A Quantum Query Complexity Trichotomy for Regular Languages

Scott Aaronson
UT Austin

Daniel Grier
MIT

Luke Schaeffer
MIT

Query complexity - Introduction

Query complexity of language $L \subseteq \Sigma^*$

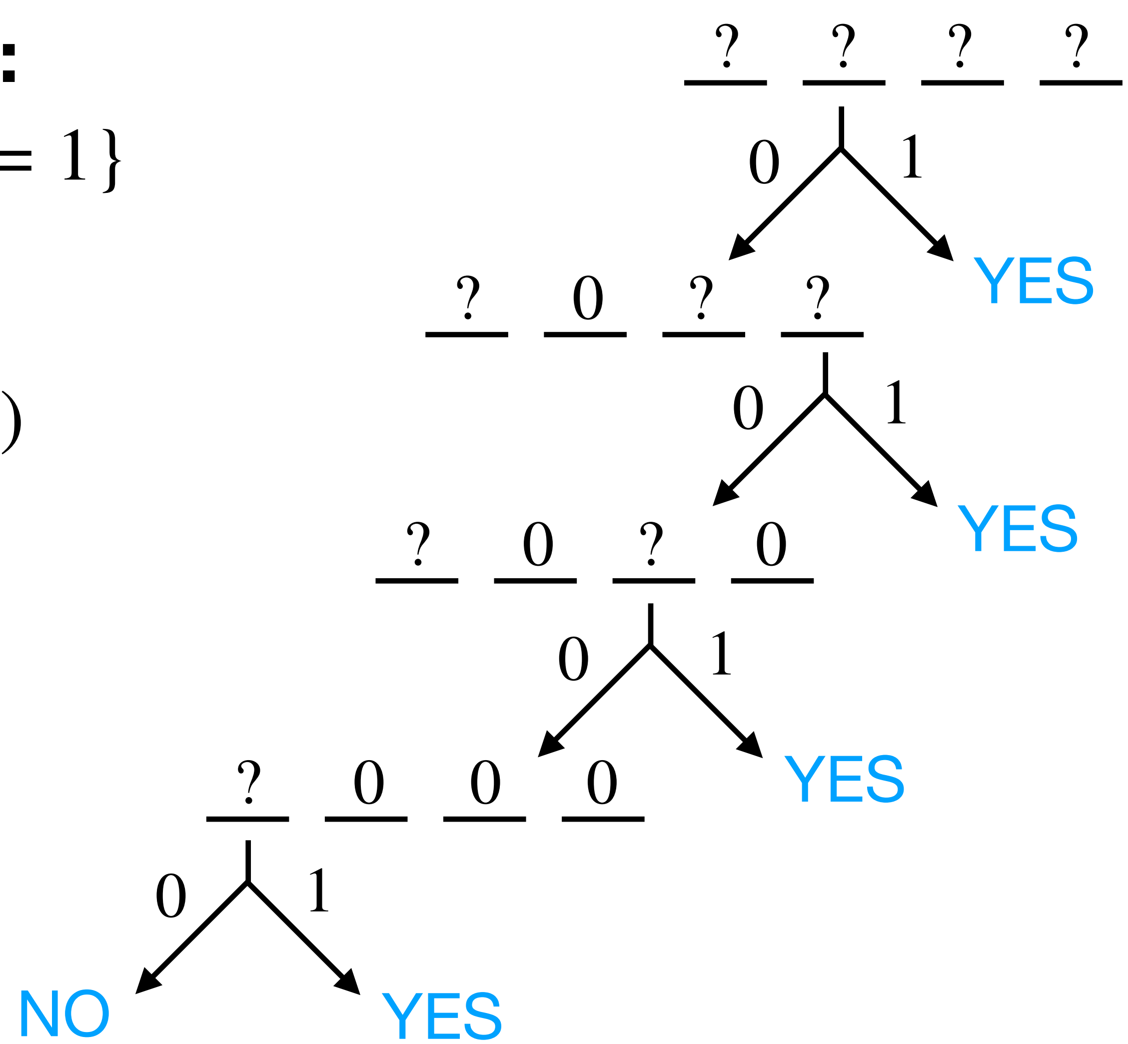
Input $x \in \Sigma^n$ *initially hidden. The query complexity of* L *is the number of input symbols revealed by the computation.*

