Computational Complexity

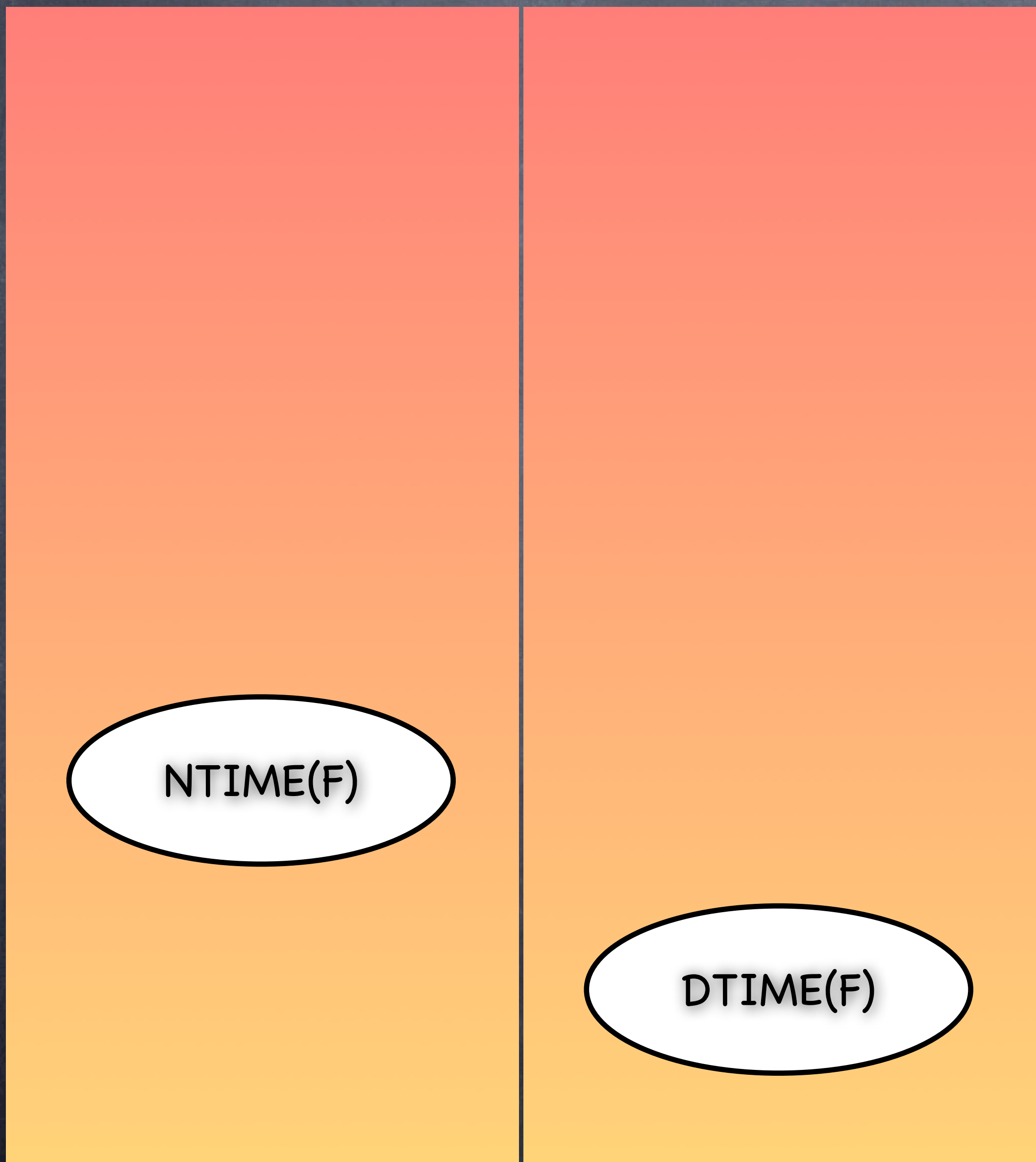
Lecture 5

in which we relate space and time,

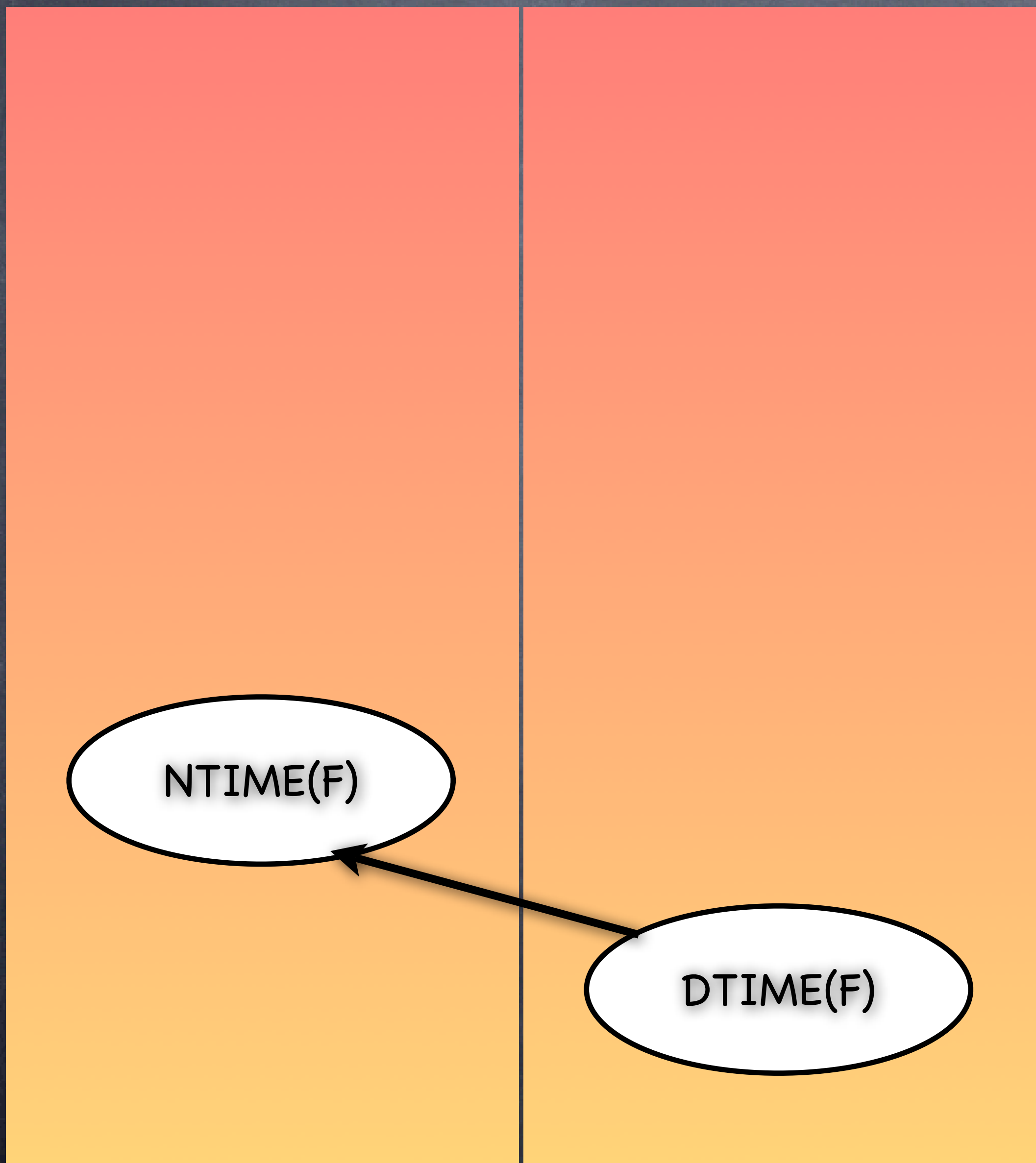
and see the essence of PSPACE (TQBF)

