Towards Infinity

A Challenge: Does it make any sense to consider finite state machines on infinite words?

If so, how would this generalization work?

Infinite Words

As a matter of principle, infinite words come in two flavors: bi-infinite

\[ \Sigma^\infty = \mathbb{Z} \rightarrow \Sigma \]

or one-way infinite

\[ \Sigma^\omega = \mathbb{N} \rightarrow \Sigma \]

Both kinds appear naturally in the analysis of symbolic dynamical systems (reversible and irreversible).

One-way infinite ones can be used to describe the properties of programs that never halt, such as operating systems and user interfaces. Protocols also naturally give rise to infinite descriptions.

Adieu Concatenation

Note that neither \( \Sigma^\infty \) nor \( \Sigma^\omega \) form a semigroup under concatenation in any conceivable sense of the word: there is no way to combine two infinite words by “placing one after the other” and get another infinite word (at least not of the kind that we are interested).

But not that there is an obvious concatenation operation

\[ \Sigma^* \times \Sigma^\omega \rightarrow \Sigma^\omega \]

and a slightly less obvious one of type

\[ \Sigma^\omega \times \Sigma^\omega \rightarrow \Sigma^\infty \]

The second one is particularly interesting in conjunction with automata on bi-infinite words, but we won’t go there: the technical details are too messy.

No Pattern Matching

The standard acceptance testing problem makes little sense in this setting.

Problem: Acceptance
Instance: An \( \omega \)-automaton \( A \) and a word \( x \in \Sigma^\omega \).
Question: Does \( A \) accept input \( x \)?

Presumably \( A \) will just be some finite data structure. But there is no general way to specify the input \( x \), the space \( \Sigma^\omega \) is uncountable. We could consider periodic words or the like, but as stated the decision problem is basically meaningless.

But: We might still be able to generalize the logic approach and use these, yet undefined, \( \omega \)-automata to solve decision problems for appropriate logics.
Key Question: How do we modify finite state machines to cope with infinite inputs?

- Transition system: same as for ordinary finite state machines.
- Acceptance condition: requires work.

What kind of acceptance condition might make sense? For finite words there is a natural answer based on path existence, but for infinite words things become a bit more complicated.

Whatever condition we choose, we should not worry about actual acceptance testing, this is a conceptual problem, not an algorithmic one.

Recall: B"uchi Automata

A B"uchi automaton $B$ is a transition system $(Q, \Sigma, \tau)$ together with an acceptance condition $I \subseteq Q$ and $F \subseteq Q$.

$B$ accepts an infinite word $x \in \Sigma^\omega$ if there is a run $\pi$ of $B$ on $x$ that starts at $I$ and such that $\text{rec}(\pi) \cap F \neq \emptyset$.

The collection of all such words is the acceptance language of $B$. A language $L \subseteq \Sigma^\omega$ is recognizable or $\omega$-regular if there is some B"uchi automaton that accepts it.

So far, this is just a definition. It seems reasonable, but it is absolutely not clear at this point that we will get any mileage out of this.

Example I

Let $\Sigma = \{a, b\}$ and

$L = \{ x \in \{a, b\}^\omega \mid 1 \leq \#_ax < \infty \}$

So $L$ is the language of all words containing at least one, but only finitely many $b$'s. This language is recognizable.

In fact, two states suffice. Here is a B"uchi automaton for $L = \{a, b\}^*ba^\omega$: 

![B"uchi Automaton Diagram]

We will usually drop the $\omega$ whenever it is obvious from context.

Acceptance

Again: Key Question: What could it possibly mean for a finite state machine to accept an infinite word?

Obviously we need some notion of acceptance that does not just depend on a finite initial segment of the input: this would ignore "most" of the input and just replay our old theory.

On the other hand, we should keep things simple and not use wildly infinitary conditions to determine acceptance; we don’t want to sink in a morass of descriptive set theory.

So here is a fairly natural condition: let’s insist that an accepting run must touch the set of final states infinitely often. More precisely, define the set of recurrent states of a run $\pi$ to be

$$\text{rec}(\pi) = \{ p \in Q \mid \exists i (p_i = p) \}$$

Thus for a B"uchi automaton $B$ we have

$$L = \{ x \in \Sigma^\omega \mid \exists \pi (x) \in \text{rec}(\pi) \}$$

Runs and Traces

Since we will not change the underlying transition systems, we can lift the definitions of run and trace to the infinite case: a run is an alternating infinite sequence

$$\pi = p_0, a_1, p_1, a_2, \ldots , p_{m-1}, a_m, p_m, \ldots$$

The corresponding infinite sequence of symbols is the trace:

$$\text{lab}(\pi) = a_1, a_2, \ldots , a_{m-1}, a_m, \ldots \in \Sigma^\omega$$

In general, the number of runs on a particular input is going to be uncountable, but that will not affect us (it is path existence that matters).

