Regular Expression Matching: History, Status, and Challenges

Philip Bille

Outline

- The problem
- Applications
- Tour of techniques for worst-case efficient regular expression matching
  - NFAs and state-set simulation.
  - NFA decompositions and micro TNFAs.
  - Tabulation-based micro TNFA simulation.
  - Word-level parallel micro TNFA simulation.
  - 2D decomposition algorithm.
- Open problems

Regular Expressions

- A character α is a regular expression.
- If S and T are regular expressions, then so is
  - The *union* S | T
  - The *concatenation* ST (S·T)
  - The *kleene star* S*

Languages

- The *language* L(R) of a regular expression R is:
- $L(\alpha) = \{\alpha\}$
- $L(S|T) = L(S) \cup L(T)$
- $L(ST) = L(S)L(T)$
- $L(S^*) = \{\varepsilon\} \cup L(S) \cup L(S)^2 \cup L(S)^3 \cup \dots$

Example

- R = a(a*)(b|c)
- L(R) = {ab, ac, aab, aac, aaab, aaac, ...}

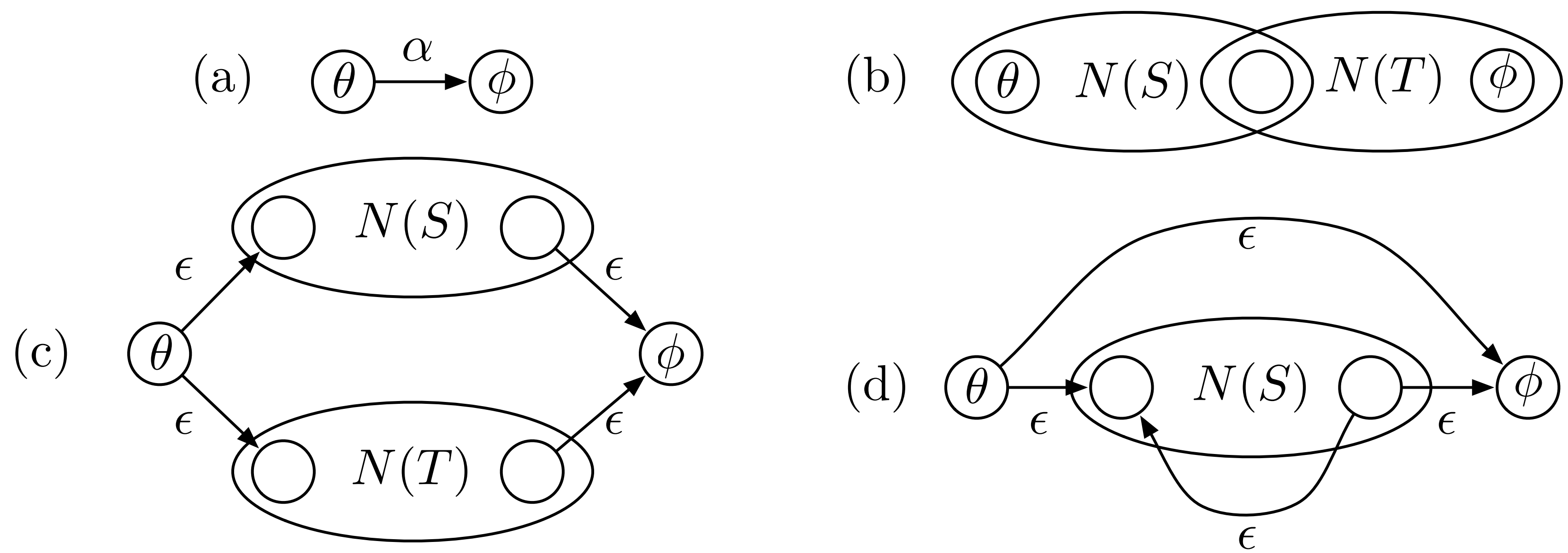
Regular Expression Matching

- Given regular expression R and string Q the regular expression matching problem is to decide if $Q \in L(R)$.
- How fast can we solve regular expression matching for $|R| = m$ and $|Q| = n$?

