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 \(\varphi\), we can construct a finite automaton \(M_\varphi\) such that the words accepted by \(M_\varphi\) are exactly the encodings of assignments under which \(\varphi\) 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 restricted 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 \(\Sigma\) 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 \(\Sigma\) is \(\alpha : \mathbb{N} \rightarrow \Sigma\). The \(i\)th symbol of string \(\alpha\) 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 \(\alpha[i, \ast]\). The collection of infinite length strings over \(\Sigma\) will be denoted as \(\Sigma^\omega\).

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 \(a\)s.](image-url)
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. Unlike 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 $a$s.

Thus, $L_\exists B(M_1) = \{ \alpha \in \{a,b\}^\omega \mid \alpha \text{ has infinitely many } a\}$.

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
- $\Sigma$ 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 $\rho$ is accepting if $\rho[i] \in F$ for infinitely many $i$.

An input $\alpha$ is said to be accepted by $M$ if $M$ has an accepting run on $\alpha$.

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

A language $A \subseteq \Sigma^\omega$ is said to be Büchi recognizable if there is a Büchi automaton $M$ such that $A = L_\exists B(M)$.

Let us look at some more examples.

Example 5. Consider the automaton shown in Figure 2. On any input string $\alpha$, the automaton $M_2$ has a run which stays in state $q_0$, i.e., the run $q_0^\omega = q_0q_0\ldots$. 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 $\alpha$ that has finitely many $a$s. The converse is also true — if $\alpha$ has only finitely many $a$s then there is an accepting run that transitions from $q_0$ to $q_1$ on the first $b$ after the last $a$ in $\alpha$, and then it stays in $q_1$. Thus, we have

$$L_\exists B(M_2) = \{ \alpha \in \{a,b\}^\omega \mid \alpha \text{ has finitely many } a\} = L_\exists B(M_1)$$

The automaton $M_2$ is nondeterministic. Is there a deterministic Büchi automaton recognizing this language?
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 $q_1$.

$$L_{\exists\forall}(M_3) = \{ \alpha \in \{a,b\}^\omega \mid \alpha \text{ has infinitely many } as \text{ and } bs \}$$

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 observation 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 $U = L(M)$ (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 $\alpha$ and since $\alpha$ 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.

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

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

Proof. Suppose (for contradiction) $A$ is recognized by a deterministic Büchi automota. 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 an $n_2$ 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 $\alpha \not\in A$, which contradicts the fact that $A = \lim(U)$. 

$\Box$
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 $A = \{ \alpha \in \{a,b\}^\omega \mid \alpha \text{ has infinitely many } a \}$ is recognized by deterministic Büchi automaton $M_1$ shown in Figure 1. However, by Proposition 9, its complement $\overline{A} = \{ \alpha \in \{a,b\}^\omega \mid \alpha \text{ has finitely many } a \}$ 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 recursively enumerable languages) is to exploit nondeterminism — given an input string $\alpha$, 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 $\alpha$, the problem $A_1 \cap A_2$ requires us to determine if $\alpha$ 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 $\alpha$ 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 $A_1 \cap A_2$. (**add example**)

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 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

$$\delta((p_1, p_2, i), a) = \begin{cases} 
\delta_1(p_1, a) \times \delta_2(p_2, a) \times \{i\} & \text{if } p_i \notin F_i \\
\delta_1(p_1, a) \times \delta_2(p_2, a) \times \{\bar{i}\} & \text{if } p_i \in F_i
\end{cases}$$

where $\bar{i} = i + 1$ if $i = 1$ and $\bar{i} = i - 1$ if $i = 2$.

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

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

**Definition 14.** Let $\Sigma_1$ and $\Sigma_2$ be finite alphabets such that $|\Sigma_2| \leq |\Sigma_1|$. A projection $\pi : \Sigma_1 \rightarrow \Sigma_2$ is an onto function.

A projection $\pi : \Sigma_1 \rightarrow \Sigma_2$ 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 15.** If $A \subseteq \Sigma_1^\omega$ is Büchi recognizable and $\pi : \Sigma_1 \rightarrow \Sigma_2$ 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

$$\pi(\delta)(q, b) = \bigcup_{a \in \Sigma^*: \pi(a) = b} \delta(q, a)$$

$\Box$

### 1.3 omega-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 $\omega$-regular languages (to distinguish them from the notion of regularity over finite words), and the notion coincides with Büchi recognizability. Before defining $\omega$-regularity, we need to introduce the operation analogous to Kleene-closure, that constructs languages of infinite strings.

**Definition 16.** For a nonempty subset $U \subseteq \Sigma^*$ with $U \neq \{\varepsilon\}$, the $\omega$-iteration of $U$ is the set $U^\omega = \{u_1 u_2 \cdots \in \Sigma^* \mid u_i \in U \setminus \{\varepsilon\}\}$.

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, $\omega$-regular languages are generated using union, concatenation and $\omega$-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 17.** $A \subseteq \Sigma^\omega$ is said to be $\omega$-regular if $A = \bigcup_{i=1}^n U_i V_i^\omega$, where $U_i, V_i$ are regular (finite word) languages.
The notion of $\omega$-regularity coincides with Büchi recognizability.

**Theorem 18.** $A$ is $\omega$-regular if and only if $A$ is Büchi recognizable.

*Proof.* ($\Rightarrow$) Consider $A = \bigcup_{i=1}^{n} U_i V_i^\omega$, 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 $U^\omega$ 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 $U^\omega$ 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 $N = (Q', \Sigma, \delta', q_0, \{q_\ast\})$, where

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

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 generality 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,

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

$\square$