Lecture 29

Game Tree

Config:

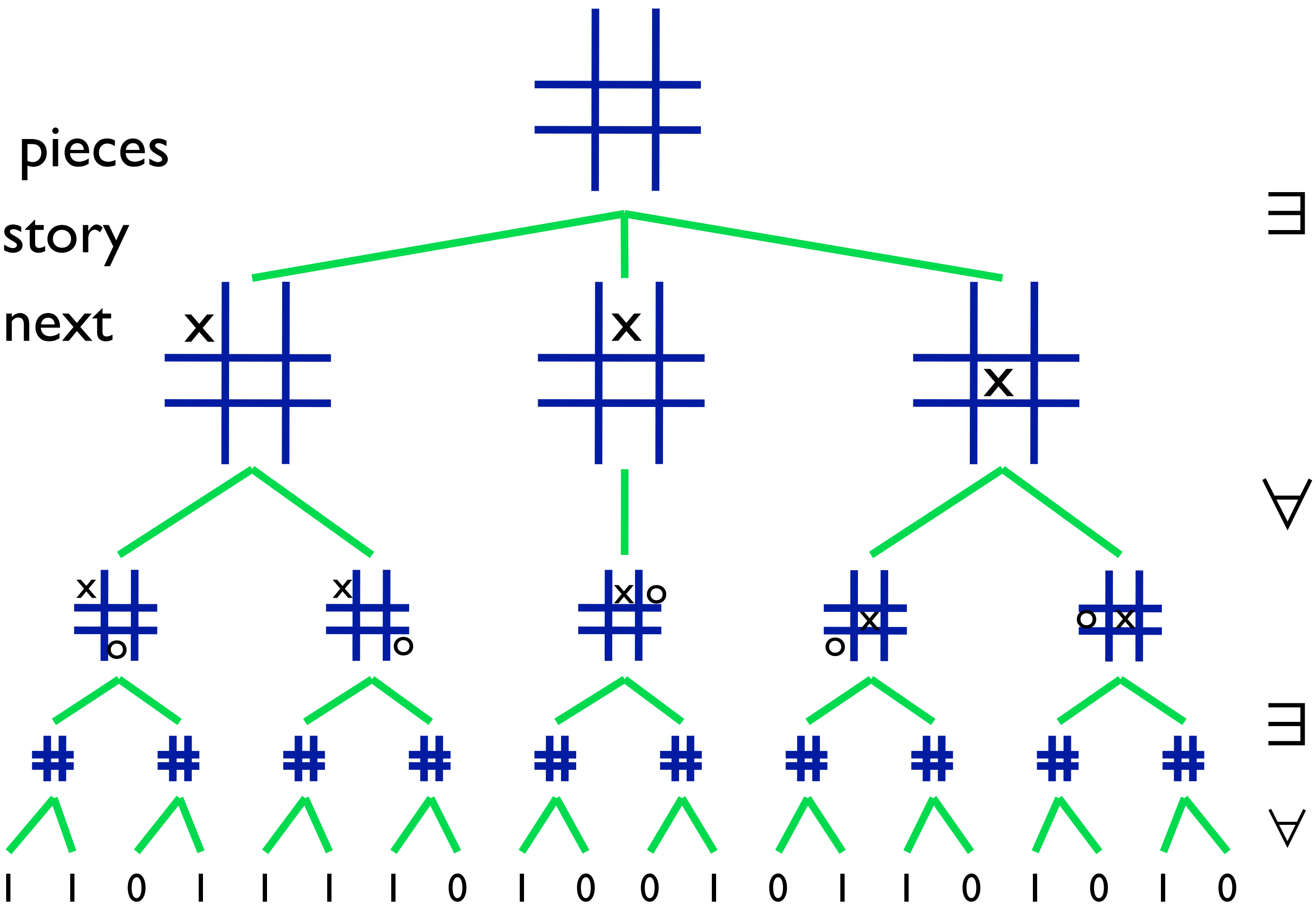
Where are pieces

Relevant history

Who goes next

Play:

All moves



Win/lose:

Game Tree

Config:

