

Programming Languages Research and Education

the topic of Ultimate Mastery

Wes Weimer

<http://www.cs.virginia.edu/~weimer>

Reasonable Initial Skepticism



Basic Plan

Show four topics in PL research

Relate each one to a project
kids might care about

Force you to do pencil-and-
paper work and participate

**Machine Learning
(Social Networks)**

**Model Checking
(Game Theory)**

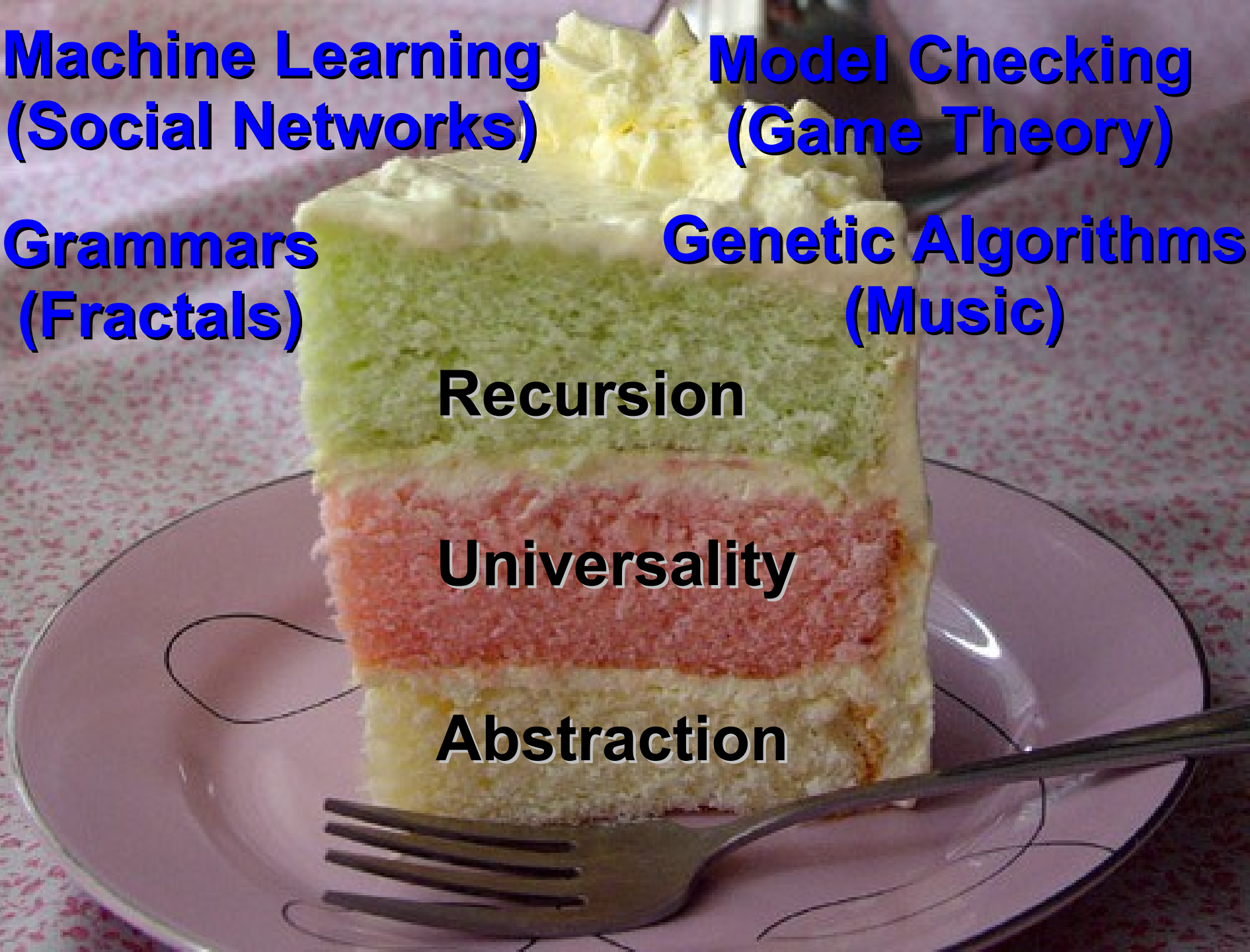
**Grammars
(Fractals)**

**Genetic Algorithms
(Music)**

Recursion

Universality

Abstraction



Goal: Fun



Thus: Interrupt!

Formal Grammars

They're
Chomsky-riffic!

- **Grammars** are used by linguists to describe the block structure of languages
 - Describe how sentences are built up **recursively** from smaller phrases
- Dave Evans discussed:
 - **word ::= anti- word**
 - **word ::= disestablishmentarianism**
 - **word ::= floccinaucinihilipilification**
- In practice, English is hard to capture but Java and HTML are codified by formal grammars.

Rewriting Systems

- Grammars can equivalently be viewed as term rewriting systems.

- Grammar:

- $E ::= 2$

- $E ::= 7$

- $E ::= E + E$

- $E ::= E - E$

- Rewriting System:

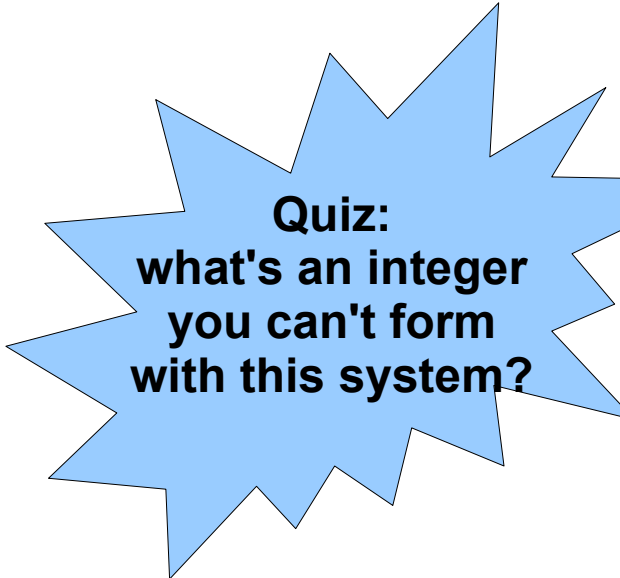
- Start: E

- Rule1: $E \rightarrow E + E$

- Rule2: $E \rightarrow E - E$

- Rule3: $E \rightarrow 2$

- Rule4: $E \rightarrow 7$



Quiz:
what's an integer
you can't form
with this system?

Example: Let's Get “1”

- Rewriting System:

- Start: E

- Rule1: $E \rightarrow E + E$

- Rule2: $E \rightarrow E - E$

- Rule3: $E \rightarrow 2$

- Rule4: $E \rightarrow 7$

- Start: E

- Rule2: $E - E$

- Rule2: $E - E - E$

- Rule2: $E - E - E - E$

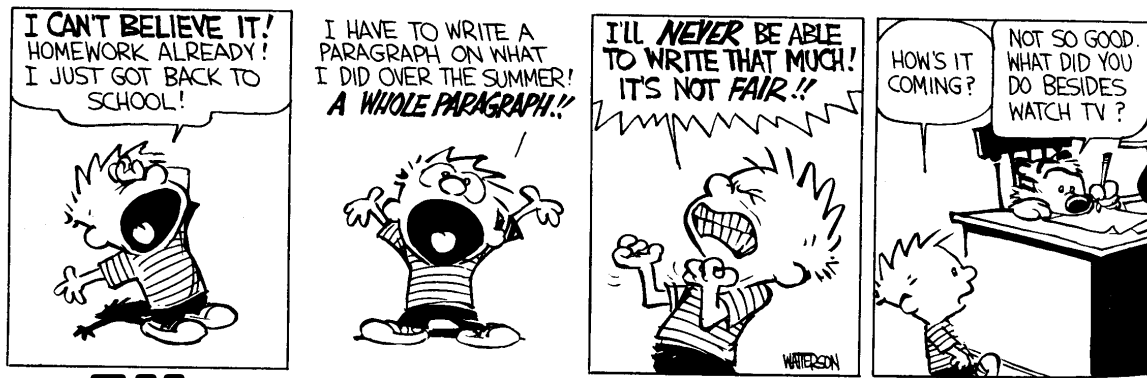
- Rule4: $7 - E - E - E$

- Rule3: $7 - 2 - 2 - 2$ (apply 3 times)

Douglas Hofstadter's MU Puzzle

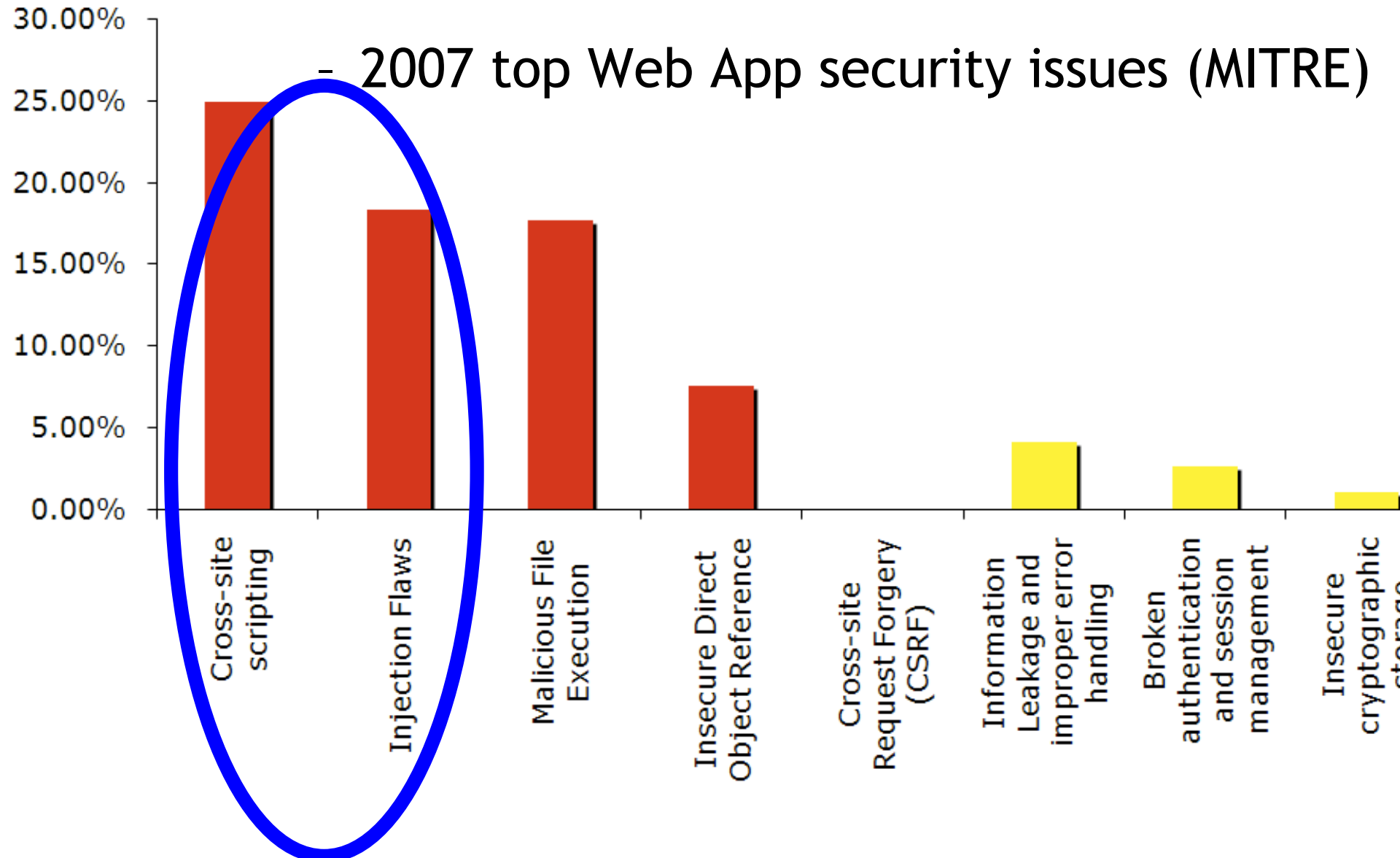
- Start: **MI**
- Rule1: $x\mathbf{I} \rightarrow x\mathbf{IU}$
- Rule2: $\mathbf{M}x \rightarrow \mathbf{M}xx$
- Rule3: $x\mathbf{IIII}y \rightarrow x\mathbf{U}y$
- Rule4: $x\mathbf{UU}y \rightarrow xy$
 - x and y are any possibly-empty sequences of letters
- Example: $MI \rightarrow MIU \rightarrow MIUIU$

- Start: **MI**
- Rule1: $xI \rightarrow xIU$
- Rule2: $Mx \rightarrow Mxx$
- Rule3: $xIIIy \rightarrow xUy$
- Rule4: $xUUy \rightarrow xy$
- Question1: How can we get **MUI** ?
- Question2: How can we get **UI** ?
- Question3: How can we get **MUIIU** ?
- Question4: How can we get **MU** ?



