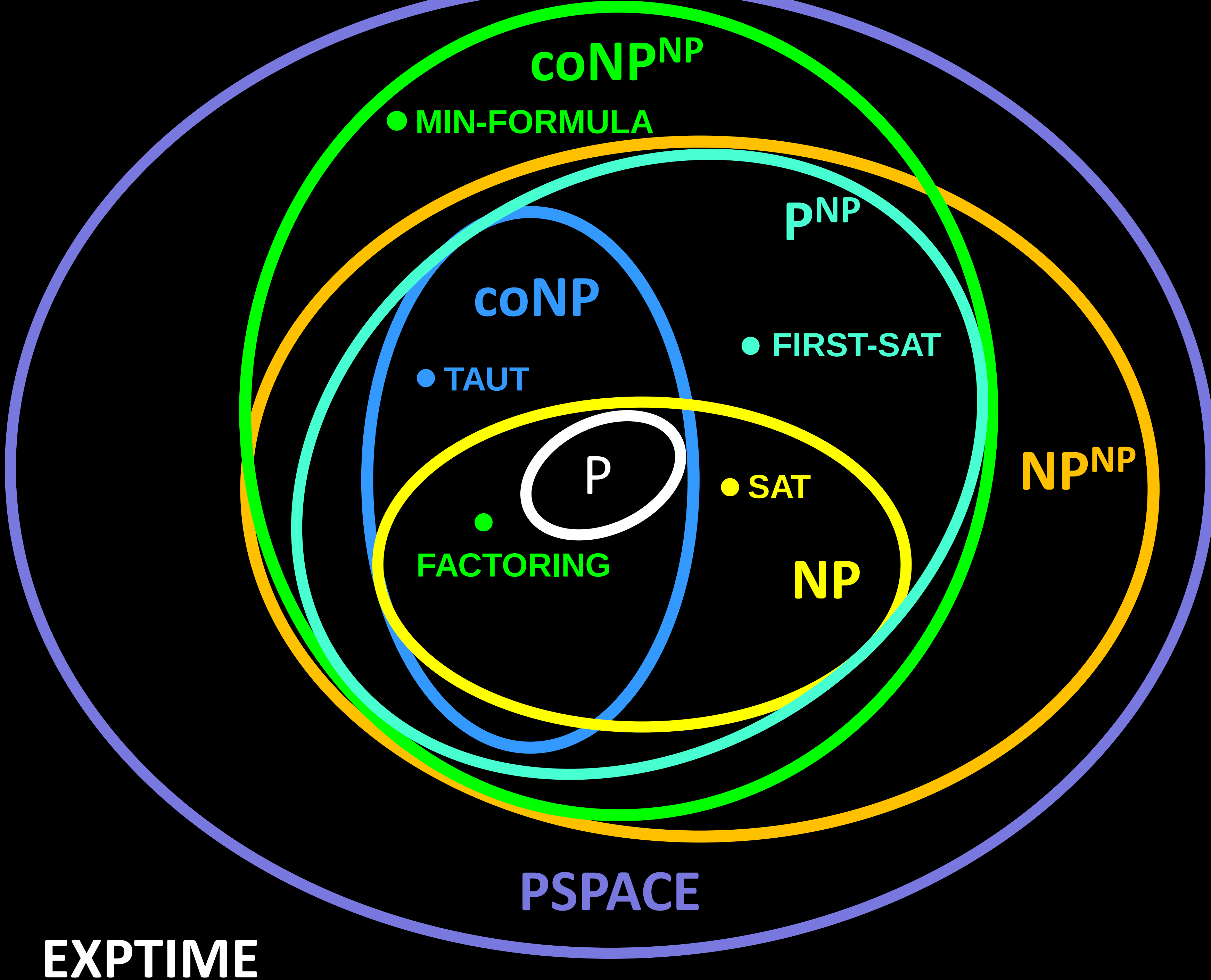
6.045

Lecture 20: PSPACE-Complete problems, Complexity as Games


$coNP^{NP}$
MIN-FORMULA
P^{NP}
coNP
FIRST-SAT
TAUT
P
SAT
NP^{NP}
FACTORING
NP
PSPACE
EXPTIME

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

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

Last time: Savitch's Theorem

$\Rightarrow$ **PSPACE = NPSPACE!**

PSPACE-complete problems

Definition: Language B is **PSPACE-complete** if:

1. B ∈ PSPACE
2. Every A in PSPACE is **poly-time reducible** to B

(i.e. B is PSPACE-hard)

Why poly-time?

