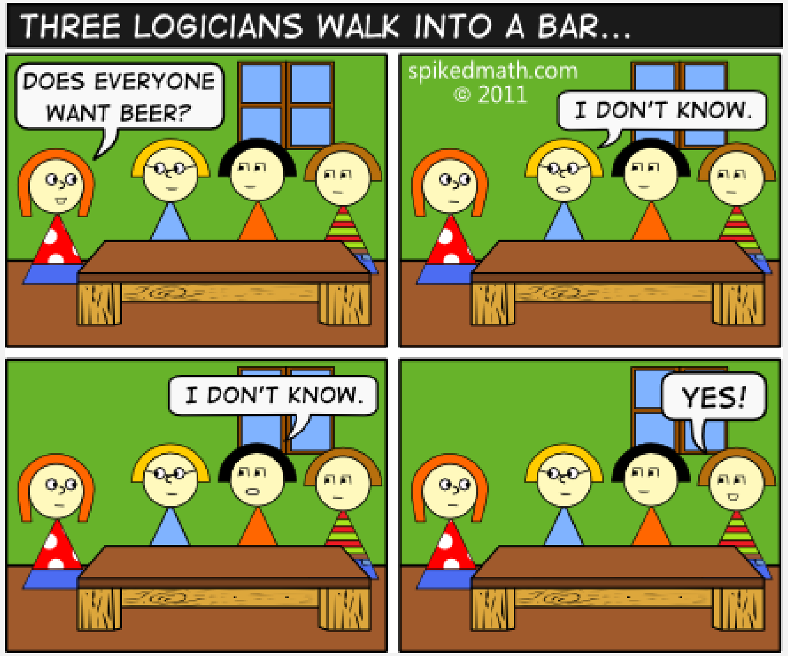
The Cook-Levin Theorem

(i.e., the first NP-Complete Problem)

Wednesday, December 2, 2020



HW10 questions?

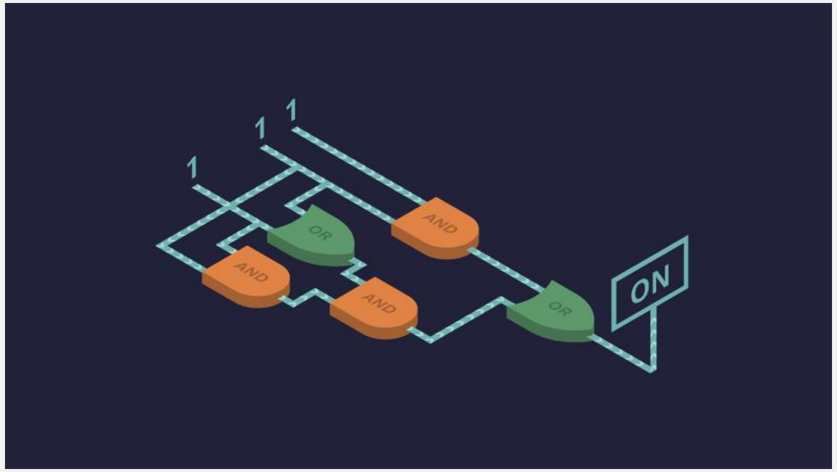
Announcements

- Chegg and other similar sites are now banned.

Today: The Cook-Levin Theorem

THEOREM 7.37

SAT is NP-complete



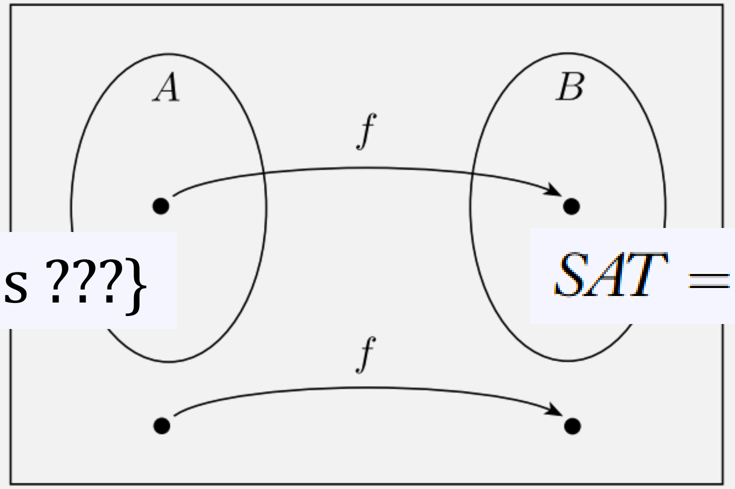
DEFINITION 7.34

A language B is ***NP-complete*** if it satisfies two conditions:

1. B is in NP, and
2. every A in NP is polynomial time reducible to B.

Hard part → 2. every A in NP

Reducing every **NP** language to **SAT**



Some **NP** lang = {w | w is ???}

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

How can we come up with reduction of some w to a Boolean formula if we don't know w?