Research: String Variables

- 2007 top Web App security issues (MITRE)



Cross-Site Scripting

- **Cross-site scripting** is a security vulnerability in which innocent browsers (you) go to some trusted site (a blog, cnn.com) and unknowingly receive malicious content (evil javascript) supplied by evildoers (script kiddies), thinking that it is from the trusted site.
 - “Obama site hacked; Redirected to Hillary Clinton” <http://blogs.zdnet.com/security/?p=1042>
 - <http://youtube.com/watch?v=NKjomr1Afq0> (disregard the video's politics, sorry!)

Our Research

- Given web application code like this:

```
age = read_string_from_evil_user();
if (age contains "0" or "1" or "2" or "3" or "4"
    or "5" or "6" or "7" or "8" or "9") then {
    output_html = "Poster's Age: <b>" + age + "</b>";
    display_to_innocent_user(output_html);
} else {
    report_error();
}
```

- Is there any way for the “output_html” shown to the innocent user to contain the word “JavaScript”?

Fun Stuff: L-System Fractals

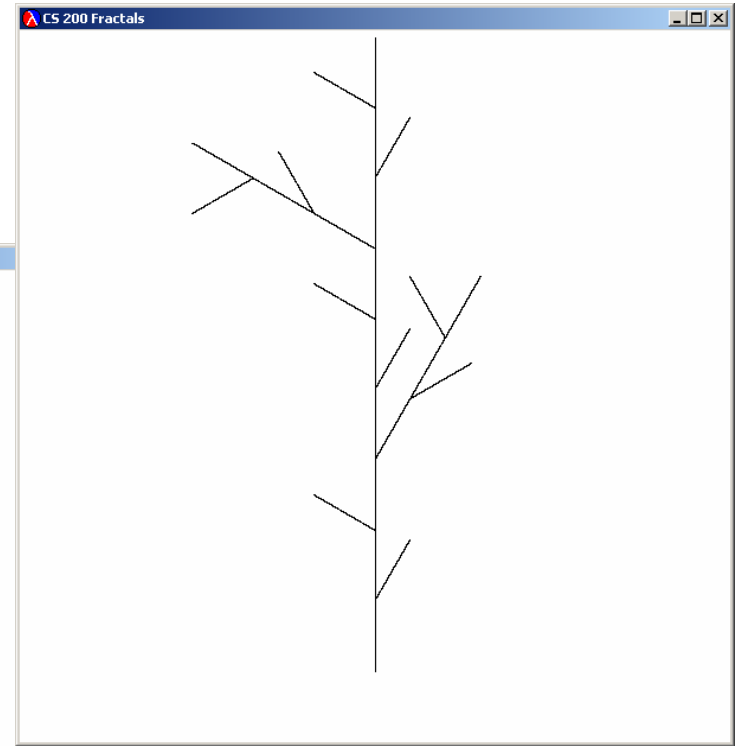
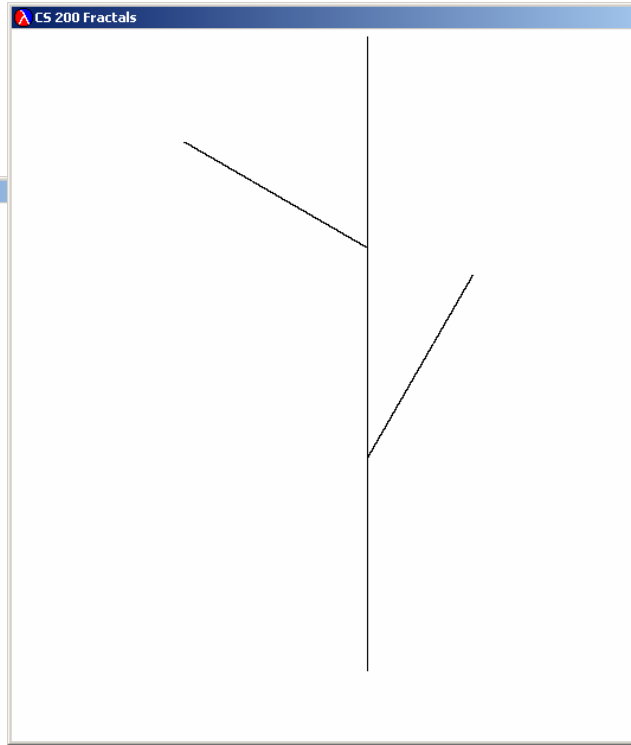
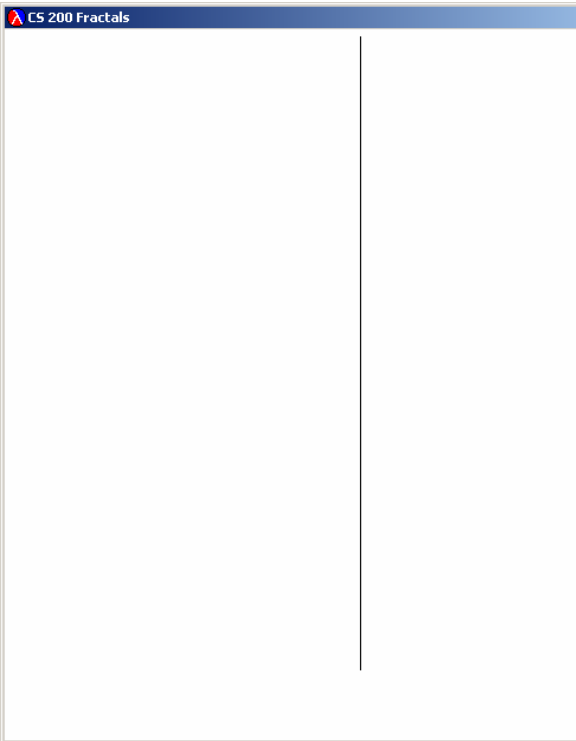
- Aristid Lindemayer, a theoretical biologist at the University of Utrecht, developed the **L-system** in 1968 as a mathematical theory of plant development. In the late 1980s, he collaborated with Przemyslaw Prusinkiewicz, a computer scientist at the University of Regina, to explore computational properties of the L-system and developed many of the ideas on which this problem set is based.

Example L-System

- Start: (F)
- Rule: $F \rightarrow (F O(R30 F) F O(R-60 F) F)$
 - F = “Forward”
 - O(x) = “Make an offshoot containing x”
 - Ry = Turn right y degrees
- Iteration 0: (F)
- Iteration 1: (F O(R30 F) F O(R-60 F) F)
- Iteration 2: (F O(R30 F) F O(R-60 F) F O(R30 F O(R30 F) F O(R-60 F) F) F O(R30 F) F O(R-60 F) F O(R-60 F O(R30 F) F O(R-60 F) F) F O(R30 F) F O(R-60 F) F)

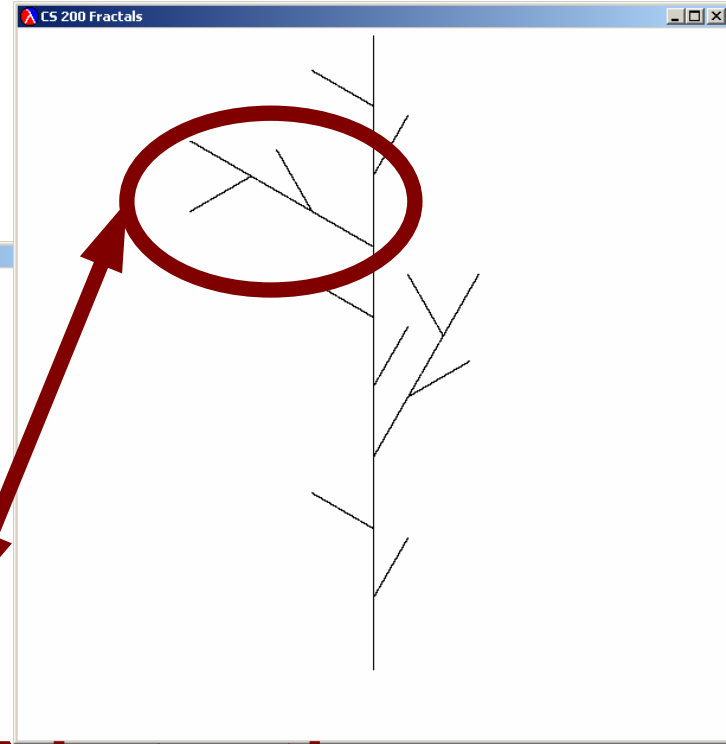
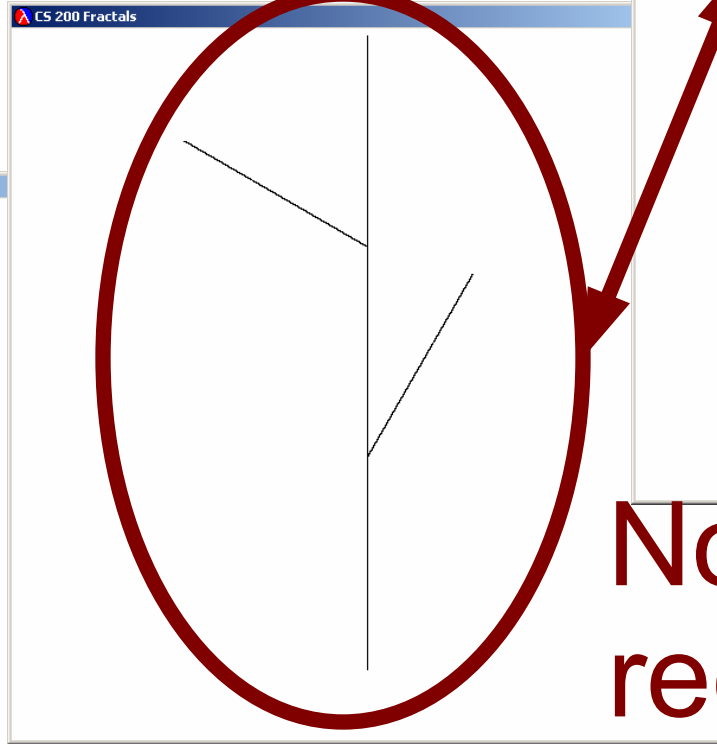
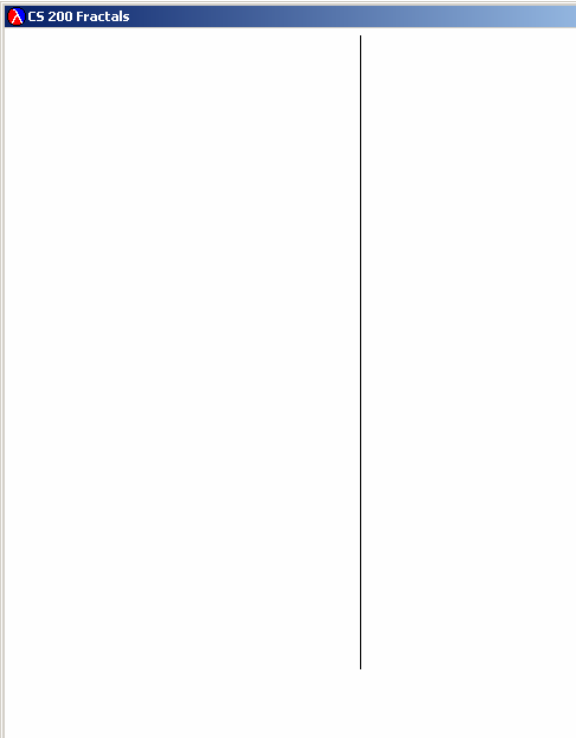
L-System Growth