Theorem: If B is **PSPACE-complete** and B is in **P** then **P = PSPACE**

Idea: Let A ∈ PSPACE. Our poly-time TM for A first reduces its input x to an instance y of B. Then it runs the poly-time TM for B on y, and outputs its answer.

Definition: Language B is **PSPACE-complete** if:

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

Why poly-time?

Theorem: If B is **PSPACE-complete** and B is in **NP** then **NP = PSPACE**

Idea: Let A ∈ PSPACE. Our NP machine for A reduces its input x to an instance y of B. Then it runs a nondeterministic poly-time TM for B on y, and outputs its answer.

Definition:

A **fully quantified Boolean formula** is a Boolean formula where *every* variable in the formula is quantified ($\exists$ or $\forall$) at the beginning the formula.

These formulas are either **true or false**

$$\exists x \exists y\, [\, x \lor \neg y\,]$$

$$\forall x\, [\, x \lor \neg x\,]$$

$$\forall x\, [\, x\,]$$

$$\forall x \exists y\, [\, (x \lor y) \land (\neg x \lor \neg y)\,]$$

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

- SAT is the special case where all quantifiers on all variables are ∃

- TAUTOLOGY is the special case where all quantifiers are ∀

So, SAT $\leq_P$ TQBF and TAUTOLOGY $\leq_P$ TQBF

Theorem (Meyer-Stockmeyer): TQBF is PSPACE-complete

TQBF is in PSPACE

QBF-SOLVER(ϕ):

1. If ϕ has no quantifiers, then it is an expression with only constants. Evaluate ϕ. **Accept** iff ϕ evaluates to 1.

2. If $\phi = \exists x\ \psi$, call QBF-SOLVER on ψ twice: first with x set to 0, then with x set to 1. **Accept** iff *at least* one call accepts.

3. If $\phi = \forall x\ \psi$, call QBF-SOLVER on ψ twice: first with x set to 0, then with x set to 1. **Accept** iff *both* calls accept.

Why does this take polynomial space?

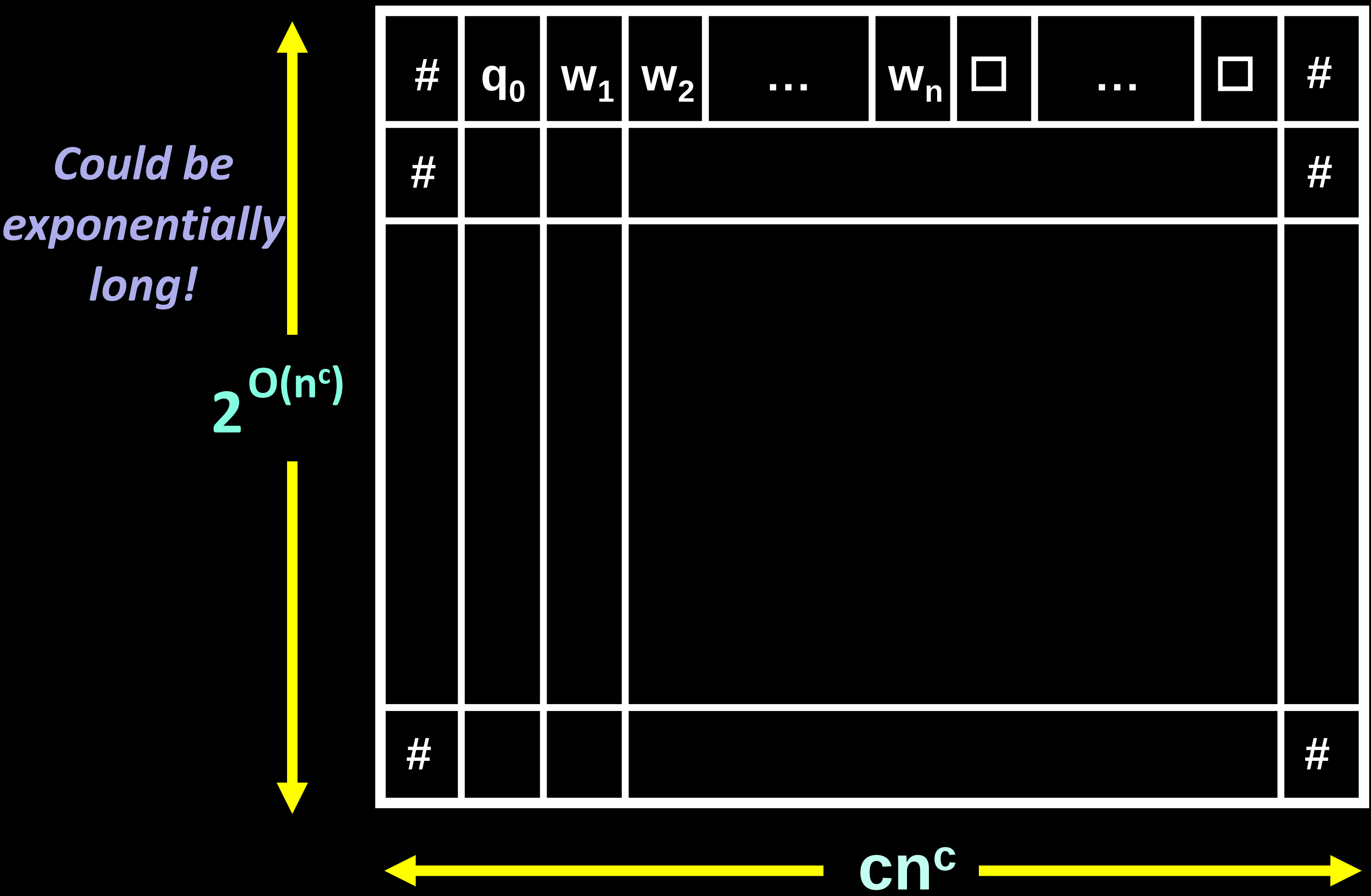
TQBF is PSPACE-hard: Every language A in PSPACE is polynomial time reducible to TQBF

