http://algs4.cs.princeton.edu

5.4 Regular Expressions

- *regular expressions*
- REs and NFAs
- NFA simulation
- NFA construction

Last updated on 11/30/15 11:30 AM

http://algs4.cs.princeton.edu

5.4 Regular Expressions

- *regular expressions*
- *REs and NFAs*
- *NFA simulation*
- *NFA construction*

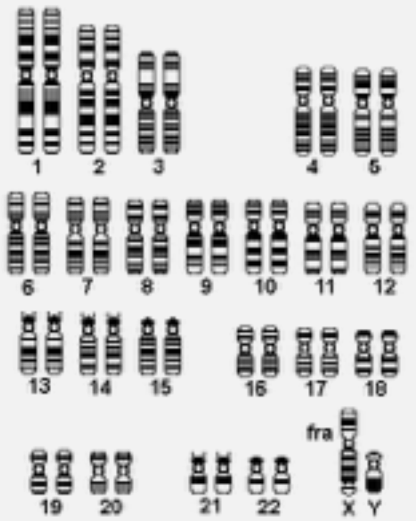
Pattern matching

Substring search. Find a single string in text.

Pattern matching. Find one of a specified set of strings in text.

Ex. [genomics]

- Fragile X syndrome is a common cause of mental retardation.
- A human's genome is a string.
- It contains triplet repeats of CGG or AGG, bracketed by GCG at the beginning and CTG at the end.
- Number of repeats is variable and is correlated to syndrome.



pattern GCG(CGG|AGG)*CTG

text GCGGCGTGTGTGCGAGAGAGTGGGTTTAAAGCTGGCGCGGAGGCGGCTGGCGCGGAGGCTG

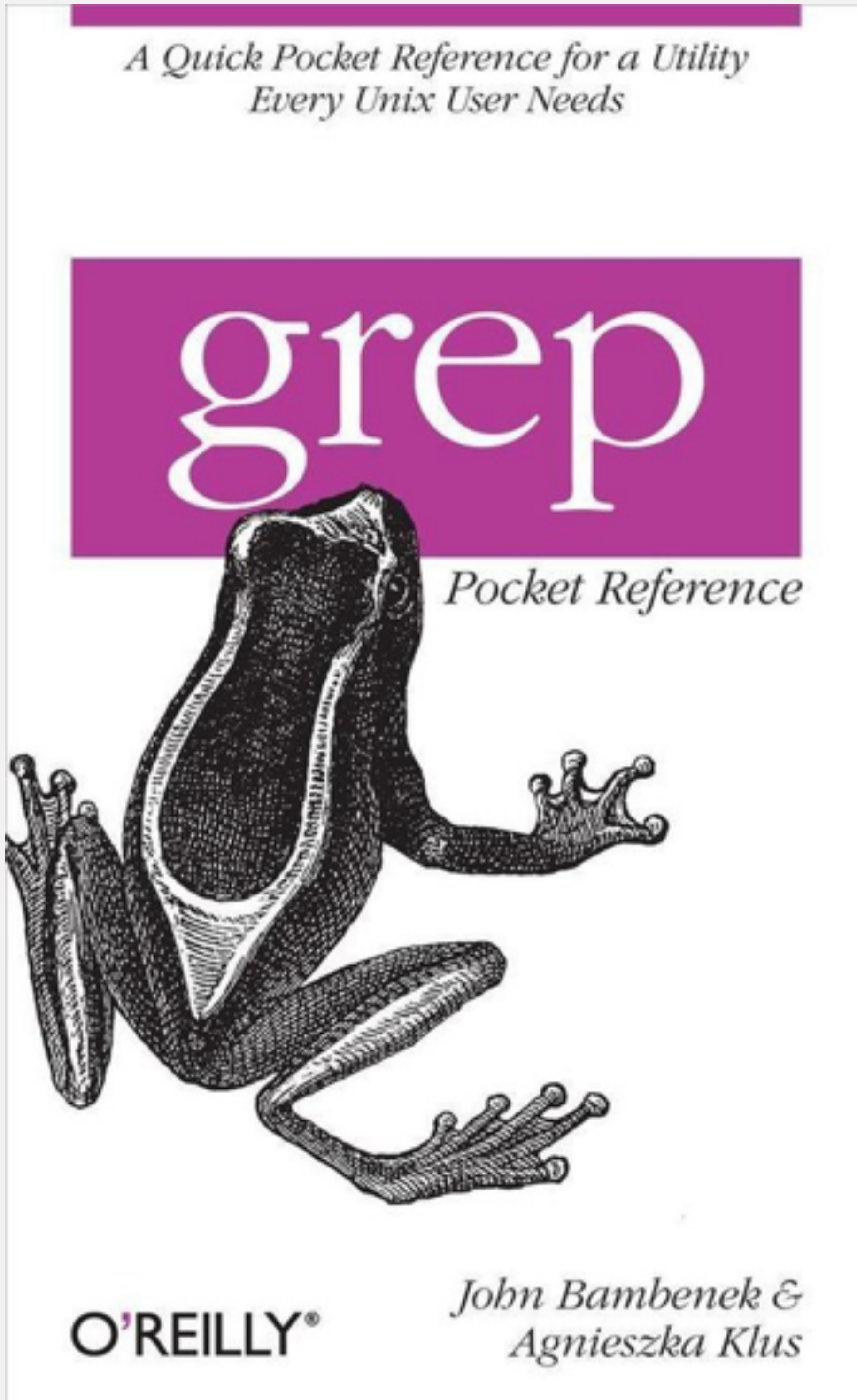
Syntax highlighting

```
/*************************************************************************
 *  Compilation:  javac NFA.java
 *  Execution:    java NFA regexp text
 *  Dependencies: Stack.java Bag.java Digraph.java DirectedDFS.java
 *
 *  % java NFA "(A*B|AC)D" AAAABD
 *  true
 *
 *  % java NFA "(A*B|AC)D" AAAAC
 *  false
 *
 *************************************************************************/

public class NFA
{
    private Digraph G;         // digraph of epsilon transitions
    private String regexp;     // regular expression
    private int M;             // number of characters in regular expression

    // Create the NFA for the given RE
    public NFA(String regexp)
    {
        this.regexp = regexp;
        M = regexp.length();
        Stack<Integer> ops = new Stack<Integer>();
        G = new Digraph(M+1);
        ...
```

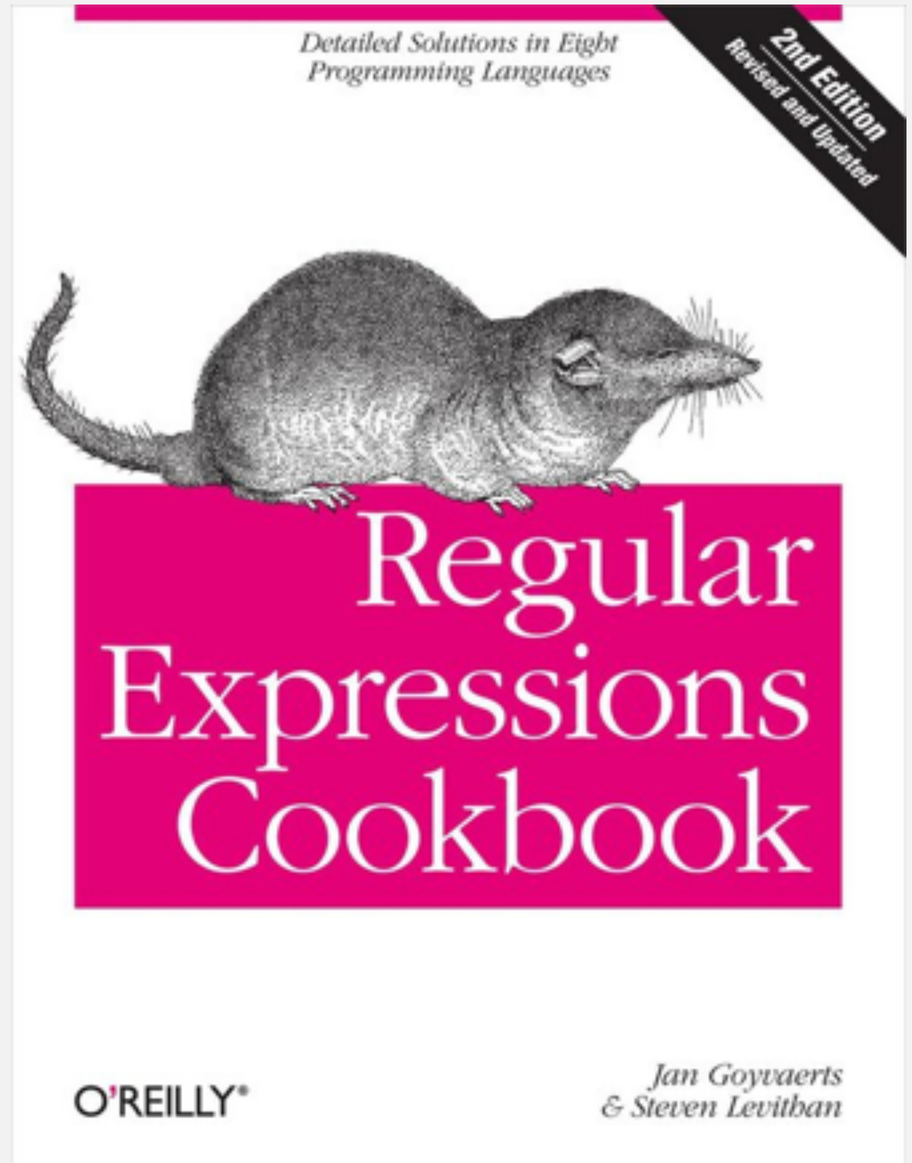
GNU source-highlight 3.1.4

input	**output**
Ada	HTML
Asm	XHTML
Applescript	LATEX
Awk	MediaWiki
Bat	ODF
Bib	TEXINFO
Bison	ANSI
C/C++	DocBook
C#	
Cobol	
Caml	
Changelog	
Css	
D	
Erlang	
Flex	
Fortran	
GLSL	
Haskell	
Html	
Java	
Javalog	
Javascript	
Latex	
Lisp	
Lua	
⋮	

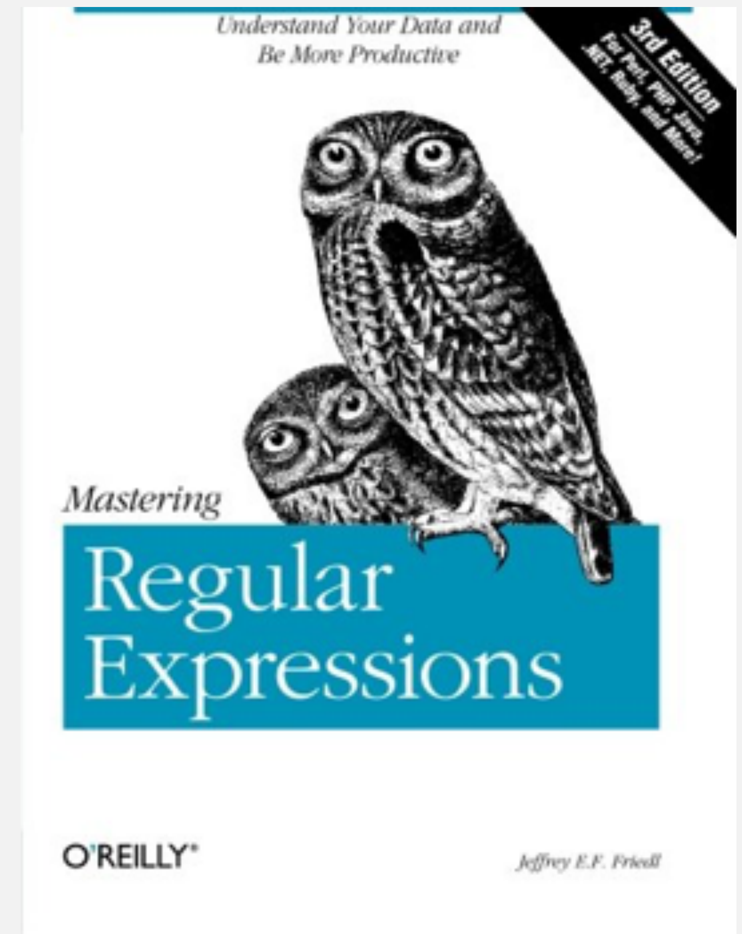
grep



A Quick Pocket Reference for a Utility Every Unix User Needs
grep
Pocket Reference
O'REILLY
John Bambenek & Agnieszka Klus



Detailed Solutions in Eight Programming Languages
2nd Edition
Revised and Updated
Regular Expressions Cookbook
O'REILLY
Jan Goyvaerts & Steven Levithan



Understand Your Data and Be More Productive
3rd Edition
Mastering
Regular Expressions
O'REILLY
Jeffrey E.F. Friedl

Regular expressions

A regular expression is a notation to specify a set of strings.

possibly infinite

operation	order	example RE	matches	does not match
concatenation	3	AABAAB	AABAAB	*every other string*
or	4	AA | BAAB	AA BAAB	*every other string*
closure	2	AB*A	AA ABBBBBBBBA	AB ABABA
parentheses	1	A(A|B)AAB	AAAAB ABAAB	*every other string*
		(AB)*A	A ABABABABABA	AA ABBA

Regular expression shortcuts

Additional operations further extend the utility of REs.

operation	example RE	matches	does not match
wildcard	`.U.U.U.`	CUMULUS JUGULUM	SUCCUBUS TUMULTUOUS
character class	`[A-Za-z][a-z]*`	word Capitalized	camelCase 4illegal
one or more	`A(BC)+DE`	ABCDE ABCBCDE	ADE BCDE
exactly k	`[0-9]{5}-[0-9]{4}`	08540-1321 19072-5541	111111111 166-54-111

Note. These operations are useful but not essential.

Ex. `[A-E]+` is shorthand for `(A|B|C|D|E)(A|B|C|D|E)*`

Regular expression examples

RE notation is surprisingly expressive.

regular expression	matches	does not match
`.*SPB.*` *(substring search)*	RASPBERRY CRISPBREAD	SUBSPACE SUBSPECIES
`[0-9]{3}-[0-9]{2}-[0-9]{4}` *(U. S. Social Security numbers)*	166-11-4433 166-45-1111	11-55555555 8675309
`[a-z]+@([a-z]+\.)+(edu|com)` *(simplified email addresses)*	wayne@princeton.edu rs@princeton.edu	spam@nowhere
`[$_A-Za-z][$_A-Za-z0-9]*` *(Java identifiers)*	ident3 PatternMatcher	3a ident#3

REs play a well-understood role in the theory of computation.

Illegally screening a job candidate

“ [First name]! and pre/2 [last name] w/7
bush or gore or republican! or democrat! or charg!
or accus! or criticiz! or blam! or defend! or iran contra
or clinton or spotted owl or florida recount or sex!
or controvers! or fraud! or investigat! or bankrupt!
or layoff! or downsiz! or PNTR or NAFTA or outsourc!
or indict! or enron or kerry or iraq or wmd! or arrest!
or intox! or fired or racis! or intox! or slur!
or controvers! or abortion! or gay! or homosexual!
or gun! or firearm! ”

— LexisNexis search pattern used by Monica Goodling to screen candidates for DOJ positions

LexisNexis™

http://www.justice.gov/oig/special/s0807/final.pdf

Regular expressions to the rescue

http://xkcd.com/208

http://algs4.cs.princeton.edu

5.4 Regular Expressions

- *regular expressions*
- *REs and NFAs*
- *NFA simulation*
- *NFA construction*

Duality between REs and DFAs

RE. Concise way to describe a set of strings.

DFA. Machine to recognize whether a given string is in a given set.

Kleene's theorem.

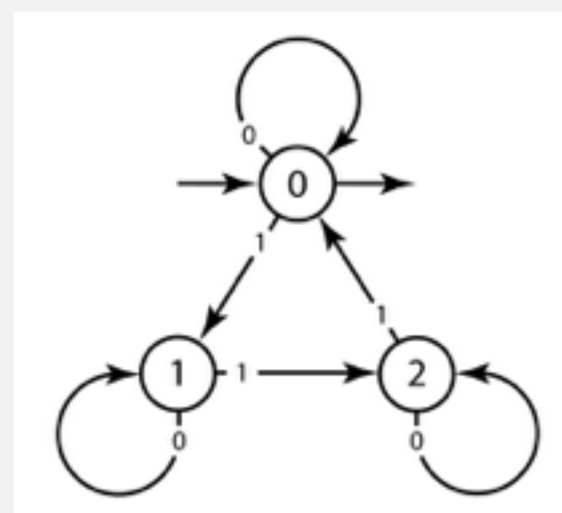
- For any DFA, there exists a RE that describes the same set of strings.
- For any RE, there exists a DFA that recognizes the same set of strings.