How to prove a theorem about an entire <u>class</u> of languages?

- We work with what we know about the langs <u>in general</u>

THEOREM 1.45 ----------

- E.g, The class of regular languages is closed under the union operation.
  - **PROOF** uses the theorem that every reg lang has an NFA accepting it

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Proof is a <u>algorithm</u> for constructing a union-recognizing NFA from any two NFAs

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

THEOREM 4.7 ----------

- A_{CFG} is a decidable language. $A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$
  - Proof uses the fact that every CFG has a Chomsky Normal Form

What do we know about strings in **NP** langs?

- They are
  - Verified by a deterministic poly time verifier (NP definition)
  - Decided by a nondeterministic poly time decider (NTM) (Thm 7.20)

Let's use this one

Review: Non-deterministic TMs

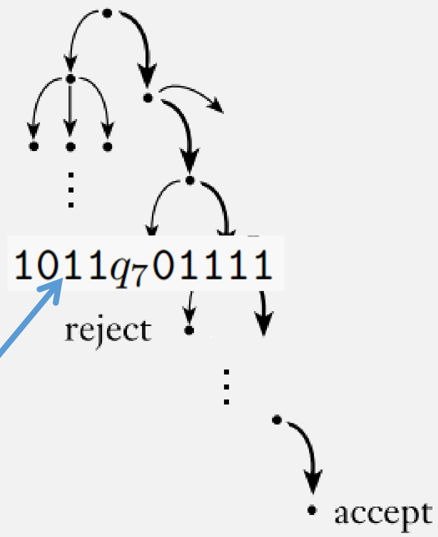
- Formally defined with states, transitions, alphabet …

Idea: We don't know the specific language or strings in the language, but …

… we know those strings must have an accepting sequence of configurations!

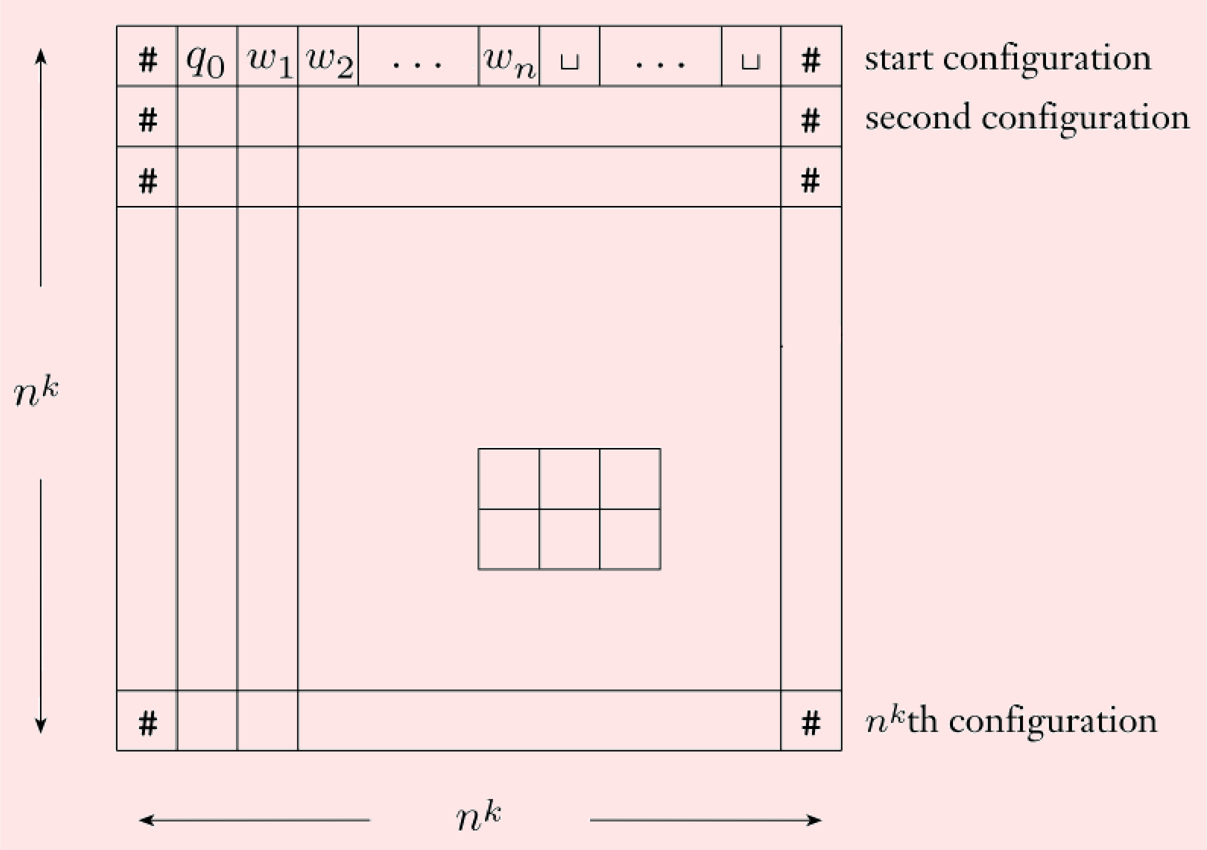
A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the ***blank symbol*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$ transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{accept} \in Q$ is the accept state, and
7. $q_{reject} \in Q$ is the reject state, where $q_{reject} \neq q_{accept}$.



