

BU CS 332 – Theory of Computation

<https://forms.gle/SChbkRQXC5cScVss8>



Lecture 6:

- Closure Properties
- Regular Expressions

Reading:

Sipser Ch 1.2-1.3

Alexander Poremba & Mark Bun

February 3, 2026

Last Time

- Nondeterministic Finite Automata
- NFAs vs. DFAs
 - Subset construction: NFA \rightarrow DFA

Closure Properties

An Analogy

In algebra, we try to identify operations which are common to many different mathematical structures

Example: The integers $\mathbb{Z} = \{\dots - 2, -1, 0, 1, 2, \dots\}$ are **closed** under

- Addition: $x + y$ $-4 + 8 = 4 \in \mathbb{Z}$
- Multiplication: $x \times y$ $(-4) \times 8 = -32 \in \mathbb{Z}$
- Negation: $-x$ $-(-4) = 4 \in \mathbb{Z}$
- ...but **NOT** Division: x / y $-4 / 8 = -\frac{1}{2} \notin \mathbb{Z}$

We'd like to investigate similar closure properties of the **class of regular languages**

Regular operations on languages E.g. $A = \{10\}$

Let $A, B \subseteq \Sigma^*$ be languages. Define

$$A^* = \{ \epsilon, 10, 1010, 101010, \dots \}$$

Union: $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$

$$B = \{10, 1\}$$

$$B^* = \{ \epsilon, 10, 1, 1010, 101, 110, 11, 101010, 10101, \dots \}$$

Concatenation: $A \circ B = \{xy \mid x \in A, y \in B\}$

$$A \circ \{ \epsilon \}$$

Star: $A^* = \{ a_1 a_2 \dots a_n \mid n \geq 0, \text{ each } a_i \in A \}$

For $a \in A$:

For $b \in B$:

Add $a \circ b$

to set

$$= \underbrace{\{ \epsilon \}}_{n=0} \cup \underbrace{A}_{n=1} \cup \underbrace{(A \circ A)}_{n=2} \cup \underbrace{(A \circ A \circ A)}_{n=3} \dots$$

$$= A$$

Sanity check: If $A = \Sigma$ where Σ is a finite alphabet, this agrees w/ our previous definition of Σ^*

Other operations

Let $A, B \subseteq \Sigma^*$ be languages. Define

Complement: $\bar{A} = \{w \mid w \notin A\}$

Intersection: $A \cap B = \{w \mid w \in A \text{ and } w \in B\}$

Reverse: $A^R = \{w \mid w^R \in A\}$

Operations on languages

Let $A, B \subseteq \Sigma^*$ be languages. Define

Regular Operations

$$\left\{ \begin{array}{l} \text{Union: } A \cup B = \{x \mid x \in A \text{ or } x \in B\} \\ \text{Concatenation: } A \circ B = \{xy \mid x \in A, y \in B\} \\ \text{Star: } A^* = \{w_1 w_2 \dots w_n \mid n \geq 0 \text{ and } w_i \in A\} \\ \\ \text{Complement: } \bar{A} = \{x \mid x \notin A\} \\ \text{Intersection: } A \cap B = \{x \mid x \in A \text{ and } x \in B\} \\ \text{Reverse: } A^R = \{a_1 a_2 \dots a_n \mid a_n \dots a_1 \in A\} \end{array} \right.$$

Theorem: The class of regular languages is **closed** under all six of these operations, i.e., if A and B are regular, applying any of these operations yields a regular language

