UMB CS622

An Intractable Problem

Wednesday, December 8, 2021

"I can't find an efficient algorithm, but neither can all these famous people."

Announcements

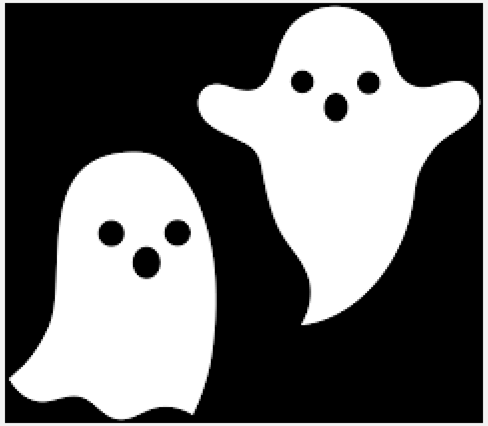
- ~~HW 10 in~~
  - ~~Due Tues 12/7 11:59pm EST~~

- HW 11 out
  - Due Tues 12/14 11:59pm EST

- Course evaluation at end of class today

Last Time: Nonexistent Algorithms

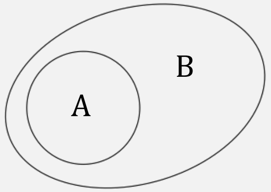
- It’s hard to prove that something doesn’t exist



- For algorithms/deciders, the best we can say is usually:
  - “There’s no known poly time algorithm that decides … e.g., *SAT*”

Last Time: Proving a Nonexistent Algorithm

1. Prove proper containment of two complexity classes,
   - e.g, if $\mathbf{A} \subset \mathbf{B}$



2. Prove completeness of a language in the larger class,
   - e.g, and if $L \in \mathbf{B}$ and L is $\mathbf{B}$-hard

3. Conclude that the language cannot be in the smaller class
   - e.g, then $L \notin \mathbf{A}$, i.e., L has no decider of some complexity!

Last Time: Hierarchy Theorems

THEOREM

Space hierarchy theorem For any space constructible function $f: \mathcal{N} \longrightarrow \mathcal{N}$, a language A exists that is decidable in $O(f(n))$ space but not in $o(f(n))$ space.

$$\text{NL} \subsetneq \text{PSPACE}$$

$$\text{PSPACE} \subsetneq \text{EXPSPACE}$$

THEOREM

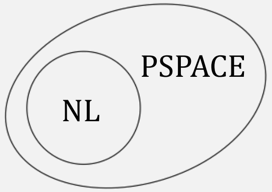
Time hierarchy theorem For any time constructible function $t: \mathcal{N} \longrightarrow \mathcal{N}$, a language A exists that is decidable in $O(t(n))$ time but not decidable in time $o(t(n)/\log t(n))$.

$$\text{P} \subsetneq \text{EXPTIME}$$

Last Time: A Nonexistent Algorithm

$TQBF = \{\langle \phi \rangle \mid \phi \text{ is a true fully quantified Boolean formula}\}$

1. <u>Prove proper containment</u> of two complexity classes,
   - e.g, **NL** ⊂ **PSPACE**

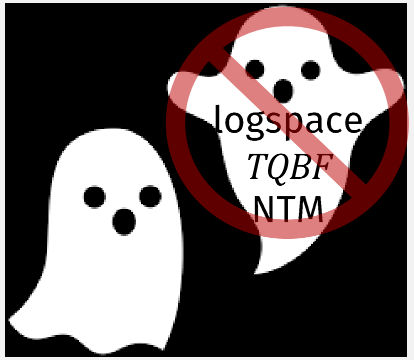


2. <u>Prove completeness</u> of a language in the larger class,
   - e.g, *TQBF* ∈ **PSPACE** and *TQBF* is **PSPACE**-hard

THEOREM

TQBF is PSPACE-complete.

