Linear Time Subsequence and Supersequence Regex Matching

Antoine Amarilli, Florin Manea, Tina Ringleb, Markus L. Schmid

MFCS 2025

August 26, 2025

Overview

1. Preliminaries
2. The $\preceq_{\text{sub}}$-Matching Problem
3. The Min- and Max-Variant of the $\preceq_{\text{sub}}$-Matching Problem
4. The Universal $\preceq_{\text{sub}}$-Matching Problem

Overview

1. Preliminaries
2. The $\preceq_{\text{sub}}$-Matching Problem
3. The Min- and Max-Variant of the $\preceq_{\text{sub}}$-Matching Problem
4. The Universal $\preceq_{\text{sub}}$-Matching Problem

Regular Expressions and εNFAs

regular expression over Σ:

- $\emptyset$ is a regular expression with $L(\emptyset) = \emptyset$
- every $x \in \Sigma \cup \{\varepsilon\}$ is a regular expression with $L(x) = \{x\}$
- if s and t are regular expressions, then the following are regular expressions:
  - $s \cdot t$, with $L(s \cdot t) = L(s) \cdot L(t)$, where $L_1 \cdot L_2 = \{uv \mid u \in L_1, v \in L_2\}$
  - $s \vee t$, with $L(s \vee t) = L(s) \cup L(t)$
  - s^*, with $L(s^*) = (L(s))^*$, where $L^0 = \{\varepsilon\}, L^k = L^{k-1} \cdot L$ for every $k \geq 1$ and $L^* = \bigcup_{k \geq 0} L^k$

Example

$r = (a \cdot b)^* \vee b \cdot a^*,\; L(r) = \{(ab)^k \mid k \geq 0\} \cup \{ba^k \mid k \geq 0\}$

Regular Expressions and εNFAs

non-deterministic finite automaton $A = (Q, \Sigma, q_0, q_f, \delta)$ with ε-transitions:

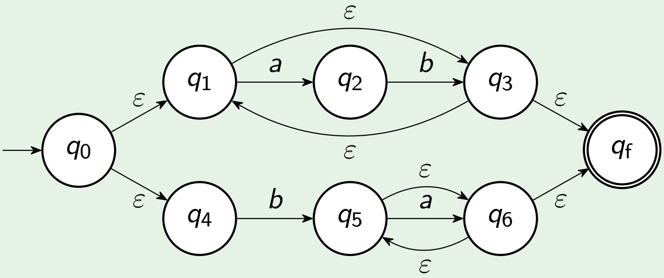
- finite set of states Q with $|Q| = n$, initial state q_0, final state q_f
- set of transitions $\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times Q$ with $|\delta| = |A| = m$
- can be interpreted as a graph with vertex set Q:
  - directed edges labelled by symbols from $\Sigma \cup \{\varepsilon\}$ given by the transitions of δ: $(p, a, q) \in \delta$ corresponds to a directed edge from p to q labelled with a
  - run of A on string w: path from q_0 to some state p which is labelled by w (when ignoring ε-labels); accepting if $p = q_f$
  - $L(A) = \{w \in \Sigma^* \mid \text{ there is an accepting run of } A \text{ on } w\}$

Regular Expressions and εNFAs

Thompson's construction: a regular expression r can be converted in time $O(|r|)$ into an εNFA A such that $L(A) = L(r)$ and $|A| = O(|r|)$

Example

$r = (a \cdot b)^* \vee b \cdot a^*$ corresponds to



String Relations

- string relation $\preceq$ (over Σ): subset of $\Sigma^* \times \Sigma^*$
- $\Lambda_{\preceq}(w) := \{u \in \Sigma^* \mid u \preceq w\}$, i. e. the set of all strings that are in $\preceq$-relation to w
- lift this notation to languages: $\Lambda_{\preceq}(L) = \bigcup_{w \in L} \Lambda_{\preceq}(w)$

prefix	$u \preceq_{\text{pre}} w$	$uv = w$ for some $v \in \Sigma^*$
infix	$u \preceq_{\text{in}} w$	$vuv' = w$ for some $v, v' \in \Sigma^*$
subsequence	$u \preceq_{\text{sub}} w$	$u = w[i_1] \ldots w[i_{\lvert u \rvert}]$, $1 \leq i_1 < \ldots < i_{\lvert u \rvert} \leq \lvert w \rvert$
supersequence	$u \preceq_{\text{sup}} w$	$w \preceq_{\text{sub}} u$
left-extension	$u \preceq_{\text{lext}} w$	$u = vw$ for some $v \in \Sigma^*$
extension	$u \preceq_{\text{ext}} w$	$u = vwv'$ for some $v, v' \in \Sigma^*$

Variants of Regex Matching

regex matching problem	$w \in L(r)$?
εNFA acceptance problem	$w \in L(A)$?
$\preceq$-matching problem	$\Lambda_{\preceq}(w) \cap L(A) \neq \emptyset$?
min-/max-variant	$\underset{u}{\text{argmin}}(/\text{-max})\{\|u\| \mid u \in \Lambda_{\preceq}(w) \cap L(A)\}$?
universal-variant	$\Lambda_{\preceq}(w) \subseteq L(A)$?

Results

①	in	pre	ext/lext	sub	sup
$\preceq$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\| + m)$	$O(\|w\| + m)$
min	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$
max	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$
$\forall$	$O(\|w\|^2 m)$	$O(\|w\|m)$	PSPACE	coNP	PSPACE

②	in/pre	ext/lext	sub	sup
$\preceq$	no $O((\|w\|m)^{1-\epsilon})$	no $O((\|w\|m)^{1-\epsilon})$	—	—
min	no $O((\|w\|m)^{1-\epsilon})$	no $O((\|w\|m)^{1-\epsilon})$	no $O(\|w\| + m)$	no $O((\|w\|m)^{1-\epsilon})$
max	no $O((\|w\|m)^{1-\epsilon})$	no $O((\|w\|m)^{1-\epsilon})$	no $O((\|w\|m)^{1-\epsilon})$	no $O(\|w\| + m)$
$\forall$	no $O((\|w\|m)^{1-\epsilon})$	PSPACE-hard	coNP-hard	PSPACE-hard

Upper bounds ① and (conditional) lower bounds ② for the different problem variants; note that m is the size of the εNFA A