- Rule: $F \rightarrow (F O(R30 F) F O(R-60 F) F)$



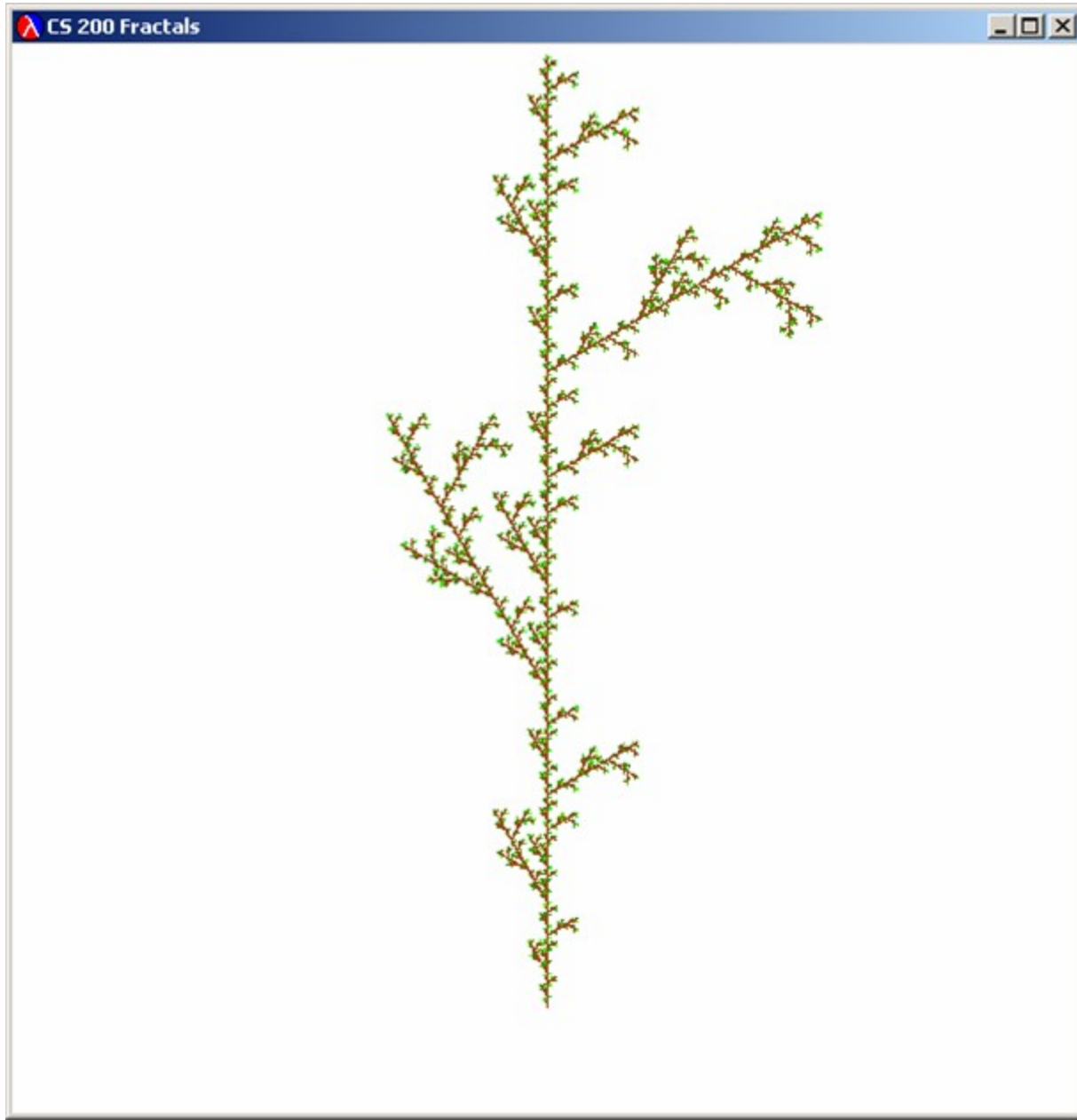
L-System Growth

- Rule: $F \rightarrow (F O(R30 F) F O(R-60 F) F)$

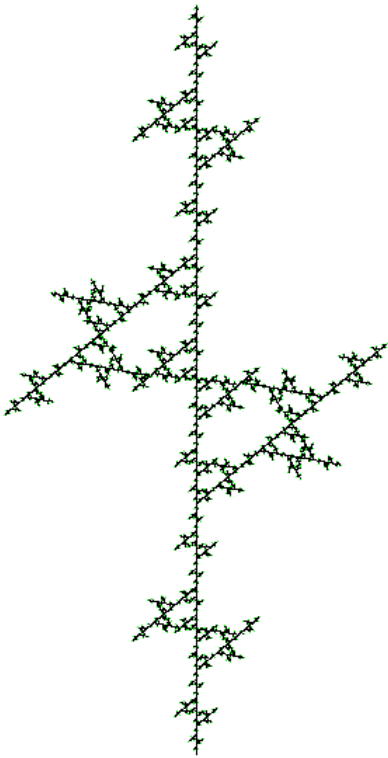


Note the recursion!

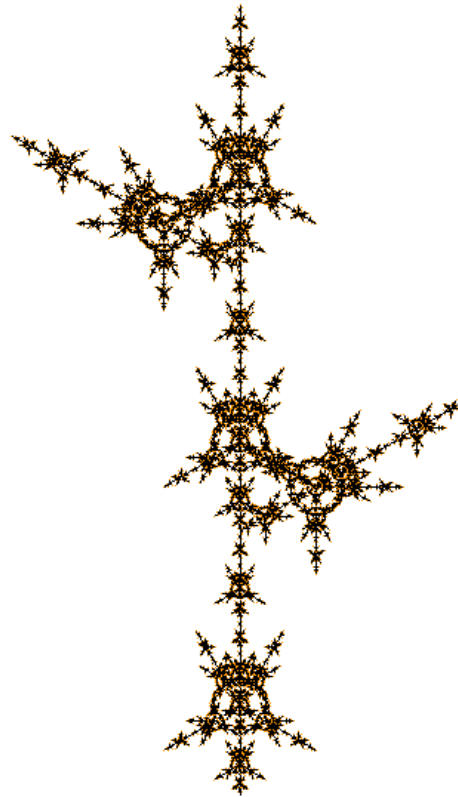
Iteration 5



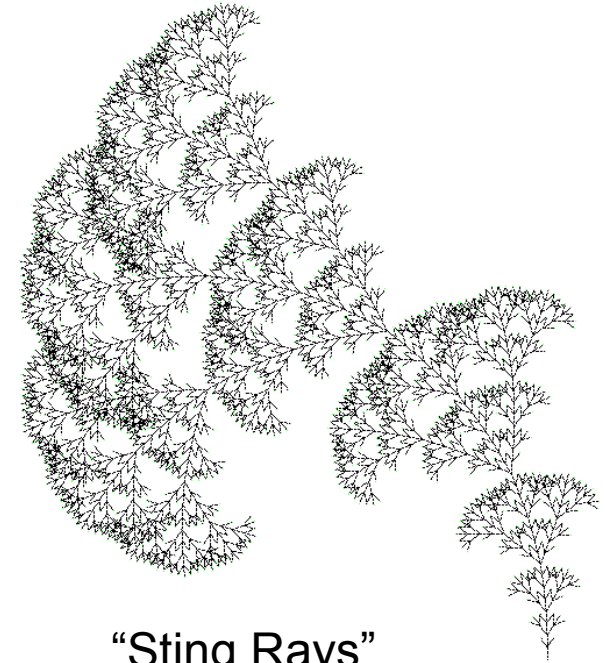
Fractals Made By 1st Semester CS students, 4 weeks in



“Stars of David”



“Clocks”



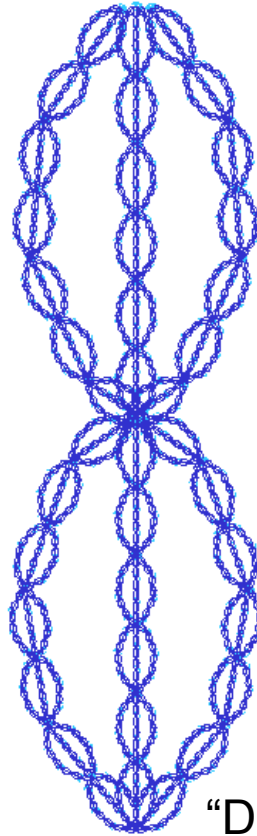
“Sting Rays”

Each fractal corresponds to a different $F \rightarrow \dots$ rewrite rule.

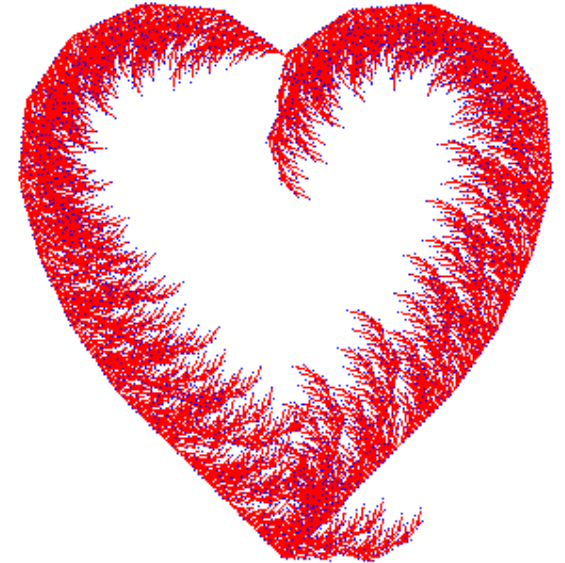
Fractals Made By 1st Semester CS students, 4 weeks in



“It Looks Pretty To Me”



“DNA Infinity”



“A Heart”

<http://www.cs.virginia.edu/~weimer/150/frac/index.html>
<http://www.cs.virginia.edu/~weimer/150/ps/ps3/>

Topic 2 - Machine Learning (aka “Finding Patterns”)

- In **frequent itemset mining**, a problem in **machine learning**, store owners attempt to discover items that are commonly purchased in tandem. This helps them arrange aisles and coordinate sales.
- Riddle: What do beer and diapers have in common?
- The same basic ideas are used to detect when your credit card has been stolen.

