Space Complexity

- We consider space (a.k.a. memory, storage, etc.).

- To consider space < n, we work with TM with two tapes:

  Input tape: contains input, read-only

  Work tape: initially blank, read-write

Only work tapes counts towards space.

Example: Recall the TM for $\{a^m b^m c^m : m \geq 0\}$:

M := “On input w:

(1) Scan tape and cross off one a, one b, and one c

(2) If none of these symbols is found, ACCEPT

(3) If not all of these symbols is found, or if found in the wrong order, REJECT

(4) Go back to (1).”

Does this fit our model of space?

Example: Recall the TM for $\{a^m b^m c^m : m \geq 0\}$:

M := "On input w:

(1) Scan tape and cross off one a, one b, and one c

(2) If none of these symbols is found, ACCEPT

(3) If not all of these symbols is found,
or if found in the wrong order, REJECT

(4) Go back to (1)."

Does this fit our model of space?

No. We cannot write on the input.
How can you modify to fit our model?

Example: TM for $\{a^m b^m c^m : m \geq 0\}$:

M := “On input w:

(0) Copy the input w on the work tape.

(1) Scan work tape and cross off one a, one b, and one c

(2) If none of these symbols is found, ACCEPT

(3) If not all of these symbols is found, or if found in the wrong order, REJECT

(4) Go back to (1).”

This fits our model of space

How much space does this use?

Example: TM for $\{a^m b^m c^m : m \geq 0\}$:

M := “On input w:

(0) Copy the input w on the work tape.

(1) Scan work tape and cross off one a, one b, and one c

(2) If none of these symbols is found, ACCEPT

(3) If not all of these symbols is found, or if found in the wrong order, REJECT

(4) Go back to (1).”

This fits our model of space
How much space does this use?

Space = n

Can you use less space?

Example: TM for $\{a^m b^m c^m : m \geq 0\}$ using less space:

M := "On input w:

Scan tape, if find symbols in wrong order, REJECT

Count the a, b, and c; write numbers on work tape

If the numbers are equal ACCEPT, else REJECT"

- How to count the a?

  Initialize a binary counter to 0 on work tape.

  While input head is on an a: {

    Move input head right

    Increase counter on work tape by 1.

  }

- How much space does this take?

Example: TM for $\{a^m b^m c^m : m \geq 0\}$ using less space:

M := “On input w:

Scan tape, if find symbols in wrong order, REJECT

Count the a, b, and c; write numbers on work tape

If the numbers are equal ACCEPT, else REJECT”

- How to count the a?

  Initialize a binary counter to 0 on work tape.

  While input head is on an a: {

    Move input head right

    Increase counter on work tape by 1.

  }

- How much space does this take? c log(n).

- Definition:
SPACE(s(n)) = languages decided by TM using space $\leq$ s(n)

- This is interesting both for s(n) $\geq$ n and for s(n) $\leq$ n,

  for example with s(n) = c log(n) you can do a lot already

- Fact: SPACE(c log n) can compute many basic functions
- It is easy to show addition is in SPACE(c log n)
- It is harder to show multiplication is in SPACE(c log n)
- It is a breakthrough paper that division is in SPACE(c log n)

- Definition:

  A configuration of a TM using space s consists of:

  state

  contents of the work tape

  position of the head on the work tape

  head positions on input tape

  How many choices for each item?

- Definition:
  A configuration of a TM using space s consists of:

state	$\| Q \|$
contents of the work tape	?
position of the head on the work tape	
head positions on input tape	

How many choices for each item?

- Definition:
  A configuration of a TM using space s consists of:

state	$\lvert Q \rvert$
contents of the work tape	$\lvert \Gamma \rvert^s$
position of the head on the work tape	?
head positions on input tape	

How many choices for each item?

- Definition:

  A configuration of a TM using space s consists of:

state	$\lvert Q \rvert$
contents of the work tape	$\lvert \Gamma \rvert^s$
position of the head on the work tape	s
head positions on input tape	?

How many choices for each item?

- Definition:

  A configuration of a TM using space s consists of:

state	$\| Q \|$
contents of the work tape	$\| \Gamma \|^s$
position of the head on the work tape	s
head positions on input tape	n

Total number of configurations is:

$|Q| \cdot | \Gamma |^s \cdot s \cdot n \leq c^s \cdot n$, for a constant c

- Claim: SPACE(s(n)) $\subseteq$ TIME($c^{s(n)}$), $\forall$ s(n) $\geq$ log n
- Proof:

  ?

- Note: Feel free to allow 2-tape TM for TIME too.