ω-Languages

So we are interested in one-way infinite words:

$$\Sigma^\omega = N \to \Sigma$$

One-way infinite words are often called $\omega$-words.

Subsets of $\Sigma^\omega$ are called $\omega$-languages.

Given an automaton for infinite words its acceptance language is denoted by $L^\omega(A)$ and so on.

Note that an $\omega$-language may well be uncountable; there cannot be a good notation system for $\omega$-words.

We will usually drop the $\omega$ whenever it is obvious from context.

ω-Languages

So we are interested in one-way infinite words:

$$\Sigma^\omega = N \to \Sigma$$

One-way infinite words are often called $\omega$-words.

Subsets of $\Sigma^\omega$ are called $\omega$-languages.

Given an automaton for infinite words its acceptance language is denoted by $L^\omega(A)$ and so on.

Note that an $\omega$-language may well be uncountable; there cannot be a good notation system for $\omega$-words.

We will usually drop the $\omega$ whenever it is obvious from context.

ω-Languages

So we are interested in one-way infinite words:

$$\Sigma^\omega = N \to \Sigma$$

One-way infinite words are often called $\omega$-words.

Subsets of $\Sigma^\omega$ are called $\omega$-languages.

Given an automaton for infinite words its acceptance language is denoted by $L^\omega(A)$ and so on.

Note that an $\omega$-language may well be uncountable; there cannot be a good notation system for $\omega$-words.

We will usually drop the $\omega$ whenever it is obvious from context.
Let $\Sigma = \{a, b, c\}$ and
$$L = \{x \in \{a, b, c\}^\omega \mid \#_ax = \#_bx = \infty \land \#_cx < \infty\}.$$ 
So $L$ contains finitely many $c$'s, but infinitely many $a$'s and $b$'s. This language is also recognizable.

Consider alphabet $\Sigma = \{a, b, c\}$. Let $L$ be the language

Every $a$ is ultimately followed by a $b$, though there may be arbitrarily man $c$'s in between, and there may be only finitely many $a$'s.

Then $L$ is recognizable.

It is clear that a Büchi automaton may have useless states. In particular, any inaccessible state (in the sense of a classical finite state machine) is clearly useless. But there is more: for example, if a final state belongs to a trivial strongly connected component it is useless: the computation can pass through the state at most once, so we might as well remove it from the set of final states.

Lemma
Useless states can be removed from a Büchi automaton in linear time.

Exercise
Give a careful definition of what it means for a state in a Büchi automaton to be useless. Then produce a linear time algorithm to eliminate useless states.

Lemma
It is undecidable if a Büchi automaton accepts a computable word.

Proof.
For any index $e$ define a computable infinite word $U_e$ by
$$U_e(s) = \begin{cases} b & \text{if } \{e\}_e(s) \text{ converges (after at most } e \text{ steps)}, \\ a & \text{otherwise}. \end{cases}$$

Then $\{e\}_e(s)$ converges iff $U_e \in a^+b^*$, which property is easily checked by a 2-state Büchi automaton. If we could test acceptance of a computable word in a Büchi automaton we could thus solve the Halting Problem.
Union and Intersection

Lemma
Recognizable languages of $\Sigma^\omega$ are closed under union and intersection.

Proof. For union simply use the disjoint sum of the Büchi automata. For intersection, we use a slightly modified product machine construction. The new state set is $Q_1 \times Q_2 \times \{0, 1, 2\}$
The transitions on $Q_1$ and $Q_2$ are inherited from the two given machines. On the last component act as follows:
- Move from 0 to 1 at the next input.
- From 1 move to 2 whenever a state in $F_1$ is encountered.
- Reset from 2 to 0 when a state in $F_2$ is encountered.

Rational Languages

Another piece of evidence for the usefulness our our definition is that recognizable language on infinite words can be written down as a type of regular expression.

Definition
A language $L \subseteq \Sigma^\omega$ is rational if it is of the form
$$L = \bigcup_{i \in \mathbb{N}} U_i V_i^\omega$$
where $U_i, V_i \subseteq \Sigma^*$ are all regular.

