

Improved Extended Regular Expression Matching

Philip Bille

Inge Li Gørtz

Rikke Schjeldrup Jessen

Extended Regular Expressions

- Extended regular expressions.

- Patterns formed by characters from an alphabet Σ combined with operators:
 - union $S|T$
 - concatenation $S \cdot T$
 - Kleene star S^*
 - Intersection $S \cap T$
 - Complement $\neg S$

extended operators

- Regular languages.

- $L(a) = \{a\}$
- $L(S|T) = L(S) \cup L(T)$
- $L(S \cdot T) = L(S) \cdot L(T)$
- $L(S^*) = \{\epsilon\} \cup L(S) \cup L(S)^2 \cup L(S)^3 \dots$
- $L(S \cap T) = L(S) \cap L(T)$
- $L(\neg S) = \Sigma^* - L(S)$

- Example.

- $\Sigma = \{a, b\}$
- $R = (a|b)^* ab (a|b)^* \cap (a|b)^* ba (a|b)^*$
- $L(R) = \{abba, baab, aabba, abaab, \dots\}$

Extended Regular Expression Matching

- Regular Expression Matching.
 - Given an extended regular expression R and a string Q , decide if $Q \in L(R)$.

Extended Regular Expression Matching

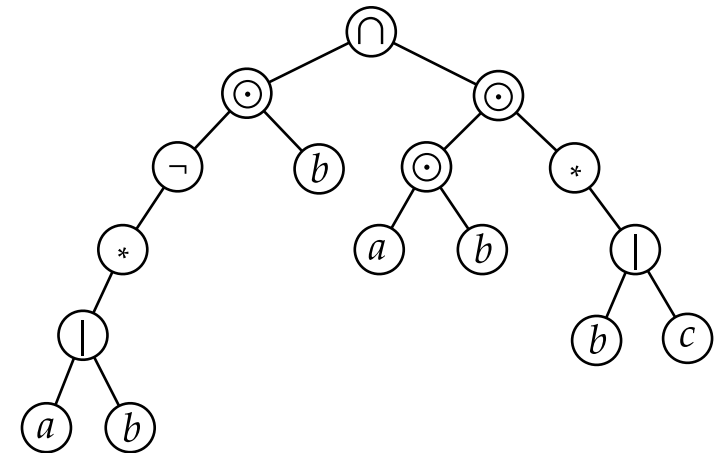
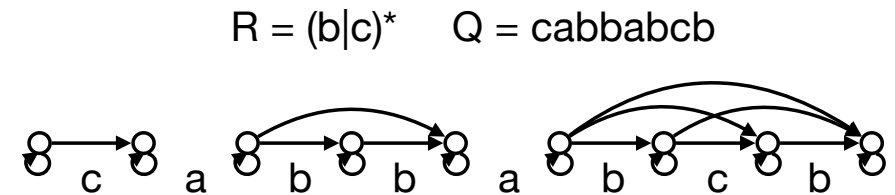
Time	Space	Reference
$O(n^3m)$	$O(n^2m)$	Hopcroft and Ullman, 1979
$O(n^{\omega}m)$	$O(n^2m)$	Hopcroft and Ullman, 1979 + matrix mul.
$O\left(\frac{n^3k + n^2m}{w} + n + m\right)$	$O\left(\frac{n^2k + nm}{w} + n + m\right)$	Yamamoto and Miyazaki, 2003
$O(n^3\lambda + n^2m)$	$O(n^2\lambda + nm)$	Yamamoto, 2001
$\Omega(n^{\omega-\epsilon}\text{poly}(m))$		Nogami and Terauchi, 2026
$O(n^{\omega}k + n^2m)$	$O(n^2 + m)$	This paper
$O\left(n^{\omega}k + \frac{n^2m}{\max(w/\log w, \log(n+m))} + m\right)$	$O(n^2 + m)$	This paper

$m = |R|$, $n = |Q|$, k = number of extended operators, λ = number of critical modules.
Word RAM with word length $w \geq \log n$.