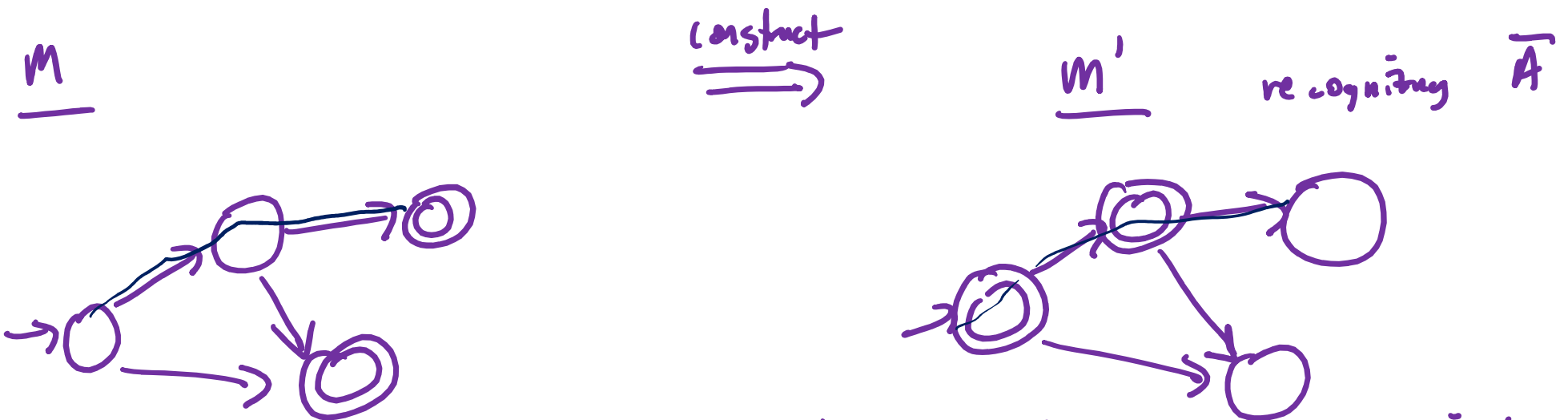
Proving Closure Properties

Complement

Complement: $\bar{A} = \{w \mid w \notin A\}$

Theorem: If A is regular, then \bar{A} is also regular

Proof idea: Let A be an arbitrary regular language
Then there exists a DFA M recognizing A



Obtain M' by interchanging accept and reject

\Rightarrow states of M \Rightarrow M' recognizes \bar{A} \Rightarrow There exists a DFA recognizing \bar{A} $\Rightarrow \bar{A}$ is regular

Complement, Formally



Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing a language A . Which of the following represents a DFA recognizing \bar{A} ?

- a) $(F, \Sigma, \delta, q_0, Q)$ set of accept states in new machine M' is exactly the set of reject states in M
- b) $(Q, \Sigma, \delta, q_0, Q \setminus F)$, where $Q \setminus F$ is the set of states in Q that are not in F
- c) $(Q, \Sigma, \delta', q_0, F)$ where $\delta'(q, s) = p$ such that $\delta(p, s) = q$
- d) None of the above

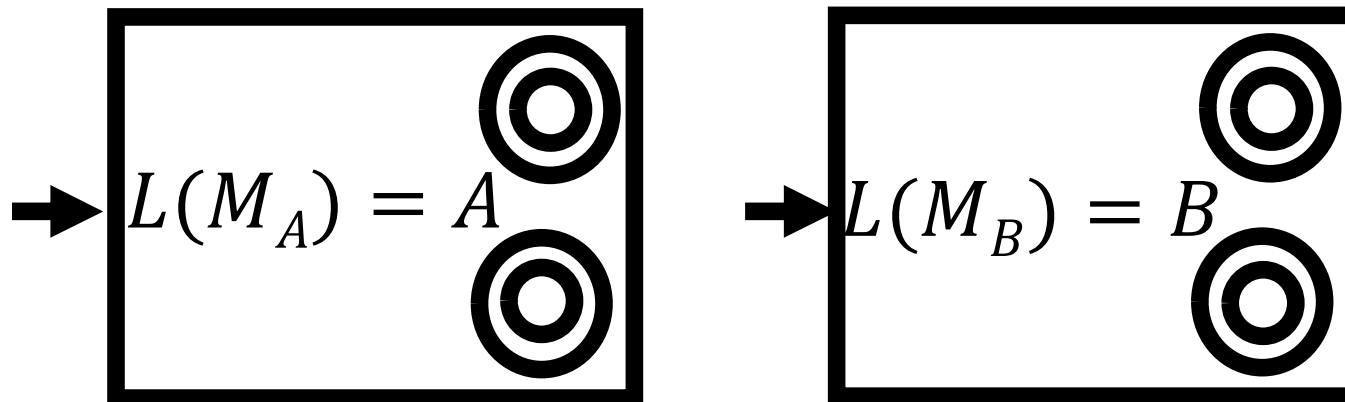
Closure under Concatenation

Concatenation: $A \circ B = \{ xy \mid x \in A, y \in B \}$

Theorem. If A and B are regular, then $A \circ B$ is also regular.

