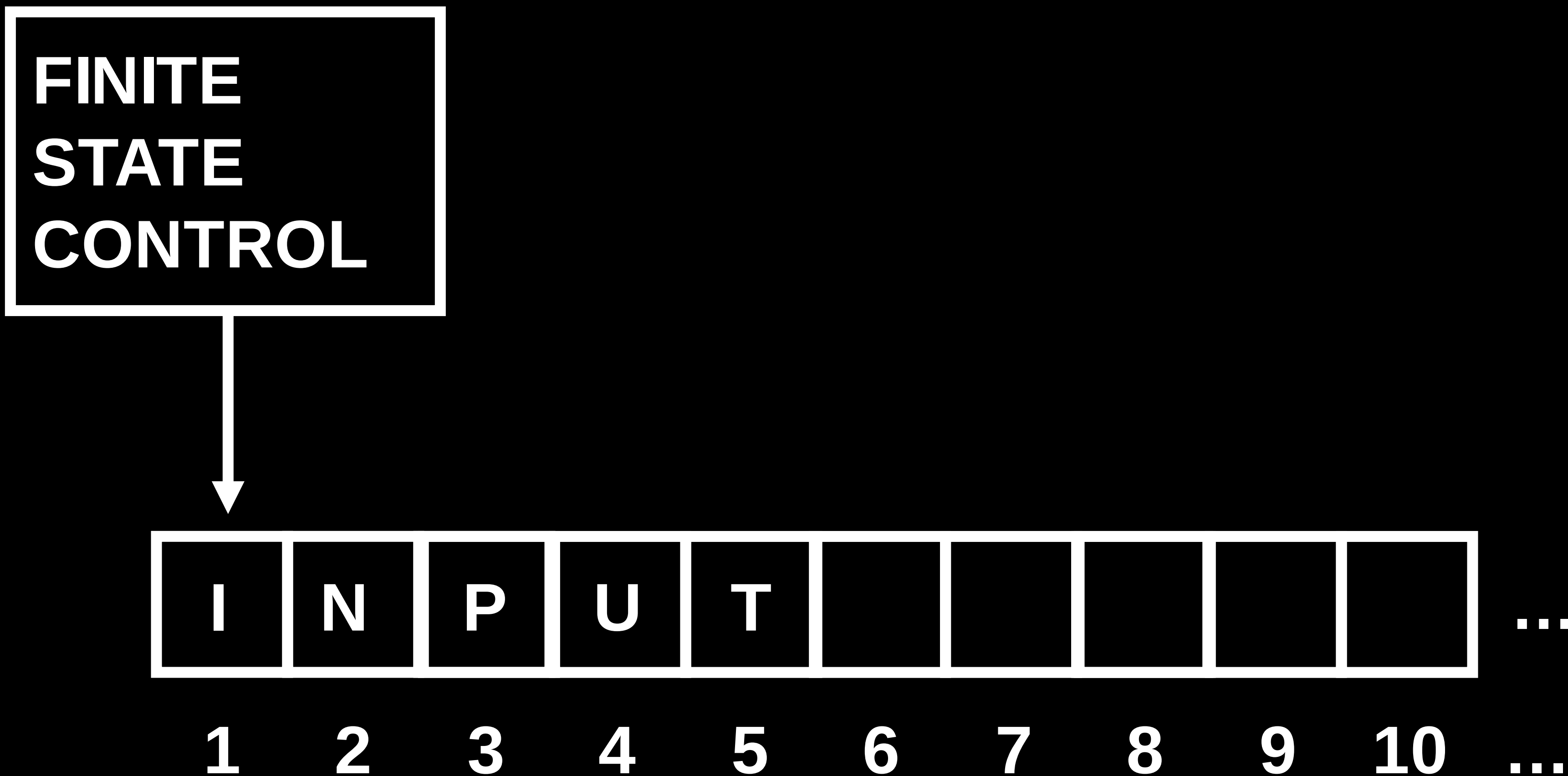
15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

Space Complexity: Savitch's Theorem and PSPACE-Completeness

Tuesday April 15

MEASURING SPACE COMPLEXITY



We measure space complexity by looking at the furthest tape cell reached during the computation

Let M = deterministic TM that halts on all inputs.

Definition: The space complexity of M is the function s : $N \to N$, where s(n) is the furthest tape cell reached by M on any input of length n.