- Claim: SPACE(s(n)) $\subseteq$ TIME($c^{s(n)}$), $\forall$ s(n) $\geq$ log n
- Proof:

  Let M be a TM running in space s(n).

  Number of possible configurations $\leq c^{s(n)} \cdot n \leq (2c)^{s(n)}$

  No two configurations may repeat.

  Hence M takes at most $(2c)^{s(n)}$ steps. ■

- Claim: TIME(t(n)) $\subseteq$ SPACE(t(n))
- Proof:

?

- Claim: TIME(t(n)) $\subseteq$ SPACE(t(n))
- Proof:

In time t you can only use t cells. ■

- Summary:

$$\mathrm{TIME}(t(n)) \subseteq \mathrm{SPACE}(t(n)) \subseteq \mathrm{TIME}(c^{t(n)}),\ \forall\ t(n) \geq \log n$$

- Next: Non-determinism

- Recall definition of NTIME:

  $\text{NTIME}(t(n)) = \{ L : \exists M : \forall x$ of length n
  $x \in L \Leftrightarrow \exists y, |y| \le t(n), M(x,y)$ accepts in $\le t(n)$

- We want to define NSPACE

- We can't write y on input or work tape,
  the model would not be what we want

- So instead we consider non-deterministic TM

- Definition: NSPACE(s(n)) = languages decided by non-deterministic TM using space < s(n)

- Intuition: "non-deterministic TM : TM = NFA : DFA"

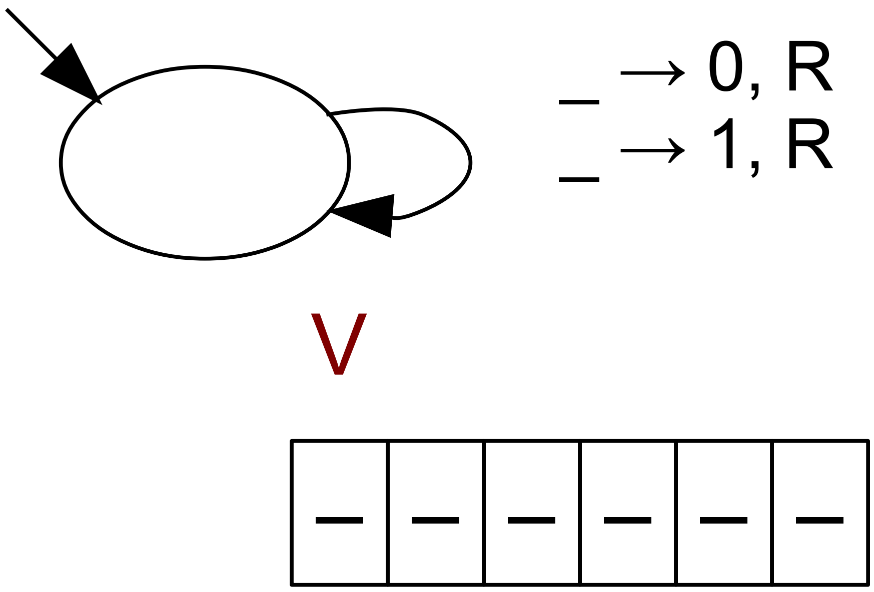
- $\delta : Q \times \Gamma^2 \rightarrow$ Powerset$(Q \times \Gamma^2 \times \{L,R\}^2)$

  Recall that we are working with two-tape TM:

- This allows the TM to "guess" strings.

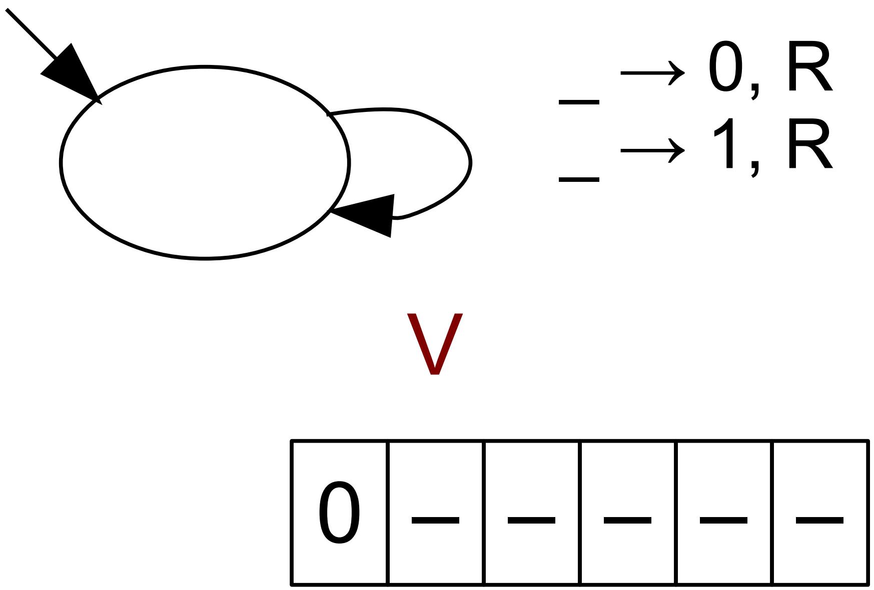
Example "Guessing a string"