Applications

- Primitive in large scale data processing:
    - Internet Traffic Analysis
    - Protein searching
    - XML queries
- Standard utilities and tools
    - Grep and Sed
    - Perl

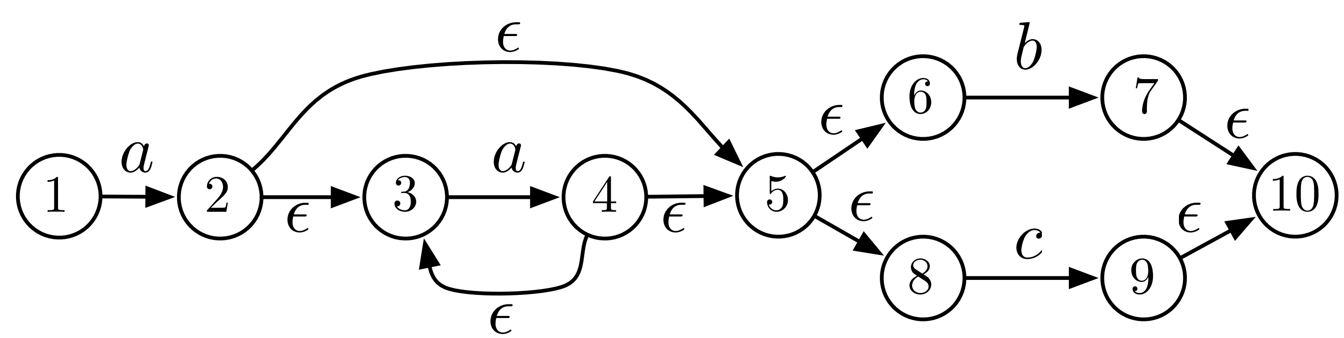
NFAs and State-Set Simulation



- Construct non-deterministic finite automaton (NFA) from R.

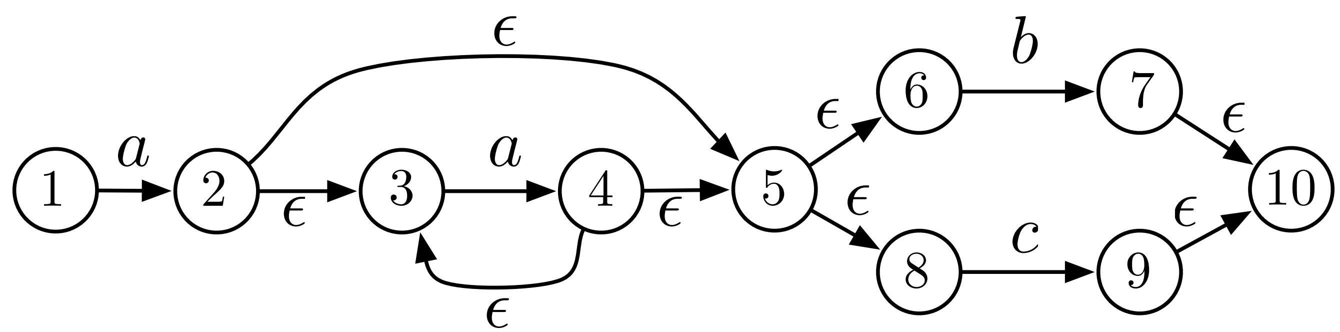
NFAs and State-Set Simulation

R = a·(a*)·(b|c)



