Scanner

Ronghui Gu
Spring 2022

Columbia University

* Course website: https://verigu.github.io/4115Spring2022/

** These slides are borrowed from Prof. Edwards.

The Big Picture

The First Question

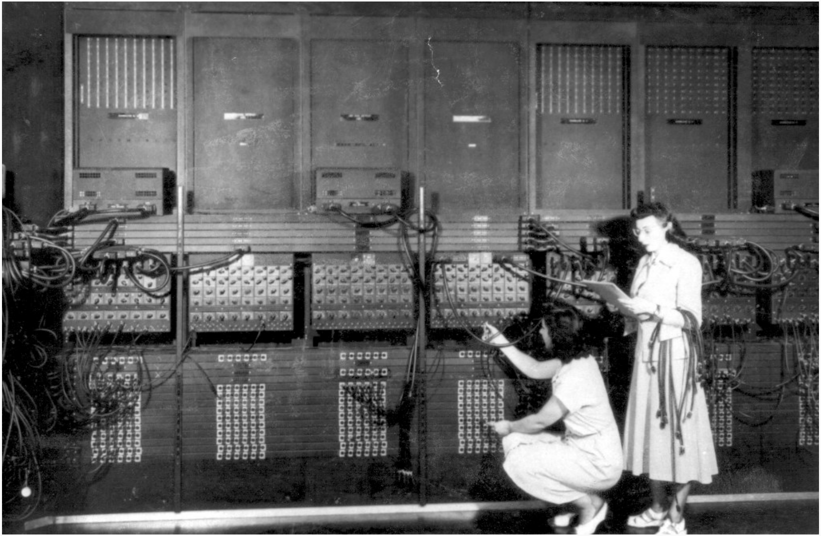
How do we describe/construct a program?

Use continuously varying values?

Very efficient, but has serious noise issues

Edison Model B Home Cylinder phonograph, 1906

The ENIAC: Programming with Spaghetti



Have one symbol per program?

Works nicely when there are only a few things

Sholes and Glidden Typewriter, E. Remington and Sons, 1874

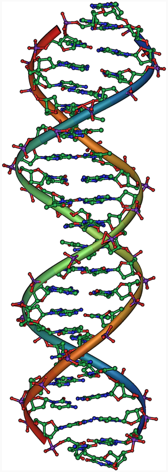
Have one symbol per program?

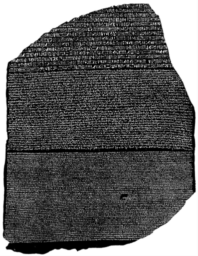
Not so good when there are many, many things

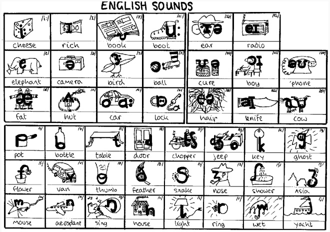
Nippon Typewriter SH-280, 2268 keys

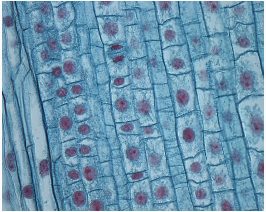
Solution: Use a Discrete Combinatorial System

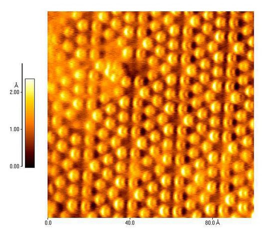
Use combinations of a small number of things to represent (exponentially) many different things.





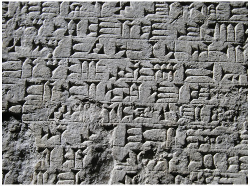




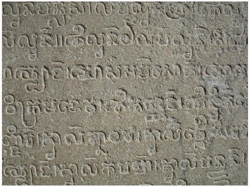


Every Human Writing System Does This

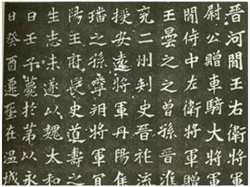
Hieroglyphics (24+)



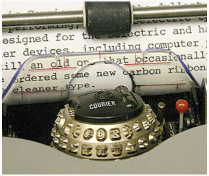
Cuneiform (1000 – 300)



Sanskrit (36)



Chinese (214 – 4000)

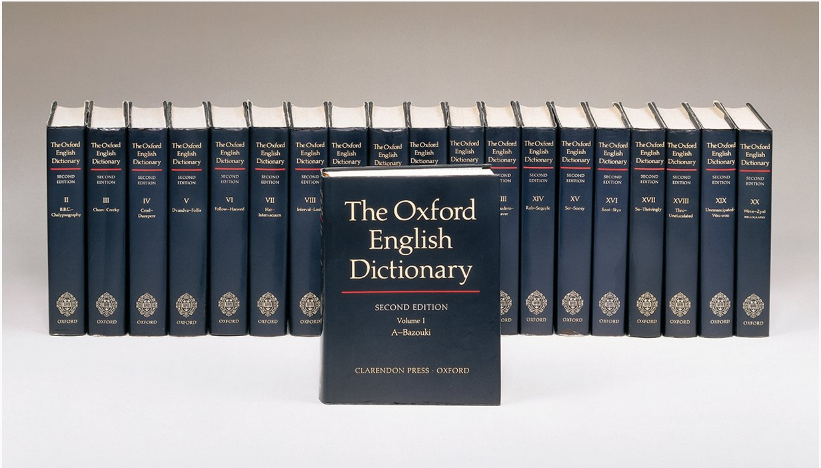


IBM Selectric (88–96)

The Second Question

How do we describe the combinations of a small number of things.

Just List Them?



Gets annoying for large numbers of combinations

Just List Them?