Deterministic:

OR $= \{x : x_i = 1\}$

$D(\textbf{OR}) = n$

$R(\textbf{OR}) = \Theta(n)$



Query complexity - Introduction

Query complexity of language $L \subseteq \Sigma^*$

Input $x \in \Sigma^n$ *initially hidden. The query complexity of* L *is the number of input symbols revealed by the computation.*

Indexing oracle: $\sum \alpha_{i,b} |i\rangle |b\rangle \rightarrow \sum \alpha_{i,b} |i\rangle |b \oplus x_i\rangle$

Quantum: The number of calls to the indexing oracle to determine membership of an input with bounded error.

$Q(\textbf{OR}) = \Theta(\sqrt{n})$ ← Grover search

$Q(\textbf{PARITY}) = \Theta(n)$

Query complexity - Introduction

Query complexity of language $L \subseteq \Sigma^*$

Input $x \in \Sigma^n$ *initially hidden. The query complexity of* L *is the number of input symbols revealed by the computation.*

Why query complexity?

- Provable lower bounds
- Lower bounds can suggest efficient algorithms

Regular languages as regular expressions

Regular languages over a finite alphabet Σ

Basic sets:		*Combination rules:*	
Empty Set	$\varnothing$	Concatenation	AB
Empty string	$\{\varepsilon\}$	Union	$A \cup B$
Literal	$\{a \in \Sigma\}$	Kleene Star	A^*