SPACE and TIME

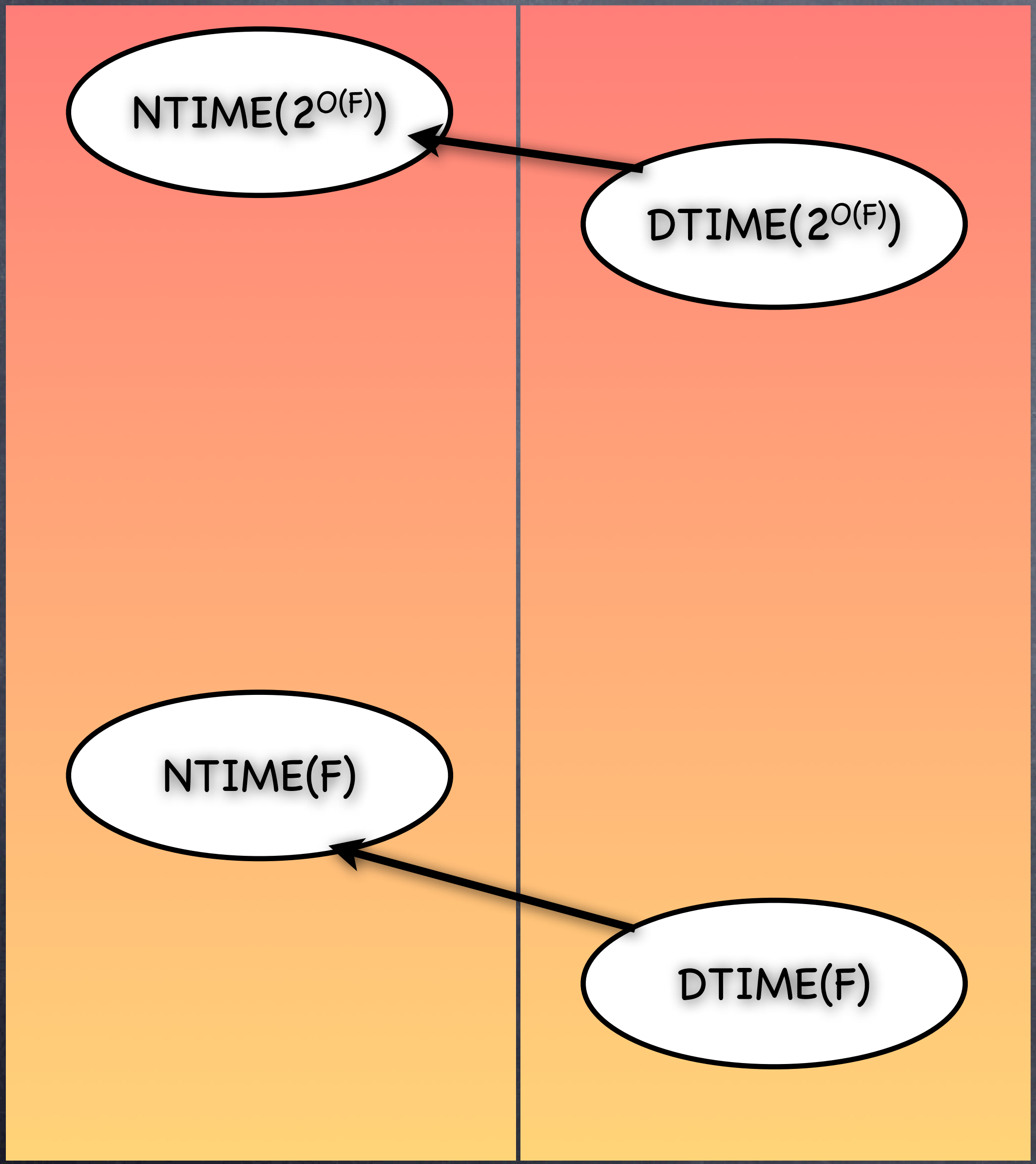
SPACE and TIME



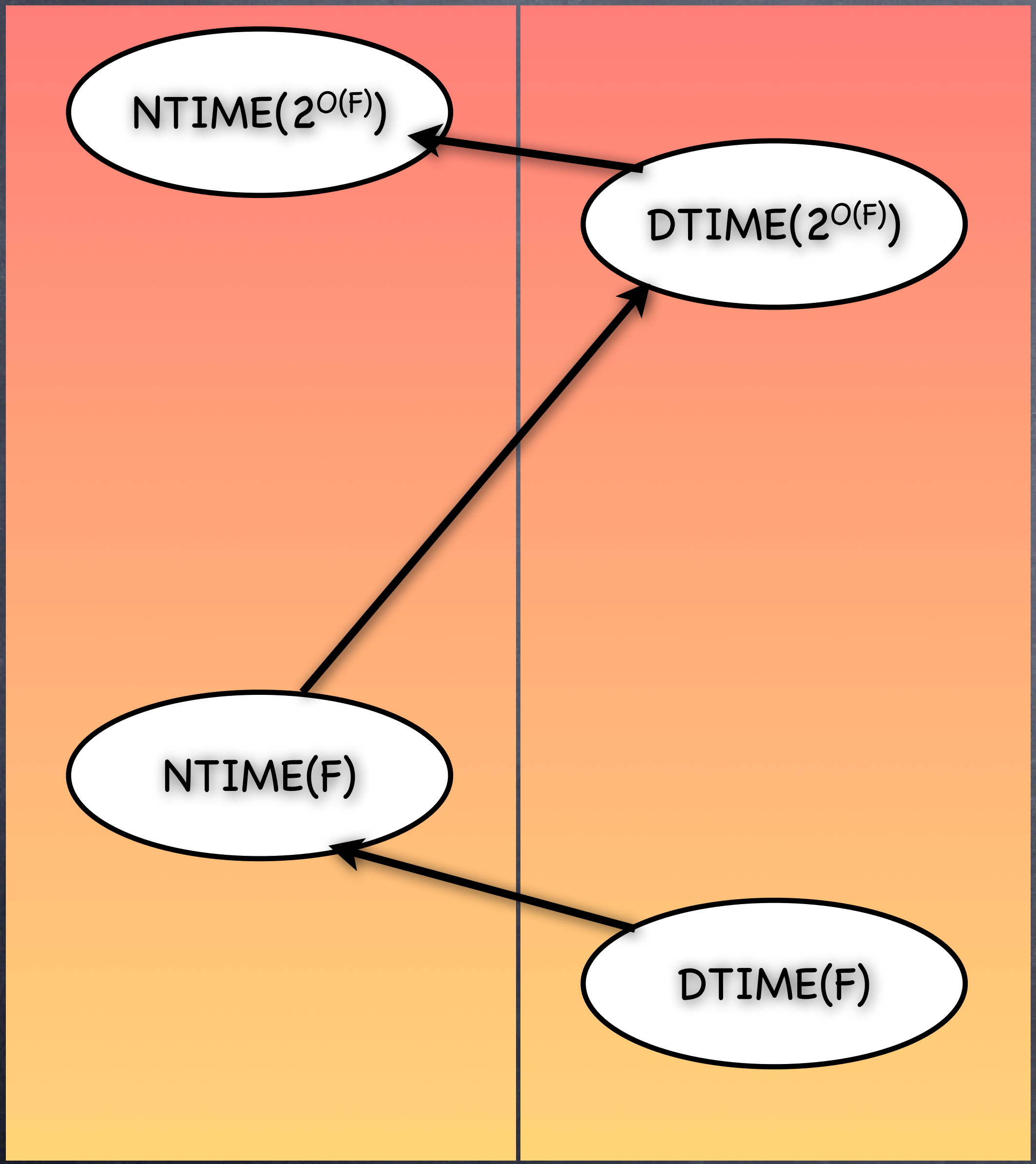
SPACE and TIME



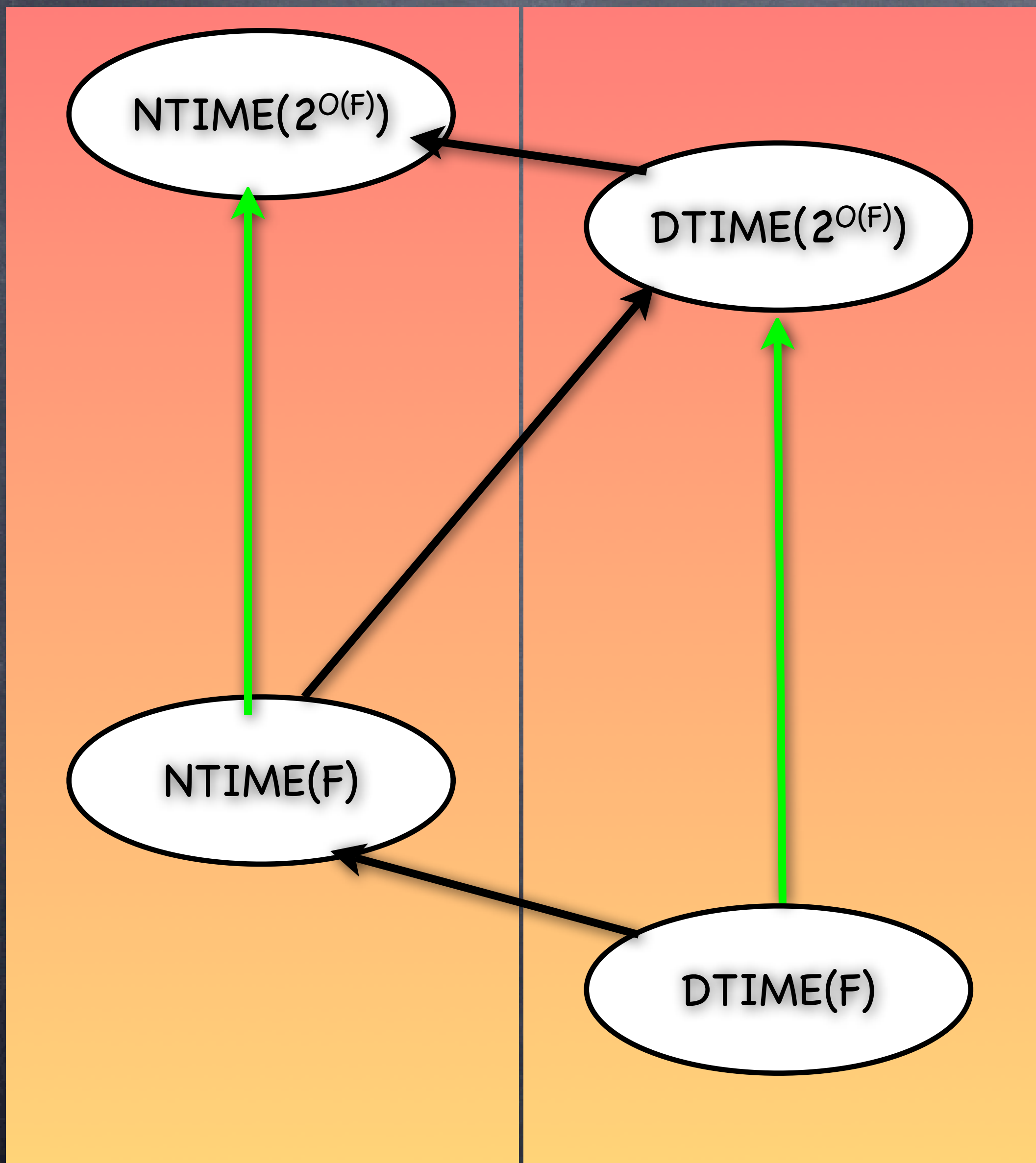
SPACE and TIME



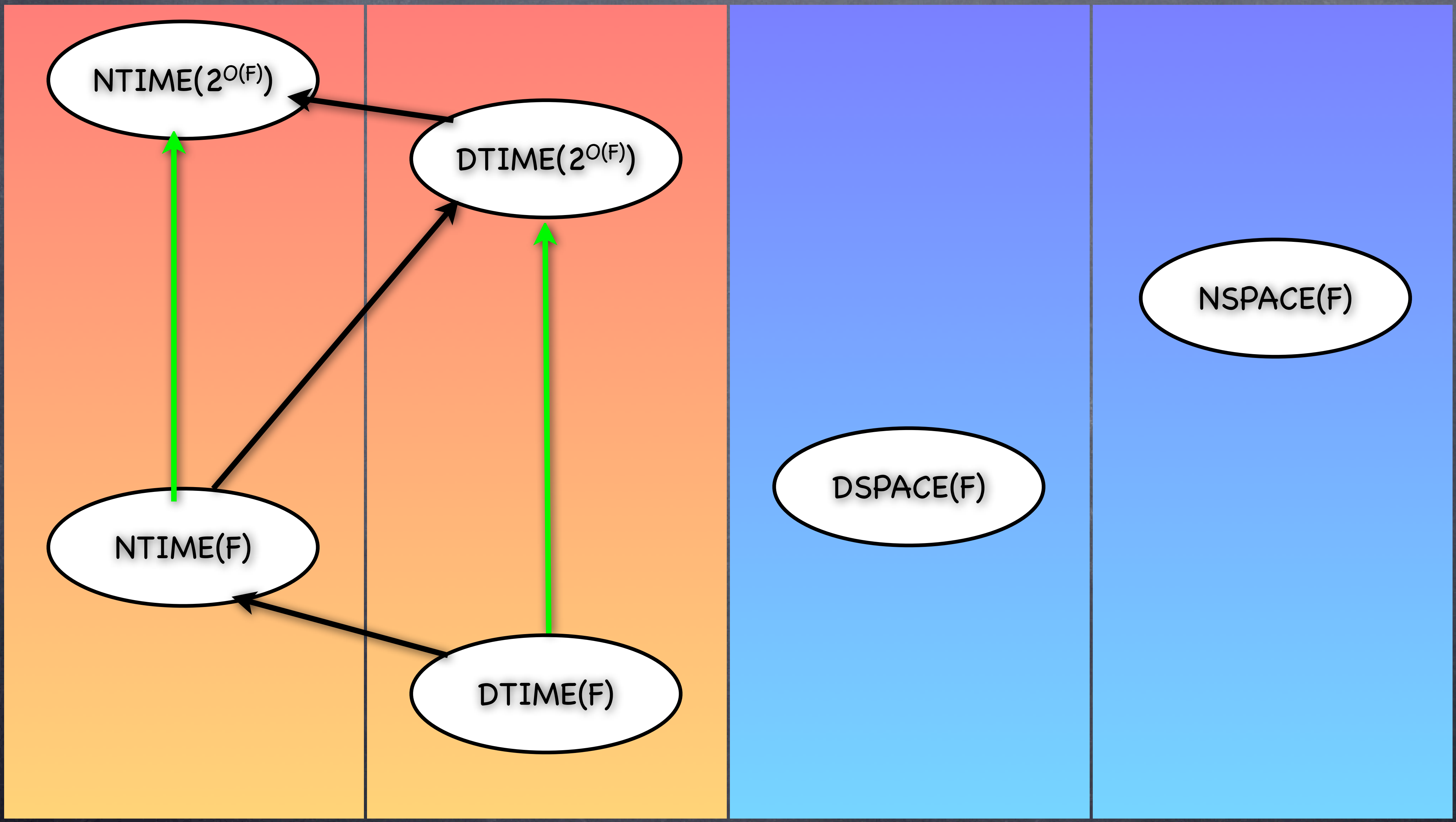
SPACE and TIME



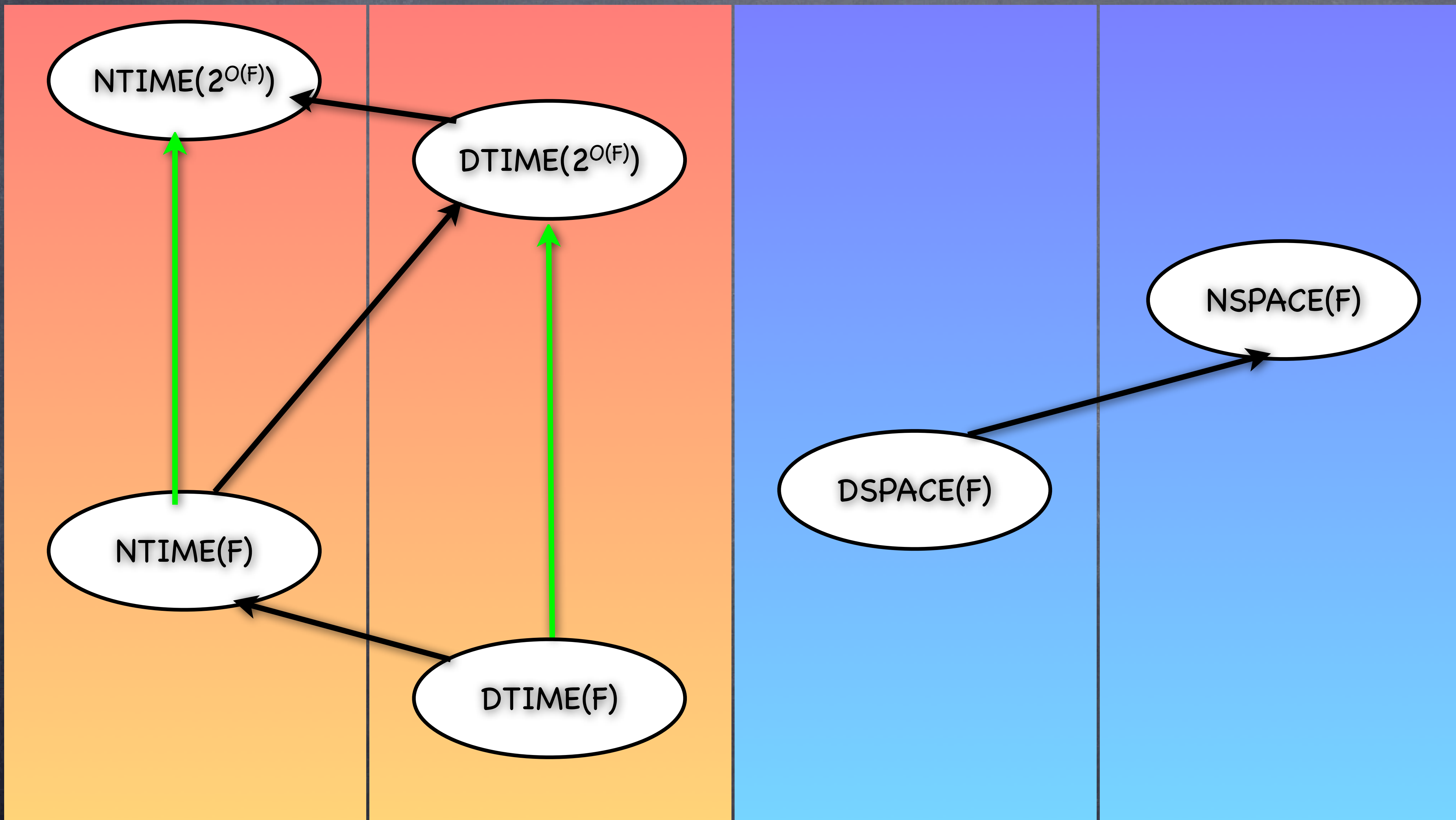
SPACE and TIME



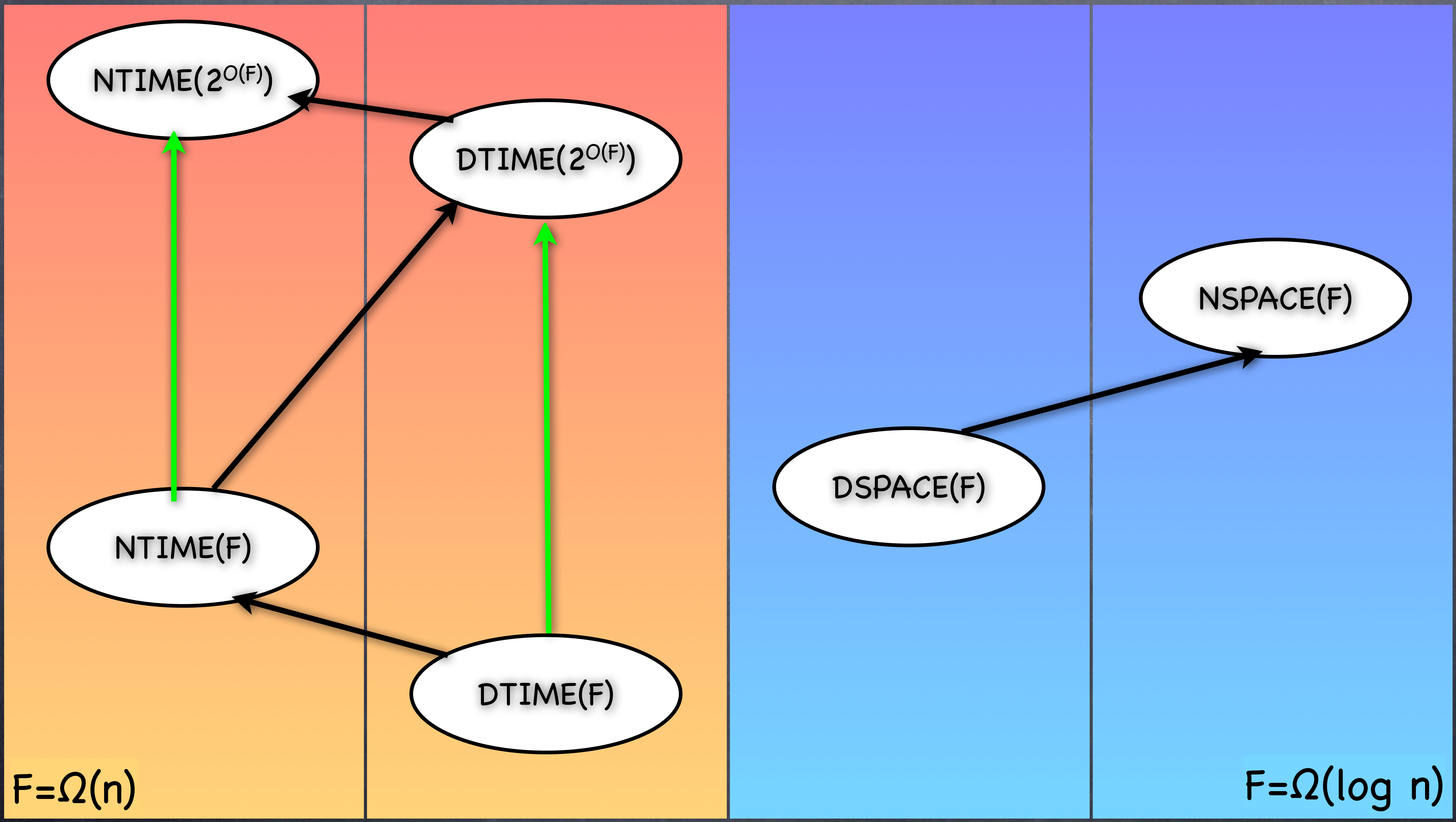
SPACE and TIME



SPACE and TIME



SPACE and TIME



SPACE and TIME

SPACE and TIME

- In time $T(n)$, can use at most $T(n)$ space

SPACE and TIME

- In time T(n), can use at most T(n) space
  - DTIME(T) ⊆ DSPACE(T)

SPACE and TIME

- In time T(n), can use at most T(n) space
  - DTIME(T) ⊆ DSPACE(T)
  - In fact, NTIME(T) ⊆ DSPACE(O(T)) (try all certificates of length at most T, one after the other)

SPACE and TIME

- In time T(n), can use at most T(n) space
  - DTIME(T) ⊆ DSPACE(T)
  - In fact, NTIME(T) ⊆ DSPACE(O(T)) (try all certificates of length at most T, one after the other)
- With space S(n), only $2^{O(S(n))}$ configurations (for S(n) = Ω(log n)). So can take at most $2^{O(S(n))}$ time (else gets into an infinite loop)

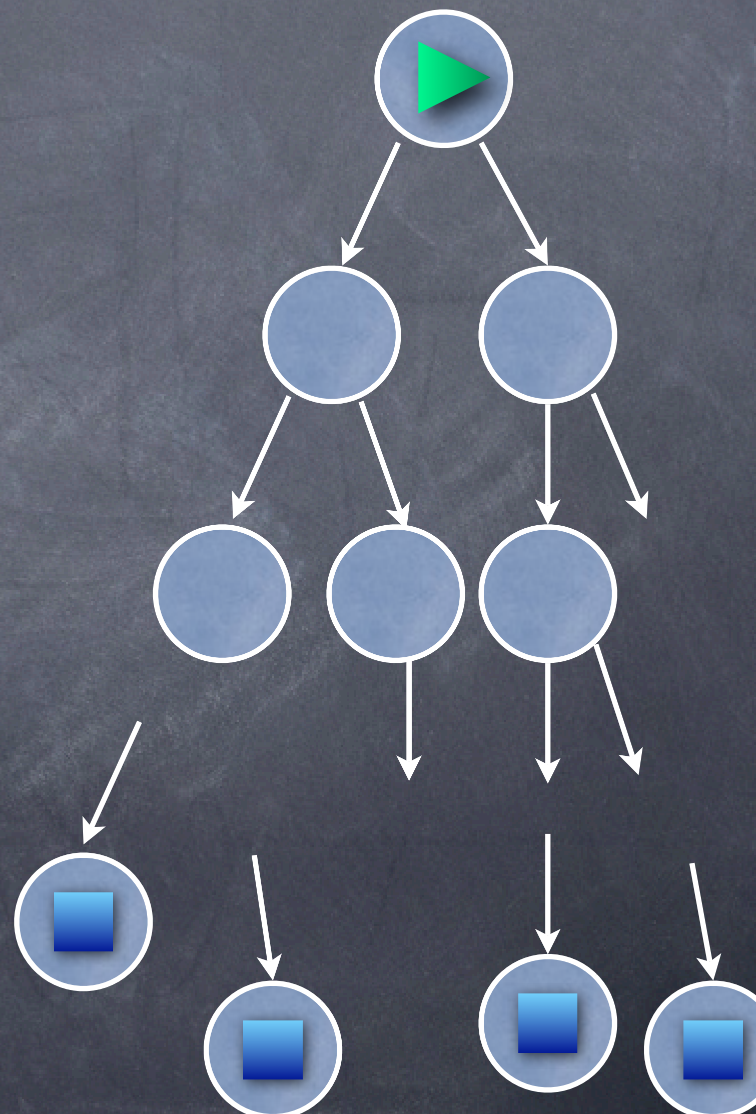
SPACE and TIME

- In time $T(n)$, can use at most $T(n)$ space
  - $\mathrm{DTIME}(T) \subseteq \mathrm{DSPACE}(T)$
  - In fact, $\mathrm{NTIME}(T) \subseteq \mathrm{DSPACE}(O(T))$ (try all certificates of length at most T, one after the other)
- With space $S(n)$, only $2^{O(S(n))}$ configurations (for $S(n) = \Omega(\log n)$). So can take at most $2^{O(S(n))}$ time (else gets into an infinite loop)
  - $\mathrm{DSPACE}(S) \subseteq \mathrm{DTIME}(2^{O(S)})$