3 AA—A A A A A A A A A A A A

A A A A A Budget Moving 16 WilbyCr . 241-5468
A A A A A Canadian Mini-Warehouse Properties 5399 EglintonW . 620-1577
1001 ArrowRd . 742-0228
24 JeffersonAv . 533-7572
4120 FinchE . 298-3126
A A A A A Critter Control . 201-4711
A A A A A Critter Control 100 Burncrest Unionville . 410-8727
A A A A A Devco Glass . 410-0371
A A A A A Drainworks Ltd Toronto East . 422-0501
A A A A A Eevening Rendezvous . 929-6848
AAAAA Elf Mini Storage 555 TretheweyDr . 247-6294
A A A A A European . 962-2033
AAAAA Expert Movers 16 WilbyCr . 242-7478
A A A A A Jewel Of The Orient . 929-9975
A A A A A Limousine Connection The . 967-5466
A A A A A Mature Escorts . 925-5433
A A A A A Move Master . 588-4656
A A A A A Neal Professional Moving Systems 2480 LawrenceAvE . 285-6325
A A A A A Prince Claude Moving . 287-6701
AAAAA Silk Stockings . 534-3509
A A A A A Woodbine Moving&Storage Ltd 65 Crockford . 751-4900
A A A A A A Alert Glass&Mirror . 638-1989
A A A A A A All Star Movers 603 Evans . 259-1578
A A A A A A Armstrong Moving& Storage . 233-2477
A A A A A A HSL Moving&Storage 603 Evans . 253-7290
A A A A A A Middup Moving&Storage 60 EsnaParkDr . 494-9451
A A A A A A--1 Moving&Storage 637 Lansdowne . 516-3536
A A A A A A Prestige Movers 703 GladstoneAv . 533-2633
AAAAAA South Western Ontario Wildlife Removal . 690-4066
AAAAAA Speedy Moving 124 Crockford . 285-6084
A-A-A-A-A-A Speedy Moving 1540 VictoriaPark . 751-9532
A A A A A A A Across The World Courier 425 AdelaideW . 504-0008
A A A A A A A Auto Glass 855 Alness . 663-8676
AAAAAAA California Dreams Escort Service . 323-3899
AAAAAAA California Dreams Massage Service . 323-3899
AAAAAAA National Auto Glass 562 Kipling . 503-3833
A A A A A A A Night&Day . 929-9975
AAAAAAA Strip 'N Tell . 964-7877
A A A A A A A Unforgettable Escorts . 398-5337
A A A A A A A A Automated Door Systems 22 Jutland . 255-7127
AAAAAAAA California Beach Club Escort Service . 323-9822

A A A A A A A A CBS Moving 130 Lansdowne . 533-7139
A A A A A A A A Dream Girls . 255-5032
A A A A A A A A A A Big Apple Escort Service . 465-2767
AAAAAAAAA Accident And Accompanying Injuries&Criminal Practice 1018 FinchW . 663-2211
A A A A A A A A A Accident Accompanying Injuries&Criminal Practice 1018 FinchW . 663-2211
AAAAAAAAA China Blue Escort Service . 323-9522
A A A A A A A A A A AAABCO Door Co 1860 BonhillRd Mississauga . 748-3667
A A A A A A A A A A Action Law 5233 DundasStW . 253-0888
A A A A A A A A A A Alert Auto Glass . 398-4585
Or . 599-3410
AAAAAAAAAA AMJ Campbell Van Lines Inc 1190 MeyersideDr . 213-5660
A A A A A A A A A A Auto Glass Hotline . 283-0042
A A A A A A A A A A Collins&Greig Cartage Ltd 33 Coronet . 239-2991
A A A A A A A A A A Competition Auto Glass . 223-1292
A A A A A A A A A A Competition Auto Glass . 283-0042
A A A A A A A A A A Competition Auto Glass . 410-7693
A A A A A A A A A A International Escorts . 929-6848
A A A A A A A A A A Jewel Dating&Escort Service . 461-0629
A A A A A A A A A A Marketing Services . 413-0444
A A A A A A A A A A Nothing But Class . 595-1884
A A A A A A A A A A On The Wild Side Sensational Female Escort Service . 255-1320
A A A A A A A A A A The Good Life Clubs 21 McCaul . 979-1422
1126 FinchW . 667-0574
1191 Kennedy . 297-7279
302 TheEastMall . 239-2783
If Busy Call . 667-0470
A A A A A A A A A A A Affordable And Aggressive Defence 4950 YongeSt . 221-7108
A A A A A A A A A A A Campbell Moving Systems . 265-4433
A A A A A A A A A A A-1 Windshields To Go 159 Dynevor . 787-8039
A A A A A A A A A A A Sunset Escorts . 622-1177
A A A A A A A A A A A A Best Of The Best . 929-3039
A A A A A A A A A A A A Bill&Son Towing 286 RoyalYork . 255-8518