$$A^* = \{a_1 \ldots a_k : k \geq 0,\ a_i \in A\}$$

Examples $\Sigma = \{0,1,2\}$

$$\Sigma = \{0\} \cup \{1\} \cup \{2\} = 0 \cup 1 \cup 2$$

$$\Sigma^* = \{\varepsilon, 0, 1, 2, 00, 01, 02, 10, \ \ldots\}$$

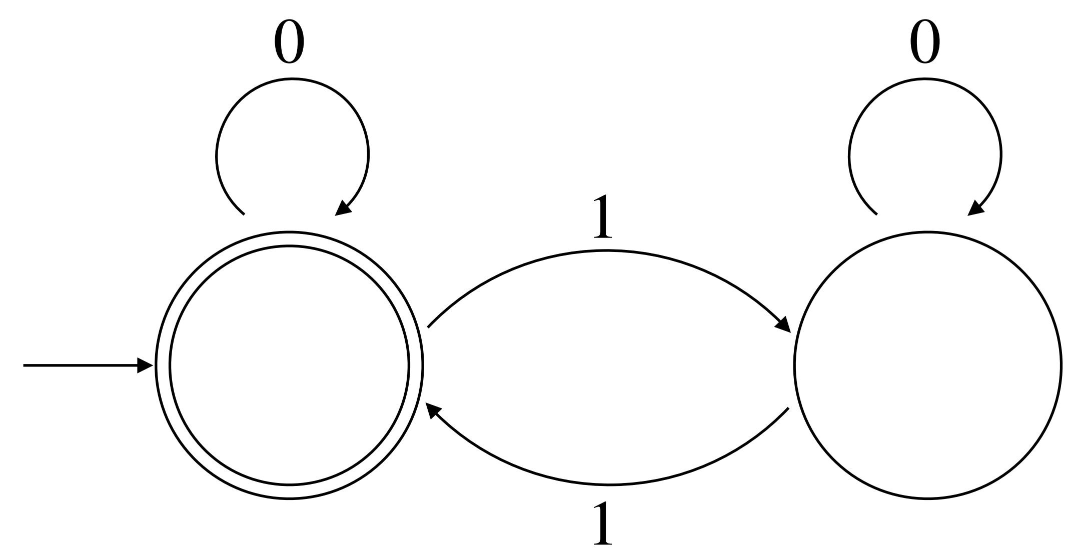
$$\textbf{OR} = 0^*1(0 \cup 1)^*$$

$$\textbf{AND-OR} = 2\textbf{OR}2\ldots2\textbf{OR}2 = 2(\textbf{OR}2)^*$$

$$\textbf{PARITY} = (0^*10^*1)^*0^*$$

Regular languages are nice

- Closed under many operations
  - Concatenation, Union, Kleene Star
  - Complement
  - Reversal
- Natural questions are decidable
  - "Is the language infinite?"
- Extremely robust definition
  - Regular expressions
  - Finite state automata
  - Recognized by finite monoids



Finite state automaton for **PARITY**

Quantum query complexity and regular languages

$1\Sigma^*1$

$\varnothing$

$O(1)$

AND-OR*

OR

$\tilde{\Theta}(\sqrt{n})$

PARITY

$\Omega(n)$

Quantum query trichotomy for regular languages

Trichotomy Theorem: Every regular language has quantum query complexity $\Theta(1)$, $\tilde{\Theta}(\sqrt{n})$, or $\Theta(n)$.

- Each query complexity corresponds to a class of regular expressions.
- All upper bounds come from explicit quantum algorithms.

Classes of regular expressions:

Trivial: Depend on $O(1)$ characters at beginning or end of string.

Star free: Regular expressions without Kleene star operation, but with the addition of the complement operation.

Regular: General regular expressions.

- trivial $\subsetneq$ star free $\subsetneq$ regular

Quantum query trichotomy for regular languages

Trichotomy Theorem: Every regular language has quantum query complexity $\Theta(1)$, $\tilde{\Theta}(\sqrt{n})$, or $\Theta(n)$.

Caveat:

Parity on even length strings: **PARITY** $\cap \, (\Sigma\Sigma)^*$

Query complexity oscillates between 0 and $\Theta(n)$.

Fix: *Redefine the standard notion of query complexity:*

Query complexity of strings of length **up to** n, rather than exactly n.

AND-OR is a star free language

Basic sets:

Empty Set $\varnothing$

Empty string $\{\varepsilon\}$

Literal $\{a \in \Sigma\}$

Combination rules:

Concatenation AB

Union $A \cup B$

Complement $\overline{A}$

$$\textbf{AND-OR} = 2\textbf{OR}2\ldots2\textbf{OR}2 = 2(\textbf{OR}2)^* = 2(0^*1(0 \cup 1)^*2)^*$$

Exercise…

$$\textbf{AND-OR} = \overline{\overline{\varnothing}2\overline{\overline{\varnothing}(1 \cup 2)\overline{\varnothing}}2\overline{\varnothing}} \cap 2\overline{\varnothing} \cap \overline{\varnothing}2$$

McNaughton's characterization of star free languages

Theorem [McNaughton]: A language is star free iff it is expressible in first-order logic with the less-than relation.

OR : $\exists i$ st. $x_i = 1$

AND-OR : $\forall i \forall j \exists k \ \ (i < j) \wedge (x_i = 2) \wedge (x_j = 2) \implies (i < k < j) \wedge (x_k = 1)$

Can extend to any constant number of alternating quantifiers

Consequence: Quantum algorithm for star free languages extends the Grover speed-up to a much larger class of string problems.

