New Algorithms For Regular Expression Matching

ICALP 2006

Philip Bille
IT University of Copenhagen

Regular Expressions

- The *regular expressions* (regexs) are defined recursively:
- A character $\alpha \in \Sigma$ is a regex.
- If S and T are regexs then so is
  - the *concatenation* ST,
  - the *union* $S|T$, and
  - the *kleene star* S^*.

Regular Expressions

- The *language* $L(R)$ of a regex R is defined by:

  $$L(\alpha) = \{\alpha\}$$

  $$L(ST) = L(S)L(T)$$

  $$L(S|T) = L(S) \cup L(T)$$

- $L(S^*) = \{\epsilon\} \cup L(S) \cup L(S)^2 \cup \cdots$

Example

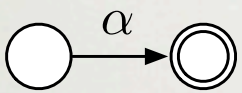
$R = ac|a^*b$

- $L(R) = \{ac, b, ab, aab, aaab, \ldots, \}$

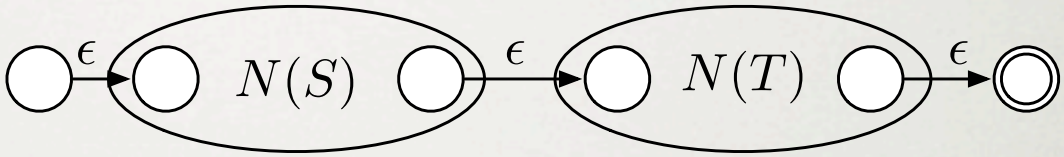
Regex Matching

- For regex R and string Q the *regex matching problem* is to decide if $Q \in L(R)$
- **Classical solution:** Convert R into a *non-deterministic finite automata* (NFA) and *simulate* it. E.g. Thompson [1968].