Dynamic Programming Algorithm

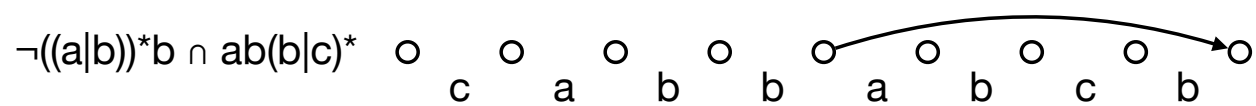
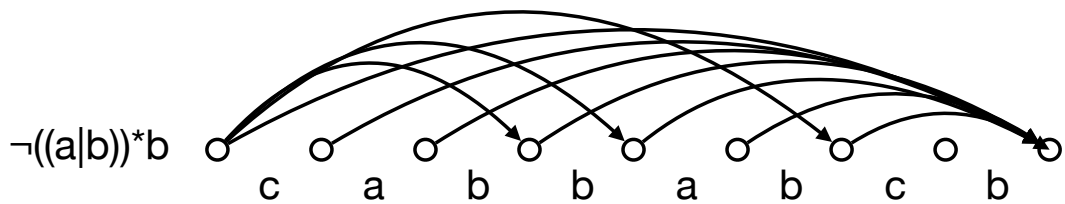
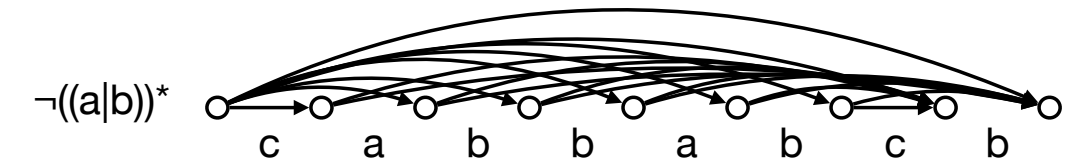
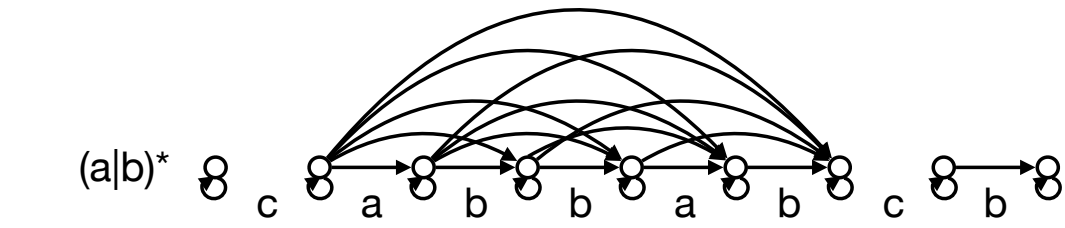
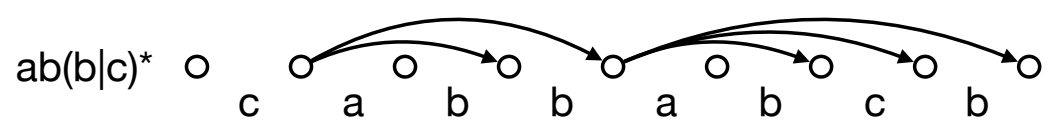
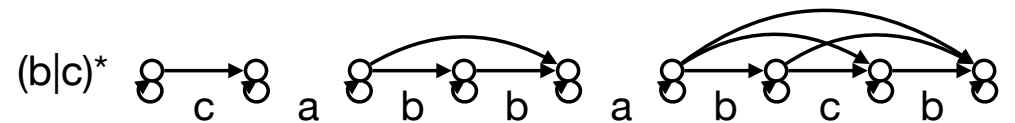
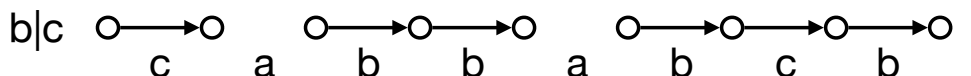
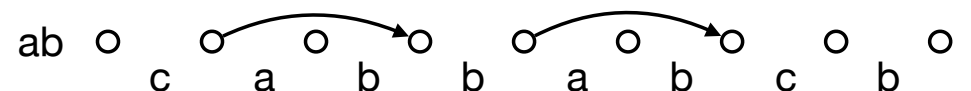
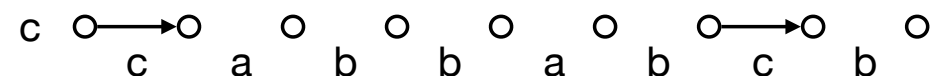
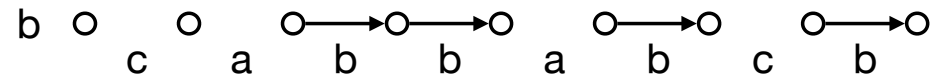
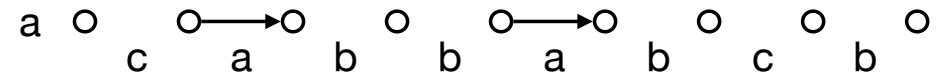
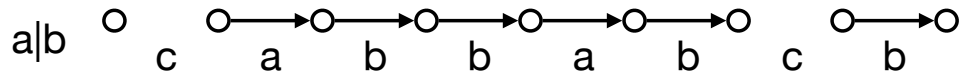
Dynamic Programming Algorithm