Results

①	in	pre	ext/lext	sub	sup
$\preceq$	$O(\lvert w\rvert m)$	$O(\lvert w\rvert m)$	$O(\lvert w\rvert m)$	$O(\lvert w\rvert + m)$	$O(\lvert w\rvert + m)$
min	$O(\lvert w\rvert m)$	$O(\lvert w\rvert m)$	$O(\lvert w\rvert m)$	$O(\lvert w\rvert m)$	$O(\lvert w\rvert m)$
max	$O(\lvert w\rvert m)$	$O(\lvert w\rvert m)$	$O(\lvert w\rvert m)$	$O(\lvert w\rvert m)$	$O(\lvert w\rvert m)$
$\forall$	$O(\lvert w\rvert^2 m)$	$O(\lvert w\rvert m)$	PSPACE	coNP	PSPACE

②	in/pre	ext/lext	sub	sup
$\preceq$	no $O((\lvert w\rvert m)^{1-\epsilon})$	no $O((\lvert w\rvert m)^{1-\epsilon})$	—	—
min	no $O((\lvert w\rvert m)^{1-\epsilon})$	no $O((\lvert w\rvert m)^{1-\epsilon})$	no $O(\lvert w\rvert + m)$	no $O((\lvert w\rvert m)^{1-\epsilon})$
max	no $O((\lvert w\rvert m)^{1-\epsilon})$	no $O((\lvert w\rvert m)^{1-\epsilon})$	no $O((\lvert w\rvert m)^{1-\epsilon})$	no $O(\lvert w\rvert + m)$
$\forall$	no $O((\lvert w\rvert m)^{1-\epsilon})$	PSPACE-hard	coNP-hard	PSPACE-hard

Upper bounds ① and (conditional) lower bounds ② for the different problem variants; note that m is the size of the εNFA A

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
  - $p \in Q$ is active at step i if $p \in S_i$
  - $w \in L(A)$ if q_f is active at step $|w|$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

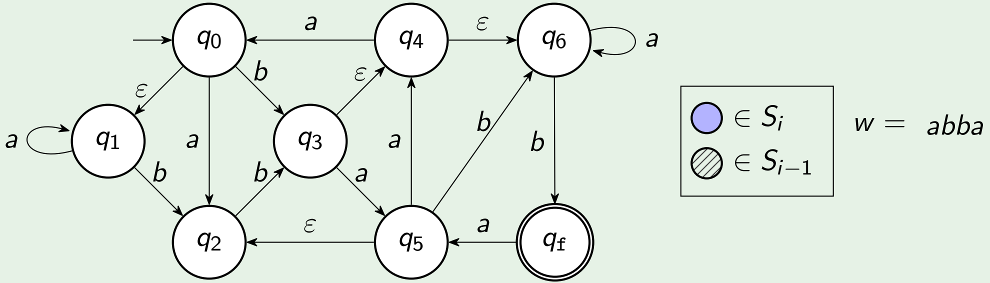
- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
  - $p \in Q$ is active at step i if $p \in S_i$
  - $w \in L(A)$ if q_f is active at step $|w|$
- update step from S_{i-1} to S_i:
  - compute $S_i' = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}_\varepsilon^*(S_i')$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}^*_\varepsilon(S'_i)$

Example

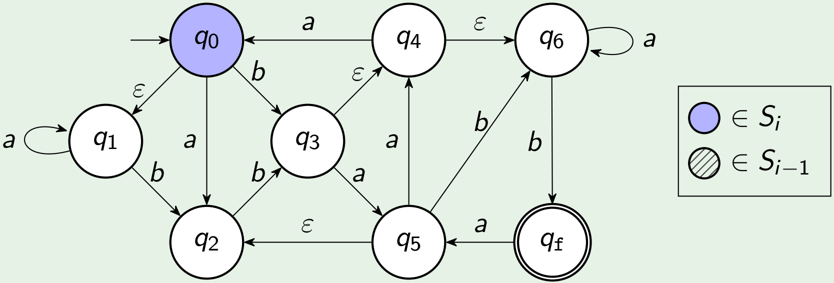


State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \dots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathsf{C}_{w[i]}(S_{i-1})$, where $\mathsf{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathsf{C}^*_\varepsilon(S'_i)$

Example



$w = abba$

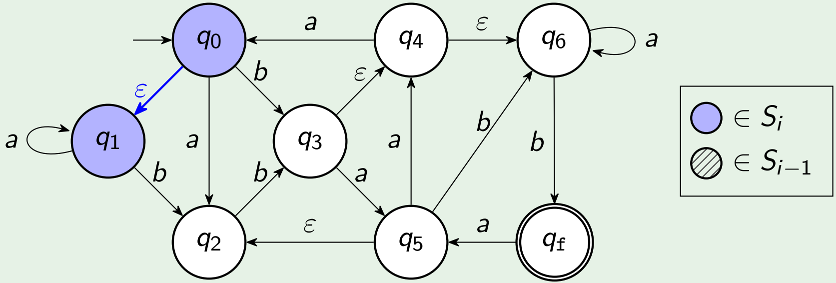
$S'_0 = \{q_0\}$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}^*_\varepsilon(S'_i)$

Example



$w = abba$

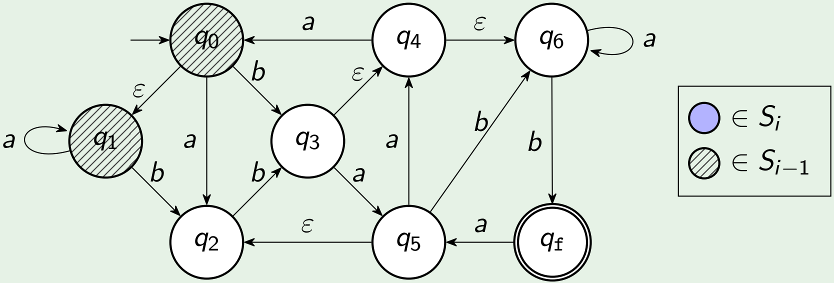
$S_0 = \{q_0\} \cup \{q_1\}$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid$ there is a $w[1:i]$-labelled path from q_0 to $p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}^*_\varepsilon(S'_i)$