- Computation can branch
- Each node in the tree represents a TM configuration
- Transitions specify valid configuration sequences

Accepting config sequence = Tableau



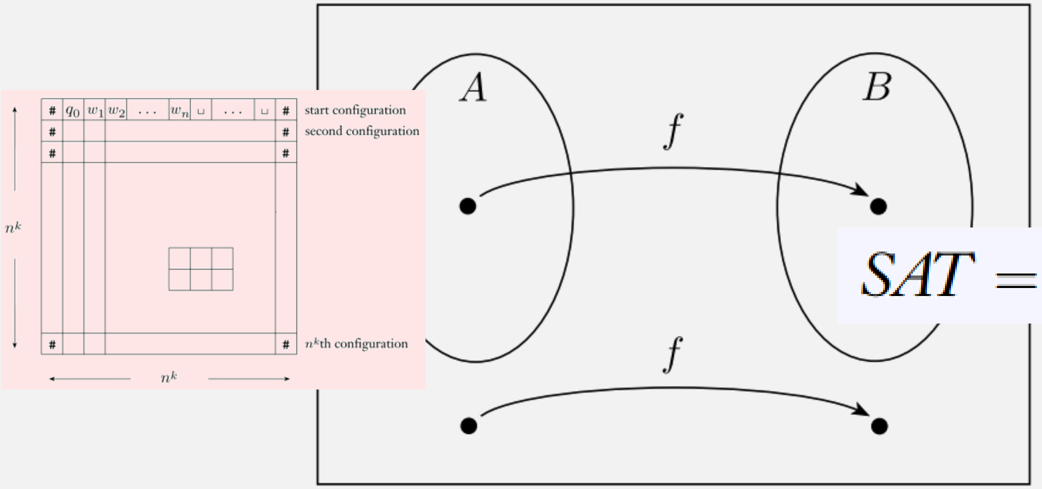
- $w = w_1 \dots w_n$
- To simplify proof, assume configs start/end with **#**
- Some config must be accepting config
- At most $\mathbf{n^k}$ configs
- Each config has length $\mathbf{n^k}$

<u>Theorem</u>: SAT is NP-complete

- <u>Proof idea</u>:
  - Give an algorithm that converts accepting tableaus to satisfiable formulas

- Thus every string in the **NP** lang will be mapped to a sat. formula
  - and vice versa

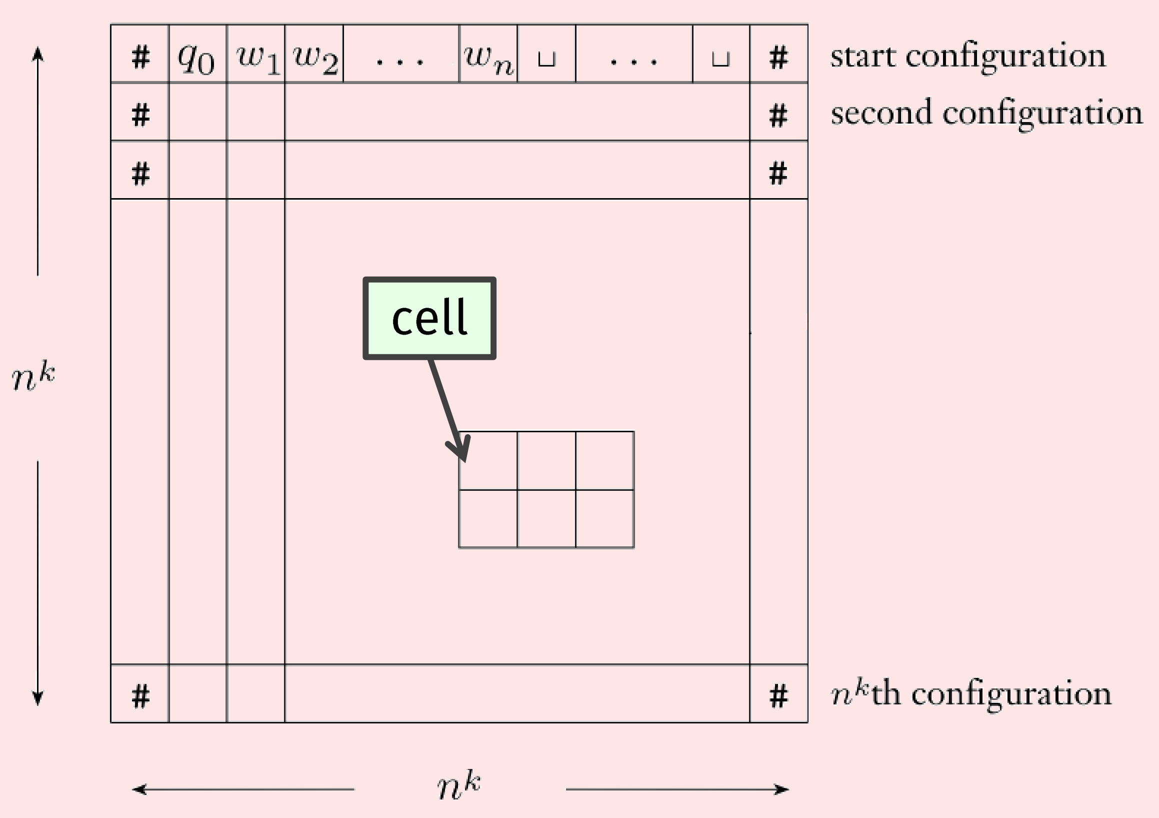
Resulting formulas will have <u>four</u> components:
$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$