We'll outline a proof of this. **The missing details aren't necessary, but *please* ask questions!**

For every language A is in PSPACE, there is some k and some deterministic TM M that decides A using space $\leq cn^c$

Our polynomial-time reduction will map every string w to a fully quantified Boolean formula ϕ of $O(n^{2c})$ size that *simulates* M on w

A **tableau for M on w** is an table whose rows are the configurations of M on input w



Fix M and w. We'll construct a QBF ϕ that is true if and only if M accepts string w of length n

Let $s(n) := cn^c$.

There is a $b \geq 1$ such that each configuration C of M on w can be written as a $b \cdot s(n)$ bit string $C = C_1 \cdots C_{b \cdot s(n)}$

For integers $k \geq 0$, we'll construct QBF $\phi_k(C,D)$

For all strings C,D, $\phi_k(C,D)$ is true if and only if M starting in config C reaches config D in $\leq 2^k$ steps

Then we'll set $\phi := \phi_{b\,s(n)}(C_{start},C_{acc})$, where

C_{start} = initial configuration of M on w,
C_{acc} = (unique) accepting configuration of M

Why would k = b s(n) suffice?

IDEA:

Encode Savitch's theorem in Logic!

$\exists$ guess the configuration in the "middle" of the computation, and use recursion and $\forall$ quantifiers to write the acceptance condition as a poly-sized QBF!

For two configurations C and D of our TM,
ϕ_k(C,D) will be true if and only if
C reaches D after $\leq 2^k$ steps.

ϕ_k(C,D) $\coloneqq$ "there exists a configuration E such that ϕ_{k-1}(C,E) is true and ϕ_{k-1}(E,D) is true"

Goal: If M uses n^c space on inputs of length n, then our final QBF ϕ will have size $O(n^{2c})$

If k = 0, then ϕ_k(C,D) should look like:

ϕ_0(C,D) = "C equals D" **OR**
"D follows from C in a single step of M"

How do we logically express "C equals D"?
Write a Boolean formula saying that the block of **b s(n) variables** representing config C *equals* the block of **b s(n) variables** representing config D

$$\wedge_{i=1}^{b\,s(n)} (C_i = D_i) \equiv \wedge_i ((C_i \vee \neg D_i) \wedge (\neg C_i \vee D_i))$$

"D follows from C in a single step of M"?
Use 2 x 3 windows as in the Cook-Levin theorem:
"For all 2 x 3 windows W between C and D, and for all illegal windows W', (W $\neq$ W')"

For $k > 0$, let's try to construct ϕ_k recursively:

$$\phi_k(C,D) = \exists E\ [\phi_{k-1}(C,E) \wedge \phi_{k-1}(E,D)]$$

$\exists e_1 \exists e_2 \ldots \exists e_S$ where $S = b\, cn^c$

But how long is this formula??