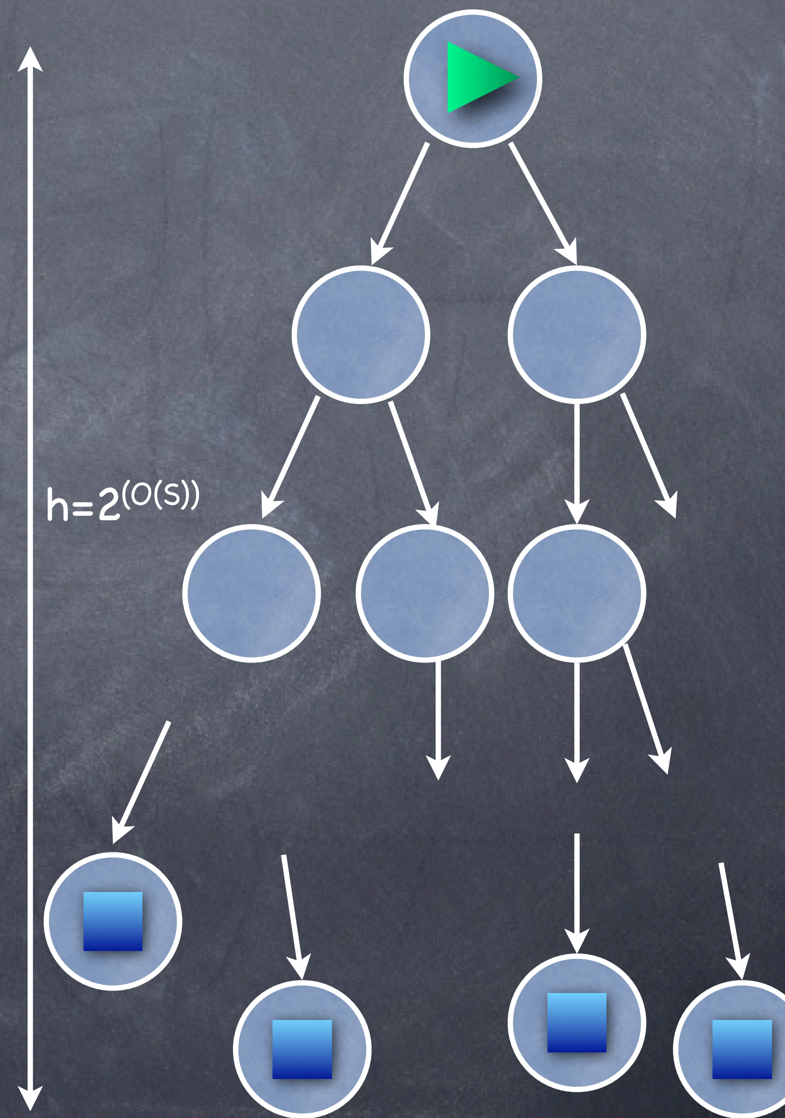
SPACE and TIME

- In time T(n), can use at most T(n) space
  - DTIME(T) ⊆ DSPACE(T)
  - In fact, NTIME(T) ⊆ DSPACE(O(T)) (try all certificates of length at most T, one after the other)
- With space S(n), only $2^{O(S(n))}$ configurations (for S(n) = Ω(log n)). So can take at most $2^{O(S(n))}$ time (else gets into an infinite loop)
  - DSPACE(S) ⊆ DTIME($2^{O(S)}$)
  - In fact, NSPACE(S) ⊆ DTIME($2^{O(S)}$)

$\mathrm{NSPACE}(S) \subseteq \mathrm{DTIME}(2^{O(S)})$

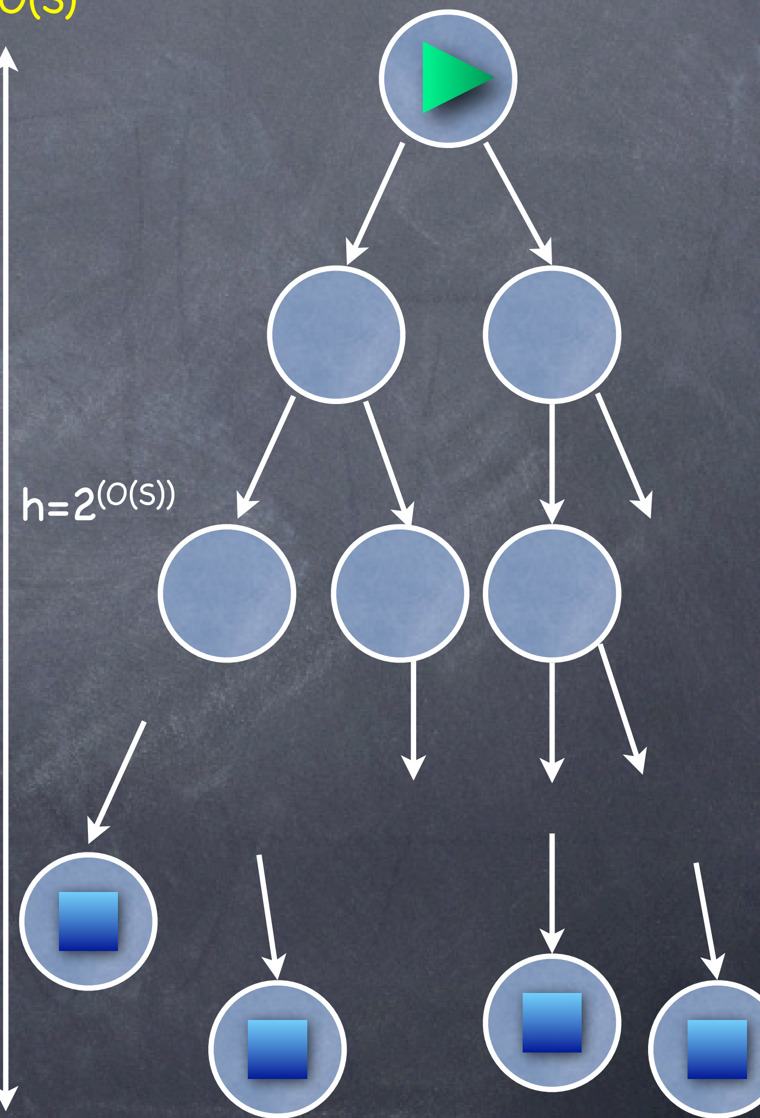


$\mathrm{NSPACE}(S) \subseteq \mathrm{DTIME}(2^{O(S)})$



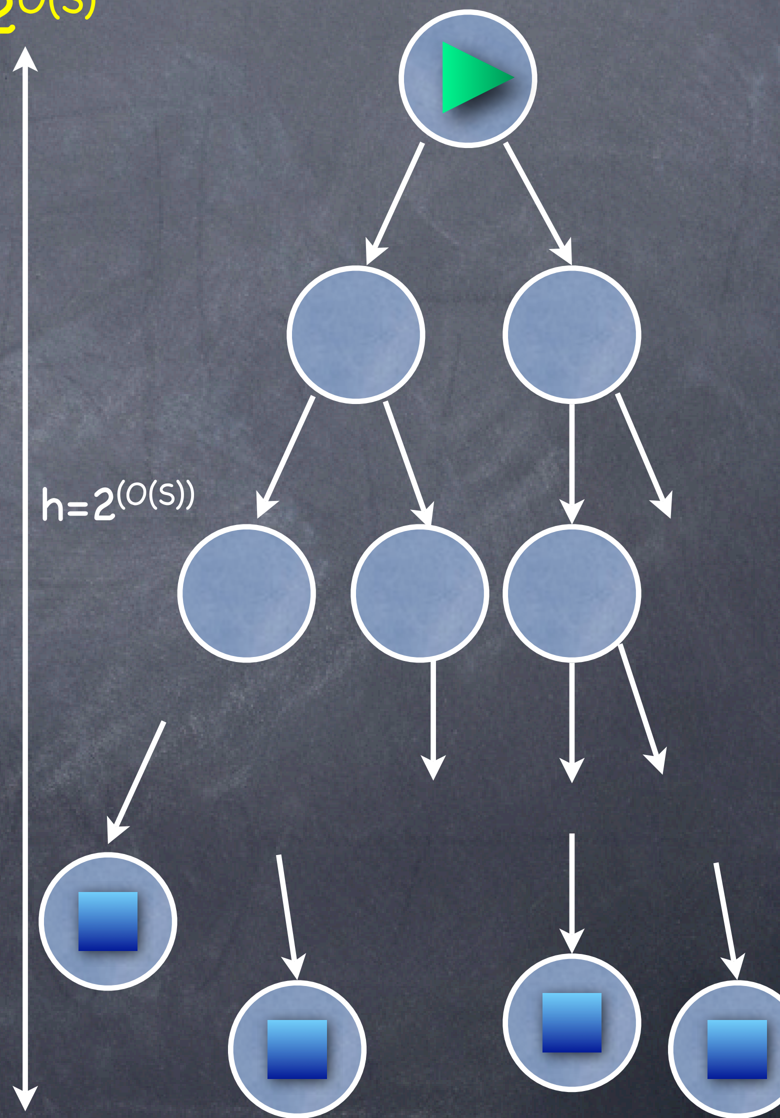
NSPACE(S) $\subseteq$ DTIME($2^{O(S)}$)

- Configuration graph as a DAG is of size $2^{O(S)}$



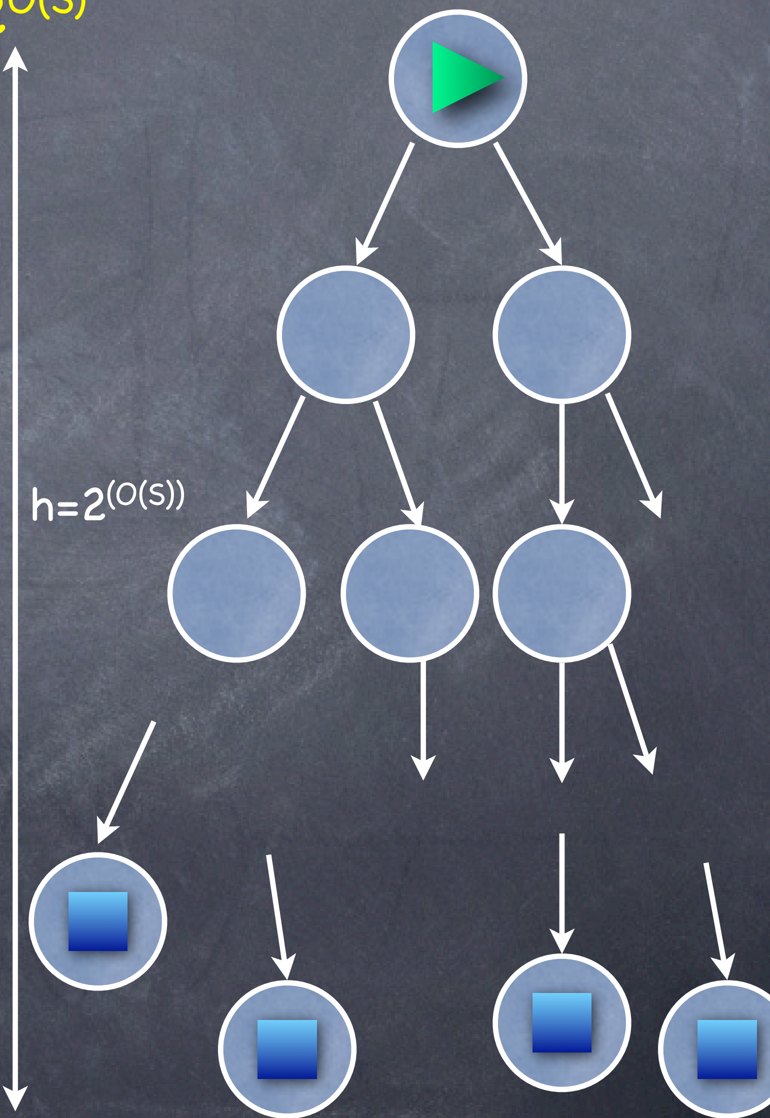
NSPACE(S) $\subseteq$ DTIME($2^{O(S)}$)

- Configuration graph as a DAG is of size $2^{O(S)}$
  - Write down all configurations and edges



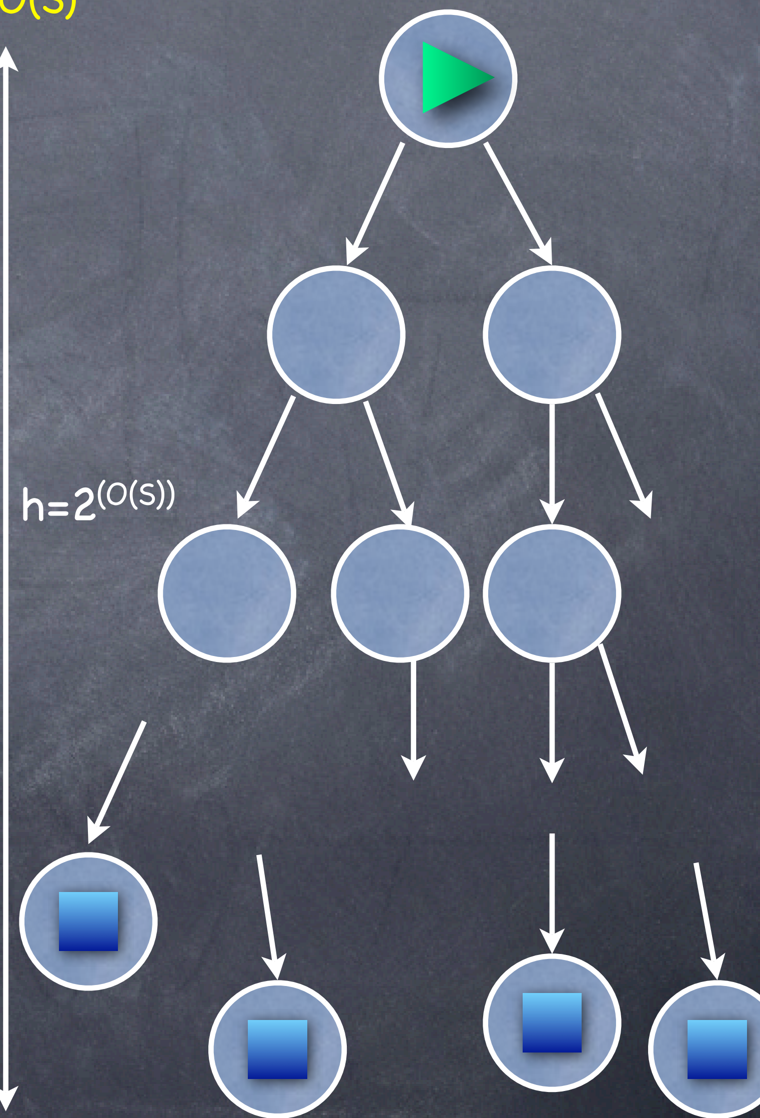
NSPACE(S) $\subseteq$ DTIME($2^{O(S)}$)

- Configuration graph as a DAG is of size $2^{O(S)}$
  - Write down all configurations and edges
    - Can do it less explicitly if space were a concern (but it's not, here)



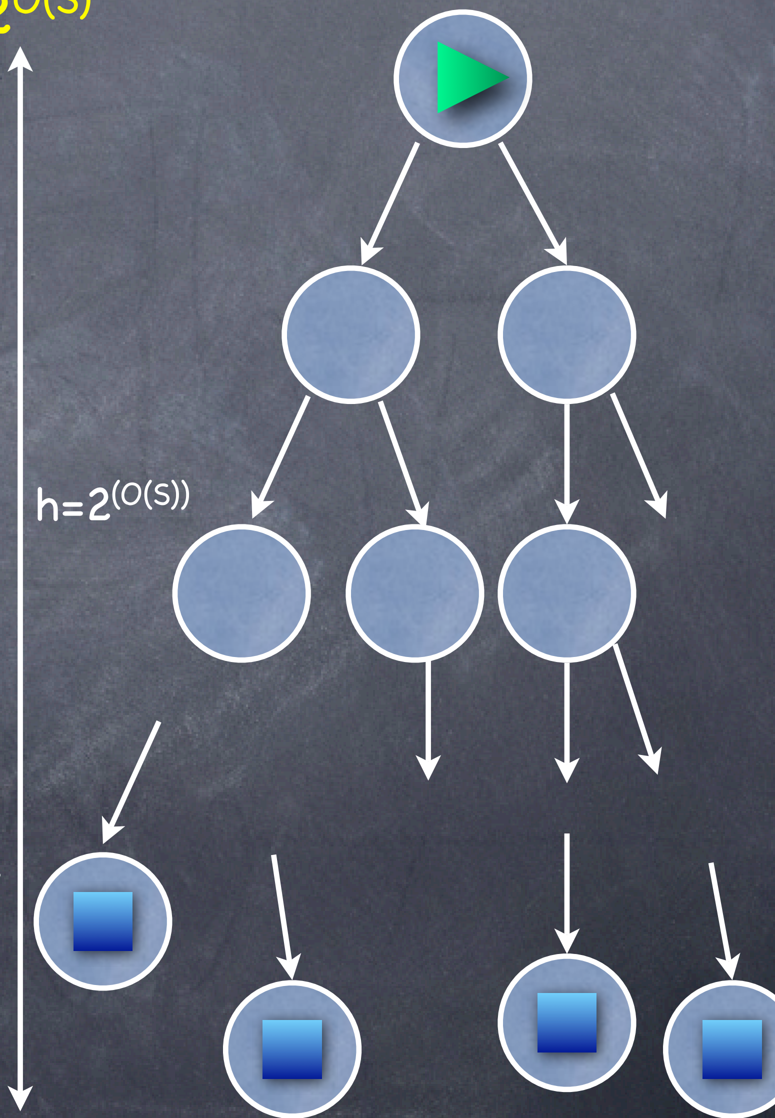
$NSPACE(S) \subseteq DTIME(2^{O(S)})$

- Configuration graph as a DAG is of size $2^{O(S)}$
  - Write down all configurations and edges
    - Can do it less explicitly if space were a concern (but it's not, here)
  - Run (in poly time) any reachability algorithm (say, breadth-first search) to see if there is a (directed) path from start config. to an accept config.