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

Tableau Terminology

- A tableau <u>cell</u> has coordinate i,j
- A cell has <u>symbol</u>
  $s \in C = Q \cup \Gamma \cup \{\#\}$

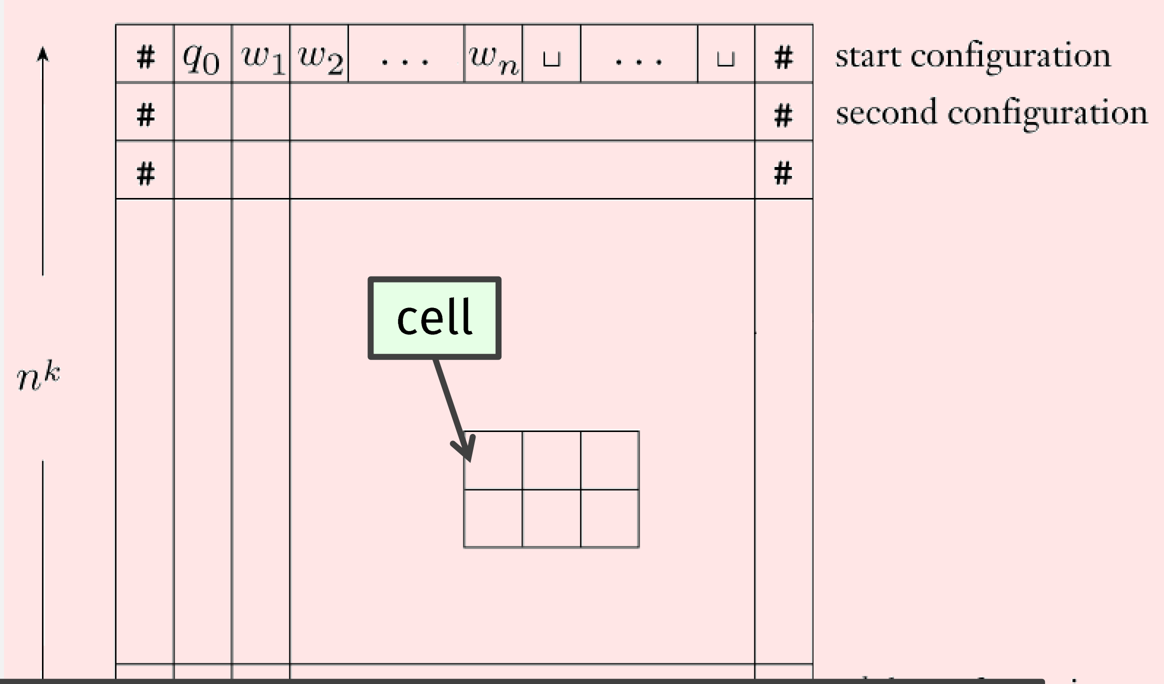


A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the ***blank symbol*** ⊔,
3. Γ is the tape alphabet, where ⊔ $\in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta: Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$ e transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{accept} \in Q$ is the accept state, and
7. $q_{reject} \in Q$ is the reject state, where $q_{reject} \neq q_{accept}$.

