6.045

Lecture 21:
Finish PSPACE,
Randomized Complexity

TQBF = { ϕ | ϕ is a true quantified Boolean formula }

Theorem: TQBF is PSPACE-Complete

FG = { ϕ | ϕ is a QBF and Player E has a winning strategy in the Formula Game on ϕ }

Theorem: FG = TQBF

The Geography Game

Two players take turns naming cities from anywhere in the world

Each city chosen must begin with the same letter that the previous city ended with

Austin → Newark → Kalamazoo → Opelika

Cities cannot be repeated

Whenever someone can no longer name any more cities, they lose and the other player wins

Generalized Geography

Geography played on a directed graph

Nodes represent cities. Edges represent moves.
An edge (a,b) means: *"if the current city is a, then a player could choose city b next"*

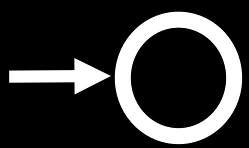
But cities cannot be repeated!
Each city can be visited at most once

Whenever a player cannot move to any adjacent city, they are "stuck"– they lose and the other player wins

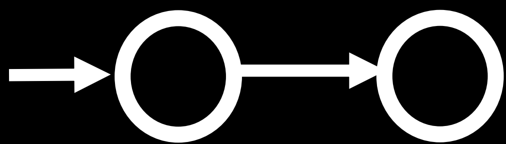
Given a graph and a node a,
does Player 1 have a winning strategy starting from a?

Like a two-player Hamiltonian path problem!

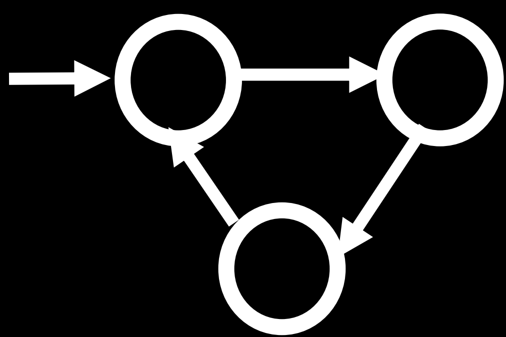
Generalized Geography: Simple Examples



Player 2 always wins:
Player 1 must go first and has no edge to take!



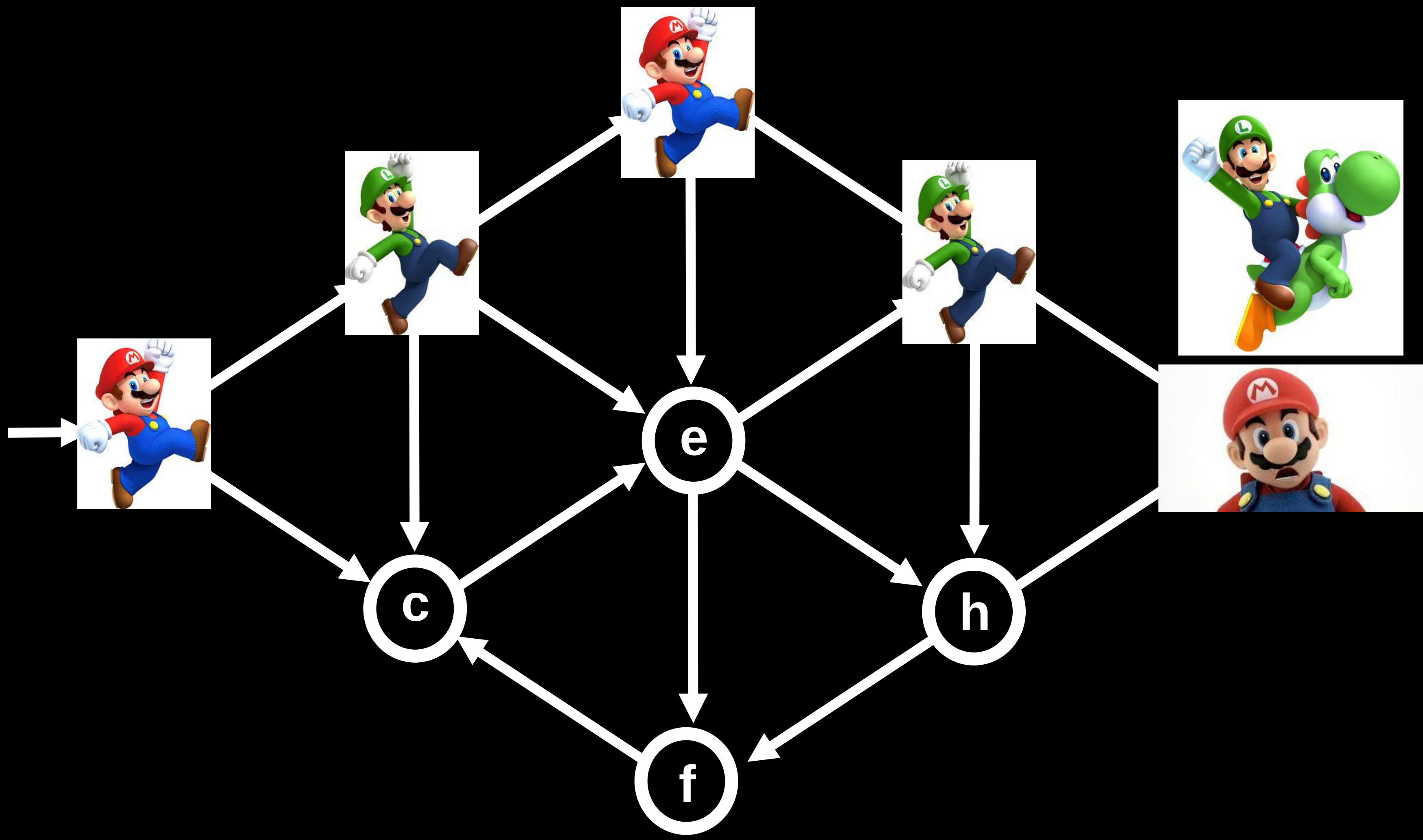
Player 1 always wins: Player 1 can take the first edge, Player 2 is stuck



