THEORY OF AUTOMATA & FORMAL LANGUAGES

Building Formal Languages

From individual strings to sets of valid strings

Generate → Select → Describe → Verify

From strings to languages

A language is a collection of strings over a chosen alphabet.

Start with an alphabet

$\Sigma = \{a, b\}$

Individual symbols: a and b

Strings: ε, a, b, aa, ab, ba, bb, ...

Select strings: a, ab, abb

$L = \{a, ab, abb\}$

Read the notation

Σ is a set of symbols.

L is a set of strings.

$a \in L$ means a belongs to L.

$ba \notin L$ means ba is not in L.

Every string in L uses symbols from Σ.

A language is a set, not a single string or an ordered list.

Constructing a finite language

A finite language contains a limited number of distinct strings.

Build a language of length two

Choose $\Sigma = \{a, b\}$.

First symbol: a or b

Second symbol: a or b

List: aa, ab, ba, bb

$L = \{aa, ab, ba, bb\}$

Count and verify

$|L| = 4$ (number of strings)

$|ab| = 2$ (length of one string)

$ab \in L$; $ba \in L$

$a \notin L$; $aba \notin L$

$\varepsilon \notin L$ because its length is 0.

For an alphabet of k symbols, exactly k^n strings have length n.

Constructing an infinite language

An infinite language has no finite limit on the number of distinct strings.

Example: one or more a’s

Choose $\Sigma = \{a, b\}$.

One a: a

Two a’s: aa

Three a’s: aaa

$L = \{a, aa, aaa, aaaa, \ldots\}$

Describe the rule

Write a^n for n copies of a.

$L = \{a^n : n \geq 1, n \text{ integer}\}$

n = 4 gives aaaa.

$\varepsilon \notin L$ because n starts at 1.

$ab \notin L$ because b is not allowed.

An infinite language can be specified by a finite mathematical rule.

Language membership: testing a rule

Membership means checking whether a given string meets every condition of a language.

Example: begins with a

$\Sigma = \{a, b\}$

L = all strings beginning with a

a → yes

abb → yes

ba → no

Check boundary cases

ε → no: no first symbol

aa → yes

aba → yes

b → no

ac → no: c is outside Σ

Check the alphabet first, then test the language condition.

Building languages using conditions

Combine the string operations studied previously with a clear selection condition.

Example: begins with a, ends with b

$\Sigma = \{a, b\}$

Minimum length is 2.

Length 2: ab

Length 3: aab, abb

Length 4: aaab, aabb, abab, abbb

Example: equal blocks

$L = \{a^n b^n : n \geq 0, n \text{ integer}\}$

n = 0: ε

n = 1: ab

n = 2: aabb

n = 3: aaabbb

Equal numbers alone do not suffice: abab is not of the form $a^n b^n$.

Empty language and empty string

The empty language and the language containing the empty string are different objects.

Empty language

$L_1 = \emptyset$

No strings are members.

$|L_1| = 0$

$\varepsilon \notin L_1$

$a \notin L_1$

Language containing ε

$L_2 = \{\varepsilon\}$

Exactly one string is a member.

$|L_2| = 1$

$\varepsilon \in L_2$ and $|\varepsilon| = 0$

$a \notin L_2$

Zero strings is not the same as one string of length zero.

From words to mathematical rules

A language can be described verbally, by listing strings, or using a mathematical condition.

Example: at most two a’s, no b

$\Sigma = \{a, b\}$

Zero a’s: ε

One a: a

Two a’s: aa

$L = \{\varepsilon, a, aa\}$

Example: even length, only a

Zero a’s: ε

Two a’s: aa

Four a’s: aaaa

$L = \{a^{2n} : n \geq 0, n \text{ integer}\}$

aaa is excluded: length 3 is odd.

Always specify the alphabet, the allowed pattern and the range of the parameter.

Practice and explained answers

Use $\Sigma = \{a, b\}$. Build languages and test membership before moving to repetition operators.

Try these questions

1. List all strings of length 1.

2. List strings of length 2 ending in a.

3. Is ε in $\{a^n : n \geq 1\}$?

4. Is abba of the form $a^n b^n$?

5. Is $\{\varepsilon, ab\}$ finite?

Answers and reasoning

1. $\{a, b\}$

2. $\{aa, ba\}$

3. No: n cannot be zero.

4. No: a occurs after b.

5. Yes: it contains two strings.

Next: systematic generation of strings and languages using Kleene star and Kleene plus.