Example



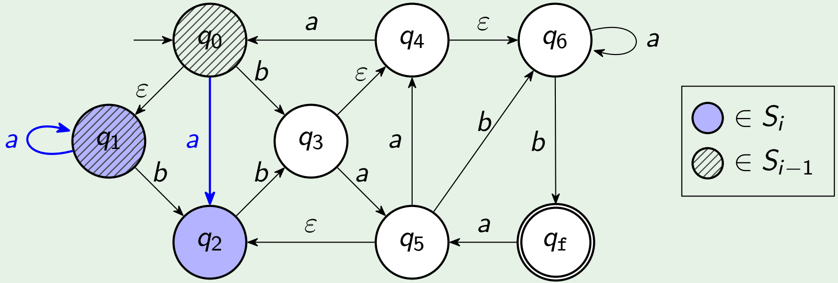
$w = \underline{a}bba$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}^*_\varepsilon(S'_i)$

Example



$w = \underline{a}bba$

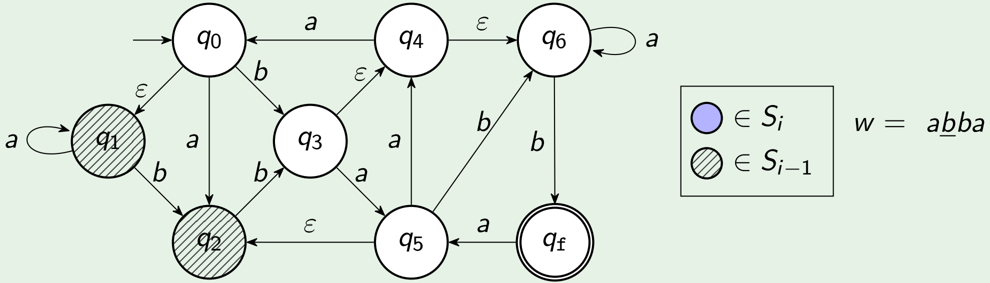
$S_1 = S'_1 = \{q_1, q_2\}$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid$ there is a $w[1:i]$-labelled path from q_0 to $p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathsf{C}_{w[i]}(S_{i-1})$, where $\mathsf{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathsf{C}^*_\varepsilon(S'_i)$

Example

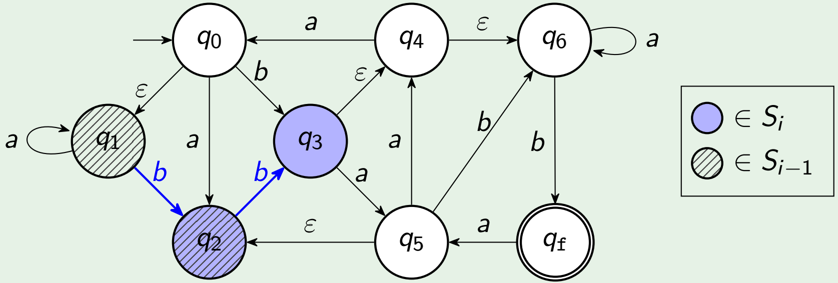


State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid$ there is a $w[1:i]$-labelled path from q_0 to $p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}^*_\varepsilon(S'_i)$

Example



$w = a\underline{b}ba$

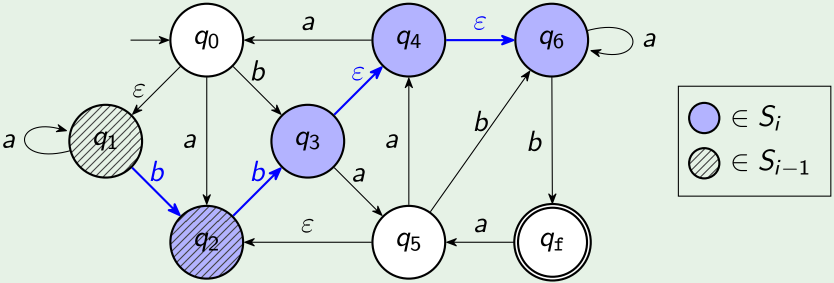
$S'_2 = \{q_2, q_3\}$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}^*_\varepsilon(S'_i)$

Example



$w = abba$

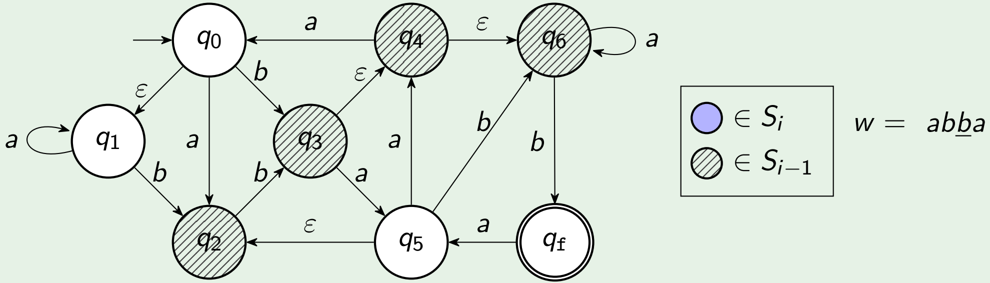
$S_2 = \{q_2, q_3\} \cup \{q_4, q_6\}$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S_i' = \mathsf{C}_{w[i]}(S_{i-1})$, where $\mathsf{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathsf{C}_\varepsilon^*(S_i')$

Example

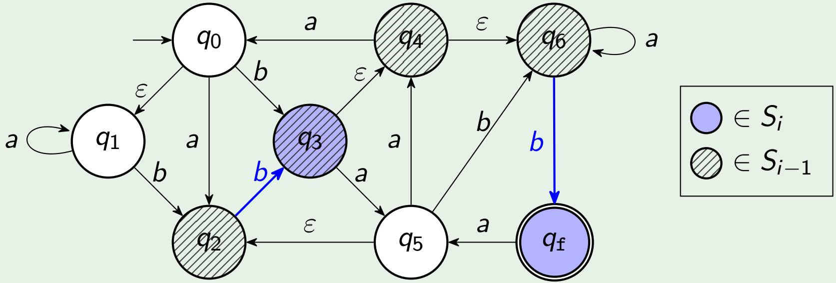


State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathsf{C}_{w[i]}(S_{i-1})$, where $\mathsf{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathsf{C}^*_\varepsilon(S'_i)$