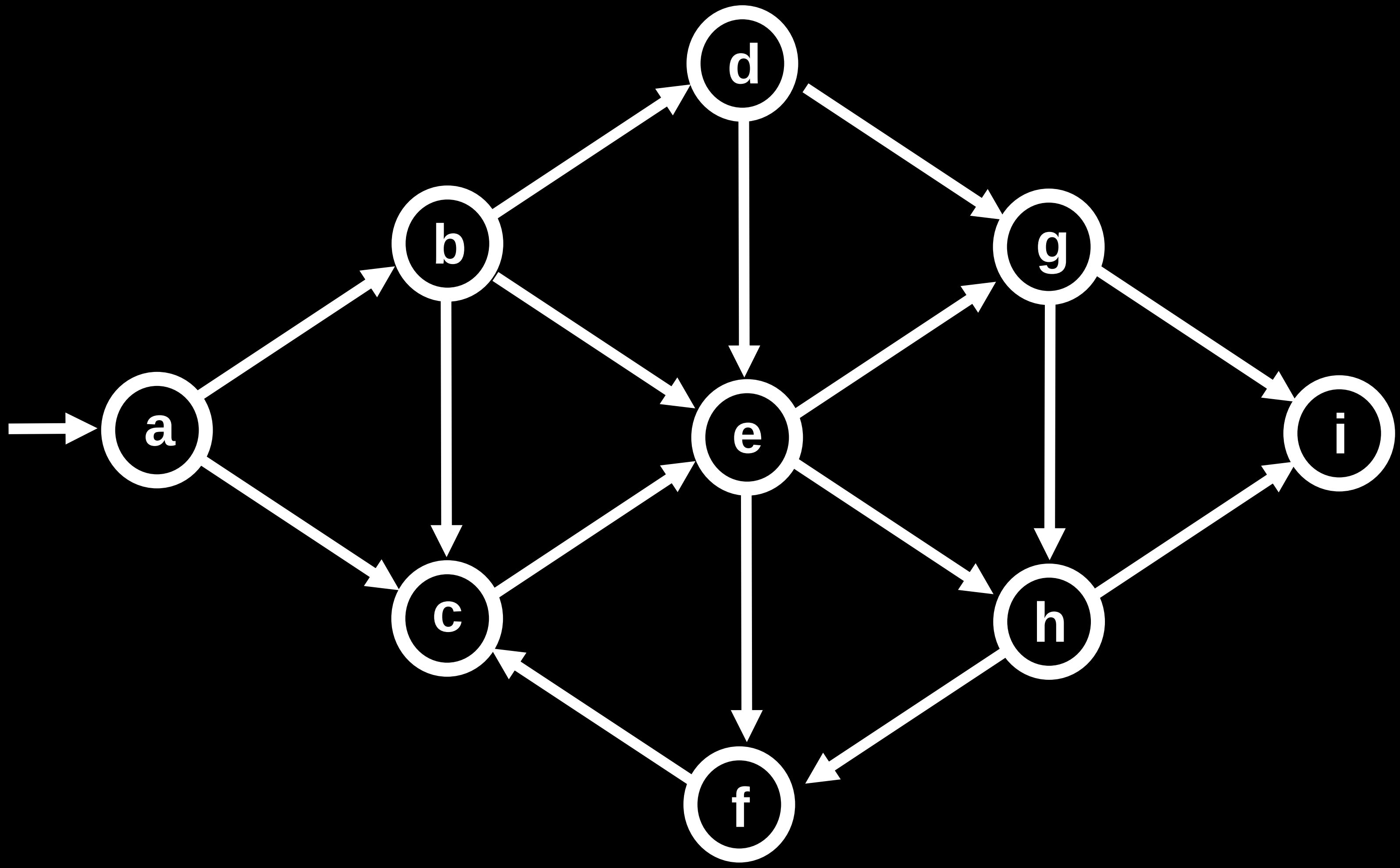
Player 2 always wins

Claim: For every graph, (exactly) one player has a winning strategy

Generalized Geography

