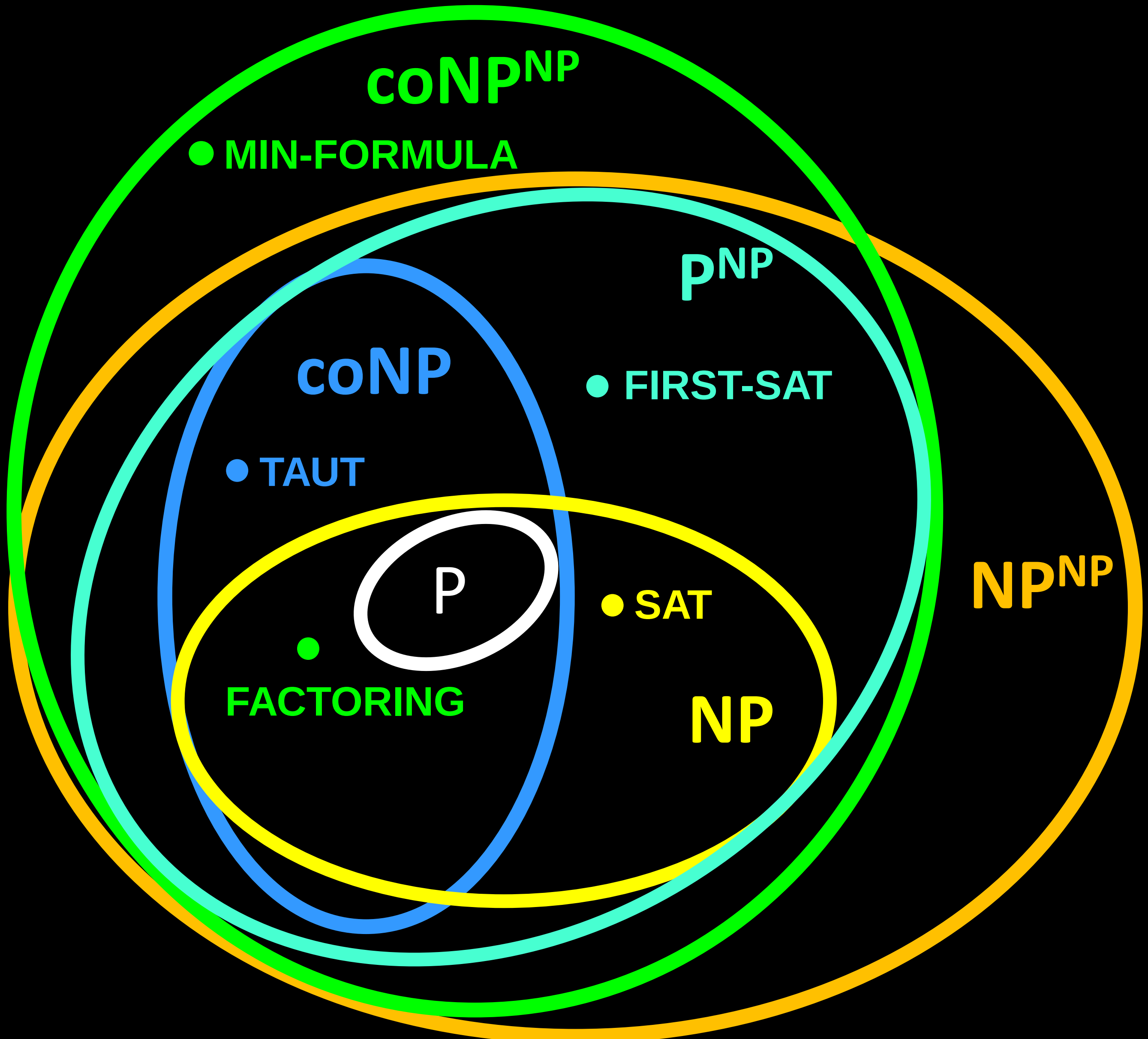