This example shows a valid sequence of configurations for a non-deterministic TM



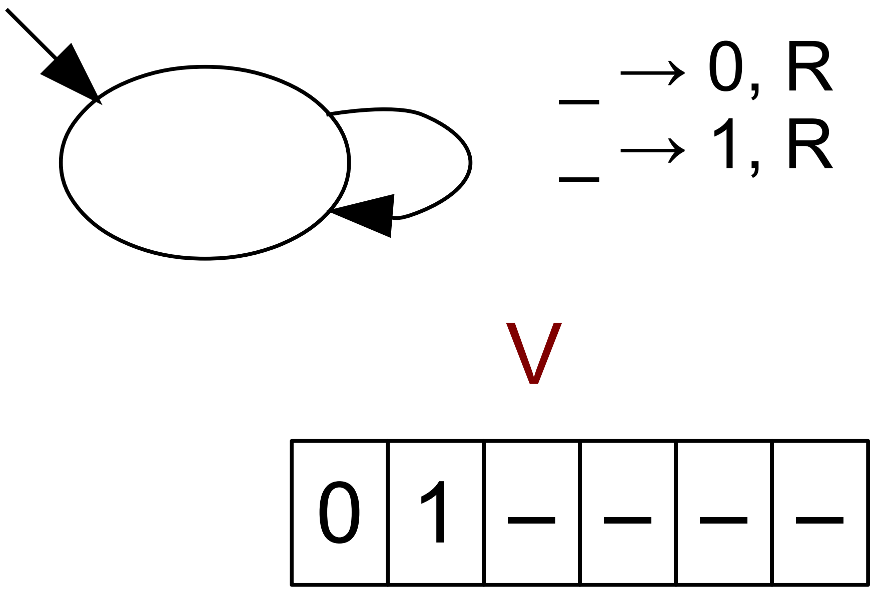
Example "Guessing a string"

This example shows a valid sequence of configurations for a non-deterministic TM



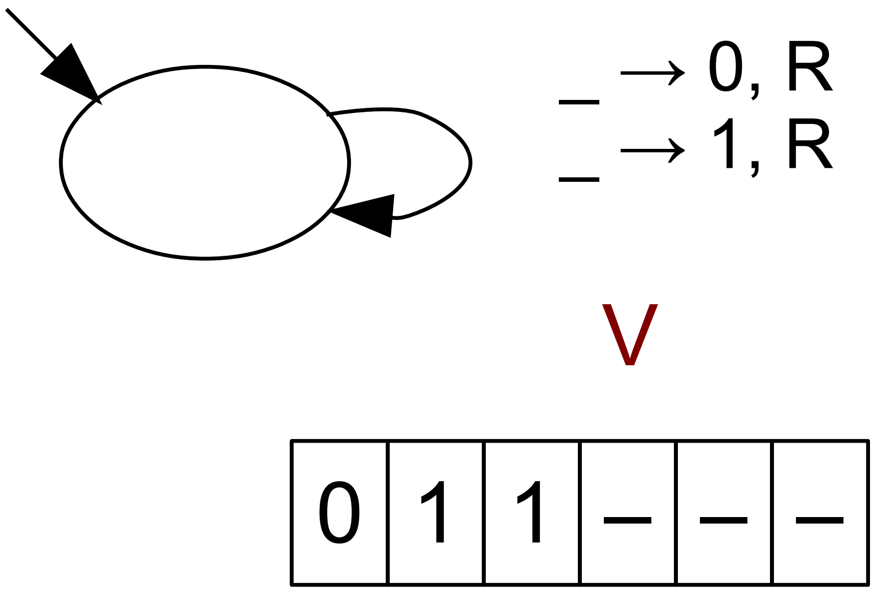
Example "Guessing a string"

This example shows a valid sequence of configurations for a non-deterministic TM



Example "Guessing a string"

This example shows a valid sequence of configurations for a non-deterministic TM



and so on...