Let N be a non-deterministic TM that halts on all inputs in all of its possible branches.

Definition: The space complexity of N is the function s : $N \to N$, where s(n) is the furthest tape cell reached by M, on any branch if its computation, on any input of length n.

Definition: SPACE(s(n)) =
{ L | L is a language decided by a O(s(n)) space deterministic Turing Machine }

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

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

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

3SAT $\in$ SPACE(n) $\subset$ PSPACE

(	x	$\vee$	$\neg$	y	$\vee$	x	)		(	y	$\vee$	x	$\vee$	y	)						
(	x	$\vee$	$\neg$	y	$\vee$	x	)		(	y	$\vee$	x	$\vee$	y	)		#	x		y	
(	x	$\vee$	$\neg$	y	$\vee$	x	)		(	y	$\vee$	x	$\vee$	y	)		#	x	0	y	0
(	x	$\vee$	$\neg$	y	$\vee$	x	)		(	y	$\vee$	x	$\vee$	y	)		#	x	0	y	1
(	x	$\vee$	$\neg$	y	$\vee$	x	)		(	y	$\vee$	x	$\vee$	y	)		#	x	1	y	0

Assume a deterministic Turing machine that halts on all inputs runs in space s(n)

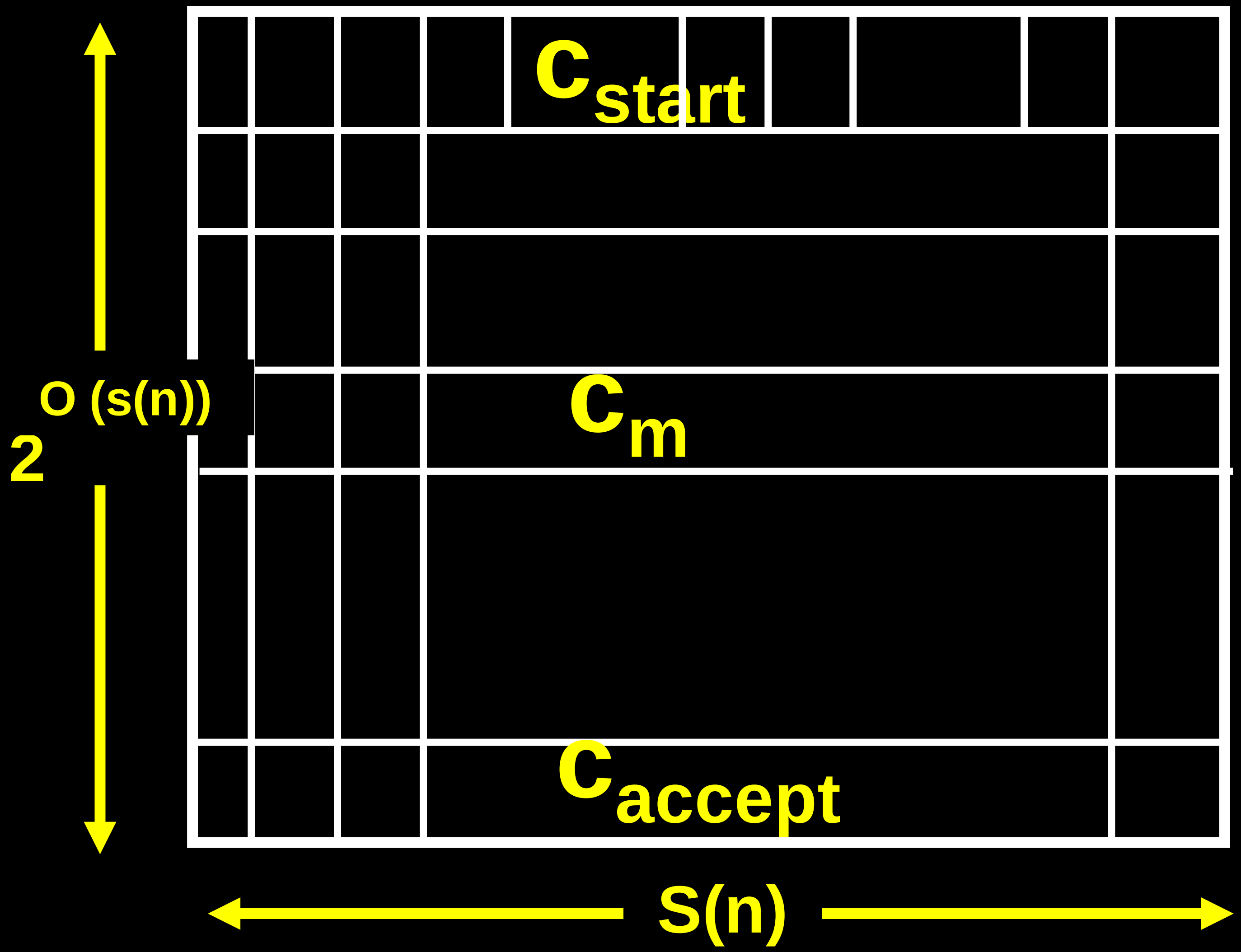
Question: What's an upper bound on the number of time steps for this machine?