RE

0* | (0*10*10*10*)*

number of 1's is a multiple of 3

DFA



number of 1's is a multiple of 3

Work by Stephen Kleene in the 1930s!

Pattern matching implementation: basic plan (first attempt)

Ken Thompson
Turing Award '83

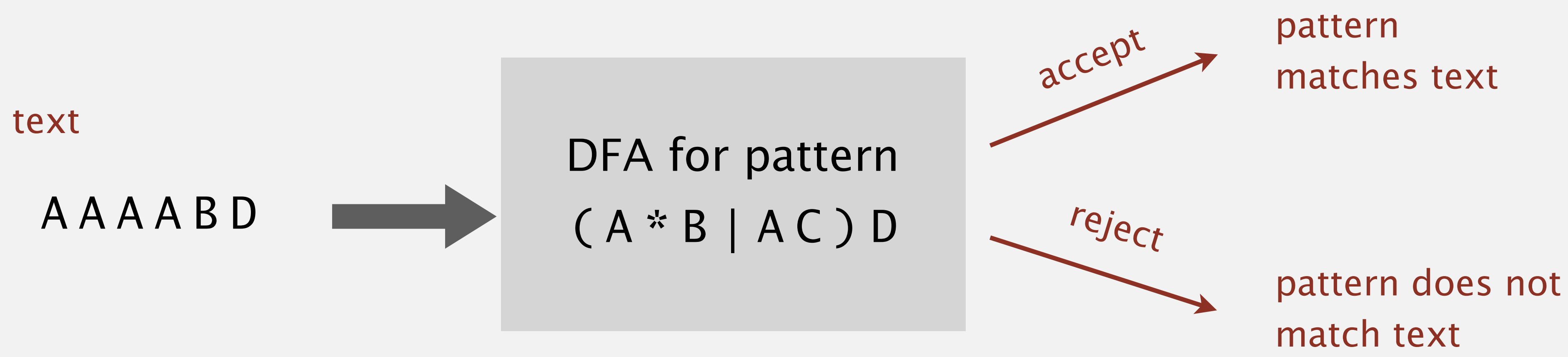
Overview is the same as for Knuth-Morris-Pratt.

- No backup in text input stream.
- Linear-time guarantee.

Underlying abstraction. Deterministic finite state automata (DFA).

Basic plan. [apply Kleene's theorem]

- Build DFA from RE.
- Simulate DFA with text as input.



Bad news. Basic plan is infeasible (DFA may have exponential # of states).

Pattern matching implementation: basic plan (revised)

Overview is similar to Knuth-Morris-Pratt.

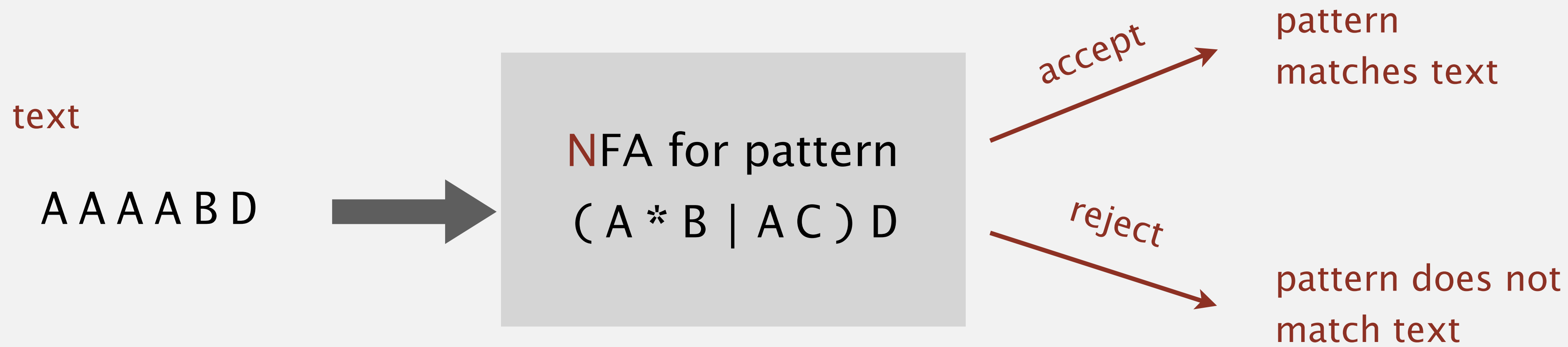
- No backup in text input stream.
- Quadratic-time guarantee (linear-time typical).

Ken Thompson
Turing Award '83

Underlying abstraction. Nondeterministic finite state automata (NFA).

Basic plan. [apply Kleene's theorem]

- Build NFA from RE.
- Simulate NFA with text as input.



Q. What is an NFA?

Nondeterministic finite-state automata

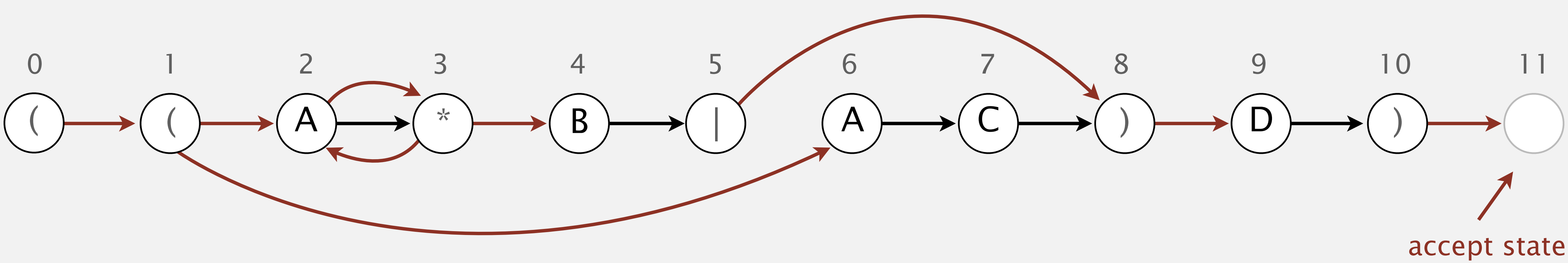
Regular-expression-matching NFA.

- We assume RE enclosed in parentheses.
- One state per RE character (start = 0, accept = M).
- Red ε-transition (change state, but don't scan text).
- Black match transition (change state and scan to next text char).
- Accept if any sequence of transitions ends in accept state.

after scanning all text characters

Nondeterminism.

- One view: machine can guess the proper sequence of state transitions.
- Another view: sequence is a proof that the machine accepts the text.



NFA corresponding to the pattern ((A * B | A C) D)

Nondeterministic finite-state automata

Q. Is A A A A B D matched by NFA?

A. Yes, because some sequence of legal transitions ends in state 11.

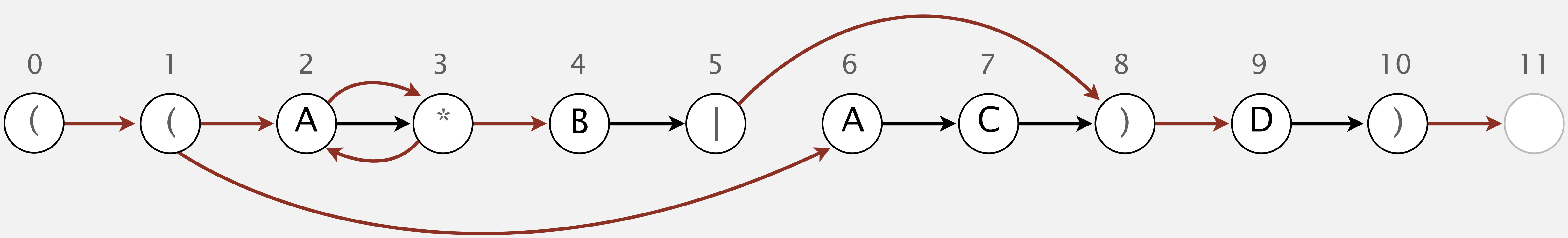
A A A A B D

0 → 1 → 2 → 3 → 2 → 3 → 2 → 3 → 2 → 3 → 4 → 5 → 8 → 9 → 10 → 11

match transition: scan to next input character and change state

ϵ-transition: change state with no match

accept state reached and all text characters scanned: pattern found



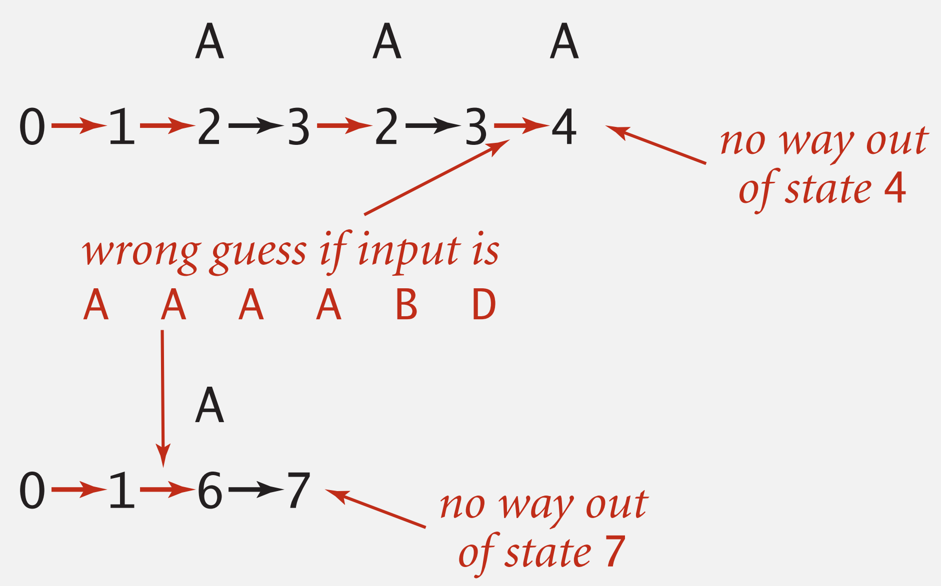
NFA corresponding to the pattern ((A * B | A C) D)

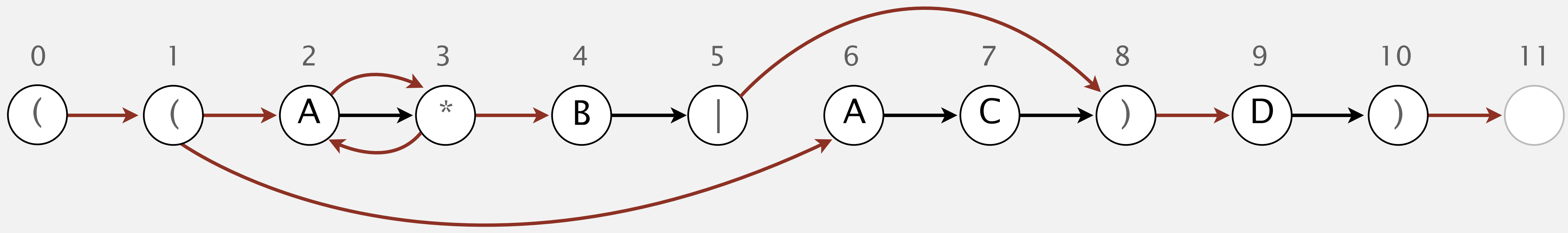
Nondeterministic finite-state automata

Q. Is A A A A B D matched by NFA?

A. Yes, because some sequence of legal transitions ends in state 11.

[even though some sequences end in wrong state or get stuck]





NFA corresponding to the pattern ((A * B | A C) D)

Nondeterministic finite-state automata

Q. Is A A A C matched by NFA?

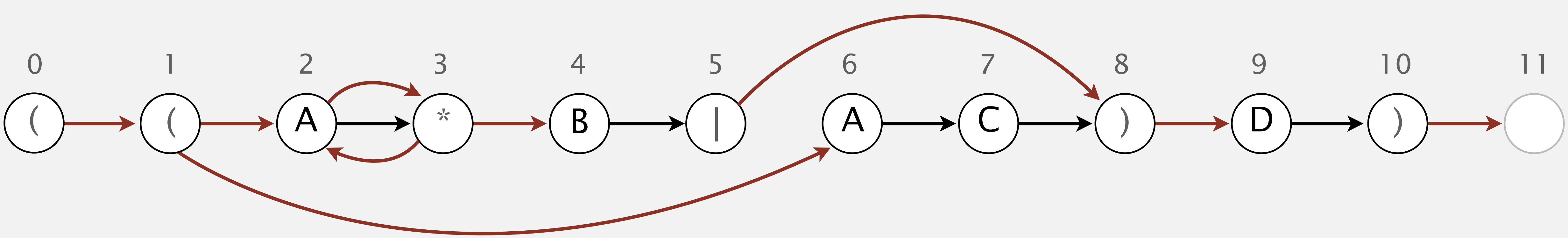
A. No, because no sequence of legal transitions ends in state 11.

[but need to argue about all possible sequences]

A A A A C

0 → 1 → 2 → 3 → 2 → 3 → 2 → 3 → 2 → 3 → 4

no way out of state 4



NFA corresponding to the pattern ((A * B | A C) D)

Nondeterminism

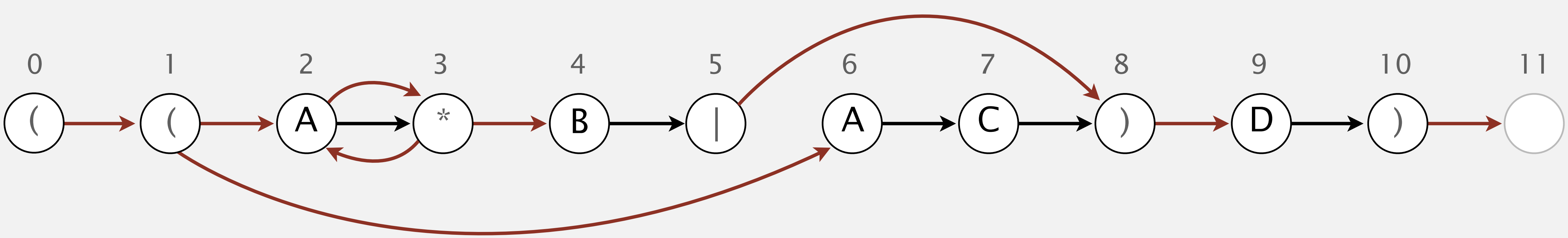
Q. How to determine whether a string is matched by an automaton?

DFA. Deterministic $\Rightarrow$ easy (only one applicable transition at each step).

NFA. Nondeterministic $\Rightarrow$ hard (can be several applicable transitions at each step; need to select the "right" ones!)

Q. How to simulate NFA?

A. Systematically consider all possible transition sequences. [stay tuned]



NFA corresponding to the pattern ((A * B | A C) D)

http://algs4.cs.princeton.edu

5.4 Regular Expressions

- *regular expressions*
- *REs and NFAs*
- *NFA simulation*
- *NFA construction*

NFA representation

State names. Integers from 0 to M.

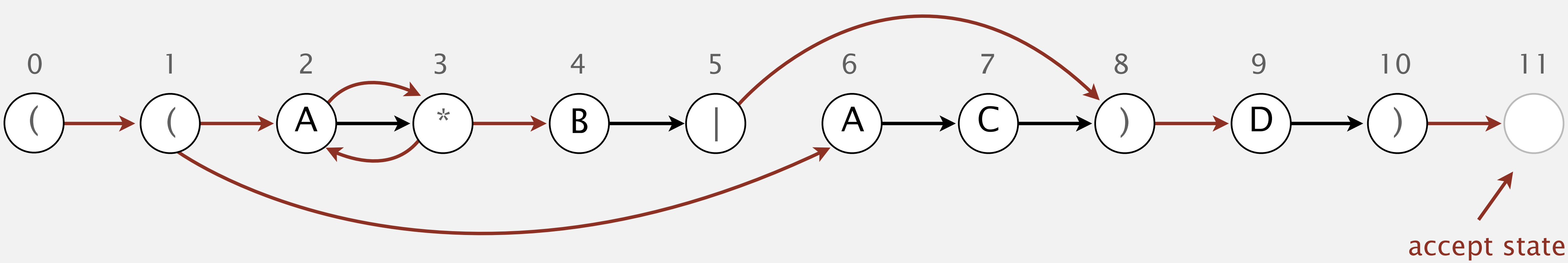
number of symbols in RE

Match-transitions. Keep regular expression in array `re[]`.

