Limits of FSMs

In this lecture we explore the limits of finite state systems.

In the process we will show how to use proof to demarcate a boundary on the use of FSMs.

Palindromes

A palindrome is a word that reads the same forwards or backwards.
e.g. kayak, eye

Given alphabet $\Sigma = \{a,b\}$ palindromes in Σ^* include:
a, b, aa, bb, aba, bab, aababaa

<u>Theorem</u>: There is no FSM that can recognise the language of palindromes over $\Sigma = \{a,b\}$.

Palindromes Proof: Step 1

Theorem: There is no FSM that can recognise the language of palindromes over $\Sigma = \{a,b\}$

holds if we can show that it is contradictory to believe that there exists a FSM that can recognise the language of palindromes over $\Sigma = \{a,b\}$.

Since any FSM can be translated into an equivalent deterministic FSM, we shall prove that:

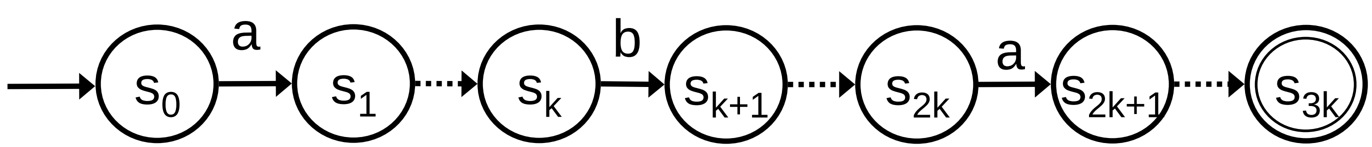
It is contradictory to believe that there exists a *deterministic* FSM that can recognise the language of palindromes over $\Sigma = \{a,b\}$.

Palindromes Proof: Step 2

Every FSM must have some finite number of states, k.

Suppose we have the palindrome $a^k b^k a^k$
(where a^k is the character a repeated k times)

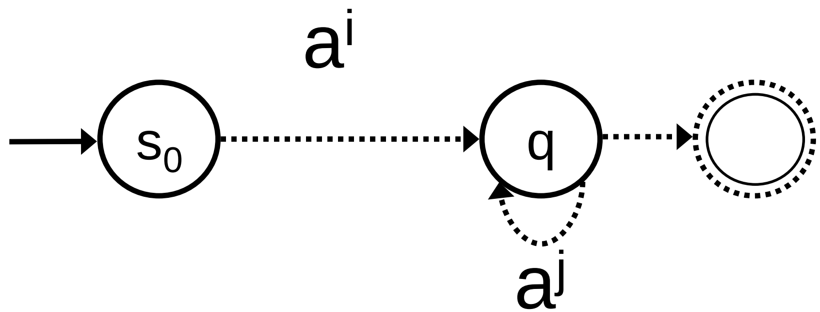
Any accepting trace for $a^k b^k a^k$ must look like this:



k+1 states for the first a^k
so we must visit the same state at least once
so there must be a loop in the FSM

Palindrome Proof: Step 3

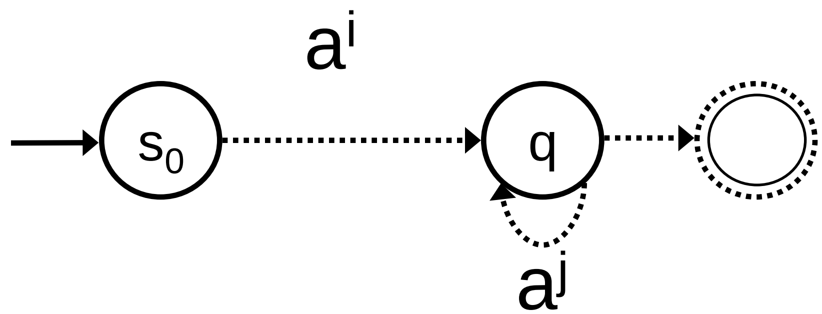
We must visit the same state more than once when reading the first a^k so we have a loop after some number (i) of a's at some state (q). The loop will consist of some number of a's (j).



Reading $a^{k-i}b^ka^k$ from state q reaches an accept state.

Palindrome Proof: Step 4

What happens if $a^{k+j}b^k a^k$ is given to this FSM?