Example



$w = ab\underline{b}a$

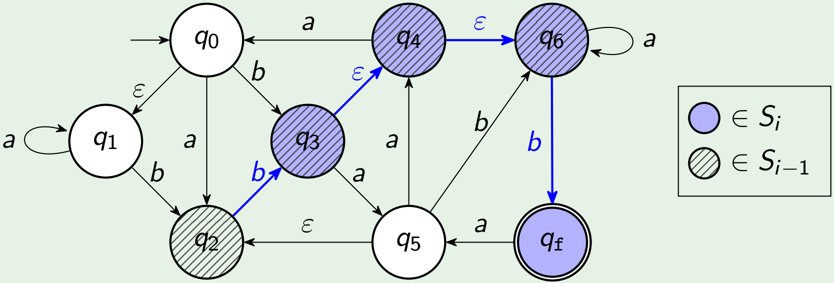
$S'_3 = \{q_3, q_f\}$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S_i' = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}_\varepsilon^*(S_i')$

Example



$w = ab\underline{b}a$

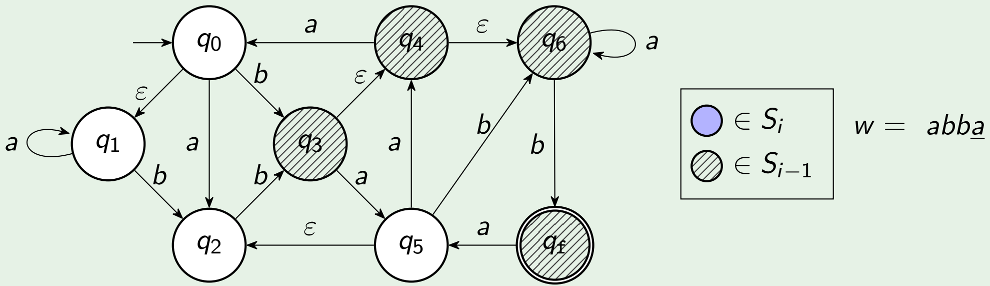
$S_3 = \{q_3, q_f\} \cup \{q_4, q_6\}$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}^*_\varepsilon(S'_i)$

Example

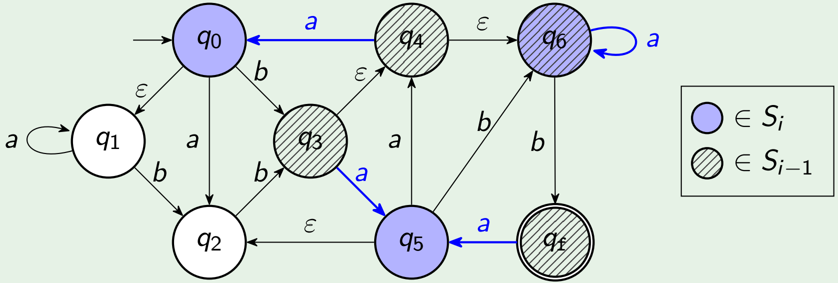


State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \ldots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}^*_\varepsilon(S'_i)$

Example



$w = abb\underline{a}$

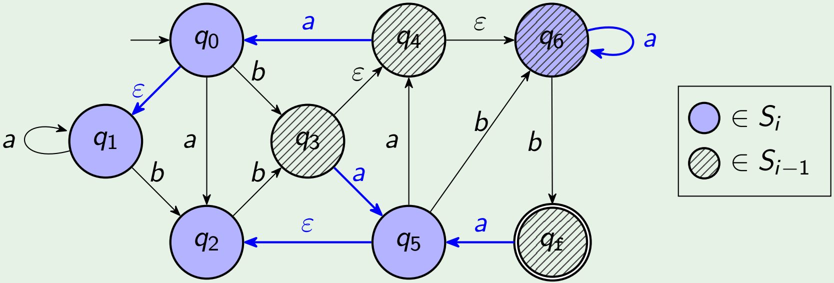
$S'_4 = \{q_0, q_5, q_6\}$

State-Set Simulation

decide whether $w \in L(A)$ in $O(|w||A|)$ time

- $S_i = \{p \in Q \mid \text{there is a } w[1:i]\text{-labelled path from } q_0 \text{ to } p\}$ is the set of active states at step $i \in \{0, 1, \dots, |w|\}$
- update step from S_{i-1} to S_i:
  - compute $S'_i = \mathrm{C}_{w[i]}(S_{i-1})$, where $\mathrm{C}_b(S) = \{q \mid p \in S, (p, b, q) \in \delta\}$
  - compute ε-closure $S_i = \mathrm{C}^*_\varepsilon(S'_i)$

Example



$w = abb\underline{a}$

$S_4 = \{q_0, q_5, q_6\} \cup \{q_1, q_2\}$

Overview

1. Preliminaries
2. The $\preceq_{\text{sub}}$-Matching Problem
3. The Min- and Max-Variant of the $\preceq_{\text{sub}}$-Matching Problem
4. The Universal $\preceq_{\text{sub}}$-Matching Problem

$O(|w| + m)$ Algorithm

Theorem

Given $w \in \Sigma^*$ and εNFA $A = (Q, \Sigma, q_0, q_f, \delta)$ of size m, the $\preceq_{\text{sub}}$-matching problem can be solved in time $O(|w| + m)$.

$O(|w| + m)$ Algorithm

Theorem

Given $w \in \Sigma^*$ and εNFA $A = (Q, \Sigma, q_0, q_f, \delta)$ of size m, the $\preceq_{\text{sub}}$-matching problem can be solved in time $O(|w| + m)$.

- transform A into εNFA A_{sub} accepting the upwards closure of A (i. e., $L(A_{sub}) = \{u \in \Sigma^* \mid \exists v \in L(A)\colon v \preceq_{\text{sub}} u\}$)
  - add transition (p, a, p) to A_{sub} for every $p \in Q, a \in \Sigma$ (= ignore letters from w)
  - 'non-ignoring' transitions of an accepting run of w on A_{sub} spell out a subsequence of w accepted by A

$O(|w| + m)$ Algorithm

Theorem

Given $w \in \Sigma^*$ and εNFA $A = (Q, \Sigma, q_0, q_f, \delta)$ of size m, the $\preceq_{\text{sub}}$-matching problem can be solved in time $O(|w| + m)$.