Grocery Cards and Amazons



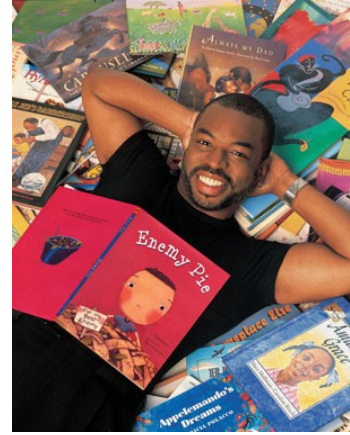
- Grocery stores can afford to give you a “discount” if you use these because they view the marketing information (e.g., what sets of items you purchase together over time) as more valuable.
- Amazon.com, Netflix, etc.
 - Recommendations!



Research: Specification Mining

- In programs, open must be followed by close, lock must be followed by unlock, and malloc must be followed by free.
- If we know these rules, we can look at the source code to your program and find bugs without testing before you ship the program (i.e., before you turn it in)!
- But ... how do we know these rules?

A Reading Rainbow



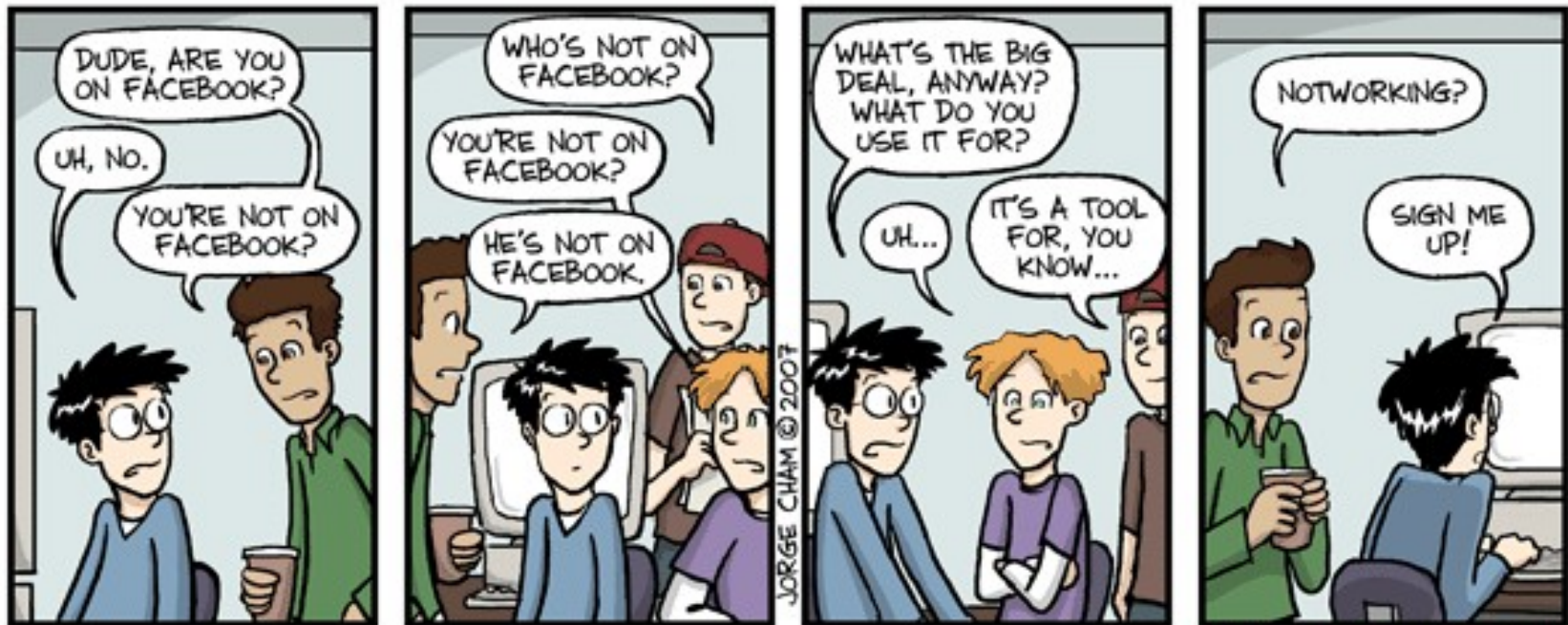
- In essence: learn the rules of English grammar by reading high school English essays.
- Look at actual program paths for patterns
- Path1: open, print, close
- Path2: open, read, print, close
- Path3: print, print, exit
- Path4: print, open, read, write, close
- Path5: open, read, print

Research: Specification Mining

- Problem: actual programs have bugs
- Our insight: this task would be easier if we could tell the A+ students from the C-students.
 - We thus incorporate software quality metrics.
- Look for rules that are followed on “good” paths and broken on “bad” paths
- Result: reduce false positive rate from 90-99% (previous state-of-art) to 5%

Fun Part: Social Networking

- **Social networking** sites like Facebook and MySpace use similar algorithms to recommend friends and groups to existing members.



Project Suggestion

- Write a FaceBook application
 - <http://fyi.oreilly.com/2008/08/how-to-write-your-own-facebook.html>
 - <http://gathadams.com/2007/06/18/how-to-write-a-facebook-application-in-10-minutes/>
 - http://developers.facebook.com/get_started.php
- It's relatively painless to convert an existing class program (e.g., Sudoku) into a FaceBook app.
 - Optionally: make an app that finds the most similar person to every person who joined
- Cellphone apps are also painless: Google Android, for example, uses standard Eclipse/Java development.

Project Suggestion 2

- Try out making movie recommendations. Netflix offers prizes from \$50,000 to \$1M for movie recommendation algorithms that can improve the state of the art.
- They provide anonymized training data (= records of what people rated various movies on Netflix in the past).
 - 1 million ratings of the >1 billion they have
- Simple algorithms can be coded in one page; provides natural intro to Big-Oh, etc.
- <http://www.netflixprize.com/>

Topic 3 - Model Checking

- Programmers spend quite a bit of time hunting down and fixing bugs ...



It's Not Just You

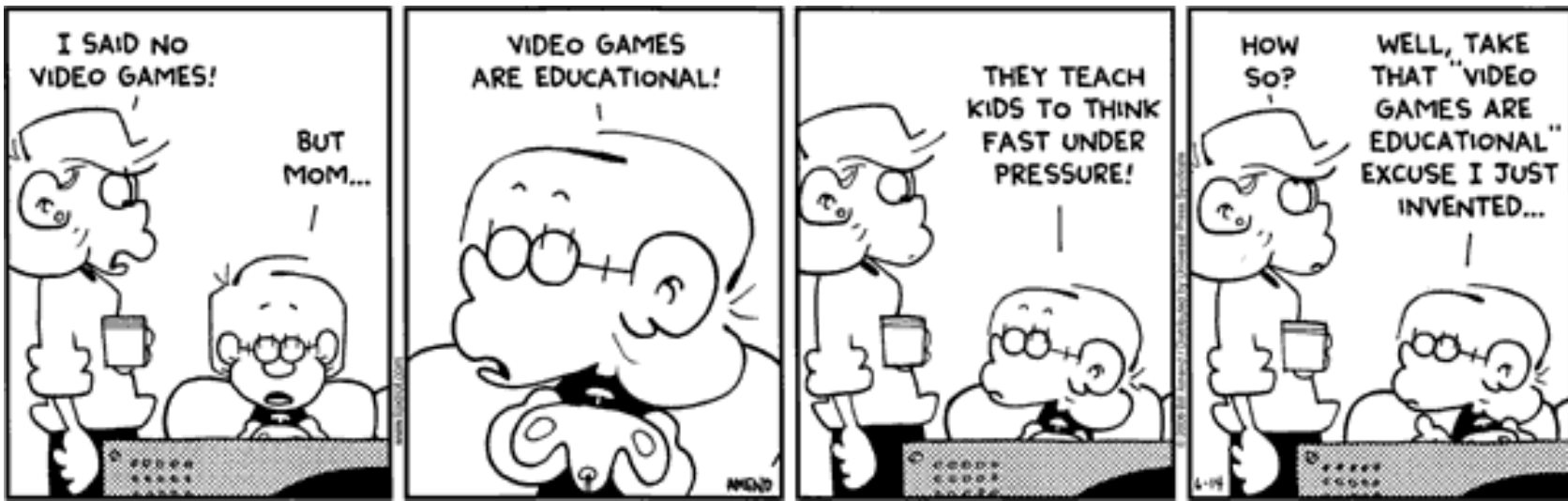
- In 2008, 139 North American firms spent a mean of \$22 million each fixing bugs. The cost of fixing a bug increases throughout development, from about \$25 while coding to \$16000 after deployment. In 2006, it took 28 days on average for maintainers to develop fixes for security flaws; in 2008 an FBI survey of over 500 large firms found that the average annual cost of security defects alone was \$289,000. **In 2002, NIST calculated the average US-wide annual cost of software errors to be \$59.5 billion, or 0.6% of the US GDP.**

Fun Part: Game Theory

- Software security and correctness can be viewed as a two-player game: one player represents the software, and another player is “the environment”.
 - The environment will try to mess you up: disk reads will fail, you'll run out of memory, evil users will perform cross-site scripting attacks, etc.
- If you have a winning strategy, you don't have bugs. **Model checking** exhaustively explores all options to see if you have one.³¹

Game Theory

- **Game Theory** is a branch of applied math used in the social sciences (econ), biology, compsci, and philosophy. Game Theory studies *strategic* situations in which one agent's success depends on the choices of other agents.



Broad Applicability

- Finding equilibria (Nash) - sets of strategies where agents are unlikely to change behavior.
- Econ: understand and predict the behavior of firms, markets, auctions and consumers.
 - Think “EBay”!
- Animals: (Fisher) communication, gender
- Ethics: normative, good and proper behavior
- PolySci: fair division, public choice. Players are voters, states, interest groups, politicians.