A configuration gives a head position, state, and tape contents. Number of configurations is at most:

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

Number of Configurations

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



MORAL:

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

PSPACE $\subseteq$ EXPTIME

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

SAVITCH'S THEOREM

Is NTIME(t(n)) $\subseteq$ TIME(t(n))?

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

We don't know in general!

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

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

$$\mathbf{s(n) \geq n}$$

SAVITCH'S THEOREM

Is NTIME(t(n)) $\subseteq$ TIME(t(n))?

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

We don't know in general!

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

therefore NPSPACE $\subseteq$ PSPACE

SAVITCH'S THEOREM

Is NTIME(t(n)) $\subseteq$ TIME(t(n))?

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

We don't know in general!

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

therefore 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) for each branch ..

SAVITCH'S THEOREM

Theorem: For functions s(n) where s(n) ≥ n

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

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) for each branch ..

There are 2^(O(2^O(s))) branches to keep track of!

We need to simulate a non-deterministic computation and save as much space as possible

IDEA: Given two configurations C_1 and C_2 of an $s(n)$ space machine N, and a number t, determine if N can get from C_1 to C_2 within t steps

Procedure CANYIELD(C_1, C_2, t):

If $t = 0$ then *accept* iff $C_1 = C_2$
If $t = 1$ then *accept* iff C_1 yields C_2 within one step.

Use transition map of N to check [uses space $O(s(n))$]

If $t > 1$, then Accept if and only if

for some configuration C_m of size $s(n)$, both
CANYIELD(C_1,C_m ,$t/2$) and CANYIELD(C_m,C_2, $t/2$) accept

IDEA: Given two configurations C_1 and C_2 of an $s(n)$ space machine N, and a number t, determine if N can get from C_1 to C_2 within t steps

Procedure CANYIELD(C_1, C_2, t):

If $t = 0$ then *accept* iff $C_1 = C_2$

If $t = 1$ then *accept* iff C_1 yields C_2 within one step.

Use transition map of N to check [uses space O(s(n))]

If $t > 1$, then Accept if and only if

for some configuration C_m of size s(n), both CANYIELD(C_1,C_m ,t/2) and CANYIELD(C_m,C_2, t/2) accept