3. <u>Conclude</u> that the language cannot be in the smaller class
   - e.g, *TQBF* ∉ **NL**,
   - i.e., *TQBF* <u>has no logspace NTM decider</u>!



What about a <u>nonexistent poly time algorithm</u>?

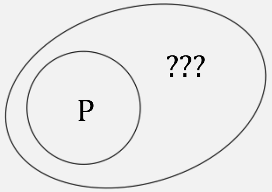
Thm: $EQ_{\text{REX}\uparrow}$ is Intractable! (not in $\mathbf{P}$!)

$EQ_{\text{REX}\uparrow} = \{\langle Q, R\rangle |\ Q$ and R are equivalent regular expressions with exponentiation$\}$

A Nonexistent Polynomial Time Algorithm

$EQ_{\mathsf{REX}\uparrow} = \{\langle Q, R\rangle \mid Q$ and R are equivalent regular expressions with exponentiation$\}$

1. Prove proper containment of two complexity classes,
   - e.g, $\mathbf{P} \subset$ **???**



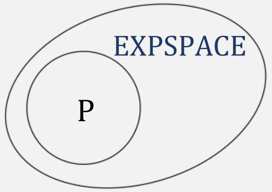
2. Prove completeness of a language in the larger class,
   - e.g, $EQ_{\mathsf{REX}\uparrow} \in$ **???** and $EQ_{\mathsf{REX}\uparrow}$ is **???**-hard

3. Conclude that the language cannot be in the smaller class
   - e.g, $EQ_{\mathsf{REX}\uparrow} \notin \mathbf{P}$,
   - i.e., $EQ_{\mathsf{REX}\uparrow}$ has no poly time decider!

A Nonexistent Polynomial Time Algorithm

$EQ_{\mathsf{REX}\uparrow} = \{\langle Q, R\rangle \mid Q$ and R are equivalent regular expressions with exponentiation$\}$

1. Prove proper containment of two complexity classes,
   - e.g, **P** ⊂ **EXPSPACE**



2. Prove completeness of a language in the larger class,
   - e.g, $EQ_{\mathsf{REX}\uparrow}$ ∈ **EXPSPACE** and $EQ_{\mathsf{REX}\uparrow}$ is **EXPSPACE** -hard

THEOREM

$EQ_{\mathsf{REX}\uparrow}$ is EXPSPACE-complete.

3. Conclude that the language cannot be in the smaller class
   - e.g, $EQ_{\mathsf{REX}\uparrow} \notin$ **P**,
   - i.e., $EQ_{\mathsf{REX}\uparrow}$ has no poly time decider!

P ⊂ **EXPSPACE**

- **P** ⊆ **PSPACE**, because
  - ⇒ A poly time algorithm uses at most poly space
  - ⇐ But a poly space algorithm can take more than poly time
    - Because space can be reused

- And Space Hierarchy Theorem says: **PSPACE** ⊂ **EXPSPACE**

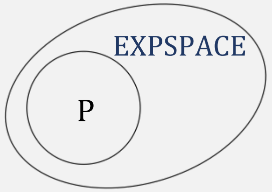
Space hierarchy theorem For any space constructible function $f : \mathcal{N} \longrightarrow \mathcal{N}$, a language A exists that is decidable in $O(f(n))$ space but not in $o(f(n))$ space.

- So **P** ⊆ **PSPACE** ⊂ **EXPSPACE**

A Nonexistent Polynomial Time Algorithm

$EQ_{\text{REX}\uparrow} = \{\langle Q, R\rangle \mid Q$ and R are equivalent regular expressions with exponentiation$\}$

☑ 1. Prove proper containment of two complexity classes,
- e.g, $\mathbf{P} \subset \mathbf{EXPSPACE}$



2. Prove completeness of a language in the larger class,
- e.g, $EQ_{\text{REX}\uparrow} \in \mathbf{EXPSPACE}$ and $EQ_{\text{REX}\uparrow}$ is **EXPSPACE** -hard

THEOREM

$EQ_{\text{REX}\uparrow}$ is EXPSPACE-complete.