- **Match graph of R and Q.**
 - Encodes all substrings of Q that match R.
 - Directed graph on $n+1$ vertices $1, 2, \dots, n+1$.
 - Edge $(i,j) \Leftrightarrow Q[i, j-1]$ matches R.
 - Represent as a $(n+1) \times (n+1)$ boolean matrix.
- **Dynamic programming algorithm.**
 - Compute match graphs for all subexpressions of R in a bottom-up traversal of the parse tree.
 - Combine match graphs using fast matrix operations.



$$R = \neg((a|b))^*b \cap ab(b|c)^*$$

$$Q = cabbabcb$$



Dynamic Programming Algorithm

- Match graph operations.

Expression	Match graph matrix operation	Time
$S T$	or	$O(n^2)$
$S \cdot T$	multiplication	$O(n^\omega)$
$\neg S$	complement	$O(n^2)$
S^*	transitive closure	$O(n^\omega)$
$S \cap T$	and	$O(n^2)$

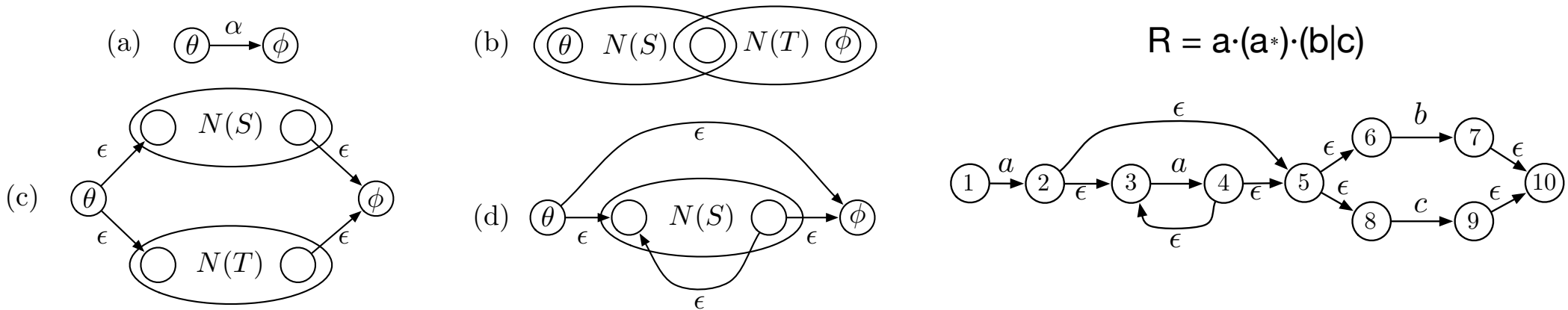
- $\Rightarrow O(n^\omega m)$ time and $O(n^2 m)$ space algorithm.

Improved Dynamic Programming Algorithm

Improved Dynamic Programming Algorithm

- Goal.
 - $O(n^{\omega}k + n^2m)$ time and $O(n^2m)$ space.
- Algorithmic idea.
 - Combine automata-based techniques for non-extended parts of the regular expression with a dynamic programming approach for the extended parts.

Non-extended Regular Expression Matching



- Thompson's NFA [Thompson, 1968].