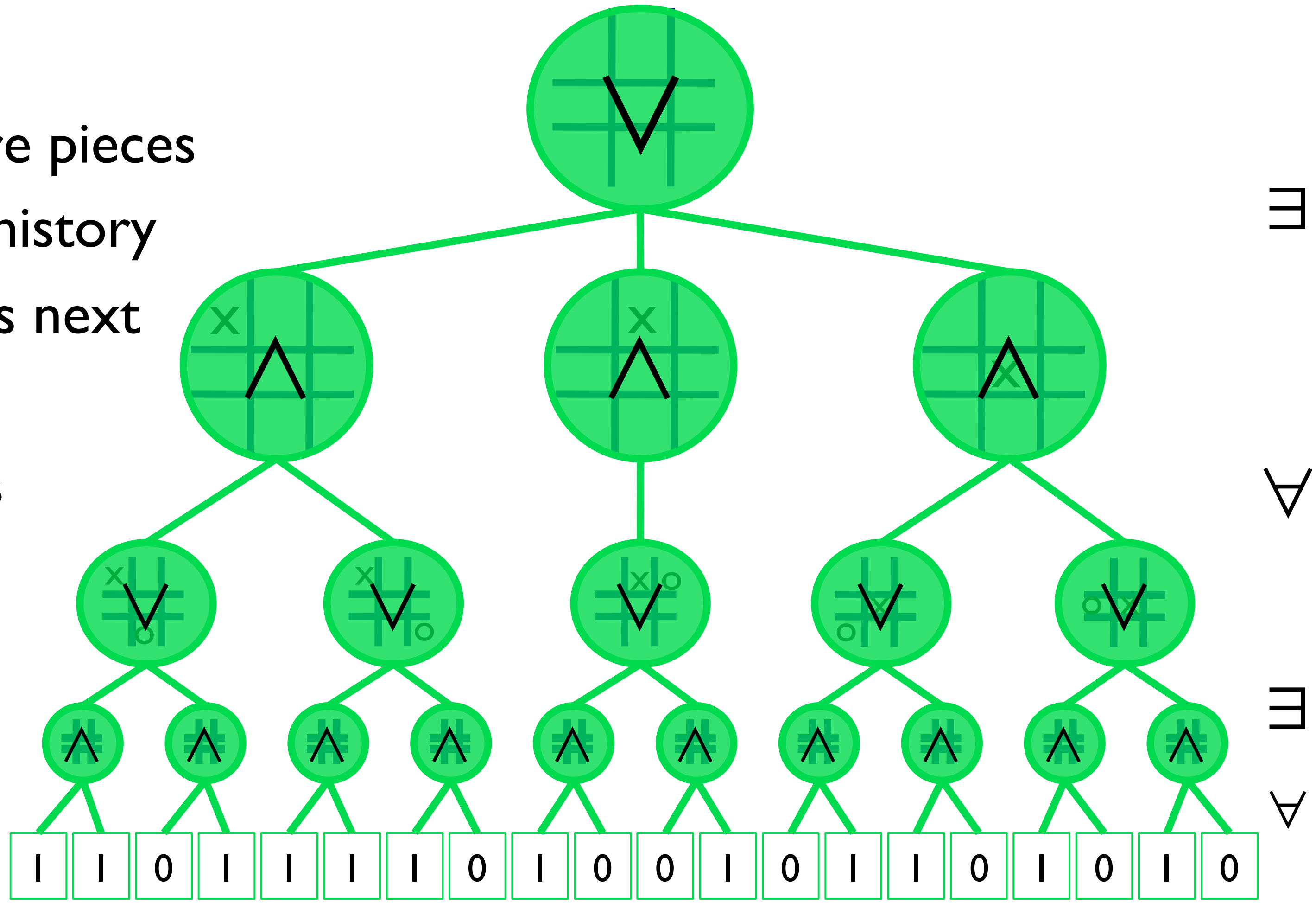
Where are pieces

Relevant history

Who goes next

Play:

All moves



Win/lose:

Winning Strategy

Config:

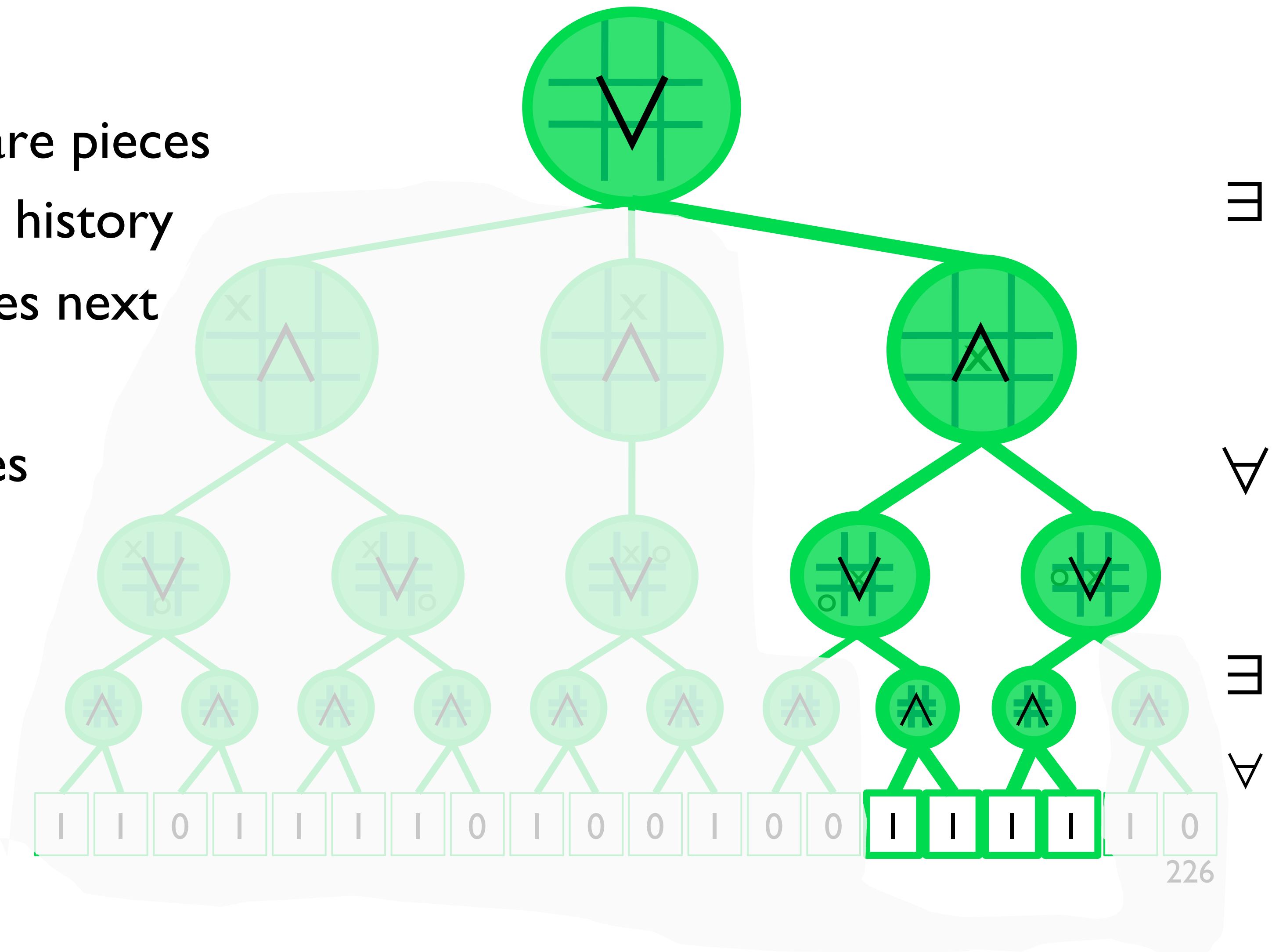
Where are pieces

Relevant history

Who goes next

Play:

All moves



Win/lose:

Complexity of 2 person, perfect information games

From above, *IF*

config (incl. history, etc.) is poly size

only poly many successors of one config

each computable in poly time

win/lose configs recognizable in poly time, and

game lasts poly # moves

THEN

in PSPACE!

Pf: depth-first search of tree, calc node values as you go.

A Game About Paths:
Which Player Has A Winning Strategy?

Given: digraph G with $2^n + 1$ vertices, movable markers s, t on two vertices

Outline:

Player I : "I have a path (from s to t)"

Player II: "I doubt it"

Play alternates, starting with player I:

Player I : places marker m on some node ("path goes thru m")

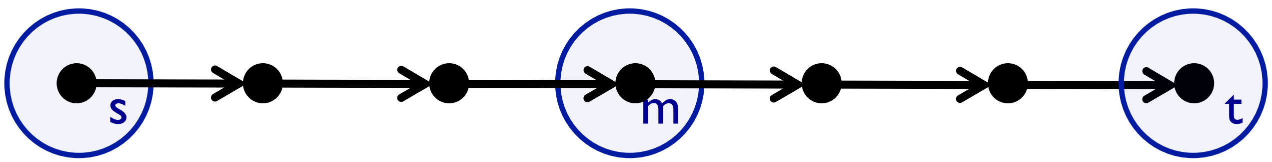
Player II: $(s,t) \leftarrow (s,m)$ or (m,t) ("I doubt this half")

Ends after n rounds; Player I wins if $s = t$, or $s \rightarrow t$ is an edge

Winning The Path Game

Player I has a winning strategy if there is an s-t path:

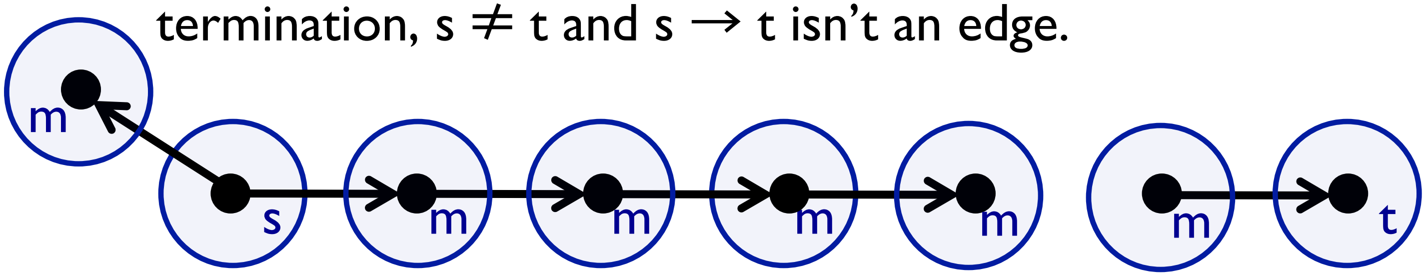
Path has ≤ 2^n edges; choosing middle vertex of that path for “m” in each round halves the remaining path length, so after n rounds, path length is ≤ 1, which is the “win” condition for Player 1.



∃m∀halves,
is path

Player II has a winning strategy if there is no s-t path:

If there is no s-t path, for every m, either there is no s-m path or no m-t path (or both). In the former case, choose (s, m), else (m, t). At termination, s ≠ t and s → t isn’t an edge.



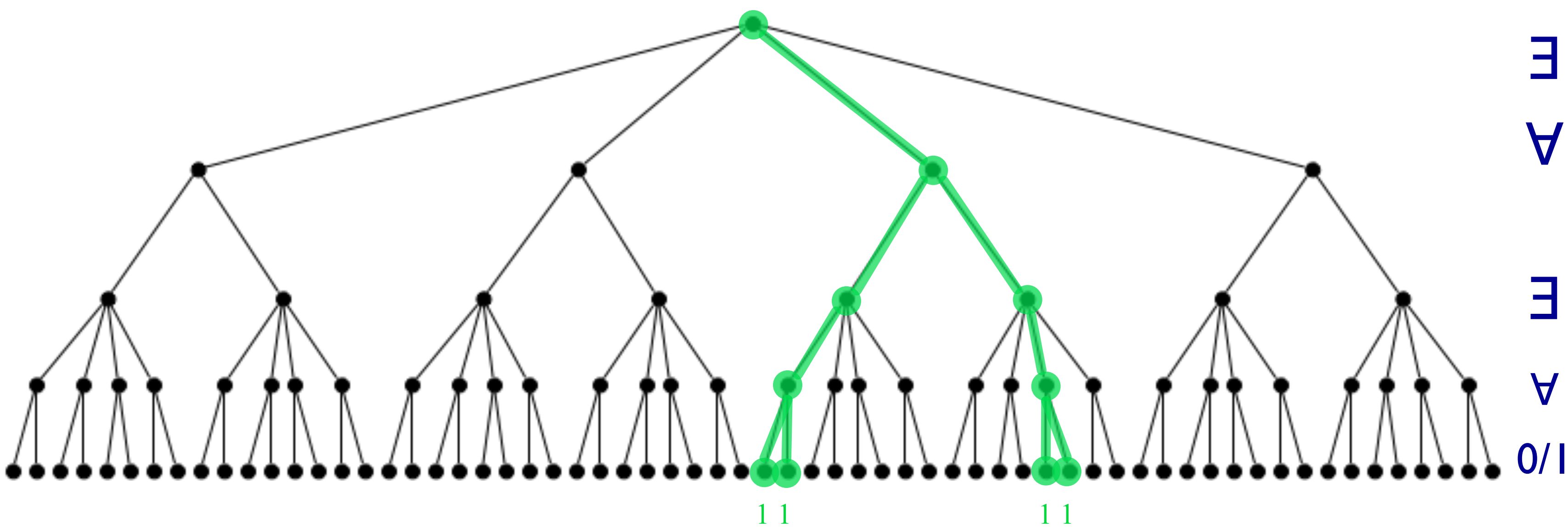
∀m∃half,
no path

Game Tree/Strategy

2n levels

Player I (∃) chooses among many possible "m" nodes

Player II (∀) chooses left/right half



Complexity & The Path Game

M: a space S(n) NTM. WLOG, before accepting, M:

- erases tape
- goes to left end of tape

So, there are unique init & accept configs, C_0, C_a.

Digraph G:

- Nodes: configs of M on fixed input x,
- Edges: $C \rightarrow C'$ iff M can move from config C to C' in 1 step.