Proof idea: Given DFAs M_A and M_B , construct an NFA for $A \circ B$ by

- Connecting all accept states in M_A to the start state in M_B .
- Making all states in M_A non-accepting.



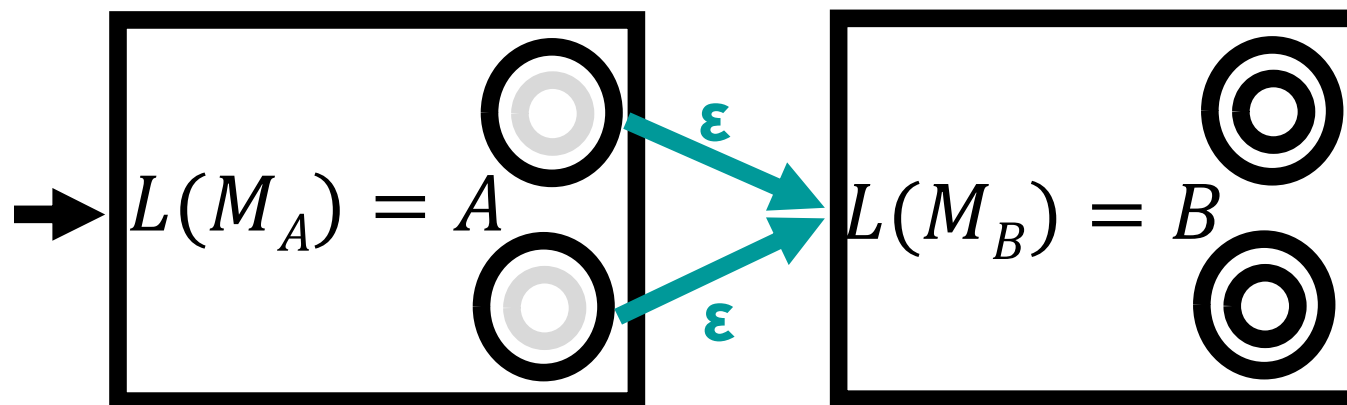
Closure under Concatenation

Concatenation: $A \circ B = \{ xy \mid x \in A, y \in B \}$

Theorem. If A and B are regular, then $A \circ B$ is also regular.