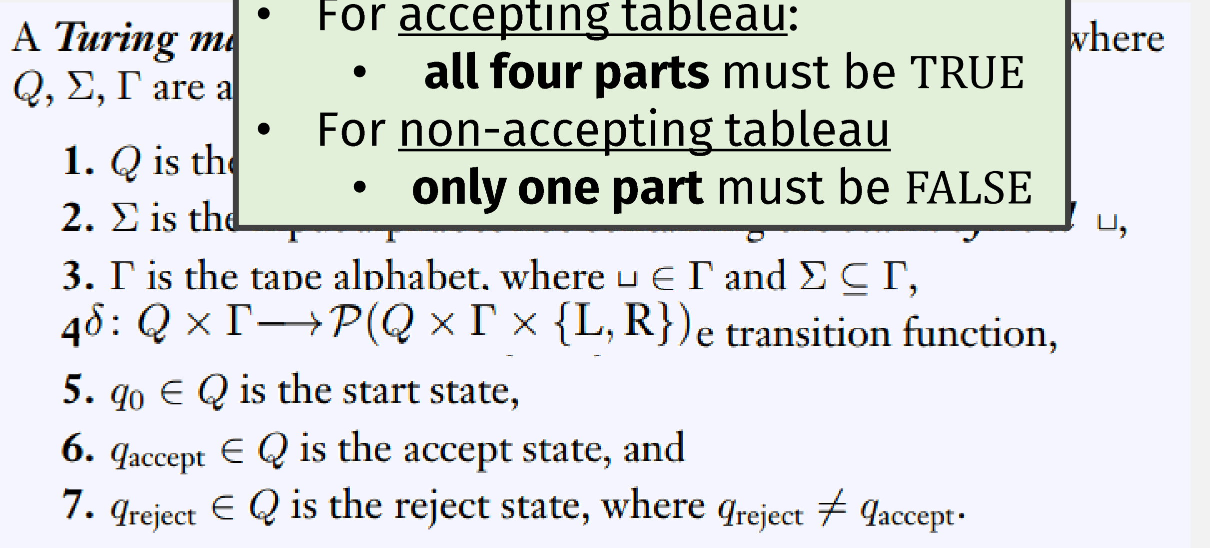
Formula Variables

- A tableau <u>cell</u> has coordinate i,j
- A cell has <u>symbol</u>
  $s \in C = Q \cup \Gamma \cup \{\#\}$
- For every i,j,s create <u>variable</u> $x_{i,j,s}$
- Total variables =
  - Number of cells * |C| =
  - $n^k * n^k * |C| = O(\mathbf{n^{2k}})$



Use these variables to create $\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$ such that:
accepting tableau ⇔ satisfying assignment

- For <u>accepting tableau</u>:
  - **all four parts** must be TRUE
- For <u>non-accepting tableau</u>
  - **only one part** must be FALSE



A ***Turing ma*** ... where
Q, Σ, Γ are a ...

1. Q is th...

2. Σ is th... ⊔,

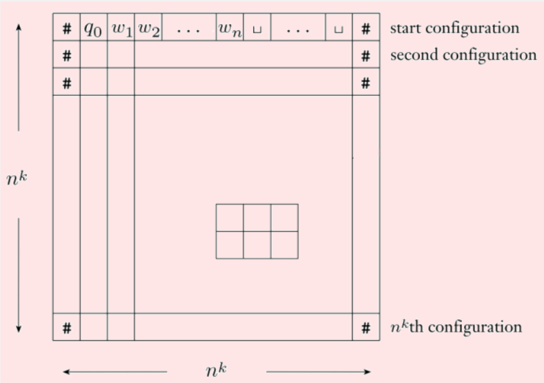
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,

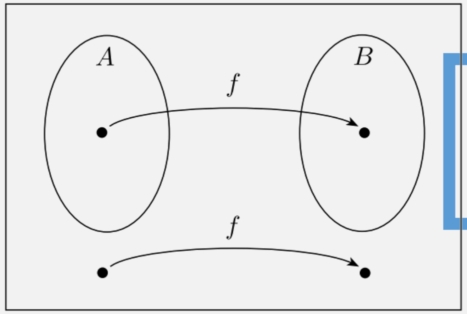
4 $\delta \colon Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$ e transition function,

5. $q_0 \in Q$ is the start state,

6. $q_{\text{accept}} \in Q$ is the accept state, and

7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.





accepting tableau: **all four** must be TRUE
non-accepting tableau: **one** must be FALSE

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

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

$$\phi_{\text{cell}} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in C \\ s \ne t}} \left(\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}} \right) \right) \right]$$

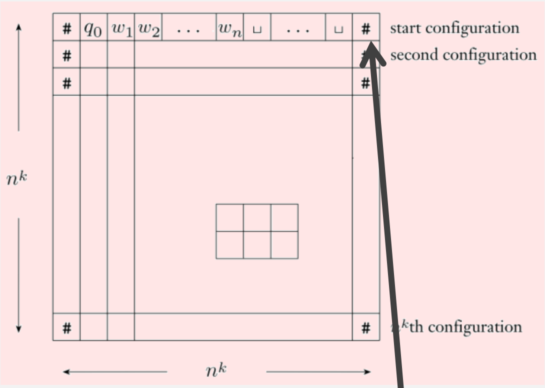
"The following must be TRUE for every cell i,j"