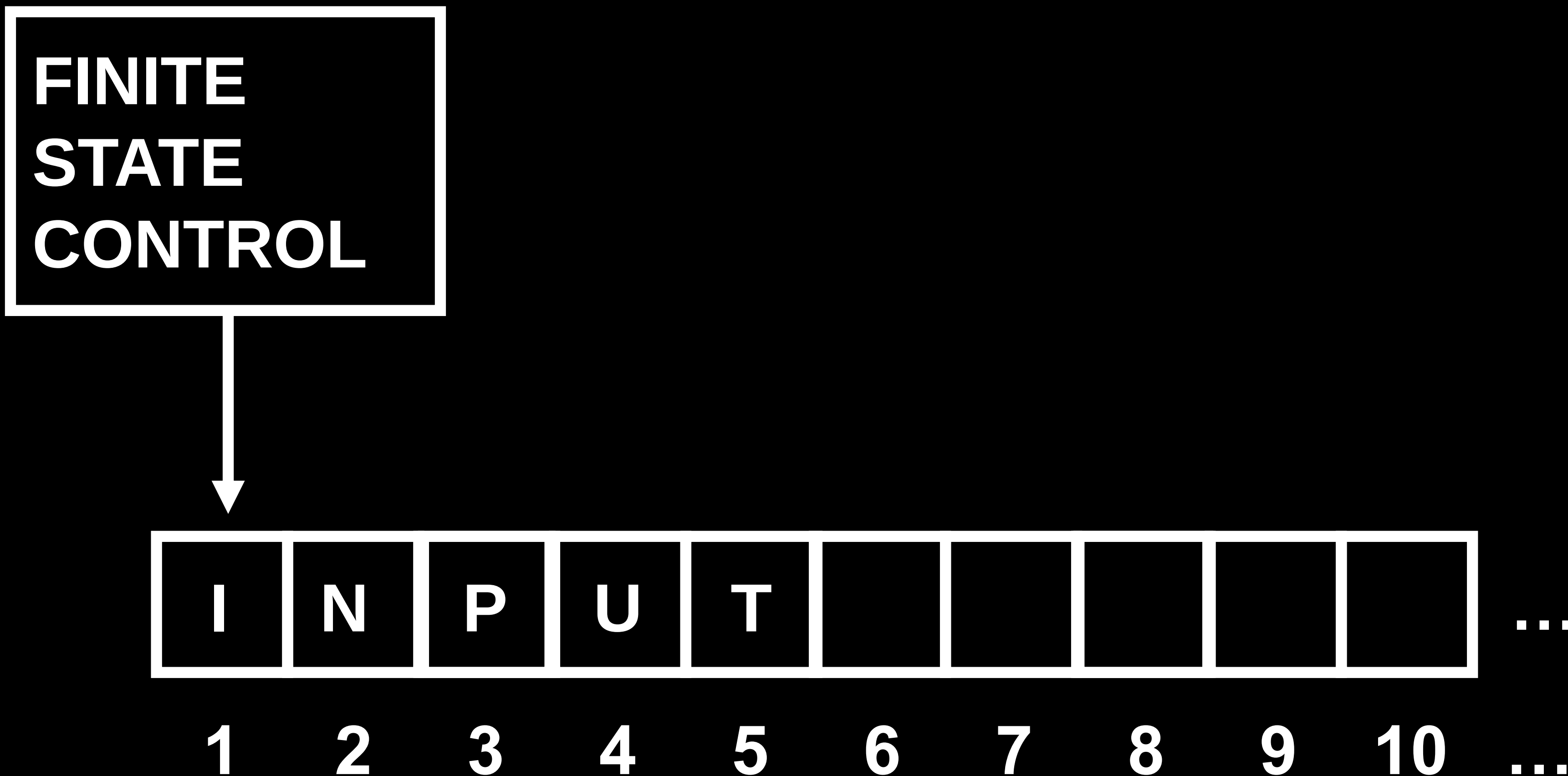
6.045