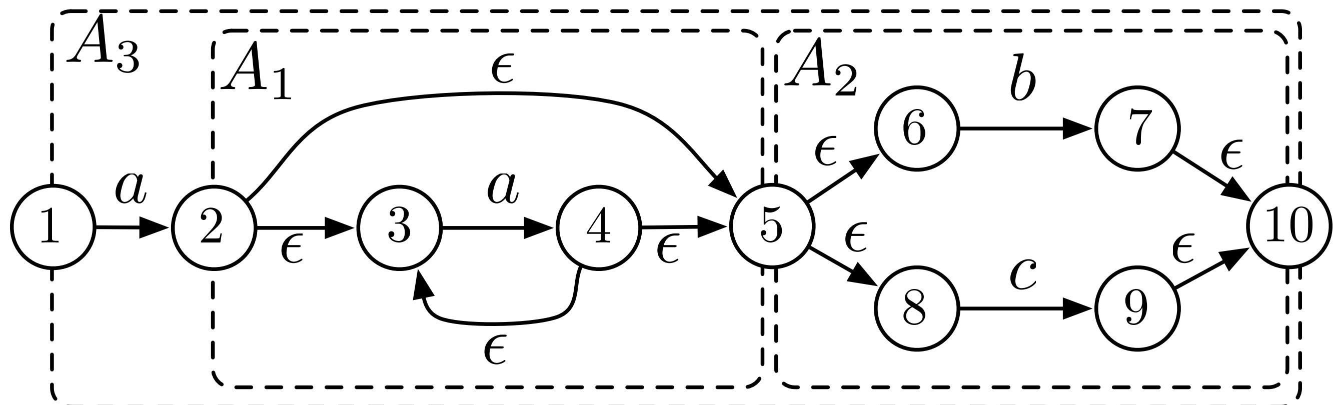
- *Thompson NFA* (TNFA) N(R) has O(|R|) = O(m) states and transitions.
- N(R) *accepts* L(R). Any path from start to accept state corresponds to a string in L(R) and vice versa.
- To solve regular expression traverse TNFA on Q one character at a time (*state-set transition*).
- O(m) per character => O(|Q|m) = O(nm) time algorithm [Thompson1968].
- Top ten list of problems in stringology 1985 [Galil1985].

Large and Small TNFAs



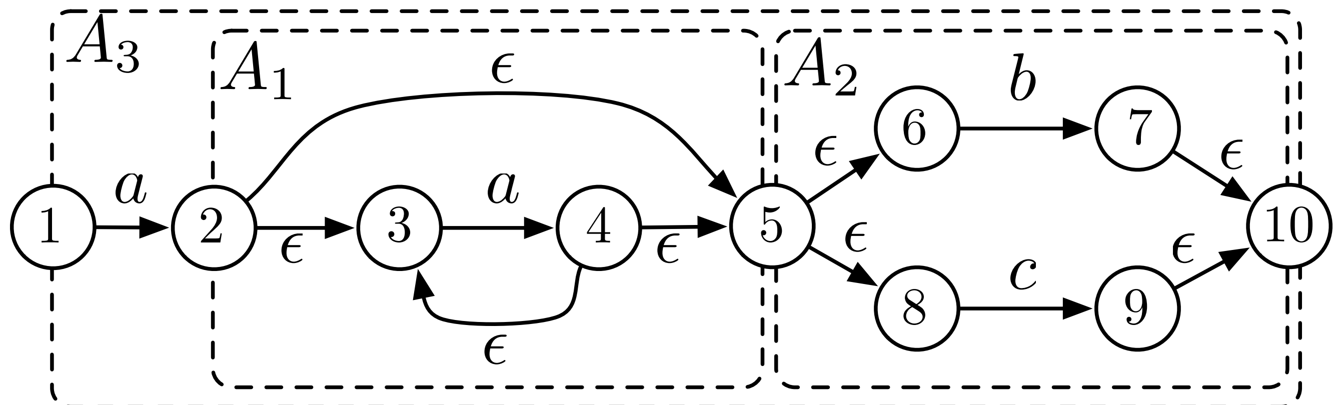
- Suppose we can do state-set transition fast on a *micro TNFA* of size x ≪ m.
- Can we use that to get efficient state-set transition for N(R)?
- Main problem is non-local dependencies from ε-transitions.