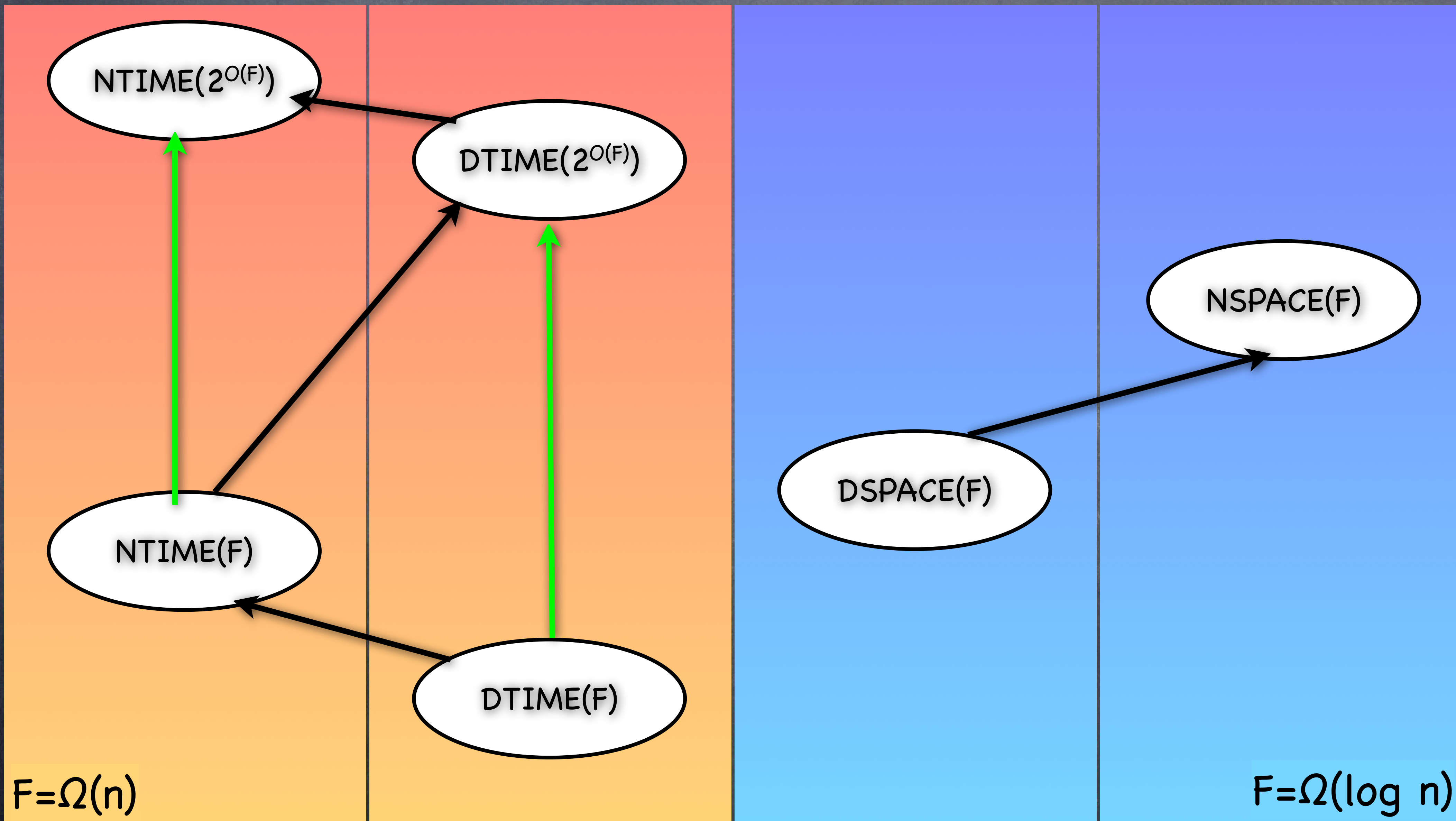


NSPACE(S) $\subseteq$ DTIME($2^{O(S)}$)

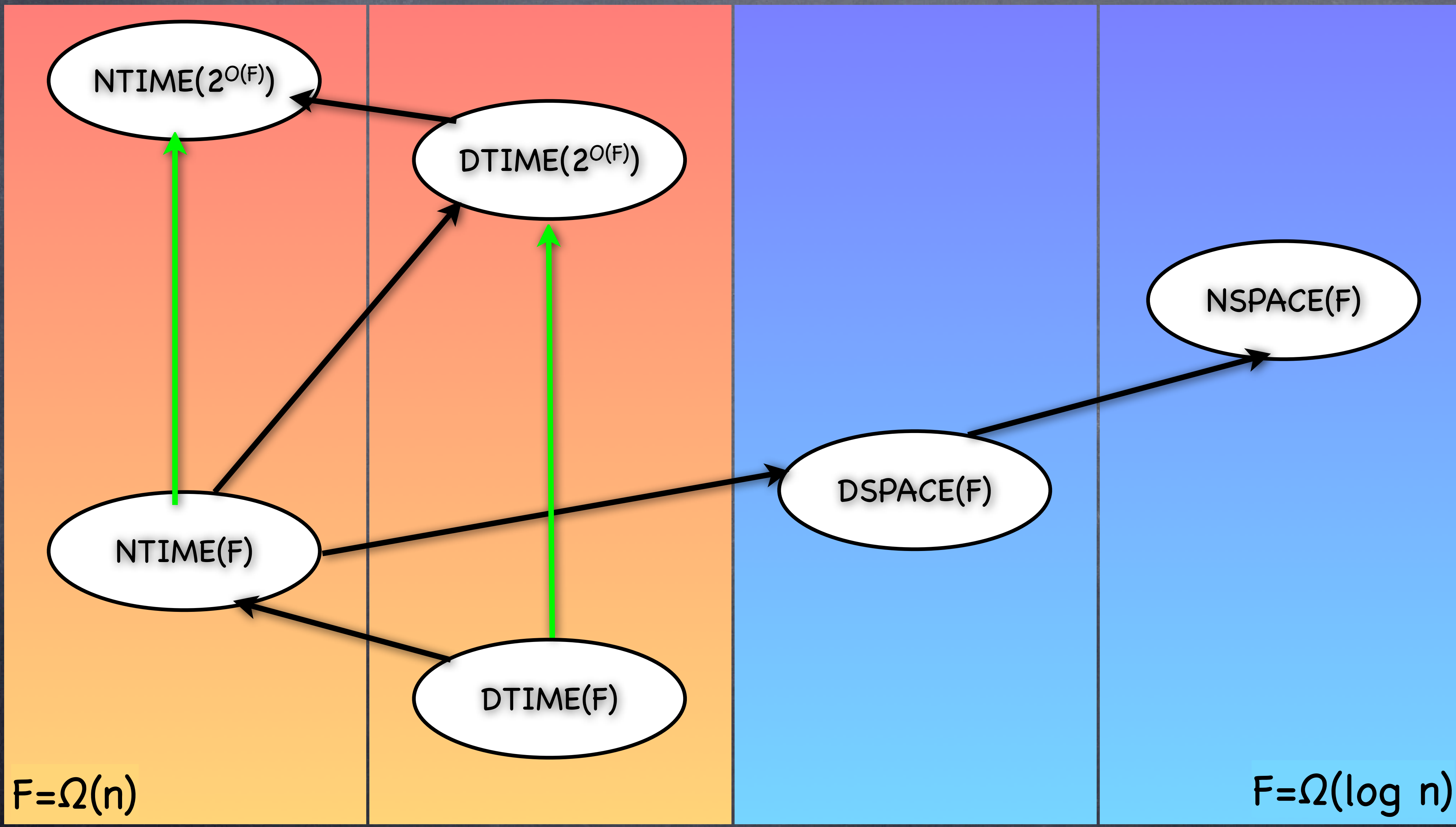
- Configuration graph as a DAG is of size $2^{O(S)}$
  - Write down all configurations and edges
    - Can do it less explicitly if space were a concern (but it's not, here)
  - Run (in poly time) any reachability algorithm (say, breadth-first search) to see if there is a (directed) path from start config. to an accept config.
    - $\mathrm{poly}(2^{O(S)}) = 2^{O(S)}$



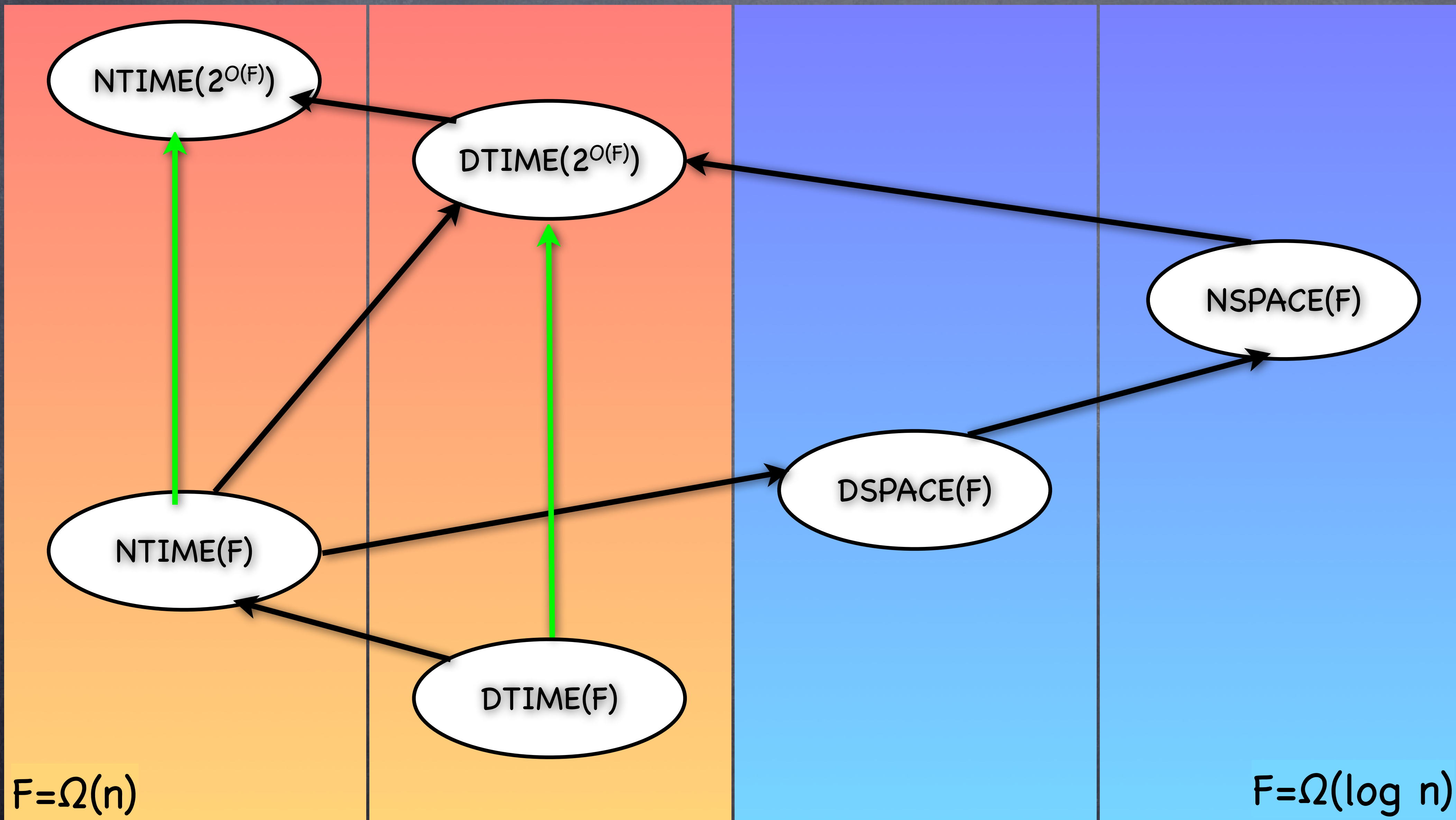
SPACE and TIME



SPACE and TIME



SPACE and TIME



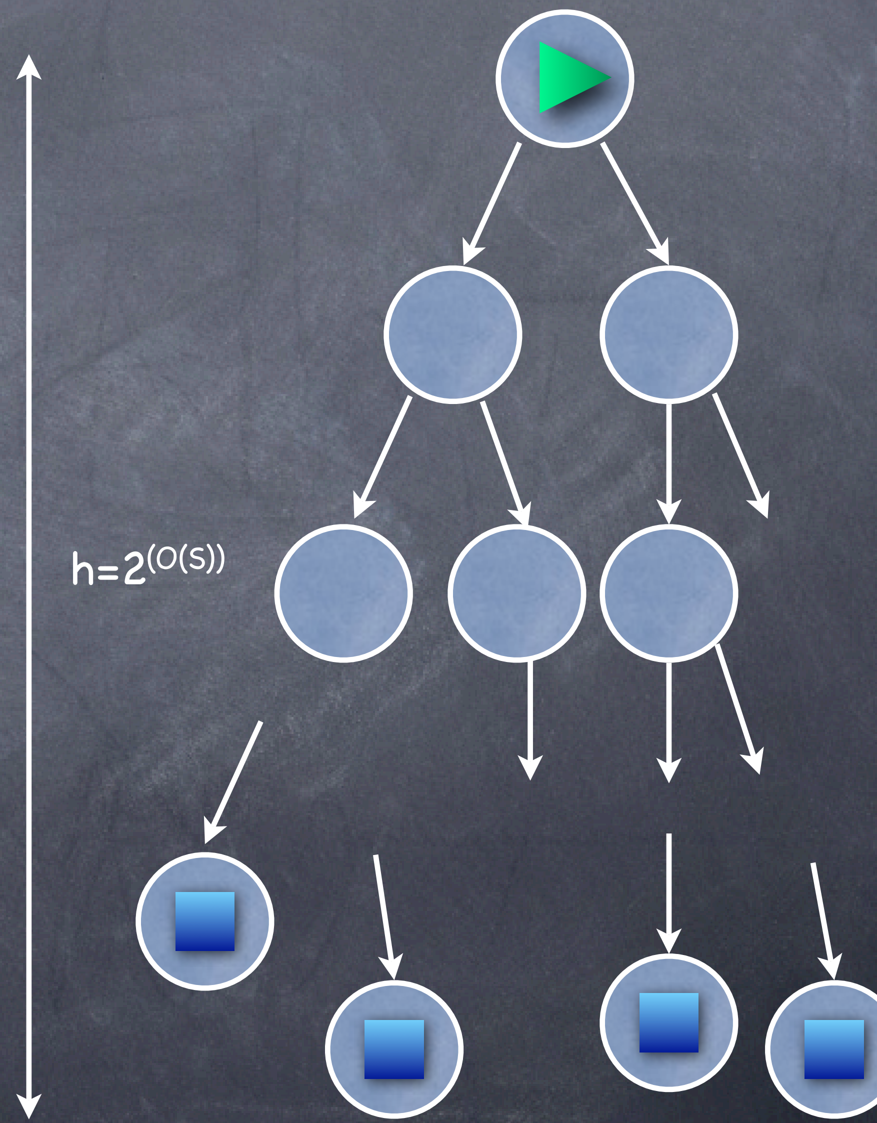
NSPACE and DSPACE: Savitch's Theorem

NSPACE and DSPACE: Savitch's Theorem

- NSPACE(S) $\subseteq$ DSPACE(S^2)

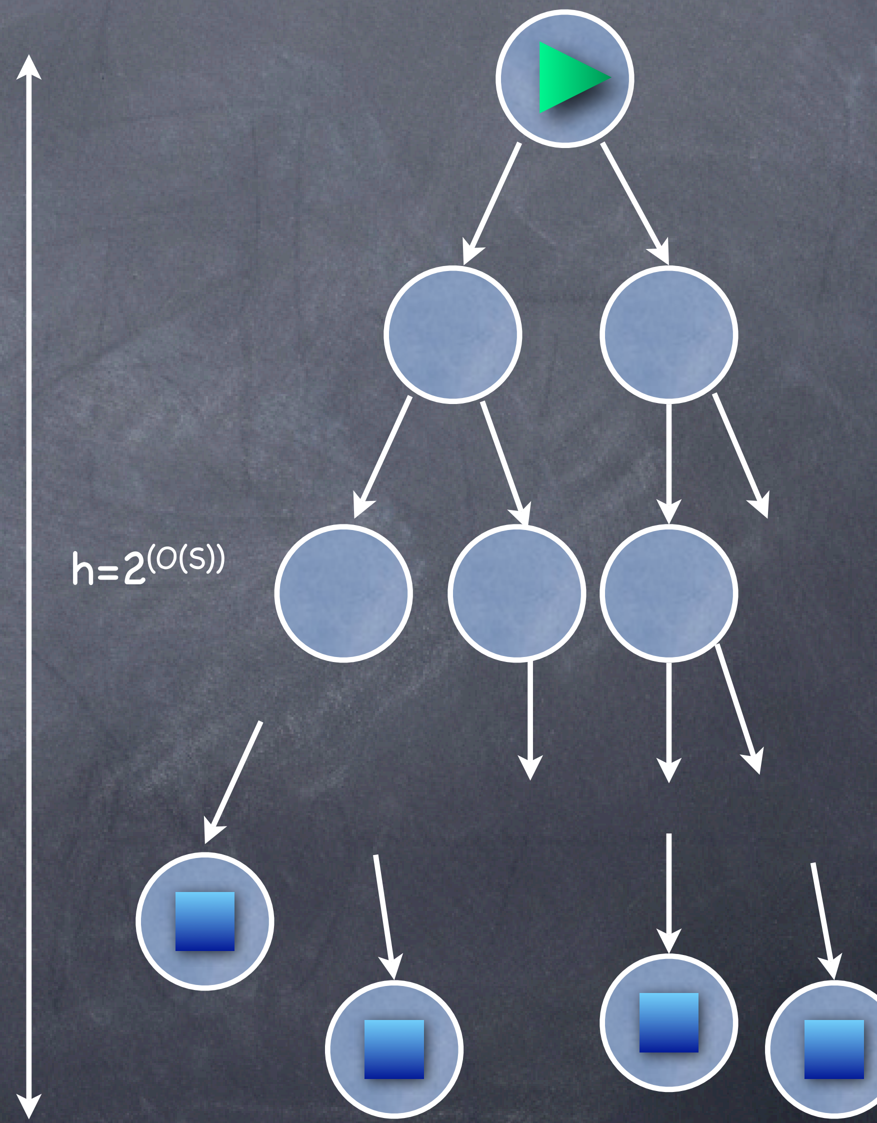
NSPACE and DSPACE: Savitch's Theorem

- NSPACE(S) ⊆ DSPACE(S^2)