Proof idea: Given DFAs M_A and M_B , construct an NFA for $A \circ B$ by

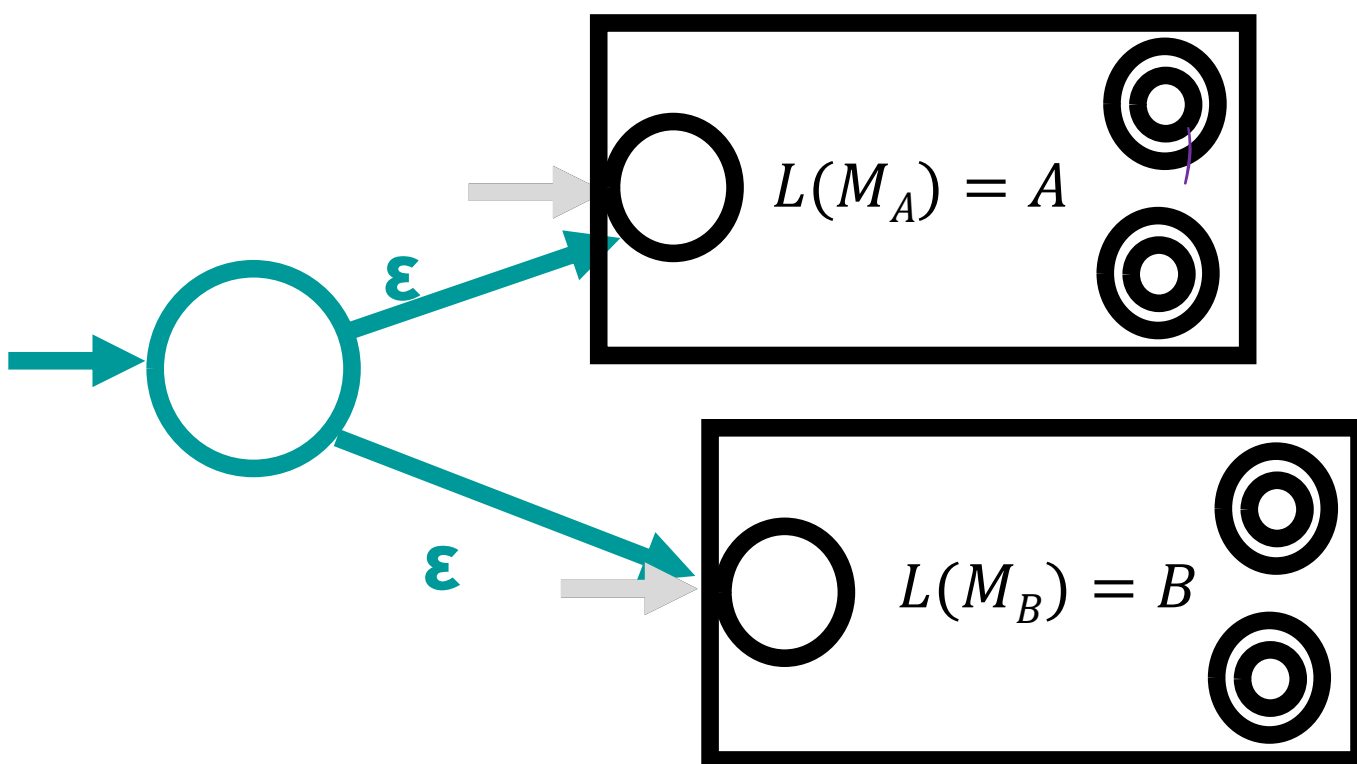
- Connecting all accept states in M_A to the start state in M_B .
- Making all states in M_A non-accepting.



- If $z \in A \circ B$
i.e. $z = xy$ for
some $x \in A, y \in B$
then new machine
accepts z
- Every string new
machine accepts
is in $A \circ B$

A Mystery Construction

Given DFAs M_A recognizing A and M_B recognizing B , what does the following NFA recognize?