Large and Small TNFAs



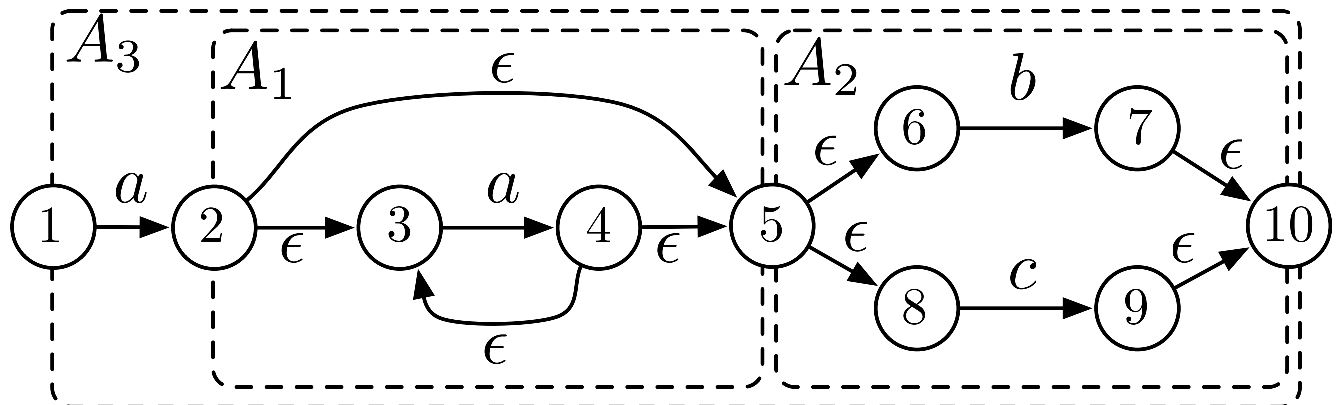
- Decompose N(R) into tree of O(m/x) micro TNFAs with at most x states. Each micro TNFA overlaps with enclosing micro TNFA in 2 states.
- To do state-set transition for N(R) using state-set simulation for micro TNFAs process micro TNFAs in topological order *twice.* Propagate reachable overlapping states.
- State-set transition for micro TNFA in time t(x) => state-set transition for N(R) in time O(m t(x) /x). [Myers1992, B2006]

Tabulation for Micro TNFAs



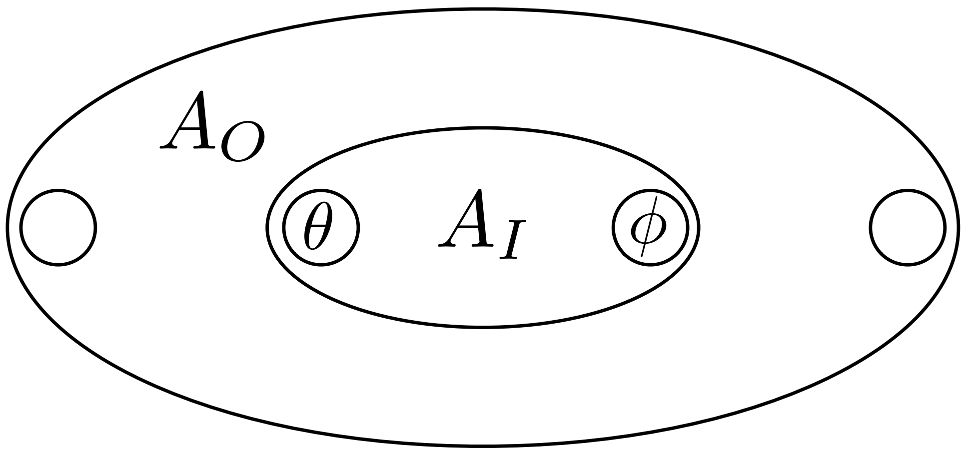
- Encode micro TNFA and state set in micro TNFA in O(x) bits [Myers 1992, BFC2005].
- Tabulate state-set transition for all possible micro TNFAs and state-sets (*determinize* micro TNFA). Table size: $2^{O(x)}$.
- With x = Θ(log n) => O(nm/log n) time and O(m + n^{ε}) space algorithm.