Application:

$\tilde{\Theta}(\sqrt{n})$ algorithm for dynamic constant-depth Boolean formulas

Outline for remainder of talk

1) Structure of trichotomy proof

    a) Upper bounds

    b) Lower bounds

2) $\tilde{O}(\sqrt{n})$ algorithm for star-free languages

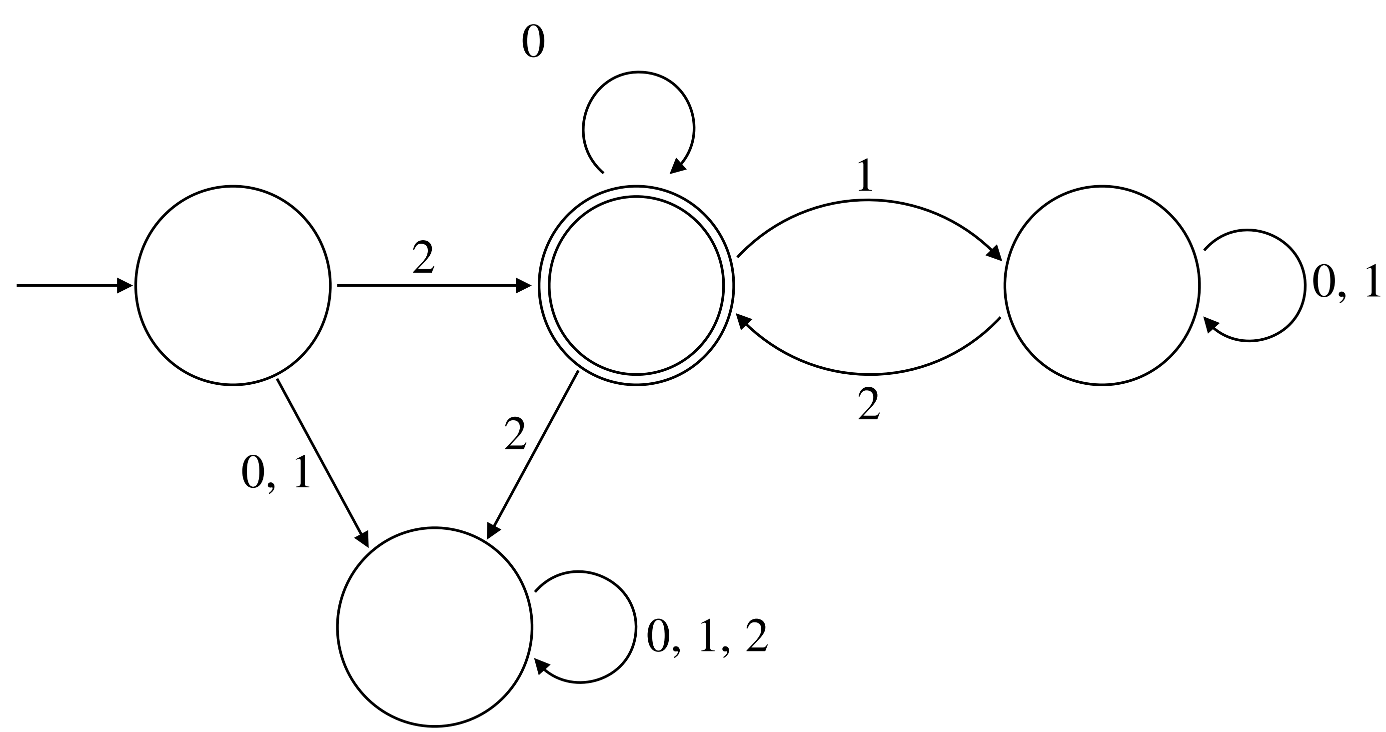
Trichotomy proof: Upper bounds

Algorithms:

Trivial: Only constantly-many symbols of input determine membership. Constant-size lookup table.

Star free: Challenging. More on this later.