	0	1	2	3	4	5	6	7	8	9	10
re[]	(	(	A	*	B	|	A	C	)	D	)

ε-transitions. Store in a digraph G.

0→1, 1→2, 1→6, 2→3, 3→2, 3→4, 5→8, 8→9, 10→11

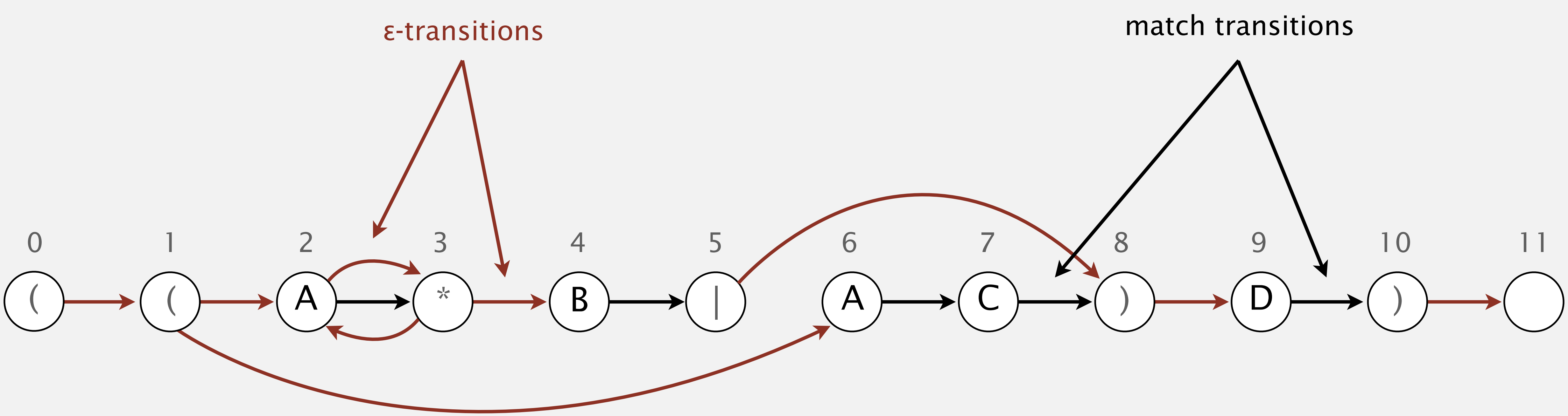


NFA corresponding to the pattern ((A * B | A C) D)

NFA simulation demo

Goal. Check whether input matches pattern.

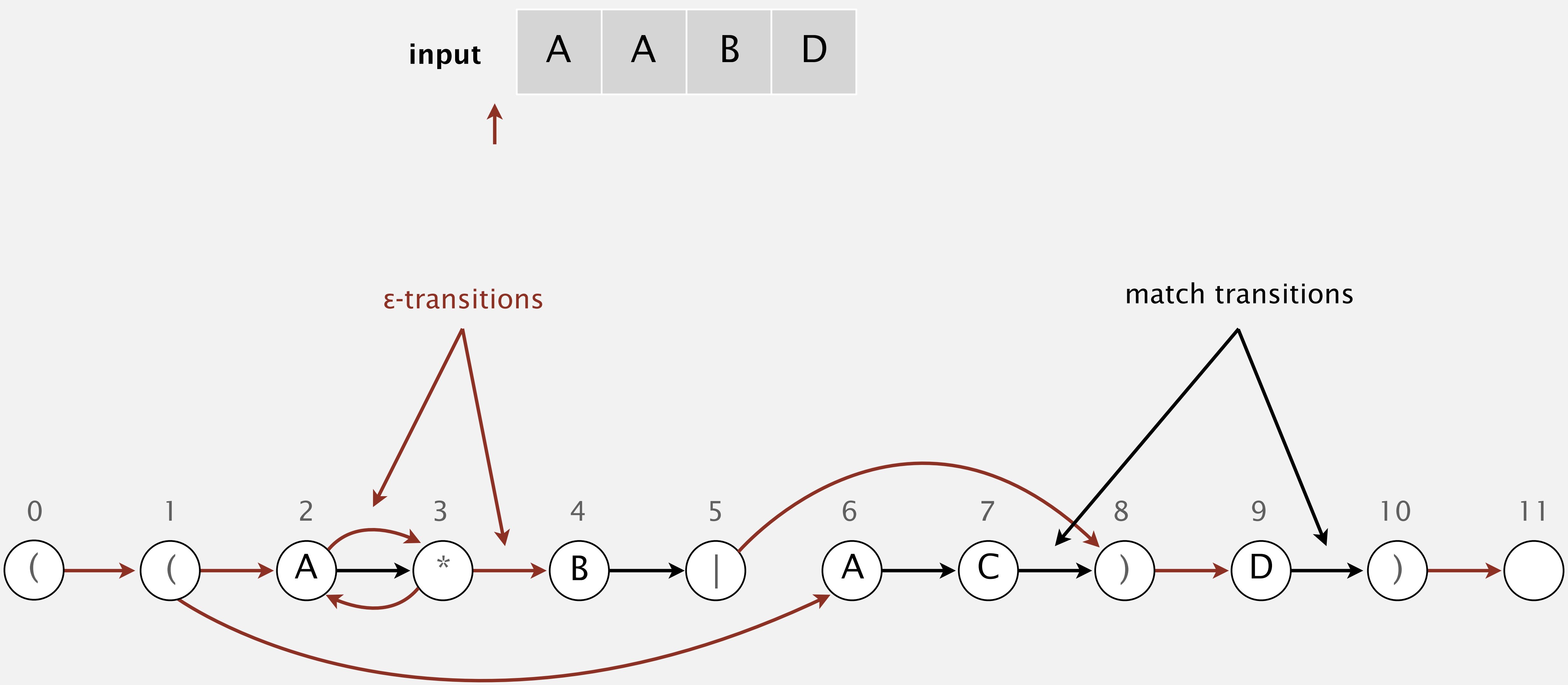
input	A	A	B	D



NFA corresponding to the pattern ((A * B | A C) D)

NFA simulation demo

Goal. Check whether input matches pattern.

