NFA and regex

- the Boolean algebra of languages
- regular expressions
The intersection of two regular languages is regular.

$L_0 = \text{even numbers}$

$L_1 = \text{odd numbers}$

$L_0 = 0 \mod 2$

$L_1 = 1 \mod 2$
The intersection of two regular languages is regular

\[ L_0 = 0 \mod 3 \]
\[ L_1 = 1 \mod 3 \]
\[ L_2 = 2 \mod 3 \]
Two examples

Input sequence is accepted if it ends with a zero.

Even binary numbers

Odd binary numbers

Input sequence is accepted if it ends with a one.
Three examples

Which binary numbers are accepted?

<table>
<thead>
<tr>
<th>mod 3</th>
<th>×2</th>
<th>×2 + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
By three or not by three?

- Divisible by three

- Not divisible by three
The complement of a regular language is regular

If $A \subseteq \Sigma^*$ is recognised by $M$
then $\bar{A} = \Sigma^* \setminus A$
is recognised by $\bar{M}$

where $\bar{M}$ and $M$ are identical except that
the accepting states of $\bar{M}$ are the non-accepting states of $M$
and vice-versa.
The intersection of two regular languages is regular

divisible by 6
≡
divisible by 2
and
divisible by 3
The intersection of two regular languages is regular

Run both machines in parallel?

Build one machine that simulates two machines running in parallel!

Keep track of the state of each machine.
The intersection of two regular languages is regular
intersection of languages
run the two machines in parallel
when a string is in both languages,
both are in an accepting state
The intersection of two regular languages is regular.
The intersection of two regular languages is regular
The intersection of two regular languages is regular
The regular languages $A \subseteq \Sigma^*$ form a Boolean Algebra

- Since they are closed under intersection and complement.
Non Determinism

In a non-deterministic machine (NFA), each state may have any number of transitions with the same input symbol, leaving to different successor states.

In a non-deterministic machine (NFA), each state may lead to any number of transitions with the same input symbol, leaving to different successor states.

We have a transition relation
Non Determinism

In a non-deterministic machine (NFA), each state may have any number of transitions with the same input symbol, leaving to different successor states.
Non Determinism

We can simulate a non-deterministic machine using a deterministic machine – by keeping track of the set of states the NFA could possibly be in.
Internal Transitions

We sometimes add an internal transition $\varepsilon$ to a non-deterministic machine (NFA). This is a state change that consumes no input.

![Diagram of a non-deterministic finite automaton (NFA) with internal transitions $\varepsilon$](image)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We sometimes add **internal transitions** – labelled $\varepsilon$ – to a non-deterministic machine (NFA).

This is a state change that consumes no input.

It introduces non-determinism in the observed behaviour of the machine.
NFA any number of start states and accepting states
sequence
RS
alternation  \( R \mid S \)
iteration $R^*$