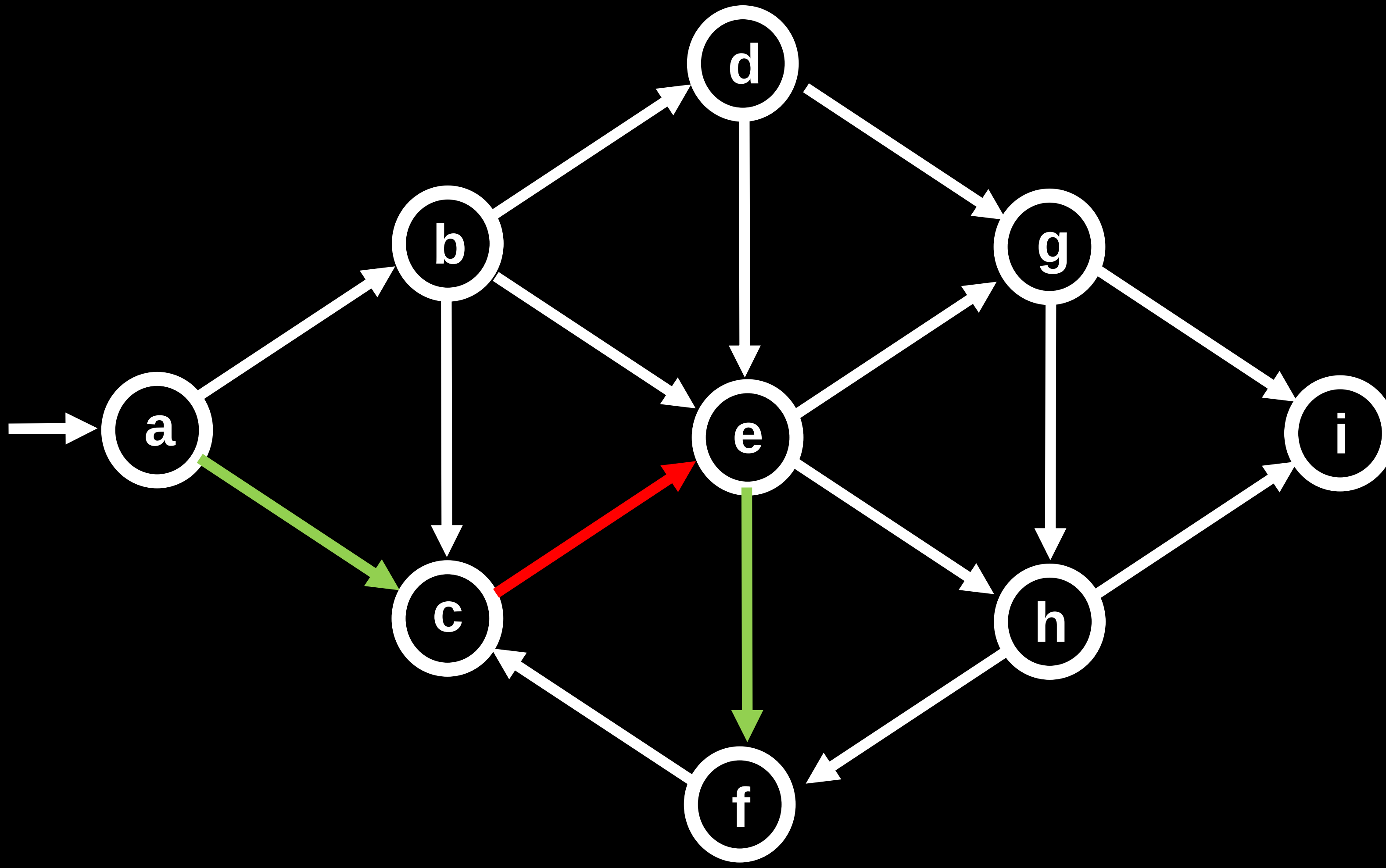


Generalized Geography



Who has a winning strategy in this game?

Generalized Geography



Player 1 has a winning strategy!