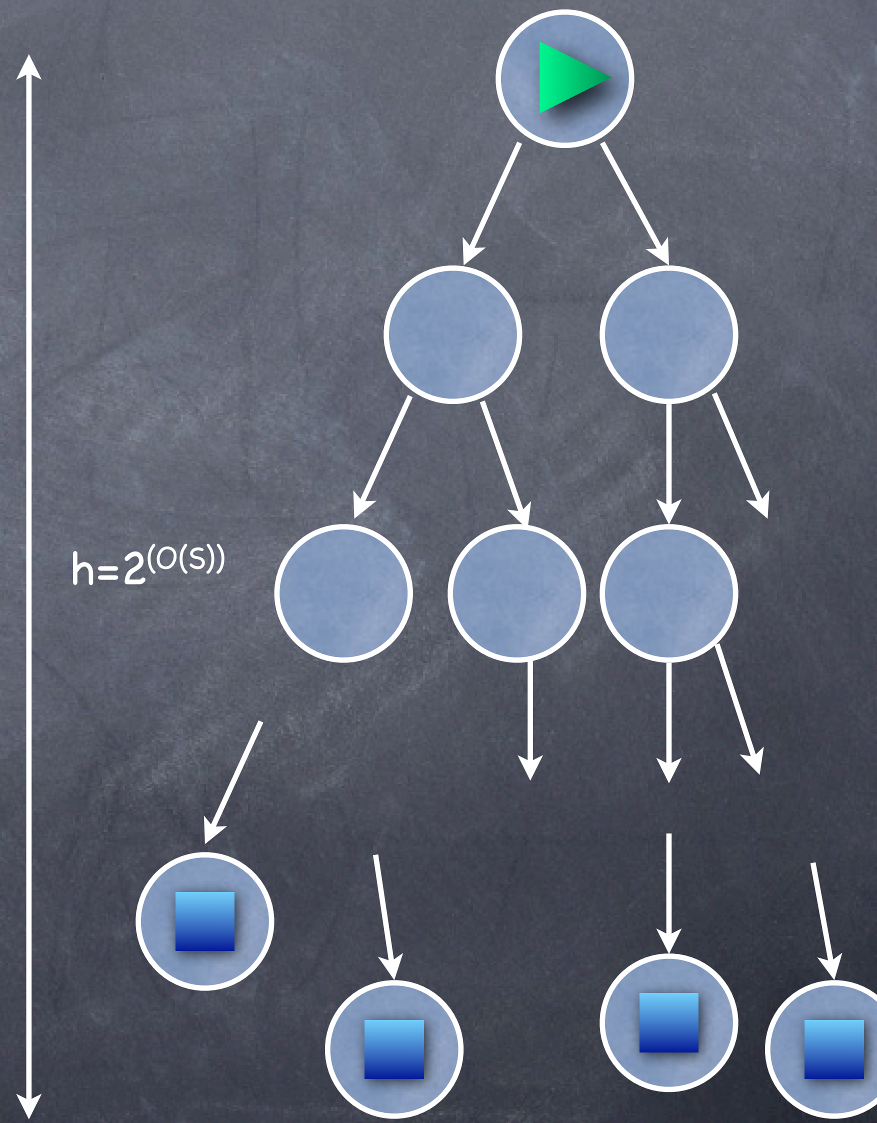
NSPACE and DSPACE: Savitch's Theorem

- NSPACE(S) ⊆ DSPACE(S^2)
  - Naive DFS (or BFS) has stack depth $h=2^{(O(S))}$



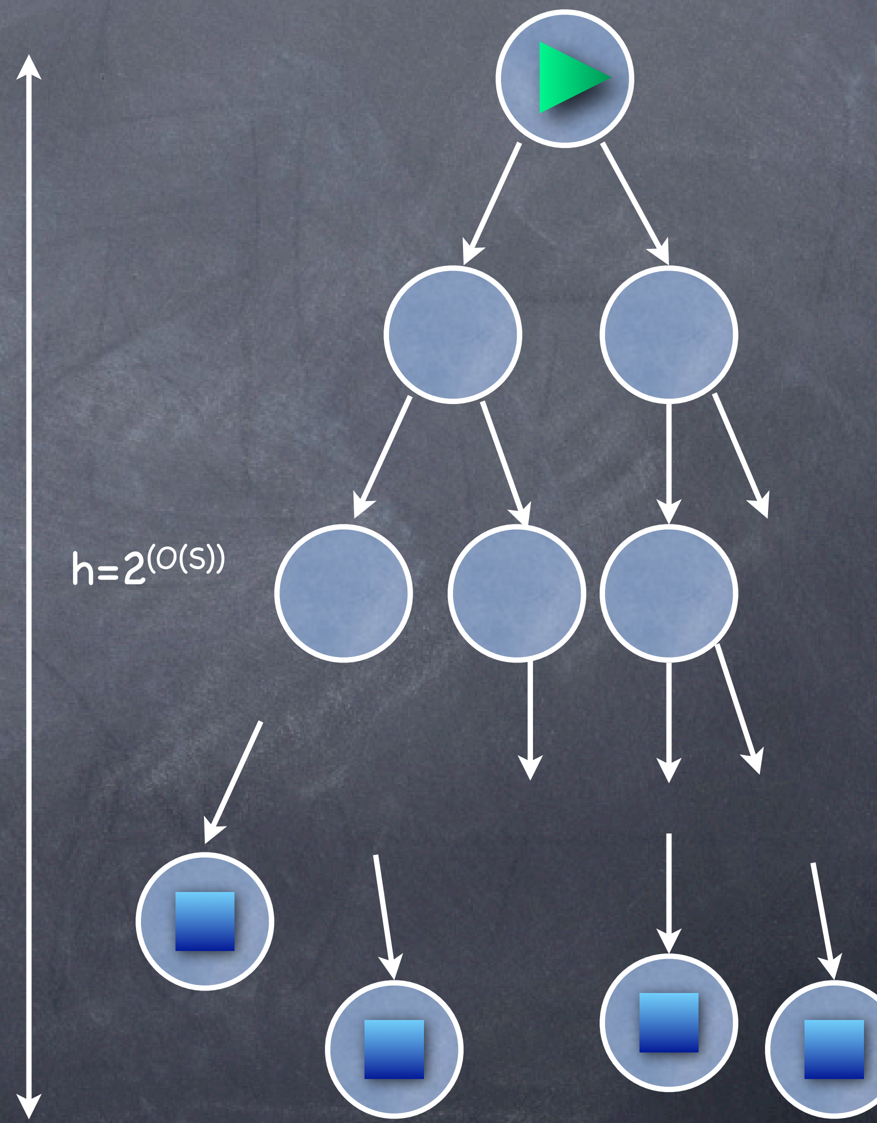
NSPACE and DSPACE: Savitch's Theorem

- NSPACE(S) ⊆ DSPACE(S^2)
  - Naive DFS (or BFS) has stack depth $h=2^{(O(S))}$
  - Look for C s.t. Start→C in h/2 steps and C→Accept in h/2 steps



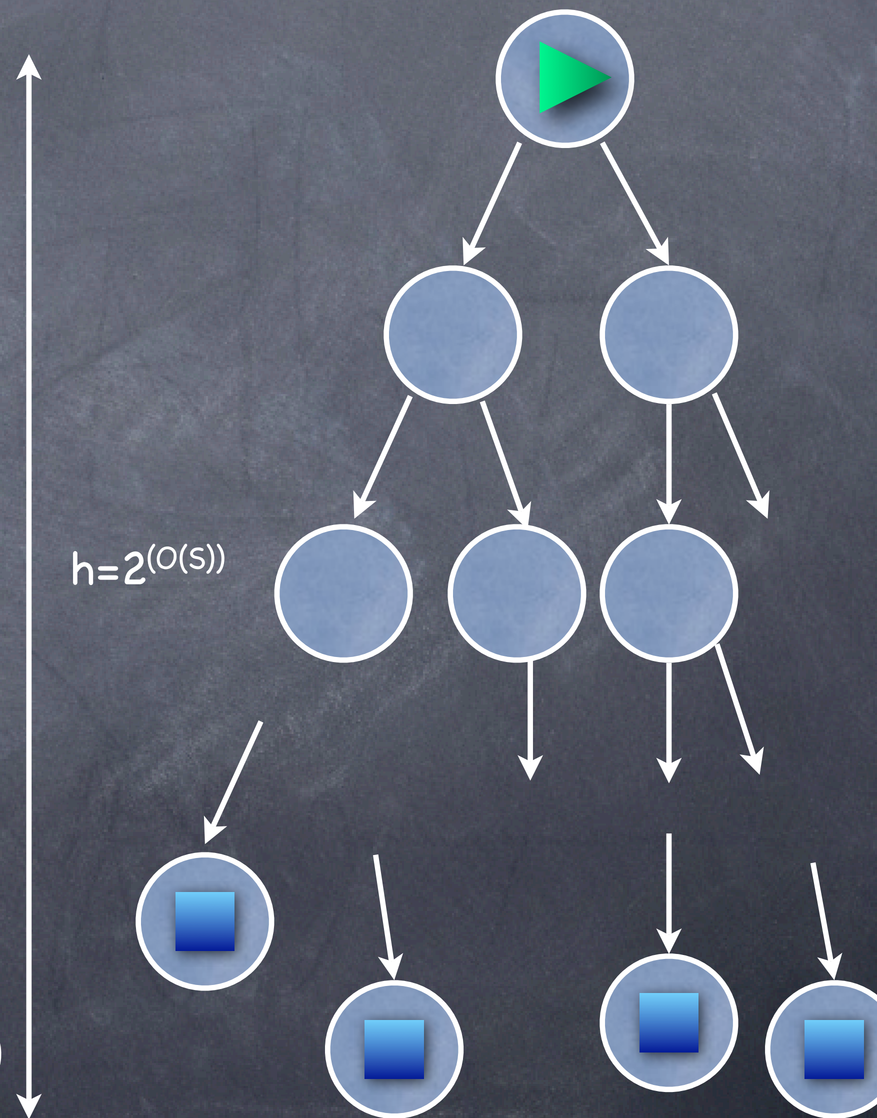
NSPACE and DSPACE: Savitch's Theorem

- NSPACE(S) ⊆ DSPACE(S^2)
  - Naive DFS (or BFS) has stack depth $h=2^{(O(S))}$
  - Look for C s.t. Start→C in h/2 steps and C→Accept in h/2 steps
  - Recursively! Depth of recursion only log h; at each level remember one configuration

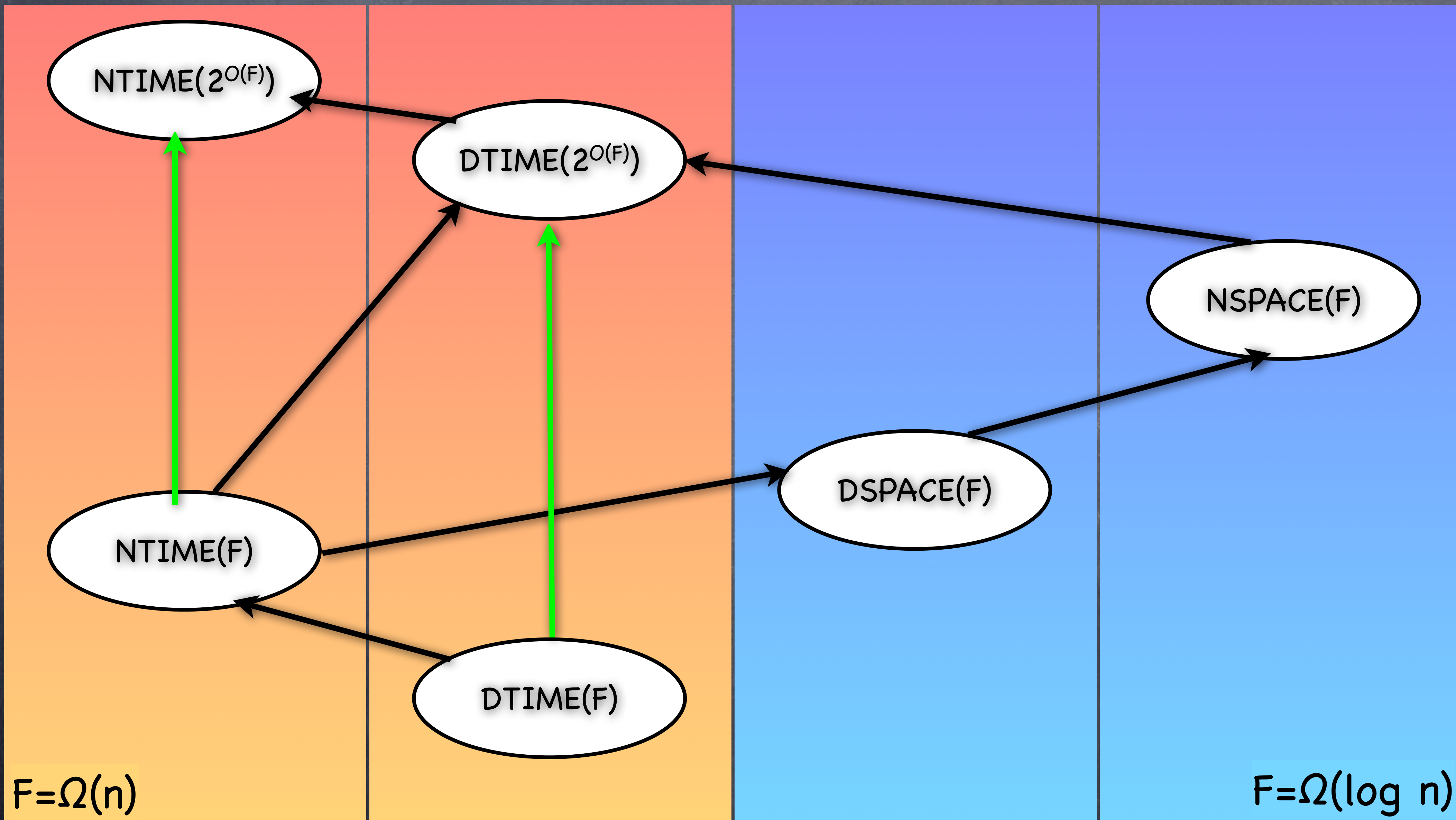


NSPACE and DSPACE: Savitch's Theorem

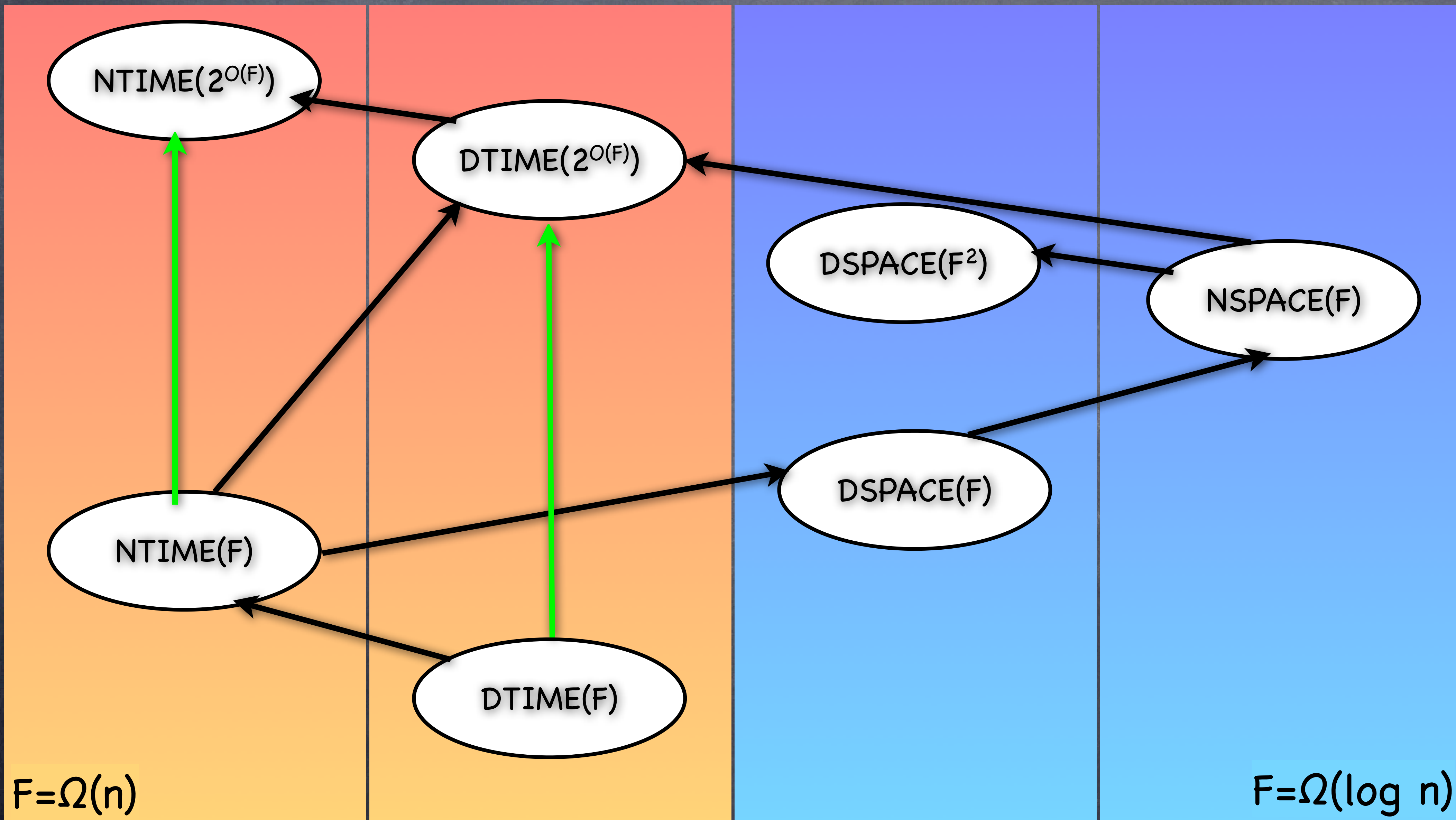
- NSPACE(S) ⊆ DSPACE(S^2)
  - Naive DFS (or BFS) has stack depth $h=2^{(O(S))}$
  - Look for C s.t. Start→C in h/2 steps and C→Accept in h/2 steps
  - Recursively! Depth of recursion only log h; at each level remember one configuration
  - Space needed = O(log h)*O(S) = O(S^2)