- transform A into εNFA A_{sub} accepting the upwards closure of A (i. e., $L(A_{sub}) = \{u \in \Sigma^* \mid \exists v \in L(A) \colon v \preceq_{\text{sub}} u\}$)
  - add transition (p, a, p) to A_{sub} for every $p \in Q, a \in \Sigma$ (= ignore letters from w)
  - 'non-ignoring' transitions of an accepting run of w on A_{sub} spell out a subsequence of w accepted by A
- [Bachmeier, Luttenberger, Schlund '15]: when a state p of A_{sub} is added to the set of active states, it stays active until the end of the state-set simulation
  - $S_0 \subseteq S_1 \subseteq \ldots \subseteq S_{|w|}$
  - at most $n + 1$ different sets of active states

$O(|w| + m)$ Algorithm

state-set simulation on A_{sub} is too slow: run state-set simulation on A while only considering 'new' active states at each step

$O(|w| + m)$ Algorithm

state-set simulation on A_{sub} is too slow: run state-set simulation on A while only considering 'new' active states at each step

- computing $\mathsf{C}_{w[i]}(S_{i-1})$: if $p \in S_{i-1}$ and state q is added due to transition $(p, w[i], q)$, then $q \in S_j$ holds for all $j \geq i$

$O(|w| + m)$ Algorithm

state-set simulation on A_{sub} is too slow: run state-set simulation on A while only considering 'new' active states at each step

- computing $\mathrm{C}_{w[i]}(S_{i-1})$: if $p \in S_{i-1}$ and state q is added due to transition $(p, w[i], q)$, then $q \in S_j$ holds for all $j \geq i$
  - transition $(p, w[i], q)$ can be ignored afterwards
  - only consider *unmarked* transitions $(p, w[i], q)$ with $p \in S_{i-1}$
  - mark transition $(p, w[i], q)$ after use
- computing $\mathrm{C}_{\varepsilon}^{*}(S')$: same idea for transitions (p, ε, q) with $p \in S_{i-1}$

$O(|w| + m)$ Algorithm

state-set simulation on A_{sub} is too slow: run state-set simulation on A while only considering 'new' active states at each step

- computing $\mathrm{C}_{w[i]}(S_{i-1})$: if $p \in S_{i-1}$ and state q is added due to transition $(p, w[i], q)$, then $q \in S_j$ holds for all $j \geq i$
  - transition $(p, w[i], q)$ can be ignored afterwards
  - only consider *unmarked* transitions $(p, w[i], q)$ with $p \in S_{i-1}$
  - mark transition $(p, w[i], q)$ after use
- computing $\mathrm{C}^*_\varepsilon(S')$: same idea for transitions (p, ε, q) with $p \in S_{i-1}$
- storing relevant transitions: array $H[\cdot]$ of lists, indexed by elements of Σ
  - store all unmarked transitions (p, a, q) with $a \in \Sigma, p \in S_{i-1}$ in list $H[a]$
  - while computing $\mathrm{C}_{w[i]}(S_{i-1})$, we mark all transitions in $H[w[i]]$
  - remove (p, a, q) from $H[a]$ after (p, a, q) has been marked

$O(|w| + m)$ Algorithm

- update step ($S_{i-1} \to S_i$):
  - compute $S' = \mathrm{C}_{w[i]}(S_{i-1})$ using only transitions in $H[w[i]]$ (while marking and removing them from $H[w[i]]$)

$O(|w| + m)$ Algorithm

- update step ($S_{i-1} \to S_i$):
  - compute $S' = \mathsf{C}_{w[i]}(S_{i-1})$ using only transitions in $H[w[i]]$ (while marking and removing them from $H[w[i]]$)
  - store all new states q (i. e., $q \notin S_{i-1}$) in queue R

$O(|w| + m)$ Algorithm

- update step ($S_{i-1} \to S_i$):
  - compute $S' = \mathrm{C}_{w[i]}(S_{i-1})$ using only transitions in $H[w[i]]$ (while marking and removing them from $H[w[i]]$)
  - store all new states q (i. e., $q \notin S_{i-1}$) in queue R
  - compute $S_i = \mathrm{C}^*_{\varepsilon}(R)$, adding to R all new states

$O(|w| + m)$ Algorithm

- update step ($S_{i-1} \to S_i$):
  - compute $S' = \mathrm{C}_{w[i]}(S_{i-1})$ using only transitions in $H[w[i]]$ (while marking and removing them from $H[w[i]]$)
  - store all new states q (i. e., $q \notin S_{i-1}$) in queue R
  - compute $S_i = \mathrm{C}^*_{\varepsilon}(R)$, adding to R all new states
  - update H, only considering transitions (p, a, q) with $p \in R$

$O(|w| + m)$ Algorithm

- update step ($S_{i-1} \to S_i$):
  - compute $S' = \mathrm{C}_{w[i]}(S_{i-1})$ using only transitions in $H[w[i]]$ (while marking and removing them from $H[w[i]]$)
  - store all new states q (i. e., $q \notin S_{i-1}$) in queue R
  - compute $S_i = \mathrm{C}^*_{\varepsilon}(R)$, adding to R all new states
  - update H, only considering transitions (p, a, q) with $p \in R$
- every transition is either
  - not marked at all, or
  - marked in the computation of some ε-closure (including the initialisation) and then ignored, or
  - put into $H[a]$ at some update step (or initialisation) and then marked at some later update step and then ignored

$O(|w| + m)$ Algorithm

- update step ($S_{i-1} \to S_i$):
  - compute $S' = \mathsf{C}_{w[i]}(S_{i-1})$ using only transitions in $H[w[i]]$ (while marking and removing them from $H[w[i]]$)
  - store all new states q (i. e., $q \notin S_{i-1}$) in queue R
  - compute $S_i = \mathsf{C}^*_\varepsilon(R)$, adding to R all new states
  - update H, only considering transitions (p, a, q) with $p \in R$
- every transition is either
  - not marked at all, or
  - marked in the computation of some ε-closure (including the initialisation) and then ignored, or
  - put into $H[a]$ at some update step (or initialisation) and then marked at some later update step and then ignored
- linear time: only $O(m)$ additional time over the whole state-set simulation in addition to the $|w|$ update steps

Overview

1. Preliminaries
2. The $\preceq_{\mathsf{sub}}$-Matching Problem
3. The Min- and Max-Variant of the $\preceq_{\mathsf{sub}}$-Matching Problem
4. The Universal $\preceq_{\mathsf{sub}}$-Matching Problem

$O(|w|m)$ Upper Bound

Theorem

Given $w \in \Sigma^*$ and εNFA $A = (Q, \Sigma, q_0, q_{\mathsf{f}}, \delta)$ of size m, the min- and max-variant of the $\preceq_{\mathsf{sub}}$-matching problem can be solved in time $O(|w|m)$.