GG = { (G, a) | Player 1 has a winning strategy for geography on graph G starting at node a }

Theorem: GG is PSPACE-Complete

GG ∈ PSPACE

Want: PSPACE machine GGM that accepts (G,a)

$\Leftrightarrow$ Player 1 has a winning strategy on (G,a)

GGM(G, a): If node a has no outgoing edges, *reject*

Remove node a and all adjacent edges,

getting a smaller graph G_{-a}

For all nodes $a_1, a_2, \ldots, a_k$ that node a pointed to,

Recursively call GGM(G_{-a}, a_i).

If all k calls accept, then *reject* else *accept*

Claim: All of the k calls accept

$\Leftrightarrow$ Player 2 has a winning strategy!

Idea: Each rec. call "reverses the roles" of the players!

GG ∈ PSPACE

Want: PSPACE machine GGM that accepts (G,a)
⇔ Player 1 has a winning strategy on (G,a)

GGM(G, a): If node a has no outgoing edges, *reject*

Remove node a and all adjacent edges,
getting a smaller graph G_{-a}

For all nodes $a_1, a_2, \ldots, a_k$ that node a pointed to,
Recursively call GGM(G_{-a}, a_i).

If all k calls accept, then *reject* else *accept*

Claim: On graphs of n nodes, GGM takes $O(n^2)$ space
(only have to store a subset of nodes at each level of recursion, and there are n levels of recursion)

GG is PSPACE-hard

We show that FG $\leq_P$ GG

Convert a quantified formula ϕ into (G, a) such that:

Player E has winning strategy in ϕ (ϕ is true)

if and only if

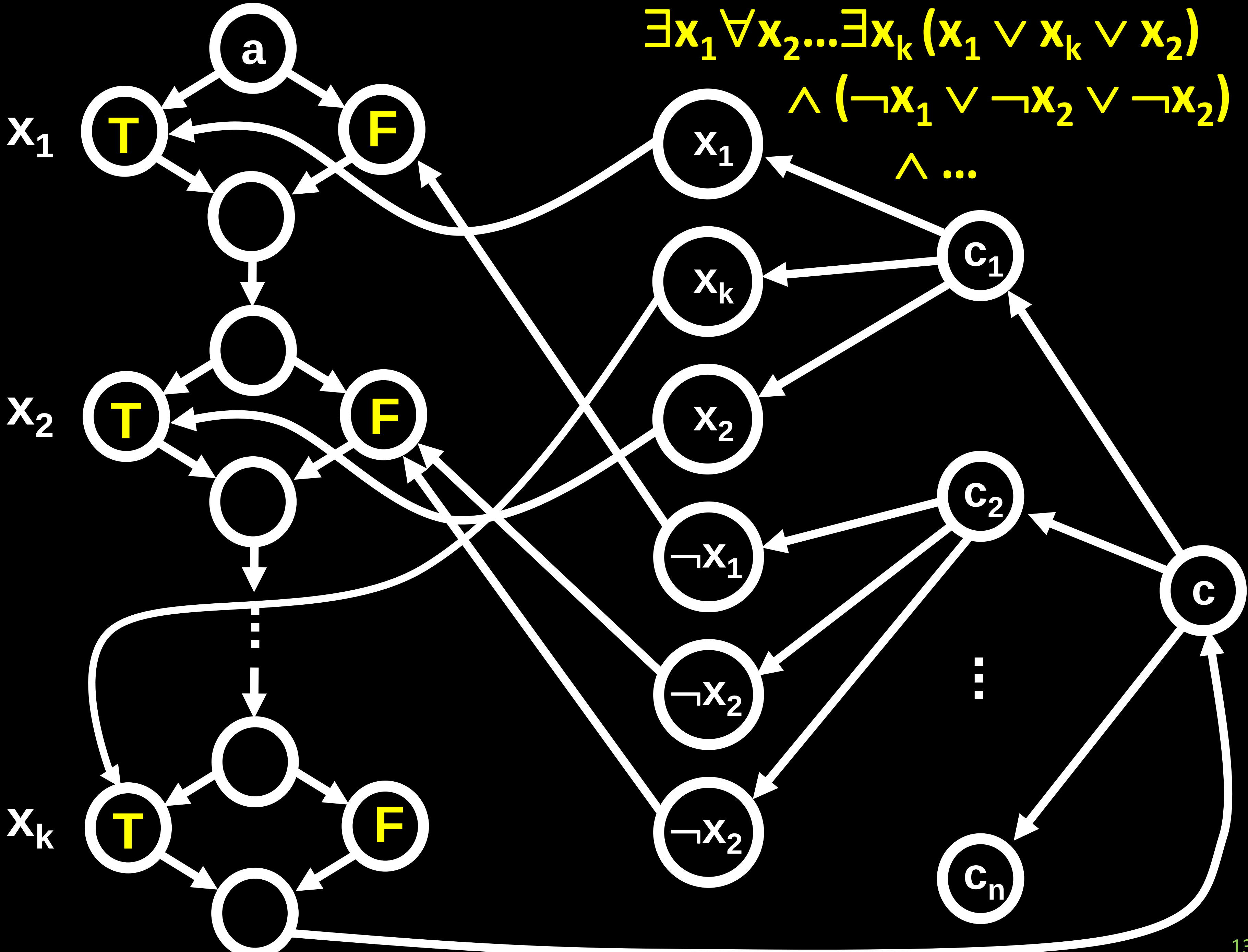
Player 1 has winning strategy in (G, a)