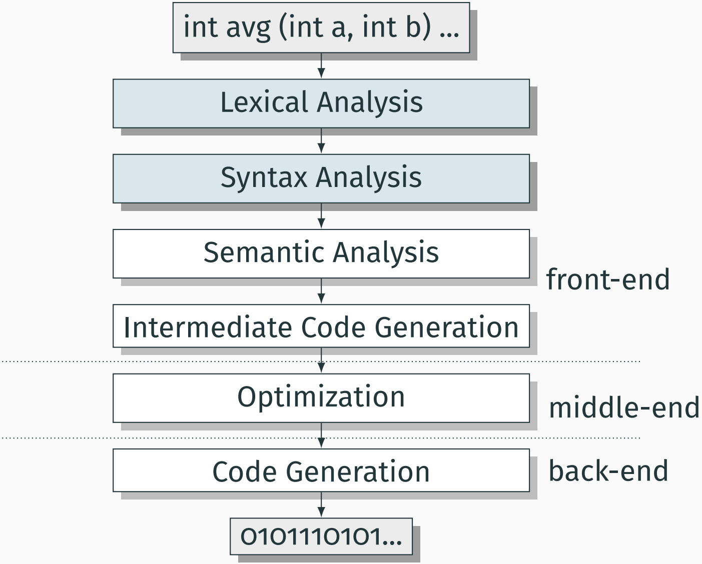
A A A A A A A A A A A A 4

A A A A A A A A A A A A Class Above Limousine 173 DanforthAv . 465-5643
AAAAAAAAAAAA Cross Movers 1232-B Woodbine . 423-0239
A A A A A A A A A A A A Miss Victoria . 967-7176
A A A A A A A A A A A A A Payless Escorts . 485-5333
A A A A A A A A A A A A A A A A A A A A A A 700 LawrenceAvW . 256-1600
A A A A A A A A A A A A A A A A A A A A A A A A A A A A . 699-6700
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A Mannie Zeller 255 DuncanMillRd . 441-9500
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A Cohen& Associates 1 StClairE . 323-0907
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A Aabaco Transmissions 285 OldKingston . 287-0000
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A Abba Movers&Storage 17 Canso . 242-6662
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A Abba Movers& Storage . 366-0237
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A Access 14-A Hazelton . 964-0138
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A Adrian The Mover 64 StClairW . 944-2018
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A Abba Auto Collision& Glass . 777-9595
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A Armor Lock And Safe 6083 Yonge . 225-5589
A A A A A A A A A A A A A A A A A A A A A A A A A A A Basement Systems Canada 38 Garnforth . 285-6002
A A A A A A A A A A A A A A A A A A A A A A A A A A A Law
250 SheppardAvE . 222-6789
If Busy Call . 222-7589
33 Isabella . 222-6311
55 TownCentre Court . 222-5867
A A A A A A A A A A A A A A A A A A A A A A A A A Ad 3420 FinchE . 499-2144
A A A A A A A A A A A A A A A A A A A A A A Law 305 Milner . 299-6688
A A A A A A A A A A A A A A A A A A A A Action Law 5233 DundasStW . 253-0888
A A A A A A A A A A A A A A A A A A A A Allan&Associates 401 Bay . 363-5431
A A A A A A A A A A A A A A A A A A A& W Auto Glass 821 Kipling . 233-4773
A A A A A A A A A A A A A A A A A A A Eagle Alarms 557 DixonRd . 247-0000
A A A A A A A A A A A A A A A A A A B Towing 18 Canso . 245-7676

A A A A A A A A A A A A A A A A B Towing 18 Canso . 245-7676
A A A A A A A A A A A A A A A A Robertson Moving&Storage 236 NorthQueen . 620-1212
AAAAAAAAAAAAAAA AAAAAA Rezz . 652-5252
A A A A A A A A A A A A A A A Access Law . 784-2020
A A A A A A A A A A A A A A A Accident Accompanying Injuries&Criminal Practice 1000 FinchW . 663-2211
A A A A A A A A A A A A A A A Accident Claims 2 StClairW . 944-2313
A A A A A A A A A A A A A A A Account Ability 2 SheppardAvE . 224-0750
A A A A A A A A A A A A A A A Advant-Edge Door Systems . 222-8322
A A A A A A A A A A A A A A A An Executive's Choice . 929-9390
A A A A A A A A A A A A A A A Automatic Garage Doors 64 Clarkson . 785-7820
Etobicoke . 252-5686
A A A A A A A A A A A A A A A Cross Alarms 280 Consumers . 494-9777
A A A A A A A A A A A A A A A Elegant Mature Escorts . 923-3333
A A A A A A A A A A A A A A A Professional Express System 425 AdelaideW . 504-9111
A A A A A A A A A A A A A A A Sweet Escorts&You . 259-3940
A AAAA AAAA AAAA AAAA Anthony De Marco 1205 StClairW . 651-2299
A AAAA AAAA AAAA AAAA Domenic Tagliola 1205 StClairW . 651-2299
AAAAAAAAAAAAAAAAA A Always Available . 465-9191
AAAAAAAAAAAAAAAAA A Touch Of Class Escort Service . 461-8110
AAAAAAAAAAAAAAAAAAAAAAAAAA Apple Auto Glass
No Charge-Dial . 1 800 506-5665
AAAAAAAAAAAAAAAAAAAAAAAAAA Cardinal Custom Building 2 BloorW . 966-4728
A A A A A A A L U Student Movers . 693-2403
A A A AAABCO Door Co 1860 BonhillRd Mississauga . Toronto 748-3667
A A A A A A B S Movers 643 LansdowneAv . 588-1499
A AA AABBCCDEF Locksmith 80 StClairE . 922-2255
A A A A A B C Movers Inc 6 Columbus . 535-3413
A A A A A G B Best Movers . 503-9321
AAAAA M O I Moving Systems 955 Middlefield . 299-4239
A A A A&B Moving 900 CaledoniaRd . 787-4964
A A AABBBEE Locksmiths . 287-6001
A A A ABC Glass Supply 11 Concord . 531-1548
AAAABCO Door&Window Co 1860 BonhillRd Mississauga . Toronto 748-3667

Can be really redundant

Scanning and Parsing



Lexical Analysis

Lexical Analysis (Scanning)

Translate a stream of characters to a stream of tokens

f o o ␣ = ␣ a + ␣ bar (0 , ␣ 42 , ␣ q) ;

ID EQUALS ID PLUS ID LPAREN NUM COMMA ID LPAREN SEMI

Token	Lexemes	Pattern
EQUALS	$=$	an equals sign
PLUS	$+$	a plus sign
ID	a foo bar	letter followed by letters or digits
NUM	0 42	one or more digits

Lexical Analysis

Goal: simplify the job of the parser and reject some wrong programs, e.g.,

```
%#$^#!#%#$
```

is not a C program†

Scanners are usually much faster than parsers.

Discard as many irrelevant details as possible (e.g., whitespace, comments).

Parser does not care that the identifer is "supercalifragilisticexpialidocious."

Parser rules are only concerned with tokens.