Word-Level Parallelism for Micro TNFAs



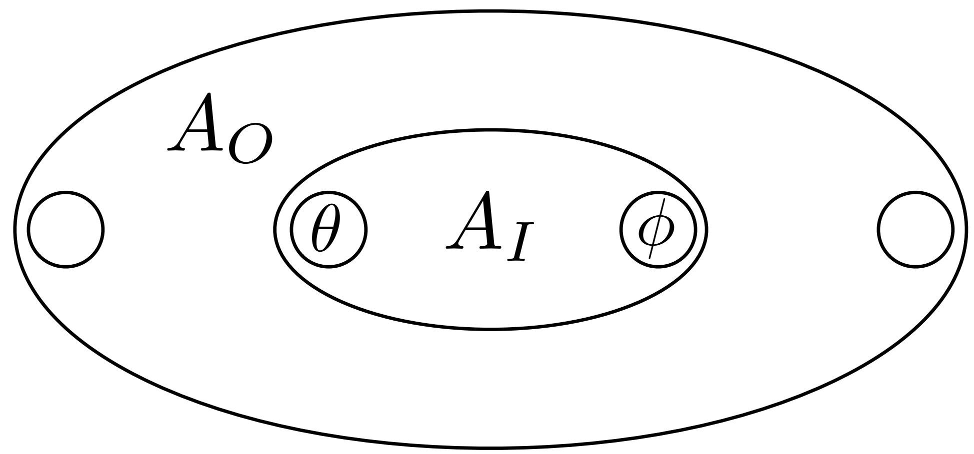
- Can we simulate micro TNFA with bitwise logical and arithmetic operations of the w-bit words instead of tabulation?
- Main challenge is long ε-transitions.

Micro TNFA Separators



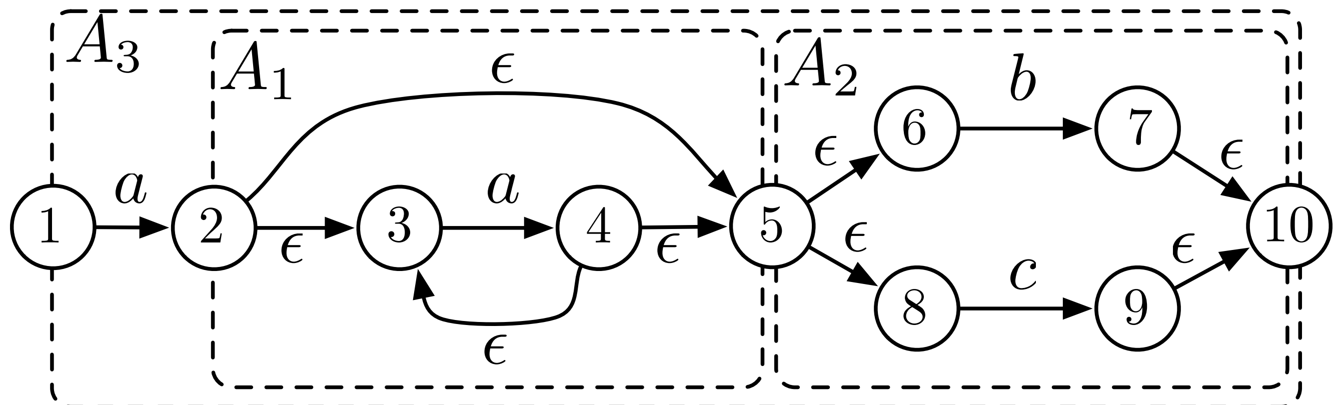
- There exists two states θ and ϕ whose removal partitions a micro TNFA A into two subgraphs, A_O and A_I, of roughly equal size such that:
- Any path from A_O to A_I goes through θ.
- Any path from A_I to A_O goes through ϕ.

Recursive Word-Level Parallel State-Set Transition



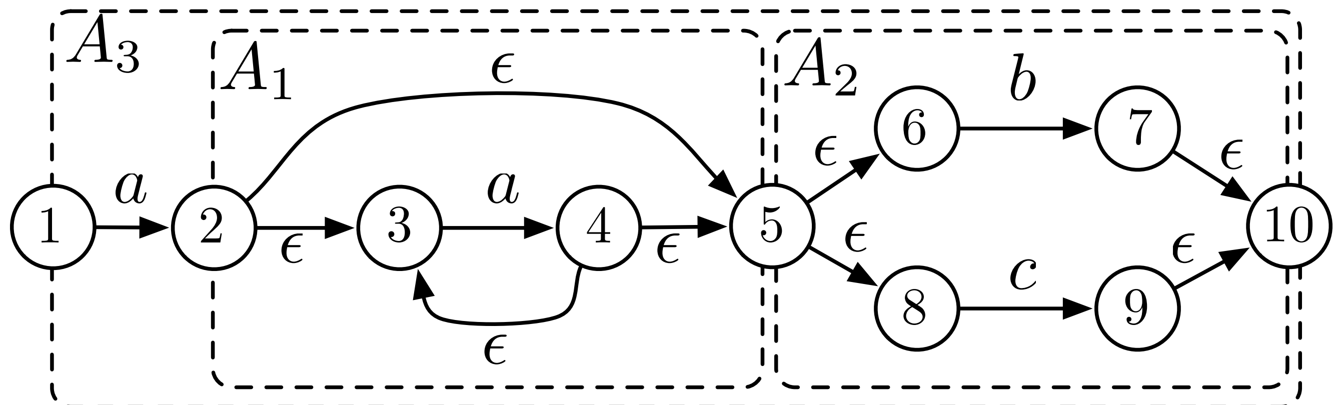
- Compute which of θ and ϕ are reachable.
- Update current set of reachable states
- Recurse on A_O and A_I *in parallel*.
- O(log w) levels of recursion each using O(1) time => O(m log w/ w) state-set transition => O(nm log w/w) time and O(m) space algorithm [B2006].