For simplicity we assume ϕ is of the form:

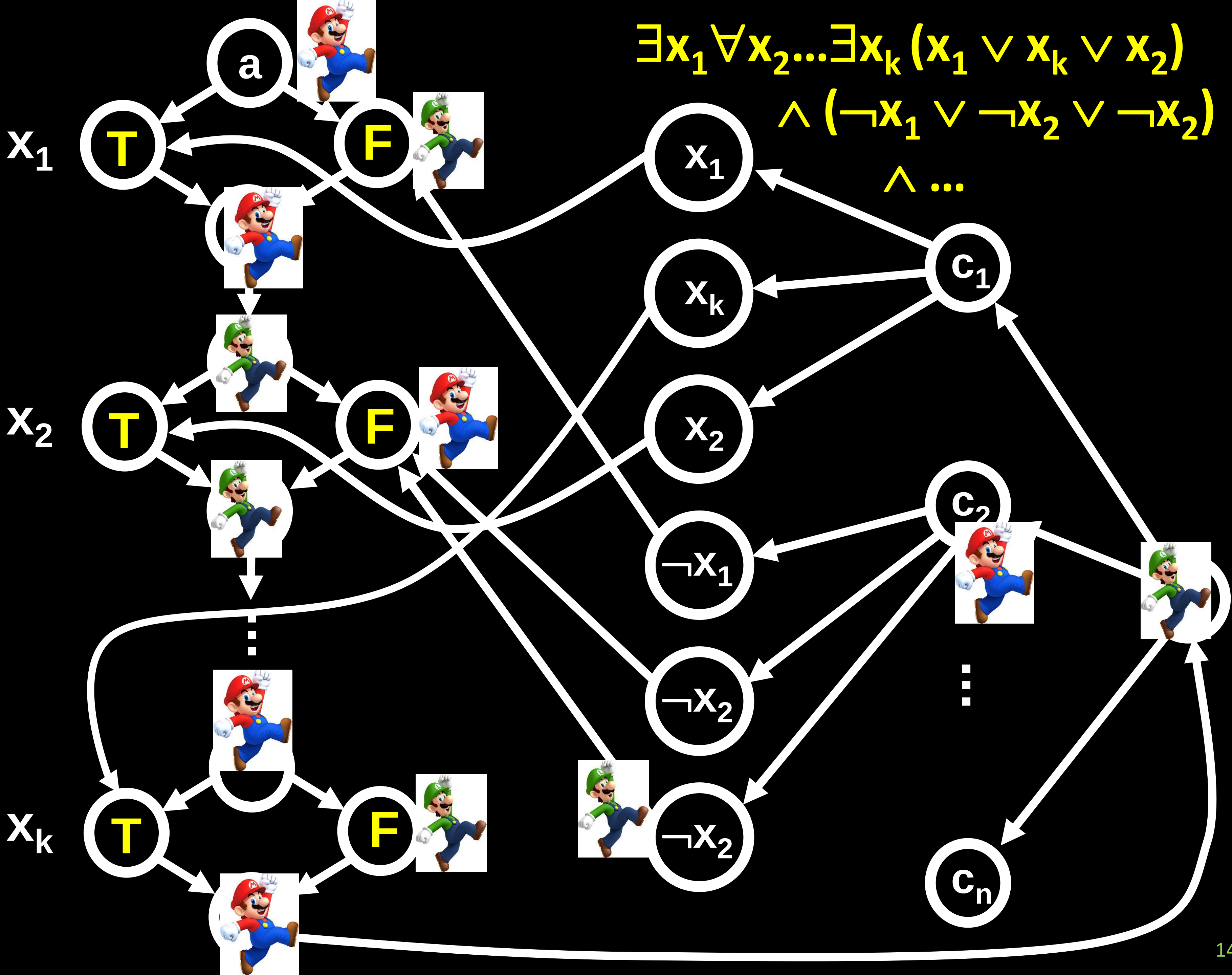
$$\phi = \exists x_1 \forall x_2 \exists x_3 \ldots \exists x_k \,[F]$$

where F is in CNF: an AND of ORs of literals.
(Quantifiers alternate, and first & last move is E's)