† It is what you type when your head hits the keyboard

Describing Tokens

Alphabet: A finite set of symbols

Examples: $\{$ 0, 1 $\}$, $\{$ A, B, C, $\ldots$, Z $\}$, ASCII, Unicode

String: A finite sequence of symbols from an alphabet

Examples: ϵ (the empty string), Ronghui, $\alpha\beta\gamma$

Language: A set of strings over an alphabet

Examples: $\emptyset$ (the empty language), $\{$ 1, 11, 111, 1111 $\}$, all English words, strings that start with a letter followed by any sequence of letters and digits

Operations on Languages

Let $L = \{ \epsilon$, wo $\}$, $M = \{$ man, men $\}$

Concatenation: Strings from one followed by the other

$LM = \{$ man, men, woman, women $\}$

Union: All strings from each language

$L \cup M = \{\epsilon,$ wo, man, men $\}$

Kleene Closure: Zero or more concatenations

$M^* = \{\epsilon\} \cup M \cup MM \cup MMM \cdots =$
$\{\epsilon,$ man, men, manman, manmen, menman, menmen, manmanman, manmanmen, manmenman, $\ldots\}$

Regular Expressions over an Alphabet Σ

A standard way to express languages for tokens.

1. ϵ is a regular expression that denotes $\{\epsilon\}$
2. If $a \in \Sigma$, a is an RE that denotes $\{a\}$
3. If r and s denote languages $L(r)$ and $L(s)$,

$(r) \mid (s)$	denotes	$L(r) \cup L(s)$
$(r)(s)$		$\{tu : t \in L(r), u \in L(s)\}$
$(r)^*$		$\cup_{i=0}^{\infty} L(r)^i$
	where	$L(r)^0 = \{\epsilon\}$
	and	$L(r)^i = L(r)L(r)^{i-1}$

Regular Expression Examples

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

Regexp.	**Language**
$a \mid b$	$\{a, b\}$
$(a \mid b)(a \mid b)$	$\{aa, ab, ba, bb\}$
a^*	$\{\epsilon, a, aa, aaa, aaaa, \ldots\}$
$(a \mid b)^*$	$\{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, \ldots\}$
$a \mid a^*b$	$\{a, b, ab, aab, aaab, aaaab, \ldots\}$

Specifying Tokens with REs

ID: letter followed by letters or digits

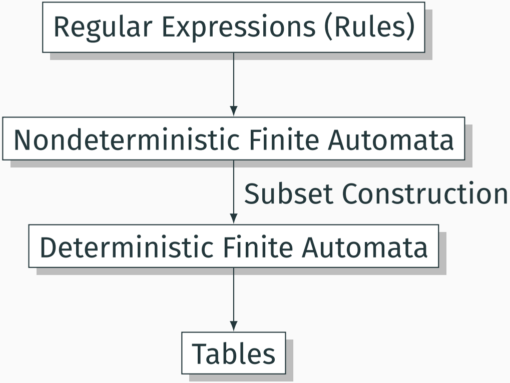
Typical choice: $\Sigma =$ ASCII characters, i.e., $\{␣, !, ", \#, \$, \ldots, 0, 1, \ldots, 9, \ldots, A, \ldots, Z, \ldots, \tilde{}\}$

letters: $A \mid B \mid \cdots \mid Z \mid a \mid \cdots \mid z$

digits: $0 \mid 1 \mid \cdots \mid 9$

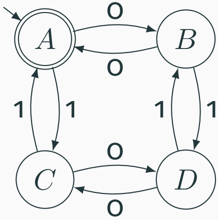
identifier: **letter** $($ **letter** $\mid$ **digit** $)^*$

Implementing Scanners Automatically



Nondeterministic Finite Automata

"All strings containing an even number of 0's and 1's"



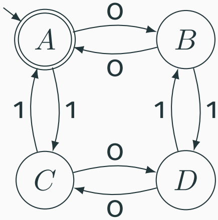
1. Set of states
   $S : \{ A \; B \; C \; D \}$
2. Set of input symbols $\Sigma : \{0, 1\}$
3. Transition function
   $\sigma : S \times \Sigma_\epsilon \to 2^S$

state	ϵ	0	1
A	$\emptyset$	$\{B\}$	$\{C\}$
B	$\emptyset$	$\{A\}$	$\{D\}$
C	$\emptyset$	$\{D\}$	$\{A\}$
D	$\emptyset$	$\{C\}$	$\{B\}$

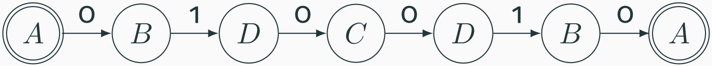
4. Start state $s_0 : A$
5. Set of accepting states
   $F : \{ A \}$

The Language induced by an NFA

An NFA accepts an input string x iff there is a path from the start state to an accepting state that "spells out" x.



Show that the string "010010" is accepted.

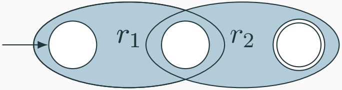


Translating REs into NFAs (Thompson's algorithm)

a

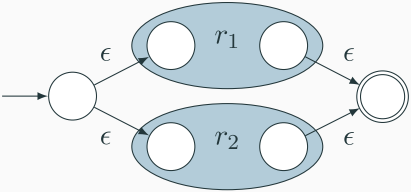
Symbol

$r_1 r_2$



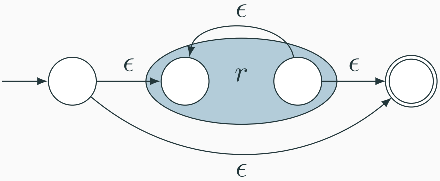
Sequence

$r_1 \mid r_2$



Choice

$(r)^*$

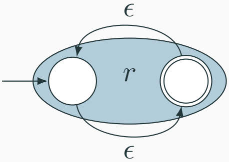


Kleene Closure

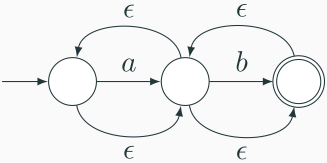
Why So Many Extra States and Transitions?