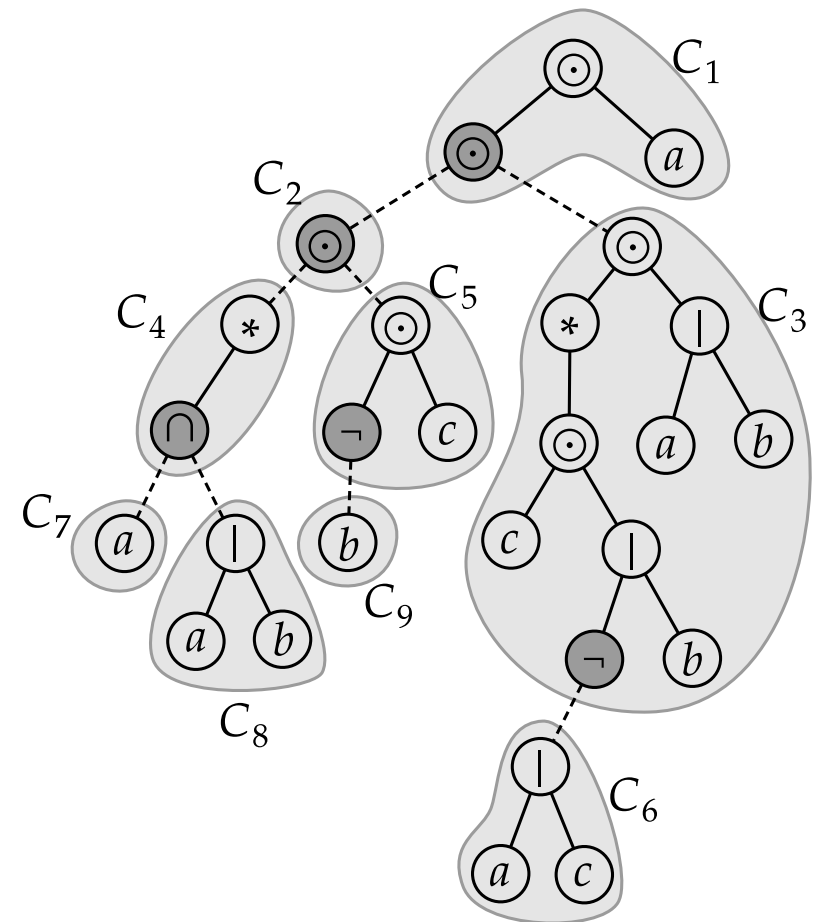
- Recursively construct non-deterministic finite automaton (NFA) A from R.
- A **accepts** the strings in $L(R)$. Each path from the start state to the accept state corresponds to a string in $L(R)$.
- $O(m)$ states and transitions.

- Regular expression matching.

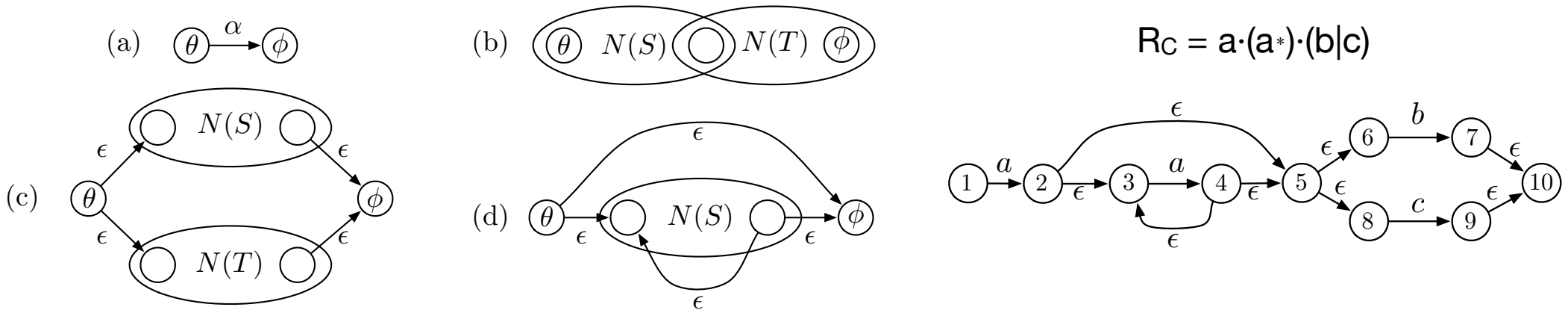
- Simulate A on Q one character at a time.
- $O(m)$ per character $\Rightarrow O(nm)$ time algorithm.