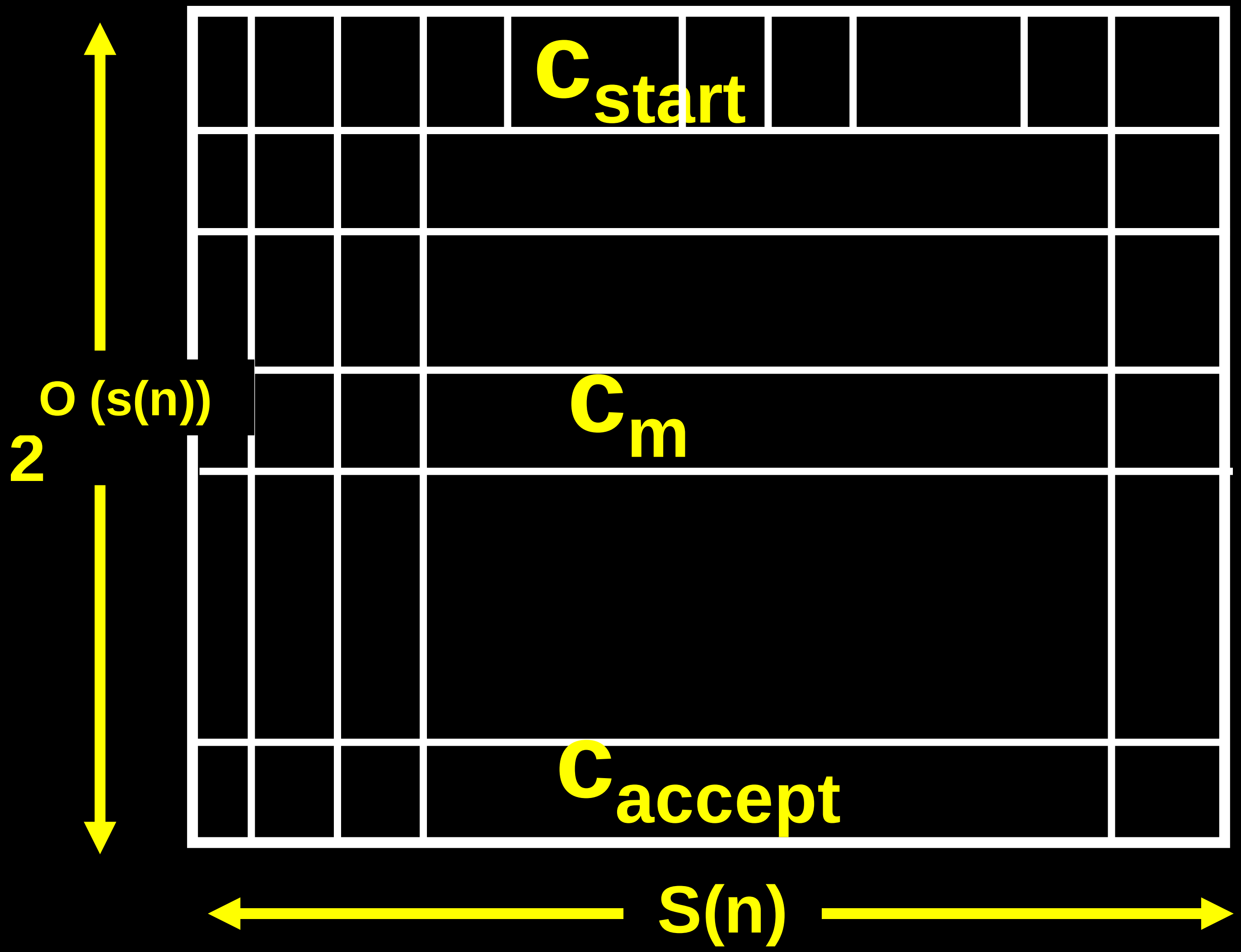
CANYIELD(C_1, C_2, t) has log(t) levels of recursion.

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

So CANYIELD(C_1, C_2, t) uses O(s(n) log(t)) space.

Number of Configurations

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



IDEA: Given two configurations C_1 and C_2 of an $s(n)$ space machine N, and a number t, determine if N can get from C_1 to C_2 within t steps

Procedure CANYIELD(C_1, C_2, t):

If $t = 0$ then *accept* iff $C_1 = C_2$
If $t = 1$ then *accept* iff C_1 yields C_2 within one step.

Use transition map of N to check [uses space O(s(n))]

If $t > 1$, then Accept if and only if

for some configuration C_m of size s(n), both
CANYIELD(C_1,C_m ,$t/2$) and CANYIELD(C_m,C_2, $t/2$) accept

CANYIELD(C_1, C_2, t) has log(t) levels of recursion.
Each level of recursion uses O(s(n)) additional space to store C_m.
So CANYIELD(C_1, C_2, t) uses O(s(n) log(t)) space.

IDEA: Given two configurations C_1 and C_2 of an s(n) space machine N, and a number t, determine if N can get from C_1 to C_2 within t steps

Procedure CANYIELD(C_1, C_2, t):

If t = 0 then *accept* iff $C_1 = C_2$
If t = 1 then *accept* iff C_1 yields C_2 within one step.

Use transition map of N to check [uses space O(s(n))]

If t > 1, then Accept if and only if

for some configuration C_m of size s(n), both
CANYIELD(C_1,C_m ,t/2) and CANYIELD(C_m,C_2, t/2) accept

M: On input w,
Output the result of CANYIELD(c_{start}, c_{accept}, $2^{ds(n)}$)
CANYIELD(C_1, C_2, $2^{ds(n)}$) uses $O(s(n) \log(2^{ds(n)}))$ space.