Nim

- *Nim* is a two-player game in which players take turns removing objects from distinct heaps.
 - Properties: non-cooperative, symmetric, sequential, perfect information, finite, **impartial**
- One each turn, a player **must remove** at least one object, and may remove **any number** of objects provided they all come from the **same heap**.
- If you cannot take an object, **you lose**.
- Similar to Chinese game “Jianshizi” (“picking stones”); European refs in 16th century

Example Nim

- Start with heaps of 3, 4 and 5 objects:

- AAA, BBBB, CCCCC

- Here's a game:

- AAA BBBB CCCCC

I take 2 from A

- A BBBB CCCCC

You take 3 from C

- A BBBB CC

I take 1 from B

- A BBB CC

You take 1 from B

- A BB CC

I take all of A

- BB CC

You take 1 from C

- BB C

I take 1 from B

- B C

You take all of C

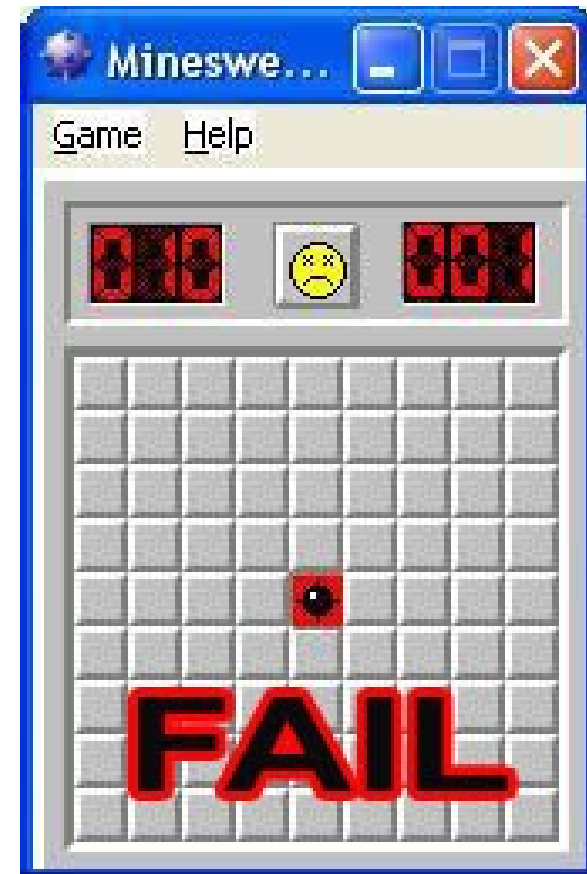
- B

I take all of B

- You lose! (you cannot go)

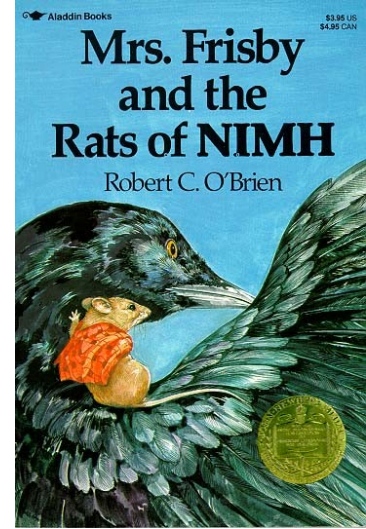
Real-Life Nim Demo

- I will now play Nim against audience members.
- Starting Board: 3, 4, 7
 - AAA, BBBB, CCCCCC
- You go first ...



The Rats of NIM

- How did I win every time?
 - Did I win every time? If not, pick on me mercilessly.
- Nim can be mathematically solved for any number of initial heaps and objects.
- There is an easy way to determine which player will win and what winning moves are available.
 - Essentially, a way to evaluate a board and determine its payoff / goodness / winning-ness₃₇



Analysis

- You lose on the empty board.
- Working backwards, you also lose on two identical singleton heaps (A, B)
 - You take one, I take the other, you're left with the empty board.
- **By induction (recursion), you lose on two identical heaps *of any size* (A^n, B^n)**
 - You take x from heap A. I also take x from heap B, reducing it to a smaller instance of “two identical heaps”.

Analysis II

- On the other hand, you win on a board with a singleton heap (C).
 - You take C, leaving me with the empty board.
- You win with a single heap of any size (C^n).
- What if we add these insights together?
 - (AA, BB) is a loss for the current player
 - (C) is a win for the current player
 - (AA, BB, C) is what?

Analysis III

- (AA, BB, C) is a win for the current player.
 - You take C, leaving me with (AA, BB) - which is just as good as leaving me with the empty board.
- When you take a turn, it becomes my turn
 - So leaving me with a board that would be a loss for you, if it were your turn
 - ... becomes a win for you!
- (AAA, BBB, C) - also a win for Player1.
- (AAAA, BBBB, CCCC) - also a win for Player1.

Generalize

- We want a way of evaluating nim heaps to see who is going to win (if you play optimally).
- Intuitively ...
- Two equal subparts cancel each other out
 - (AA, BB) is the same as the empty board (,)
- Win plus Loss is Win
 - (CC) is a win for me, (A,B) is a loss for me,
(A,B,CC) is a win for me.
- What do we know that's kind of like addition but cancels out equal numbers?

The Trick!

- *Exclusive Or*
 - XOR, \oplus , vector addition over GF(2), or *nim-sum*
- If the XOR of all of the heaps is 0, you lose!
 - empty board = 0 = lose
 - (AAA, BBB) = $3 \oplus 3 = 0 = \text{lose}$
- Otherwise, goal is to leave opponent with a board that XORs to zero
 - (AAA, BBB, C) = $3 \oplus 3 \oplus 1 = 1$, so move to
 - (AAA, BBB) or (AA, BBB, C) or (AAA, BB, C)

Real-Life Nim Demo II

- I played Nim against audience members.
- Starting Board: 3, 4, 7
 - AAA, BBBB, CCCCCC
- The nim sum is $3 \oplus 4 \oplus 7 = 0$
 - A loss for the first player!
- This time, I'll go first.




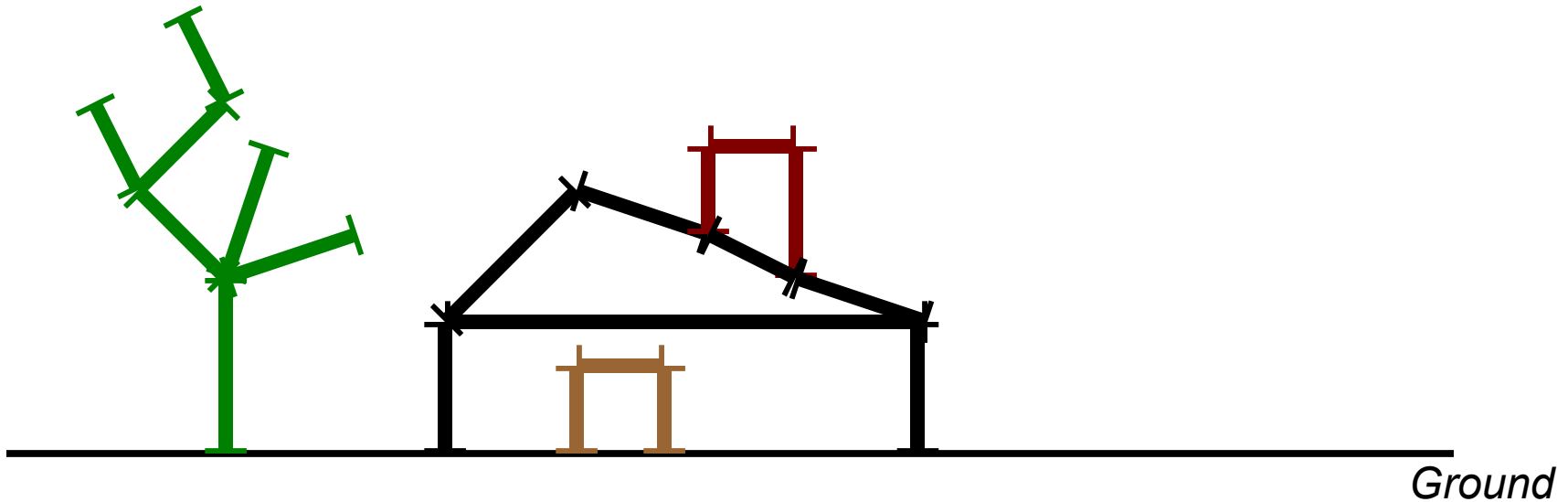
- You, the audience, must beat me. Muahaha!₄₃

Hackenbush

- *Hackenbush* is a two-player impartial game played on any configuration of line segments connected to one another by their endpoints and to a **ground**.
- On your turn, you “cut” (**erase**) a **line segment** of your choice. Line segments no longer connected to the ground are erased.
- If you cannot cut anything (empty board) **you lose**.

Hackenbush Example

- Each  is a line segment. Ignore color.
- Let's play! I'll go first.

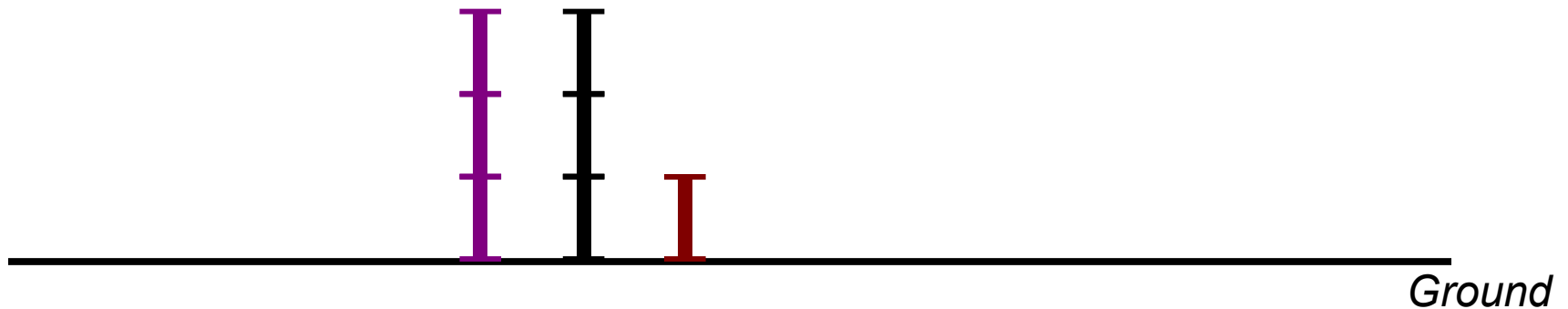


House of Blue Jeans ...



Hackenbush Subsumes Nim

- Consider (AAA, BBB, C) = (3, 3, 1) in Nim
- Who wins this *completely unrelated* Hackenbush game?



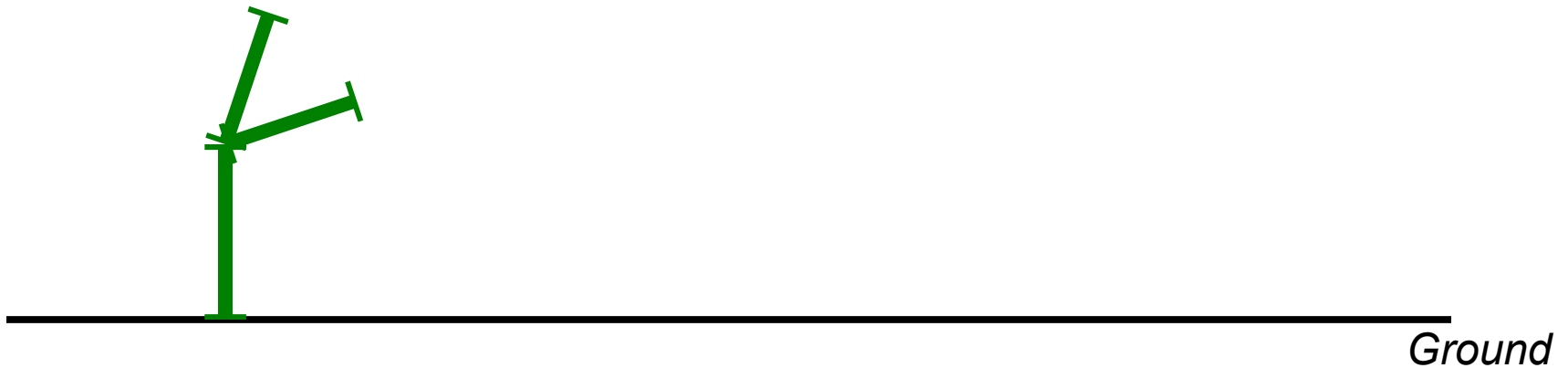
A Thorny Problem

- What about that Hackenbush tree?
- What value (nim-sum) does it have? Who wins?