Lecture 19: Space Complexity


coNP^NP
MIN-FORMULA
P^NP
coNP
FIRST-SAT
TAUT
P
SAT
NP^NP
FACTORING
NP

Space Problems

Measuring Space Complexity



We measure *space* complexity by finding the *largest tape index reached* during the computation

Let M be a deterministic Turing machine (not necessarily halting)

Definition: The **space complexity** of M is the function $S : \mathbb{N} \to \mathbb{N}$, where $S(n)$ is the **largest tape index** reached by M on any input of length n.

Definition: SPACE($S(n)$) =

{ L | L is decided by a Turing machine with O($S(n)$) space complexity}

Theorem: 3SAT $\in$ SPACE(n)

Proof Idea: Given formula ϕ of length n, try all possible assignments A to the (at most n) variables. Evaluate ϕ on each A, and accept iff you find A such that ϕ(A) = 1.
All of this can be done in O(n) space.

Theorem: NTIME(t(n)) is in SPACE(t(n))

Proof Idea: Try all possible computation paths of t(n) steps for an NTM on length-n input. This can be done in O(t(n)) space (store a sequence of t(n) transitions).

One Tape vs Many Tapes

Theorem: Let $s : \mathbb{N} \to \mathbb{N}$ satisfy $s(n) \geq n$, for all n.
Then every $s(n)$ space multi-tape TM has an equivalent $O(s(n))$ space one-tape TM

The simulation of multitape TMs by one-tape TMs already achieves this!

Corollary: The number of tapes doesn't matter for space complexity!

One tape TMs are as good as any other model!

Space Hierarchy Theorem

Intuition: If you have more *space* to work with, then you can solve strictly more problems!

Theorem: For functions s, S : $\mathbb{N} \to \mathbb{N}$ where $s(n)/S(n) \to 0$

$$\mathrm{SPACE}(s(n)) \subsetneq \mathrm{SPACE}(S(n))$$

Proof Idea: Diagonalization

Make a Turing machine N that on input M, simulates the TM M on input <M> using up to S(|M|) space, *then* flips the answer.

Show L(N) is in SPACE(S(n)) but not in SPACE(s(n))

$$\text{PSPACE} = \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k)$$

Since for every k, NTIME(n^k) is in SPACE(n^k), we have:

$$\text{P} \subseteq \text{NP} \subseteq \text{PSPACE}$$

The class PSPACE formalizes the set of problems solvable by computers with ***bounded memory***.

Fundamental (Unanswered) Question:
How does time relate to space, in computing?

SPACE(n^2) problems could potentially take much **longer** than **n^c time** to solve, for *any* c!

Intuition: You can always re-use space, but how can you re-use time?

Is P = PSPACE?

Time Complexity of SPACE[S(n)]

Let M be a **halting** TM with **S(n) space complexity**

How many time steps could M possibly take
on inputs of length n? *Is there an upper bound?*

The number of time steps is at most
the total number of possible *configurations*!

(If a configuration repeats, the machine is looping!)

A configuration of M specifies a
head position, **state**, and **S(n) cells of tape content**.
The total number of configurations is at most:

$$S(n)\ |Q|\ |\Gamma|^{S(n)} = 2^{O(S(n))}$$

Theorem:

For every space-$S(n)$ TM, there is a TM running in $2^{O(S(n))}$ time that decides the same language.

$$\mathrm{SPACE}(s(n)) \subseteq \bigcup_{c \in N} \mathrm{TIME}(2^{c \cdot s(n)})$$

Proof Idea: For each $s(n)$-space bounded TM M there is a $c > 0$ so that on all inputs x, if M runs for more than $2^{c\, s(|x|)}$ time steps on x, then *M must have repeated a configuration*, so M will never halt.

$$\text{PSPACE} = \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k)$$

$$\text{EXPTIME} = \bigcup_{k \in \mathbb{N}} \text{TIME}(2^{n^k})$$

$$\text{PSPACE} \subseteq \text{EXPTIME}$$

$$P \subseteq NP \subseteq PSPACE$$

Is $NP^{NP} \subseteq PSPACE$?