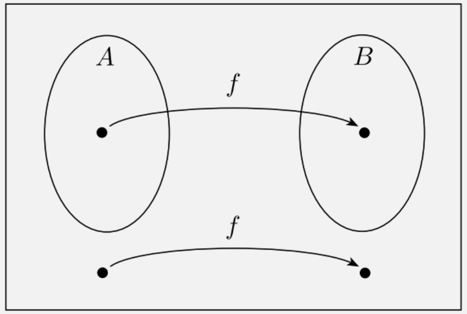
"The variable for one s must be TRUE"

And only one variable for some s must be TRUE

- Does an accepting tableau correspond to a satisfiable (sub)formula?
  - **Yes**, assign $x_{i,j,s}$ = TRUE if it's in the tableau,
  - and assign other vars = FALSE
- Does a non-accepting tableau correspond to an unsatisfiable formula?
  - Not necessarily

accepting tableau: **all four** must be TRUE
non-accepting tableau: **one** must be FALSE



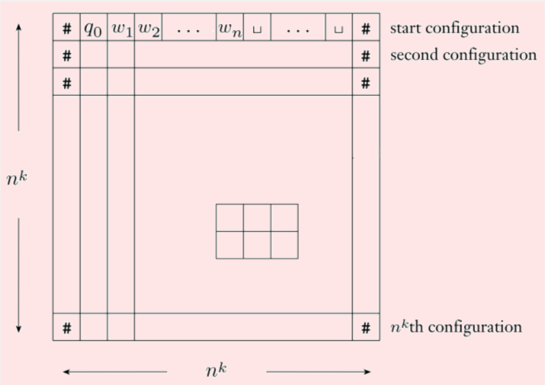


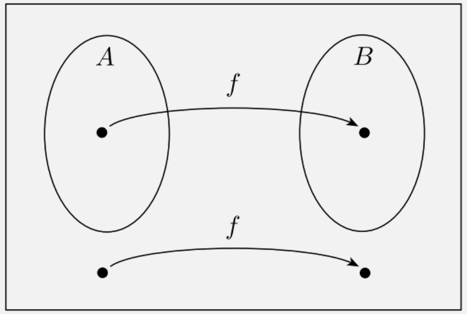
$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$$

The variables in the start config, ANDed together

$$\begin{aligned}\phi_{\text{start}} = {} & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}\end{aligned}$$

- Does an <u>accepting tableau</u> correspond to a satisfiable (sub)formula?
  - **Yes**, assign $x_{i,j,s}$ = TRUE if it's in the tableau,
  - and assign other vars = FALSE
- Does a <u>non-accepting tableau</u> correspond to an unsatisfiable formula?
  - Not necessarily





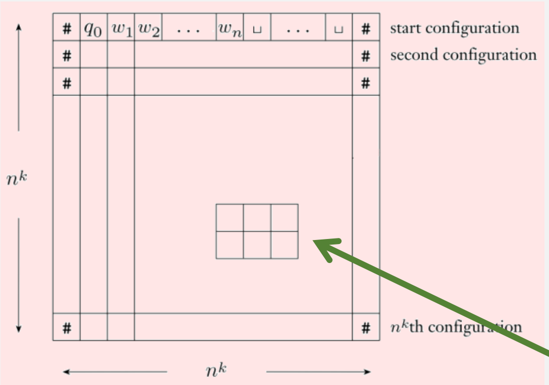
accepting tableau: **all four** must be TRUE
non-accepting tableau: **one** must be FALSE

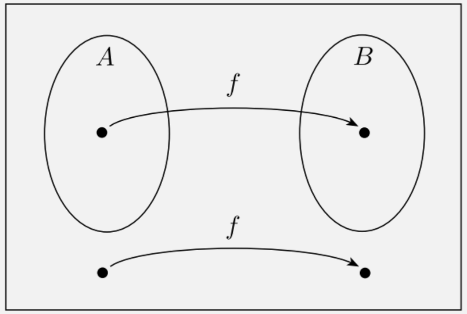
$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$$

$$\phi_{\text{accept}} = \bigvee_{1 \le i,j \le n^k} x_{i,j,q_{\text{accept}}}$$

The state q_{accept} must appear in some cell

- Does an <u>accepting tableau</u> correspond to a satisfiable (sub)formula?
  - **Yes**, assign $x_{i,j,s}$ = TRUE if it's in the tableau,
  - and assign other vars = FALSE
- Does a <u>non-accepting tableau</u> correspond to an unsatisfiable formula?
  - **Yes**, because it wont have q_{accept}





<u>accepting</u> tableau: **all four** must be TRUE
<u>non-accepting</u> tableau: **one** must be FALSE

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

- Ensures that every configuration is <u>legal</u> according to the previous configuration and the TM's δ transitions
- Only need to verify every 2x3 "window"
  - Why?
  - Because in one step, only the cell at the head can change
- E.g., if $\delta(q_1, \mathtt{b}) = \{(q_2,\mathtt{c},\mathrm{L}), (q_2,\mathtt{a},\mathrm{R})\}$
  - Which are <u>legal</u>?