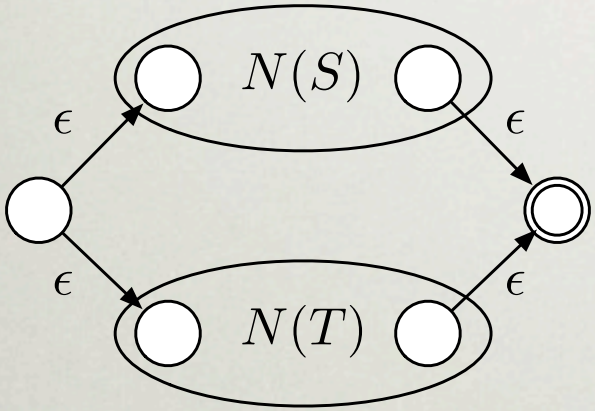
Thompson's Algorithm



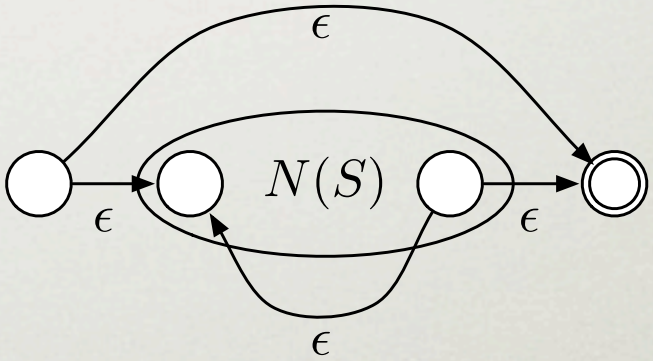
$N(\alpha)$



$N(ST)$

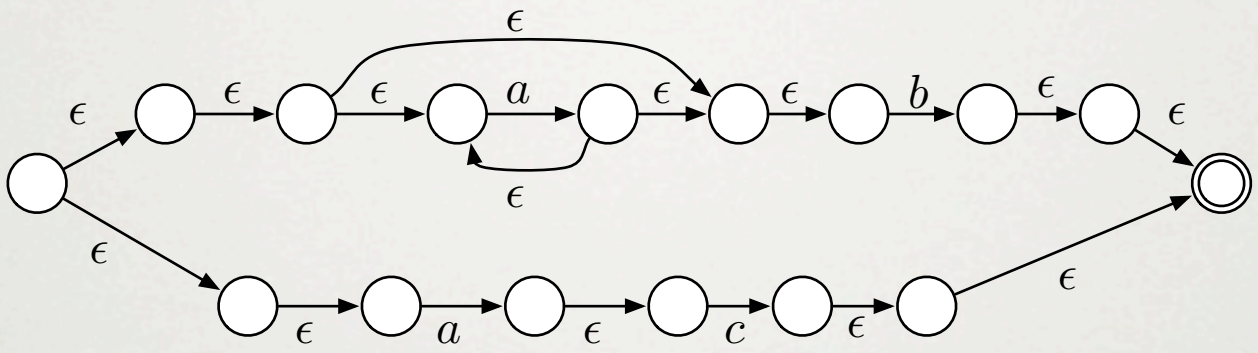


$N(S|T)$



$N(S^*)$

Thompson NFA



- *Thompson-NFA* (TNFA) for $R = ac|a^*b$
- $N(R)$ *accepts* Q iff there is path from θ to ϕ that "spells" out Q.
- $Q \in L(R)$ iff $N(R)$ accepts Q.

Properties of TNFAs

- Linear number of states and transitions.
- Incoming transitions to a state have the same label.
- States with an incoming transition labeled $\alpha \in \Sigma$ (α-states) have exactly 1 predecessor.

Simulating TNFAs

- Let A be TNFA with m states. To test acceptance we use the following operations. Let S be a state-set.
- Move(S, α): Find set of states reachable from S via a single α-transition.
- Close(S): Find set of states reachable from S via a path of ϵ-transitions.
- $O(m)$ time for both operations.

Simulating TNFAs

- Let Q be a string of length n
- The *state-set simulation* of A on Q produces state-sets $S_0, S_1, \ldots, S_n$

  $S_0 := \mathsf{Close}(\{\theta\})$

  $S_i := \mathsf{Close}(\mathsf{Move}(S_{i-1}, Q[i]))$
- $Q \in L(R)$ iff $\phi \in S_n$.
- $O(nm)$ time and $O(m)$ space.

Results

Time	Space	Reference
$O(nm)$	$O(m)$	[Tho68]
$O\left(\frac{nm}{\log n} + (n+m)\log n\right)$	$O(n)$	[Mye92], [BFC05]
$\begin{cases} O(n\frac{m\log w}{w} + m\log w) & \text{if } m > w \\ O(n\log m + m\log m) & \text{if } \sqrt{w} < m \leq w \\ O(\min(n+m^2, n\log m + m\log m)) & \text{if } m \leq \sqrt{w}. \end{cases}$	$O(m)$	This paper

Practical Issues

- Previous algorithms based on “Four-Russian” technique.
  - Large tables.
  - Many expensive cache-misses.
- New solution based on fast bitwise and arithmetic operations.

Simple Algorithm for small TNFAs

- Suppose A is a TNFA with $m = O(\sqrt{w})$ states.
- Order the states such that the (unique) predecessor of α-state i is $i - 1$.
- Represent state-set as bit string

$$S = s_1 s_2 \ldots s_m$$

- where $s_i = 1$ iff i in state-set.

Move Operation

- Precompute for each $\alpha \in \Sigma$ the string

$$D_\alpha = d_1 \ldots d_m$$

- where $d_i = 1$ iff i is an α-state
- $\mathsf{Move}(S, \alpha)$ is computed as:

$$S' := (S >> 1) \,\&\, D_\alpha$$

$i \in S'$ iff $i - 1 \in S$ and i is an α-state.

Close Operation

- Precompute the string

$$E = 0e_{1,1} \dots e_{1,m} 0 e_{2,1} \dots e_{2,m} 0 \cdots 0 e_{m,1} \dots e_{m,m}$$

- where $e_{i,j} = 1$ iff i is ϵ-reachable from j.
- and 3 constants:

$$I = (10^m)^m$$
$$X = 1(0^m)^{m-1}$$
$$C = 1(0^{m-1})^{m-1}$$

Close Operation

- Close(S) is computed as:

$$Y := (S \times X) \,\&\, E$$
$$Z := ((Y \mid I) - (I >> m)) \,\&\, I$$
$$S' := ((Z \times C) << w - m(m+1)) >> w - m$$

Example

- Suppose $m = 3$ and let $S = s_1 s_2 s_3$
- Close$(s_1 s_2 s_3)$ proceeds as follows:

Step 1: $Y := (S \times X)\ \&\ E$

$$
\begin{array}{rl}
S \times X = & s_1 s_2 s_3 \times 1\ 0\ 0\ 0\ \ 1\ 0\ 0\ 0\ \ 1 \\
\hline
 & \begin{array}{cccccccccccc}
 & & & & & & & & 0 & s_1 & s_2 & s_3 \\
 & & & & & & & & 0 & 0 & 0 & \\
 & & & & & & \cdots & & 0 & 0 & & \\
 & & & & & & & & 0 & & & \\
 & & & & 0 & s_1 & s_2 & s_3 & & & & \\
 & & & & 0 & 0 & 0 & & & & & \\
 & & \cdots & & 0 & 0 & & & & & & \\
 & & & & 0 & & & & & & & \\
0 & s_1 & s_2 & s_3 & & & & & & & &
\end{array} \\
\hline
 & 0\ s_1\ s_2\ s_3\ 0\ s_1\ s_2\ s_3\ 0\ s_1\ s_2\ s_3 \\
\& & 0\ e_{1,1}\ e_{1,2}\ e_{1,3}\ 0\ e_{2,1}\ e_{2,2}\ e_{2,3}\ 0\ e_{3,1}\ e_{3,2}\ e_{3,3} \\
\hline
Y = & 0\ y_{1,1}\ y_{1,2}\ y_{1,3}\ 0\ y_{2,1}\ y_{2,2}\ y_{2,3}\ 0\ y_{3,1}\ y_{3,2}\ y_{3,3}
\end{array}
$$

$y_{i,j} = 1$ iff $j \in S$ and i is ϵ-reachable from j.

Step 2: $Z := ((Y \mid I) - (I >> m)) \,\&\, I$

$$
\begin{array}{rl}
Y & 0\, y_{1,1}\, y_{1,2}\, y_{1,3}\, 0\, y_{2,1}\, y_{2,2}\, y_{2,3}\, 0\, y_{3,1}\, y_{3,2}\, y_{3,3} \\
\mid I & 1\ 0\ \ 0\ \ 0\ 1\ 0\ \ 0\ \ 0\ 1\ 0\ \ 0\ \ 0 \\
\hline
 & 1\, y_{1,1}\, y_{1,2}\, y_{1,3}\, 1\, y_{2,1}\, y_{2,2}\, y_{2,3}\, 1\, y_{3,1}\, y_{3,2}\, y_{3,3} \\
-(I >> 3) & 0\ 0\ \ 0\ \ 1\ 0\ 0\ \ 0\ \ 1\ 0\ 0\ \ 0\ \ 1 \\
\hline
 & z_1\ *\ \ *\ \ *\ z_2\ *\ \ *\ \ *\ z_3\ *\ \ *\ \ * \\
\& I & 1\ 0\ \ 0\ \ 0\ 1\ 0\ \ 0\ \ 0\ 1\ 0\ \ 0\ \ 0 \\
\hline
Z = & z_1\ 0\ \ 0\ \ 0\ z_2\ 0\ \ 0\ \ 0\ z_3\ 0\ \ 0\ \ 0
\end{array}
$$

$z_i = 1$ iff i is ϵ-reachable.

Step 3:

$$S' := ((Z \times C) << w - m(m+1)) >> w - m$$

$$
\begin{array}{rl}
Z \times C = & z_1 000 z_2 000 z_3 \times 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \\
\hline
 & \qquad\qquad\quad z_1 \; 0 \; 0 \; 0 \; z_2 \; 0 \; 0 \; 0 \; z_3 \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 0 \\
 & \qquad\qquad\quad z_1 \; 0 \; 0 \; 0 \; z_2 \; 0 \; 0 \; 0 \; z_3 \qquad\quad 0 \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\quad 0 \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\quad 0 \\
 & z_1 \; 0 \; 0 \; 0 \; z_2 \; 0 \; 0 \; 0 \; z_3 \\
\hline
 & z_1 \; 0 \; 0 \; z_1 \; z_2 \; 0 \; z_1 \; z_2 \; z_3 \; 0 \; z_2 \; z_3 \; 0 \; 0 \; z_3
\end{array}
$$

Step 3:

$$S' := ((Z \times C) << w - m(m+1)) >> w - m$$

$$
\begin{array}{rcl}
Z \times C = & z_1000z_2000z_3 & \times\ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1 \\
\hline
 & z_1\ 0\ 0\ 0\ z_2\ 0\ 0\ 0\ z_3 & \\
 & z_1\ 0\ 0\ 0\ z_2\ 0\ 0\ 0\ z_3 & \\
 & z_1\ 0\ 0\ 0\ z_2\ 0\ 0\ 0\ z_3 & \\
 & z_1\ 0\ 0\ 0\ z_2\ 0\ 0\ 0\ z_3 & \\
\hline
 & z_1\ 0\ 0\ z_1\ z_2\ 0\ z_1\ z_2\ z_3\ 0\ z_2\ z_3\ 0\ 0\ z_3 &
\end{array}
$$