YES

And $coNP^{NP} \subseteq PSPACE$!

Example: MIN-FORMULA is in PSPACE

MIN-FORMULA = { ϕ | ϕ is minimal }

Recall the coNP^{NP} algorithm for MIN-FORMULA:

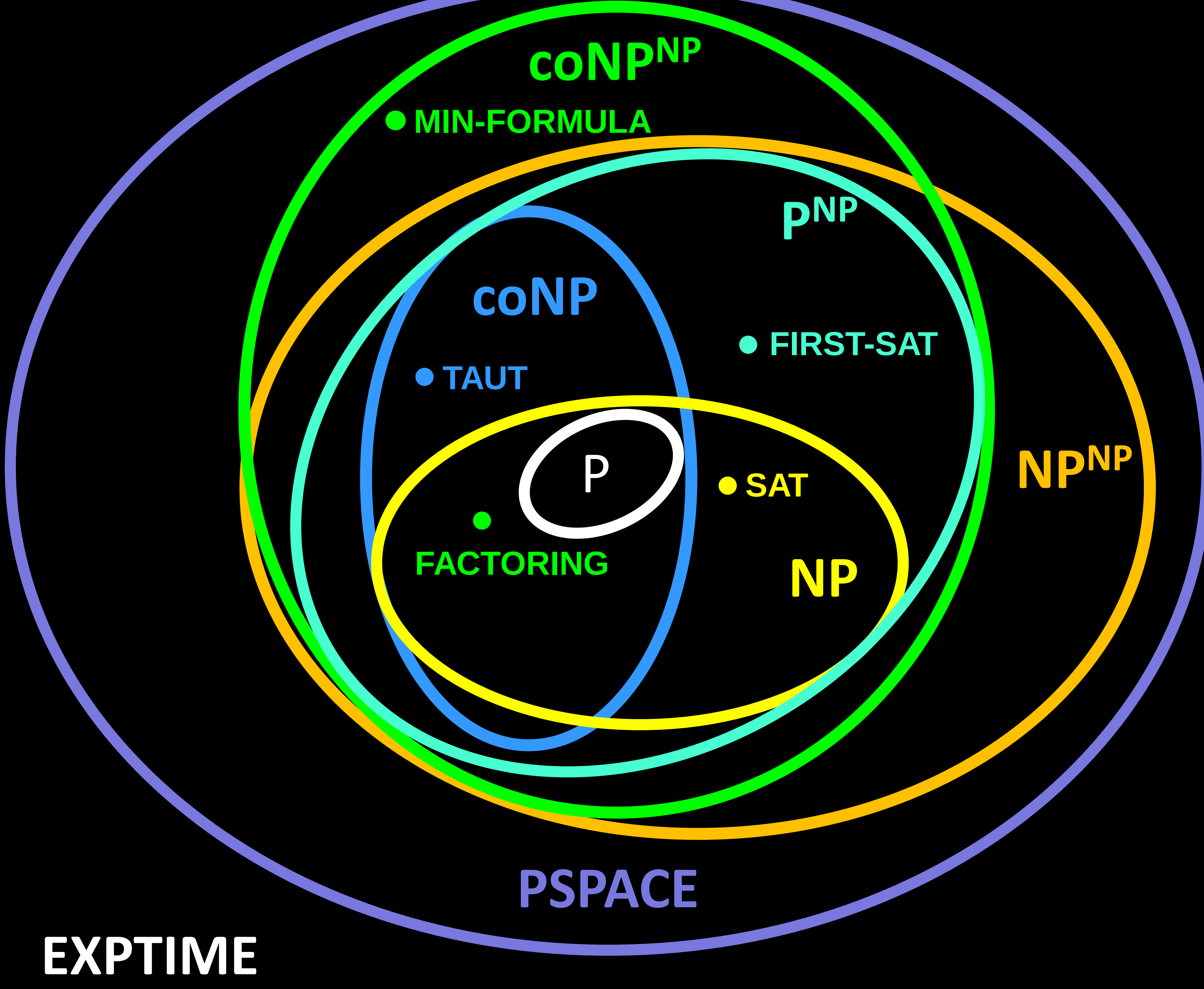
Given a formula ϕ,
Try all formulas ψ such that ψ is smaller than ϕ.
If $((\phi, \psi) \in$ NEQUIV) then *accept* else *reject*