Beyond State-Set Simulation



- To explicitly read/write state-sets at each character we need Ω(m/w) time for state-set transition.
- => Any algorithm takes Ω(nm/w) time with this approach.
- Can we process multiple characters quickly?
- Even larger challenges from non-local ε-transitions.

2D Decomposition Algorithm



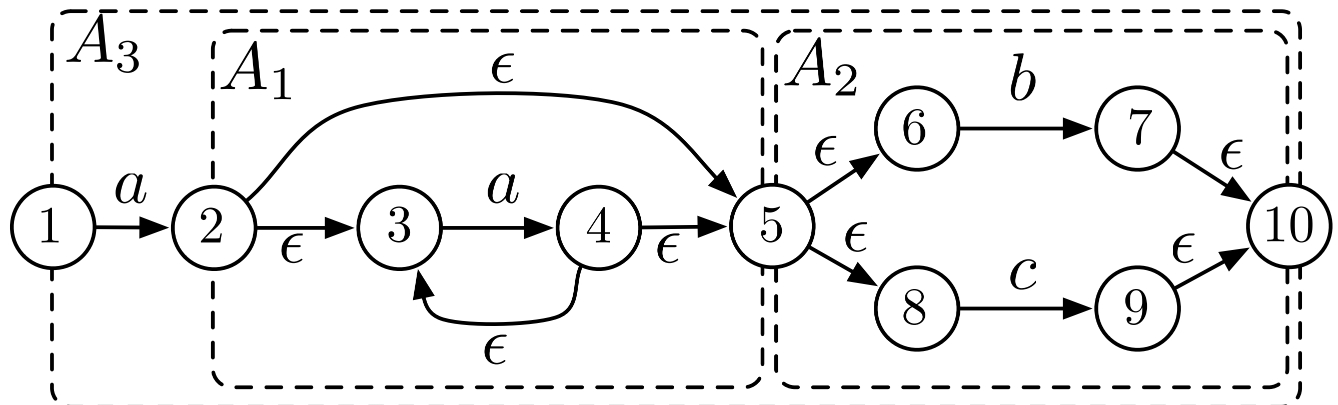
- Decompose N(R) into O(m/x) micro TNFAs with at most x = Θ(log n) states [as earlier).
- Partition Q into segments of length y = Θ($\log^{1/2}$ n).
- State-set transition on segments in O(m/x) time.
- => algorithm using O(nm/xy) = O(nm/$\log^{1,5}$n) time.

2D Decomposition Algorithm: Overview

- Goal: Do a state set transition on y = $\Theta(\log^{1/2} n)$ characters in O(m/x) = O(m/log n) time.
- Algorithm: 4 traversals on tree of micro TNFAs.
  - 1-3 iteratively "builds" information.
  - 4 computes the actual state-set transition.
- Tabulation to do each traversal in constant time per micro TNFA