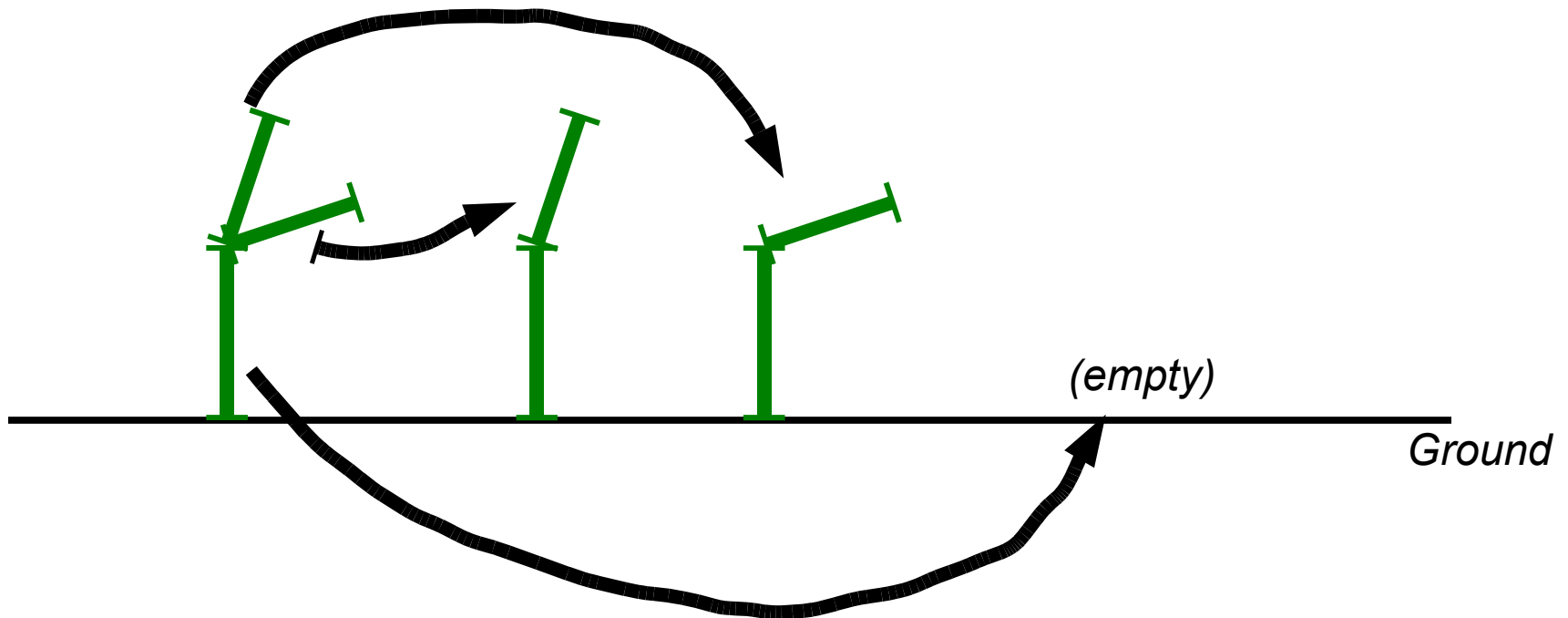
A Simple Twig

- Consider a simpler tree ...
- What moves do you have?



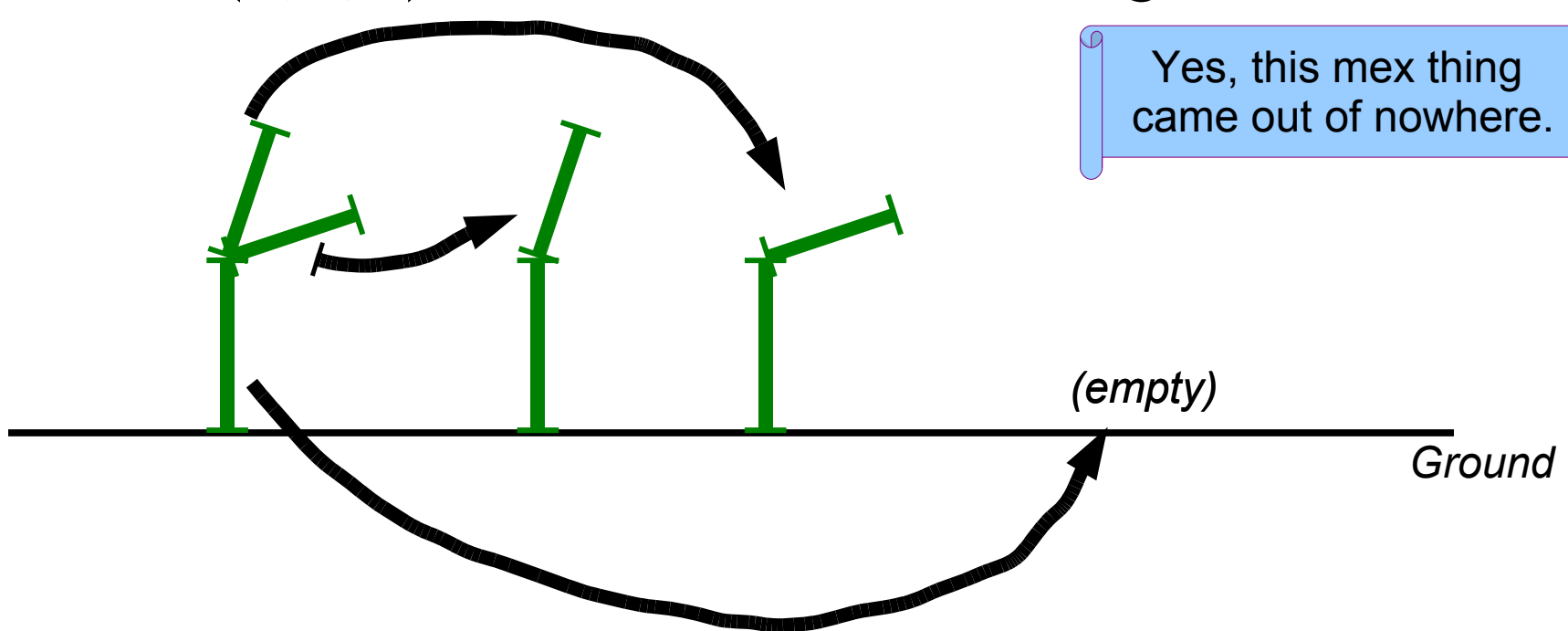
Twig Analysis

- Consider a simpler tree ...
- What moves do you have?



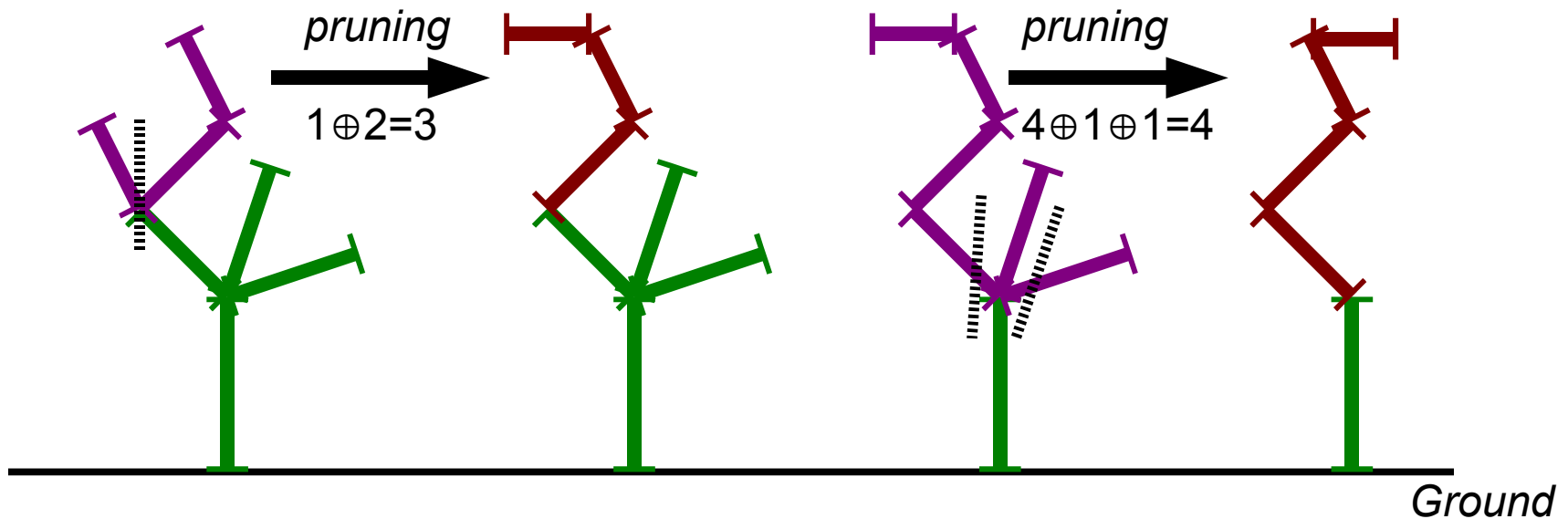
Maximum Excluded

- You can move to “2”, “2” or “0”.
- The *minimal excluded* of (2,2,0) is 1
 - $\text{mex}(2,2,0) = 1 = \text{value of that twig}$



Generalized Pruning

- Can replace any subtree above a single branch point with the XOR of those branches
 - Via similar game-equivalence argument



The whole tree has value "5".