Since we already have a notation system for regular languages of finite words it is easy to obtain a notation system for recognizable languages of $\omega$-words: add one operation $\omega$ with the understanding that this operation can only be used once and on the right hand side.

Exercise
Fill in the details in the last construction.

Rational Expressions

A basic expression has the form
$$\alpha \beta^\omega$$
where $\alpha$ and $\beta$ are ordinary regular expressions. As we will see shortly, sums of these expressions then produce exactly all the recognizable $\omega$-languages.

For the examples from above we have fairly simple expressions
$$a^*b(a + b)^*a^\omega$$
$$((b + c)^*(c + ac^*b))^\omega$$

Equivalence Recognizable and Rational

Lemma
An $\omega$-language is recognizable if, and only if, it is rational.

Proof. First assume $A$ is a Büchi automaton accepting some language $L$. For each final state $p$ define two new automata
$$A_p^0 = A(I, p) \quad A_p^1 = A(p, p)$$
and let $U_p = \mathcal{L}(A_p^0)$, $V_p = \mathcal{L}(A_p^1)$. Then
$$L = \bigcup_{p \in Q} U_p V_p^\omega$$
since recognizability implies that one particular state $p \in F$ must appear infinitely often.

Proof, contd.

The initial states are of the form
$$I_1 \times I_2 \times \{0\}$$
and the final states are
$$F = Q_1 \times Q_2 \times \{0\}$$
The infinitely many visits to $F$ imply infinitely many visits to $F_1$ and $F_2$, and conversely.

There is a message here: though the construction is similar to the finite case it is a bit more complicated. You have to stay alert. Also note that we have not dealt with complements.

Exercise
Fill in the details in the last construction.
Deterministic Büchi Automata

Definition
A Büchi automaton is deterministic if it has one initial state and its transition system is deterministic.

We may safely assume that a deterministic Büchi automaton is also complete: otherwise we can simply add one sink state. In a deterministic and complete Büchi automaton there is exactly one run from the initial state for any input.

Note that the undecidability result for computable words holds already for deterministic Büchi automata.

Still, deterministic automata should be of interest if one tries to compute complements: to get a machine for the complement, manipulate the acceptance condition.

Another Catastrophe

Proposition
Let $L = \{ x \in \{a,b\}^\omega \mid \#x < \infty \}$. Then $L$ is recognizable but cannot be accepted by any deterministic Büchi automaton.

Proof. To see this, suppose there is some deterministic Büchi automaton that accepts $L$.

Hence, for some $n_1$, $\delta(q_0, ba^{n_1}) \in F$. Moreover, for some $n_2$, $\delta(q_0, ba^{n_1}ba^{n_2}) \in F$. By induction we produce an infinite word $ba^{n_1}ba^{n_2}ba^{n_3} \ldots$ accepted by the automaton. Contradiction.

This shows that a deterministic transition system together with a Büchi type acceptance condition $\text{rec}(\pi) \cap F \neq \emptyset$ is not going to work: not only do we have to construct a deterministic transition system, we also have to modify our acceptance conditions. Alas, it is far from clear how one should do this.

Deterministic Recognizable Languages

One might wonder whether the languages recognized by deterministic Büchi automata have some natural characterization. Since you asked ...

Definition
Let $L \subseteq \Sigma^*$ be a language. Define the adherence of $L$ to be

$$\bar{L} = \{ x \in \Sigma^* \mid x \text{ has infinitely many prefixes in } L \}$$

The best way to visualize this to think of $\Sigma^*$ as an infinite tree. Mark the nodes in this tree that belong to $L$. Then $\bar{L}$ is the set of all branches in the tree that touch infinitely many marked nodes. Note that there may be no such branches even when $L$ is infinite.

$$ \begin{array}{c|c|c}
L & (ab)^* & (ab)^{\omega} \\
(a^*b)^* & (ab)^* & (a^*b)^{\omega} \\
\end{array} $$

The Characterization

Lemma
An $\omega$-language $L$ is recognized by a deterministic Büchi automaton if, and only if, $L$ is the adherence of some regular language.

Proof
We already know that $L^\omega(A) \subseteq \overline{L^\omega(A)}$ for any Büchi automaton $A$.

Now suppose $A$ is in addition deterministic. Then equality holds and we are done.

For the opposite direction assume $L = \overline{K}$ where $K \subseteq \Sigma^*$ is regular. Then $K$ is accepted by a deterministic finite state machine, for example, the minimal automaton $A$ for $K$.