Regular: Linear time deterministic algorithm from machine definition: "Read-only Turing machines"



Trichotomy proof: Lower bounds

Completing the classification requires:

$L \notin$ trivial $\implies Q(L) = \Omega(\sqrt{n})$

$L \notin$ star free $\implies Q(L) = \Omega(n)$

$\tilde{O}(\sqrt{n})$ **algorithm**

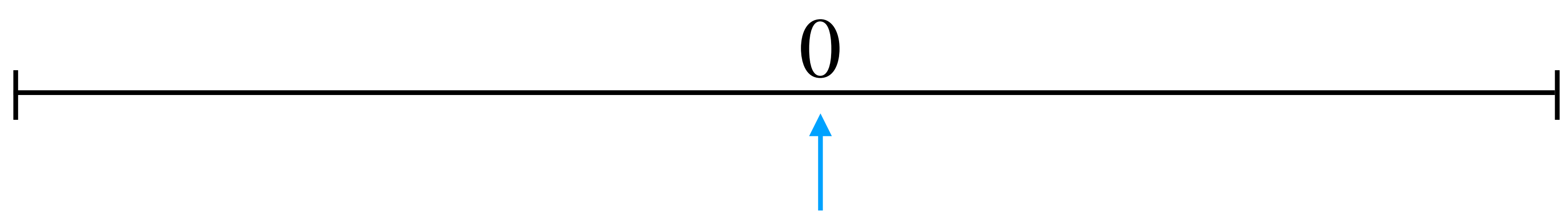
for star-free languages

$\tilde{\Theta}(\sqrt{n})$ quantum algorithm for AND-OR

Idea: Search for a substring $20{*}2$ *violating the* **OR**

First attempt: Grover search.

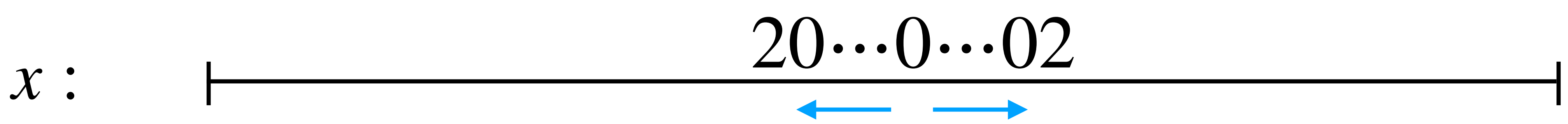
$x:$



$\tilde{\Theta}(\sqrt{n})$ quantum algorithm for AND-OR

Idea: Search for a substring $20{*}2$ *violating the* **OR**

First attempt: Grover search.



Grover iterations: $O(\sqrt{n})$

Work per iteration: $O(n)$

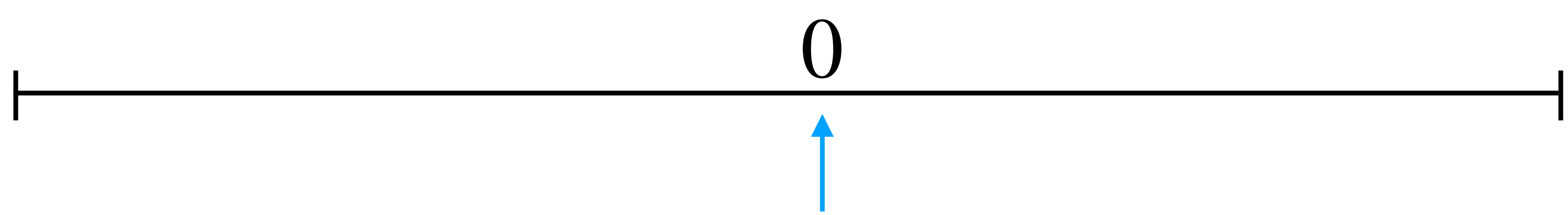
Total time: $O(n^{3/2})$

$\tilde{\Theta}(\sqrt{n})$ quantum algorithm for AND-OR

Idea: Search for a substring 20^*2 *violating the* **OR**

Second attempt: Grover within Grover.