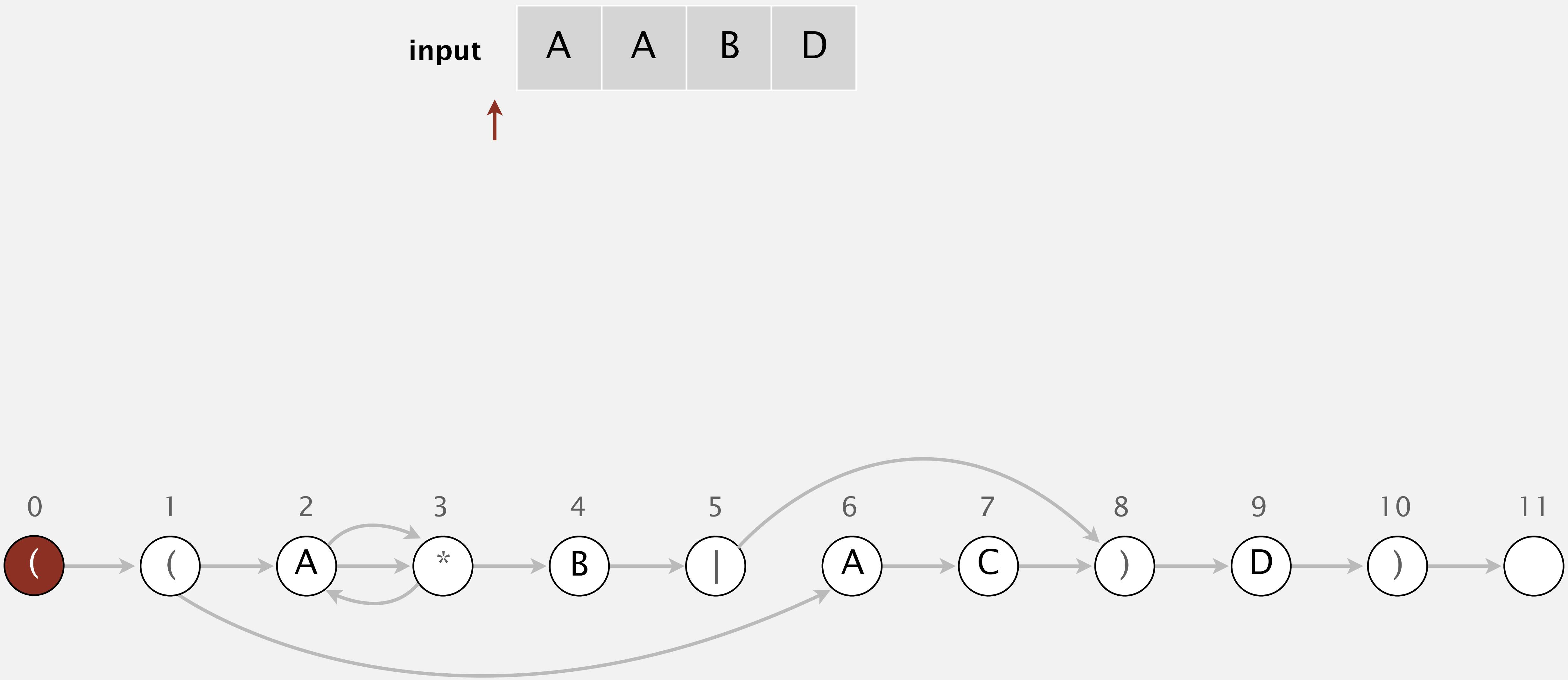


NFA corresponding to the pattern `( ( A * B | A C ) D )`

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

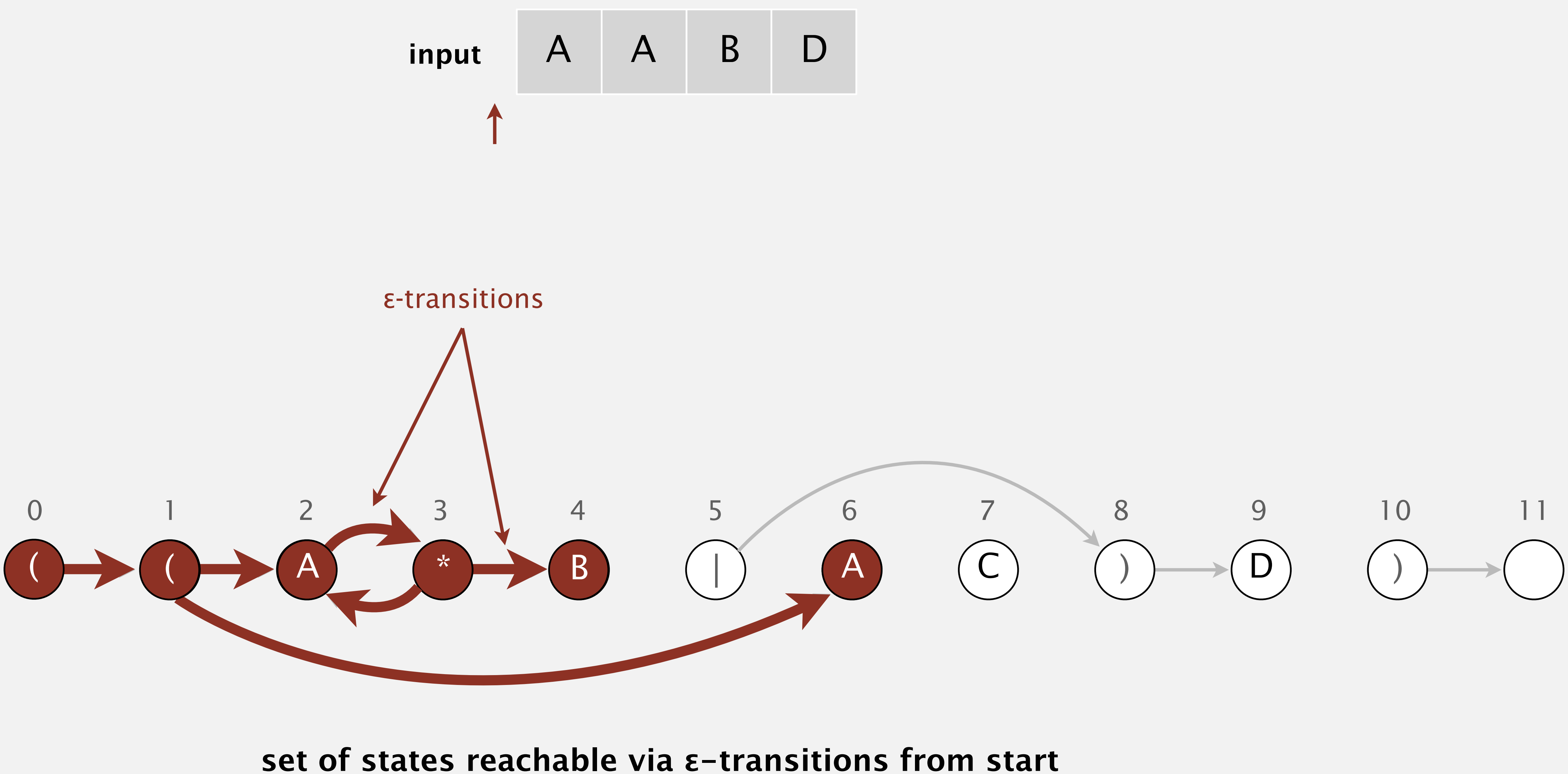


set of states reachable from start: 0

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

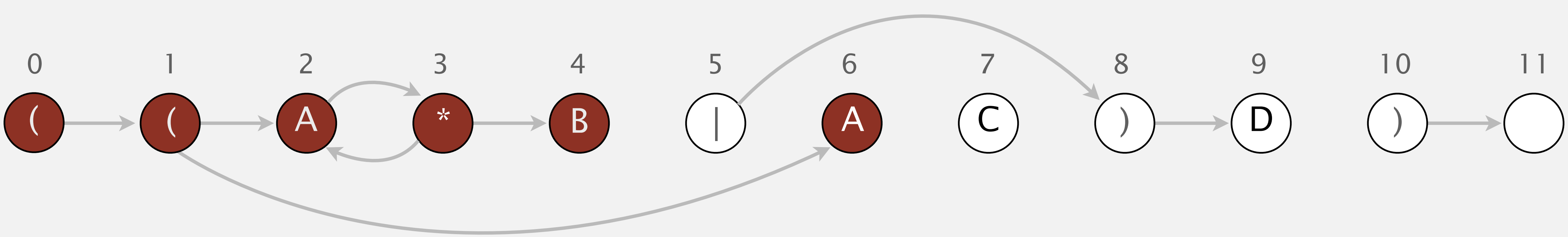


NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

input	A	A	B	D

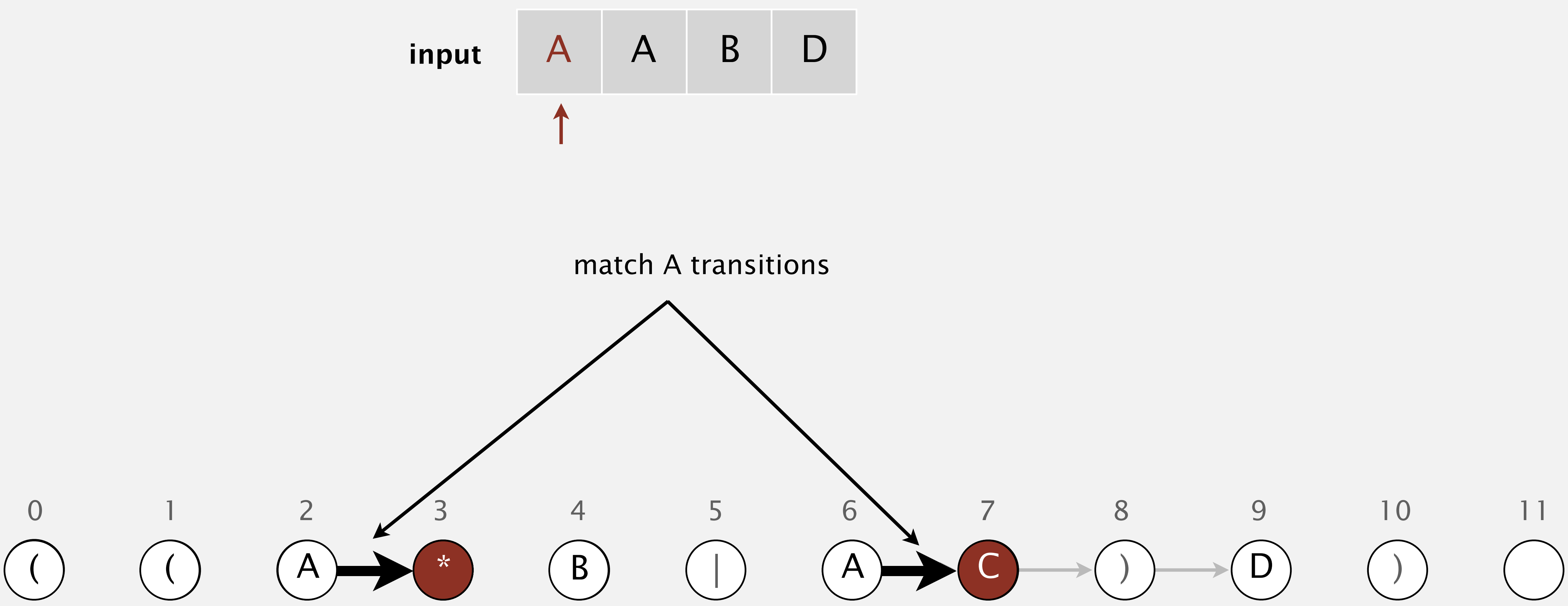


set of states reachable via ε−transitions from start : { 0, 1, 2, 3, 4, 6 }

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

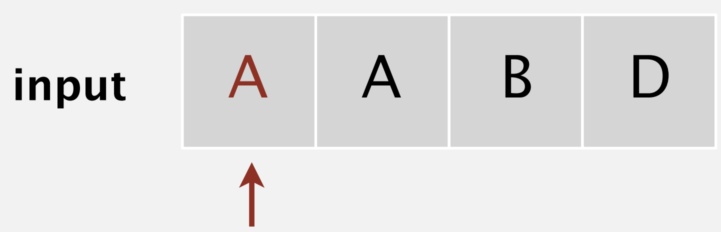


set of states reachable after matching A

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

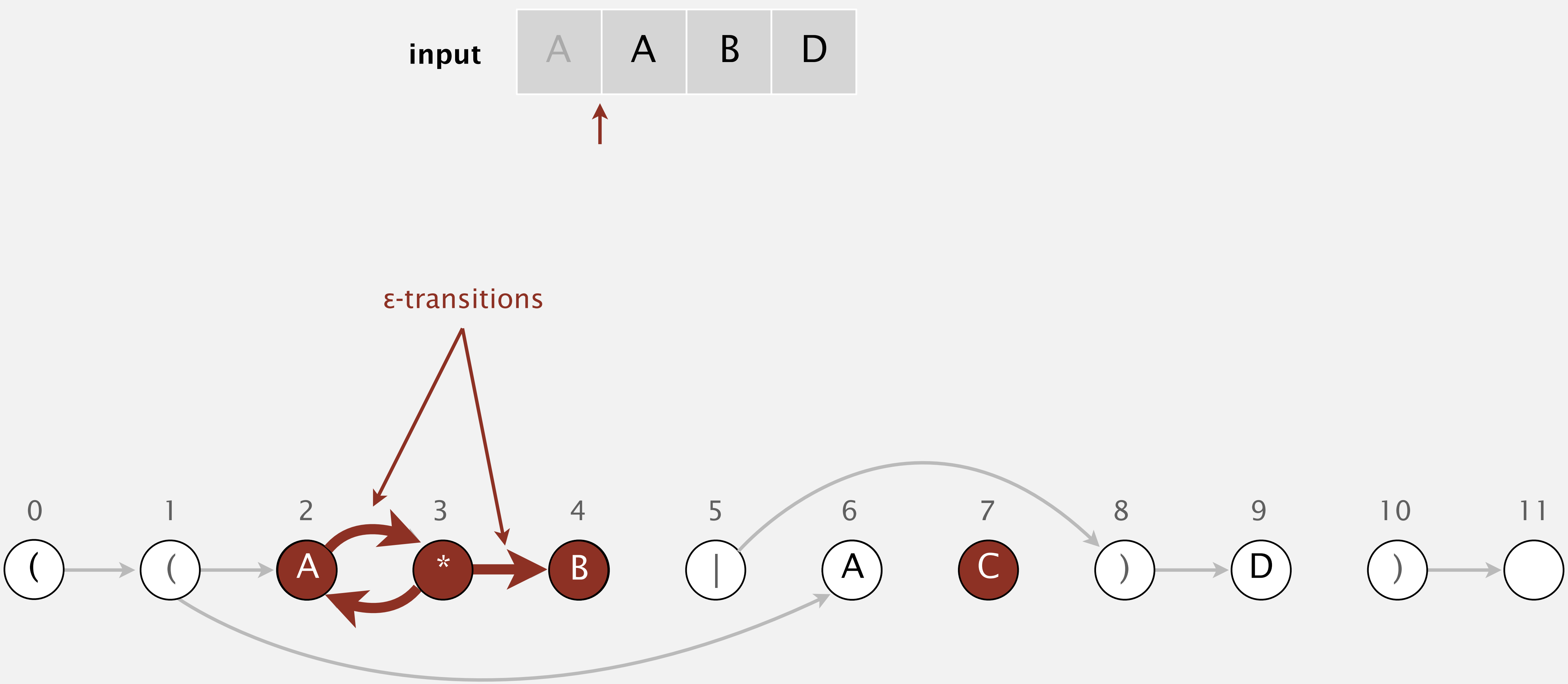


set of states reachable after matching A : { 3, 7 }

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

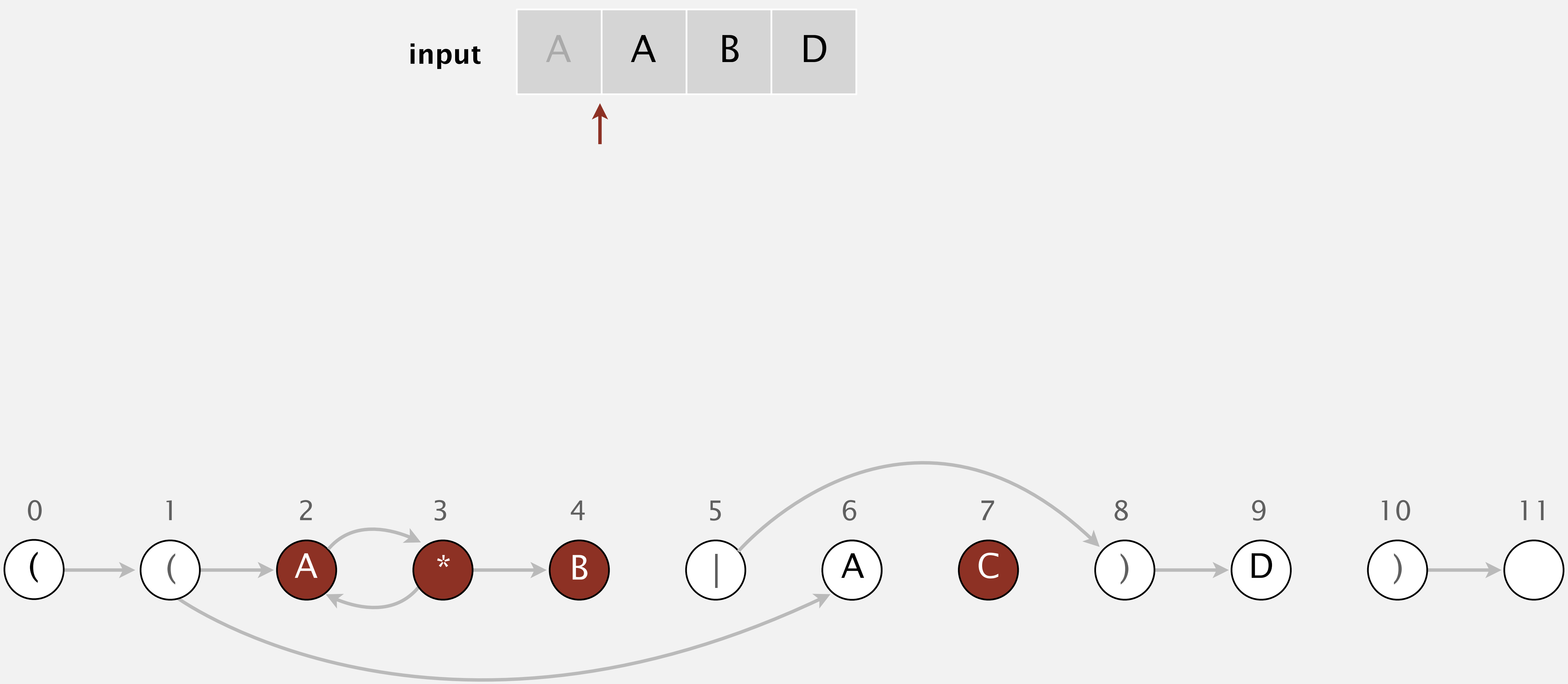


set of states reachable via ε-transitions after matching A

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

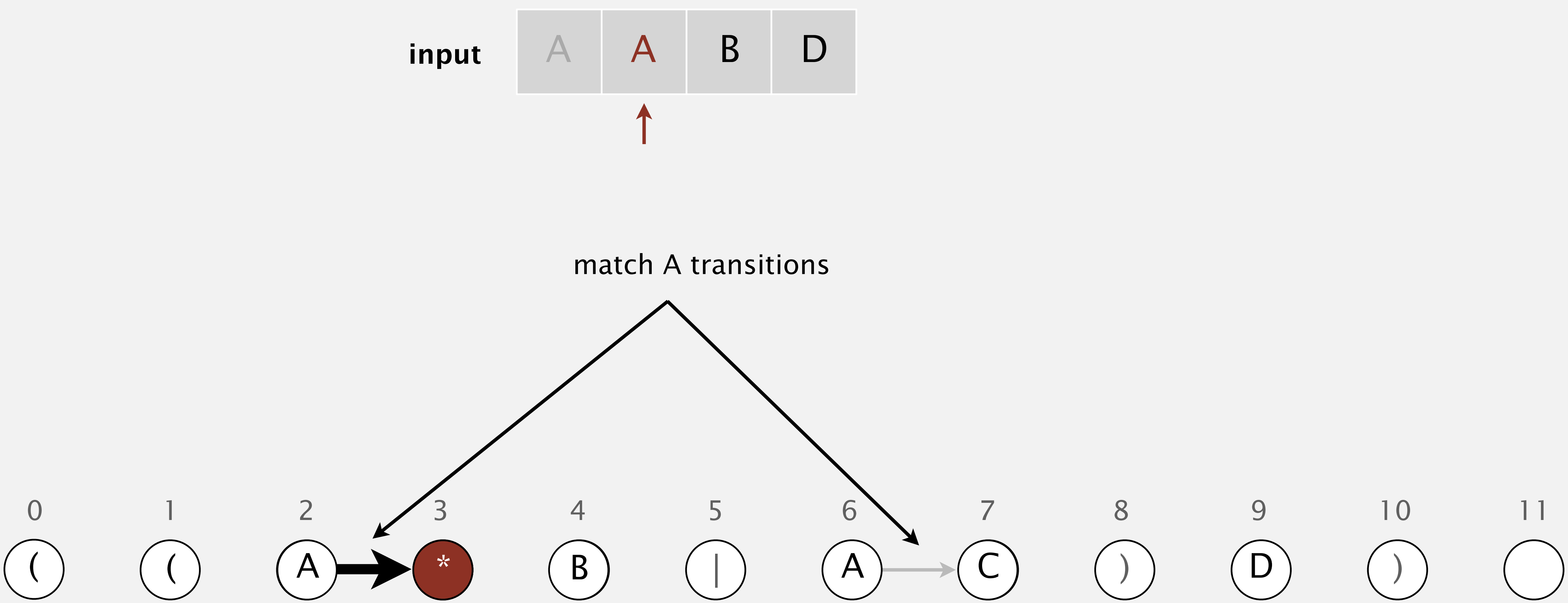


set of states reachable via ε-transitions after matching A : { 2, 3, 4, 7 }

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

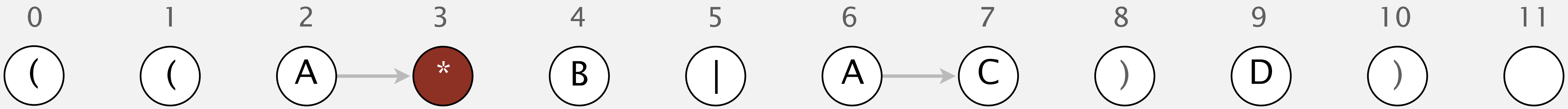


set of states reachable after matching A A

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

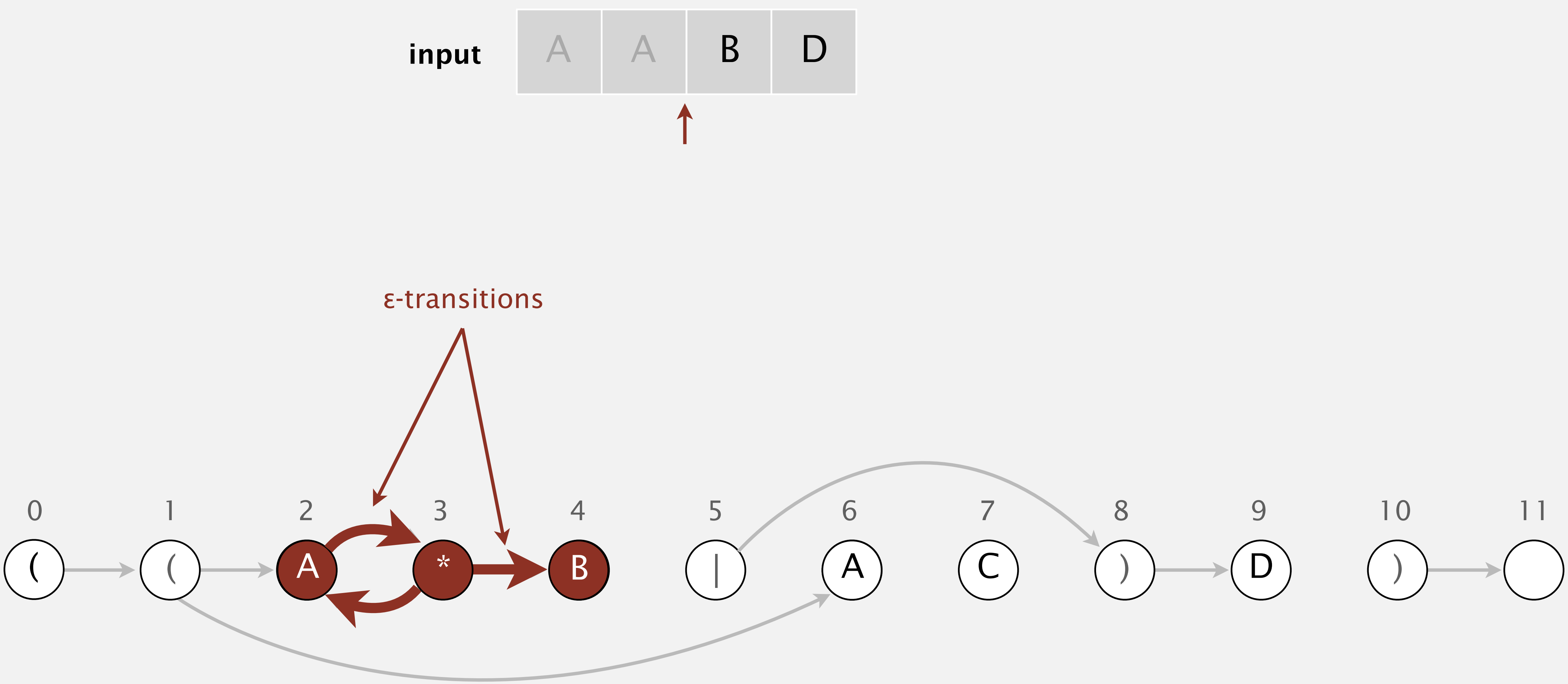


set of states reachable after matching A A : { 3 }

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions



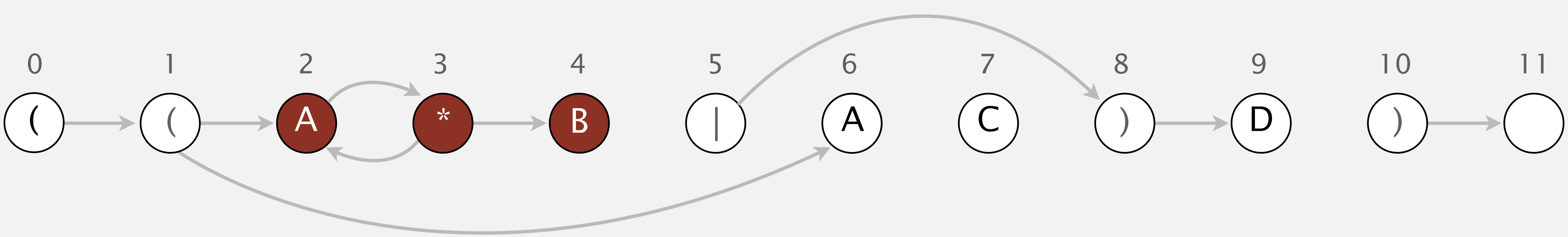
set of states reachable via ε-transitions after matching A A