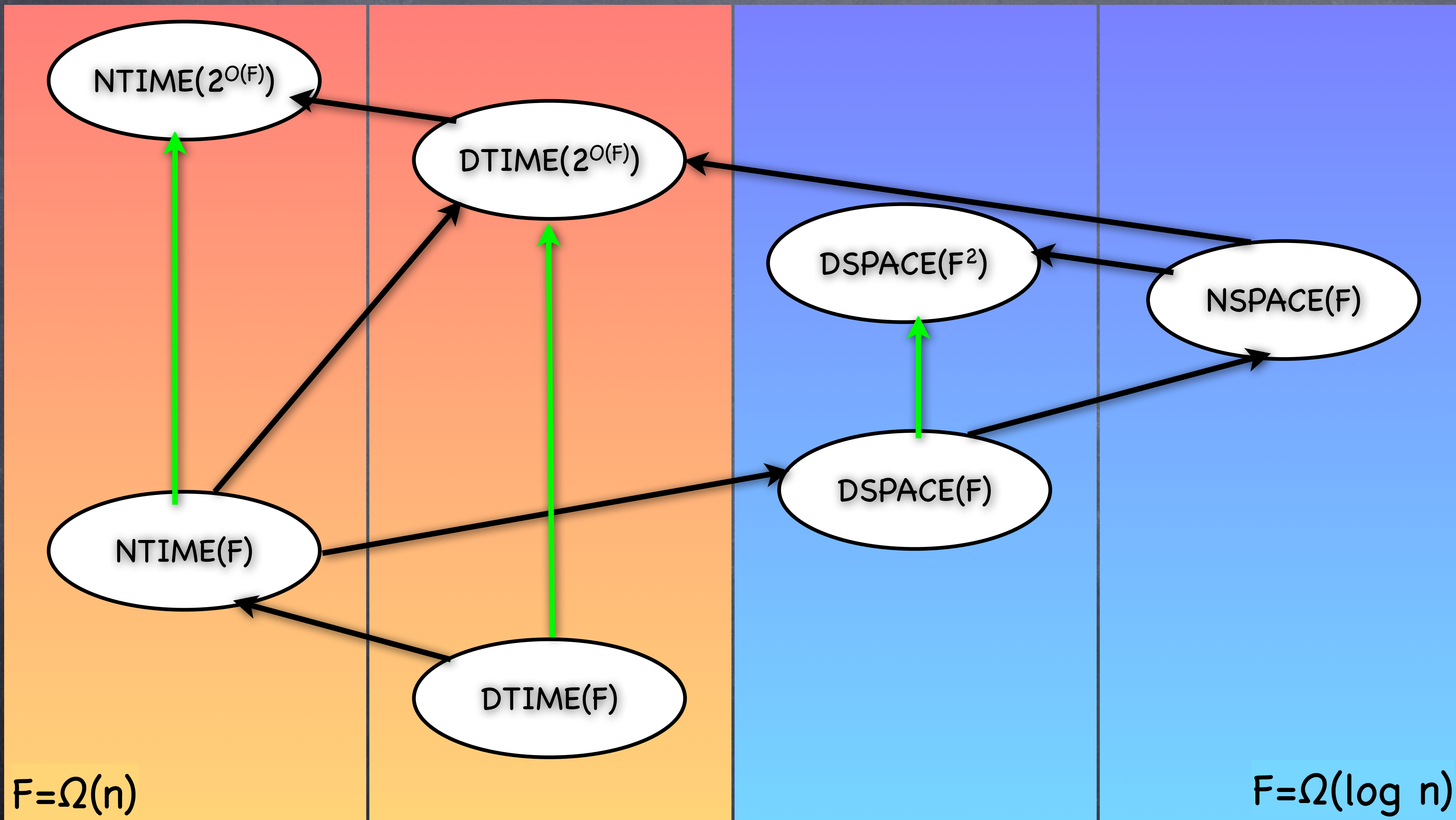
SPACE and TIME



SPACE and TIME



SPACE and TIME



Zoo, so far

Zoo, so far

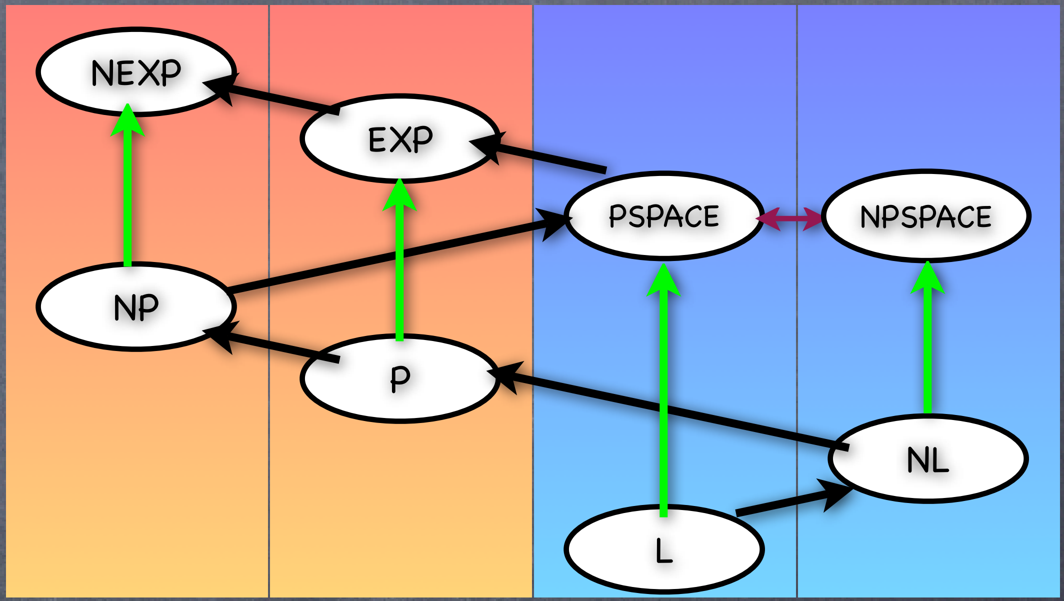
- Major classes of interest (so far):

Zoo, so far

- Major classes of interest (so far):
  - P, EXP; NP, NEXP; L, NL; PSPACE, NPSPACE

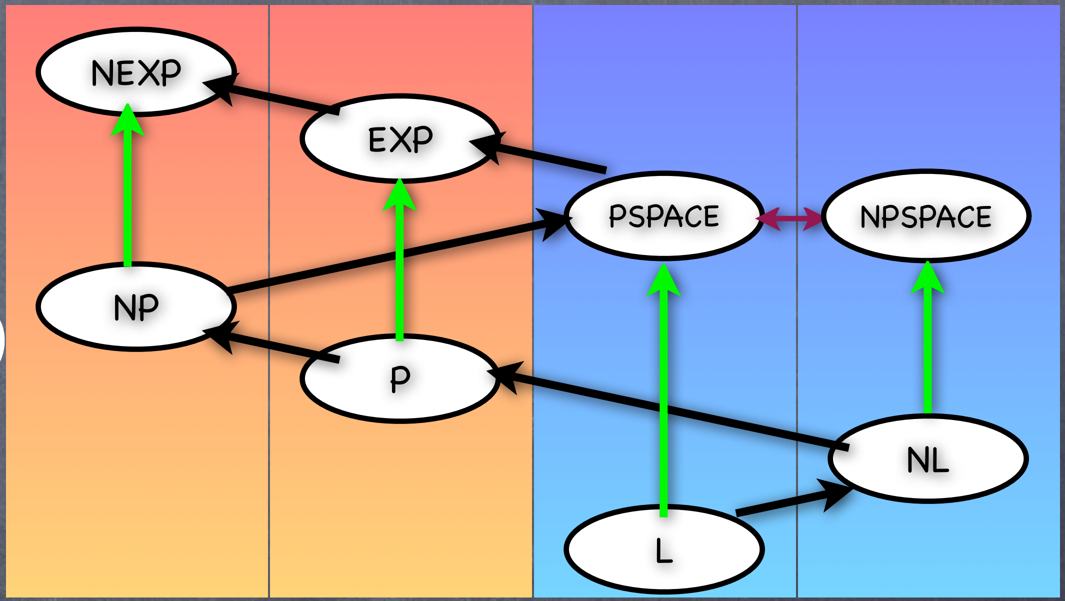
Zoo, so far

- Major classes of interest (so far):
  - P, EXP; NP, NEXP; L, NL; PSPACE, NPSPACE



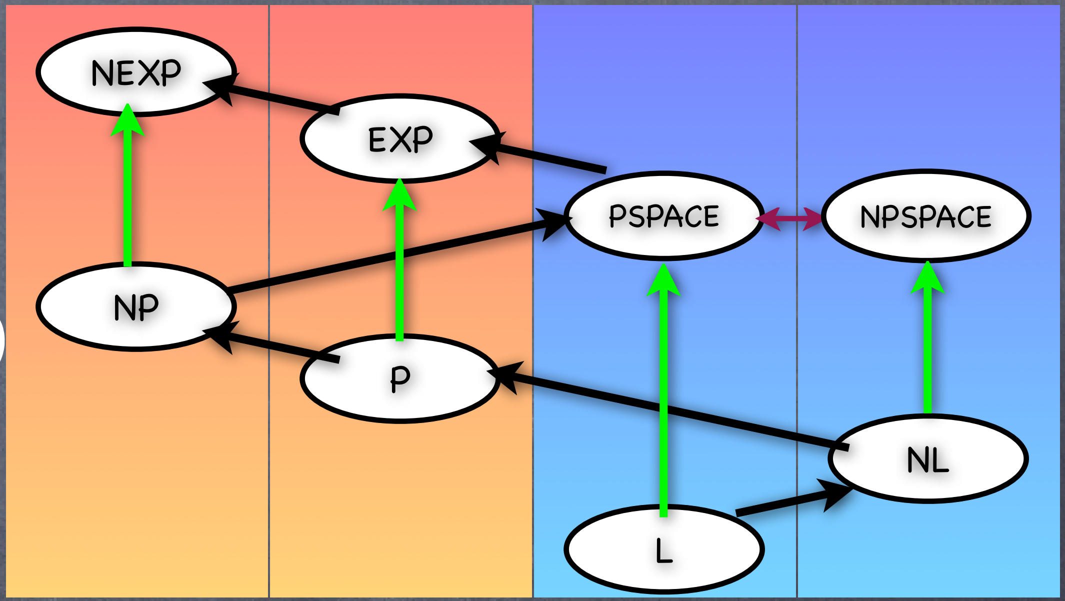
Zoo, so far

- Major classes of interest (so far):
    - P, EXP; NP, NEXP; L, NL; PSPACE, NPSPACE
    - PSPACE = NPSPACE (by Savitch)



Zoo, so far

- Major classes of interest (so far):
  - P, EXP; NP, NEXP; L, NL; PSPACE, NPSPACE
  - PSPACE = NPSPACE (by Savitch)
- Coming up:
  - PSPACE-completeness



PSPACE completeness

PSPACE completeness

- A language L is PSPACE-Complete if for all L′ in PSPACE, $L' \leq_p L$ and L in PSPACE

PSPACE completeness

- A language L is PSPACE-Complete if for all L′ in PSPACE, $L' \leq_p L$ and L in PSPACE
- Trivial PSPACE-complete problem:
  SPACETM = { $(M,z,1^n)$ | TM M accepts z within space n }

PSPACE completeness

- A language L is PSPACE-Complete if for all L′ in PSPACE, $L' \leq_p L$ and L in PSPACE
- Trivial PSPACE-complete problem:
  SPACETM = { $(M,z,1^n)$ | TM M accepts z within space n }
- (An) essence of PSPACE: Understanding 2-player games

PSPACE completeness

- A language L is PSPACE-Complete if for all L′ in PSPACE, $L' \leq_p L$ and L in PSPACE
- Trivial PSPACE-complete problem:
  SPACETM = { $(M,z,1^n)$ | TM M accepts z within space n }
- (An) essence of PSPACE: Understanding 2-player games
  - Can the first/second player always win?

QBF Game

QBF Game

- Two players: Alice and Adversary, each given n (mutually disjoint) sets of variables (sets numbered [1,n])

QBF Game

- Two players: Alice and Adversary, each given n (mutually disjoint) sets of variables (sets numbered [1,n])
- Given a boolean formula over these variables

QBF Game

- Two players: Alice and Adversary, each given n (mutually disjoint) sets of variables (sets numbered [1,n])
- Given a boolean formula over these variables
  - In i^{th} round players set the values of the variables in their i^{th} sets. Say Alice moves first.

QBF Game

- Two players: Alice and Adversary, each given n (mutually disjoint) sets of variables (sets numbered [1,n])
- Given a boolean formula over these variables
  - In i^{th} round players set the values of the variables in their i^{th} sets. Say Alice moves first.
  - When all variables set, formula evaluated. If true Alice wins, else adversary wins

QBF Game

- Two players: Alice and Adversary, each given n (mutually disjoint) sets of variables (sets numbered [1,n])
- Given a boolean formula over these variables
  - In i^{th} round players set the values of the variables in their i^{th} sets. Say Alice moves first.
  - When all variables set, formula evaluated. If true Alice wins, else adversary wins
- Given a QBF game does Alice have a sure-to-win strategy

QBF game: examples

QBF game: examples

- Vars: x_1, y_1, x_2, y_2, x_3, y_3. Formula: $\varphi(x_1,y_1,x_2,y_1,x_3,y_3)$

QBF game: examples

- Vars: x_1, y_1, x_2, y_2, x_3, y_3. Formula: $\varphi(x_1,y_1,x_2,y_1,x_3,y_3)$
- Say, no variables for Adversary. Only x_1

QBF game: examples

- Vars: x_1, y_1, x_2, y_2, x_3, y_3. Formula: $\varphi(x_1,y_1,x_2,y_1,x_3,y_3)$
- Say, no variables for Adversary. Only x_1
  - Strategy for Alice? Is "$\exists x_1\ \varphi(x_1)$" true?

QBF game: examples

- Vars: x_1, y_1, x_2, y_2, x_3, y_3. Formula: $\varphi(x_1,y_1,x_2,y_1,x_3,y_3)$
- Say, no variables for Adversary. Only x_1
  - Strategy for Alice? Is "$\exists x_1\ \varphi(x_1)$" true?
- Say, no variables for Alice. Only y_1

QBF game: examples

- Vars: x_1, y_1, x_2, y_2, x_3, y_3. Formula: $\varphi(x_1,y_1,x_2,y_1,x_3,y_3)$
- Say, no variables for Adversary. Only x_1
  - Strategy for Alice? Is "$\exists x_1\ \varphi(x_1)$" true?
- Say, no variables for Alice. Only y_1
  - "Strategy" for Alice? Is "$\forall y_1\ \varphi(y_1)$" true?

QBF game: examples

- Vars: x_1, y_1, x_2, y_2, x_3, y_3. Formula: $\varphi(x_1,y_1,x_2,y_1,x_3,y_3)$
- Say, no variables for Adversary. Only x_1
  - Strategy for Alice? Is "$\exists x_1\ \varphi(x_1)$" true?
- Say, no variables for Alice. Only y_1
  - "Strategy" for Alice? Is "$\forall y_1\ \varphi(y_1)$" true?
- Say only x_1, y_1 (now, that's more like a game):

QBF game: examples

- Vars: x_1, y_1, x_2, y_2, x_3, y_3. Formula: $\varphi(x_1,y_1,x_2,y_1,x_3,y_3)$
- Say, no variables for Adversary. Only x_1
  - Strategy for Alice? Is "$\exists x_1\ \varphi(x_1)$" true?
- Say, no variables for Alice. Only y_1
  - "Strategy" for Alice? Is "$\forall y_1\ \varphi(y_1)$" true?
- Say only x_1, y_1 (now, that's more like a game):
  - Strategy for Alice? Is "$\exists x_1 \forall y_1\ \varphi(x_1,y_1)$" true?

QBF game: examples

- Vars: x_1, y_1, x_2, y_2, x_3, y_3. Formula: $\varphi(x_1,y_1,x_2,y_1,x_3,y_3)$
- Say, no variables for Adversary. Only x_1
  - Strategy for Alice? Is "$\exists x_1\ \varphi(x_1)$" true?
- Say, no variables for Alice. Only y_1
  - "Strategy" for Alice? Is "$\forall y_1\ \varphi(y_1)$" true?
- Say only x_1, y_1 (now, that's more like a game):
  - Strategy for Alice? Is "$\exists x_1 \forall y_1\ \varphi(x_1,y_1)$" true?
- In general, winning strategy for Alice exists iff

QBF game: examples

- Vars: x_1, y_1, x_2, y_2, x_3, y_3. Formula: $\varphi(x_1,y_1,x_2,y_1,x_3,y_3)$
- Say, no variables for Adversary. Only x_1
  - Strategy for Alice? Is "$\exists x_1\ \varphi(x_1)$" true?
- Say, no variables for Alice. Only y_1
  - "Strategy" for Alice? Is "$\forall y_1\ \varphi(y_1)$" true?
- Say only x_1, y_1 (now, that's more like a game):
  - Strategy for Alice? Is "$\exists x_1 \forall y_1\ \varphi(x_1,y_1)$" true?
- In general, winning strategy for Alice exists iff
  - $\exists x_1 \forall y_1 \ldots \exists x_n \forall y_n\ \varphi(x_1,y_1,\ldots,x_n,y_n)$ is true

QBF game: examples

- Vars: x_1, y_1, x_2, y_2, x_3, y_3. Formula: $\varphi(x_1,y_1,x_2,y_1,x_3,y_3)$
- Say, no variables for Adversary. Only x_1
  - Strategy for Alice? Is "$\exists x_1\ \varphi(x_1)$" true?
- Say, no variables for Alice. Only y_1
  - "Strategy" for Alice? Is "$\forall y_1\ \varphi(y_1)$" true?
- Say only x_1, y_1 (now, that's more like a game):
  - Strategy for Alice? Is "$\exists x_1 \forall y_1\, \varphi(x_1,y_1)$" true?
- In general, winning strategy for Alice exists iff
  - $\exists x_1 \forall y_1 \ \ldots\ \exists x_n \forall y_n\ \varphi(x_1,y_1,\ldots,x_n,y_n)$ is true
  - Else adversary has a winning strategy

TQBF, the language

TQBF, the language

- True Quantified Boolean Formula:

TQBF, the language

- True Quantified Boolean Formula:
  - $\psi := \exists x_1 \forall y_1 \ \dots \ \exists x_n \forall y_n \ \varphi(x_1,y_1,\dots,x_n,y_n)$

TQBF, the language

- True Quantified Boolean Formula:
  - $\psi := \exists x_1 \forall y_1 \ \dots \ \exists x_n \forall y_n \ \varphi(x_1,y_1,\dots,x_n,y_n)$
  - TQBF = $\{\psi \mid \psi \text{ is true}\}$

TQBF, the language

- True Quantified Boolean Formula:
  - $\psi := \exists x_1 \forall y_1 \ \dots \ \exists x_n \forall y_n \ \varphi(x_1,y_1,\dots,x_n,y_n)$
  - TQBF = $\{\psi \mid \psi \text{ is true}\}$
- e.g. ψ_1: $\exists x \forall y\ (x=y)$, ψ_2: $\forall y \exists x\ (x=y)$

TQBF is in PSPACE

TQBF is in PSPACE

- When is a QBF true?