IDEA: Given two configurations C_1 and C_2 of an s(n) space machine N, and a number t, determine if N can get from C_1 to C_2 within t steps

Procedure CANYIELD(C_1, C_2, t):

If t = 0 then *accept* iff $C_1 = C_2$
If t = 1 then *accept* iff C_1 yields C_2 within one step.

Use transition map of N to check [uses space O(s(n))]

If t > 1, then Accept **if and only if**

for some configuration C_m of size s(n), both CANYIELD(C_1,C_m ,t/2) and CANYIELD(C_m,C_2, t/2) accept

M: On input w,
Output the result of CANYIELD(c_{start}, c_{accept}, $2^{ds(n)}$)
Here d > 0 is chosen so that $2^{ds(|w|)}$ upper bounds the number of configurations of N(w)

Theorem: For a function s where $s(n) \geq n$

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

Proof:

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

Modify N so that when it accepts, it goes to a special state q_s, clears its tape, and moves its head to the leftmost cell

N has a UNIQUE accepting configuration: $C_{acc} = q_s\square\ldots\square$

Theorem: For a function s where $s(n) \geq n$

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

Proof:

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

Modify N so that when it accepts, it goes to a special state q_s, clears its tape, and moves its head to the leftmost cell

N has a UNIQUE accepting configuration: $C_{acc} = q_s\square\ldots\square$

Construct a deterministic M that on input w, runs CANYIELD(C_0, C_{acc}, $2^{ds(|w|)}$)

Here $d > 0$ is chosen so that $2^{d\,s(|w|)}$ upper bounds the number of configurations of N(w)

=> $2^{ds(|w|)}$ is an upper bound on the running time of N(w).

Theorem: For a function s where $s(n) \geq n$

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

Proof:

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

Modify N so that when it accepts, it goes to a special state q_s, clears its tape, and moves its head to the leftmost cell

N has a UNIQUE accepting configuration: $C_{acc} = q_s\square\ldots\square$

Construct a deterministic M that on input w,
runs CANYIELD(C_0, C_{acc}, $2^{ds(|w|)}$)

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

Theorem: For a function s where $s(n) \geq n$

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

Proof:

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

Modify N so that when it accepts, it goes to a special state q_s, clears its tape, and moves its head to the leftmost cell