PATH = {(G,s,t) : G is a directed graph with a path from s to t }

- Claim: PATH $\in$ NSPACE(?)

PATH = {(G,s,t) : G is a directed graph with a path from s to t }

- Claim: PATH $\in$ NSPACE(10 log n)
- Proof:

?

PATH = {(G,s,t) : G is a directed graph with a path from s to t }

- Claim: PATH $\in$ NSPACE(10 log n)
- Proof:

M := "On input (G,s,t):

Let v := s.

For i = 0 to |G|

?

?

?

REJECT"

PATH = {(G,s,t) : G is a directed graph with a path from s to t }

- **Claim**: PATH $\in$ NSPACE(10 log n)
- **Proof**:

M := “On input (G,s,t):

Let v := s.

For i = 0 to |G|
    Guess a neighbor w of v.
    Let v := w.
    If v = t, ACCEPT

REJECT”

Space needed = ?

PATH = {(G,s,t) : G is a directed graph with a path from s to t }

- Claim: PATH $\in$ NSPACE(10 log n)
- Proof:

M := "On input (G,s,t):

Let v := s.

For i = 0 to |G|
  Guess a neighbor w of v.
  Let v := w.
  If v = t, ACCEPT

REJECT"

Space needed = |v| + |i| = c log |G|. ∎

- By definition SPACE(s(n)) ⊆ NSPACE(?)

- By definition SPACE(s(n)) $\subseteq$ NSPACE(s(n)).

- We showed SPACE(s(n)) $\subseteq$ TIME(?)

- By definition SPACE(s(n)) $\subseteq$ NSPACE(s(n)).

- We showed SPACE(s(n)) $\subseteq$ TIME($2^{c\ s(n)}$), $\forall$ s(n) $\geq$ log n

- Next NSPACE(s(n)) $\subseteq$ TIME(?)

- By definition SPACE(s(n)) $\subseteq$ NSPACE(s(n)).

- We showed SPACE(s(n)) $\subseteq$ TIME($2^{c\ s(n)}$), $\forall$ s(n) $\geq$ log n

- Next NSPACE(s(n)) $\subseteq$ TIME($2^{c\ s(n)}$), $\forall$ s(n) $\geq$ log n

- Claim: $NSPACE(s(n)) \subseteq TIME(2^{c\, s(n)})$, $\forall\, s(n) \geq \log n$
- Proof:

- Claim: NSPACE(s(n)) $\subseteq$ TIME($2^{c\ s(n)}$), $\forall$ s(n) $\geq$ log n

- Proof:

- Let M be a non-deterministic TM using space s(n).

- Define M' :=

  "On input x,

  Compute the configuration graph G of M on input x.

  Nodes = configurations

  Edges = {(c,c') : c yields c' on input x }

  ???

  "

- Claim: NSPACE(s(n)) $\subseteq$ TIME($2^{c\, s(n)}$), $\forall$ s(n) $\geq$ log n

- Proof:

- Let M be a non-deterministic TM using space s(n).

- Define M' :=

  “On input x,

  Compute the configuration graph G of M on input x.

  Nodes = configurations

  Edges = {(c,c') : c yields c' on input x }

  If c_{accept} is reachable from c_{start} in G, ACCEPT

  else REJECT”

- |G| = ?

- Claim: NSPACE(s(n)) $\subseteq$ TIME($2^{c\, s(n)}$), $\forall$ s(n) $\geq$ log n

- Proof:

- Let M be a non-deterministic TM using space s(n).

- Define M' :=
  "On input x,
  Compute the configuration graph G of M on input x.
  Nodes = configurations
  Edges = {(c,c') : c yields c' on input x }

  If c_{accept} is reachable from c_{start} in G, ACCEPT
  else REJECT"

- Because |G| = $c^{s(n)}$ and reachability can be solved in polynomial time, M' runs in time $c^{s(n)}$ ■

P vs. NP for space ?

P vs. NP for space ?

P = NP!

UNLIKE TIME,

SPACE CAN BE REUSED!

Theorem: NSPACE(s(n)) $\subseteq$ SPACE(c s^2 (n)), $\forall$ s(n) $\geq$ log n

This is known as Savitch's theorem

Proof: ?

Theorem: NSPACE(s(n)) $\subseteq$ SPACE($c\, s^2(n)$), $\forall$ $s(n) \geq \log n$

Proof: Let N be a non-deterministic TM using space s(n).

Define M := "On input w,

Return REACH(C_{start}, C_{accept} , $d^{\,s(n)}$)."