Can store a formula ψ in space $O(|\phi|)$

Can check $(\phi, \psi) \in$ NEQUIV by trying all assignments to the variables of ϕ and ψ

Can store a variable assignment in space $O(|\phi|)$

Evaluating ψ or ϕ on an assignment uses $O(|\phi|)$ space



coNP^NP
MIN-FORMULA
P^NP
coNP
FIRST-SAT
TAUT
P
SAT
NP^NP
FACTORING
NP
PSPACE
EXPTIME

$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$

Theorem: $P \neq EXPTIME$

Why? The Time Hierarchy Theorem!

$$TIME(2^n) \not\subseteq P$$

Therefore $P \neq EXPTIME$

Corollary: At least one of the following is true:

$P \neq NP$, $NP \neq PSPACE$, or $PSPACE \neq EXPTIME$

Proving any one of them would be major!

PSPACE and Nondeterminism

Definition: SPACE(s(n)) =
{ L | L is decided by a Turing machine with
O(s(n)) space complexity}

Definition: NSPACE(s(n)) =
{ L | L is decided by a ***non-deterministic***
Turing Machine with O(s(n)) space complexity}

Recall:

Space S(n) computations can be simulated in at most $2^{O(S(n))}$ time steps

$$\mathrm{SPACE}(s(n)) \subseteq \bigcup_{c \in \mathbb{N}} \mathrm{TIME}(2^{c \cdot s(n)})$$

Idea: After $2^{O(s(n))}$ time steps, a s(n)-space bounded computation must have repeated a configuration, after which it will provably never halt.

Theorem:

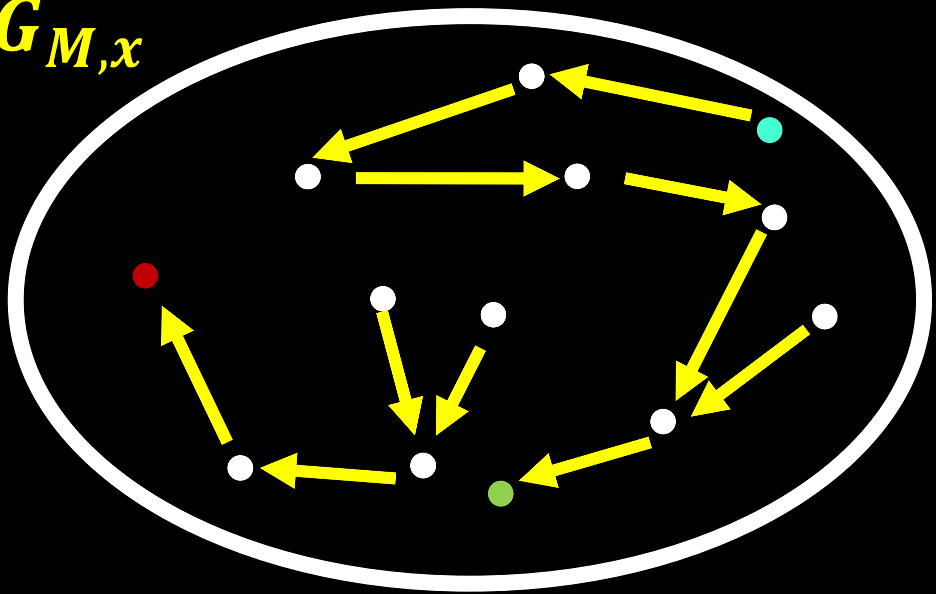
NSPACE S(n) computations can also be simulated in at most $2^{O(S(n))}$ time steps

$$\mathrm{NSPACE}(s(n)) \subseteq \bigcup_{c \in \mathbb{N}} \mathrm{TIME}(2^{c \cdot s(n)})$$

Key Idea: Think of the problem of simulating NSPACE(s(n)) as a problem on graphs.