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

input	A	A	B	D

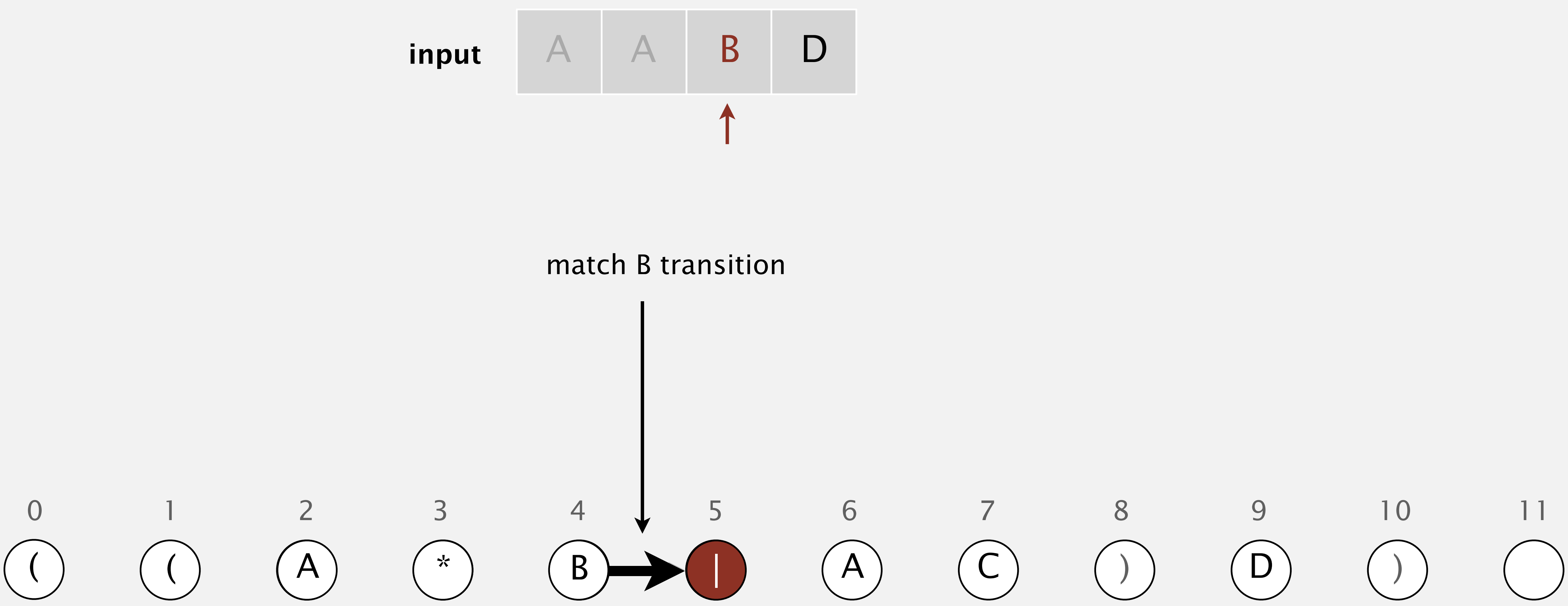


set of states reachable via ε-transitions after matching A A : { 2, 3, 4 }

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

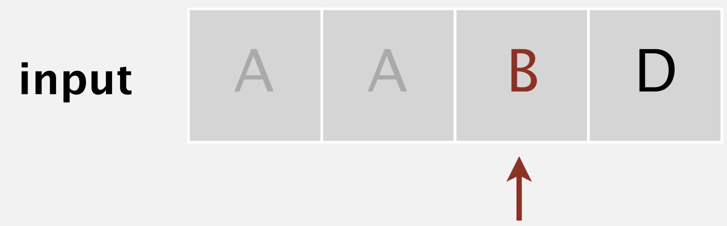


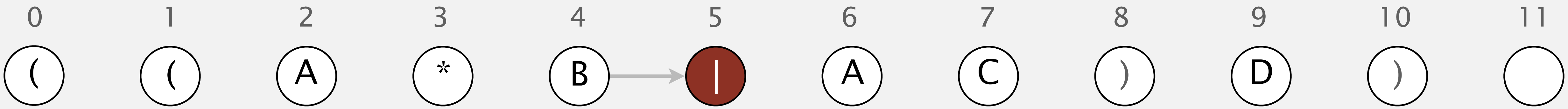
set of states reachable after matching A A B

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions



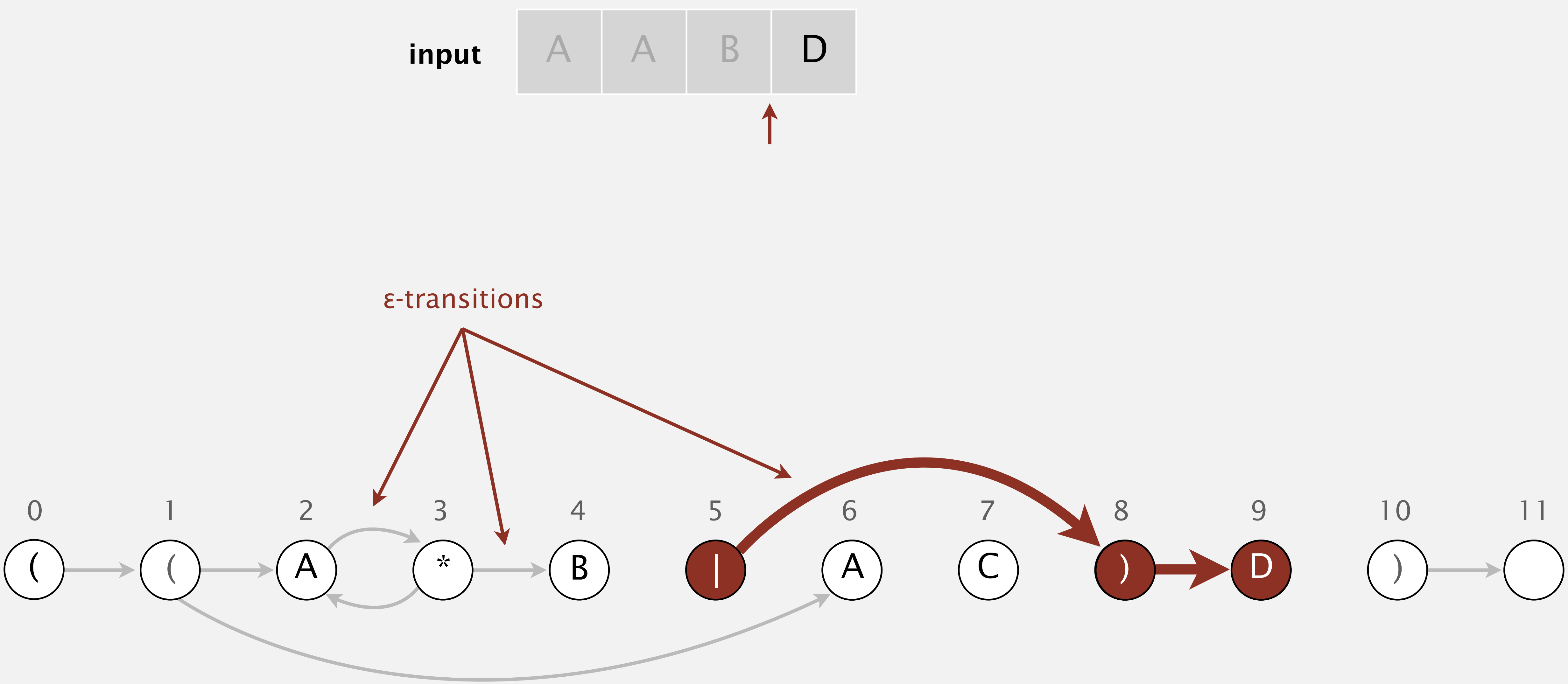


set of states reachable after matching A A B : { 5 }

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

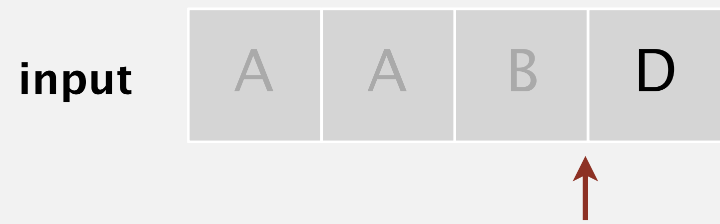


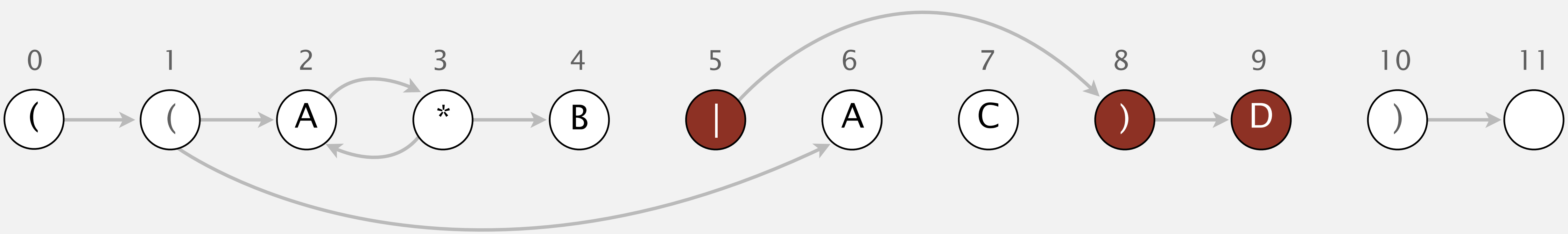
set of states reachable via ε-transitions after matching A A B

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions



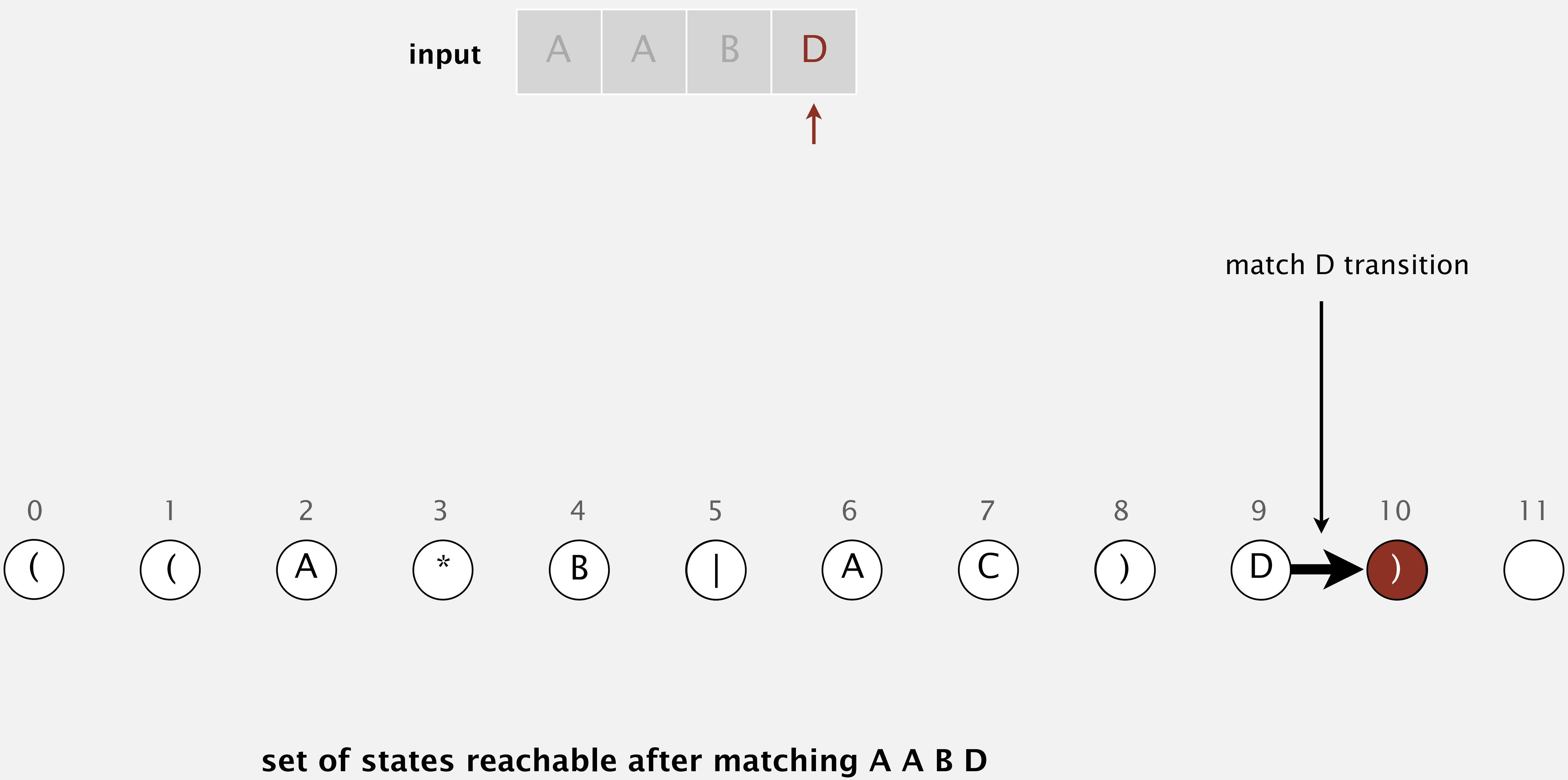


set of states reachable via ε-transitions after matching A A B : { 5, 8, 9 }

NFA simulation demo

Read next input character.

