_1-seeking mode and a state in F_2 in the F_2-seeking mode). This can be simplified to $F = Q_1 \times F_2 \times \{2\}$ because reentering the F_2-seeking mode infinitely often implies visiting the states in F_1 infinitely often.

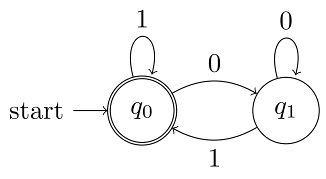
$\square$

Example. Let M_1=



From section 3, M_1 accepts all strings with an infinite number of 0s.

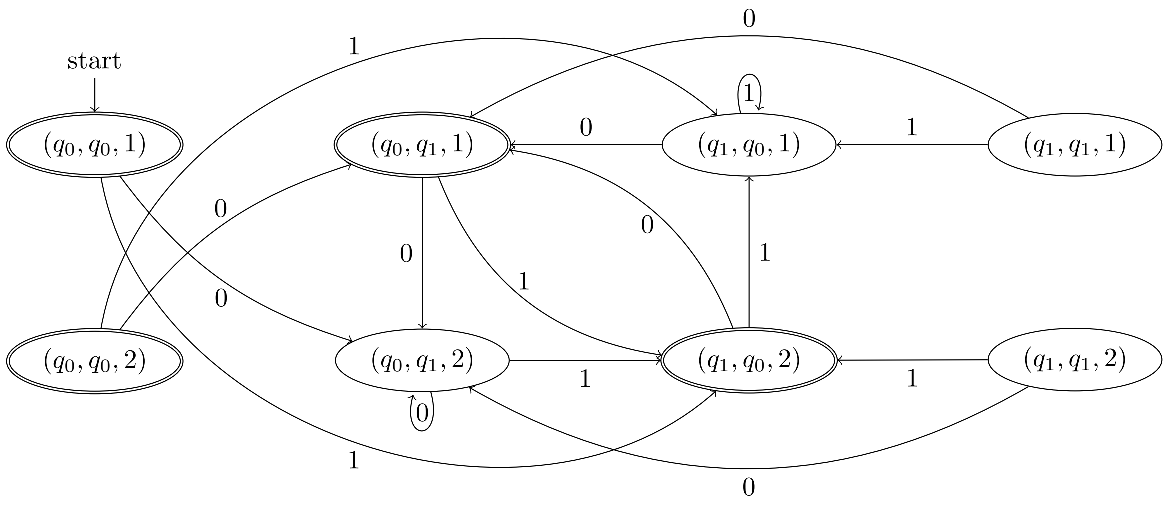
Example. Let M_2 =



M_2 accepts all strings with an infinite number of 1s.

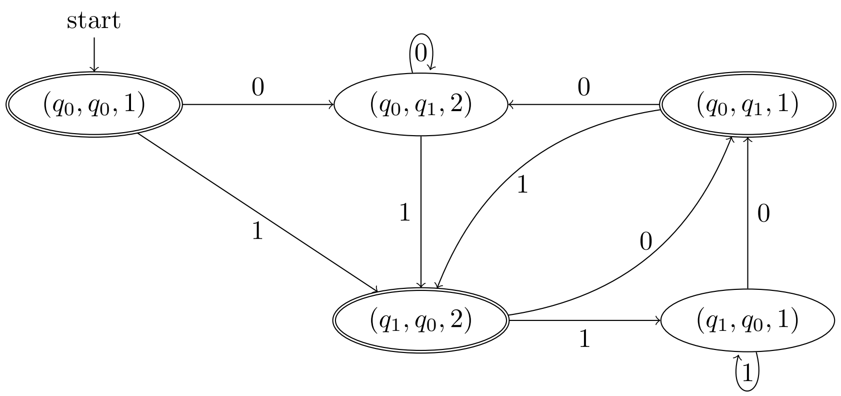
Using the algorithm above to construct M.

Let "mode 1" be the F_1-seeking mode, and "mode 2" be the F_2-seeking mode.



That was a pain to arrange in LaTeX...