Invariant: Single start state; single end state; at most two outgoing arcs from any state: helpful for simulation.

What if we used this simpler rule for Kleene Closure?



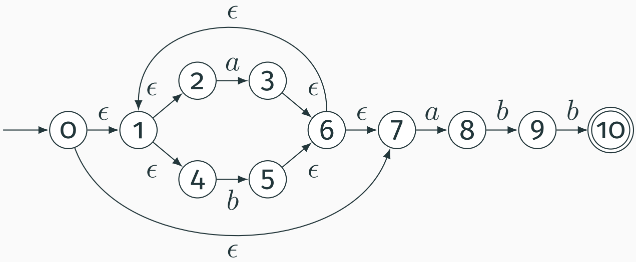
Now consider a^*b^* with this rule:



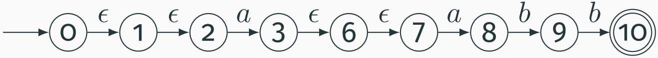
Is this right?

Translating REs into NFAs

Example: Translate $(a \mid b)^* abb$ into an NFA. Answer:



Show that the string "$aabb$" is accepted. Answer:



Simulating NFAs

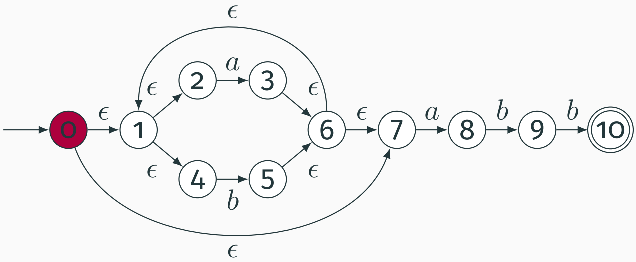
Problem: you must follow the "right" arcs to show that a string is accepted. How do you know which arc is right?

Solution: follow them all and sort it out later.

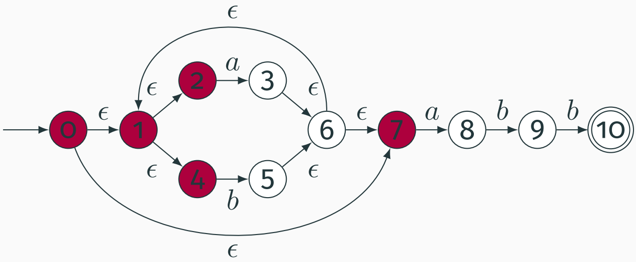
"Two-stack" NFA simulation algorithm:

1. Initial states: the ϵ-closure of the start state
2. For each character c,
   - New states: follow all transitions labeled c
   - Form the ϵ-closure of the current states
3. Accept if any final state is accepting

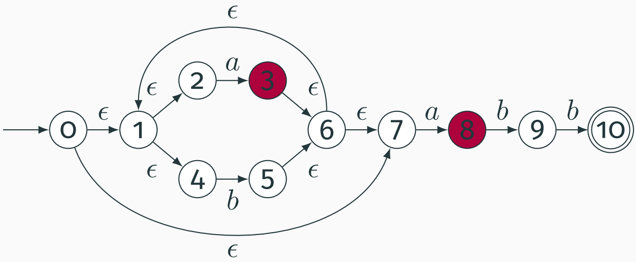
Simulating an NFA: $\cdot aabb$, Start



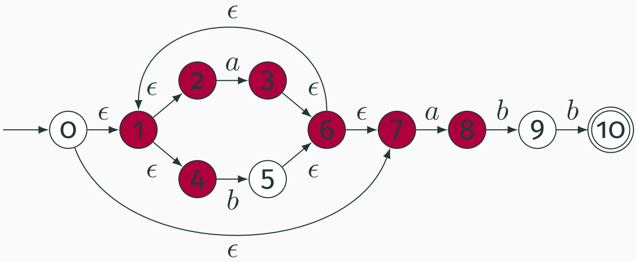
Simulating an NFA: $\cdot aabb$, ϵ-closure



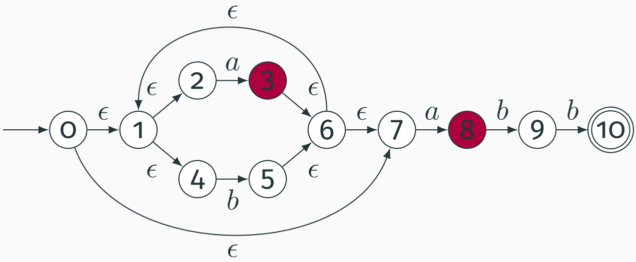
Simulating an NFA: $a{\cdot}abb$



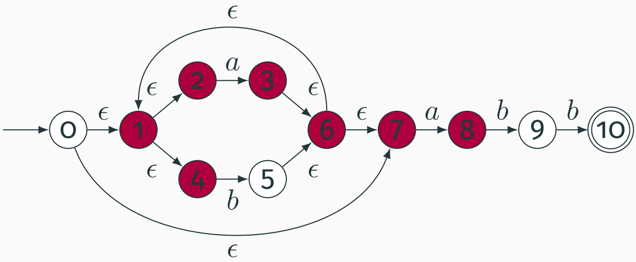
Simulating an NFA: $a \cdot abb$, ϵ-closure