Clustering

- **Mark nodes in the parse tree.**
 - Nodes with an extended operator.
 - Lowest common ancestor of such nodes.
 - Cut edges to children of nodes.
- **Clusters properties.**
 - $O(k)$ node-disjoint connected subtrees.
 - Leaf cluster: non-extended regular expression.
 - Internal cluster: non-extended regular expression + 1 marked node.
- **Improved dynamic programming algorithm.**
 - Dynamic programming on the **cluster tree**.



Leaf Cluster



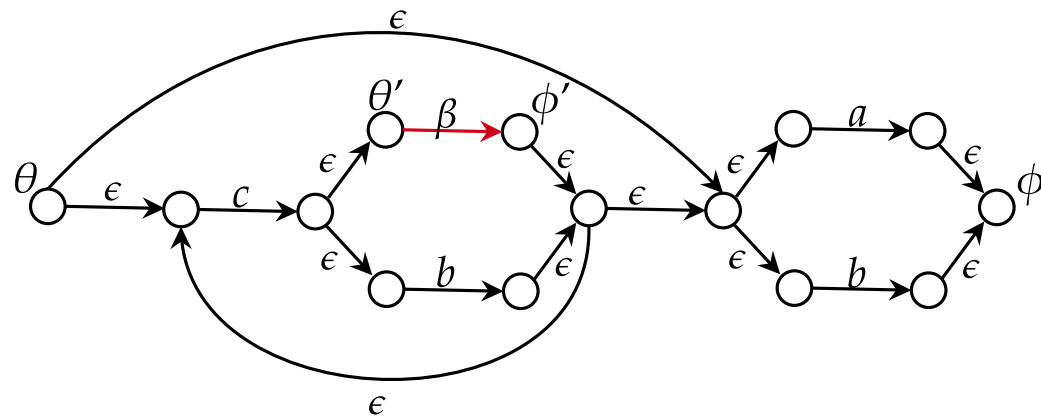
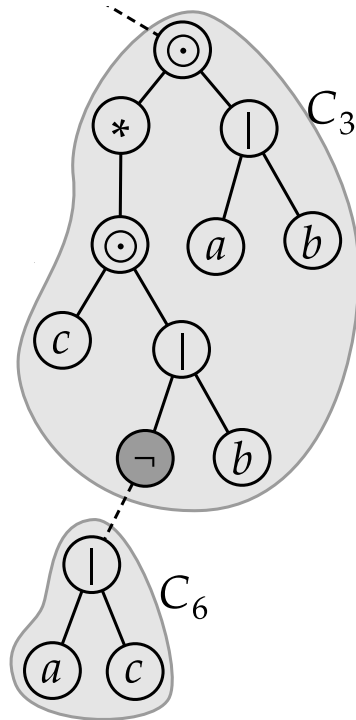
- Leaf cluster representation.

- Construct Thompson's NFA A_C from the subexpression R_C .

- Match graph computation for cluster.

- For each suffix $Q[i,n]$:
 - Traverse A on Q one character at a time and record whenever we reach the accept state.
- $O(m_C)$ per char $\Rightarrow O(nm_C)$ time per suffix $\Rightarrow O(n^2m_C)$ time to compute match graph for C .