3. Conclude that the language cannot be in the smaller class
- e.g, $EQ_{\text{REX}\uparrow} \notin \mathbf{P}$,
- i.e., $EQ_{\text{REX}\uparrow}$ has no poly time decider!

Flashback: Regular Expressions

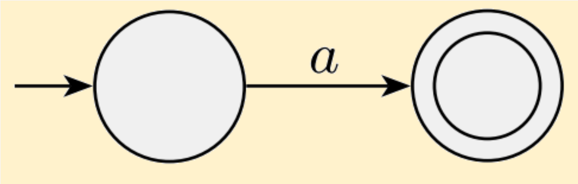
R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*), where R_1 is a regular expression.

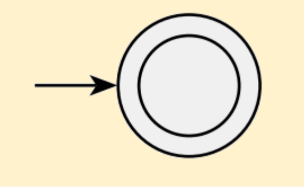
Flashback: RegExpr→NFA

R is a ***regular expression*** if R is

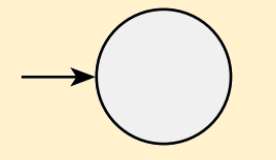
1. a for some a in the alphabet Σ,



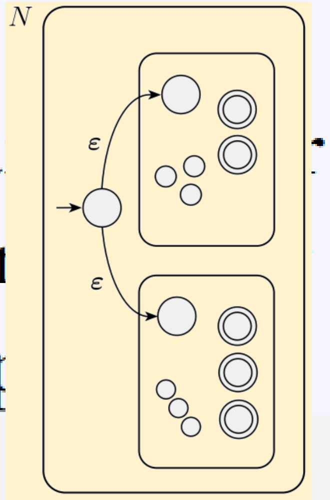
2. ε,

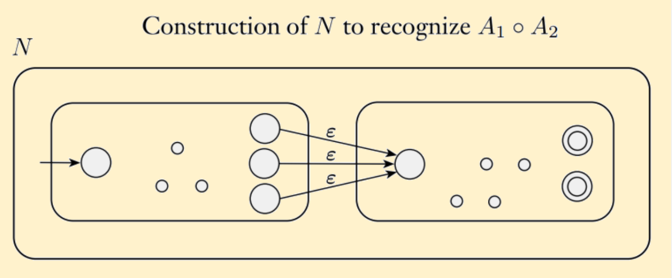


3. $\emptyset$,



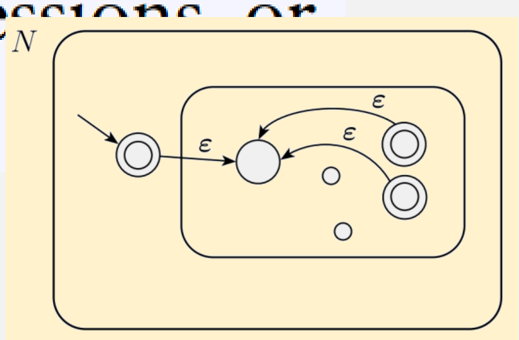
4. $(R_1 \cup R_2)$, where R_1 and R_2 a … r e





5. $(R_1 \circ R_2)$, where R_1 and R_2 ar … expressions, or

6. (R_1^*), where R_1 is a regular exp



RegExpr→NFA is in **PSPACE**

- **From** HW10, Problem # 2

EQ_{NFA} is in **PSPACE**

- Prove not $\overline{EQ_{\mathrm{NFA}}}$ is in **PSPACE**
  - From HW10, Problem # 3

- And prove **PSPACE** closed under complement
  - From HW10, Problem # 1

$\overline{EQ}_{\text{NFA}}$ is in **NPSPACE (= PSPACE)**

Flashback: Nondeterministic Space Usage

$$ALL_{\mathsf{NFA}} = \{\langle A \rangle | \ A \text{ is an NFA and } L(A) = \Sigma^*\}$$

Nondeterministic decider for $\overline{ALL_{\mathsf{NFA}}}$

$N =$ "On input $\langle M \rangle$, where M is an NFA:

1. Place a marker on the start state of the NFA.
2. Repeat 2^q times, where q is the number of states of M:
3. Nondeterministically select an input symbol and change the positions of the markers on M's states to simulate reading that symbol.
4. *Accept* if stages 2 and 3 reveal some string that M rejects; that is, if at some point none of the markers lie on accept states of M. Otherwise, *reject*."

