LEX7: DFA Complexity

Lexical Analysis

CMPT 379: Compilers

Instructor: Anoop Sarkar

anoopsarkar.github.io/compilers-class

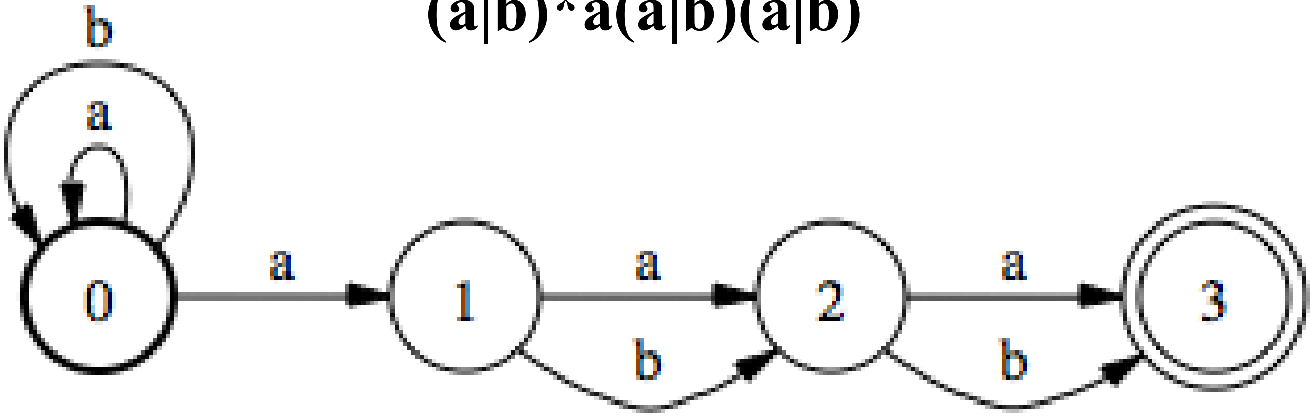
NFA to DFA

- converting NFA to DFA
- Complexity:
  - For FSAs, we measure complexity in terms of initial cost (creating the automaton) and per string cost
  - Let *r* be the length of the regexp and *n* be the length of the input string
  - NFA, Initial cost: O(r); Per string: O(rn)
  - DFA, Initial cost: O(r^2s); Per string: O(n)
  - DFA, common case, $s = r$, but worst case $s = 2^r$