Thinking of $A$ as a Büchi automaton, we have $L^\omega(A) = L$. □
Consider again the automaton

![Automaton Diagram]

Note that $(abb)^* \subseteq \mathcal{L}^*(A)$.
But then $(abb)^\omega \in \mathcal{L}^*(A) - \mathcal{L}(A)\omega$.

The problem is that our finite computations do not have infinite extensions.

**Lemma**

Deterministic recognizable languages are closed under union and intersection.

**Proof.**

Union follows directly from the last lemma and $A \cup B = \overline{A \cup \overline{B}}$.
For intersections note that the modified product machine construction from above preserves determinism.

**Exercise**

Check all the details in the last proof.

---

**Other ω-Automata**

As we have seen, there are recognizable ω-languages that cannot be accepted by any deterministic Büchi automaton. Alas, without determinization it is unclear how we could deal with complements, which we need to handle negation.

So, we need to find alternative machine models that allow for deterministic descriptions of recognizable languages.

As before, we will not change the transition system, just the acceptance condition.

One fairly natural possibility is to completely pin down $\text{rec}(\pi)$.

**Definition**

A Muller automaton consists of a deterministic transition system $(Q, \Sigma, \tau)$ and an acceptance condition $q_0 \in Q$ and $\mathcal{F} \subseteq \wp(Q)$.

$\mathcal{A}$ accepts an infinite word $x \in \Sigma^\omega$ if there is a run $\pi$ of $\mathcal{A}$ on $x$ that starts at $q_0$ and such that $\text{rec}(\pi) \in \mathcal{F}$.

$\mathcal{F}$ is often referred to as the table of the Muller automaton. Note that the table may have size exponential in the size of the transition system.

But, complementation is easy (just as it was easy for DFAs): make sure the machine is complete, then replace the old table $\mathcal{F}$ by $\wp(Q) - \mathcal{F}$.

Note that this simple operation might produce an exponential blow-up if done in a ham-fisted way.

The at-least-one-but-finitely-many-$b$’s language from above is accepted by the following Muller automaton.

The table has the form $\mathcal{F} = ((2), (3))$.

![Muller Automaton Diagram]

Note that this automaton distinguishes between an even and odd number of $b$’s. This is a bit scary since the distinction is by no means obvious from the original Büchi automaton.
The Complement

We can complement the table to get a machine for the complement of the language.

\[
\begin{array}{ccc|ccc}
1 & 2 & 3 & 1 & 2 & 3 \\
\hline
a & a & a & 0 & 0 & \emptyset \\
\end{array}
\]

The complement table contains several useless entries (other than \(0\)):

\[
\begin{array}{ccc|ccc}
1 & 2 & 3 & 1 & 2 & 3 \\
\hline
a^* & \emptyset & \emptyset & a^*(ba^*)^* & 0 & \emptyset \\
\end{array}
\]

However, the two non-empty entries duly produce no \(b\)'s or infinitely many \(b\)'s, exactly the complement of the language.

Muller versus Deterministic Büchi

**Lemma**

A language \(L \subseteq \Sigma^\omega\) is recognizable by a Muller automaton if, and only if, it is of the form

\[
L = \bigcup_{i \leq n} U_i - V_i
\]

where \(U_i, V_i \subseteq \Sigma^\omega\) are recognizable by a deterministic Büchi automaton. In other words, \(L\) must lie in the Boolean algebra generated by deterministic Büchi languages.

**Proof.**

Suppose \(L\) is recognized by \(A\) with table \(F\). Since \(L = \bigcup_{F \in \mathcal{F}} \mathcal{L}(A(F))\) we only need to deal with tables of size 1. But

\[
\mathcal{L}(A(F)) = \bigcap_{p \in F} \mathcal{L}(A(p)) - \bigcup_{q \in \mathcal{F}} \mathcal{L}(A(q))
\]

where the automata on the right hand side are deterministic Büchi. Done by the closure properties of deterministic Büchi languages.

Rabin Automata

Another possibility to modify acceptance conditions is to augment the positive condition of Büchi automata by a negative condition: a successful run must ultimately avoid a certain set of states.

**Definition**

A Rabin automaton consists of a deterministic transition system \((Q, \Sigma, \tau)\) and an acceptance condition \(q_0 \in Q\) and \(R \subseteq \mathcal{P}(Q) \times \mathcal{P}(Q)\).

\(A\) accepts an infinite word \(x \in \Sigma^\omega\) if there is a run \(\pi\) of \(A\) on \(x\) that starts at \(q_0\) and such that for some \((L, R) \in \mathcal{R}\): \(rec(\pi) \cap L = \emptyset\) and \(rec(\pi) \cap R \neq \emptyset\).