TQBF is in PSPACE

- When is a QBF true?
  - e.g. $\exists a,b\ \forall c\ \varphi(a,b,c)$

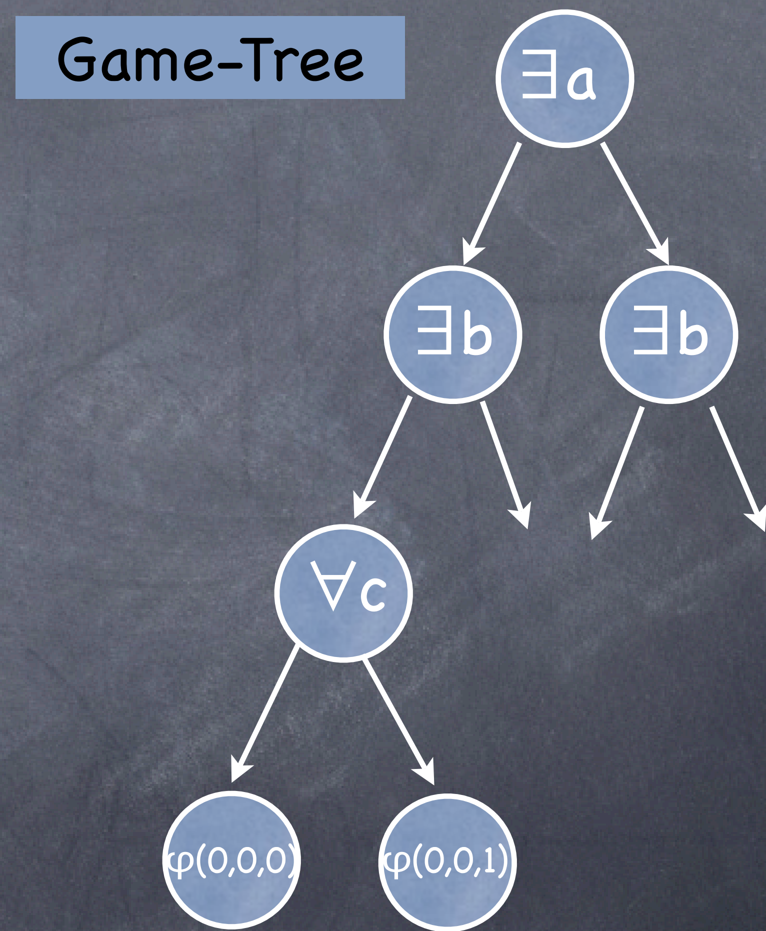
TQBF is in PSPACE

Game-Tree

- When is a QBF true?
  - e.g. $\exists a,b\ \forall c\ \varphi(a,b,c)$

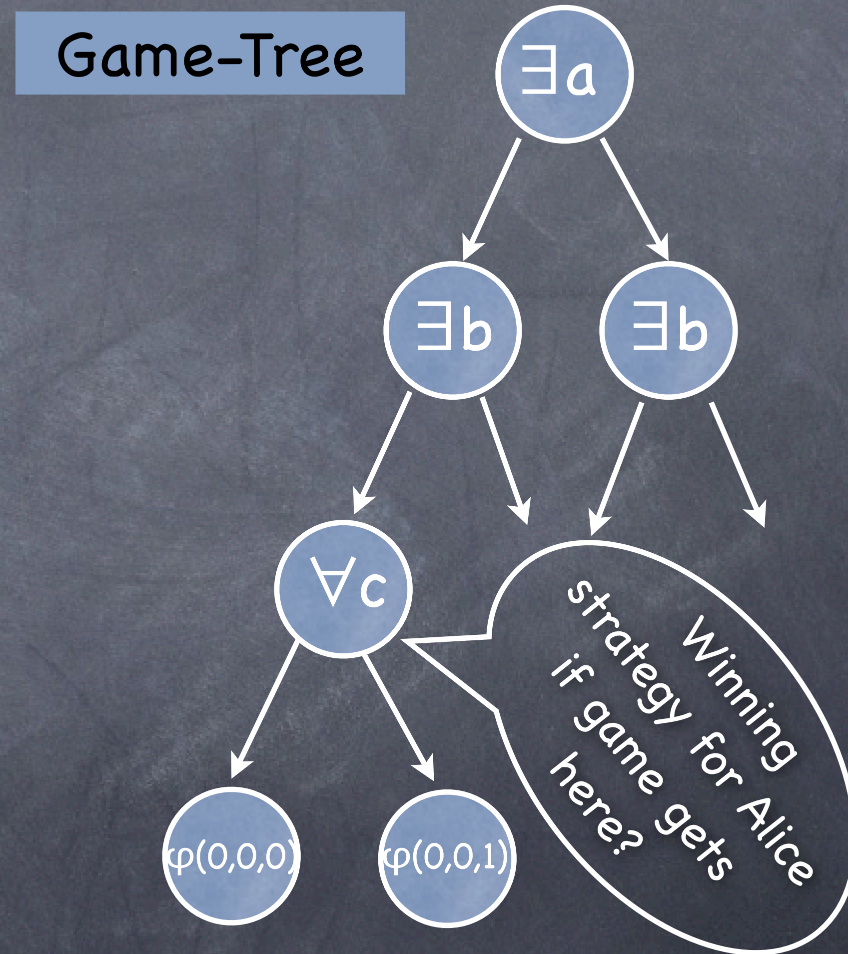
TQBF is in PSPACE

- When is a QBF true?
  - e.g. $\exists a,b\ \forall c\ \varphi(a,b,c)$



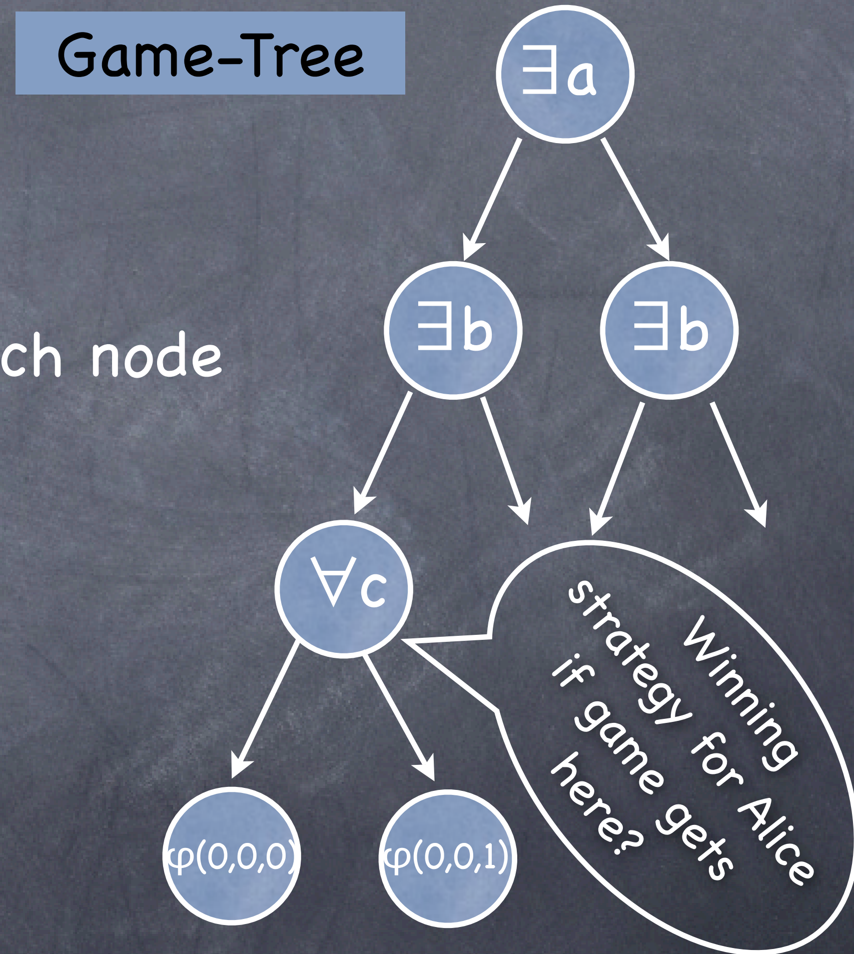
TQBF is in PSPACE

- When is a QBF true?
  - e.g. ∃a,b ∀c φ(a,b,c)



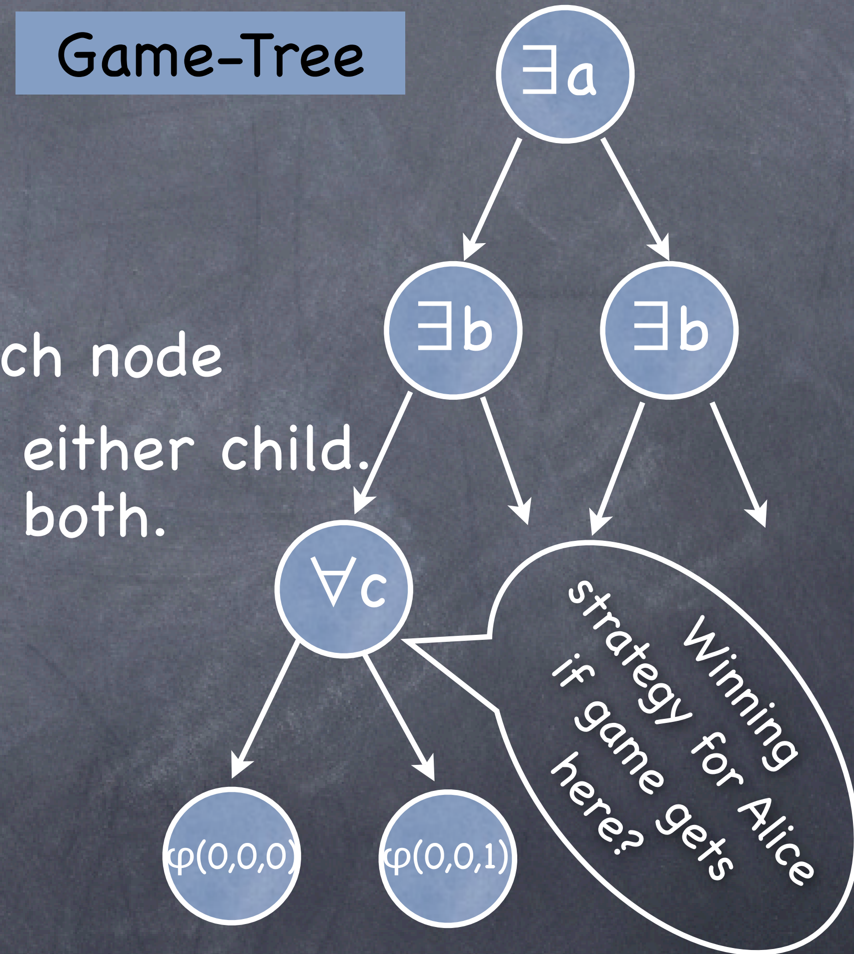
TQBF is in PSPACE

- When is a QBF true?
  - e.g. $\exists a,b\ \forall c\ \varphi(a,b,c)$
  - Ask if winning strategy from each node



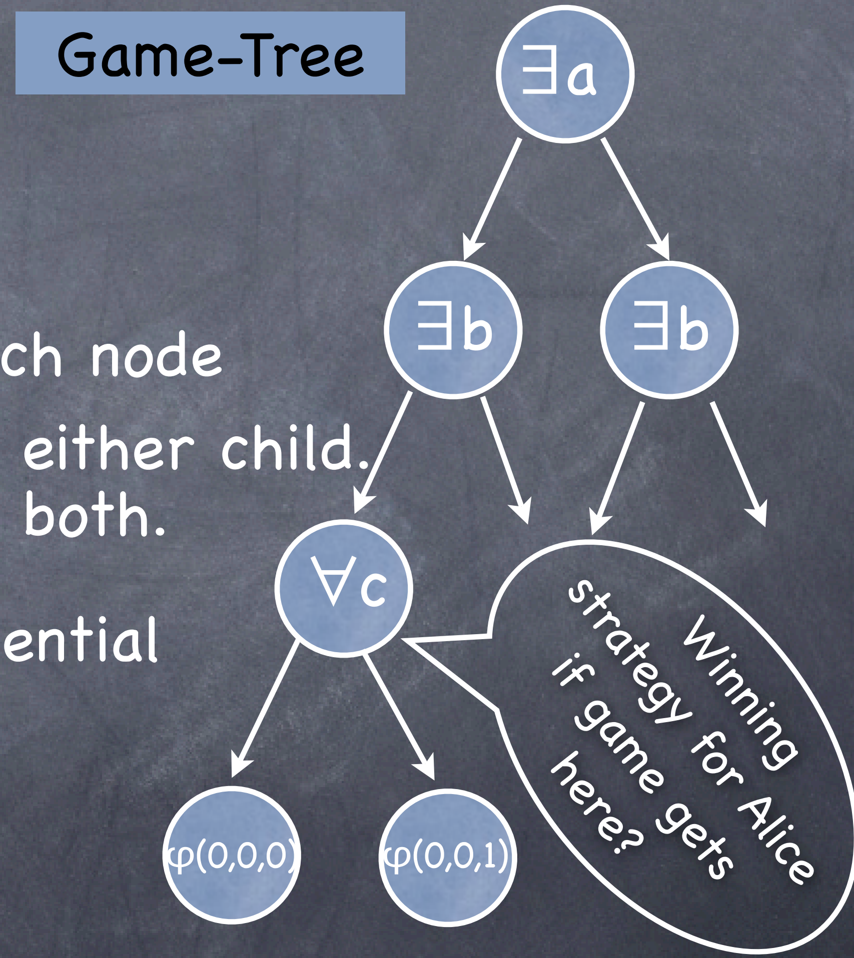
TQBF is in PSPACE

- When is a QBF true?
  - e.g. $\exists a,b\ \forall c\ \varphi(a,b,c)$
  - Ask if winning strategy from each node
    - Yes from $\exists$ node if yes from either child. Yes from $\forall$ node if yes from both.



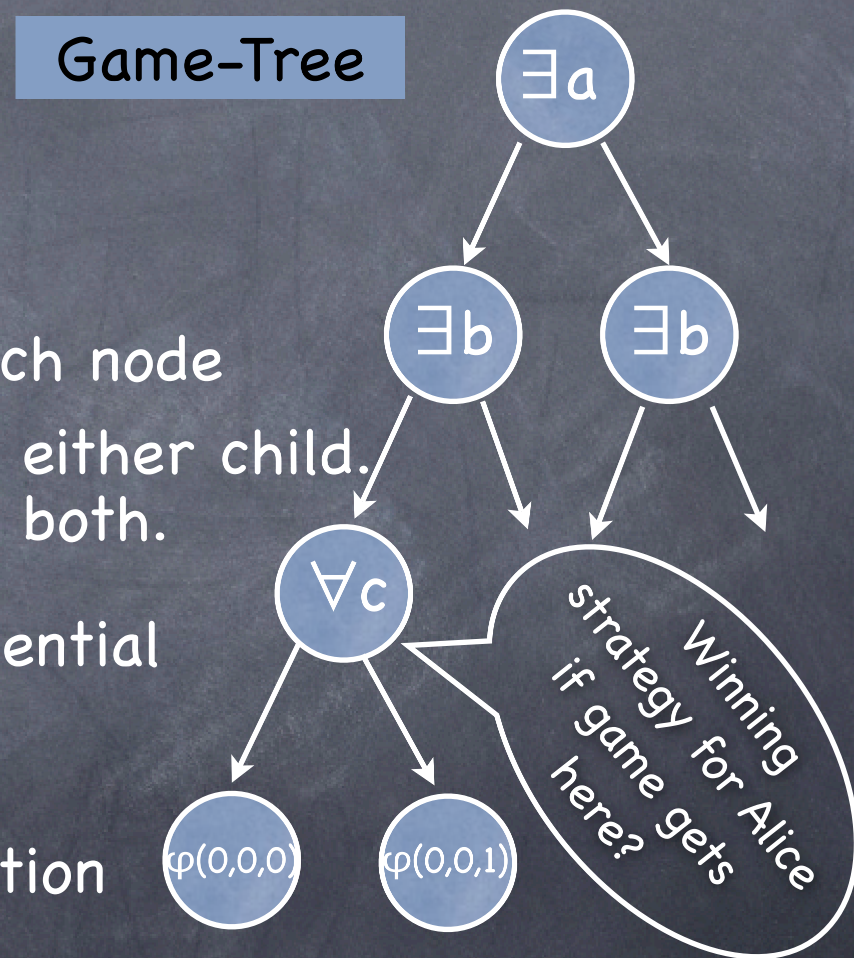
TQBF is in PSPACE

- When is a QBF true?
  - e.g. $\exists a,b\ \forall c\ \varphi(a,b,c)$
  - Ask if winning strategy from each node
    - Yes from $\exists$ node if yes from either child. Yes from $\forall$ node if yes from both.
    - Naive evaluation takes exponential space (and time)