N has a UNIQUE accepting configuration: $C_{acc} = q_s\square\ldots\square$

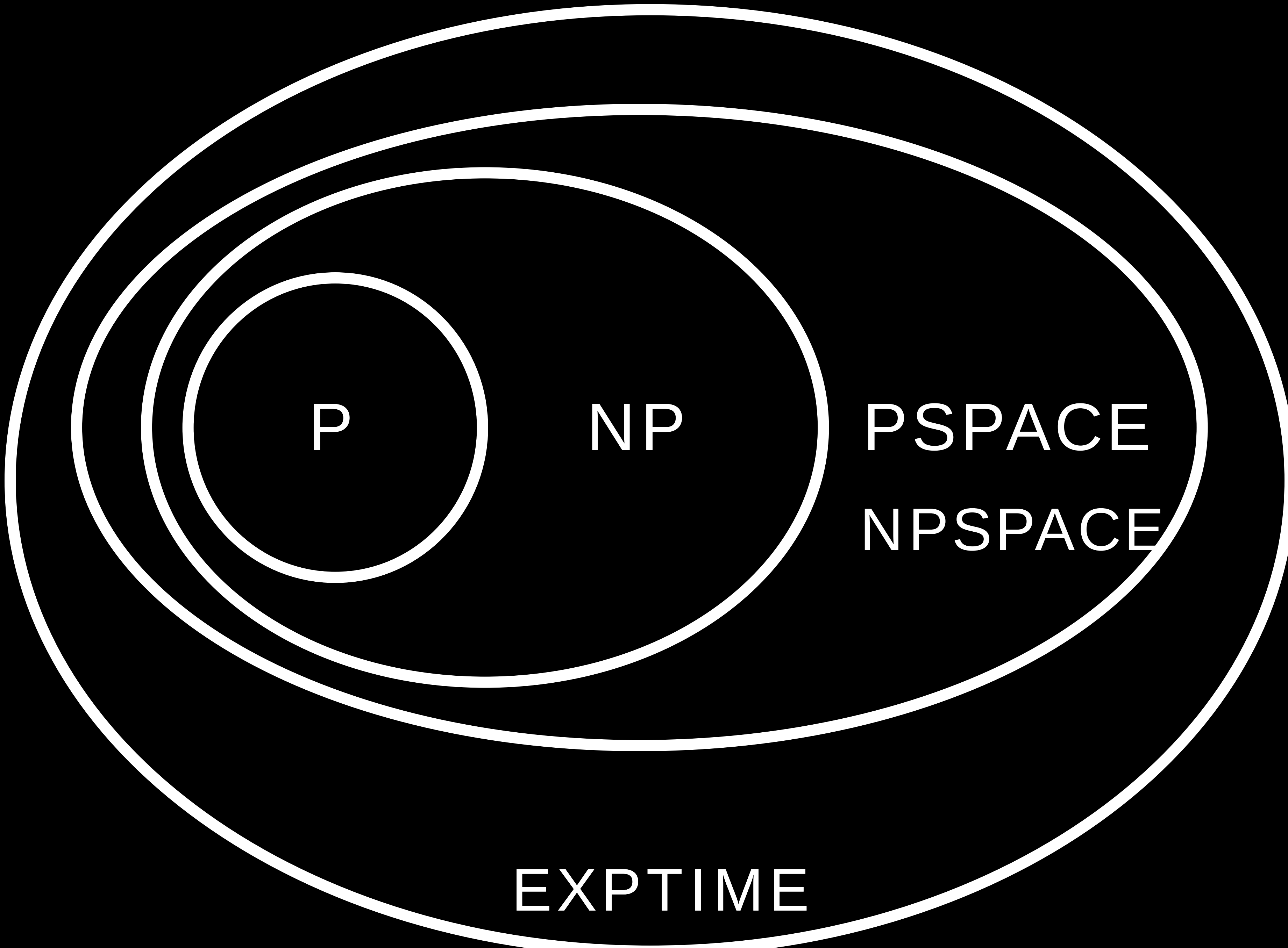
Construct a deterministic M that on input w, runs CANYIELD(C_0, C_{acc}, $2^{ds(|w|)}$)

Uses $\log(2^{d\,s(|w|)})$ recursions. Each level of recursion uses $O(s(n))$ extra space. Therefore uses $O(s(n)^2)$ space!

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

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

PSPACE = NPSPACE


P
NP
PSPACE
NPSPACE
EXPTIME

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME}$$

$$\mathbf{P} \neq \mathbf{EXPTIME}$$

TIME HIERARCHY THEOREM

TIME HIERARCHY THEOREM

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

Theorem: **For functions f, g where $g(n)/(f(n))^2 \to \infty$**

$$\mathrm{TIME}(g(n)) \not\subset \mathrm{TIME}(f(n))$$

So, for all k, since $2^n/n^{2k} \to \infty$,

$$\mathrm{TIME}(2^n) \not\subset \mathrm{TIME}(n^k)$$

Therefore, $\mathrm{TIME}(2^n) \not\subset \mathrm{P}$

TIME HIERARCHY THEOREM

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

Theorem: For functions f, g where $g(n)/(f(n))^2 \rightarrow$ ▯

$$\mathrm{TIME}(g(n)) \not\subset \mathrm{TIME}(f(n))$$

Proof IDEA: Diagonalization

Make a machine M that works in g(n) time and "does the opposite" of all f(n) time machines on at least one input

So L(M) is in TIME(g(n)) but not TIME(f(n))

TIME HIERARCHY THEOREM

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

Theorem: **For functions f, g where $g(n)/(f(n))^2 \to \infty$**

$$\mathrm{TIME}(g(n)) \not\subseteq \mathrm{TIME}(f(n))$$

Proof IDEA: Diagonalization

Need $g(n) >> f(n)^2$ to ensure that you can simulate an arbitrary machine running in $f(n)$ time with a single machine that runs in $g(n)$ time.

So L(M) is in TIME(g(n)) but not TIME(f(n))

Definition: Language B is PSPACE-complete if:

1. $B \in$ PSPACE
2. Every A in PSPACE is poly-time reducible to B (i.e. B is PSPACE-hard)