Machine tracks "current" states of NFA:
q states = 2^q possible combinations
(so exponential time)

Each loop uses only $O(q)$ space!

Additionally, need a counter to count to 2^q: requires $\log(2^q) = q$ extra space

So the whole machine runs in (nondeterministic) linear $O(q)$ space!

$\overline{EQ}_{\text{NFA}}$ is in **NPSPACE (= PSPACE)**

N = "On input $\langle N_1, N_2 \rangle$, where N_1 and N_2 are NFAs:

1. Place a marker on each of the start states of N_1 and N_2.

2. Repeat $2^{q_1+q_2}$ times, where q_1 and q_2 are the numbers of states in N_1 and N_2:

3. Nondeterministically select an input symbol and change the positions of the markers on the states of N_1 and N_2 to simulate reading that symbol.

4. If at any point a marker was placed on an accept state of one of the finite automata and not on any accept state of the other finite automaton, *accept*. Otherwise, *reject*."

Track 2 sets of "current" states

Machine runs in:
- nondeterministic $O(\boldsymbol{q})$ space
- deterministic $O(\boldsymbol{q}^2)$ space

EQ_{REX} is in **PSPACE**

$EQ_{\text{REX}} = \{\langle Q, R\rangle |\ Q$ and R are equivalent regular expressions$\}$

From HW10, Problem # 4

1. Convert regular expressions to NFAs (**PSPACE**)

2. Check if NFAs are equivalent (**PSPACE**)

Regular Expressions + Exponentiation

Let $\uparrow$ be the ***exponentiation operation***.

- If R is a regular expression, then

$$R^k = R \uparrow k = \overbrace{R \circ R \circ \cdots \circ R}^{k}$$

  - I.e., exponentiation = concatenation k times

- So regular expressions with exponentiation …
  - … still equivalent to regular langs!

Thm: $EQ_{\mathsf{REX}\uparrow}$ is Intractable! (not in $\mathbf{P}$!)

$EQ_{\mathsf{REX}\uparrow} = \{\langle Q, R\rangle \mid Q$ and R are equivalent regular expressions with exponentiation$\}$

THEOREM

$EQ_{\mathsf{REX}\uparrow}$ is EXPSPACE-complete.

EXPSPACE-Completeness

DEFINITION

A language B is ***EXPSPACE-complete*** if

1. $B \in$ EXPSPACE, and

2. every A in EXPSPACE is polynomial time reducible to B.

THEOREM

$EQ_{\text{REX}\uparrow}$ is EXPSPACE-complete.

$EQ_{\mathsf{REX}\uparrow}$ is in **EXPSPACE**

$EQ_{\mathsf{REX}\uparrow} = \{\langle Q, R\rangle \mid Q \text{ and } R \text{ are equivalent regular expressions with exponentiation}\}$

Similar to EQ_{REX} decider from HW10, Problem #4

E = "On input $\langle R_1, R_2\rangle$, where R_1 and R_2 are regular expressions with exponentiation:

1. Convert R_1 and R_2 to equivalent regular expressions B_1 and B_2 that use repetition instead of exponentiation.
2. Convert B_1 and B_2 to equivalent NFAs N_1 and N_2, using the conversion procedure given in the proof of Lemma 1.55.
3. Use the deterministic version of algorithm N to determine whether N_1 and N_2 are equivalent."