Step 3:

$$S' := ((Z \times C) << w - m(m+1)) >> w - m$$

$$Z \times C = z_1000z_2000z_3 \times 1\ 0\ 0\ 1\ 0\ 0\ 1$$

$$\begin{array}{cccccccccccccccccccc}
 & & & & & & & z_1 & 0 & 0 & 0 & z_2 & 0 & 0 & 0 & z_3 \\
 & & & & & & & & & & & & & & 0 & \\
 & & & & & & & & & & & & & 0 & & \\
 & & z_1 & 0 & 0 & 0 & z_2 & 0 & 0 & 0 & z_3 & & & & & \\
 & & & & & & & & & 0 & & & & & & \\
 & & & & & & & & 0 & & & & & & & \\
z_1 & 0 & 0 & 0 & z_2 & 0 & 0 & 0 & z_3 & & & & & & & \\
\hline
z_1 & 0 & 0 & z_1 & z_2 & 0 & z_1 & z_2 & z_3 & 0 & z_2 & z_3 & 0 & 0 & z_3
\end{array}$$

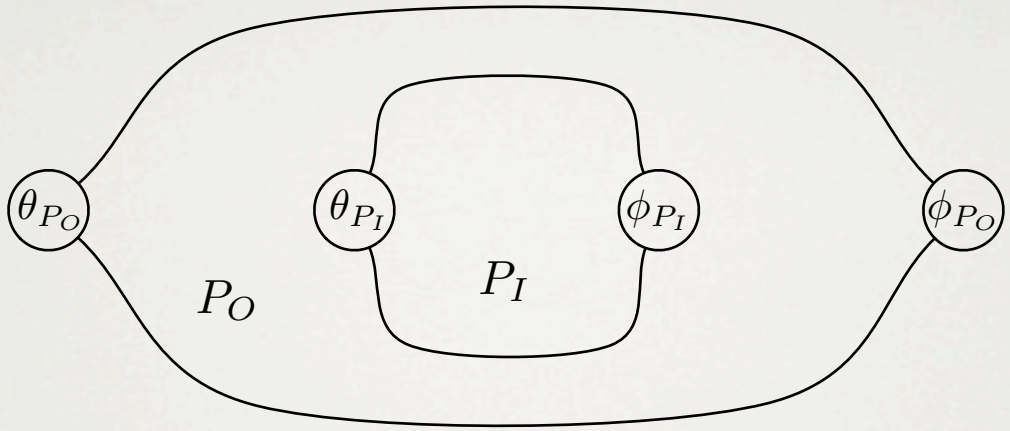
Shifting gives $S' = z_1 z_2 z_3$

Result

- **Lemma** For TNFAs with $m = O(\sqrt{w})$ states we can support Move and Close in $O(1)$ time using $O(m)$ space and $O(m^2)$ preprocessing.
- $\Rightarrow$ For string Q and regex R of lengths n and $m = O(\sqrt{w})$ resp. regex matching can be solved in $O(m^2 + n)$ time and $O(m)$ space.

Another Algorithm

- Main bottleneck: Need an $\Omega(m^2)$ length string to represent the transitive closure of ϵ-transitions.
- Idea: Compute a "good" separator for TNFAs and use a Divide-and-Conquer strategy.



- There exists two states θ_{P_I} and ϕ_{P_I} whose removal partitions a TNFA into two subgraphs, P_I and P_O, of roughly equal size.

- Any path from P_O to P_I goes through θ_{P_I}

- Any path from P_I to P_O goes through ϕ_{P_I}

Recursive Closure

- Determine which of θ_{P_I} and ϕ_{P_I} are ϵ-reachable
- Update the state-set.
- Recurse in parallel on P_I and P_O.

- $O(\log m)$ levels of recursion, each level can be handled in parallel in $O(1)$ time.

- **Lemma** For TNFAs with $m = O(w)$ states we can support Move and Close in $O(\log m)$ time using $O(m)$ space and $O(m \log m)$ preprocessing.

- $\Rightarrow$ For string Q and regex R of lengths n and $m = O(w)$ resp. regex matching can be solved in $O(n \log m + m \log m)$ time and $O(m)$ space.