$x:$

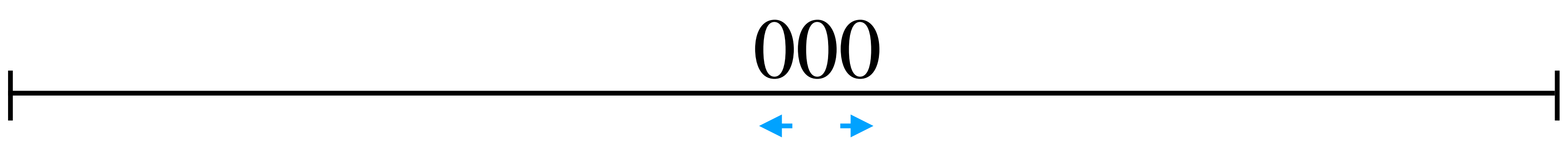


$\tilde{\Theta}(\sqrt{n})$ quantum algorithm for AND-OR

Idea: Search for a substring $20{*}2$ *violating the* **OR**

Second attempt: Grover within Grover.

x :



$\tilde{\Theta}(\sqrt{n})$ quantum algorithm for AND-OR

Idea: Search for a substring 20^*2 *violating the* **OR**

Second attempt: Grover within Grover.

$x:$ $0\cdots000\cdots0$

$\tilde{\Theta}(\sqrt{n})$ quantum algorithm for AND-OR

Idea: Search for a substring $20{*}2$ *violating the* **OR**

Second attempt: Grover within Grover.

$$20\cdots\cdots0\cdots000\cdots0\cdots\cdots02$$

$x:$

$\ell =$ length of match

Outer Grover: $O(\sqrt{n})$

Inner Grover: $O(\sqrt{1}) + O(\sqrt{2}) + O(\sqrt{4}) + \cdots + O(\sqrt{2^k}) = \tilde{O}(\sqrt{\ell})$

Total time: $\tilde{O}(n)$

$\tilde{\Theta}(\sqrt{n})$ quantum algorithm for AND-OR

Idea: Search for a substring 20^*2 *violating the* **OR**

Complete: Grover within Grover with multiple marked items.

$$x: \quad \underbrace{20\cdots\cdots0\cdots000\cdots0\cdots\cdots02}_{\ell}$$

Grover search with multiple marked items: When there are t marked items, Grover search only requires $O(\sqrt{n/t})$ iterations.

Full strategy:

Exponential search over length of the match: $\ell = 1, 2, 4, 8, \ \dots$

Grover search for index in the middle of the 20^*2 substring.

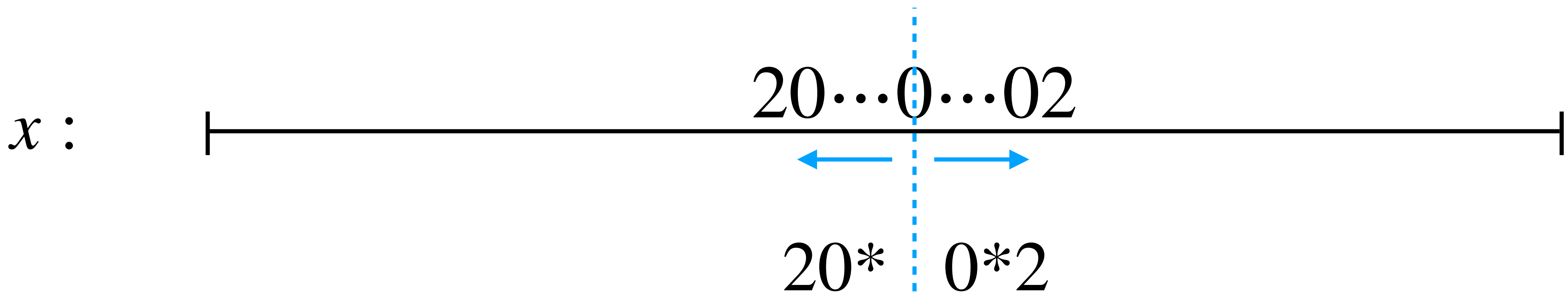
Grover/binary search to find 2 on each side at distance at most ℓ .