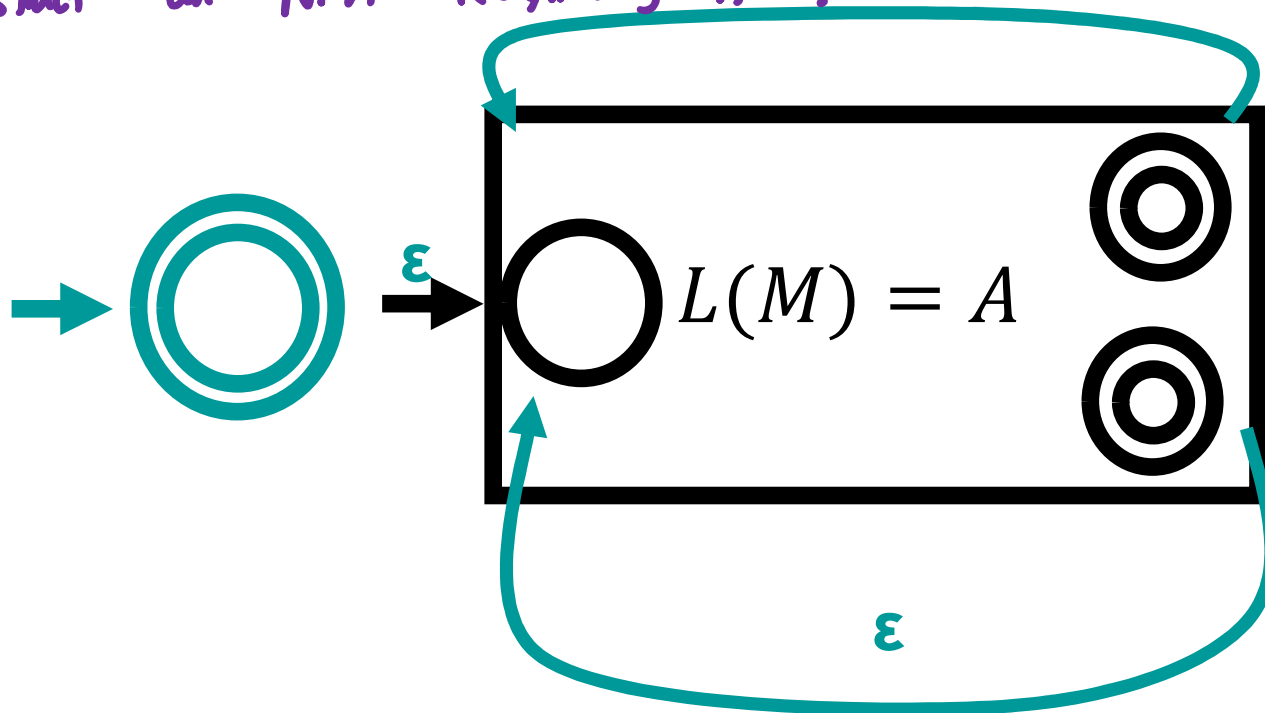
- a) $A \cup B$
- b) $A \circ B$
- c) $A \cap B$
- d) $\{\epsilon\} \cup A \cup B$

Closure under Star

Star: $A^* = \{ a_1 a_2 \dots a_n \mid n \geq 0 \text{ and each } a_i \in A \}$

Theorem. If A is regular, then A^* is also regular.

Let A be an arbitrary regular language recognized by M
Construct an NFA recognizing A^* :



On proving your own closure properties

You'll have homework/test problems of the form "show that the regular languages are closed under some operation"

wts: \forall regular languages A, B that $op(A, B)$ is $op(A, B)$
also regular

What would Sipser do?

- Give the "proof idea": Explain how to take machine(s) recognizing regular language(s) and create a new machine
- Explain in a few sentences why the construction works
- Give a formal description of the construction
- No need to formally prove that the construction works

Regular Expressions

Regular Expressions

- A different way of describing regular languages
- A regular expression expresses a (possibly complex) language by combining simple languages using the regular operations

“Simple” languages: \emptyset , $\{\varepsilon\}$, $\{a\}$ for some $a \in \Sigma$

Regular operations:

Union: $A \cup B$

Concatenation: $A \circ B = \{ab \mid a \in A, b \in B\}$

Star: $A^* = \{a_1 a_2 \dots a_n \mid n \geq 0 \text{ and } a_i \in A\}$

Regular Expressions – Syntax

A regular expression R is defined recursively using the following rules:

1. ε , \emptyset , and a are regular expressions for every $a \in \Sigma$
2. If R_1 and R_2 are regular expressions, then so are $(R_1 \cup R_2)$, $(R_1 \circ R_2)$, and (R_1^*)

Examples: (over $\Sigma = \{a, b, c\}$)

$(a \circ b)$ $((((a \circ (b^*)) \circ c) \cup (((a^*) \circ b))^*))$ (\emptyset^*)