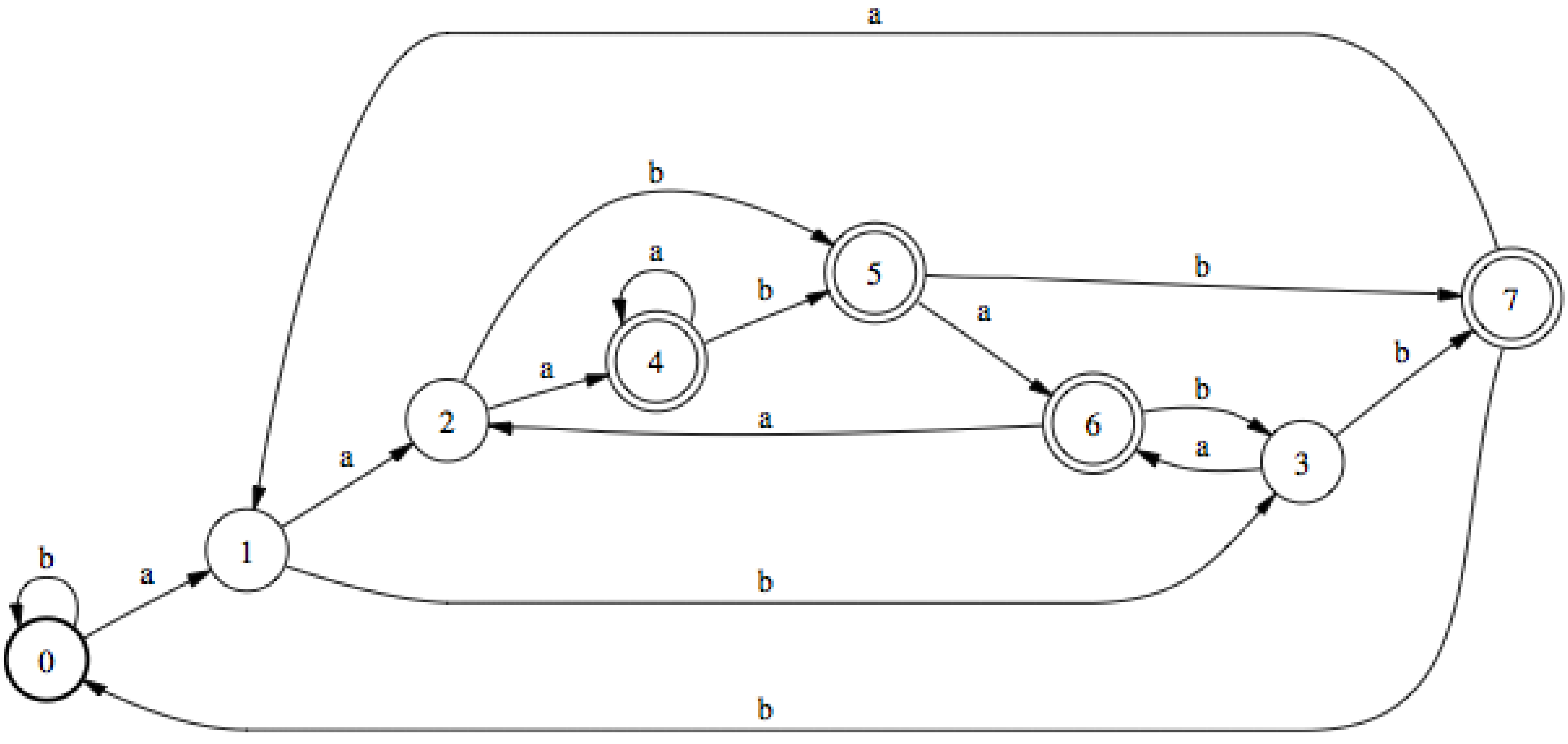
NFA to DFA

- A regexp of size *r* can become a *2^r* state DFA, an exponential increase in complexity
  - Try the subset construction on NFA built for the regexp **A*aA^{n-1}** where **A** is the regexp **(a|b)**

(a|b)*a(a|b)(a|b)