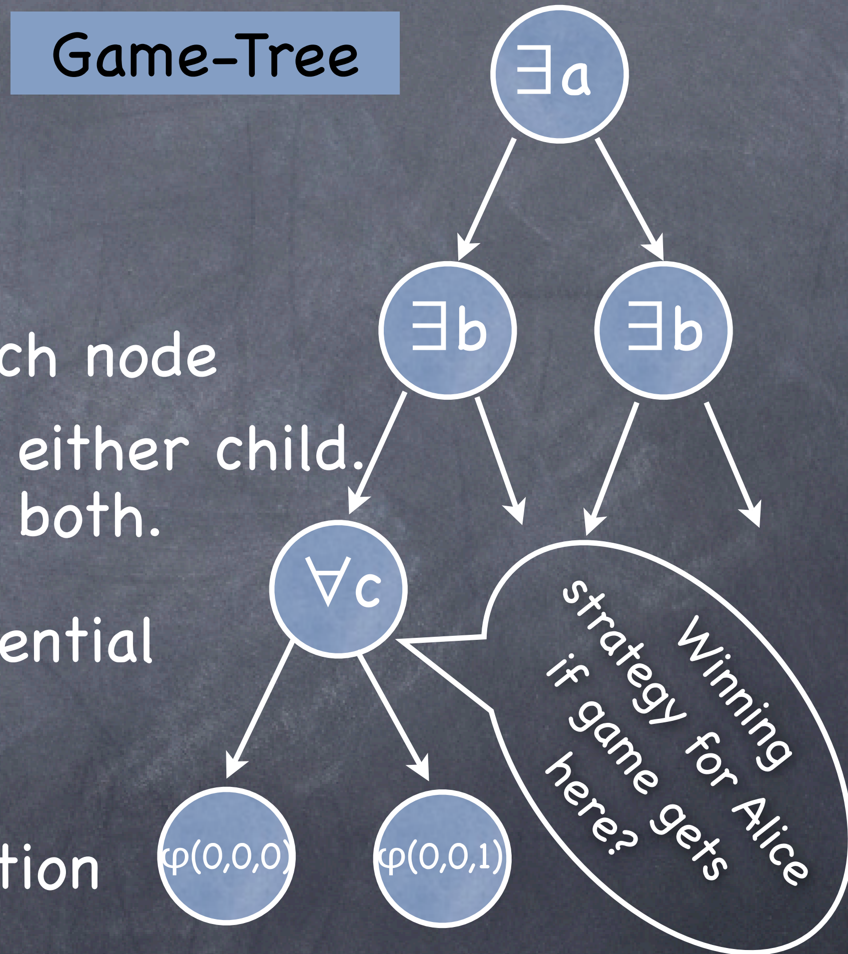
TQBF is in PSPACE

- When is a QBF true?
  - e.g. $\exists a,b\ \forall c\ \varphi(a,b,c)$
  - Ask if winning strategy from each node
    - Yes from $\exists$ node if yes from either child. Yes from $\forall$ node if yes from both.
    - Naive evaluation takes exponential space (and time)
    - Can reuse left child computation space for the right child



TQBF is in PSPACE

- When is a QBF true?
  - e.g. $\exists a,b\ \forall c\ \varphi(a,b,c)$
  - Ask if winning strategy from each node
    - Yes from $\exists$ node if yes from either child. Yes from $\forall$ node if yes from both.
    - Naive evaluation takes exponential space (and time)
    - Can reuse left child computation space for the right child
      - Space needed = O(depth) + for evaluation = poly(|QBF|)



TQBF is PSPACE-hard

TQBF is PSPACE-hard

- For L in PSPACE (i.e., TM M_L decides L in space poly(n), or with configs of size S(n)=poly(n)), show L $\leq_p$ TQBF

TQBF is PSPACE-hard

- For L in PSPACE (i.e., TM M_L decides L in space poly(n), or with configs of size S(n)=poly(n)), show $L \leq_p$ TQBF
- Given x, output f(x) = ψ, s.t. ψ is true iff M_L accepts x

TQBF is PSPACE-hard

- For L in PSPACE (i.e., TM M_L decides L in space poly(n), or with configs of size S(n)=poly(n)), show L $\leq_p$ TQBF
- Given x, output f(x) = ψ, s.t. ψ is true iff M_L accepts x
  - $x \rightarrow \psi$ in poly time. In particular size of ψ is poly(n)

TQBF is PSPACE-hard

- For L in PSPACE (i.e., TM M_L decides L in space poly(n), or with configs of size S(n)=poly(n)), show L $\leq_p$ TQBF
- Given x, output f(x) = ψ, s.t. ψ is true iff M_L accepts x
  - x→ψ in poly time. In particular size of ψ is poly(n)
  - Note: As in Cook's theorem, can build an <u>unquantified</u> formula φ (even 3CNF) s.t. φ is true iff M_L accepts x

TQBF is PSPACE-hard

- For L in PSPACE (i.e., TM M_L decides L in space poly(n), or with configs of size S(n)=poly(n)), show L $\leq_p$ TQBF
- Given x, output f(x) = ψ, s.t. ψ is true iff M_L accepts x
  - x$\rightarrow\psi$ in poly time. In particular size of ψ is poly(n)
  - Note: As in Cook's theorem, can build an <u>unquantified</u> formula φ (even 3CNF) s.t. φ is true iff M_L accepts x
    - But size is poly(time bound on M_L) = exp(n)

TQBF is PSPACE-hard

- For L in PSPACE (i.e., TM M_L decides L in space poly(n), or with configs of size S(n)=poly(n)), show L $\leq_p$ TQBF
- Given x, output f(x) = ψ, s.t. ψ is true iff M_L accepts x
  - x$\rightarrow\psi$ in poly time. In particular size of ψ is poly(n)
  - Note: As in Cook's theorem, can build an <u>unquantified</u> formula φ (even 3CNF) s.t. φ is true iff M_L accepts x
    - But size is poly(time bound on M_L) = exp(n)
    - Use power of quantification to write it succinctly

TQBF is PSPACE-hard

TQBF is PSPACE-hard

- An exponential QBF:

TQBF is PSPACE-hard

- An exponential QBF:
  - $\exists\ C_1\ C_2\ \ldots\ C_T\ \psi_0(C_{start},C_1) \wedge \psi_0(C_1,C_2) \wedge \ldots\ \psi_0(C_T,C_{accept})$

TQBF is PSPACE-hard

- An exponential QBF:
  - $\exists\ C_1\ C_2\ \ldots\ C_T\ \psi_0(C_{start},C_1) \wedge \psi_0(C_1,C_2) \wedge \ldots\ \psi_0(C_T,C_{accept})$
  - Here C_i are variables whose value assignments correspond to configurations. $|C_i| = O(S(n))$, $|\psi_0(C,C')| = O(S(n))$, $T=2^{O(S(n))}$

TQBF is PSPACE-hard

- An exponential QBF:
  - $\exists\ C_1\ C_2\ \ldots\ C_T\ \psi_0(C_{start},C_1) \wedge \psi_0(C_1,C_2) \wedge \ldots\ \psi_0(C_T,C_{accept})$
  - Here C_i are variables whose value assignments correspond to configurations. $|C_i| = O(S(n))$, $|\psi_0(C,C')| = O(S(n))$, $T=2^{O(S(n))}$
  - $\psi_0(C,C')$ is an unquantified formula (only variables being C,C'), s.t. it is true iff C evolves into C' in one step

TQBF is PSPACE-hard

- An exponential QBF:
  - $\exists\ C_1\ C_2\ \dots\ C_T\ \psi_0(C_{start},C_1) \wedge \psi_0(C_1,C_2) \wedge \dots\ \psi_0(C_T,C_{accept})$
  - Here C_i are variables whose value assignments correspond to configurations. $|C_i| = O(S(n))$, $|\psi_0(C,C')| = O(S(n))$, $T=2^{O(S(n))}$
  - $\psi_0(C,C')$ is an unquantified formula (only variables being C,C'), s.t. it is true iff C evolves into C' in one step
    - F be the (const. sized) formula to derive each bit of new config from a few bits in the previous config. $\psi_0(C,C')$ is conjunction of equality conditions enforcing consistency with F. $|\psi_0(C,C')| = O(|C|)$

TQBF is PSPACE-hard

- An exponential QBF:
  - $\exists\ C_1\ C_2\ \ldots\ C_T\ \psi_0(C_{start},C_1) \wedge \psi_0(C_1,C_2) \wedge \ldots\ \psi_0(C_T,C_{accept})$
  - Here C_i are variables whose value assignments correspond to configurations. $|C_i| = O(S(n))$, $|\psi_0(C,C')| = O(S(n))$, $T=2^{O(S(n))}$
  - $\psi_0(C,C')$ is an unquantified formula (only variables being C,C'), s.t. it is true iff C evolves into C' in one step
    - F be the (const. sized) formula to derive each bit of new config from a few bits in the previous config. $\psi_0(C,C')$ is conjunction of equality conditions enforcing consistency with F. $|\psi_0(C,C')| = O(|C|)$
      - $\psi_0(C,C')$: $\bigwedge_j \left(C'^{(j)} = F(C^{(j-c)},\ldots,C^{(j+c)})\right)$

TQBF is PSPACE-hard

TQBF is PSPACE-hard

- Plan for a more succinct ψ: A partly quantified BF ψ_i s.t. $\psi_i(C,C')$ is fully quantified and is true iff C' reachable from C in the configuration graph $G(M_L,x)$ within 2^i steps. Output $\psi=\psi_{S(n)}(start,accept)$

TQBF is PSPACE-hard

- Plan for a more succinct ψ: A partly quantified BF ψ_i s.t. $\psi_i(C,C')$ is fully quantified and is true iff C' reachable from C in the configuration graph $G(M_L,x)$ within 2^i steps. Output $\psi=\psi_{S(n)}(\text{start},\text{accept})$
  - Base case (i=0): an unquantified formula, ψ_0

TQBF is PSPACE-hard

- Plan for a more succinct ψ: A partly quantified BF ψ_i s.t. $\psi_i(C,C')$ is fully quantified and is true iff C' reachable from C in the configuration graph $G(M_L,x)$ within 2^i steps. Output $\psi=\psi_{S(n)}(\text{start},\text{accept})$
  - Base case (i=0): an unquantified formula, ψ_0
  - $\psi_{i+1}(C,C') := \exists C''\ \psi_i(C,C'') \wedge \psi_i(C'',C')$

TQBF is PSPACE-hard

- Plan for a more succinct ψ: A partly quantified BF ψ_i s.t. $\psi_i(C,C')$ is fully quantified and is true iff C' reachable from C in the configuration graph $G(M_L,x)$ within 2^i steps. Output $\psi=\psi_{S(n)}(\text{start},\text{accept})$
  - Base case (i=0): an unquantified formula, ψ_0
  - $\psi_{i+1}(C,C') := \exists C''\ \psi_i(C,C'') \wedge \psi_i(C'',C')$

c.f. Savitch's theorem

TQBF is PSPACE-hard

- Plan for a more succinct ψ: A partly quantified BF ψ_i s.t. $\psi_i(C,C')$ is fully quantified and is true iff C' reachable from C in the configuration graph $G(M_L,x)$ within 2^i steps. Output $\psi=\psi_{S(n)}(start,accept)$
  - Base case (i=0): an unquantified formula, ψ_0
  - $\psi_{i+1}(C,C') := \exists C''\ \psi_i(C,C'') \wedge \psi_i(C'',C')$
    - Needs to be rewritten in "Prenex Normal form"

c.f. Savitch's theorem

TQBF is PSPACE-hard

- Plan for a more succinct ψ: A partly quantified BF ψ_i s.t. $\psi_i(C,C')$ is fully quantified and is true iff C' reachable from C in the configuration graph $G(M_L,x)$ within 2^i steps. Output $\psi=\psi_{S(n)}(start,accept)$

c.f. Savitch's theorem

  - Base case (i=0): an unquantified formula, ψ_0
  - $\psi_{i+1}(C,C') := \exists C'' \; \psi_i(C,C'') \wedge \psi_i(C'',C')$
    - Needs to be rewritten in "Prenex Normal form"
    - Problem: $|\psi_{S(n)}|$ still exponential in $S(n)$

TQBF is PSPACE-hard

- Plan for a more succinct ψ: A partly quantified BF ψ_i s.t. $\psi_i(C,C')$ is fully quantified and is true iff C' reachable from C in the configuration graph $G(M_L,x)$ within 2^i steps. Output $\psi = \psi_{S(n)}(start,accept)$

c.f. Savitch's theorem

  - Base case (i=0): an unquantified formula, ψ_0
  - $\psi_{i+1}(C,C') := \exists C''\ \psi_i(C,C'') \land \psi_i(C'',C')$
    - Needs to be rewritten in "Prenex Normal form"
    - Problem: $|\psi_{S(n)}|$ still exponential in S(n)
      - In fact, same as naive formula!

TQBF is PSPACE-hard

TQBF is PSPACE-hard

- $\psi_{i+1}(C,C') := \exists C''\ \psi_i(C,C'') \wedge \psi_i(C'',C')$

TQBF is PSPACE-hard

- $\psi_{i+1}(C,C') := \exists C'' \; \psi_i(C,C'') \wedge \psi_i(C'',C')$
  - Problem: $|\psi_{S(n)}|$ exponential in $S(n)$

TQBF is PSPACE-hard

- $\psi_{i+1}(C,C') := \exists C'' \; \psi_i(C,C'') \wedge \psi_i(C'',C')$
  - Problem: $|\psi_{S(n)}|$ exponential in $S(n)$
  - More variables/quantification to "reuse" formula

TQBF is PSPACE-hard

- $\psi_{i+1}(C,C') := \exists C''\ \psi_i(C,C'') \wedge \psi_i(C'',C')$
  - Problem: $|\psi_{S(n)}|$ exponential in $S(n)$
  - More variables/quantification to "reuse" formula
- $\psi_{i+1}(C,C') := \exists C''\ \forall (D,D')\ \ ((D,D')=(C,C'') \vee (D,D')=(C'',C')) \Rightarrow \psi_i(D,D')$

TQBF is PSPACE-hard

- $\psi_{i+1}(C,C') := \exists C''\ \psi_i(C,C'') \wedge \psi_i(C'',C')$
  - Problem: $|\psi_{S(n)}|$ exponential in $S(n)$
  - More variables/quantification to "reuse" formula
- $\psi_{i+1}(C,C') := \exists C''\ \forall(D,D')\ \ ((D,D')=(C,C'') \vee (D,D')=(C'',C')) \Rightarrow \psi_i(D,D')$
  - $=$ and $\Rightarrow$ shorthands for slightly longer formulas

TQBF is PSPACE-hard

- $\psi_{i+1}(C,C') := \exists C''\ \psi_i(C,C'') \wedge \psi_i(C'',C')$
  - Problem: $|\psi_{S(n)}|$ exponential in $S(n)$
  - More variables/quantification to "reuse" formula

c.f. Savitch's theorem

- $\psi_{i+1}(C,C') := \exists C''\ \forall(D,D')\ \ ((D,D')=(C,C'') \vee (D,D')=(C'',C')) \Rightarrow \psi_i(D,D')$
  - $=$ and $\Rightarrow$ shorthands for slightly longer formulas

TQBF is PSPACE-hard

- $\psi_{i+1}(C,C') := \exists C'' \ \psi_i(C,C'') \wedge \psi_i(C'',C')$
  - Problem: $|\psi_{S(n)}|$ exponential in $S(n)$
  - More variables/quantification to "reuse" formula

c.f. Savitch's theorem

- $\psi_{i+1}(C,C') := \exists C'' \ \forall (D,D') \ \ ((D,D')=(C,C'') \vee (D,D')=(C'',C')) \Rightarrow \psi_i(D,D')$
  - $=$ and $\Rightarrow$ shorthands for slightly longer formulas
- $|\psi_{S(n)}| = O(S(n)) + |\psi_{S(n)-1}| = O(S(n)^2) + |\psi_0| = O(S(n)^2)$

TQBF is PSPACE-hard

- $\psi_{i+1}(C,C') := \exists C''\ \psi_i(C,C'') \wedge \psi_i(C'',C')$
  - Problem: $|\psi_{S(n)}|$ exponential in $S(n)$
  - More variables/quantification to "reuse" formula

c.f. Savitch's theorem

- $\psi_{i+1}(C,C') := \exists C''\ \forall (D,D')\ \ ((D,D')=(C,C'') \vee (D,D')=(C'',C')) \Rightarrow \psi_i(D,D')$
  - $=$ and $\Rightarrow$ shorthands for slightly longer formulas
- $|\psi_{S(n)}| = O(S(n)) + |\psi_{S(n)-1}| = O(S(n)^2) + |\psi_0| = O(S(n)^2)$
- "Quantification is a powerful programming language"

TQBF

TQBF

- PSPACE-complete

TQBF

- PSPACE-complete
- Generalizes SAT and SAT^c (which have only one quantifier)

TQBF

- PSPACE-complete
- Generalizes SAT and SAT^c (which have only one quantifier)
- How about 2, 3, 4, ... quantifier alternations?

TQBF

- PSPACE-complete
- Generalizes SAT and SAT^c (which have only one quantifier)
- How about 2, 3, 4, ... quantifier alternations?
  - Coming soon!

Today

Today

- Zoo (more later)

Today

- Zoo (more later)
- TQBF

Today

- Zoo (more later)
- TQBF
    - PSPACE complete

Today

- Zoo (more later)
- TQBF
  - PSPACE complete
  - Will see more of it soon

Today

- Zoo (more later)
- TQBF
  - PSPACE complete
  - Will see more of it soon
- Next Lecture: NL

Today

- Zoo (more later)
- TQBF
  - PSPACE complete
  - Will see more of it soon
- Next Lecture: NL
  - NL-completeness

Today

- Zoo (more later)
- TQBF
  - PSPACE complete
  - Will see more of it soon
- Next Lecture: NL
  - NL-completeness
  - NL = co-NL