Regular Expressions – Semantics

$L(R)$ = the language a regular expression describes

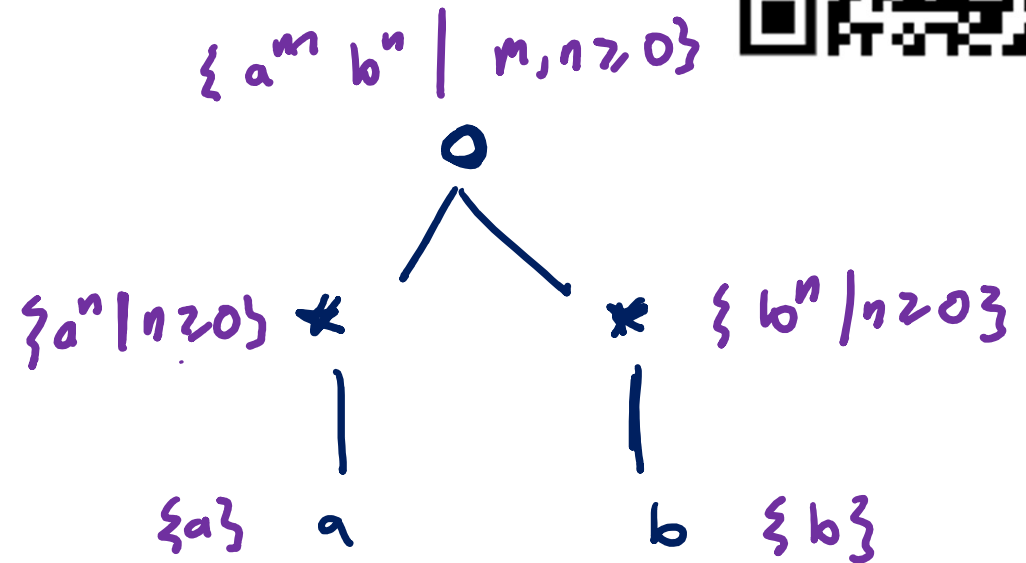
1. $L(\emptyset) = \emptyset$
2. $L(\varepsilon) = \{\varepsilon\}$
3. $L(a) = \{a\}$ for every $a \in \Sigma$
4. $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$
5. $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$
6. $L((R_1^*)) = (L(R_1))^*$

Regular Expressions – Example



$$L(((a^*) \circ (b^*))) =$$

- a) $\{a^n b^n \mid n \geq 0\}$
- b) $\{a^m b^n \mid m, n \geq 0\}$**
- c) $\{(ab)^n \mid n \geq 0\}$
- d) $\{a, b\}^*$



Simplifying Notation

- Omit \circ symbol: $(ab) = (a \circ b)$
- Omit many parentheses, since union and concatenation are associative:

$$(a \cup b \cup c) = (a \cup (b \cup c)) = ((a \cup b) \cup c)$$

- Order of operations: Evaluate star, then concatenation, then union

$$ab^* \cup c = (a(b^*)) \cup c$$

Examples

$0, 1, \phi, \epsilon, \cup, \circ, *, (,)$

Let $\Sigma = \{0, 1\}$

1. $\{w \mid w \text{ contains exactly one } 1\}$

$\{x1y \mid x \in L(0^*), y \in L(0^*)\}$

$0^* 1 0^*$

2. $\{w \mid w \text{ has length at least 3 and its third symbol is } 0\}$

$0^* \underline{1}^* \underline{0} \underline{0}^* 1^*$
100101

$(0 \cup 1)(0 \cup 1) 0 (0 \cup 1)^*$

$(0^* \cup 1^*)(0^* \cup 1^*) 0 (0^* \cup 1^*)$

3. $\{w \mid \text{every odd position of } w \text{ is } 1\}$

$(1(1 \cup 0 \cup \epsilon))^*$

generates

(1)(10)

$(1(0 \cup 1))^*$

does not generate 1

$(1(0 \cup 1))^* \cup (1(0 \cup 1))^* 1$

even length, every odd position is 1

odd length, every odd position is 1

$(1(0 \cup 1))^* (\epsilon \cup 1)$

Syntactic Sugar

- For alphabet Σ , the regex Σ represents $L(\Sigma) = \Sigma$
- For regex R , the regex $R^+ = RR^*$

Regexes in the Real World

`grep` = globally search for a regular expression and print matching lines

```
$ grep '^xy*z' myfile
xyz
xyzde
xzz
xz
xyyz
xyyyz
xyyyyz
$ grep '^x.*z' myfile
xyz
xyzde
xxz
xzz
x\z
x*z
xz
x z
xYz
xyyz
xyyyz
xyyyyz
$ grep '^x\*z' myfile
x*z
$ grep '\\\z' myfile
x\z
$
```