- REACH(c, c', t) decides if c' reachable from c in $\leq$ t steps in configuration graph of N on input w

  C_{start} = start configuration
  C_{accept} = accept configuration
  $d^{\,s(n)}$ = number of configurations of N, for a constant d

- Key point is how to implement REACH

REACH(c, c', t) :=   \\ is c' reachable from c in t steps?

"Enumerate all configurations c_m {

If REACH(c,c_m ,t/2) and REACH(c_m ,c', t/2), ACCEPT

}

REJECT"

Define S(t) := space for REACH(c,c',t).

S(t) ≤ ?

REACH(c, c', t) := \\ is c' reachable from c in t steps?

"Enumerate all configurations c_m {

If REACH(c,c_m ,t/2) and REACH(c_m ,c', t/2), ACCEPT

}

REJECT"

Define S(t) := space for REACH(c,c',t).

S(t) ≤ d s(n) + S(t/2). Reuse space for two calls to REACH.

Space for REACH(C_{start}, C_{accept}, d $^{s(n)}$) ≤

?

REACH(c, c', t) :=   \\ is c' reachable from c in t steps?

"Enumerate all configurations c_m {

If REACH(c,c_m ,t/2) and REACH(c_m ,c', t/2), ACCEPT

}

REJECT"

Define S(t) := space for REACH(c,c',t).

$S(t) \le d\ s(n) + S(t/2)$. Reuse space for two calls to REACH.

Space for REACH(C_{start}, C_{accept}, $d^{s(n)}$) ≤

$$d\ s(n) + d\ s(n) + \ldots + d\ s(n) \le d^2 s^2 (n) \qquad \blacksquare$$

- Theorem: NSPACE(s(n)) $\subseteq$ SPACE(c s^2 (n)), $\forall$ s(n) $\geq$ log n
- We just proved this.
- Corollary: NSPACE(log n) $\subseteq$ SPACE(?)

- Theorem: NSPACE(s(n)) $\subseteq$ SPACE(c s^2 (n)), $\forall$ s(n) $\geq$ log n
- We just proved this.
- Corollary: NSPACE(log n) $\subseteq$ SPACE(c $\log^2$ n)
  U_c NSPACE(n^c) = U_c SPACE(?)

- Theorem: $\mathrm{NSPACE}(s(n)) \subseteq \mathrm{SPACE}(c\ s^2(n)),\ \forall\ s(n) \geq \log n$

- We just proved this.

- Corollary: $\mathrm{NSPACE}(\log n) \subseteq \mathrm{SPACE}(c \log^2 n)$

  $\bigcup_c \mathrm{NSPACE}(n^c) = \bigcup_c \mathrm{SPACE}(n^c)$

- Compare with open question for time:

  $\bigcup_c \mathrm{NTIME}(n^c) = \bigcup_c \mathrm{TIME}(n^c)$?

- Is NTIME(t) closed under complement?

- Is NTIME(t) closed under complement?

  Unknown, not believed to be the case.

- Is NSPACE(s) closed under complement?

- Is NTIME(t) closed under complement?

  Unknown, not believed to be the case.

- Is NSPACE(s) closed under complement?

  We just showed NSPACE(s) $\subseteq$?

- Is NTIME(t) closed under complement?

  Unknown, not believed to be the case.

- Is NSPACE(s) closed under complement?

  We just showed NSPACE(s) $\subseteq$ SPACE($c\ s^2$)

  So if L $\in$ NSPACE(s) then not L is in SPACE(?

- Is NTIME(t) closed under complement?

  Unknown, not believed to be the case.

- Is NSPACE(s) closed under complement?

  We just showed NSPACE(s) $\subseteq$ SPACE($c\ s^2$)

  So if L $\in$ NSPACE(s) then not L is in SPACE($c\ s^2$)

- Can you avoid squaring the space?

- Is NTIME(t) closed under complement?

  Unknown, not believed to be the case.

- Is NSPACE(s) closed under complement?

  We just showed NSPACE(s) $\subseteq$ SPACE(c s^2)

  So if L $\in$ NSPACE(s) then not L is in SPACE(c s^2)

- Can you avoid squaring the space?

  Yes! If L $\in$ NSPACE(s) then not L is in SPACE(c s)

  This is weird!

Theorem $\overline{\text{PATH}} \in$ NSPACE(d log n), for a constant d.

Theorem $\overline{\text{PATH}} \in \text{NSPACE}(d \log n)$, for a constant d.

Proof: Want a non-deterministic TM that given G, s, and t accepts $\Leftrightarrow$ there is no path from s to t in G.