Uses exponentially more space (→ step 1)

From HW10 (→ steps 2 and 3)

EXPSPACE-Completeness

DEFINITION

A language B is ***EXPSPACE-complete*** if

1. $B \in$ EXPSPACE, and

2. every A in EXPSPACE is polynomial time reducible to B.

THEOREM

$EQ_{\mathsf{REX}\uparrow}$ is EXPSPACE-complete.

$EQ_{\mathsf{REX}\uparrow}$ Is **EXPSPACE**-Hard

$$EQ_{\mathsf{REX}\uparrow} = \{\langle Q, R\rangle \mid Q \text{ and } R \text{ are equivalent regular expressions with exponentiation}\}$$

Flashback: Undecidability By Checking TM Configs

$ALL_{\text{CFG}} = \{\langle G \rangle \mid G \text{ is a CFG and } L(G) = \Sigma^*\}$

Proof, by contradiction

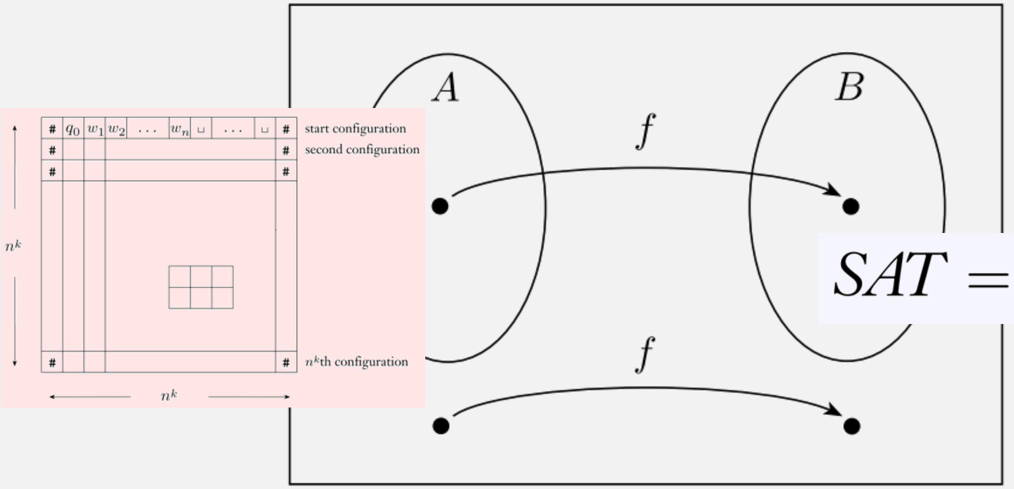
- Assume ALL_{CFG} has a decider R. Use it to create decider for A_{TM}:

On input <M, w>:

- Construct a PDA P that rejects sequences of M configs that accept w
- Convert P to a CFG G
- Give G to R:
  - If R accepts, then M has no accepting config sequences for w, so reject
  - If R rejects, then M has an accepting config sequence for w, so accept

Any machine that can validate TM config sequences could be used to prove undecidability?

Flashback: Reducing every **NP** language to **SAT**

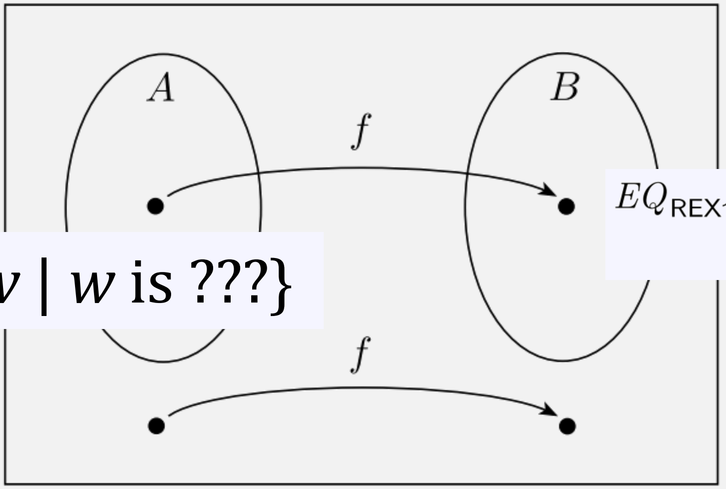


$SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$

$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$

We know **NP** languages have a poly time NTM M!
So reduce M accepting config sequences to a satisfiable formula!

