Regular Languages

COMPSCI 3331

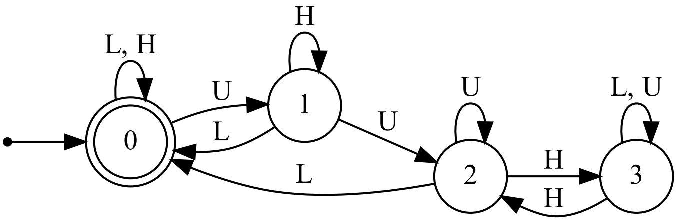
Outline

- Motivation for regular languages.
- Regular languages
- Deterministic finite automata.

Regular Languages - Motivation

- Languages recognized with a fixed amount of memory.
- Developed to model **how circuits work** and early **models of neural behaviour**.
- Used in text matching, regular expressions, compilers, model checking, protocol verification, natural language processing.

Example: Key Fob



Deterministic Finite Automaton

A deterministic finite automaton (DFA) consists of:

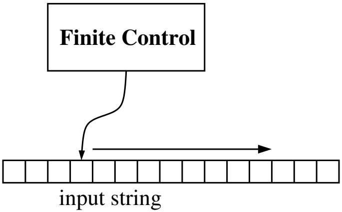
- a finite set of **states** the DFA can be in;
- an **alphabet**, specifying the set of letters the DFA can process;
- a **transition function**, which specifies how to update our state based on the current input letter.
- **start** and **final** states, which specify how to accept or reject words.

Deterministic Finite Automaton

- The alphabet in the fob example is `L,U,H`.
- Transition Function: what effect do actions from our alphabet have?
- Could give names to states: “hatch open”, “driver’s door unlocked”, etc.

Deterministic Finite Automaton

How do we visualize our DFA working?



“Finite control”: the current state and instructions provided by the transition function.

Deterministic Finite Automaton

Formally, a DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states,
- Σ is a finite alphabet.
- $\delta : Q \times \Sigma \to Q$ is the transition function.
- $q_0 \in Q$ is the start state.
- $F \subseteq Q$ is the final state.

Drawing DFAs

- Draw DFA with states, labelled transitions.
- Special indications for start and final states.

Accepting words and languages

- Each word $w \in \Sigma^*$ traces a path from the start state to some state in the automaton.
- If the state we reach is a final state ($\in F$) then the word w is **accepted** by M. Otherwise, it is rejected.
- The language accepted by a DFA M is the set of all strings accepted by M.

Acceptance: Formal Definition

- The transition function acts on letters from Σ.
- Extend it to work on **words** from Σ^* with a recursive definition:
  - $\delta(q, \varepsilon) = q$ for all $q \in Q$;.
  - $\delta(q, wa) = \delta(\delta(q, w), a)$ for all $q \in Q$, $w \in \Sigma^*$ and $a \in \Sigma$.
- A word $w \in \Sigma^*$ is accepted by a DFA $M = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, w) \in F$. The language accepted by M is

$$L(M) = \{w \in \Sigma^* \ : \ \delta(q_0, w) \in F\}.$$

Language accepted by a DFA

- Can ask "given a DFA M, what language does it accept?"
- Establish through proof: define a language L and establish that $L(M) \subseteq L$ and $L \subseteq L(M)$.

Finding a suitable DFA

- "For this language L, find a DFA M such that $L(M) = L$."
- **Assumption**: for the language L, **there exists** a DFA M such that $L(M) = L$.
- Tips:
  - Think of the states as "what do we need to remember?"
  - Define set of states, and then the letters that move us from state to state.
  - **Learn by doing!**

The Regular Languages

A language $L \subseteq \Sigma^*$ is a **regular language** if there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L$.

- Not every language is a regular language.

Are DFAs Unique?

If M_1, M_2 are (in some way) distinct DFAs, is it true that $L(M_1) \neq L(M_2)$?

- can always have superfluous unconnected states.
- can have different ways to define the language.

Return to Motivation

What can DFAs be used to model?

- Finite sets.
- Objects which only require a fixed amount of memory.
- “Easy” jobs in programming language recognition jobs.