Simplify this by deleting the states that are never reached.



M starts in mode 1. Since it starts in q_0 of M_1, which is already a final state, it will switch to mode 2.

Once M reads the target symbol in its mode, it will transition to a final state. If not, M will keep looping. If the input string runs out of 0s or 1s, M will loop forever in $(q_1, q_0, 1)$ or $(q_0, q_1, 2)$, only visiting the final states finitely often.

ω-languages are only allowed to concatenate after regular languages. The concatenation of two infinite languages is not well-defined.

Lemma. If $R \subseteq \Sigma^*$ is regular and $L \subseteq \Sigma^\omega$ is Büchi-recognizable, then the ω-language RL is Büchi-recognizable.

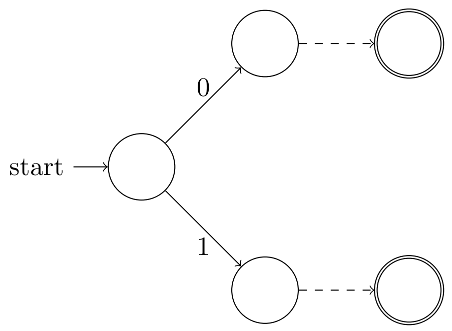
Proof Idea. Let D be a DFA that recognizes R, and M be an NBA that recognizes L. We can simulate D until a final state is visited and then draw nondeterministic arrows to the initial state of M to begin the M simulation. This produces an ϵ-NBA, which can be easily converted to an NBA.

Now, let's look at ω-iteration, equivalent to the Kleene star operation in regular languages.

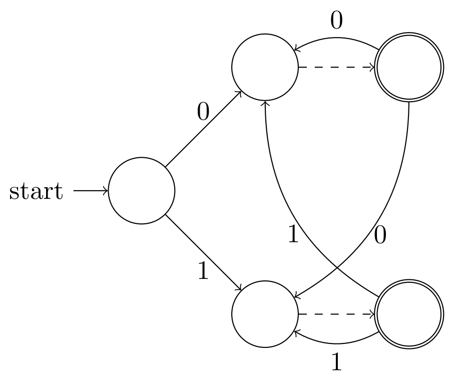
Lemma. If R is regular, then the ω-language R^ω is Büchi-recognizable.

Proof Idea. Let M be a DFA that recognizes R. Add nondeterministic arrows from every final state to the starting state. Since the set of final states is finite, Any language in the form R^ω will visit at least one final state infinitely often.

$M =$



$M' =$



Theorem. (Büchi's Theorem) A language $L \subseteq \Sigma^*$ is Büchi recognizable if and only if L is a finite union of sets RS^ω, where R, S are regular languages.

Proof. $\Rightarrow$: A Büchi automaton accepts a string if a final state $q_f \in F$ is visited infinitely often. There are two stages in its run:

1. The run goes from q_0 to q_f.
2. The run loops around q_f infinitely often.

In a finite automaton, the language of all finite strings that bring a state to another state is regular.

Formally, for a Büchi automaton $M = (Q, \Sigma, \delta, q_0, F)$,

$$L(M) = \bigcup_{q_f \in F} R_{q_0,q_f} \left(R_{q_f,q_f}\right)^\omega,$$

where $R_{q_a,q_b} = L((Q, \Sigma, \delta, q_a, q_b))$, all strings that take M from q_a to q_b.

Every accepted string will go over these two stages on some final state. Since the number of final states is finite, we only need a finite union.

$\Leftarrow$: We have constructed Büchi automata to recognize the finite union, concatenation, and ω-iteration of Büchi-recognizable languages. $\square$

Therefore, if an ω-language cannot be written as a finite union of RS^ω, where R, S are regular, it is not Büchi recognizable.

For example, the language $(0^n 1^n)^\omega, n > 0$ is not Büchi-recognizable. We can use the pumping lemma to show the looping branch is not possible with finitely many states.

Theorem. Büchi recognizable languages are closed under complement.