Def: The **configuration graph of M on x** has **nodes** C for every configuration C of M on x, and **edges** (C, C') if and only if C **yields** C'

$G_{M,x}$



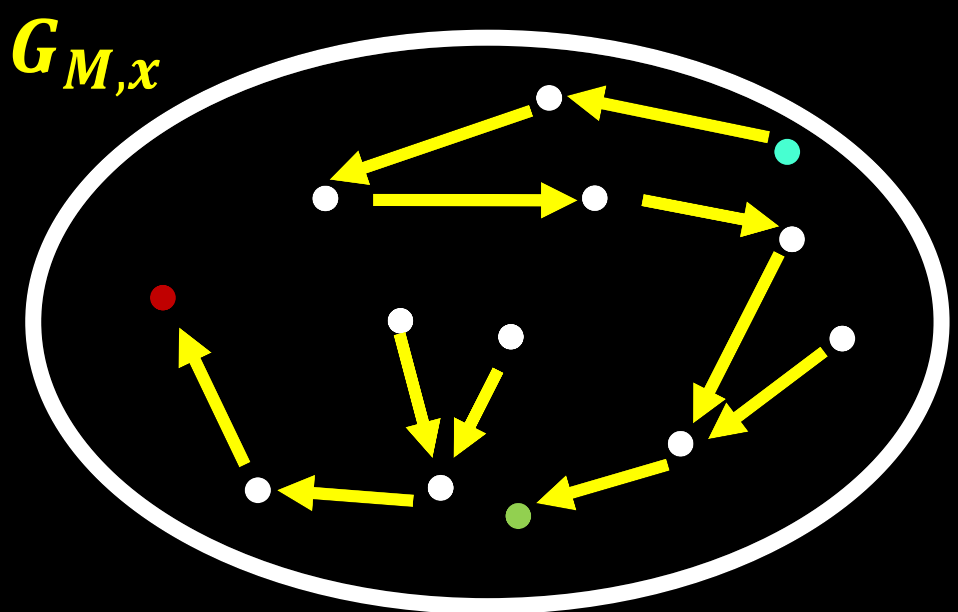
M has space complexity $S(n)$ $\Rightarrow G_{M,x}$ has $2^{d \cdot S(|x|)}$ nodes

M is deterministic $\Rightarrow$ every node has outdegree ≤ 1

M is nondeterministic $\Rightarrow$ some nodes may have outdegree > 1

M accepts x $\Leftrightarrow$ there is a path in $G_{M,x}$ from the initial configuration node to a node in an accept state

Def: The **configuration graph of M on x** has **nodes** C for every configuration C of M on x, and **edges** (C, C') if and only if C **yields** C'



M has space complexity $S(n)$ $\Rightarrow G_{M,x}$ has $2^{d \cdot S(|x|)}$ nodes

M is deterministic $\Rightarrow$ every node has outdegree ≤ 1

M is nondeterministic $\Rightarrow$ some nodes may have outdegree > 1

To simulate a non-deterministic M in $2^{O(S(|x|))}$ time: do BFS in $G_{M,x}$ from the initial configuration!

$$\mathrm{PSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{SPACE}(n^k)$$

$$\mathrm{NPSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{NSPACE}(n^k)$$

SPACE versus NSPACE

Is $\mathrm{NTIME}(n) \subseteq \mathrm{TIME}(n^2)$?

Is $\mathrm{NTIME}(n) \subseteq \mathrm{TIME}(n^k)$ for some $k > 1$?