Theorem $\overline{\text{PATH}} \in$ NSPACE(d log n), for a constant d.

Proof: Want a non-deterministic TM that given G, s, and t accepts ←→ there is no path from s to t in G.

Suppose TM knows c := number of nodes reachable from s

Key idea: there is no path from s to t ←→ there are c nodes such that ???????

Theorem $\overline{\text{PATH}} \in \text{NSPACE}(d \log n)$, for a constant d.

Proof: Want a non-deterministic TM that given G, s, and t accepts $\Leftrightarrow$ there is no path from s to t in G.

Suppose TM knows c := number of nodes reachable from s

Key idea: there is no path from s to t $\Leftrightarrow$ there are c nodes different from t reachable from s

Define M := " ?

Theorem $\overline{\text{PATH}} \in$ NSPACE(d log n), for a constant d.

Proof: Want a non-deterministic TM that given G, s, and t accepts ⟷ there is no path from s to t in G.

Suppose TM knows c := number of nodes reachable from s

Key idea: there is no path from s to t ⟷ there are c nodes different from t reachable from s

Define M := "On input G, s, t, and c:

```
Initialize Count = 0;
Enumerate over all nodes v ≠ t  {
   Guess a path from s of length n.
   If reach v, Count ++
}
If Count = c  ACCEPT, else REJECT"
```

How to compute c.

Let A_i be the nodes at distance ≤ from s, and let $c_i := |A_i|$.
Note $A_0 = \{s\}$, $c_0 = 1$.

We want $c = c_n$

To compute c_{i+1} from c_i :=

“$c_{i+1} = 0$

Enumerate nodes v (candidate in A_{i+1})

For each v, enumerate over all w nodes in A_i ,

and check if $w \to v$ is an edge. If so, c_{i+1} ++ ;”

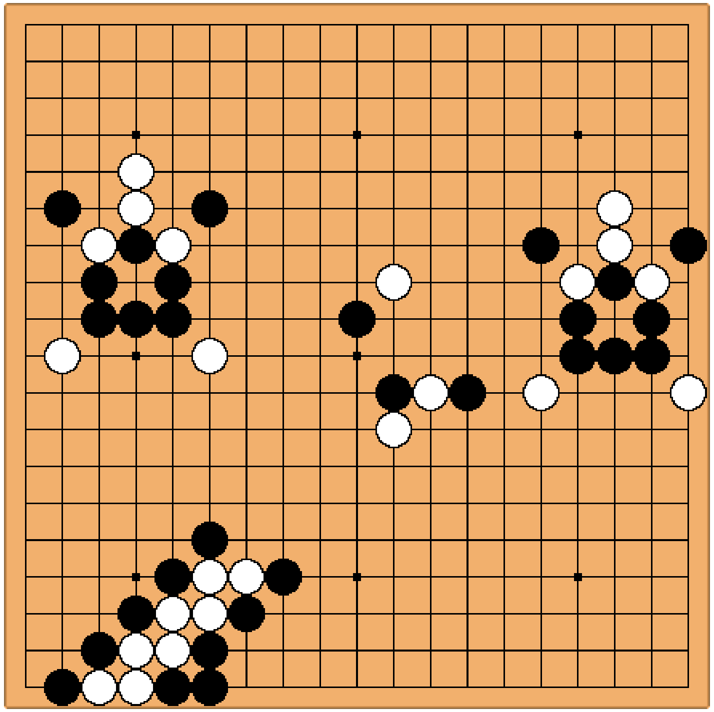
The enumeration over A_i is done guessing c_i nodes and paths from s. If we don't find c_i nodes, we REJECT. ∎

- Next: Two cool things about PSPACE = $\cup_c$ SPACE(n^c)

We saw NP captures videogames, board games, etc.

PSPACE captures 2-player games

For example, given a Go board, how should you move?



We saw NP is a one-message proof system.

We also saw interactive proof systems, and gave such systems for problems not believed to be in NP.

What can interactive proof systems do?

We saw NP is a one-message proof system.

We also saw interactive proof systems, and gave such systems for problems not believed to be in NP.

What can interactive proof systems do?