- build $\Sigma \cup \{\varepsilon\}$-labelled directed product graph $G_{A,w}$ of size $O(|w|m)$ with edge weights of 0 or 1, source s, and sink t

$O(|w|m)$ Upper Bound

Theorem

Given $w \in \Sigma^*$ and εNFA $A = (Q, \Sigma, q_0, q_f, \delta)$ of size m, the min- and max-variant of the $\preceq_{\text{sub}}$-matching problem can be solved in time $O(|w|m)$.

- build $\Sigma \cup \{\varepsilon\}$-labelled directed product graph $G_{A,w}$ of size $O(|w|m)$ with edge weights of 0 or 1, source s, and sink t
- shortest/longest $u \preceq_{\text{sub}} w$ with $u \in L(A)$ corresponds to minimum/maximum weight s-t-path (path labelled with u, weight $|u|$)
- find shortest/longest path in $O(|G_{A,w}|) = O(|w|m)$ time using basic graph algorithmic techniques

$O(|w|m)$ Upper Bound

Theorem

Given $w \in \Sigma^*$ and εNFA $A = (Q, \Sigma, q_0, q_f, \delta)$ of size m, the min- and max-variant of the $\preceq_{\text{sub}}$-matching problem can be solved in time $O(|w|m)$.

- build $\Sigma \cup \{\varepsilon\}$-labelled directed product graph $G_{A,w}$ of size $O(|w|m)$ with edge weights of 0 or 1, source s, and sink t :
  - vertices $\{0, 1, \ldots, |w|\} \times Q$, $s = (0, q_0)$ and $t = (|w|, q_f)$

$O(|w|m)$ Upper Bound

Theorem

Given $w \in \Sigma^*$ and εNFA $A = (Q, \Sigma, q_0, q_f, \delta)$ of size m, the min- and max-variant of the $\preceq_{\text{sub}}$-matching problem can be solved in time $O(|w|m)$.

- build $\Sigma \cup \{\varepsilon\}$-labelled directed product graph $G_{A,w}$ of size $O(|w|m)$ with edge weights of 0 or 1, source s, and sink t :
  - vertices $\{0, 1, \dots, |w|\} \times Q$, $s = (0, q_0)$ and $t = (|w|, q_f)$
  - transition (p, ε, q) and $i \in \{0, \dots, |w|\}$: add edge from (i, p) to (i, q) with label ε and weight 0
  - transition (p, a, q) and $i \in \{1, \dots, |w|\}$ with $w[i] = a$: add edge from $(i - 1, p)$ to (i, q) with label a and weight 1 ($=$ taking letter $w[i]$)

$O(|w|m)$ Upper Bound

Theorem

Given $w \in \Sigma^*$ and εNFA $A = (Q, \Sigma, q_0, q_f, \delta)$ of size m, the min- and max-variant of the $\preceq_{\text{sub}}$-matching problem can be solved in time $O(|w|m)$.

- build $\Sigma \cup \{\varepsilon\}$-labelled directed product graph $G_{A,w}$ of size $O(|w|m)$ with edge weights of 0 or 1, source s, and sink t :
  - vertices $\{0, 1, \ldots, |w|\} \times Q$, $s = (0, q_0)$ and $t = (|w|, q_f)$
  - transition (p, ε, q) and $i \in \{0, \ldots, |w|\}$: add edge from (i, p) to (i, q) with label ε and weight 0
  - transition (p, a, q) and $i \in \{1, \ldots, |w|\}$ with $w[i] = a$: add edge from $(i - 1, p)$ to (i, q) with label a and weight 1 (= taking letter $w[i]$)
  - state $q \in Q$ and $i \in \{1, \ldots, |w|\}$: add edge from $(i - 1, q)$ to (i, q) with label ε and weight 0 (= ignoring letter $w[i]$)

Conditional Lower Bound for the Min-Variant

Theorem

If the min-variant of the $\preceq_{\mathsf{sub}}$-matching problem can be solved in time $O(|w| + m)$, then we can decide whether a given dense graph G has a triangle in time $O(|G|)$.

Any truly subcubic (in the number of nodes) combinatorial algorithm for triangle detection yields a truly subcubic combinatorial algorithm for Boolean matrix multiplication, which is considered unlikely [Williams, Williams '18].

Conditional Lower Bound for the Min-Variant

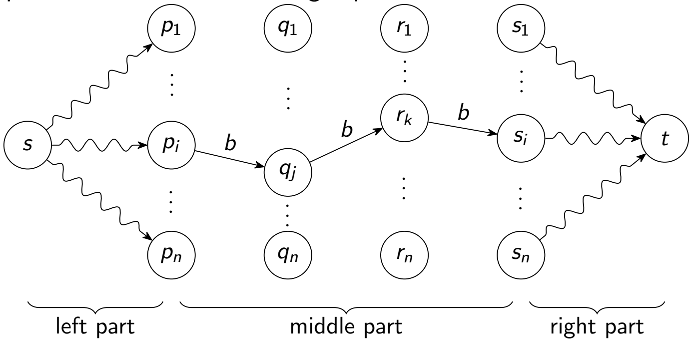
- transform dense graph $G = (\{v_1, \ldots, v_n\}, E)$ into NFA M_G:

Conditional Lower Bound for the Min-Variant

- transform dense graph $G = (\{v_1, \ldots, v_n\}, E)$ into NFA M_G:
  - states $\{p_i, q_i, r_i, s_i \mid 1 \leq i \leq n\}$, source s, target t
  - p_i: i-th 'entry point', s_i: i-th 'exit point'

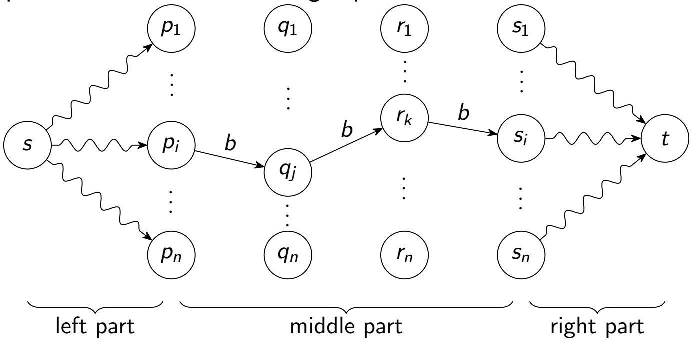
Conditional Lower Bound for the Min-Variant

