Chapter 9
Lecture 17
Chris J. Myers
Finite Automata
A DFA accepting all strings ending in 111
An NFA (Non-deterministic Finite Automaton)

{ }
Example for DECIDE-Finite Automata

\[ \{ x \in \mathcal{L} \mid x \in \mathcal{I} \} = \mathcal{L}^+ = \mathcal{L} \cup \mathcal{L} \]

The union of two languages \( \mathcal{L} \) and \( \mathcal{J} \) is:

\[ \{ x \in \mathcal{L} \mid x \in \mathcal{J} \} = \mathcal{L} \mathcal{J} \]

The product of two languages \( \mathcal{L} \) and \( \mathcal{J} \) is:

\[ \{ x \in \mathcal{J} \mid x \in \mathcal{I} \} = \mathcal{J}^\mathcal{I} \]

The set of strings accepted by a FA is called the language (\( \mathcal{L} \)).
If we delete the empty string, we get \( \mathcal{X}^+ \).

Every member of \( \mathcal{X}^+ \) is finite, but is an infinite set.

\[
\begin{align*}
\bigcup_{u=0}^{n} \mathcal{X} \quad & = \quad *\mathcal{X} \\
\mathcal{X} \cup \mathcal{X} \quad & = \quad 1+\mathcal{X} \\
\{\} \quad & = \quad 0\mathcal{X}
\end{align*}
\]

The star operation is defined recursively as follows:

**Definition 9.1.3** For \( \mathcal{X} \subseteq \mathcal{X}^* \), the result of the star operation is defined recursively as follows:

Every DFA \( \mathcal{A} \) is regular.

Theorem 9.1.1 For every regular language \( \mathcal{J} \), there is a DFA whose language is \( \mathcal{J} \).

Conversely, the language \( \mathcal{J} \) of a DFA \( \mathcal{A} \) is regular.

Not every set \( X \) is regular. For every \( x \in \mathcal{X} \), the subset \( \{x\} \) of \( \mathcal{X} \) is regular.

Definition 9.1.4 A set \( \mathcal{X} \subseteq \mathcal{X} \) is regular if it's a finite number of union of singletons, and if it can be obtained from the empty set and the set of languages of regular sets (or languages).
In 011
Strings ending
then 00 or 11
nonzero 0s
Input set,
Input length
Two elements,
Empty string
One element,
Empty language

Regular Expressions

\( (q + \epsilon) \ast 0 \) accepting DFA and NFA
A DFA accepting $(1 + 0)(1 + 0)$.

A DFA for a Module 3 Counter and its Complement.
DFA Synthesis

1. Construct a binary parse tree for the given regular expression \( R \).
2. Construct from \( N(\tau) \) an NFA which accepts the same language represented by \( R \).
3. Construct from \( N(\tau) \) a DFA which accepts the same language.
Rule for Product of Two Regular Expressions

Incorrect Product Rule
Rule for Union of Two Regular Expressions

Rule for Closure of Two Regular Expressions
\[ *(q, v) \text{ is NFA language} \]

```
(L'L)subs-cons
\{sfp-unu\} \cap _{V} = _{V} \quad (L'L \ni \forall \gamma_{N}) \text{I}
\{(L'L, sfp-unu)\} \cap _{\text{subsim}} = \text{Is}
I + sfp-unu = sfp-unu
(0 = \text{visited}) \text{I}
break
I = \text{path}
\{\forall \gamma \ni (L'L, \gamma)\} \text{I}
\forall \text{foreach}
0 = \text{visited}
\forall\text{foreach}
\forall x.L \cap \forall x.L = \forall x.L
(\forall x.s{x})\forall x.s{x} = \forall x.L
\forall x.\text{foreach}
\emptyset = \forall x.L
(\forall x.s{x})\forall x.s{x}
```
The automaton acceptance condition is:

1. Some path is crossed an infinite number of times.
2. The set of states that are reached an infinite number of times are all contained in one cycle set.

DFA whose language is \( \{a^n b^n c^n \}_n \).
\( \mathcal{C} \) is a set of \( \mathcal{L} \) cycle sets \( \{C_1, \ldots, C_n\} \).

\( \mathcal{E} \) is the set of recurrent edges.

\( \mathcal{S} \subseteq \mathcal{S}_i \) is the set of initial (reset) states.

\( S \) is the next-state function:

\( S : 2^S \rightarrow X \times S \) or \( S : \emptyset \rightarrow X \times S \).

\( S \) is the (finite, non-empty) set of states.

\( X \) is the input alphabet (a finite, non-empty set of input values).

\( \langle \forall \mathcal{C}, \forall \mathcal{E}, \forall \mathcal{S}, \forall X \rangle \)

\( \text{Regular Automata (L-Automata)} \)

An L-Automaton for Expressing a Safety Property
An L-Automaton Recognizing a Tape with at Most $2 \theta$

Infinite Number of Head Substrings

An L-Automaton Recognizing a Tape Containing an
An Example of a Product Automaton

\[ \psi = \bigcup_{n \geq 1} \mathcal{L}_1 \]

5. Language containment check succeeds if and only if

\[ ((\hat{\gamma})^1_{\text{c}} \cdot (x, \hat{\gamma})) \sim \mathcal{L} \cdot (\hat{\gamma}, x) \mathcal{L} \mathcal{E} = (x)^1_{\text{c}} \mathcal{N} \]

as follows:

1. Compute the language containment (c)

(\{\cdot\} \sigma, (x, \hat{\gamma}), \mathcal{L} \mathcal{E}) \mathcal{F}

2. Compute the transitive closure of \( \mathcal{L} \mathcal{E} \)

(\{\cdot\} \sigma, (x, \hat{\gamma}), \mathcal{L} \mathcal{F})

3. Compute the transitive closure, of \( \mathcal{L} \mathcal{E} \)

4. For each cycle set \( \mathcal{C} \), compute the set of cycles or not contained in \( \mathcal{E} \).

5. Remove the repeat edges and restrict to \( \mathcal{F} \).

Procedure Language Containment (c)
Illustration of Non-Containment in Cycle Set

Language Containment Test on Product Automaton
Summary

- \( \mu \)-regular languages
- Synthesis of DFAs from regular languages
- Finite automata and regular languages