Reducing every **EXPSPACE** lang to $EQ_{\mathsf{REX}\uparrow}$

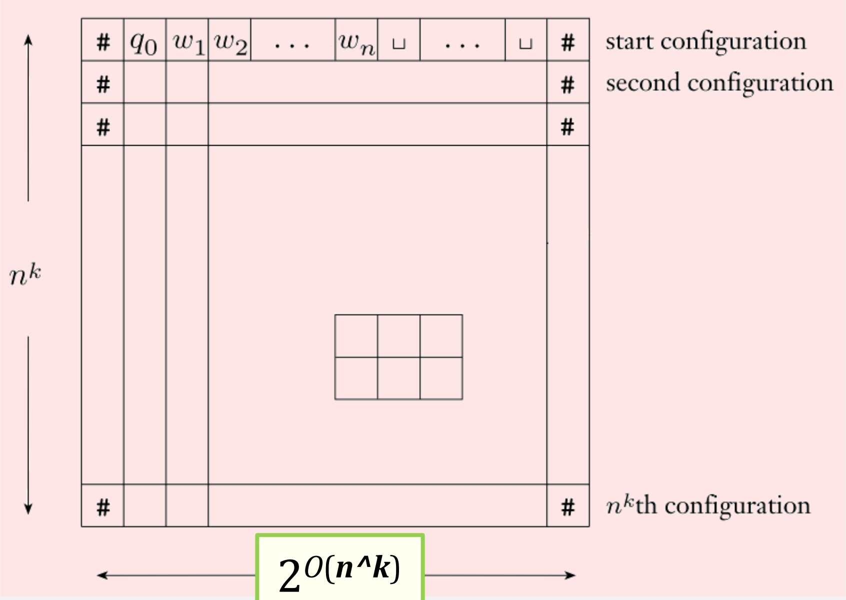


$EQ_{\mathsf{REX}\uparrow} = \{\langle Q, R\rangle \mid Q \text{ and } R \text{ are equivalent regular expressions with exponentiation}\}$

Some **EXPSPACE** lang = $\{w \mid w \text{ is ???}\}$

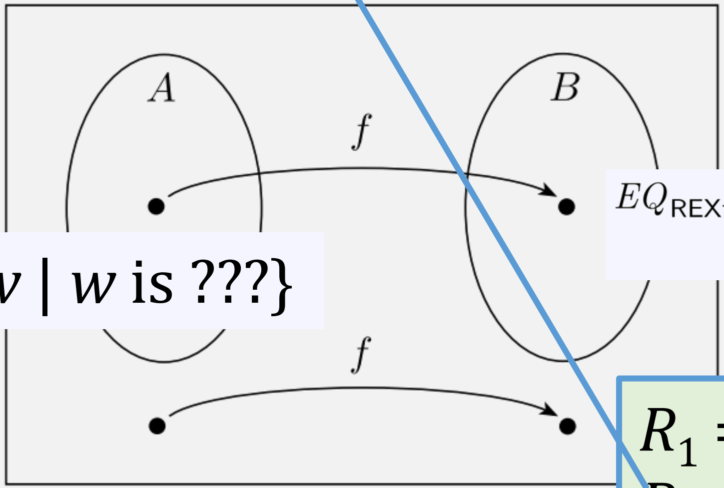
Need to reduce some w to <u>2 equivalent regular expressions</u>???

We know the language has an exp space decider!