M accepts x iff there is a path from C_0 to C_a in G

Savitch's Theorem

Theorem:

$\mathrm{NSPACE}(S(n)) \subseteq \mathrm{DSPACE}(S^2(n))$

Pf:

Accept iff Player I wins path game

Game tree has height log(#configs) = $O(S(n))$

Each node needs $O(S(n))$ bits to describe 2-3 configs (s,m,t)

Can evaluate win/lose at each leaf by examining 2 configs

So, evaluate tree in $O(S^2(n))$ space.

Corollary:

DetPSPACE = NondetPSPACE (So we just say “PSPACE”)

Analogous result for P-TIME is of course the famous $P \stackrel{?}{=} NP$ question.

TQBF
"True Quantified Boolean Formulas"

TQBF = { $\exists y_1 \forall x_1 \exists y_2 \ldots f$ | assignment x,y satisfies formula f }
(each x_i, y_i may be one or many bits; doesn't matter.)

TQBF in PSPACE: think of it as a game between $\exists$, $\forall$; $\exists$ wins if formula satisfied. Do DFS of game tree as in examples above, evaluating nodes ($\wedge$, $\vee$) as you backtrack.

TQBF is PSPACE-complete

"TQBF is to PSPACE as SAT is to NP"

TQBF = { $\exists y_1 \forall x_1 \exists y_2 \ldots f$ | assignment x,y satisfies formula f }

Theorem: TQBF is PSPACE-complete

Pf Idea:

TQBF in PSPACE: above

M an arbitrary n^k space TM, show L(M) $\leq_p$ TQBF: below

y_k: the n^k-bit config "m" picked by $\exists$-player in round k
x_k: 1 bit; $\forall$-player chooses which half-path is challenged
Formula f: x's select the appropriate pair of y configs; check that 1st moves to 2nd in one step (alá Cook's Thm)

More Detail

For "x selects a pair of y's", use the following trick:

$f_1(s_1,t_1) = \exists y_1 \forall x_1\ g(s_1,t_1,y_1,x_1)$

becomes

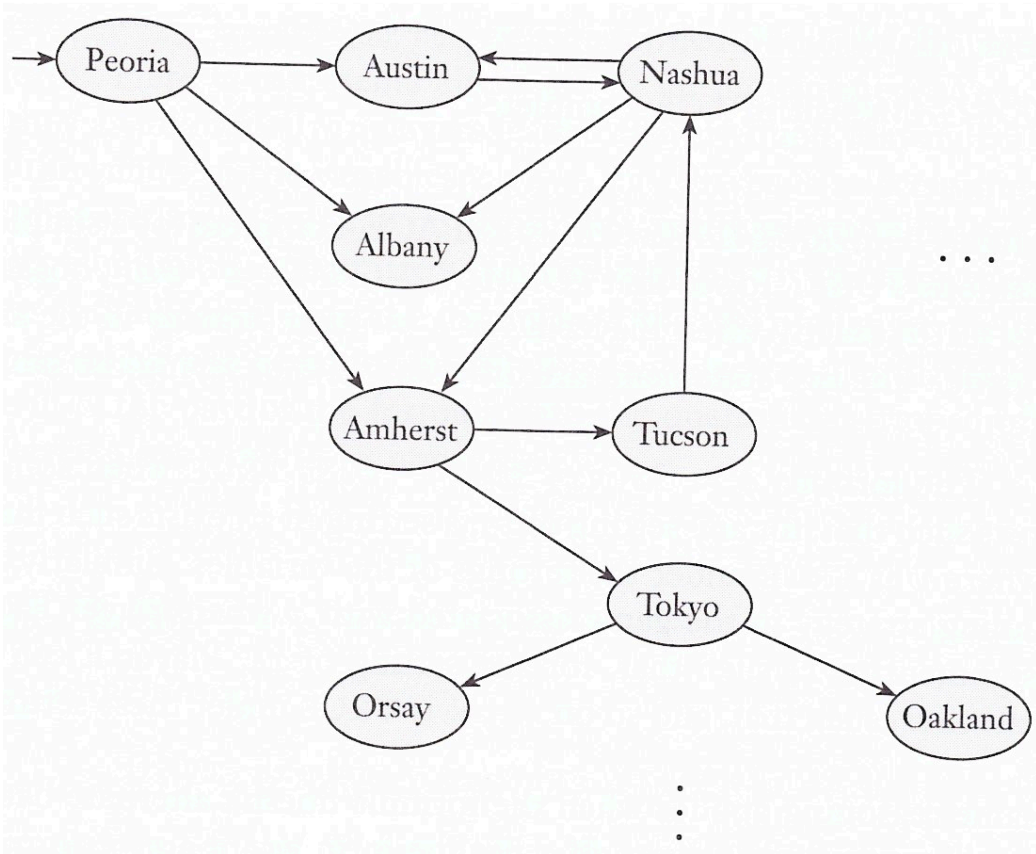
$$\exists y_1 \forall x_1\ \exists s_2,t_2 [\ (\ x_1 \rightarrow (s_2 = s_1 \wedge t_2 = y_1)) \wedge$$
$$(\neg x_1 \rightarrow (s_2 = y_1 \wedge t_2 = t_1)) \wedge f_2(s_2,t_2)\ ]$$