The proof is omitted here. When complementing an NFA, we first converted the NFA to a DFA and then flipped the final states. However, an NBA cannot be determinized, as we shall see later. Even for a DBA, flipping the final states does not produce the complement. An accepting run can still visit a non-final state infinitely often.

The idea is to convert a Büchi automaton M to an equivalent deterministic Rabin automaton, which is easy to complement, and then convert it back to Büchi.

5 Deterministic Büchi Automata

Unfortunately, deterministic Büchi automata (DBA) are less expressive than NBA.

Proposition. No DBA can recognize the language

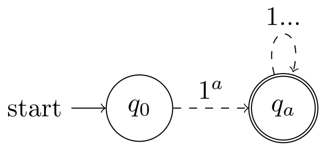
$$L = (0+1)^*1^\omega$$

(a string with only finitely many 0s), but some NBA can.

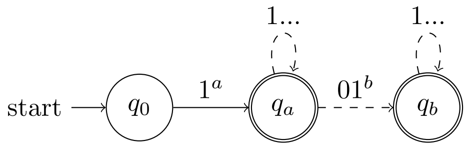
Proof. L is in the form $L = RS^\omega$ where $R = (0+1)^*$, $S = 1$ are regular. By Büchi's theorem, L can be recognized by some nondeterministic Büchi automaton.

Assume some DBA D also recognizes L.

Let $s_0 = 1^\omega$. $s_0 \in L$ because it has no 0s, and D accepts it. The run first reads some prefix $1^a, a \geq 0$ of 1^ω and transitions to a final state q_a. Next, it reads blocks of 1s and loops around q_a.

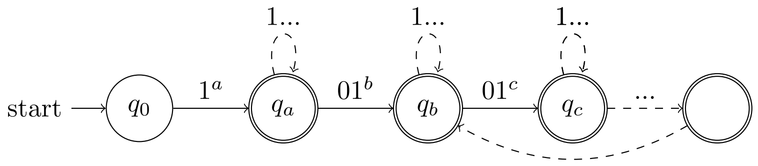


Now add a 0 after the 1^a block. Let $s_1 = 1^a01^\omega$. Again, the run consists of a prefix and loop stage. If 0 is in the loop stage, D can accept strings with infinitely many 0s. Therefore, the added 0 must be in the prefix stage. Due to the determinism, D will visit the final state q_1 after reading 1^a. Next, it will read 01^b for some $b \geq 0$, transition to a final state q_b, and loop there.



Repeat the construction above. Because D is deterministic, it will always visit a final state after reading a block of 1s. The 1-block may be empty. In this case, D transitions to another final state after reading the added 0. $s_2 = 1^a01^b01^\omega \in L$.

Repeat this process infinitely and feed the string $s_\omega = 1^a01^b01^c0...$ to D. Again, a final state will be visited after reading every block of 1. Because the set of final states is finite, at least one final state must be visited infinitely often. D will accept s_ω.



However, $s_\omega \notin L$ because it has infinitely many 0s, leading to a contradiction. ☐

6 Applications and Limitations

ω-automata provide a finite way to represent and abstract infinite runs.

ω-automata are helpful in finding the long-term behaviors of systems that are not expected to terminate, such as hardware, operating systems, and control systems. People want to know on an input if an acceptance eventually follows.

Omega automata is also used to prove the decidability of some problems in mathematical logic.

However, Büchi automata lack some nice properties of DFAs or Turing machines. Its deterministic and nondeterministic versions do not have the same expressive power. As a result, researchers explore more complicated accepting conditions but with nicer properties.

References

[1] Meghyn Bienvenu. Automata on infinite words and trees, Jan 2010.

[2] Javier Esparza. Automata theory, Aug 2017.

[3] K Narayan Kumar. Büchi automata, 2006.

[4] Amaldev Manuel and R. Ramanujam. Automata over infinite alphabets, Jul 2009.

[5] Guillaume Sadegh. Complementing büchi automata, May 2009.

[6] Michael Sipser. *Introduction to the theory of Computation.* Cengage Learning, 2021.

[7] Ivan Zuzak. Fsm2regex.