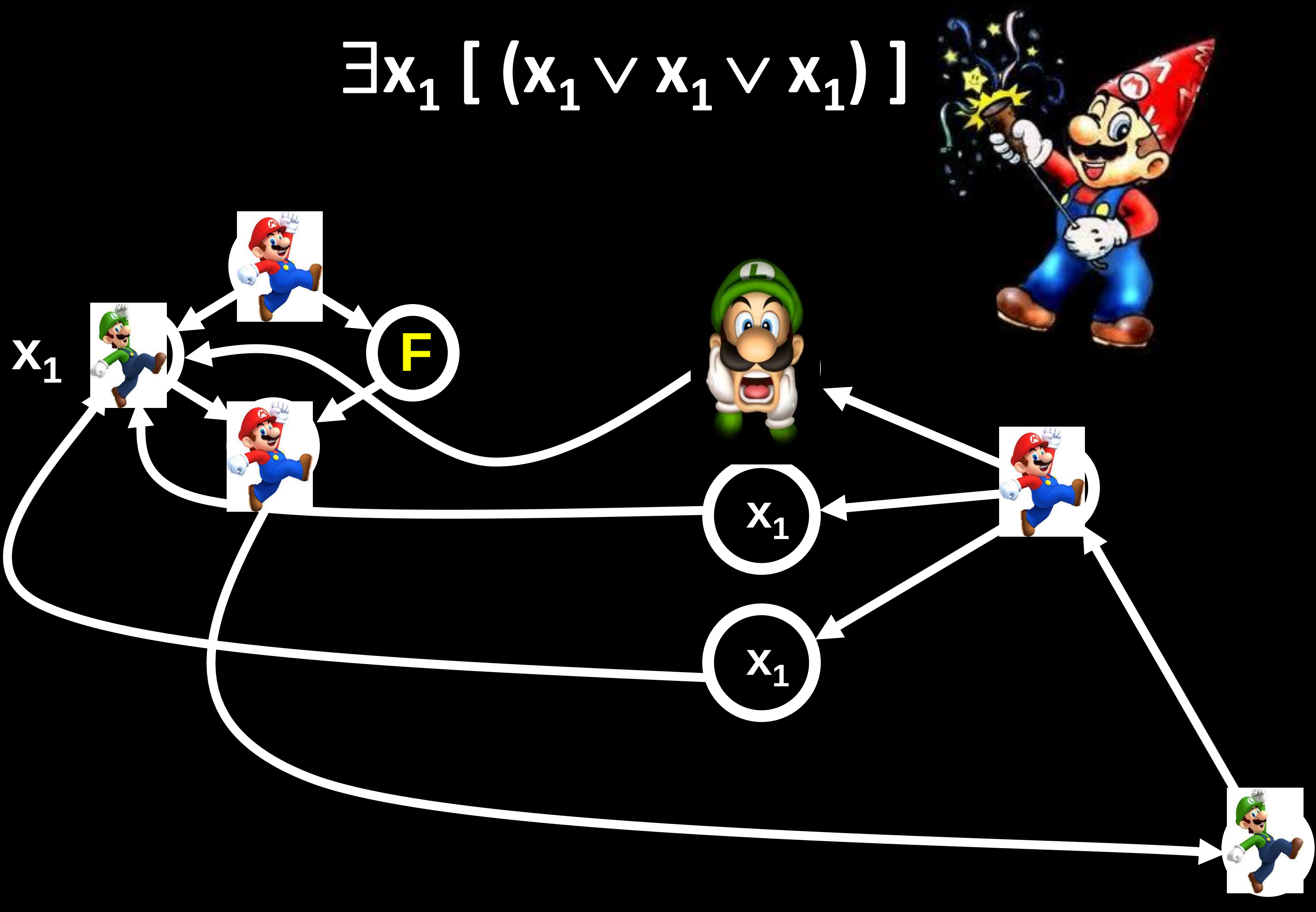
$$\exists x_1 \forall x_2 \ldots \exists x_k (x_1 \vee x_k \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge \ldots$$



$$\exists x_1 \forall x_2 \ldots \exists x_k \, (x_1 \vee x_k \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge \ldots$$



$$\exists x_1 \, [\, (x_1 \vee x_1 \vee x_1) \,]$$



GG = { (G, a) | Player 1 has a winning strategy for geography on graph G starting at node a }