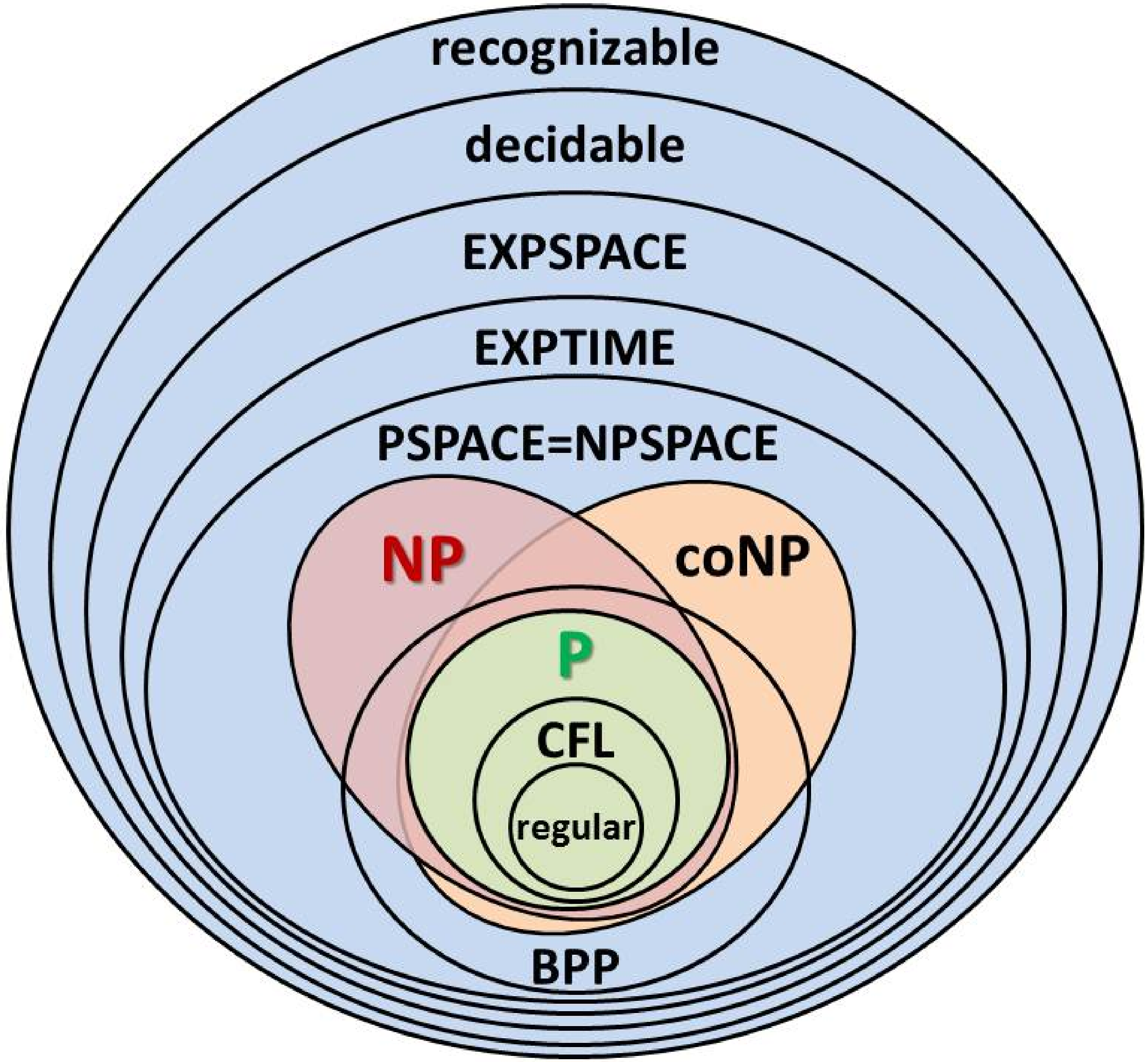
Theorem:

PSPACE = INTERACTIVE PROOF SYSTEMS

In particular,

there is an interactive proof system for playing Go

Summary of some classes we saw



http://www.cse.psu.edu/~sxr48/cmpsc464/

- Omitted slides

PSPACE

SAT: truth of $\exists x_1 \exists x_2 \ldots \exists x_n \varphi (x_1 , x_2 , \ldots , x_n)$
NP-complete

QBF: truth of $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \varphi (x_1, x_2, \ldots, x_n)$, $Q_i \in \{\exists, \forall\}$
PSPACE-complete

Claim: QBF $\in$ PSPACE
Proof:
Exercise

Claim: QBF is PSPACE-hard

Proof: Let M be a PSPACE machine and x an input.

We compute in time poly|x| a QBF formula φ :

φ true

⟺ M accepts x

⟺ c_{accept} reachable from c_{start} in M's configuration graph

$\varphi(c,c')_t$:= is c' reachable from c in ≤ t steps?

= ?