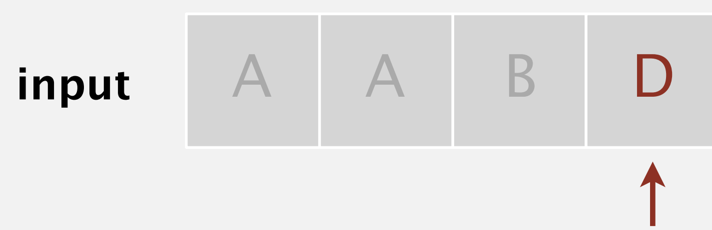
- Find states reachable by match transitions.
- Find states reachable by ε-transitions

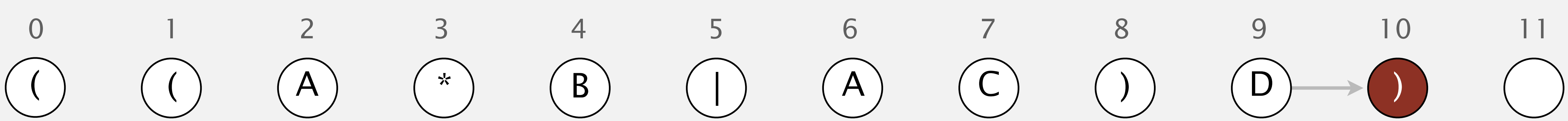


NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions



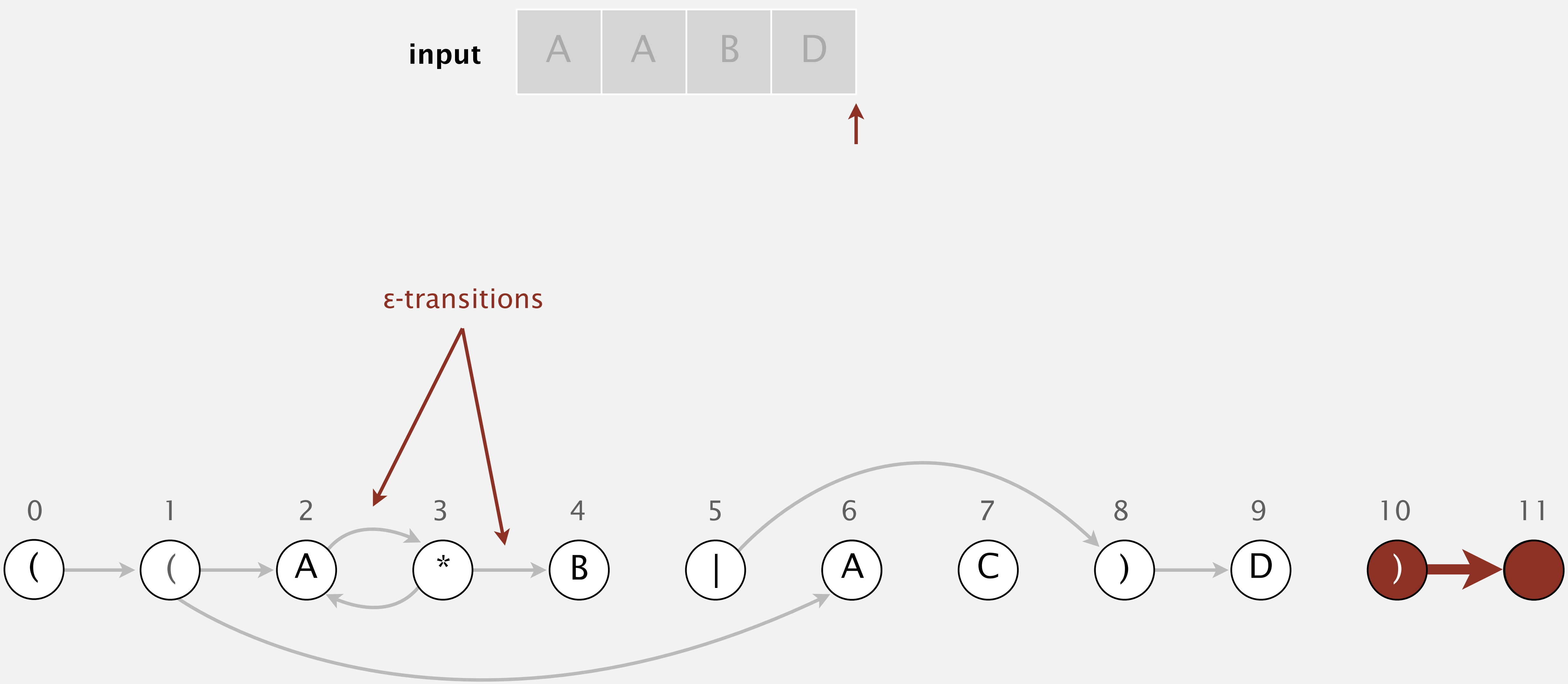


set of states reachable after matching A A B D : { 10 }

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

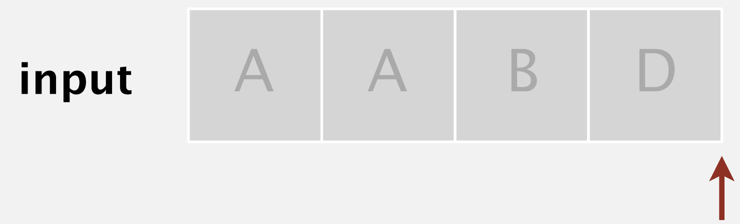


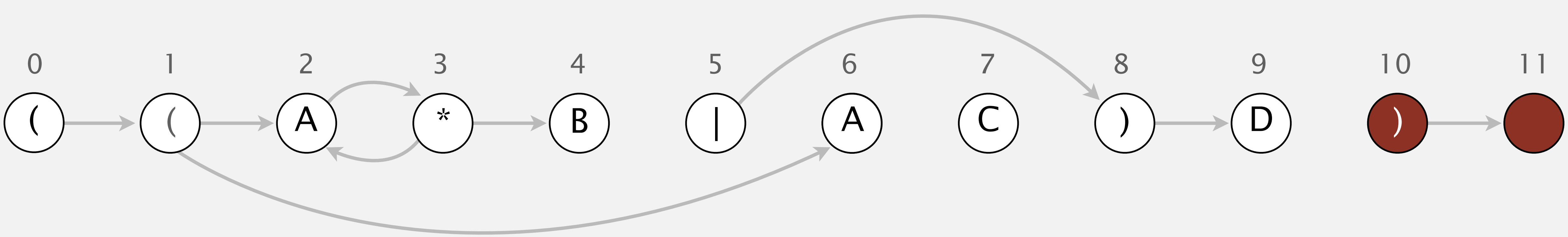
set of states reachable via ε-transitions after matching A A B D

NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions



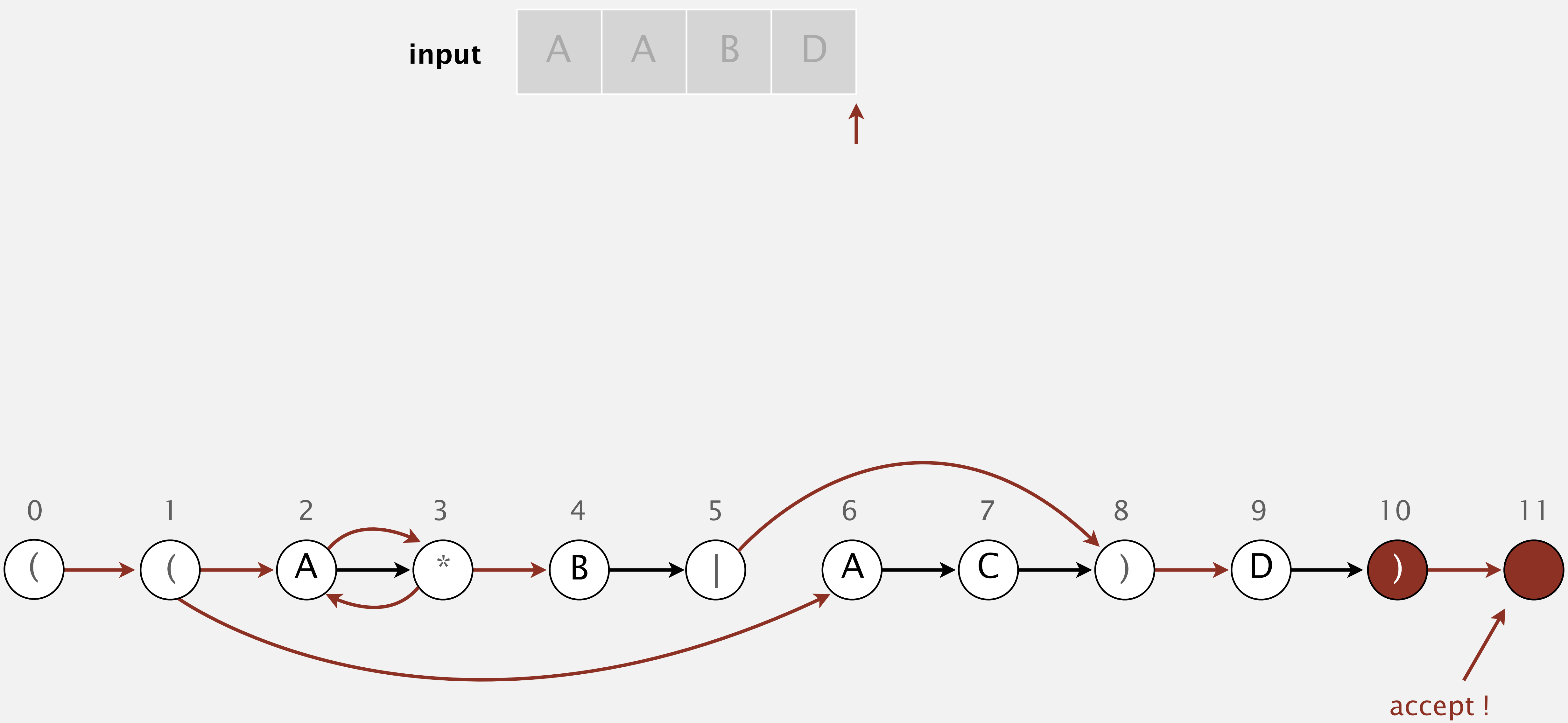


set of states reachable via ε-transitions after matching A A B D : { 10, 11 }

NFA simulation demo

When no more input characters:

- Accept if any state reachable is an accept state.
- Reject otherwise.



set of states reachable : { 10, 11 }

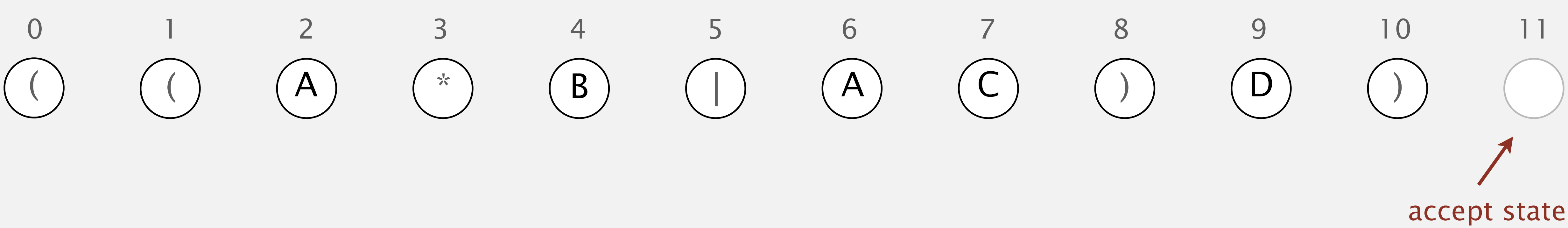
http://algs4.cs.princeton.edu

5.4 Regular Expressions

- *regular expressions*
- *REs and NFAs*
- *NFA simulation*
- *NFA construction*

Building an NFA corresponding to an RE

States. Include a state for each symbol in the RE, plus an accept state.



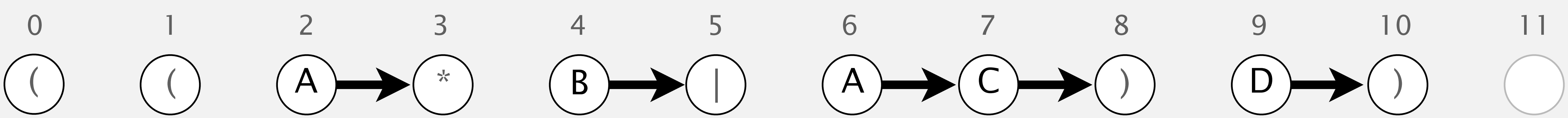
NFA corresponding to the pattern ((A * B | A C) D)

Building an NFA corresponding to an RE

Concatenation. Add match-transition edge from state corresponding to characters in the alphabet to next state.

Alphabet. A B C D

Metacharacters. ( ) . * |



NFA corresponding to the pattern ((A * B | A C) D)

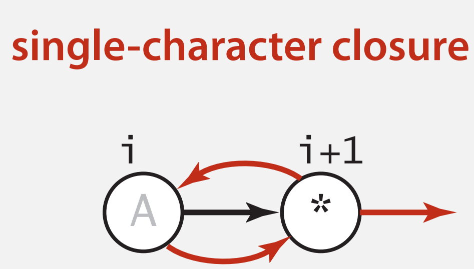
Building an NFA corresponding to an RE

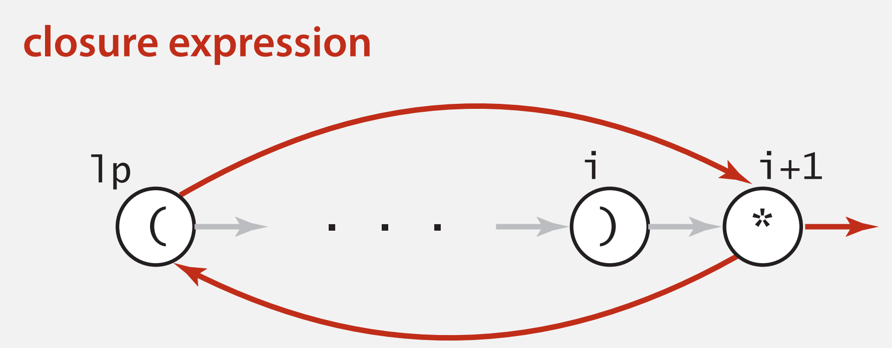
Parentheses. Add ε-transition edge from parentheses to next state.

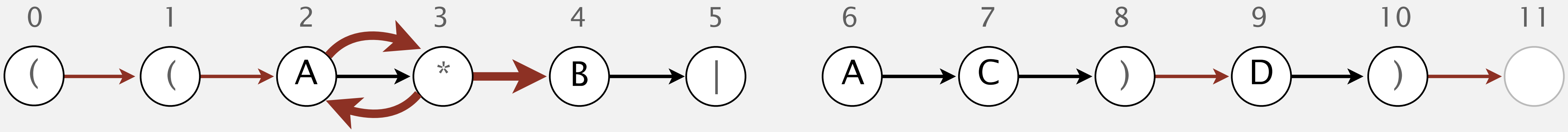
NFA corresponding to the pattern ((A * B | A C) D)

Building an NFA corresponding to an RE

Closure. Add three ε-transition edges for each * operator.



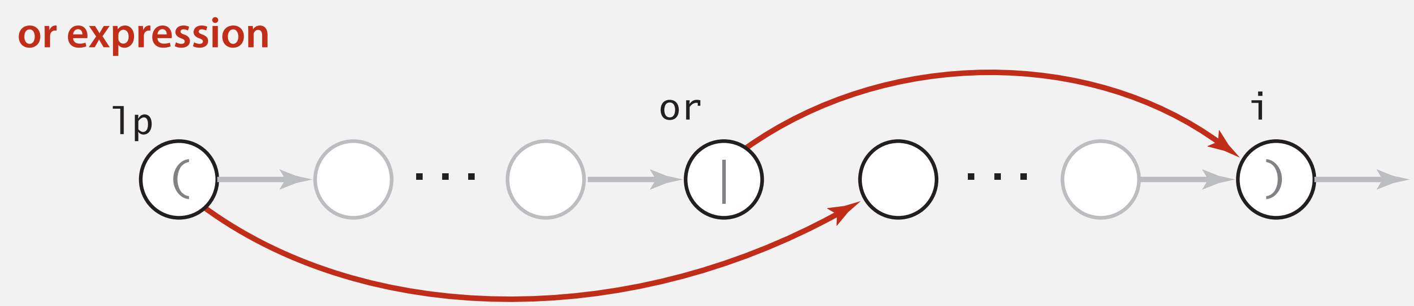


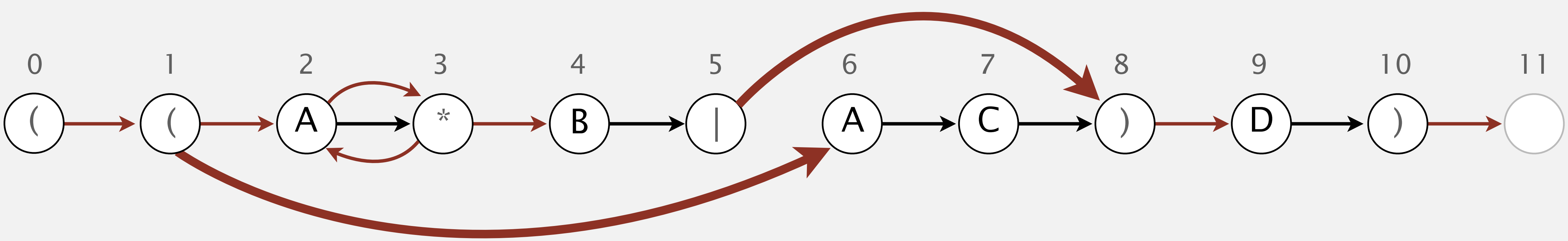


NFA corresponding to the pattern ((A * B | A C) D)

Building an NFA corresponding to an RE

2-way or. Add two ε-transition edges for each | operator.