Computing Accepted Substrings

q = aab

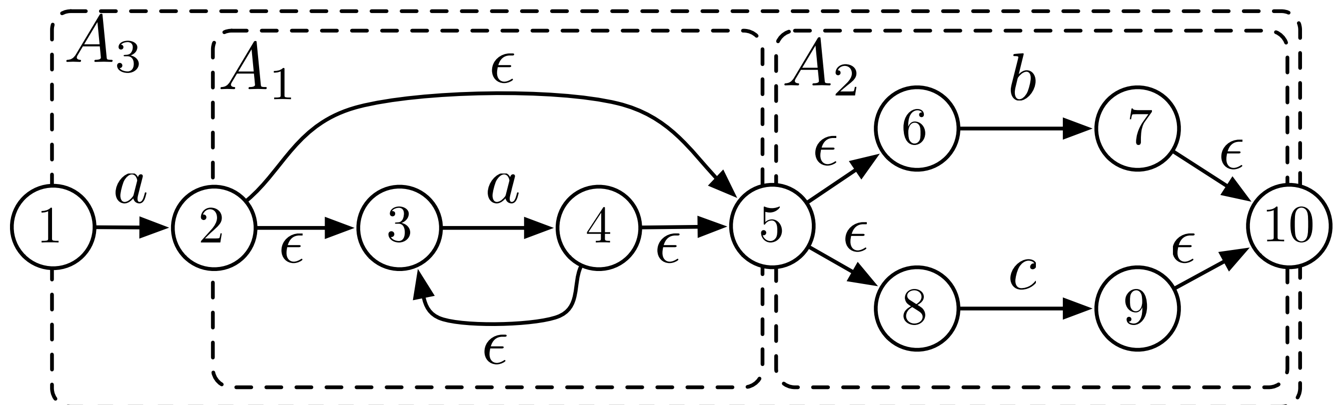


- Goal: For micro TNFA A compute the substrings of q that are accepted by $\bar{A}$. We have A_1 : {ε,a,aa}, A_2 : {b}, A_3 : {ab,aab}.
- Bottom-up traversal using tabulation in constant time per micro TNFA.
- Encode set of substrings in $O(y^2) = O(\log n)$ bits.
- Table input: micro TNFA, substrings of children, q.
- Table size $2^{O(x + y^2 + y)} = 2^{O(x + y^2)} = O(n^\varepsilon)$.

Computing Path Prefixes to Accepting States

q = aab

S = {1,3}

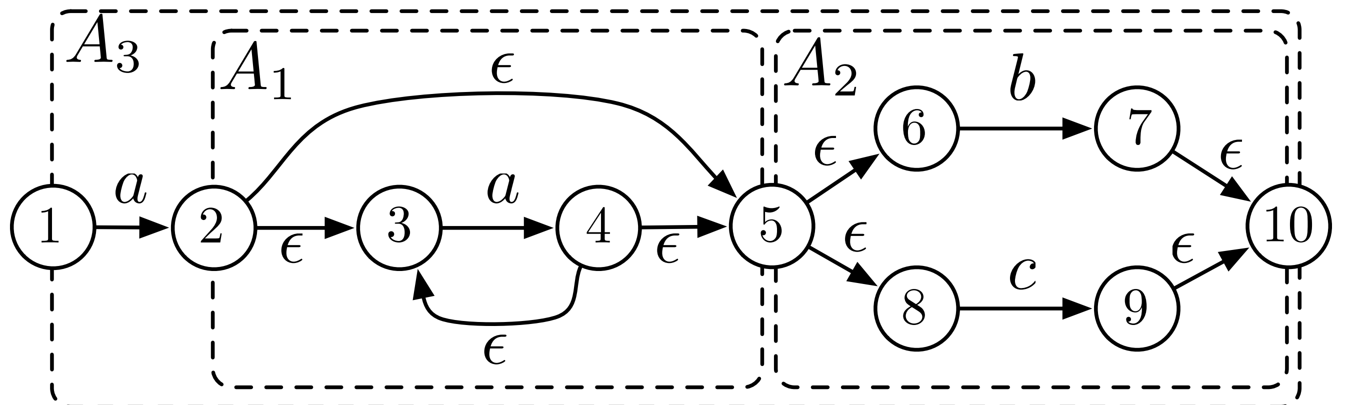


- Goal: For micro TNFA A compute the prefixes of q matching a path from S to the accepting state in $\bar{A}$. We have A_1 : {a, aa}, A_2 : ∅, A_3 : {aab}.
- Bottom-up traversal using tabulation in constant time per micro TNFA.
- Encode prefixes in $O(y) = O(\log^{1/2} n)$ bits.
- Table input: micro TNFA, substrings and path prefixes of children, q, state-set for A.
- Table size $2^{O(x + y^2)} = O(n^{\varepsilon})$.