Analysis: $O(\sqrt{n/\ell}) \cdot \tilde{O}(\sqrt{\ell}) = \tilde{O}(\sqrt{n})$

Inner Grover

Outer Grover

Generalizing the AND-OR algorithm - Splitting



Splitting: Language $L \subseteq \Sigma^*$ splits as $\bigcup_{i=1}^{k} A_i B_i$ if

1) $L = \bigcup_{i=1}^{k} A_i B_i$ for some constant k.
2) $\forall x \in L$ and decompositions $x = uv, \exists i$ such that $u \in A_i$ and $v \in B_i$

Example: $20*2$ splits as $(20*2)\varepsilon \cup (20*)(0*2) \cup \varepsilon(20*2)$

Splitting implies infix search

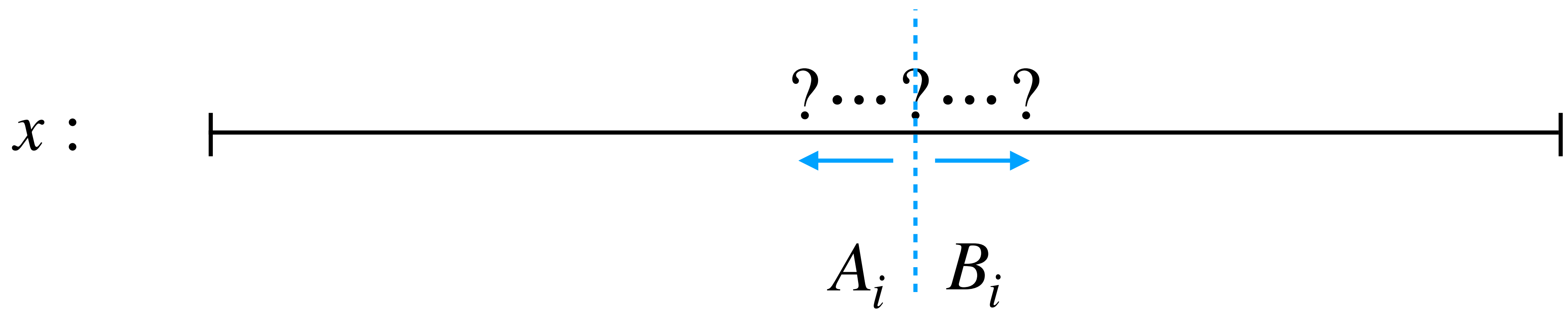
Infix Search: Let language L split as $\bigcup_{i=1}^{k} A_i B_i$ and suppose

$$Q(\Sigma^* A_i) = \tilde{O}(\sqrt{n}) \text{ for all } i$$

$$Q(B_i \Sigma^*) = \tilde{O}(\sqrt{n}) \text{ for all } i$$

Then $Q(\Sigma^* L \Sigma^*) = \tilde{O}(\sqrt{n})$.

Proof: Use same algorithm from **AND-OR.**



Schützenberger's theorem and star-free languages

Schützenberger's theorem: *(very informal)*
Given any star-free language, there is a hierarchy of component star-free languages. A language at one level of the hierarchy can be expressed as a combination of "simpler" languages from lower levels in the following way:

$$(\Sigma^* A_1 \cap A_2\Sigma^*) - \boxed{\Sigma^* A_3 \Sigma^*}$$

Remarkable fact: A_3 splits into simpler languages.

Plan: *Recursive algorithm:*
Find $\tilde{O}(\sqrt{n})$ algorithms for all component languages.

Not obvious: this will imply $\tilde{O}(\sqrt{n})$ algorithms for prefix and suffix problems: $\Sigma^* A_1, A_2\Sigma^*, \ldots$

Regular languages and monoids

Definition: A language L is recognized by a monoid M if there exists a homomorphism $\varphi : \Sigma^* \to M$ and a subset $S \subseteq M$ such that

$$L = \{w \in \Sigma^* : \varphi(w) \in S\}$$

A monoid is a semi-group with an identity element.