NFA corresponding to the pattern ((A * B | A C) D)

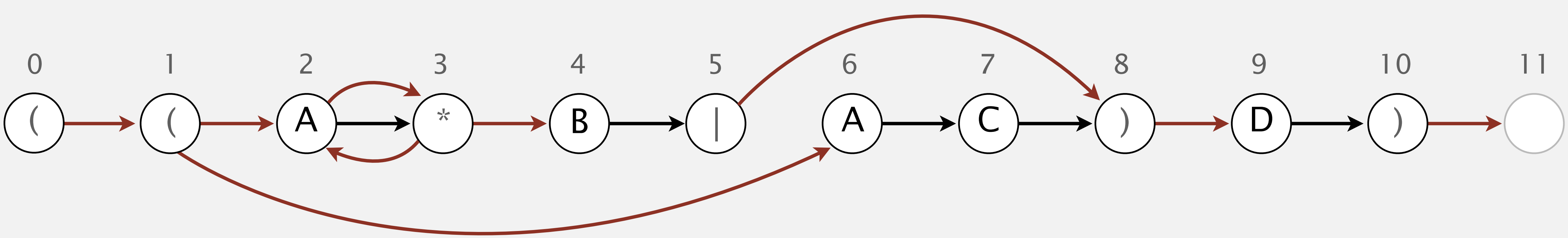
NFA construction: implementation

Goal. Write a program to build the ε-transition digraph.

Challenges. Remember left parentheses to implement closure and or; remember | symbols to implement or.

Solution. Maintain a stack.

- (symbol: push (onto stack.
- | symbol: push | onto stack.
-) symbol: pop corresponding (and any intervening |; add ε-transition edges for closure/or.



NFA corresponding to the pattern ((A * B | A C) D)

NFA construction demo

stack

((A * B | A C) D)

NFA construction demo

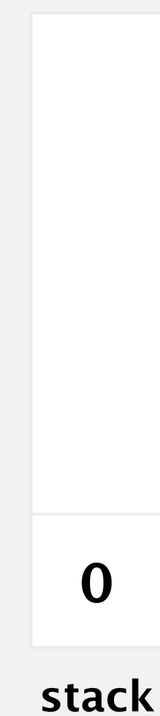
stack

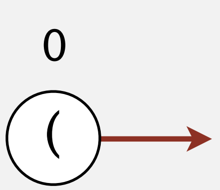
((A * B | A C) D)

NFA construction demo

Left parenthesis.

- Add ε-transition to next state.
- Push index of state corresponding to (onto stack.



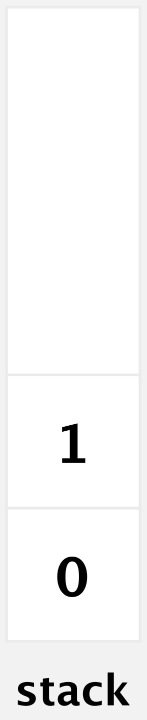


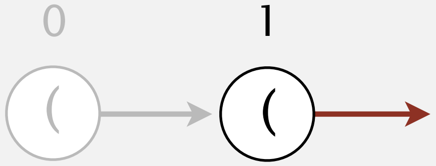
((A * B | A C) D)

NFA construction demo

Left parenthesis.

- Add ε-transition to next state.
- Push index of state corresponding to (onto stack.



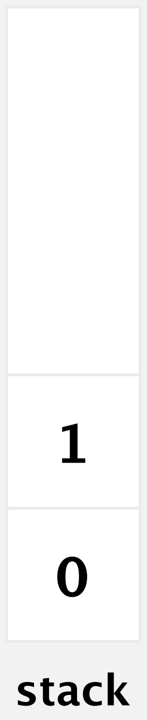


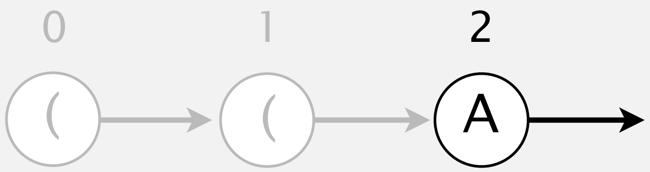
((A * B | A C) D)

NFA construction demo

Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add ε-transitions if next character is *.



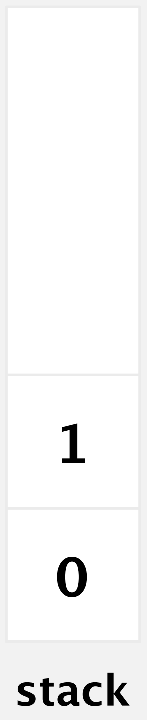


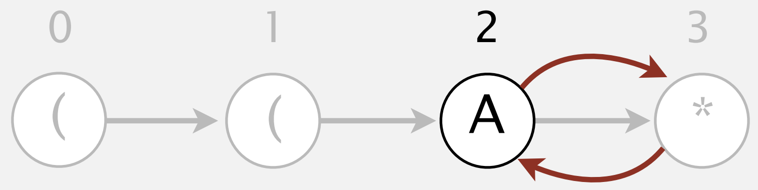
((A * B | A C) D)

NFA construction demo

Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add ε-transitions if next character is *.



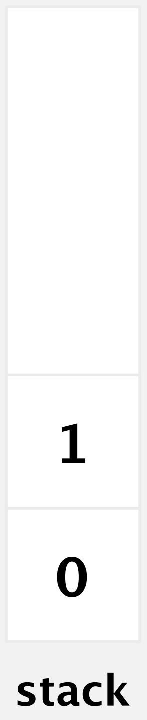


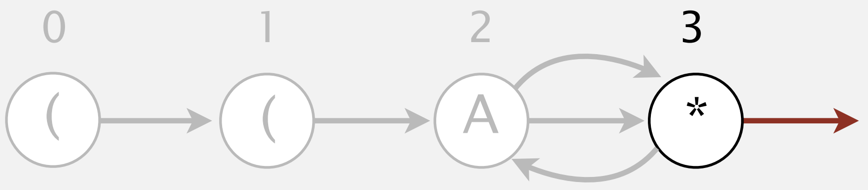
((A * B | A C) D)

NFA construction demo

Closure symbol.

- Add ε-transition to next state.



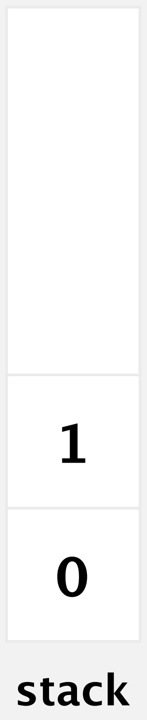


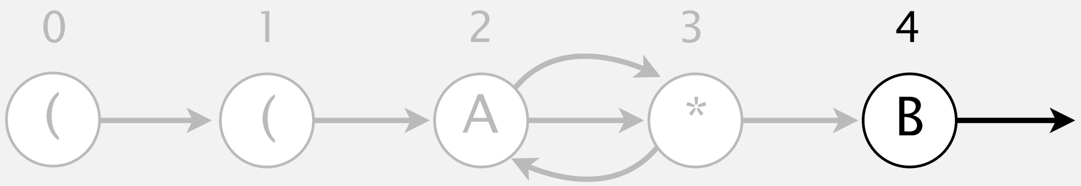
((A * B | A C) D)

NFA construction demo

Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add ε-transitions if next character is *.



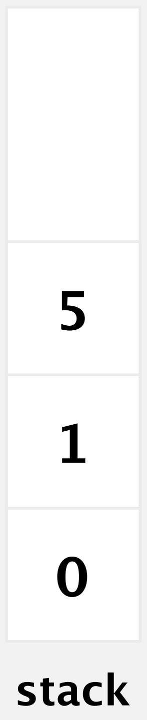


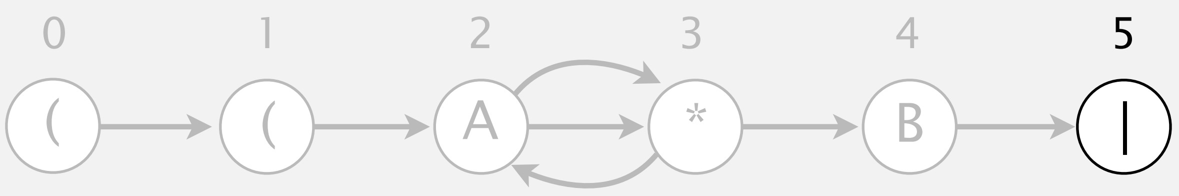
((A * B | A C) D)

NFA construction demo

Or symbol.

- Push index of state corresponding to | onto stack.



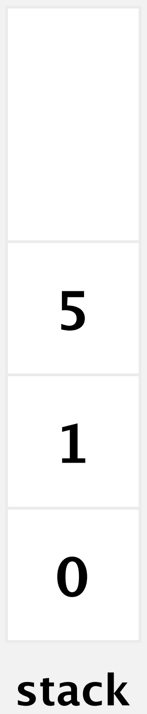


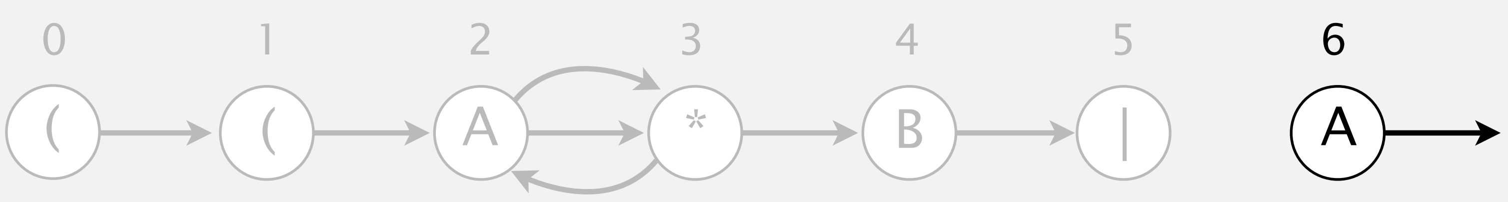
((A * B | A C) D)

NFA construction demo

Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add ε-transitions if next character is *.



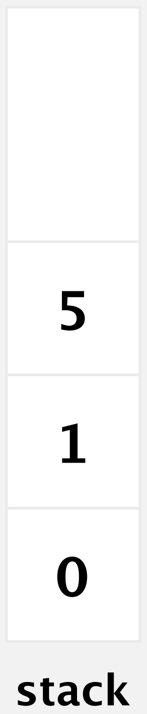


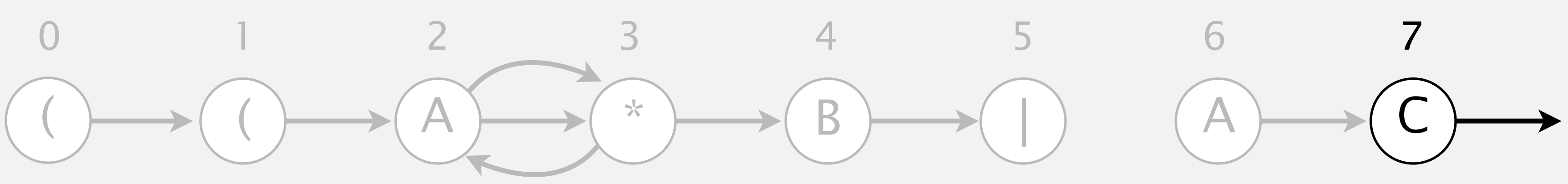
((A * B | A C) D)

NFA construction demo

Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add ε-transitions if next character is *.



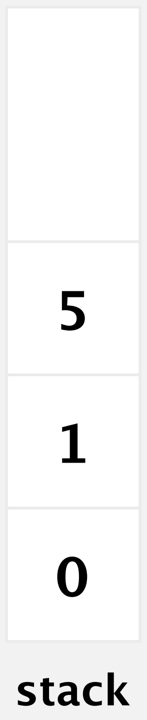


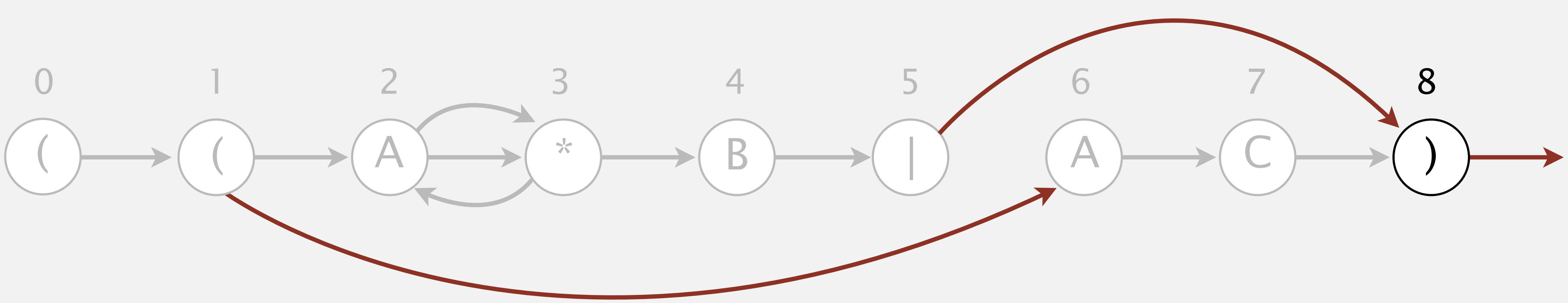
((A * B | A C) D)

NFA construction demo

Right parenthesis.

- Add ε-transition to next state.
- Pop corresponding (and any intervening |; add ε-transition edges for or.
- Do one-character lookahead: add ε-transitions if next character is *.

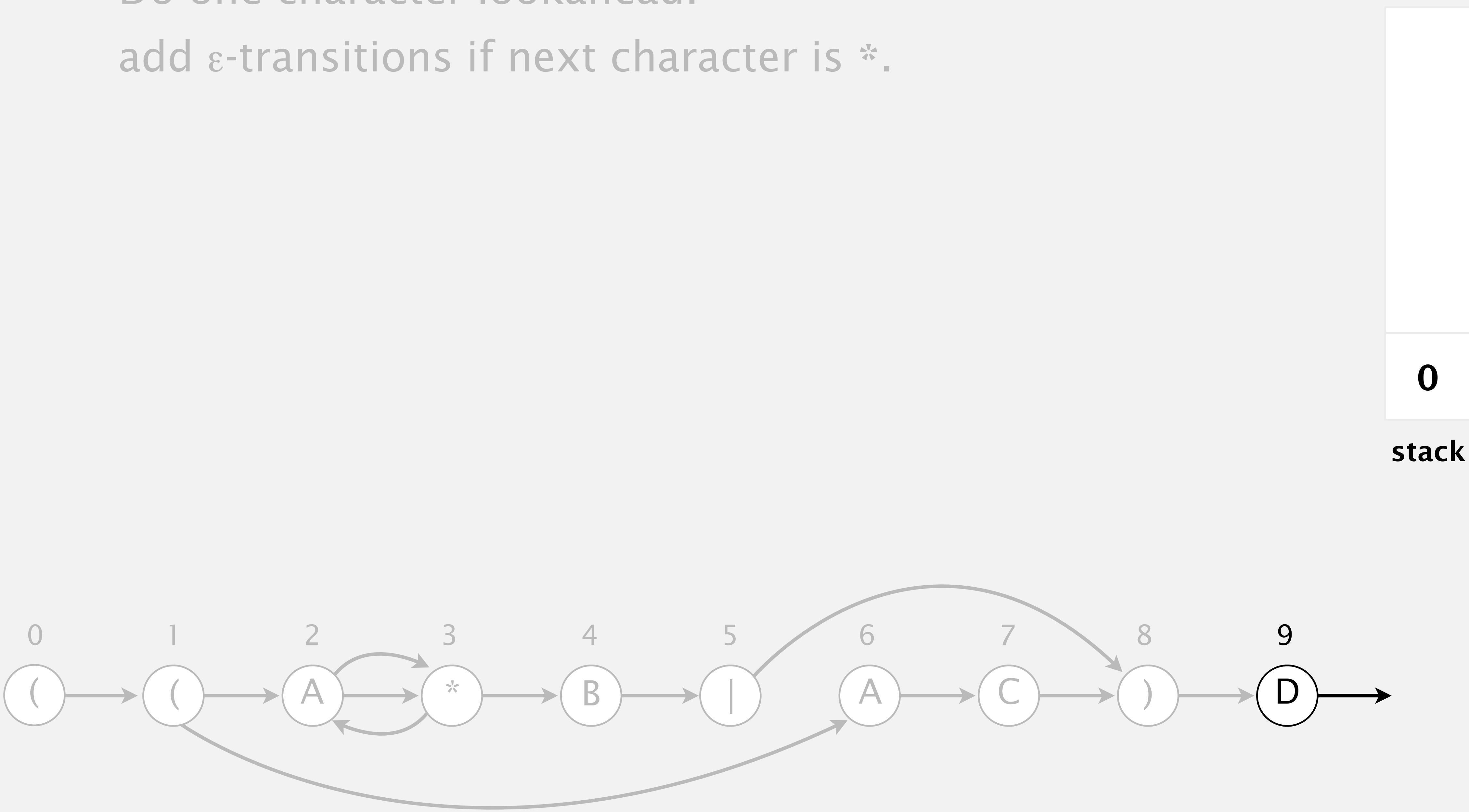




NFA construction demo

Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add ε-transitions if next character is *.