Theorem: GG is PSPACE-Complete

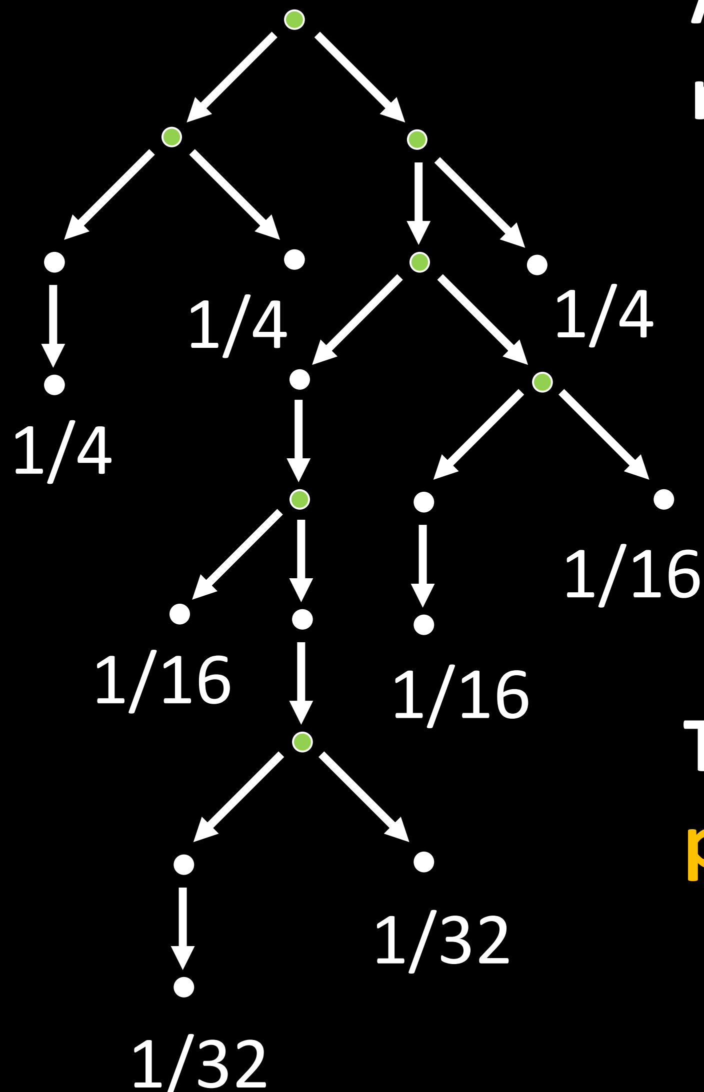
Question:
Is Chess a PSPACE-complete problem?

No, because determining whether a player has a winning strategy takes CONSTANT time and space *(OK, the constant is large…)*

But *generalized* versions of Chess, Go, Hex, Checkers, etc. (on n x n boards) can be shown to be PSPACE-hard

Randomized / Probabilistic Complexity

Probabilistic TMs



A **probabilistic TM M** is a nondeterministic TM where:

- Each nondeterministic step is called a **coin flip**
- Each nondeterministic step has only two legal next moves **(heads or tails)**

The probability that M runs on a **path p** is: $\Pr[\,p\,] = 2^{-k}$

where k is the number of coin flips that occur on path p

Probabilistic/Randomized Algorithms

Why study randomized algorithms?

1. **They can be simpler than deterministic algorithms**
2. **They can be more efficient than deterministic algorithms**
3. **Can randomness be used to solve problems *provably* much faster than deterministic algorithms?**

This is an open question!

$$\Pr[\, M \text{ accepts } w \,] = \sum_{\substack{p \text{ is a path on which} \\ M \text{ on } w \textit{ accepts}}} \Pr[\, p \,]$$

Theorem: A language A is in NP if there is a nondeterministic polynomial time TM M such that for all strings w:

$$w \in A \Rightarrow \Pr[\, M \text{ accepts } w \,] > 0$$

$$w \notin A \Rightarrow \Pr[\, M \text{ accepts } w \,] = 0$$

Theorem: A language A is in coNP if there is a nondeterministic polynomial time TM M such that for all strings w:

$$w \in A \Rightarrow \Pr[\ M \text{ accepts } w\ ] = 0$$

$$w \notin A \Rightarrow \Pr[\ M \text{ accepts } w\ ] > 0$$

Theorem: A language A is in NP if there is a nondeterministic polynomial time TM M such that for all strings w:

$$w \in A \Rightarrow \Pr[\ M \text{ accepts } w\ ] > 0$$

$$w \notin A \Rightarrow \Pr[\ M \text{ accepts } w\ ] = 0$$

Definition. A probabilistic TM M decides a language A with error ε if for all strings w,

$$w \in A \Rightarrow \Pr[\, M \text{ accepts } w \,] \geq 1 - \varepsilon$$

$$w \notin A \Rightarrow \Pr[\, M \text{ doesn't accept } w \,] \geq 1 - \varepsilon$$

Error Reduction Lemma

Lemma: Let ε be a constant, $0 < \varepsilon < 1/2$, let $k \in \mathbb{N}$.