Here, x_1 is a single bit; others represent n^k-bit configs, and "=" means the $\wedge$ of bitwise $\leftrightarrow$ across all bits of a config

The final piece of the formula becomes $\exists z\ g(s_k,t_k,z)$, where $g(s_k,t_k,z)$, ~ as in Cook's Thm, is true if config s_k equals t_k or moves to t_k in 1 step according to M's nondet choice z.

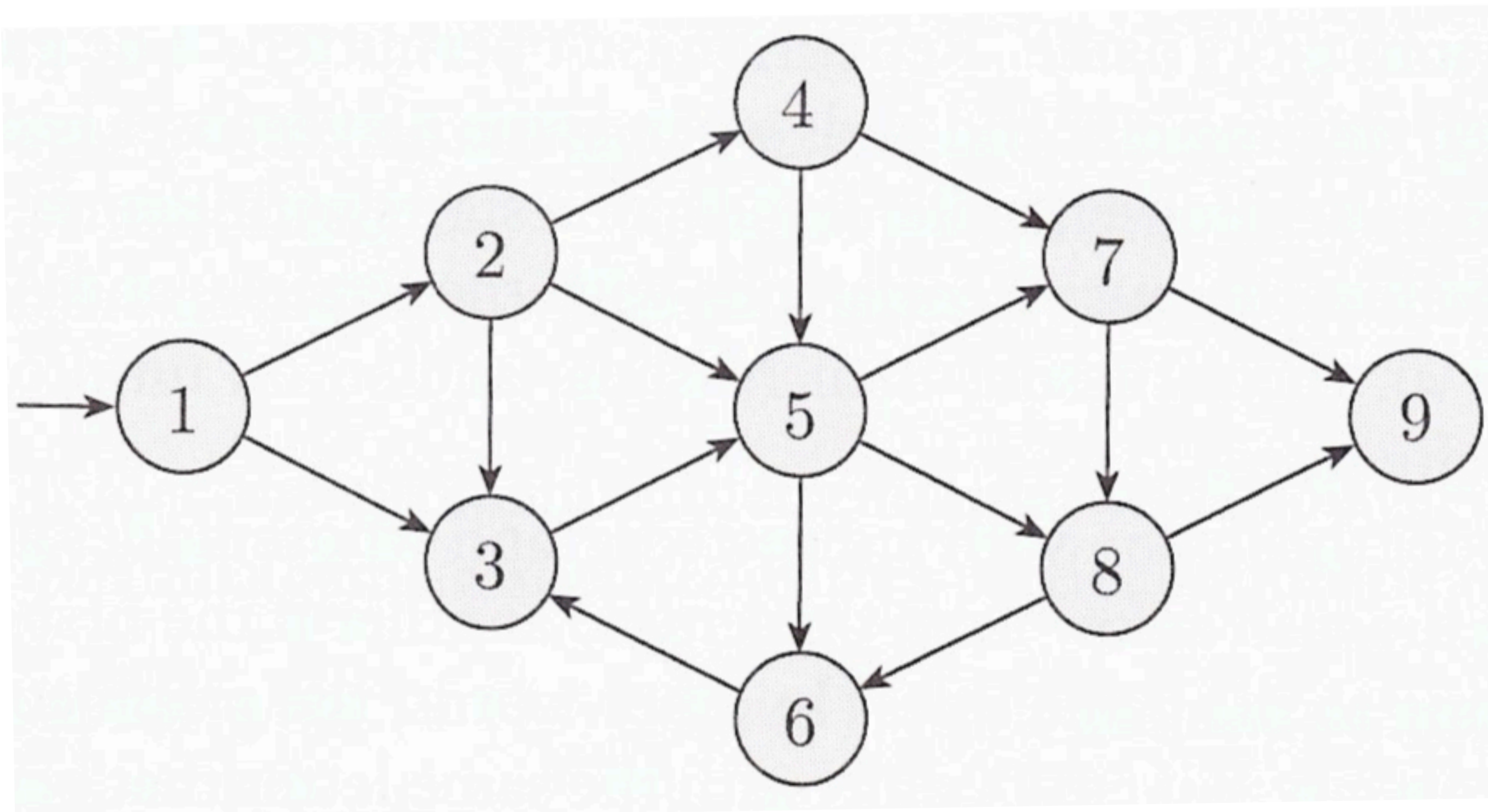
A key point: formula is poly computable (e.g., poly length)