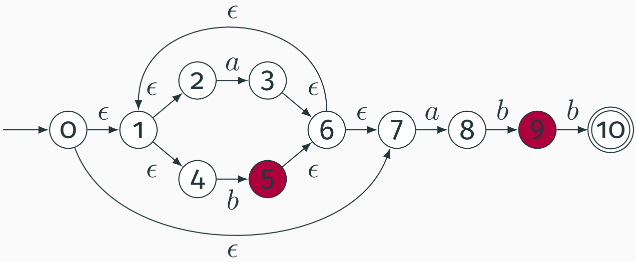
Simulating an NFA: $aa{\cdot}bb$



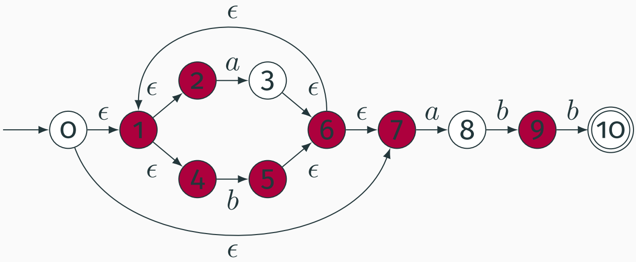
Simulating an NFA: $aa{\cdot}bb$, ϵ-closure



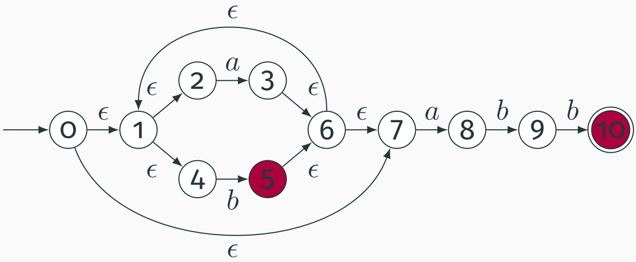
Simulating an NFA: $aab{\cdot}b$



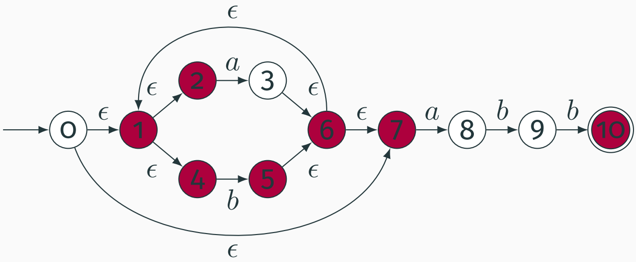
Simulating an NFA: $aab{\cdot}b$, ϵ-closure



Simulating an NFA: $aabb\cdot$



Simulating an NFA: $aabb\cdot$, Done



Deterministic Finite Automata

Restricted form of NFAs:

- No state has a transition on ϵ
- For each state s and symbol a, there is at most one edge labeled a leaving s.

Differs subtly from the definition used in COMS W3261 (Sipser, *Introduction to the Theory of Computation*)

Very easy to check acceptance: simulate by maintaining current state. Accept if you end up on an accepting state. Reject if you end on a non-accepting state or if there is no transition from the current state for the next symbol.

Deterministic Finite Automata

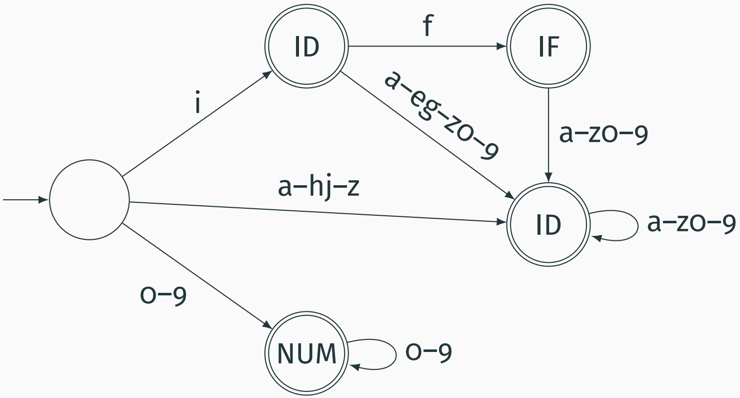
```
{
   type token = ELSE | ELSEIF
}

rule token =
  parse "else"   { ELSE }
      | "elseif" { ELSEIF }
```

Deterministic Finite Automata

```
{ type token = IF | ID of string | NUM of string }

rule token =
  parse "if"                                     { IF }
      | ['a'-'z'] ['a'-'z' '0'-'9']* as lit { ID(lit) }
      | ['0'-'9']+                        as num { NUM(num) }
```



Building a DFA from an NFA

Subset construction algorithm

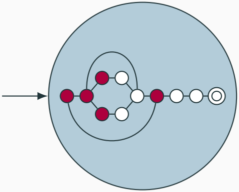
Simulate the NFA for all possible inputs and track the states that appear.

Each unique state during simulation becomes a state in the DFA.

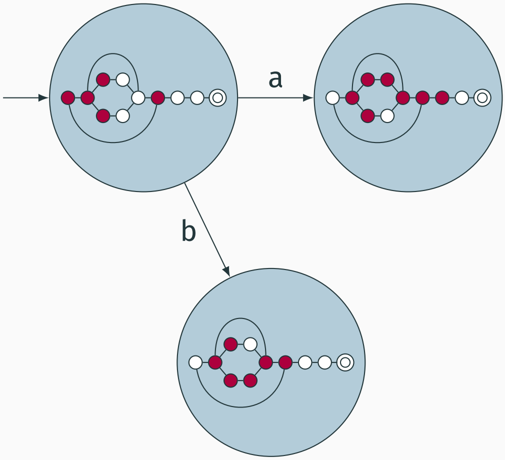
The Subset Construction Algorithm

1. Create the start state of the DFA by taking the ε-closure of the start state of the NFA.
2. Perform the following for the new DFA state: For each possible input symbol:
   - Apply move to the newly-created state and the input symbol; this will return a set of states.
   - Apply the ε-closure to this set of states, possibly resulting in a new set. This set of NFA states will be a single state in the DFA.
3. Each time we generate a new DFA state, we must apply step 2 to it. The process is complete when applying step 2 does not yield any new states.
4. The finish states of the DFA are those which contain any of the finish states of the NFA.