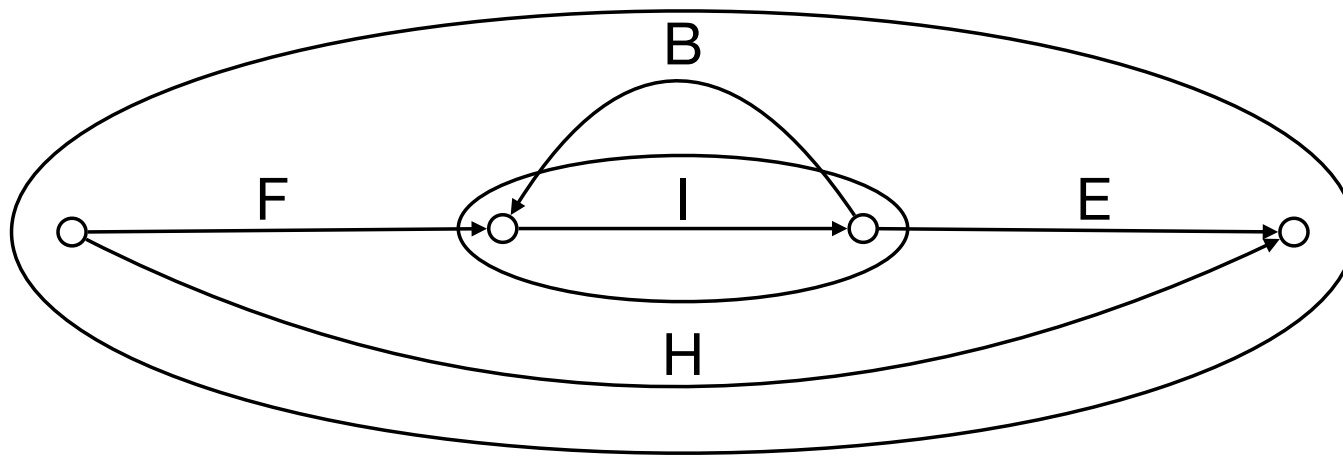
Internal Cluster



- Internal cluster representation.

- Thompson's NFA with a special transition labeled $\beta \notin \Sigma$ that represents the marked node.
- Match graph I that represents the substrings of Q that can replace β .

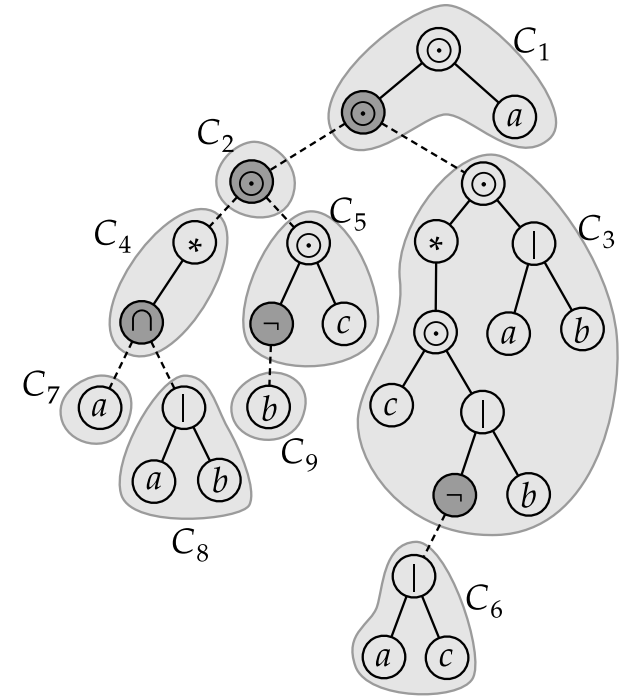
Match Graph Computation for Cluster



$$G = H \mid F \cdot I \cdot (B \cdot I)^* \cdot E$$

Improved Dynamic Programming Algorithm

- Match graph computation.
 - $O(n^2 m_C)$ time for leaf cluster.
 - $O(n^\omega + n^2 m_C)$ time for internal cluster.
- $\Rightarrow O(n^\omega k + n^2 m)$ time and $O(n^2 k + m)$ space algorithm.



Improved Dynamic Programming Algorithm

- **Space reduction.**

- Improve space via heavy-path decomposition and traversal of cluster tree.
- $\Rightarrow O(n^{\omega}k + n^2m)$ time and $O(n^2 + m)$ space algorithm.

- **Black-box NFA simulation reduction.**

- Plug in any (almost) state-of-the-art regular expression matching algorithm instead of Thompson.

- $\Rightarrow O\left(n^{\omega}k + \frac{n^2m}{\max(w/\log w, \log(n+m))} + m\right)$ time and $O(n^2 + m)$ space algorithm.

- **Intervals.**

- The interval operator $R^{[i,j]}$ represents between i and j copies of R .
- $L(R^{[i,j]}) = L(R)^i \cup L(R)^{i+1} \cup L(R)^{i+2} \cup \dots \cup L(R)^j$.
- Include $R^{[i,j]}$ in the algorithm with logarithmic overhead per interval operator.