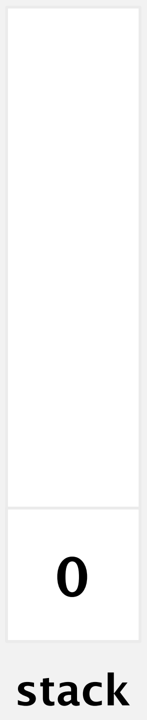


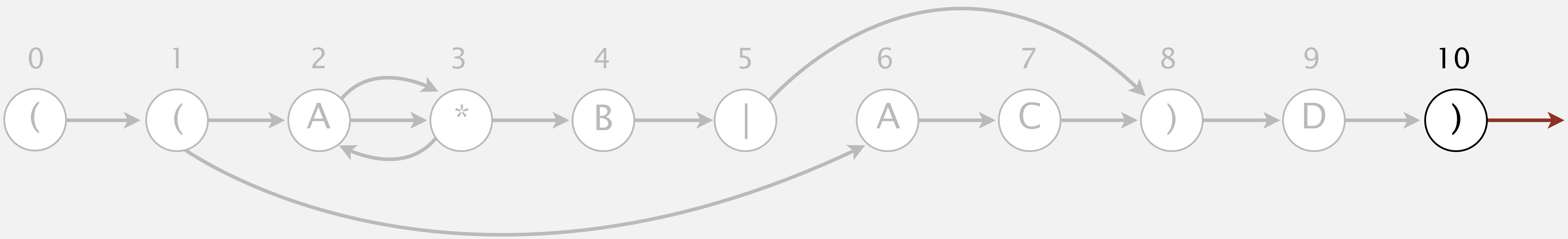
((A * B | A C) D)

NFA construction demo

Right parenthesis.

- Add ε-transition to next state.
- Pop corresponding (and any intervening |; add ε-transition edges for or.
- Do one-character lookahead: add ε-transitions if next character is *.



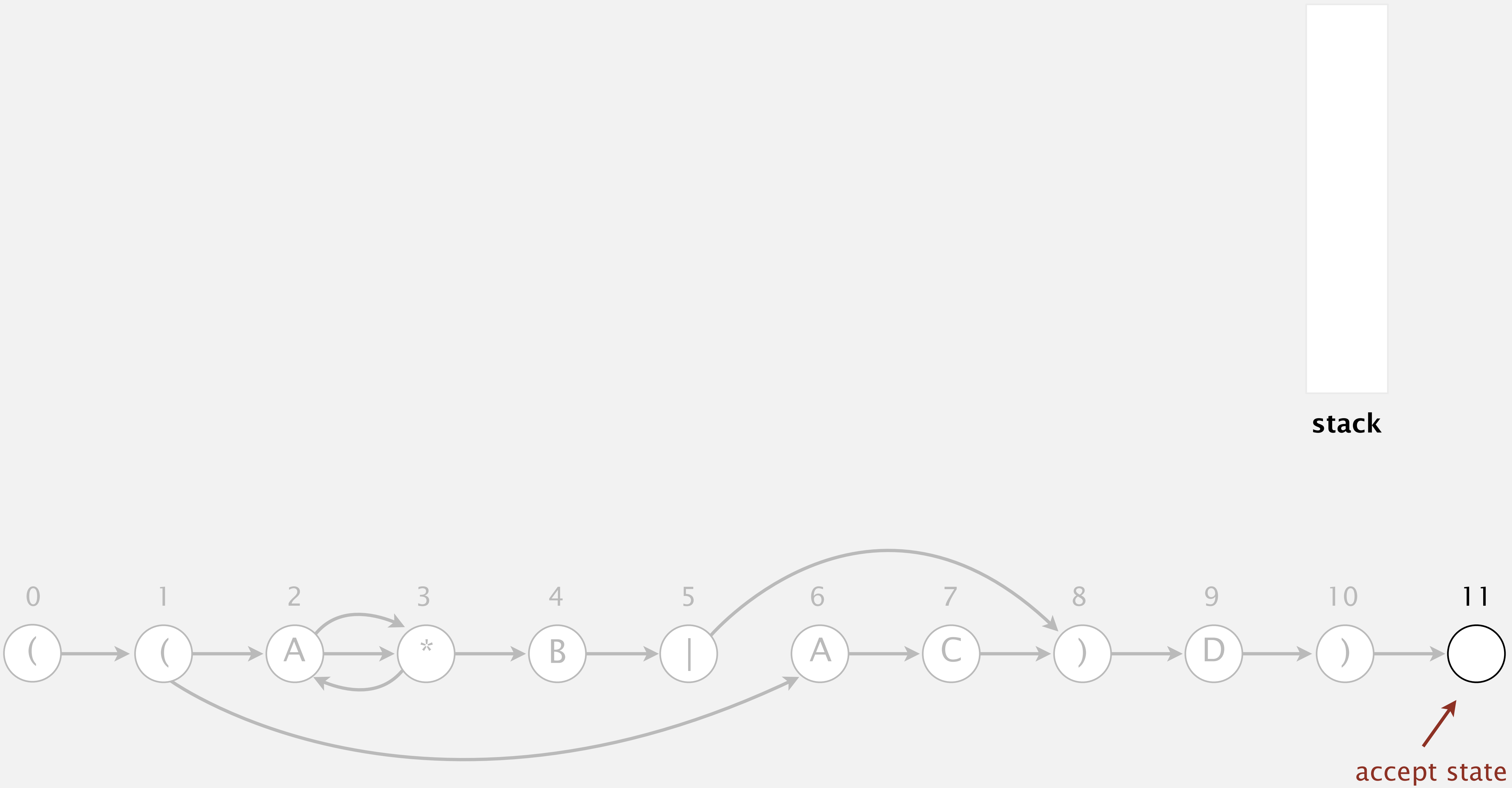


((A * B | A C) D)

NFA construction demo

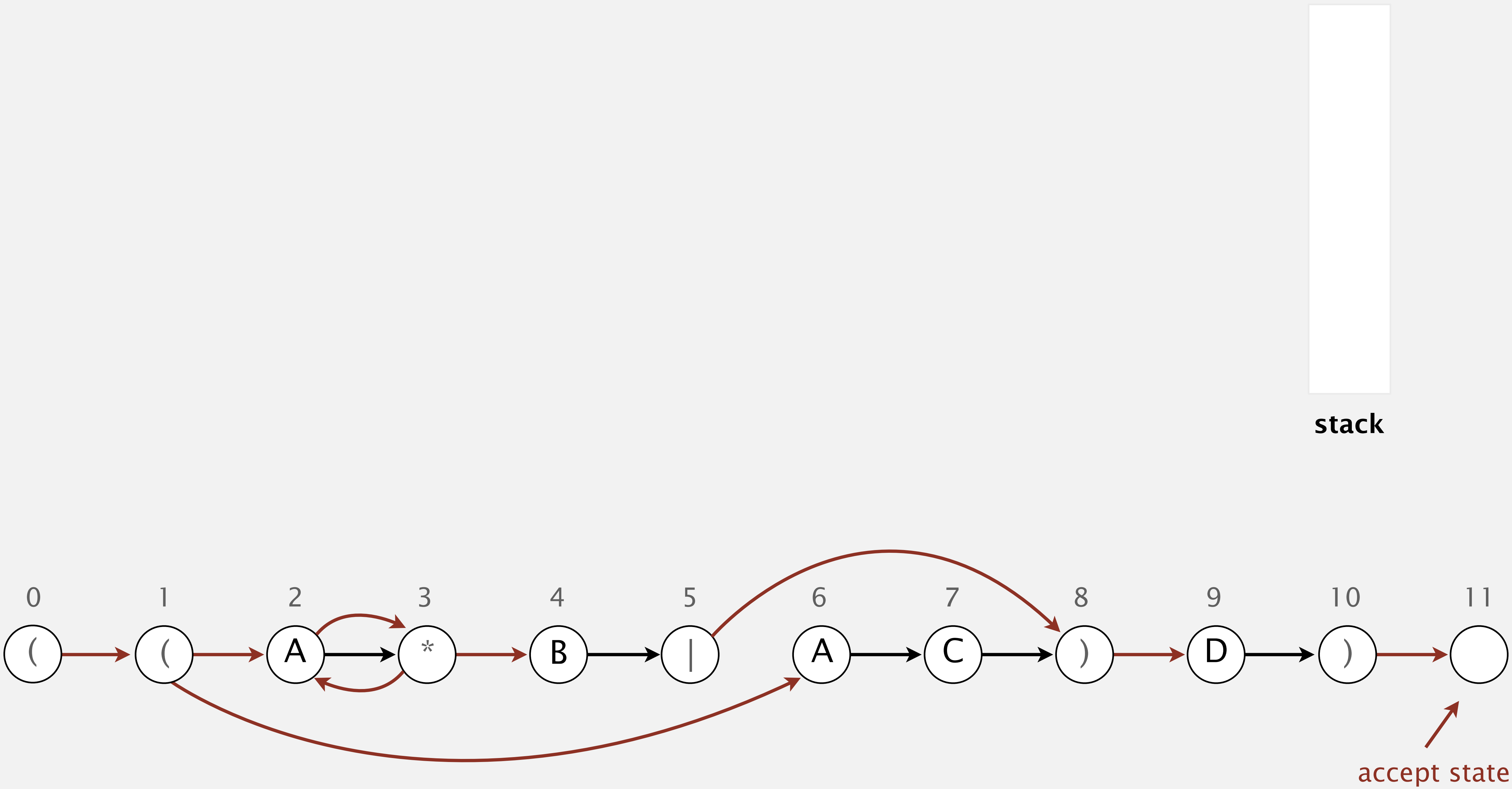
End of regular expression.

- Add accept state.



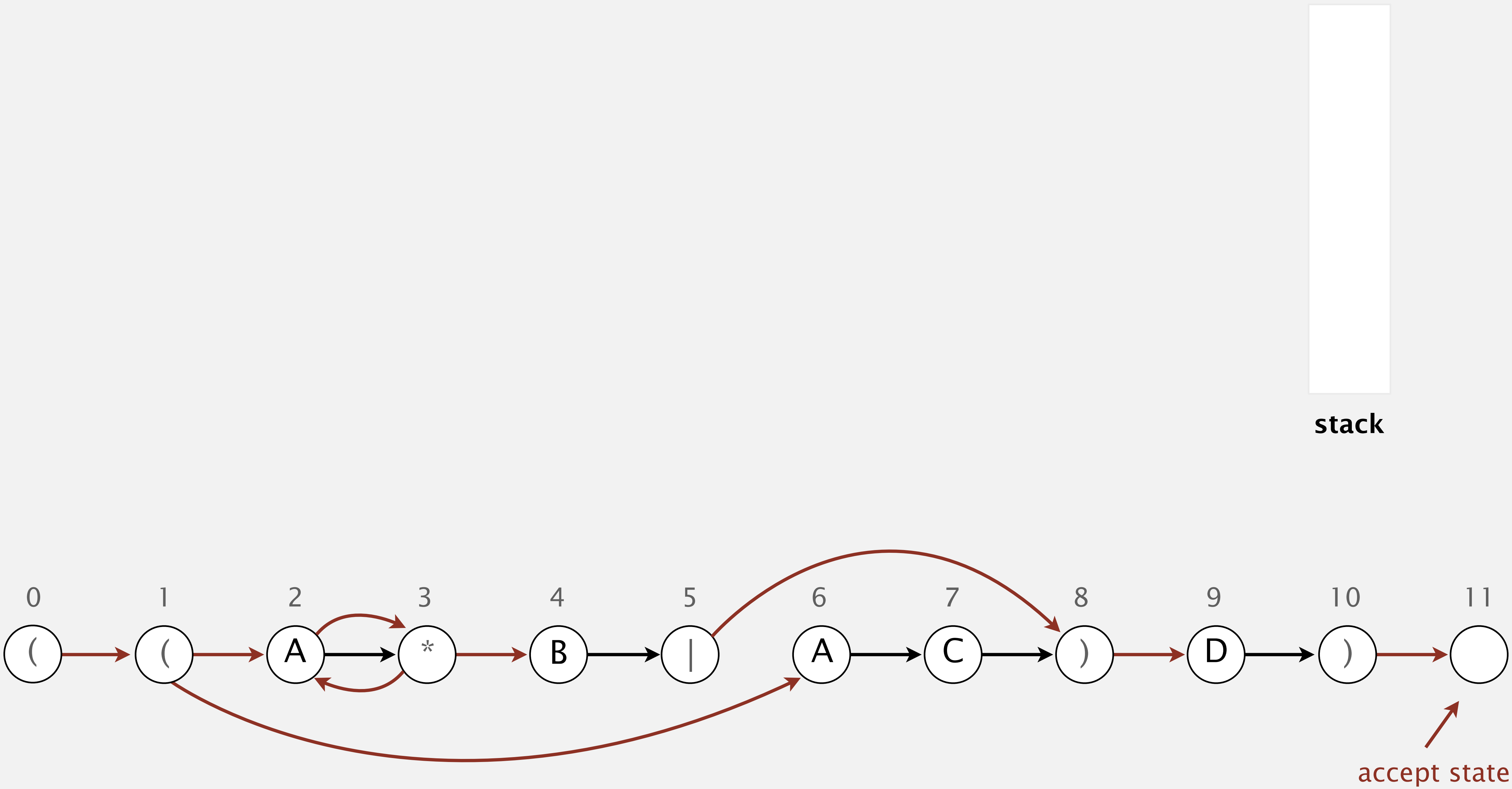
((A * B | A C) D)

NFA construction demo



NFA corresponding to the pattern ((A * B | A C) D)

NFA construction demo



NFA corresponding to the pattern `( ( A * B | A C ) D )`

Context

Abstract machines, languages, and nondeterminism.

- Basis of the theory of computation.
- Intensively studied since the 1930s.
- Basis of programming languages.

Compiler. A program that translates a program to machine code.

- KMP string $\Rightarrow$ DFA.
- `grep` RE $\Rightarrow$ NFA.
- `javac` Java language $\Rightarrow$ Java byte code.

	KMP	grep	Java
pattern	string	RE	program
parser	unnecessary	check if legal	check if legal
compiler output	DFA	NFA	byte code
simulator	DFA simulator	NFA simulator	JVM

Summary of pattern-matching algorithms

Programmer.

- Implement substring search via DFA simulation.
- Implement RE pattern matching via NFA simulation.

Theoretician.

- RE is a compact description of a set of strings.
- NFA is an abstract machine equivalent in power to RE.
- DFAs, NFAs, and REs have limitations.

You.

- Core CS principles provide useful tools that you can exploit now.
- REs and NFAs provide introduction to theoretical CS.

Example of essential paradigm in computer science.

- Build the right intermediate abstractions.
- Solve important practical problems.