Exercise

Can either of the following languages (over alphabet \{a, b\}): be accepted by DFAs?
- \(a^n b^m\) for any \(n, m \in \mathbb{N}\) (i.e. words such as \(\epsilon\), \(aaaabb\), \(abbbb\), \(bbb\))
- \(a^n b^n\) for any \(n \in \mathbb{N}\) (i.e. words such as \(\epsilon\), \(ab\), \(aabb\), \(aaabbb\), \ldots)

We can argue that the second of these languages is not accepted by any DFA.

General idea: finite number of states means finite memory. As an input string is scanned from left to right, DFA needs to count number of initial \(a\)'s occurring before first \(b\). But any DFA can only store a limited number of distinct numbers (at most the number of states, in fact). With no limit on length of possible input strings, any DFA will eventually be defeated. (See handouts for a more rigorous proof by contradiction.)

The Pumping Lemma

The following result gives a property of languages that are recognised by deterministic finite automata. It can often be used to show that certain languages are not accepted by DFAs.

Pumping Lemma

Let \(L\) be a language which is accepted by some deterministic finite automaton with \(N\) states. Then every word \(z \in L\) with \(|z| \geq N\) can be written in the form \(z = uvw\) for some \(u, v, w \in A^*\) with
1. \(v \neq \epsilon\)
2. \(|uv| \leq N\)
3. \(uv^n w \in L\) for \(n = 1, 2, 3, \ldots\)

Proof of Pumping Lemma (sketch)

Suppose a DFA has \(N\) states. If the DFA scans a word with more than \(N\) letters, it must enter some state on two (or more) distinct letters.

Consider the section of that word which lies between the two places where the same state is visited. What happens if instead of one copy of that sub-word, we inserted multiple copies?

At the end of each copy, the DFA returns to the same state. Afterwards, things continue as if only one copy had been scanned.

So if the original word was accepted, so also would be words obtained by inserting multiple copies of that section.
Reminder: accepting/rejecting paths notation

If \( w = a_1a_2 \ldots a_n \in A^* \) and \( \phi(p, w) = q \) then there are states \( r_1, r_2, \ldots, r_{n-1} \) with
\[
\begin{align*}
    r_1 &= \phi(p, a_1), \\
    r_2 &= \phi(r_1, a_2), \\
    r_3 &= \phi(r_2, a_3), \\
    &\ldots \\
    q &= \phi(r_{n-1}, a_n).
\end{align*}
\]
We say that the states \( p \) and \( q \) are connected by a path with label \( w \) and write
\[
\begin{array}{c}
    p \xrightarrow{a_1} r_1 \xrightarrow{a_2} r_2 \xrightarrow{a_3} r_3 \xrightarrow{\ldots} \xrightarrow{a_{n-2}} r_{n-1} \xrightarrow{a_n} q
\end{array}
\]
or
\[
\begin{array}{c}
p \rightarrow \ldots \rightarrow q.
\end{array}
\]
In particular a word \( w \) is accepted by an automaton if and only if there is a path from the initial state to a terminal state with label \( w \) (which we call an accepting path).

Proof of Pumping Lemma

If a word \( z \) belongs to a language accepted by an automaton with \( N \) states and \( |z| \geq N \) then the accepting path for \( z \) must contain a repeated state and therefore will have the form
\[
\begin{align*}
    i &\xrightarrow{u} q \xrightarrow{v} q \xrightarrow{w} t
\end{align*}
\]
where \( i \) is the initial state, \( t \) is an accepting state and \( z = uvw \), where \( v \neq \epsilon \).

Now we can repeat the path labelled with \( v \) any number \( n \) of times and obtain an accepting path with label \( uv^n w \). Hence \( uv^n w \) is also accepted by the automaton. Also if \( q \) is taken to be the first repeated state then the path from \( i \) to the second \( q \) can contain at most \( N + 1 \) states, so its label \( uv \) has length at most \( N \).

The word \( uv^n w \) can be thought of as having been obtained by “pumping” the substring \( v \) of \( z = uvw \).

Problem. Use pumping lemma to show that no DFA can accept palindromes over an alphabet with \( \geq 2 \) letters.

Solution. Suppose for a contradiction that there is an \( N \)-state automaton that accepts palindromes. Let \( a, b \) be distinct letters in \( A \). The word \( a^{N+1}ba^{N+1} \) is a palindrome and has length \( \geq N \) so by the pumping lemma we can write
\[
a^{N+1}ba^{N+1} = uvw \text{ where } v \neq \epsilon \text{ and } |uv| \leq N
\]
and \( uv^n w \) is a palindrome for every \( n \geq 1 \). But now
\[
a^{N+1}ba^{N+1} = uvw \text{ where } uv \text{ is shorter than } a^{N+1}.
\]
Comparing the letters in these two words we see that \( uv \) must be a string of \( a \)'s and hence \( v \) must be a string of at least one \( a \). It now follows that \( uv^2 w = a^{N+r}ba^{N+1} \) for some \( r \geq 2 \). By the pumping lemma this word must belong to the language \( P \) of all palindromes, but is clearly not a palindrome: contradiction!
Summary

We have seen various languages which are recognised by DFAs, and some that are not.
Pumping Lemma gives a way of proving that some languages do not have accepting DFAs. (example: strings of 1’s whose length is a square number, ie 1, 1111, 111111111, 1111111111111111, etc.)
It doesn’t work for all languages not accepted by DFAs.

Example

“Strings of 1’s whose length is a square number” is not a regular language.

remarks: this language is a bit artificial, but it’s certainly a formal language. Would like to know what sort of mechanism can recognise/describe it.

If the language is regular, then for some large enough \( m \), \( 1^m \) is in the language, and some substring of length \( r \) can be ”pumped”...

That is, \( 1^{m+r} \), \( 1^{m+2r} \), \( 1^{m+3r} \) etc are members of the language.

This is impossible since the gap between 2 consecutive square numbers increases as the numbers increase, and so this sequence can’t just contain square numbers. Conclude the language is not regular.

Another example

Use pumping lemma to prove that the set of words over \( \{a,b\} \) having the same number of \( a \)’s as \( b \)’s is not regular.

If the language is regular, there exists some \( m \) such that all words longer than \( m \) have a sub-word within the first \( m \) symbols that can be ”pumped”...

Consider the word \( a^mb^m \), which is a member of the language. If we choose any subword to be pumped from amongst the first \( m \) symbols, we get a string of \( a \)'s. If we insert another copy of that subword, the new word is not a member of language.

Example of language not accepted by a DFA for which you cannot directly use pumping lemma to prove that it is not accepted by a DFA:

The set of words over the 0, 1 alphabet which either start with one 0 then have a square number of 1’s or start with at least two 0’s. ie. 01, 0111, 0111111111, 01111111111111111, etc. together with words such as 00, 0001, 001101, etc.

Every word in this language has the prefix 0 which will generate further words in the language if it is ”pumped”
Given a language $L$ that uses alphabet $A$, the complement of $L$ is defined to be all words over $A$ that are not members of $L$. The reverse of a language is the set of words which, if their letters were reversed, would give words in $L$.

1. Prove that if $L$ is a regular language then the complement of $L$ is also regular.
2. Prove that if $L$ is a regular language then so is the reverse of $L$.

(then, use that second fact to prove that the language of the previous slide is irregular!)