- transform dense graph $G = (\{v_1, \ldots, v_n\}, E)$ into NFA M_G:
  - states $\{p_i, q_i, r_i, s_i \mid 1 \leq i \leq n\}$, source s, target t
  - p_i: i-th 'entry point', s_i: i-th 'exit point'
  - split into left, middle, and right part:



Conditional Lower Bound for the Min-Variant

- transform dense graph $G = (\{v_1, \ldots, v_n\}, E)$ into NFA M_G:
  - states $\{p_i, q_i, r_i, s_i \mid 1 \leq i \leq n\}$, source s, target t
  - p_i: i-th 'entry point', s_i: i-th 'exit point'
  - split into left, middle, and right part:



  - middle part: edges $(p_i, b, q_j), (q_i, b, r_j), (r_i, b, s_j)$ for every $\{v_i, v_j\} \in E$
  - G has a triangle containing v_i $\iff$ there is a bbb-labelled path from the i-th entry point to the i-th exit point

Conditional Lower Bound for the Min-Variant

- transform dense graph $G = (\{v_1, \dots, v_n\}, E)$ into NFA M_G:
  - states $\{p_i, q_i, r_i, s_i \mid 1 \leq i \leq n\}$, source s, target t
  - p_i: i-th 'entry point', s_i: i-th 'exit point'
  - middle part: edges $(p_i, b, q_j), (q_i, b, r_j), (r_i, b, s_j)$ for every $\{v_i, v_j\} \in E$
  - G has a triangle containing v_i $\iff$ there is a bbb-labelled path from the i-th entry point to the i-th exit point

Conditional Lower Bound for the Min-Variant

- transform dense graph $G = (\{v_1, \ldots, v_n\}, E)$ into NFA M_G:
  - states $\{p_i, q_i, r_i, s_i \mid 1 \leq i \leq n\}$, source s, target t
  - p_i: i-th 'entry point', s_i: i-th 'exit point'
  - middle part: edges $(p_i, b, q_j), (q_i, b, r_j), (r_i, b, s_j)$ for every $\{v_i, v_j\} \in E$
  - G has a triangle containing $v_i \iff$ there is a bbb-labelled path from the i-th entry point to the i-th exit point
  - left part: length-$(2(n+1) - i)$ path from s to p_i, where i edges are labelled with a (rest labelled with b)
  - right part: length-$(n+1+i)$ path from s_i to t, where $n - i$ edges are labelled with a (rest labelled with b)

Conditional Lower Bound for the Min-Variant

- transform dense graph $G = (\{v_1, \ldots, v_n\}, E)$ into NFA M_G:
  - states $\{p_i, q_i, r_i, s_i \mid 1 \leq i \leq n\}$, source s, target t
  - p_i: i-th 'entry point', s_i: i-th 'exit point'
  - middle part: edges $(p_i, b, q_j), (q_i, b, r_j), (r_i, b, s_j)$ for every $\{v_i, v_j\} \in E$
  - G has a triangle containing $v_i \iff$ there is a bbb-labelled path from the i-th entry point to the i-th exit point
  - left part: length-$(2(n+1) - i)$ path from s to p_i, where i edges are labelled with a (rest labelled with b)
  - right part: length-$(n + 1 + i)$ path from s_i to t, where $n - i$ edges are labelled with a (rest labelled with b)
- every string accepted by M_G must go through exactly one entry point and exactly one exit point
- an accepted string going through the i-th entry- and j-th exit point
  1. has length $3(n+1) + (j - i) + 3$
  2. has $n - (j - i)$ occurrences of a

Conditional Lower Bound for the Min-Variant

- an accepted string going through the i-th entry- and j-th exit point
  1. has length $3(n+1)+(j-i)+3$
  2. has $n-(j-i)$ occurrences of a

Conditional Lower Bound for the Min-Variant

- an accepted string going through the i-th entry- and j-th exit point
  1. has length $3(n+1)+(j-i)+3$
  2. has $n-(j-i)$ occurrences of a
- consider string $w = b^{2(n+1)}bbb(ab^{2(n+1)}bbb)^n$
- query: Is the minimal length of a subsequence of w accepted by M_G equal to $N := 3(n+1)+3$?

Conditional Lower Bound for the Min-Variant

- an accepted string going through the i-th entry- and j-th exit point
  1. has length $3(n+1)+(j-i)+3$
  2. has $n-(j-i)$ occurrences of a
- consider string $w = b^{2(n+1)}bbb(ab^{2(n+1)}bbb)^n$
- query: Is the minimal length of a subsequence of w accepted by M_G equal to $N := 3(n+1)+3$?
- N is a lower bound on the length of accepted subsequences:
  - let $u \preceq_{\text{sub}} w$ be accepted using the i-th entry and j-th exit point
  - u has at most n occurrences of a
  - using (2): $j \geq i$, thus $|u| = 3(n+1)+(j-i)+3 \geq N$

Conditional Lower Bound for the Min-Variant

- an accepted string going through the i-th entry- and j-th exit point
  1. has length $3(n+1) + (j - i) + 3$
  2. has $n - (j - i)$ occurrences of a
- consider string $w = b^{2(n+1)} bbb(ab^{2(n+1)} bbb)^n$
- query: Is the minimal length of a subsequence of w accepted by M_G equal to $N := 3(n+1) + 3$?
- N is a lower bound on the length of accepted subsequences:
  - let $u \preceq_{\text{sub}} w$ be accepted using the i-th entry and j-th exit point
  - u has at most n occurrences of a
  - using (2): $j \geq i$, thus $|u| = 3(n+1) + (j - i) + 3 \geq N$
- length-N subsequence of w can only be accepted using the same entry- and exit point i, which is only possible if G has a triangle containing v_i