Monoid for **OR**: M : $\begin{array}{c|cc} & \mathbf{0} & \mathbf{1} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{array}$

$$\begin{array}{l} \varphi : \{0,1\}^* \to M \\ \hline \varphi(\varepsilon) = \varphi(0) = \mathbf{0} \\ \varphi(1) = \mathbf{1} \end{array}$$

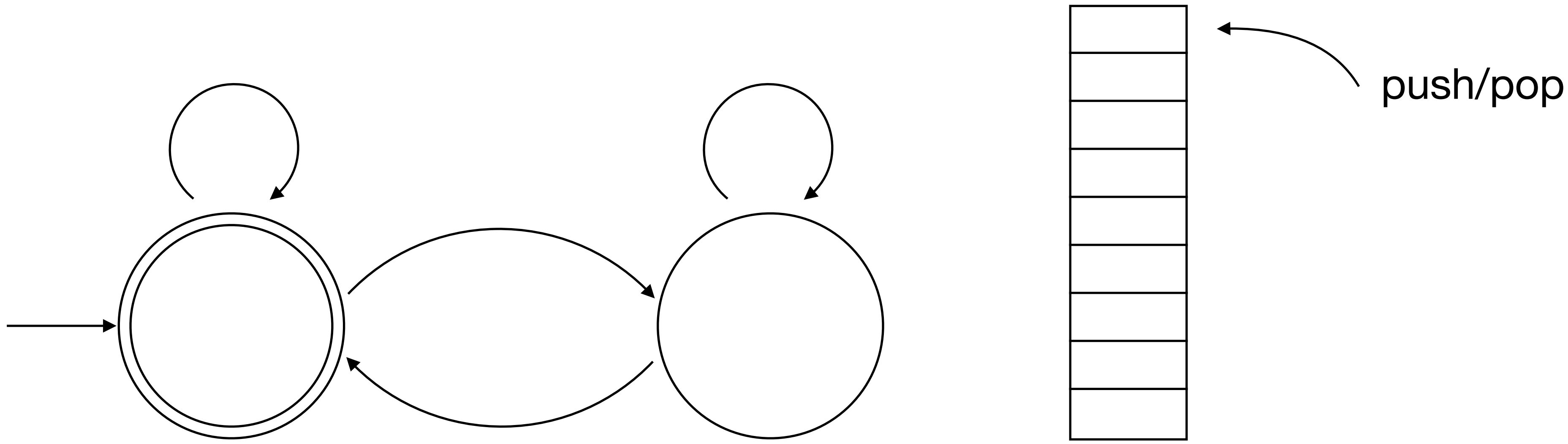
$S = \{\mathbf{1}\}$

Theorem (Schützenberger): A language is star free iff it is recognized by a *finite aperiodic* monoid.

Aperiodic: for all $m \in M$ there exists $n \geq 0$ such that $m^n = m^{n+1}$.

Proof sketch: $(\Sigma^* A_1 \cap A_2 \Sigma^*) - \Sigma^* A_3 \Sigma^*$

Context-free languages break trichotomy



Theorem: For every algebraic number $c \in [1/2,1]$, there exists a context-free language L such that $Q(L) = \Theta(n^c)$.

$O(n^{c+\epsilon})$ and $\Omega(n^{c-\epsilon})$ for all $\epsilon \geq 0$ for all limit computable $c \in [1/2,1]$.

Theorem: If L is context free and $Q(L) = \Theta(n^c)$, then c is limit computable.

Open Problems

1) Can you remove the log factors from the star-free algorithm?

2) Complete the classification for context-free languages. Can a CFL have query complexity $\tilde{\Theta}(n^c)$ for some $c \in (0,1/2)$?

3) Applications of star-free algorithm?