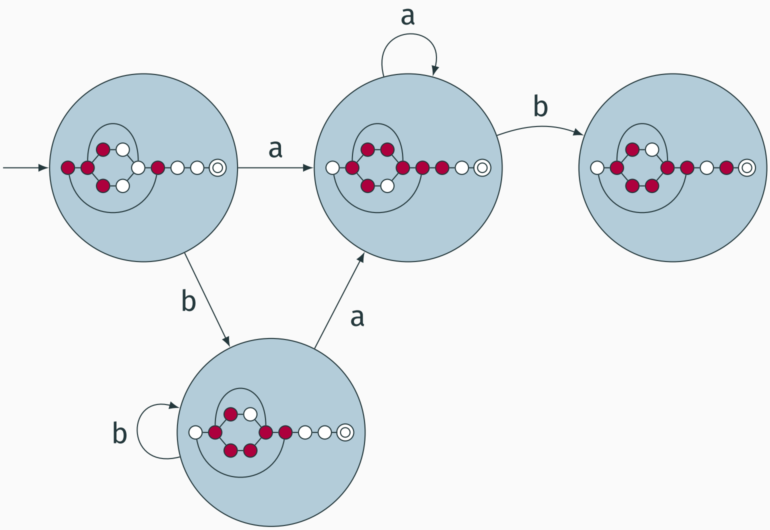
Subset construction for $(a \mid b)^*abb$



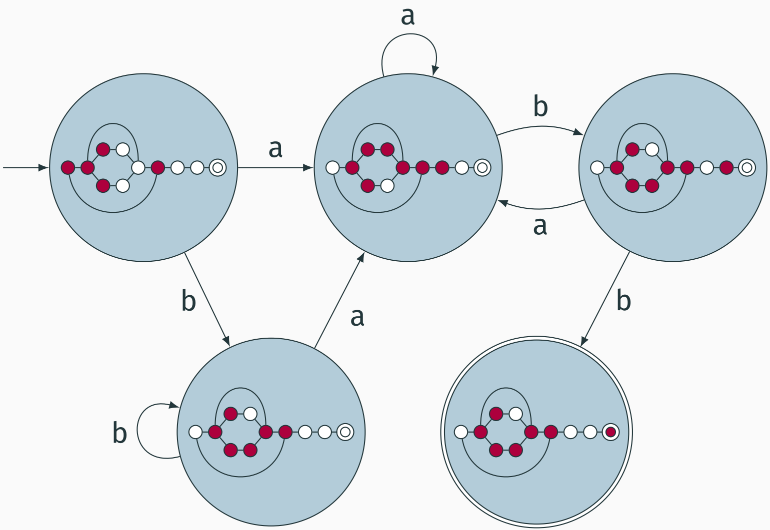
Subset construction for $(a \mid b)^*abb$



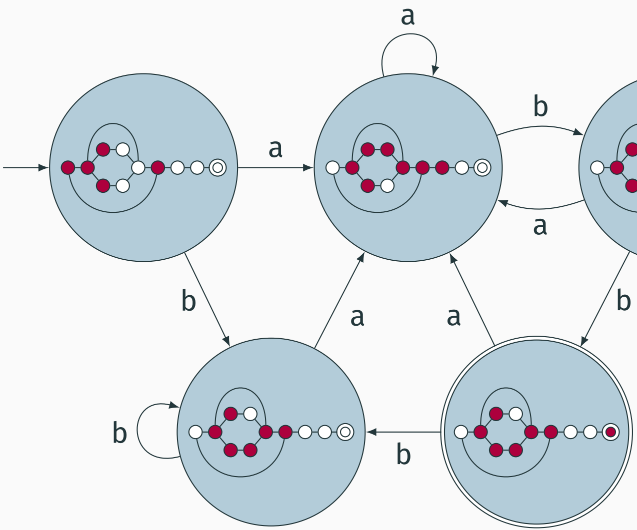
Subset construction for $(a \mid b)^*abb$



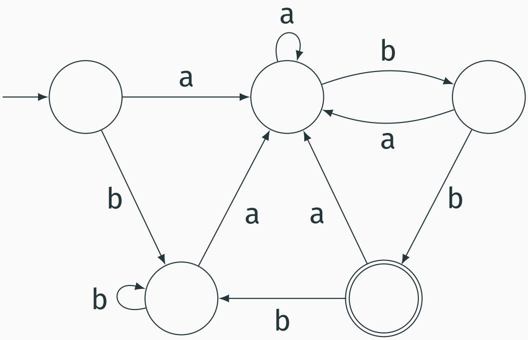
Subset construction for $(a \mid b)^*abb$



Subset construction for $(a \mid b)^*abb$

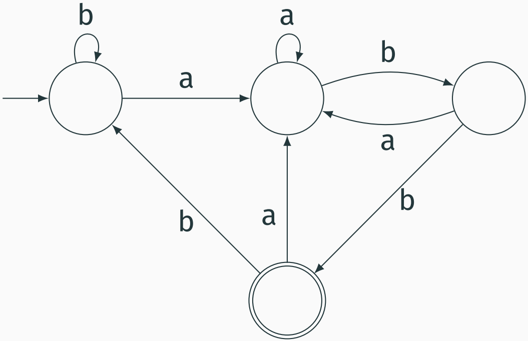


Result of subset construction for $(a \mid b)^*abb$



Is this minimal?