If M_1 has error $1/2-\varepsilon$ and runs in $t(n)$ time

then there is an equivalent machine M_2 such that

M_2 has error $< 1/2^{n^k}$ and runs in $O(n^k \cdot t(n)/\varepsilon^2)$ time

Proof Idea:

On input w, M_2 runs M_1 on w for $m = 10\, n^k/\varepsilon^2$ random independent trials, records the m answers of M_1 on w, returns most popular answer (accept or reject)

Can use Chernoff Bound to show the error is $< 1/2^{n^k}$

Probability that the Majority answer over $10m/\varepsilon^2$ trials is *different* from the $1/2+\varepsilon$ prob event is $< 1/2^m$

Error Reduction Lemma

Lemma: Let ε be a constant, $0 < \varepsilon < 1/2$, let $k \in \mathbb{N}$.

If M_1 has error $1/2-\varepsilon$ and runs in $t(n)$ time

then there is an equivalent machine M_2 such that

M_2 has error $< 1/2^{n^k}$ and runs in $O(n^k \cdot t(n)/\varepsilon^2)$ time

Proof Idea:

On input w, M_2 runs M_1 on w for $m = 10\, n^k/\varepsilon^2$ random independent trials, records the m answers of M_1 on w, returns most popular answer (accept or reject)

Define indicator $X_i = 1$ iff M_1 outputs right answer in trial i

Set $X = \sum_i X_i$. Then $E[X] = \sum_i E[X_i] \geq (1/2+\varepsilon)m$

Show: $\Pr[M_2(w) \text{ is wrong}] = \Pr[X < m/2] < 1/2^{\varepsilon^2 m/10}$

BPP = Bounded Probabilistic P

BPP = { L | L is recognized by a probabilistic polynomial-time TM with error at most **1/3** }

Why 1/3?

It doesn't matter what error value we pick, as long as the error is smaller than $1/2 - 1/n^k$ for some constant k

When the error is smaller than $1/2$, we can apply the error reduction lemma and get $1/2^{n^c}$ error

Checking Matrix Multiplication

CHECK = { (M_1,M_2,N) | M_1, M_2 and N are matrices and $M_1 \cdot M_2 = N$ }

If M_1 and M_2 are n x n matrices, computing $M_1 \cdot M_2$ takes $O(n^3)$ time normally,
and $O(n^{2.373})$ time using very sophisticated methods.

Here is an $O(n^2)$-time randomized algorithm for CHECK:

Pick a 0-1 bit vector r at *random*, test if $M_1 \cdot M_2 r = Nr$

Claim: If $M_1 \cdot M_2 = N$, then $\Pr[M_1 \cdot M_2 r = Nr] = 1$
If $M_1 \cdot M_2 \neq N$, then $\Pr[M_1 \cdot M_2 r = Nr] \leq 1/2$

If we pick 20 random vectors and test them all, what is the probability of incorrect output?

Checking Matrix Multiplication

CHECK = { (M_1, M_2, N) | M_1, M_2 and N are matrices and $M_1 \cdot M_2 = N$ }

Pick a 0-1 bit vector r at *random*, test if $M_1 \cdot M_2 r = Nr$

Claim: If $M_1 \cdot M_2 \neq N$, then $\Pr[M_1 \cdot M_2 r = Nr] \leq 1/2$

Proof: Define $M' = N - (M_1 \cdot M_2)$. M' is a non-zero matrix.

Some row M'_i is non-zero, some entry $M'_{i,j}$ is non-zero.

Want to show: $\Pr[M'r = \vec{0}] \leq 1/2$

We have: $\Pr[M'r = \vec{0}] \leq \Pr[\langle M'_i, r\rangle = 0]$

$= \Pr[\sum_k M'_{i,k} \cdot r_k = 0]$ (def of inner product)

$= \Pr[-r_j = (\sum_{k \neq j} M'_{i,k} \cdot r_k)/M'_{i,j}] \leq 1/2$

Why $\leq 1/2$? After everything else is assigned on RHS, there is at most one value of r_k that satisfies the equation!

An arithmetic formula is like a Boolean formula, except it has +, –, and * instead of OR, NOT, AND.

ZERO-POLY = { p | p is an arithmetic formula that is *identically* zero}

Identically zero means: all coefficients are 0

Two examples of formulas in ZERO-POLY:

$(x + y)\cdot(x + y) - x\cdot x - y\cdot y - 2\cdot x\cdot y$

Abbreviate as: $(x + y)^2 - x^2 - y^2 - 2xy$

$(x^2 + a^2)\cdot(y^2 + b^2) - (x\cdot y - a\cdot b)^2 - (x\cdot b + a\cdot y)^2$

There is a rich history of polynomial identities in mathematics. Useful also in program testing!

Testing Univariate Polynomials

Let p(x) be a polynomial in one variable over Z

$$p(x) = a_0 + a_1x + a_2x^2 + \dots + a_dx^d$$

Suppose p is hidden in a "black box" –
we can only see its inputs and outputs.
Want to determine if p is *identically* 0

Simply evaluate p on d+1 distinct values!
Non-zero degree d polynomials have ≤ d roots.
But the *zero polynomial* has every value as a root.