(a)

a	q_1	b
q_2	a	c

(b)

a	q_1	b
a	a	q_2

???
(c)

a	a	q_1
a	a	b

(d)

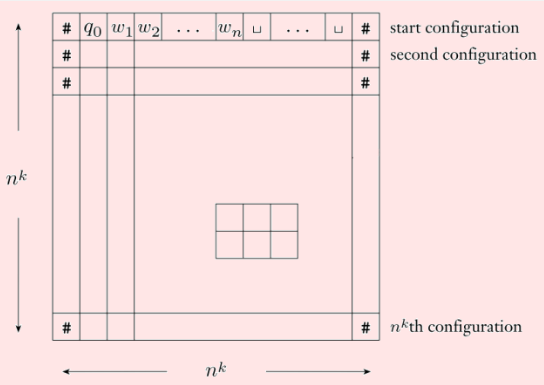
#	b	a
#	b	a

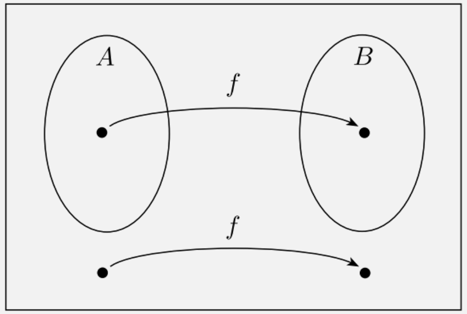
(e)

a	b	a
a	b	q_2

(f)

b	b	b
c	b	b





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

<u>accepting</u> tableau: **all four** must be TRUE
<u>non-accepting</u> tableau: **one** must be FALSE

$$\phi_{\text{move}} = \bigwedge_{1 \le i < n^k,\ 1 < j < n^k} \left(\text{the } (i, j)\text{-window is legal}\right)$$

i,j = upper center cell

$$\bigvee_{\substack{a_1,\ldots,a_6 \\ \text{is a legal window}}} \left(x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6}\right)$$

- Does an <u>accepting tableau</u> correspond to a satisfiable (sub)formula?
  - **Yes**, assign $x_{i,j,s}$ = TRUE if it's in the tableau,
  - and assign other vars = FALSE
- Does a <u>non-accepting tableau</u> correspond to an unsatisfiable formula?
  - Not necessarily

Time complexity of the reduction

- Number of cells = $O(\mathbf{n^{2k}})$

$$\phi_{\text{cell}} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

$O(\mathbf{n^{2k}})$

"The following must be TRUE for every cell i,j"

"The variable for one s must be TRUE"

And only one variable for some s must be TRUE

Time complexity of the reduction

- Number of cells = $O(\mathbf{n^{2k}})$

$$\phi_{\text{cell}} = \bigwedge_{1\le i,j\le n^k} \left[\left(\bigvee_{s\in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t\in C\\ s\neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

$O(\mathbf{n^{2k}})$

$$\begin{aligned}\phi_{\text{start}} = {} & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}\end{aligned}$$

$O(\mathbf{n^k})$

The variables in the start config, ANDed together

Time complexity of the reduction

- Number of cells = $O(\mathbf{n^{2k}})$

$$\phi_{\text{cell}} = \bigwedge_{1\le i,j\le n^k} \left[\left(\bigvee_{s\in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t\in C \\ s\neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

$O(\mathbf{n^{2k}})$

$$\begin{aligned}\phi_{\text{start}} = {} & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}\end{aligned}$$

$O(\mathbf{n^{k}})$

$$\phi_{\text{accept}} = \bigvee_{1\le i,j\le n^k} x_{i,j,q_{\text{accept}}}$$

The state q_{accept} must appear in some cell

$O(\mathbf{n^{2k}})$

Time complexity of the reduction

- Number of cells = $O(\mathbf{n^{2k}})$

$$\phi_{\text{cell}} = \bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right) \wedge \left(\bigwedge_{\substack{s,t\in C\\ s\neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}})\right)\right]$$

$O(\mathbf{n^{2k}})$

$$\begin{aligned}\phi_{\text{start}} = {} & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}\end{aligned}$$