The pairs \((L, R)\) are called Rabin pairs: \(L\) is the negative condition and \(R\) the positive condition.

In the special case where \(\mathcal{R} = \{(\emptyset, F)\}\) we are dealing with a deterministic Büchi automaton.

Example: Rabin

The Muller automaton from above can also be turned into a Rabin automaton with Rabin pairs

\[
\mathcal{R} = \{(1, 2; 3), (1, 3; 2)\}
\]

The excluded sets force a tail end of the run to look like \(2^\omega\) or \(3^\omega\).
The acceptance condition for all three has the form
- initial states $I$ (a singleton for Muller and Rabin)
- a family $F \subseteq \mathcal{P}(Q)$ of permissible values for the recurrent state set of a run.

Note that we may safely assume that $F$ contains only strongly connected sets.

For Büchi automata the family is trivial: $F = \{ F \}$ and thus a data structure of size $O(n)$.

For Muller automata it is explicitly specified and potentially large.

For Rabin automata the specification is implicit: all $X \subseteq Q$ such that $\exists (L, R) \in R (X \cap L = \emptyset, X \cap R \neq \emptyset)$. Each Rabin pair is $O(n)$, but there may be exponentially many.

The opposite direction is harder.

Let $\langle Q, \Sigma, \delta; F \rangle$ be a Muller automaton, say, $F = \{F_1, \ldots, F_k\}$. Consider a new transition system on state set

$$Q' = \mathcal{P}(F_1) \times \cdots \times \mathcal{P}(F_k) \times Q$$

and transitions

$$(U_1, \ldots, U_k, p) \xrightarrow{a} (U_1', \ldots, U_k', \delta(p, a))$$

where $U_i' = \emptyset$ if $U_i = F_i$, and $F_i \cap (U_i \cup \{\delta(p, a)\})$ otherwise. The Rabin pairs are defined by

$$L_i = \{ (U_1, \ldots, U_k, p) \mid p \notin F_i \} \quad R_i = \{ (U_1, \ldots, U_k, p) \mid U_i = F_i \}$$

One can verify that the new machine is equivalent to the given Muller automaton. \qed

The reason these equivalences are so important is the following theorem (which we are not going to prove here).

Theorem (Safra 1988)

There is an algorithm to convert a Büchi automaton into an equivalent Rabin (or Muller) automaton.

The algorithm has running time $2^{O(n \log n)}$

Unfortunately, this is optimal: there are examples where the deterministic Büchi, Büchi, Muller and Rabin are all equivalent and strictly stronger than deterministic Büchi. This result is similar to equivalences between various types of automata on finite strings, but the arguments are more complicated.

We will not consider nondeterministic versions of Muller and Rabin automata here.

So now we have four classes of automata:
- deterministic Büchi,
- Büchi,
- Muller and
- Rabin.

We will see that Büchi, Muller and Rabin are all equivalent and strictly stronger than deterministic Büchi. This result is similar to equivalences between various types of automata on finite strings, but the arguments are more complicated.

We will not consider nondeterministic versions of Muller and Rabin automata here.
Consider a Muller automaton with a singleton table, $\mathcal{F} = \{F\}$.

Here is a construction of a Büchi automaton that avoids powersets.

Let $F = \{q_1, \ldots, q_{n-1}\}$.

We construct a nondeterministic Büchi automaton on states $Q' = Q \cup (F \times [n])_0$ where $[n]_0 = \{0, 1, \ldots, n-1\}$.

Let $q = \delta(p, a)$ and set

\[
\tau(p, a) = \begin{cases} 
(q) & \text{if } q \notin F, \\
(q', (q, 0)) & \text{otherwise.}
\end{cases}
\]

\[
\tau((p, k), a) = \begin{cases} 
(q, k) & \text{if } p = q_k, k > 0, \\
(q, k + 1 \mod n) & \text{if } p = q_k, k > 0, \\
(q, 1) & \text{if } k = 0.
\end{cases}
\]

$F' = Q \times \{0\}$.

In the conversion Muller to Rabin the transition system is unchanged. However, we introduce exponentially many entries in the table, for each Rabin pair.

For the direction Rabin to Muller the new transition system already has potentially exponential size in the size of the old transition system ($k$ can be exponential in the number of states). The number of pairs is also exponential, and each pair has perhaps exponential size.

A Rabin automaton with $n$ states and $m$ pairs can be simulated by a Büchi automaton of size $O(nm)$.

Overall, these conversions will only be feasible for rather small machines.