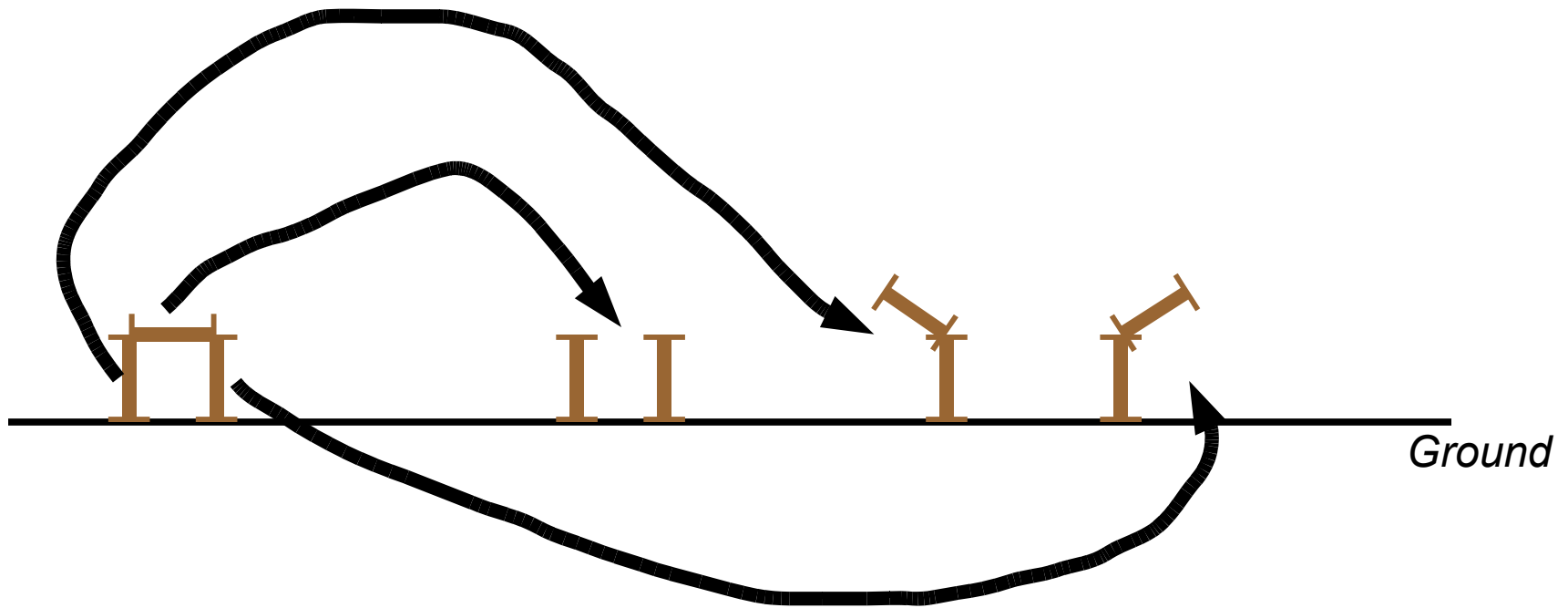
Door Analysis

- What about cycles?
- What is the value (nim-sum) of this door?



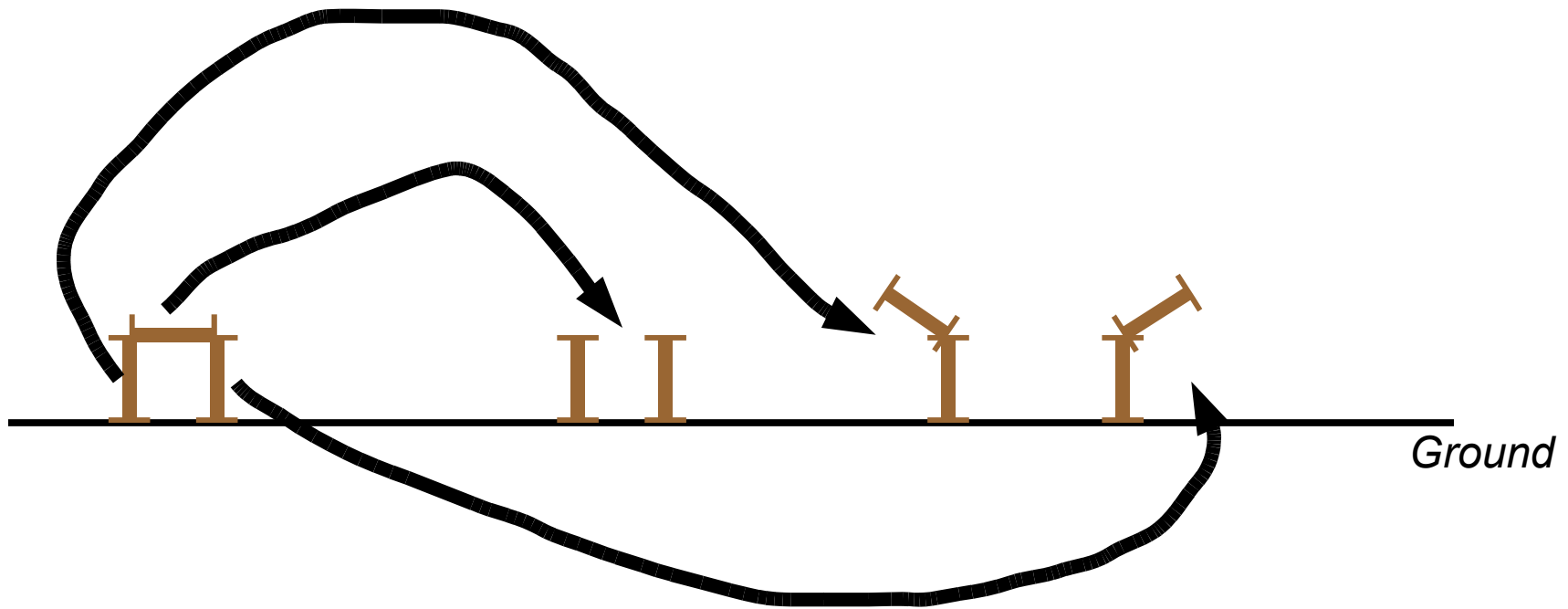
Door Analysis

- Well, what moves can you take from here?



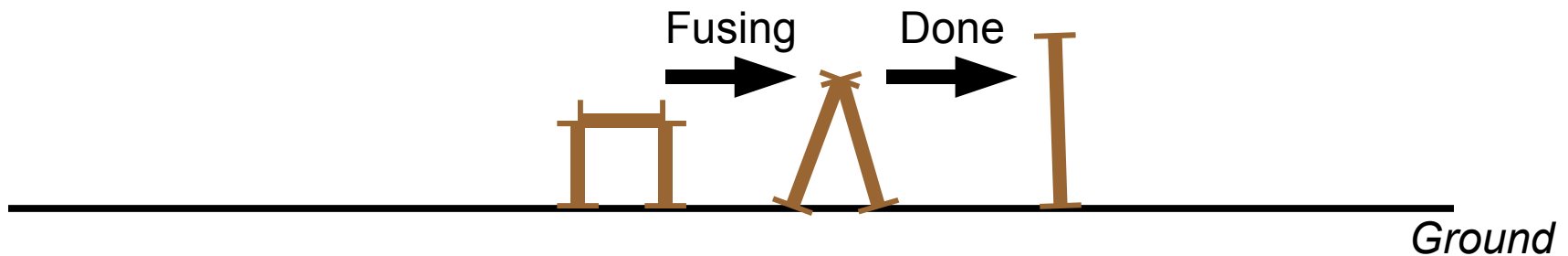
Door Analysis

- You can move to “0”, “2” or “2”.
 - $\text{mex}(2,2,0) = 1$ (recall: minimal excluded)
 - Value of door = 1



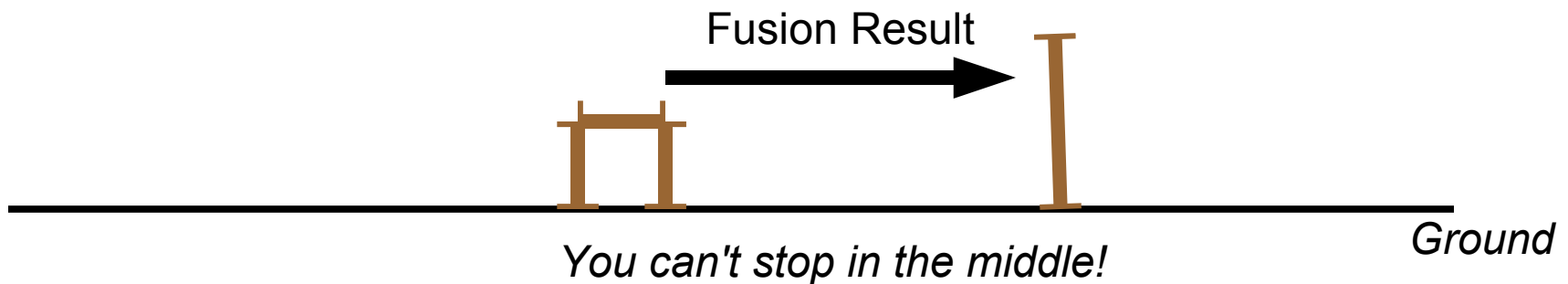
Fusion Principle

- We may replace any cycle with an equivalent subgraph where all of the non-ground vertices of that cycle are fused into one vertex and all of the ground vertices of that cycle are fused into another vertex.



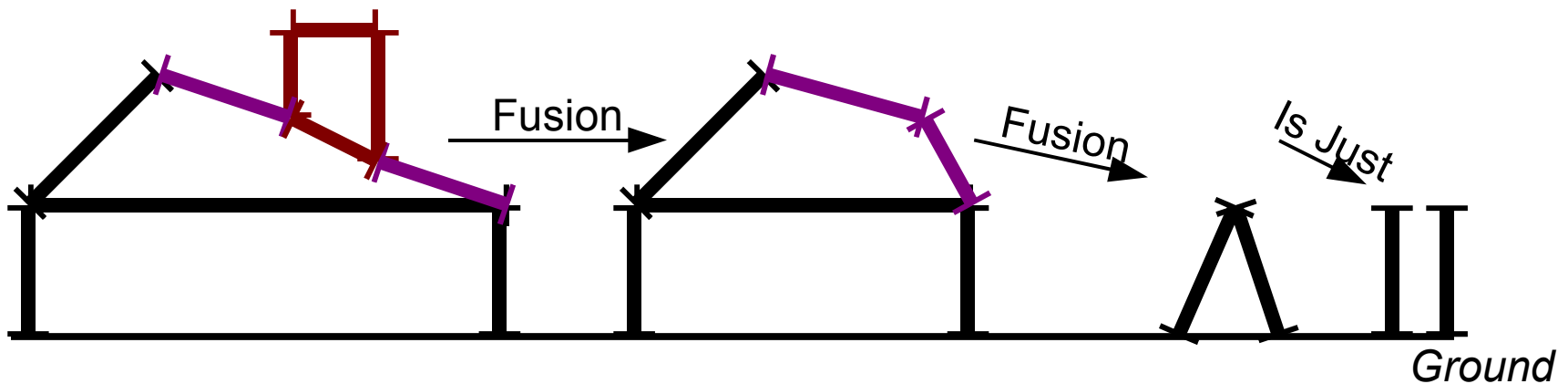
Fusion Principle

- We may replace any cycle with an equivalent subgraph where all of the non-ground vertices of that cycle are fused into one vertex and all of the ground vertices of that cycle are fused into another vertex.



Cold Fusion

- Let's boil the house down to something simple!

