Introduction to Theoretical Computer Science

Lecture 3: Beyond the Regular Languages

Dr. Liam O'Connor

University of Edinburgh
Semester 1, 2023/2024

Non-regular languages

What are some examples of *non-regular* languages?

Canonical examples: Matching parentheses,
$L_1 = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$, $L_2 = \{\mathtt{c}^i\mathtt{a}^j\mathtt{b}^k \mid i = 1 \Rightarrow j = k + 1\}$.

Intuition

Recognising L_1 requires counting the number of $\mathtt{a}$s in the string, which is an unbounded natural number, which requires unbounded memory (not a finite amount of states).

How would we prove this?

Pumping

Suppose a DFA with k states accepts a word of length greater than k. **What must have happened?**
$\Rightarrow$ The DFA must have visited a state more than once
$\Rightarrow$ There is a loop.

Therefore, if we go through that loop any number of times, the DFA should accept those words also. We call this *pumping*.

The Pumping Lemma

Theorem (Pumping Lemma)

If $L \subseteq \Sigma^*$ is regular then there exists a *pumping length* $p \in \mathbb{N}$ such that for any $w \in L$ where $|w| \geq p$, we may split w into three pieces $w = xyz$ satisfying three conditions:

1. xy^iz for all $i \in \mathbb{N}$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

The proof of this relies on the pigeonhole principle.

We can prove a language is non-regular by taking the contrapositive of this.

$$\text{can't be pumped} \quad \Rightarrow \quad \text{not regular}$$

.

Using the Pumping Lemma

To prove a negation (e.g. non-regularity), a common technique is to assume to the contrary that the proposition holds and show that it would lead to a contradiction.

Example (For L_1)

Consider $L_1 = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$. Assume to the contrary that L_1 *is* regular and that p is its pumping length. We know $\mathtt{a}^p\mathtt{b}^p$ is $\in L_1$. No matter how we split this word into xyz, none of these splits satisfies the three conditions of the Pumping Lemma.

Case y consists only of $\mathtt{a}$s: Then $xyyz$ contains more $\mathtt{a}$s than $\mathtt{b}$s, violating condition 1.

Case y contains $\mathtt{b}$s: Then $|xy| > p$ violating condition 3.

Case y is empty (ε): Then $|y| = 0$ violating condition 2.

Another Non-Regular Language

Recall the language $L_2 = \{\mathtt{c}^i\mathtt{a}^j\mathtt{b}^k \mid i = 1 \Rightarrow j = k + 1\}$.

Definition

Define the *left quotient* of a language L, written $w \backslash L$ to be the set of suffixes that can be added to w to produce a word in L:

$$w \backslash L = \{v \mid wv \in L\}$$

Exercise: Prove that $w \backslash L$ is regular when L is regular.

Observe that $\mathtt{ca} \backslash L_2 = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\} = L_1$, which is not regular. Therefore L_2 is also **not regular**.

Limitations of the Pumping Lemma

We have seen that $L_2 = \{\mathtt{c}^i\mathtt{a}^j\mathtt{b}^k \mid i = 1 \Rightarrow j = k + 1\}$ is not regular, but it is possible to pump this.

Assume that L_2 is regular and that p is its pumping length, and that $w \in L_2$ where $|w| \geq p$. We choose x, y (and implicitly z) based on the number of c's in w, written C:

Case $C = 0$: Choose $x = \varepsilon$ and $y =$ first letter of w

Case $0 < C \leq 3$: Choose $x = \varepsilon$ and $y = \mathtt{c}^C$

Case $C > 3$: Choose $x = \varepsilon$ and $y = \mathtt{cc}$

In each case, we can pump (i.e. repeat y arbitrarily many times and stay in L_2.)

So, the converse of the pumping lemma does not hold:

$$\text{can't be pumped} \quad \not\Leftarrow \quad \text{not regular}$$

.

Beyond the Pumping Lemma

The pumping lemma is useful, but not satisfying, because it is not an exact characterisation.

Definition

Let $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$. If there exists a suffix string z such that $xz \in L$ but $yz \notin L$ (or vice-versa), then x and y are *distinguishable* by L.
If x and y are not distinguishable by L, we say $x \equiv_L y$. This is an *equivalence relation*.

The Myhill-Nerode Theorem

A language L is regular iff the number of $\equiv_L$ equivalence classes is finite.
Proof Sketch if time allows.

Using Myhill-Nerode

To use Myhill-Nerode to show that L is non-regular, we must show that there are infinite $\equiv_L$ equivalence classes.

In detail

More specifically, we find an infinite sequence $u_0 u_1 u_2 \ldots$ of strings such that for any i and j (where $i \neq j$), there is a string w_{ij} such that $u_i w_{ij} \in L$ but $u_j w_{ij} \notin L$ (or vice-versa).

Example

- $L_1 = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$, choose $u_i = \mathtt{a}^i$ and $w_{ij} = \mathtt{b}^i$.
- $L_2 = \{\mathtt{c}^i\mathtt{a}^j\mathtt{b}^k \mid i = 1 \Rightarrow j = k + 1\}$, choose $u_i = \mathtt{c}\mathtt{a}^{i+1}$ and $w_{ij} = b^i$.

Context-Free Languages

What would happen if we added recursion to regexps?

Definition

A *Context-free grammar* (CFG) is a 4-tuple (N, Σ, P, S) where:

- N is a finite set of *variables* or *non-terminals*,
- Σ is a finite set of *terminals*
- $P \subseteq N \times (N \cup \Sigma)^*$ is a finite set of *rules* or *productions*. Typically productions are written like:

  $$A \rightarrow \mathtt{a}B\mathtt{c}$$

  Productions with common heads can be combined:

  $$A \rightarrow \mathtt{a} \mid A\mathtt{a} \mid \mathtt{b}A\mathtt{b}$$
- $S \in N$ is the *start variable*.

Context-Free Grammars

Notation: We use α, β, γ etc. to refer to *sequences of terminals*.

Definition (Derivations)

We make a *derivation step* $\alpha A \beta \Rightarrow_G \alpha\gamma\beta$ whenever $(A \to \gamma) \in P$.
The language of a CFG G is:

$$\mathcal{L}(G) = \{w \in \Sigma^* \mid S \Rightarrow^*_G w\}$$

Where $\Rightarrow^*_G$ is the *reflexive transitive closure* of $\Rightarrow_G$.

Example

Given the CFG G:

$$G = (\{S\}, \{0, 1, \{S \to \varepsilon \mid 0S1\}, S)$$

What is the language of G?