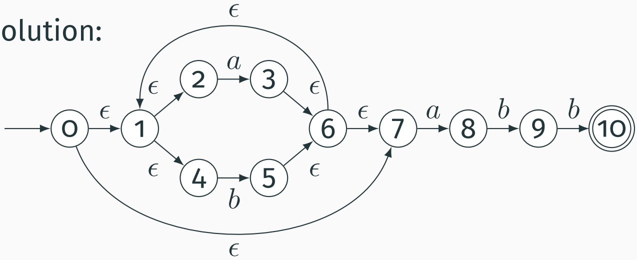
Minimized result for $(a \mid b)^*abb$



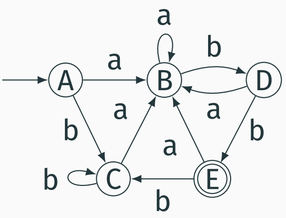
Transition Table Used In the Dragon Book

Problem: Translate $(a \mid b)^*abb$ into an NFA and perform subset construction to produce a DFA.

Solution:



NFA State	DFA State	a	b
{0,1,2,4,7}	A	B	C
{1,2,3,4,6,7,8}	B	B	D
{1,2,4,5,6,7}	C	B	C
{1,2,4,5,6,7,9}	D	B	E
{1,2,4,5,6,7,10}	E	B	C



Subset Construction

An DFA can be exponentially larger than the corresponding NFA.

n states versus 2^n

Tools often try to strike a balance between the two representations.

Lexical Analysis with Ocamllex

Constructing Scanners with Ocamllex

scanner.mll → ocamllex (subset construction) → scanner.ml

An example:

scanner.mll

```
{ open Parser }

rule token =
  parse [' ' '\t' '\r' '\n'] { token lexbuf }
      | '+'                  { PLUS }
      | '-'                  { MINUS }
      | '*'                  { TIMES }
      | '/'                  { DIVIDE }
      | ['0'-'9']+ as lit    { LITERAL(int_of_string lit) }
      | eof                  { EOF }
```

Ocamllex Specifications

```
{
  (* Header: verbatim OCaml code; mandatory *)
}

(* Definitions: optional *)
let ident = regexp
let ...

(* Rules: mandatory *)
rule entrypoint1 [arg1 ... argn] =
  parse pattern1 { action (* OCaml code *) }
      | ...
      | patternn { action }
and entrypoint2 [arg1 ... argn]} =
  ...
and ...

{
  (* Trailer: verbatim OCaml code; optional *)
}
```

Patterns (In Order of Decreasing Precedence)

Pattern	Meaning
'c'	A single character
_	Any character (underline)
eof	The end-of-file
"foo"	A literal string
['1' '5' 'a'-'z']	"1," "5," or any lowercase letter
[^ '0'-'9']	Any character except a digit
(*pattern*)	Grouping
identifier	A pattern defined in the let section
pattern *	Zero or more *patterns*
pattern +	One or more *patterns*
pattern ?	Zero or one *patterns*
$pattern_1$ $pattern_2$	$pattern_1$ followed by $pattern_2$
$pattern_1$ | $pattern_2$	Either $pattern_1$ or $pattern_2$
pattern as *id*	Bind the matched pattern to variable *id*

An Example

```
{ type token = PLUS | IF | ID of string | NUM of int }
let letter = ['a'-'z' 'A'-'Z']
let digit = ['0'-'9']

rule token =
 parse [' ' '\n' '\t'] { token lexbuf } (* Ignore whitespace *)

     | '+' { PLUS }                     (* A symbol *)

     | "if" { IF }                      (* A keyword *)
                                        (* Identifiers *)
     | letter (letter | digit | '_')* as id { ID(id) }
                                        (* Numeric literals *)
     | digit+ as lit { NUM(int_of_string lit) }

     | "/*" { comment lexbuf }          (* C-style comments *)

and comment =
  parse "*/" { token lexbuf } (* Return to normal scanning *)
      | _ { comment lexbuf }  (* Ignore other characters *)
```

Nested Comments

```
{ type token = PLUS | ID of string | NUM of int }

let letter = ['a'-'z' 'A'-'Z']
let digit = ['0'-'9']

rule token =
 parse [' ' '\n' '\t'] { token lexbuf } (* Ignore whitespace *)

     | '+' { PLUS }                       (* A symbol *)

     | letter (letter | digit | '_')* as id { ID(id) }
     | digit+ as lit { NUM(int_of_string lit) }

     | "/*" { comment 0 lexbuf }           (* C-style comments *)

and comment level =
  parse "*/" { if level == 0 then token lexbuf
        else comments (level - 1) lexbuf }
      | "/*" { comment (level + 1) lexbuf }
      | _ { comment level lexbuf }  (* Ignore other characters *)
```

Free-Format Languages

Typical style arising from scanner/parser division

Program text is a series of tokens possibly separated by whitespace and comments, which are both ignored.

- keywords (if while)
- punctuation (, (+)
- identifiers (foo bar)
- numbers (10 -3.14159e+32)
- strings ("A String")

Free-Format Languages

Java C C++ C# Algol Pascal

Some deviate a little (e.g., C and C++ have a separate preprocessor)

But not all languages are free-format.

Python

The Python scripting language groups with indentation

```
i = 0
while i < 10:
    i = i + 1
    print i     # Prints 1, 2, ..., 10

i = 0
while i < 10:
    i = i + 1
print i         # Just prints 10
```

This is succinct, but can be error-prone.

How do you wrap a conditional around instructions?

Syntax and Language Design

Does syntax matter? Yes and no

More important is a language's *semantics*—its meaning.

The syntax is aesthetic, but can be a religious issue.

But aesthetics matter to people, and can be critical.

Verbosity does matter: smaller is usually better.

Too small can be problematic: APL is a succinct language with its own character set.

There are no APL programs, only puzzles.

Syntax and Language Design

Some syntax is error-prone. Classic FORTRAN example:

```
DO 5 I = 1,25   ! Loop header (for i = 1 to 25)
DO 5 I = 1.25   ! Assignment to variable DO5I
```

Trying too hard to reuse existing syntax in C++:

```
vector< vector<int> > foo;
vector<vector<int>> foo; // Syntax error
```

C distinguishes $>$ and $>>$ as different operators.

Bjarne Stroustrup tells me they have finally fixed this.