It will be of length $\geq 2^k$. Every level of the recursion reduces k by 1, but roughly *doubles* the formula size!

We can get around this. Modify the formula to be:

$$\phi_k(C,D) = \exists E\ \forall X,Y\ [(\ (X,Y)=(C,E) \vee (X,Y)=(E,D)\ ) \Rightarrow \phi_{k-1}(X,Y)\ ]$$

This "folds" the two recursive sub-formulas into one!

$\phi_k(C,D) = \exists E\ \forall X,Y\ [(\ (X,Y)=(C,E) \vee (X,Y)=(E,D)\ )$
$\Rightarrow\ \phi_{k-1}(X,Y)\ ]$

Set $\phi = \phi_h(C_{start}, C_{acc})$ where $h = b\ s(n)$

ϕ is true $\Leftrightarrow$ On w, reach C_{acc} from C_{start} in $\leq 2^{b\ s(n)}$ steps
$\Leftrightarrow$ M accepts w

Each recursive step in ϕ_k adds a subformula of size O(s(n))

The size of ϕ_k satisfies the recurrence

$$size(k) \leq size(k-1) + O(s(n)),\quad size(0) \leq O(s(n))$$

which solves to $size(k) \leq O(k\ s(n))$

Number of levels of recursion in ϕ is $h \leq O(s(n))$
Therefore the size of ϕ is $O(s(n)^2)$

Complexity Theory as Games

NP captures many "one-player" games with perfect information

Example 1: Generalized versions of many games Super Mario, Donkey Kong, Legend of Zelda, etc. are NP-hard

https://arxiv.org/pdf/1203.1895v1.pdf

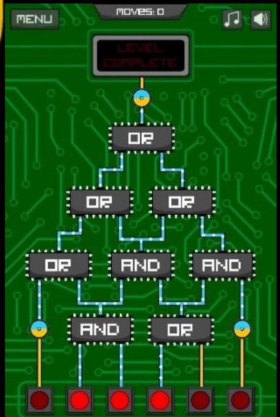
In particular, it is NP-hard to tell if you can finish an arbitrary level of these games!

Complexity Theory as Games

NP captures many "one-player" games with perfect information

Example 2: There are Android games which are *literally* the Circuit-SAT problem!

See Circuit Scramble and Make It True

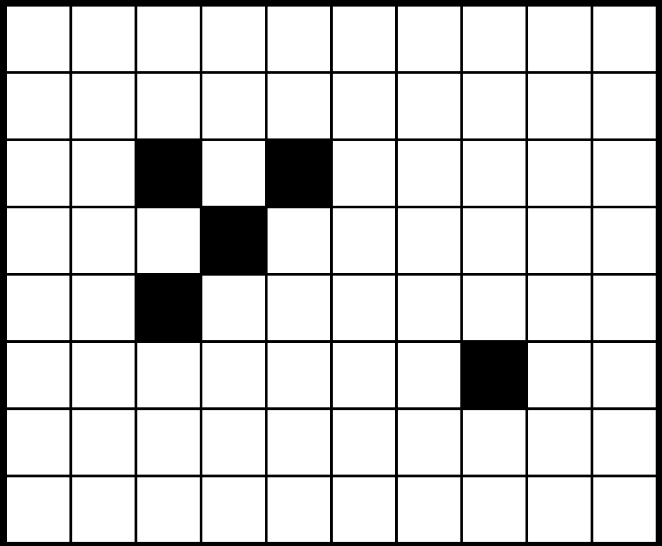


Complexity Theory as Games

P captures short "zero-player" games

(Letting this game play out by itself, will it lead to a "win" or not?)

Example of a Zero-Player Game: Conway's Game of Life



Played on an infinite 2d grid

Each cell is "alive" or "dead"

In one step of the game:

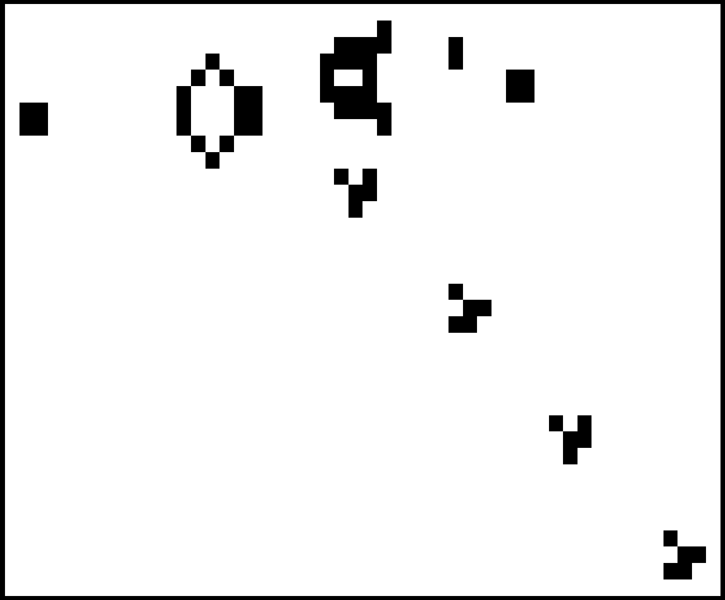
- Any live cell with 2 or 3 live neighbors remains live
- Any dead cell with 3 live neighbors becomes live
- All other cells are dead

Complexity Theory as Games

P captures short "zero-player" games

(Letting this game play out by itself, will it lead to a "win" or not?)

Example of a Zero-Player Game: Conway's Game of Life



Played on an infinite 2d grid

Each cell is "alive" or "dead"

In one step of the game:

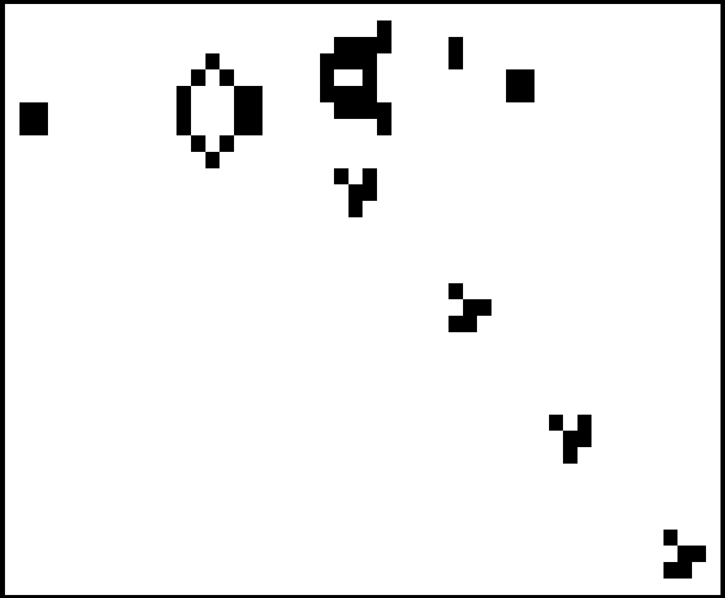
- **Any live cell with 2 or 3 live neighbors remains live**
- **Any dead cell with 3 live neighbors becomes live**
- **All other cells are dead**

Complexity Theory as Games

P captures short “zero-player” games

(Letting this game play out by itself, will it lead to a “win” or not?)

Example of a Zero-Player Game: Conway’s Game of Life



Theorem: Given an arbitrary 2d grid with finitely many alive cells and another given pattern, it is *undecidable* to determine if that pattern will ever eventually appear!

A fundamentally unpredictable and universal little game!

Complexity Theory as Games

PSPACE is…

a complexity class for

***two-player* games of perfect information!**

For formalizations of
many popular two-player games,
it is PSPACE-complete to decide
***which player* has a winning strategy**
on a game board!

TQBF as a Two-Player Game

Two players, called E and A

Given a fully quantified Boolean formula

$$\exists y \forall x\, [\, (x \vee y) \wedge (\neg x \vee \neg y)\,]$$

The game starts at the leftmost quantified variable

E chooses values for variables quantified by $\exists$

A chooses values for variables quantified by $\forall$

E wins if the resulting formula evaluates to true

A wins otherwise

Examples: $\forall x \exists y\,[\,(x \vee y) \wedge (\neg x \vee \neg y)\,]$

E has a winning strategy: no matter what A sets x to, E can set y to make the formula true

$\exists x \forall y\,[\;x \vee \neg y\;]$

E has a winning strategy: set x = 1

FG = { ϕ | Player E has a winning strategy on ϕ }

Theorem: FG is PSPACE-Complete

Proof:

FG = TQBF

ϕ is true $\Leftrightarrow$ Player E has a winning strategy on ϕ!

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!