Reducing every **EXPSPACE** lang to $EQ_{\text{REX}\uparrow}$

R_2 equals $R_{\text{bad-start}} \cup R_{\text{bad-window}} \cup R_{\text{bad-reject}}$



$EQ_{\text{REX}\uparrow} = \{\langle Q, R \rangle \mid Q$ and R are equivalent regular expressions with exponentiation$\}$

Some **EXPSPACE** lang = $\{w \mid w$ is ???$\}$

$R_1 = \Delta^*$, $\Delta = \Gamma \cup Q \cup \{\#\}$

R_2 = non-rejecting M config seqs for w

$M = (Q, \Sigma, \Gamma, \delta, q_{\text{accept}}, q_{\text{reject}})$

We know the language has an exp space decider!

⇒ If M accepts w,
there are no rejecting M config seqs for w so $R_1 = R_2$

⇐ If M rejects w,
there are rejecting M config seqs for w so $R_1 \neq R_2$

Rejecting Config Sequences

A rejecting sequence of M configs on w:

- <u>Starts</u> in start state q_0 with w on the tape
- Each <u>step must be valid</u> according to δ
- <u>Ends</u> in config with state q_{reject}

- R_2 generates config seqs that **<u>don't</u> satisfy (at least 1 of) these**

$$R_{\text{bad-start}} \cup R_{\text{bad-window}} \cup R_{\text{bad-reject}}$$

- <u>Important</u>:
  - R_2 must be polynomial in length to have poly time reduction!

$$R_{\text{bad-start}} = S_0 \cup S_1 \cup \cdots \cup S_n \cup S_b \cup S_{\#}$$

$R_{\text{bad-start}}$ = all strings not beginning with start config of M with w

$\Delta = \Gamma \cup Q \cup \{\#\}$

Δ_{-x} = all chars in Δ except for x

- $w = w_1, \ldots, w_n$ (length n)
- $S_0 = \Delta_{-q0}\,\Delta^*$ = all strings that don't start with q_0
- $S_i = \Delta^i\,\Delta_{-wi}\,\Delta^*$ = all strings whose i+1th char isn't w_i
  - These are all poly length (can be generated in poly time)
- S_b = all strings that don't have a blank in pos $n + 2$ to $2^{n\wedge k}$
  - Could be exponential in length …
  - … unless we use exponentiation!

Exponential exponent … takes $\log(2^{n\wedge k})$ space = n^k space

$$S_b = \Delta^{n+1}\,(\Delta \cup \varepsilon)^{2^{(n^k)}-n-2}\,\Delta_{-\sqcup}\,\Delta^*$$