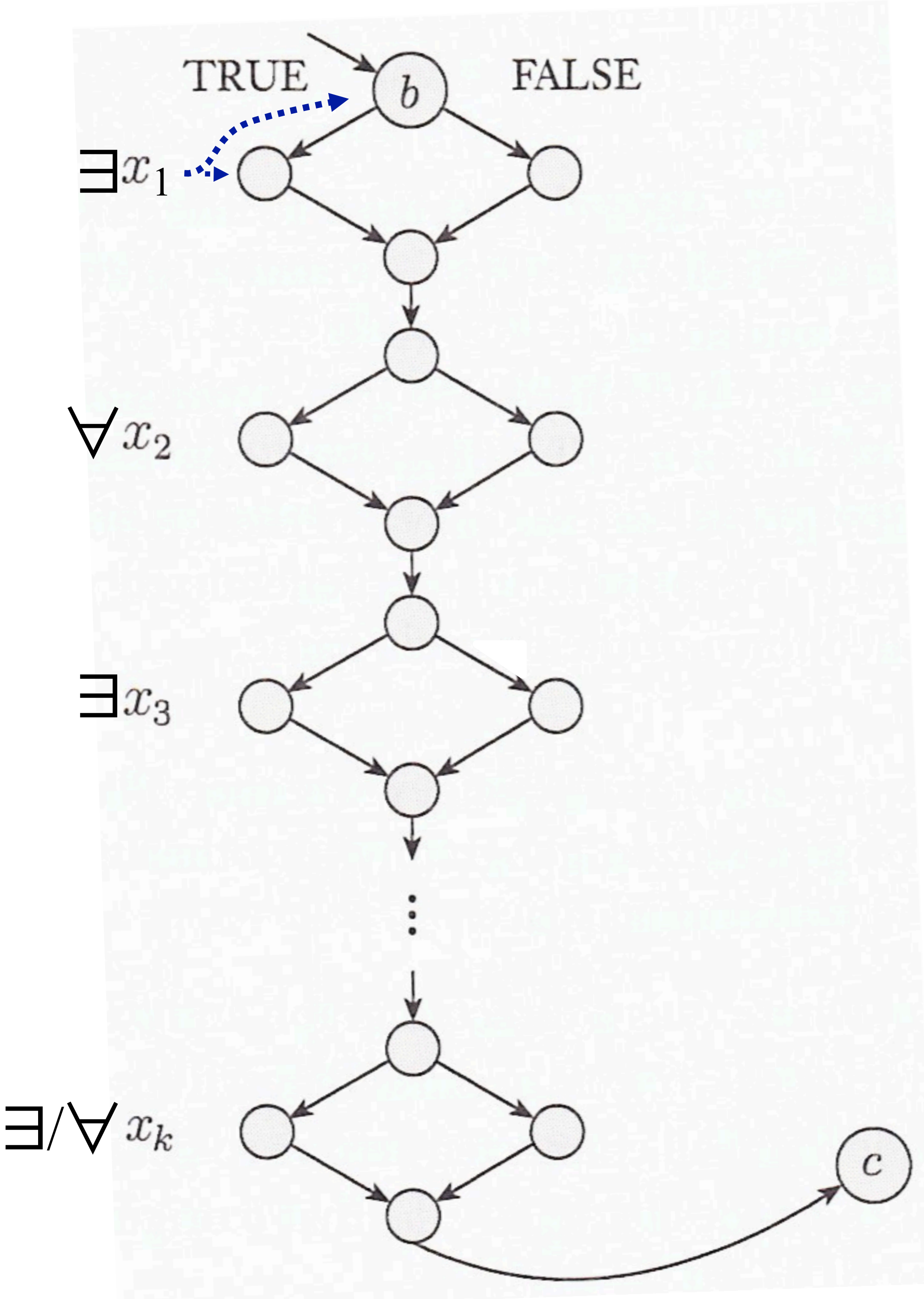
“Geography”