Nobody knows!

If the answer is yes, then P = NP in fact!
What about the space-bounded setting?

Is $\mathrm{NSPACE}(s(n)) \subseteq \mathrm{SPACE}(s(n)^k)$ for some k? Is PSPACE = NPSPACE?

Savitch's Theorem

Theorem: For functions $s(n)$ where $s(n) \geq n$

$$\mathrm{NSPACE}(s(n)) \subseteq \mathrm{SPACE}(s(n)^2)$$

Proof Try:

Let N be a non-deterministic TM with space complexity s(n).

Construct a deterministic machine M that tries every possible branch of N.

Since each branch of N uses space at most s(n), then M uses space at most s(n)…

There are $2^{\wedge}(2^{s(n)})$ branches to keep track of!

Given configurations C_1 and C_2 of a s(n) space machine N, and a number k (in binary), want to know if N can get from C_1 to C_2 within 2^k steps

Procedure SIM(C_1, C_2, k):

If k = 0 then *accept* iff $C_1 = C_2$ or C_1 yields C_2 within one step.
[uses space O(s(n))]

If k > 0, then for *every* config C_m of O(s(n)) symbols,
if SIM(C_1,C_m,k-1) and SIM(C_m,C_2,k-1) accept
then return *accept*
return *reject* if no such C_m is found

SIM(C_1, C_2, k) has O(k) levels of recursion

Each level of recursion uses O(s(n)) additional space.

Theorem: SIM(C_1, C_2, k) uses only $O(k \cdot s(n))$ space

Theorem: For functions $s(n)$ where $s(n) \geq n$

$$\mathrm{NSPACE}(s(n)) \subseteq \mathrm{SPACE}(s(n)^2)$$

Proof:

Let N be a nondeterministic TM using $s(n)$ space

Let $d > 0$ be such that the number of configurations of $N(w)$ is at most $2^{d\, s(|w|)}$

Here's a deterministic $O(s(n)^2)$ space algorithm for N:

M(w): For all configurations C_a of N(w) in the accept state,
If SIM(q_0w, C_a, $d\, s(|w|)$) accepts, then *accept*
else *reject*

Claim: $L(M) = L(N)$ and M uses $O(s(n)^2)$ space

Theorem: For functions $s(n)$ where $s(n) \geq n$

$$NSPACE(s(n)) \subseteq SPACE(s(n)^2)$$

Proof:

Let **N** be a nondeterministic TM using **s(n) space**

Let **d > 0** be such that the number of configurations of **N(w)** is **at most $2^{d\,s(|w|)}$**

Here's a **deterministic $O(s(n)^2)$** space algorithm for N:

M(w): For all configurations C_a of N(w) in the accept state,
If SIM($q_o w$, C_a, $d\,s(|w|)$) accepts, then *accept*
else *reject*

Why does it take only $s(n)^2$ space?

Theorem: For functions $s(n)$ where $s(n) \geq n$

$$\mathrm{NSPACE}(s(n)) \subseteq \mathrm{SPACE}(s(n)^2)$$

Proof:

Let N be a nondeterministic TM using $s(n)$ space

Let $d > 0$ be such that the number of configurations of $N(w)$ is at most $2^{d\, s(|w|)}$

Here's a deterministic $O(s(n)^2)$ space algorithm for N:

M(w): For all configurations C_a of N(w) in the accept state,
If SIM($q_o w$, C_a, $d\, s(|w|)$) accepts, then *accept*
else *reject*

SIM uses $O(k \cdot s(|w|))$ space to simulate 2^k steps of N(w).

For $k = d\, s(|w|)$ we have $O(k \cdot s(|w|)) \leq O(s(|w|)^2)$ space

$$\text{PSPACE} = \bigcup_{k \in N} \text{SPACE}(n^k)$$

$$\text{NPSPACE} = \bigcup_{k \in N} \text{NSPACE}(n^k)$$

PSPACE = NPSPACE!