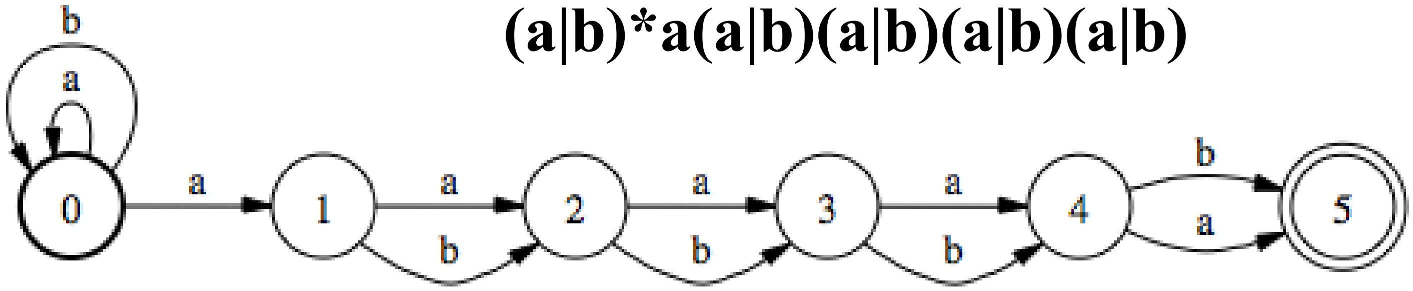


NFA to DFA



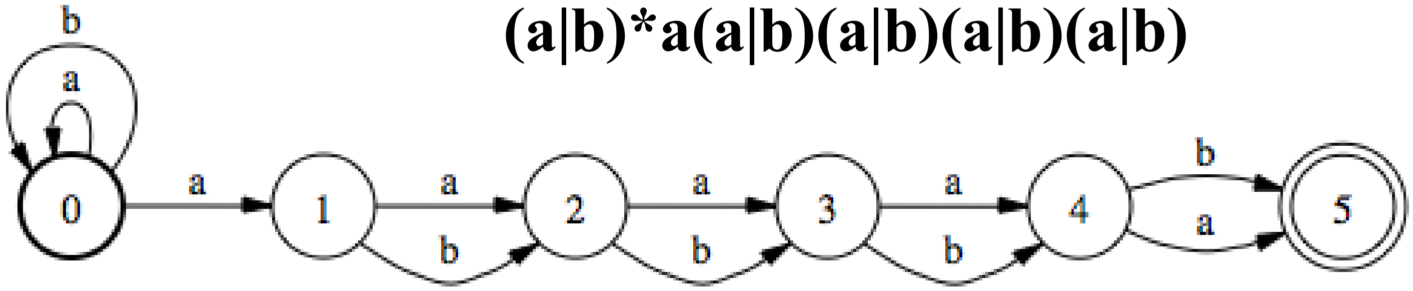
NFA to DFA

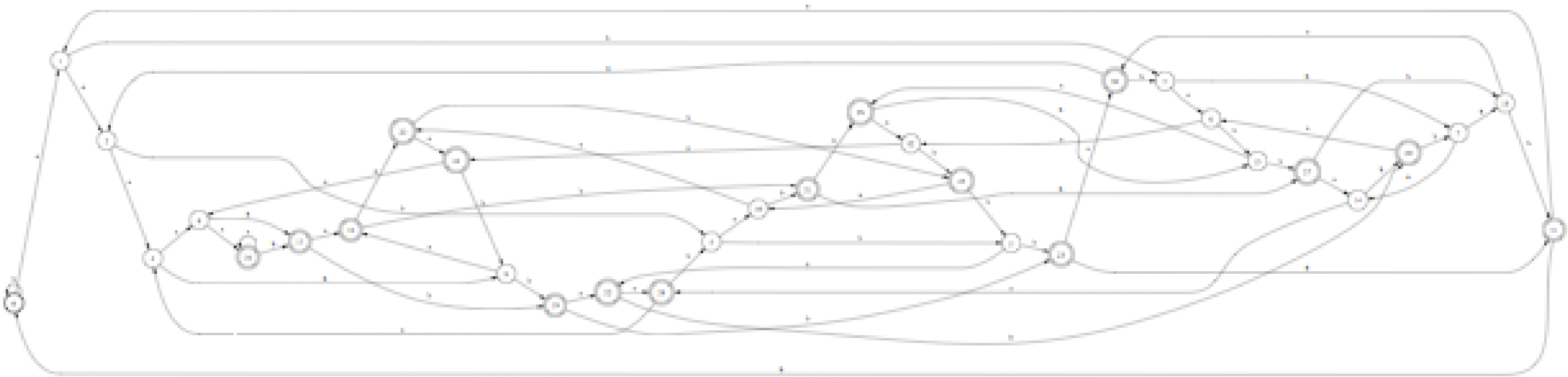
(a|b)*a(a|b)(a|b)(a|b)(a|b)



NFA to DFA

(a|b)*a(a|b)(a|b)(a|b)(a|b)





$2^5 = 32$ states

NFA vs. DFA in the wild

Engine Type	**Programs**
DFA	*awk* (most versions), *egrep* (most versions), *flex*, *lex*, MySQL, Procmail
Traditional NFA	GNU *Emacs*, Java, *grep* (most versions), *less*, *more*, .NET languages, PCRE library, Perl, PHP (pcre routines), Python, Ruby, *sed* (most versions), vi
POSIX NFA	*mawk*, MKS utilities, GNU *Emacs* (when requested)
Hybrid NFA/DFA	GNU *awk*, GNU *grep*/*egrep*, Tcl

Extensions to Regular Expressions

- Most modern regexp implementations provide extensions:

Extensions to Regular Expressions

- Most modern regexp implementations provide extensions:
  - matching groups; \1 refers to the string matched by the first grouping (), \2 to the second match, etc.,
    - e.g. ([a-z]+)\1 which matches abab where \1=ab

Extensions to Regular Expressions

- Most modern regexp implementations provide extensions:
  - matching groups; \1 refers to the string matched by the first grouping (), \2 to the second match, etc.,
    - e.g. ([a-z]+)\1 which matches abab where \1=ab
  - match and replace operations,
    - e.g. s/([a-z]+)/\1\1/g which changes ab into abab where \1=ab

Extensions to Regular Expressions

- Most modern regexp implementations provide extensions:
  - matching groups; \1 refers to the string matched by the first grouping (), \2 to the second match, etc.,
    - e.g. ([a-z]+)\1 which matches abab where \1=ab
  - match and replace operations,
    - e.g. s/([a-z]+)/\1\1/g which changes ab into abab where \1=ab
- These extensions are no longer "regular". In fact, extended regexp matching is NP-hard

Extensions to Regular Expressions

- Most modern regexp implementations provide extensions:
  - matching groups; \1 refers to the string matched by the first grouping (), \2 to the second match, etc.,
    - e.g. ([a-z]+)\1 which matches abab where \1=ab
  - match and replace operations,
    - e.g. s/([a-z]+)/\1\1/g which changes ab into abab where \1=ab
- These extensions are no longer "regular". In fact, extended regexp matching is NP-hard
  - Extended regular expressions (including POSIX and Perl) are called REGEX to distinguish from regexp (which *are* regular)

Extensions to Regular Expressions

- Most modern regexp implementations provide extensions:
  - matching groups; \1 refers to the string matched by the first grouping (), \2 to the second match, etc.,
    - e.g. ([a-z]+)\1 which matches abab where \1=ab
  - match and replace operations,
    - e.g. s/([a-z]+)/\1\1/g which changes ab into abab where \1=ab
- These extensions are no longer "regular". In fact, extended regexp matching is NP-hard
  - Extended regular expressions (including POSIX and Perl) are called REGEX to distinguish from regexp (which *are* regular)
- In order to capture these difficult cases, the algorithms used even for simple regexp matching run in time exponential in the length of the input

NFA to DFA

- A regexp of size *r* can become a 2^r state DFA, an exponential increase in complexity
  - Try the subset construction on NFA built for the regexp **A*aA**$^{\mathbf{n-1}}$ where **A** is the regexp **(a|b)**
- Note that the NFA for regexp of size *r* will have O(*r*) states
- Minimization can reduce the number of states
- But minimization requires determinization