Peoria
Austin
Nashua
Albany
Amherst
Tucson
Tokyo
Orsay
Oakland

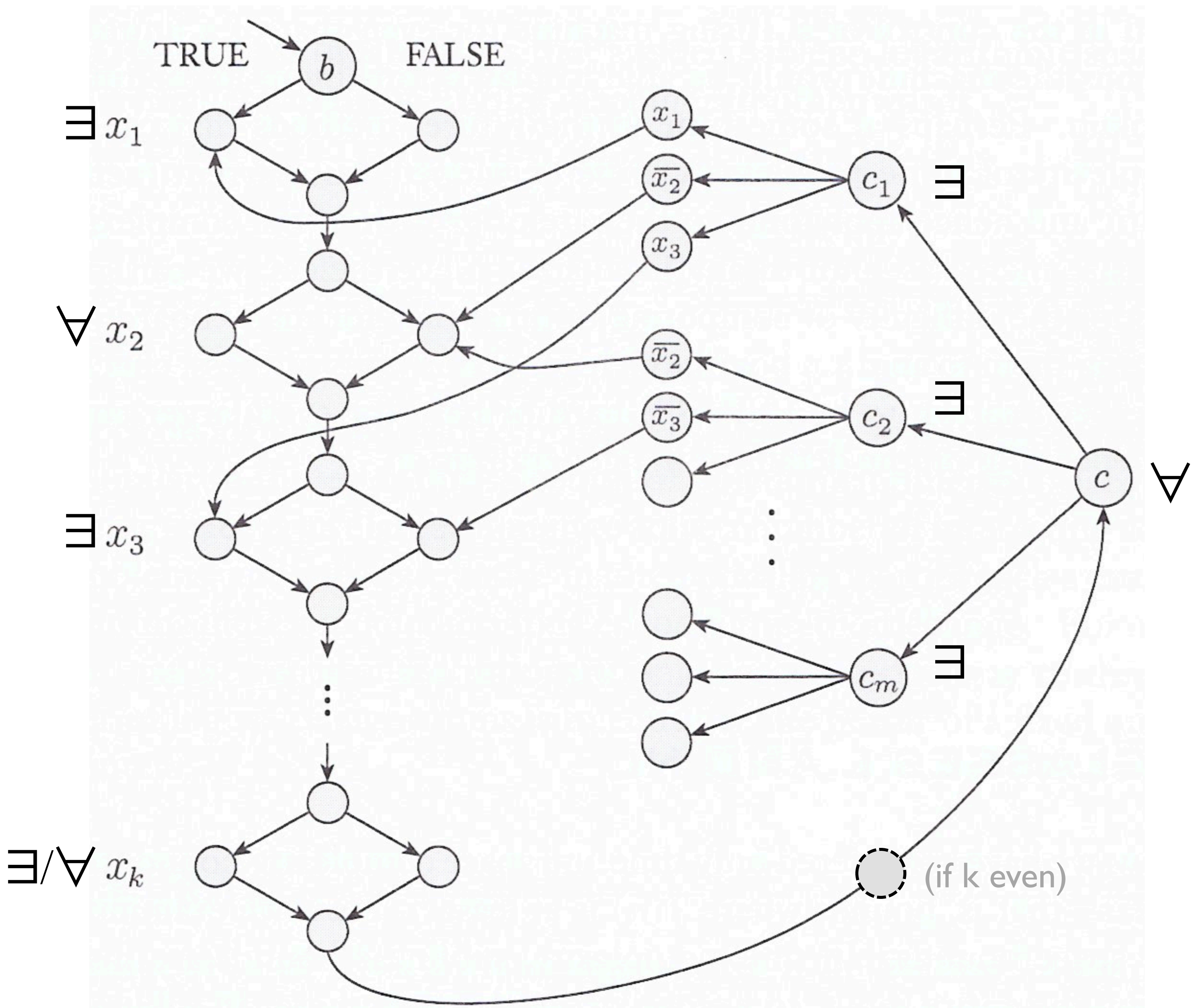
“Generalized Geography”





TQBF $\leq_p$ Generalized Geography

And so GGEO is PSPACE-complete



$$\phi = \exists x_1 \forall x_2 \cdots \mathsf{Q} x_k \left[(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee \cdots) \wedge \cdots \wedge (\qquad)\right]$$

SPACE: Summary

Defined on TMs (as usual) but largely model-independent

Time T $\subseteq$ Space T $\subseteq$ Time 2^{cT}

Cor: NP $\subseteq$ PSPACE

Savitch: Nspace(S) $\subseteq$ Dspace(S^2)

Cor: Pspace = NPspace (!)

TQBF is PSPACE-complete (analog: SAT is NP-complete)

PSPACE and games (and games have serious purposes: auctions, allocation of shared resources, hacker vs firewall,…)

An Analogy

NP is to PSPACE as Solitaire is to Chess

I.e., NP probs involve finding a solution to a fixed, static puzzle with no adversary other than the structure of the puzzle itself

PSPACE problems, of course, just plain use poly space. But they often involve, or can be viewed as, games where an interactive adversary dynamically thwarts your progress towards a solution

The former, tho hard, seems much easier than the later–part of the reason for the (unproven) supposition that NP $\subsetneq$ PSPACE