- Read i a’s to get to state q. Leaves $a^{k+j-i}b^k a^k$
- Do one loop at q, reading j a’s. Leaves $a^{k-i}b^k a^k$
- But $a^{k-i}b^k a^k$ from state q reaches an accept state.

So a FSM that (correctly) accepts $a^k b^k a^k$ must also (incorrectly) accept $a^{k+j}b^k a^k$ for some j.

Palindrome Proof: Conclusion

- We have shown that a FSM that (correctly) accepts $a^k b^k a^k$ must also (incorrectly) accept $a^{k+j} b^k a^k$ for some j.
- Therefore it is contradictory to believe that there exists a *deterministic* FSM that can recognise the language of palindromes over $\Sigma = \{a,b\}$.
- Therefore there is no FSM that can recognise the language of palindromes over $\Sigma = \{a,b\}$.

General Rule of Thumb

- No FSM if we need to count up to an arbitrarily high value.
- This is because a FSM has no “memory” other than its current state, and it has a finite set of states.
- Examples:
  - $a^k b^k a^k$ is a palindrome but $a^n b^k a^k$ is not (if $n \neq k$)
  - $L = \{0^k 1^k \mid k \in \mathbb{N}\}$ has no FSM

Note that in both these cases the only way to ensure the later number is right is to remember the value of the earlier number.

Context Free Grammar

A context free grammar is a set of production rules that defines how strings in a language are generated from an initial start symbol, grounding in terminal symbols.

CFG production rules for palindromes:

$$S \rightarrow aSa$$
$$S \rightarrow bSb$$
$$S \rightarrow \varepsilon \mid a \mid b$$

CFG Accepting a Palindrome

1. $S \rightarrow aSa$
2. $S \rightarrow bSb$
3. $S \rightarrow \varepsilon \mid a \mid b$

$$
\begin{array}{c}
aababaa \\
\uparrow_3 \\
aabSbaa \\
\uparrow_2 \\
aaSaa \\
\uparrow_1 \\
aSa \\
\uparrow_1 \\
S
\end{array}
$$

Grammars for Regular Expressions

- Consider a language defined by a regular expression, say $ab^*a|ba^*$
- Can we define a context-free grammar for this language?

Grammars for Regular Expressions

- Consider a language defined by a regular expression, say $ab^*a|ba^*$
- Can we define a context-free grammar for this language?
- Let us try to do this part by part for the regular expression

Grammars for Regular Expressions

- Consider a language defined by a regular expression, say $ab^*|ba^*$
- Can we define a context-free grammar for this language?
- Let us try to do this part by part for the regular expression
- For a, CFG is: $S \rightarrow a$

Grammars for Regular Expressions

- Consider a language defined by a regular expression, say $ab^*|ba^*$
- Can we define a context-free grammar for this language?
- Let us try to do this part by part for the regular expression
- For a, CFG is: $S \rightarrow a$
- For b^*, CFG is: $S \rightarrow bS$; $S \rightarrow \varepsilon$

Grammars for Regular Expressions

- Consider a language defined by a regular expression, say $ab^*|ba^*$
- Can we define a context-free grammar for this language?
- Let us try to do this part by part for the regular expression
- For a, CFG is: $S \rightarrow a$
- For b^*, CFG is: $S \rightarrow bS$; $S \rightarrow \varepsilon$
- For ab^*, CFG is: $S \rightarrow S_1S_2$; $S_1 \rightarrow a$; $S_2 \rightarrow bS_2$; $S_2 \rightarrow \varepsilon$

Grammars for Regular Expressions

- Consider a language defined by a regular expression, say $ab^*|ba^*$
- For ab^*, CFG is: $S \rightarrow S_1S_2$; $S_1 \rightarrow a$; $S_2 \rightarrow bS_2$; $S_2 \rightarrow \varepsilon$
- For ba^*, CFG is: $S \rightarrow S_1S_2$; $S_1 \rightarrow b$; $S_2 \rightarrow aS_2$; $_S \rightarrow \varepsilon$
- For $ab^*|ba^*$, CFG is: $T \rightarrow S$; $T \rightarrow S'$; $S \rightarrow S_1S_2$; $S_1 \rightarrow a$; $S_2 \rightarrow bS_2$; $S' \rightarrow S_1'S_2'$; $S_1' \rightarrow b$; $S_2' \rightarrow aS_2'$; $S_2' \rightarrow \varepsilon$