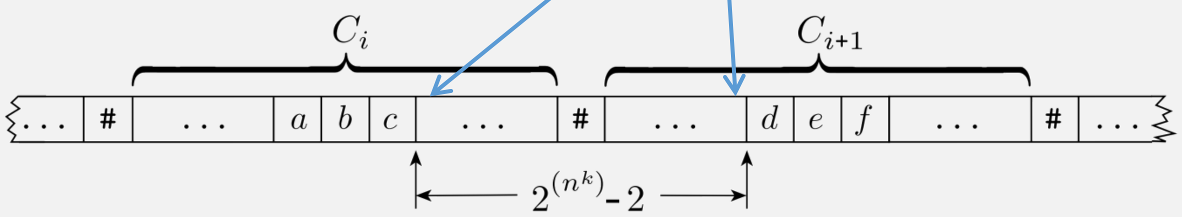
Bad Reject

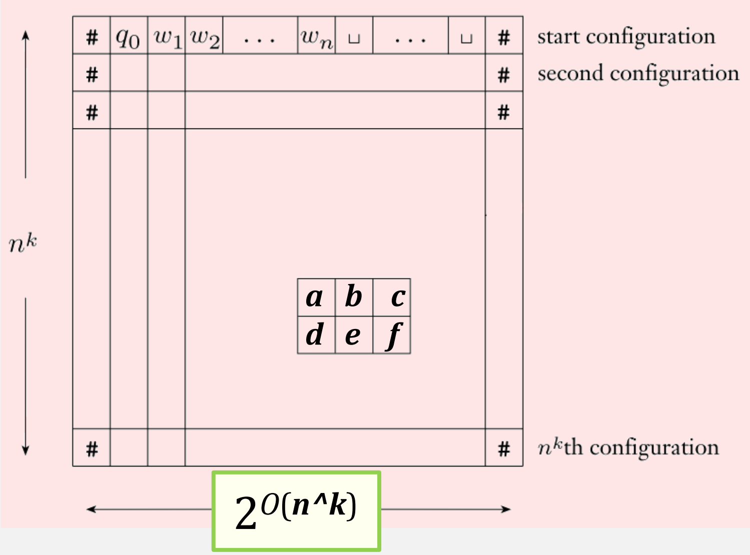
$$R_{\text{bad-reject}} = \Delta^*_{-q_{\text{reject}}}$$

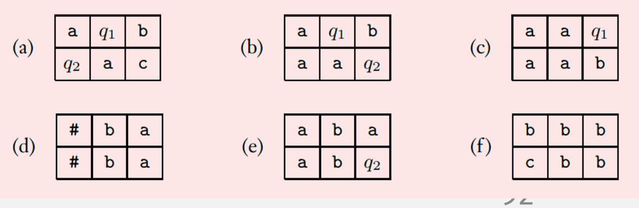
Bad Window

- bad(*abc*, *def*) means window *abc* → *def* not valid according to δ

$$R_{\text{bad-window}} = \bigcup_{\text{bad}(abc,def)} \Delta^* \, abc \, \Delta^{(2^{(n^k)}-2)} \, def \, \Delta^*$$







R_2 Total Length (Time)

- $R_{\text{bad-start}} = S_0 \cup S_1 \cup \cdots \cup S_n \cup S_b \cup S_\#$
- $O(n^k)$

Exponential exponent … takes $\log(2^{n^\wedge k})$ space = n^k space …
Can be generated in poly time

$$S_b = \Delta^{n+1} (\Delta \cup \varepsilon)^{2^{(n^k)}-n-2} \Delta_{-\sqcup} \Delta^*$$

- $R_{\text{bad-reject}} = \Delta^*_{-q_{\text{reject}}}$
- $O(1)$
- $R_{\text{bad-window}} = \bigcup_{\text{bad}(abc,def)} \Delta^* \, abc \, \Delta^{(2^{(n^k)}-2)} \, def \, \Delta^*$
- $O(n^k)$

Total Time: $O(n^k)$

EXPSPACE-Completeness

DEFINITION

A language B is ***EXPSPACE-complete*** if

☑ **1.** $B \in$ EXPSPACE, and

☑ **2.** every A in EXPSPACE is polynomial time reducible to B.

THEOREM

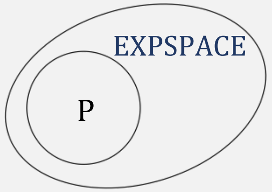
$EQ_{\mathsf{REX}\uparrow}$ is EXPSPACE-complete. ■

A Nonexistent Polynomial Time Algorithm

$EQ_{\mathsf{REX}\uparrow} = \{\langle Q, R\rangle \mid Q$ and R are equivalent regular expressions with exponentiation$\}$

☑ 1. Prove proper containment of two complexity classes,

- e.g, **P** ⊂ **EXPSPACE**



☑ 2. Prove completeness of a language in the larger class,

- e.g, $EQ_{\mathsf{REX}\uparrow}$ ∈ **EXPSPACE** and $EQ_{\mathsf{REX}\uparrow}$ is **EXPSPACE** -hard

THEOREM ....

$EQ_{\mathsf{REX}\uparrow}$ is EXPSPACE-complete.

☑ 3. Conclude that the language cannot be in the smaller class

- e.g, $EQ_{\mathsf{REX}\uparrow} \notin$ **P**,
- i.e., $EQ_{\mathsf{REX}\uparrow}$ has no poly time decider!

No Quiz 12/8

Fill out course evaluation