Conditional Lower Bound for the Min-Variant

- an accepted string going through the i-th entry- and j-th exit point
  1. has length $3(n+1)+(j-i)+3$
  2. has $n-(j-i)$ occurrences of a
- consider string $w = b^{2(n+1)}bbb(ab^{2(n+1)}bbb)^n$
- query: Is the minimal length of a subsequence of w accepted by M_G equal to $N := 3(n+1)+3$?
- N is a lower bound on the length of accepted subsequences:
  - let $u \preceq_{\text{sub}} w$ be accepted using the i-th entry and j-th exit point
  - u has at most n occurrences of a
  - using (2): $j \geq i$, thus $|u| = 3(n+1)+(j-i)+3 \geq N$
- length-N subsequence of w can only be accepted using the same entry- and exit point i, which is only possible if G has a triangle containing v_i
- $|w| + |M_G| = O(n^2 + |G|) = O(|G|)$, since $|G| = \Omega(n^2)$: if we can not decide whether G has a triangle in $O(|G|)$ time, then we cannot solve the min-variant in linear time

Conditional Lower Bound for the Max-Variant

Theorem

If the max-variant of the $\preceq_{\text{sub}}$-matching problem can be solved in time $O((|w|m)^{1-\epsilon})$ for some $\epsilon > 0$, then the Strong Exponential Time Hypothesis (SETH) fails.

Conditional Lower Bound for the Max-Variant

Theorem

If the max-variant of the $\preceq_{\text{sub}}$-matching problem can be solved in time $O((|w|m)^{1-\epsilon})$ for some $\epsilon > 0$, then the Strong Exponential Time Hypothesis (SETH) fails.

- [Abboud, Backurs, Williams '15]: If we can compute the length of the longest common subsequence of two strings u and v in time $O((|u||v|)^{1-\epsilon})$ for some $\epsilon > 0$, then SETH fails.

Conditional Lower Bound for the Max-Variant

Theorem

If the max-variant of the $\preceq_{\text{sub}}$-matching problem can be solved in time $O((|w|m)^{1-\epsilon})$ for some $\epsilon > 0$, then the Strong Exponential Time Hypothesis (SETH) fails.

- [Abboud, Backurs, Williams '15]: If we can compute the length of the longest common subsequence of two strings u and v in time $O((|u||v|)^{1-\epsilon})$ for some $\epsilon > 0$, then SETH fails.
- construct εNFA $A_{u,sub}$ accepting exactly the subsequences of $u \in \Sigma^*$ (note: $|A_{u,sub}| = |u|$)

Conditional Lower Bound for the Max-Variant

Theorem

If the max-variant of the $\preceq_{\text{sub}}$-matching problem can be solved in time $O((|w|m)^{1-\epsilon})$ for some $\epsilon > 0$, then the Strong Exponential Time Hypothesis (SETH) fails.

- [Abboud, Backurs, Williams '15]: If we can compute the length of the longest common subsequence of two strings u and v in time $O((|u||v|)^{1-\epsilon})$ for some $\epsilon > 0$, then SETH fails.
- construct εNFA $A_{u,sub}$ accepting exactly the subsequences of $u \in \Sigma^*$ (note: $|A_{u,sub}| = |u|$)
- solving the max-variant of the $\preceq_{\text{sub}}$-matching problem for string v and εNFA $A_{u,sub}$ amounts to computing the longest common subsequence of u and v $\Longrightarrow$ SETH-conditional lower bound carries over

Overview

1. Preliminaries
2. The $\preceq_{\mathsf{sub}}$-Matching Problem
3. The Min- and Max-Variant of the $\preceq_{\mathsf{sub}}$-Matching Problem
4. The Universal $\preceq_{\mathsf{sub}}$-Matching Problem

coNP-Completeness

Theorem

Given string $w \in \Sigma^*$ and εNFA A, deciding whether $\Lambda_{\preceq_{\text{sub}}}(w) \subseteq L(A)$ is coNP-complete.

- $\in$ coNP: guess subsequence $u \preceq_{\text{sub}} w$, check if $u \notin L(A)$

coNP-Completeness

Theorem

Given string $w \in \Sigma^*$ and εNFA A, deciding whether $\Lambda_{\preceq_{\text{sub}}}(w) \subseteq L(A)$ is coNP-complete.

- $\in$ coNP: guess subsequence $u \preceq_{\text{sub}} w$, check if $u \notin L(A)$
- coNP-hard:
  - consider 3CNF formula F over n variables
  - build εNFA A_F that accepts all $\{0,1\}$-strings of lengths $k \in \{0, 1, \ldots, n-1, n+1, \ldots, 2n\}$ and all $\{0,1\}$-strings of length n that represent non-satisfying assignments of F

coNP-Completeness

Theorem

Given string $w \in \Sigma^*$ and εNFA A, deciding whether $\Lambda_{\preceq_{\text{sub}}}(w) \subseteq L(A)$ is coNP-complete.

- $\in$ coNP: guess subsequence $u \preceq_{\text{sub}} w$, check if $u \notin L(A)$
- coNP-hard:
  - consider 3CNF formula F over n variables
  - build εNFA A_F that accepts all $\{0,1\}$-strings of lengths $k \in \{0, 1, \ldots, n-1, n+1, \ldots, 2n\}$ and all $\{0,1\}$-strings of length n that represent non-satisfying assignments of F
  - $\Lambda_{\preceq_{\text{sub}}}((01)^n) \subseteq L(A_F) \iff F$ is not satisfiable

Results

①	in	pre	ext/lext	sub	sup
$\preceq$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\| + m)$	$O(\|w\| + m)$
min	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$
max	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$	$O(\|w\|m)$
$\forall$	$O(\|w\|^2 m)$	$O(\|w\|m)$	PSPACE	coNP	PSPACE

②	in/pre	ext/lext	sub	sup
$\preceq$	no $O((\|w\|m)^{1-\epsilon})$	no $O((\|w\|m)^{1-\epsilon})$	—	—
min	no $O((\|w\|m)^{1-\epsilon})$	no $O((\|w\|m)^{1-\epsilon})$	no $O(\|w\| + m)$	no $O((\|w\|m)^{1-\epsilon})$
max	no $O((\|w\|m)^{1-\epsilon})$	no $O((\|w\|m)^{1-\epsilon})$	no $O((\|w\|m)^{1-\epsilon})$	no $O(\|w\| + m)$
$\forall$	no $O((\|w\|m)^{1-\epsilon})$	PSPACE-hard	coNP-hard	PSPACE-hard

Upper bounds ① and (conditional) lower bounds ② for the different problem variants; note that m is the size of the εNFA A

Thank you for your attention!