Computing Path Prefixes to Start States

q = aab

S = {1,3}

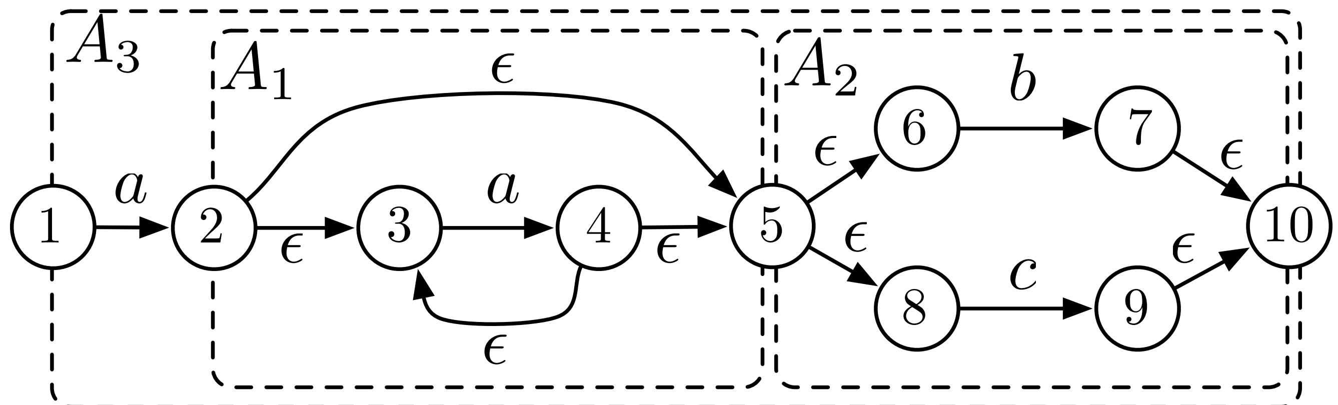


- Goal: For micro TNFA A compute the prefixes of q matching a path from S to the start state in N(R). We have A_1 : {a}, A_2 : {a, aa}, A_3 : {ε}.
- Top-down traversal using tabulation in constant time per micro TNFA.
- Tabulation: Similar to previous traversal.

Updating State-Sets

q = aab

S = {1,3}



- Goal: For micro TNFA A compute the next state-set. We have $A_1 : \varnothing$, A_2 : {7,10}, A_3 : {10}.
- Traversal using tabulation in constant time per micro TNFA.
- Tabulation: Similar to previous traversal.

2D Decomposition Algorithm: Algorithm Summary

- Tabulation in $2^{O(x + y^2)} = O(n^\varepsilon)$ time and space.
- 4 traversals each using $O(m/x)$ time to process length y segment of Q.
- => algorithm using $O(nm/xy) = O(nm/\log^{1,5} n)$ time and $O(n^\varepsilon)$ space [BT2009]

Challenges

- Better than polylog improvements of $O(nm)$ algorithm?
- Hardness reductions to other problems?

References

- S. C. Kleene. Representation of events in nerve nets and finite automata. In Automata Studies, Ann. Math. Stud. No. 34, 1956.
- K. Thompson. Regular expression search algorithm. Comm. ACM, 11:419–422, 1968.
- Z. Galil. Open problems in stringology. In A. Apostolico and Z. Galil, editors, Combinatorial problems on words, NATO ASI Series, Vol. F12, 1985.
- E. W. Myers. A four-russian algorithm for regular expression pattern matching. J. ACM, 39(2):430–448, 1992.
- P. Bille and M. Farach-Colton. Fast and compact regular expression matching. Theoret. Comput. Sci., 409:486 – 496, 2008.
- P. Bille. New algorithms for regular expression matching. In Proc. 33rd ICALP, 2006.
- P. Bille and M. Thorup. Faster regular expression matching. In Proc. 36th ICALP 2009