$O(\mathbf{n^{k}})$

$$\phi_{\text{accept}} = \bigvee_{1\le i,j\le n^k} x_{i,j,q_{\text{accept}}}$$

$O(\mathbf{n^{2k}})$

$$\phi_{\text{move}} = \bigwedge_{1\le i<n^k,\ 1<j<n^k} (\text{the } (i,j)\text{-window is legal})$$

$O(\mathbf{n^{2k}})$

Time complexity of the reduction

Total:
$O(\mathbf{n^{2k}})$

- Number of cells = $O(\mathbf{n^{2k}})$

$$\phi_{\text{cell}} = \bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right) \wedge \left(\bigwedge_{\substack{s,t\in C\\ s\neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}})\right)\right]$$

$O(\mathbf{n^{2k}})$

$$\begin{aligned}\phi_{\text{start}} = {} & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}\end{aligned}$$

$O(\mathbf{n^{k}})$

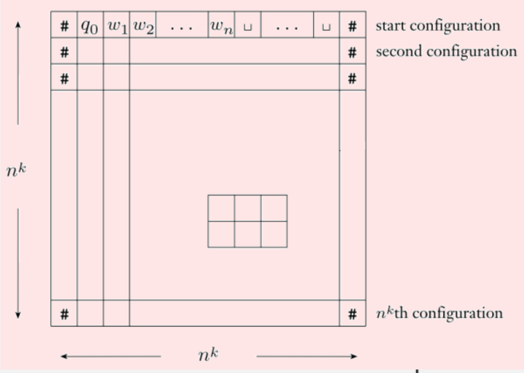
$$\phi_{\text{accept}} = \bigvee_{1\le i,j\le n^k} x_{i,j,q_{\text{accept}}}$$

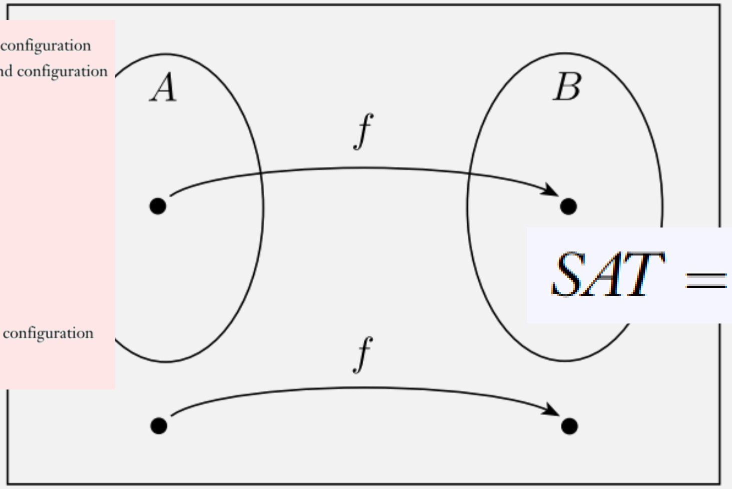
$O(\mathbf{n^{2k}})$

$$\phi_{\text{move}} = \bigwedge_{1\le i<n^k,\ 1<j<n^k} (\text{the } (i,j)\text{-window is legal})$$

$O(\mathbf{n^{2k}})$

QED: *SAT* is NP-complete





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

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

THEOREM 7.36

If B is NP-complete and $B \leq_P C$ for C in NP, then C is NP-complete.

known

unknown

To use this theorem, C must be in NP

Proof:

- For every language A in **NP**, reduce A to C by:
  - First use the reduction from A to B
    - This exists because B is **NP**-Complete
  - Then B to C
    - This is given
- This runs in poly time because of the definition of **NP**-completeness and poly time reducibility

Theorem: *3SAT* is NP-complete.

- Proof: To use thm 7.36, must show poly time reduction from:
  - *SAT* (known to be **NP**-Complete) $SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$
  - to *3SAT* (known to be in **NP**) $3SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable 3cnf-formula}\}$
- Given an arbitrary *SAT* formula:
  1. First convert to CNF (an AND of OR clauses) O(**n**)
     - Use DeMorgan's Law to push negations onto literals

$$\neg(P \vee Q) \iff (\neg P) \wedge (\neg Q) \qquad \neg(P \wedge Q) \iff (\neg P) \vee (\neg Q)$$

     - Distribute ORs to get ANDs outside of parens O(**n**)

$$(P \vee (Q \wedge R)) \Leftrightarrow ((P \vee Q) \wedge (P \vee R))$$

  - Then convert to 3cnf by adding new variables O(**n**)

$$(a_1 \vee a_2 \vee a_3 \vee a_4) \Leftrightarrow (a_1 \vee a_2 \vee z) \wedge (\overline{z} \vee a_3 \vee a_4)$$

THEOREM 7.36

If B is NP-complete and $B \leq_P C$ for C in NP, then C is NP-complete.

Check-in Quiz 12/2

On gradescope

End of Class Survey 12/2

See course website