Claim: QBF is PSPACE-hard

Proof: Let M be a PSPACE machine and x an input.

We compute in time poly|x| a QBF formula φ :

φ true

⟺ M accepts x

⟺ c_{accept} reachable from c_{start} in M's configuration graph

$\varphi(c,c')_t$:= is c' reachable from c in ≤ t steps?

$$= \exists\, d : \forall\, (a,b) \in \{(c,d), (d,c')\} : \varphi\, (a,b)_{t/2}$$

$| \varphi(c,c')_t | = ?$

Claim: QBF is PSPACE-hard

Proof: Let M be a PSPACE machine and x an input.

We compute in time poly|x| a QBF formula φ :

φ true

⟺ M accepts x

⟺ c_{accept} reachable from c_{start} in M's configuration graph

$\varphi(c,c')_t$:= is c' reachable from c in ≤ t steps?

$$= \exists\, d : \forall\, (a,b) \in \{(c,d), (d,c')\} : \varphi\, (a,b)_{t/2}$$

$$|\, \varphi(c,c')_t \,| = O(|\text{config}|) + |\, \varphi(c,c')_{t/2} \,|$$

For $t = 2^{poly(n)}$, $|\, \varphi(c_{start},c_{accept})_t \,| = ?$

Claim: QBF is PSPACE-hard

Proof: Let M be a PSPACE machine and x an input.

We compute in time poly|x| a QBF formula φ :

φ true

$\Leftrightarrow$ M accepts x

$\Leftrightarrow$ c_{accept} reachable from c_{start} in M's configuration graph

$\varphi(c,c')_t$:= is c' reachable from c in ≤ t steps?

$$= \exists\, d : \forall\, (a,b) \in \{(c,d), (d,c')\} : \varphi\,(a,b)_{t/2}$$

$$|\ \varphi(c,c')_t\ | = O(|\text{config}|) + |\ \varphi(c,c')_{t/2}\ |$$

For $t = 2^{poly(n)}$, $|\ \varphi(c_{start},c_{accept})_t\ | = |\text{config}| \cdot poly(n) = poly(n)$

■

- Same idea as Savitch's theorem

- Definition:

L := $\cup_c$ SPACE(c log n)

NL := $\cup_c$ NSPACE(c log n)

PSPACE := $\cup_c$ SPACE(n^c)

NPSPACE := $\cup_c$ NSPACE(n^c)

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE$$

• Space hierarchy theorem

$\forall$ functions f, g : $f(n) = o(g(n))$,
SPACE(f(n)) strictly contained in SPACE(g(n))

So L ≠ PSPACE

Def. A function $f : \{0,1\}^* \to \{0,1\}^*$ is computable in SPACE(s(n)) if the function
$f'(x,i) : \{0,1\}^* \to \{0,1\}$, $f'(x,i) := f(x)_i$
is in SPACE(s(n)).

Exercise:
Consider the alternative definition where TM are equipped with a write-only tape, that does not count towards space, where TM is supposed to write f(x).
Show the two definitions are equivalent when, say, $|f(x)| = \text{poly}|x|$, $s(n) = O(\log n)$.

- What problem is NP-complete?

  3SAT

- What problem is NSPACE(c log(n))-complete?

  PATH

- Theorem:

PATH $\in$ SPACE(c log n) ➔ NSPACE(log n) = SPACE(c log n)

- Proof:

?

- Theorem:

PATH $\in$ SPACE(c log n) ➔ NSPACE(log n) = SPACE(c log n)

- Proof:

Let N be a non-deterministic TM using space log n.

Let G be the graph where the nodes are configurations of N, and node C is connected to C' if C yields C'.

Note: $|G| \leq$?

- Theorem:

PATH $\in$ SPACE(c log n) $\rightarrow$ NSPACE(log n) = SPACE(c log n)

- Proof:

Let N be a non-deterministic TM using space log n.

Let G be the graph where the nodes are configurations of N, and node C is connected to C' if C yields C'.

Note: $|G| \leq n^d$ for some constant d

Define TM M := "On input w
Run TM for PATH on ?

- Theorem:

PATH $\in$ SPACE(c log n) ➔ NSPACE(log n) = SPACE(c log n)

- Proof:

Let N be a non-deterministic TM using space log n.
Let G be the graph where the nodes are configurations of N, and node C is connected to C' if C yields C'.

Note: |G| ≤ n^d for some constant d

Define TM M := "On input w
Run TM for PATH on (G, C_{start}, C_{accept})
Return the answer" ■

- Detail: M cannot write down G. Instead, when TM for PATH needs an edge, M will compute in on the fly.