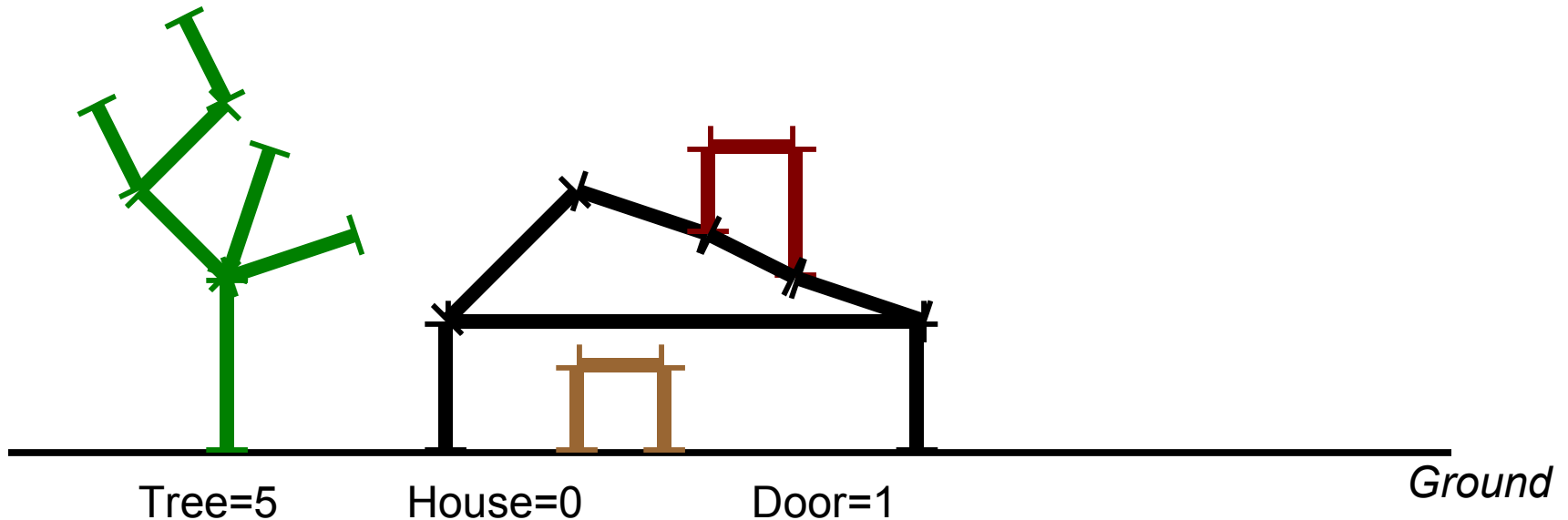


The whole house has value $1 \oplus 1 = 0$.

How would I check that?

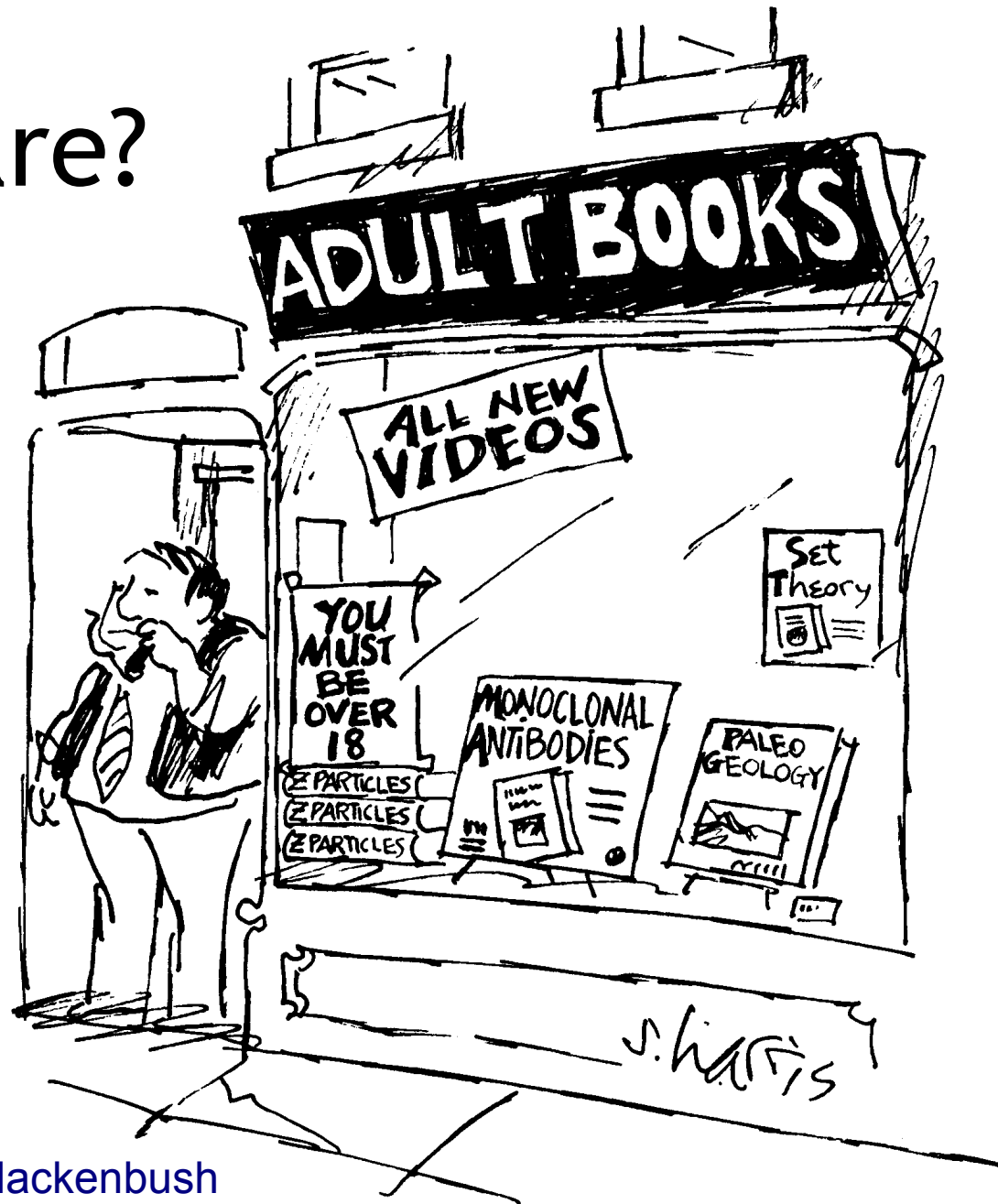
Hackenbush Example

- This board has value $5 \oplus 0 \oplus 1 = 4$.
- You go first. Beat me. (Time permitting.)



Why Do We Care?

- ... about Nim and Hackenbush?
- **Theorem (Sprague-Grundy, '35-'39).**
Every impartial game is equivalent to a nim sum.
- Proof by induction.
- **Universality!**



<http://en.wikipedia.org/wiki/Hackenbush>

http://en.wikipedia.org/wiki/Winning_Ways_for_your_Mathematical_Plays

<http://en.wikipedia.org/wiki/Nim>

Final Topic - Genetic Algorithms

- **Genetic Algorithms** are a search technique used to find exact or approximate solutions to search or optimization problems. Inspired by evolutionary biology, they maintain a population of individuals, subject them to random mutations, evaluate their fitness, and allow the fittest to survive and breed to form the next generation.

Magically Delicious

- Genetic algorithms are competitive with humans at tasks like synthesizing (inventing) audio amplifiers, electronic thermometers, or robot soccer players.
- Our research: use genetic algorithms to evolve your programs to fix bugs in them!
 - Automatically repaired 18 defects in 180,000 lines of code; takes about 3 minutes on average. Fun stuff, but ...

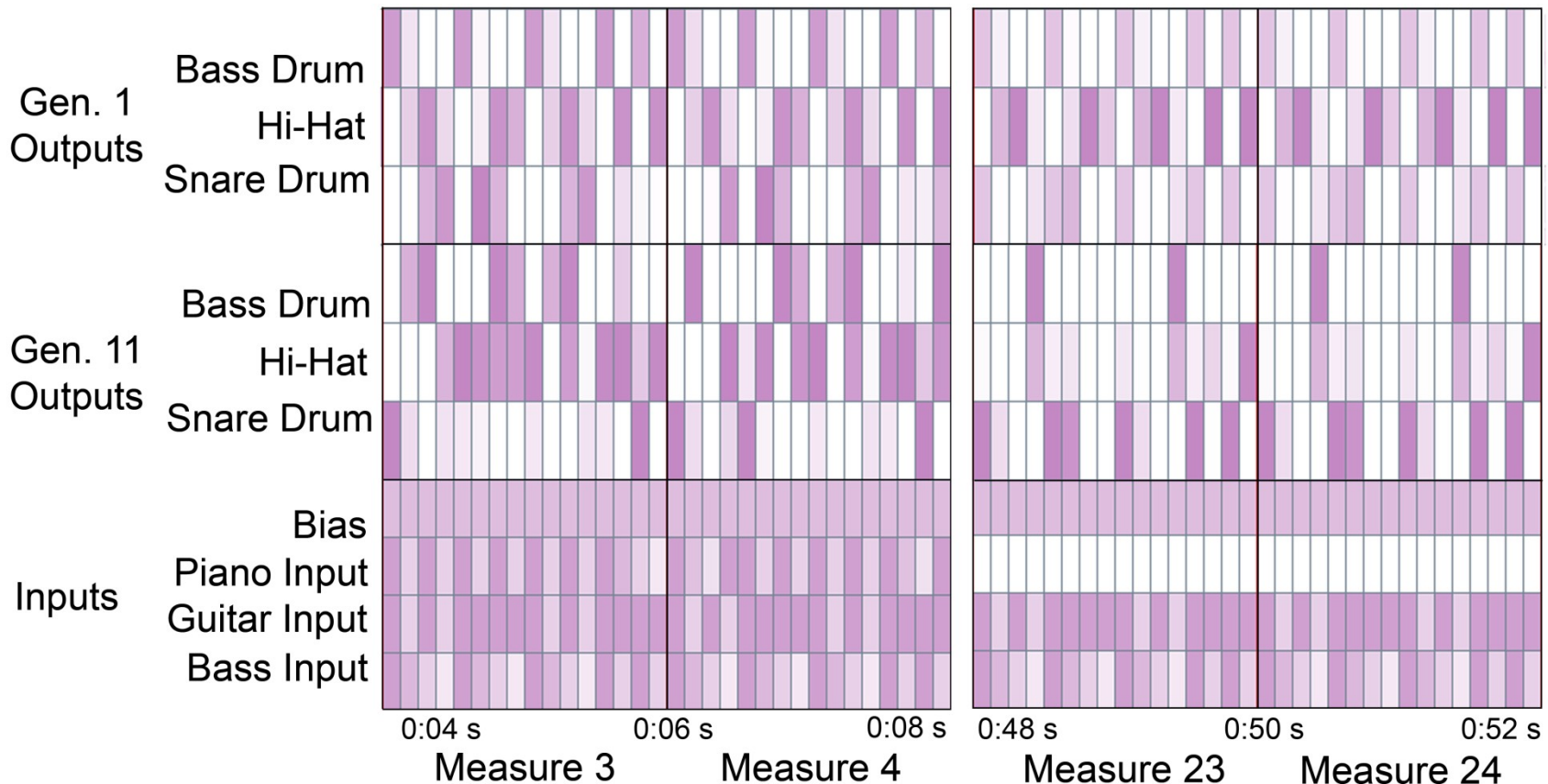
Fun Stuff: Let's Do Music Instead

- Creativity is typically viewed as uniquely human.
- Let's use genetic algorithms to make music.
 - In particular, we'll use GA to add accompaniment to music; this is tricky because existing tracks are a scaffold upon which the accompaniment is built, and the accompaniment must carry the global structure.
- The work I'll describe was done by:
- **Amy K. Hoover**, Michael P. Rosario, and Kenneth O. Stanley from the School of Electrical Engineering and Computer Science at the University of Central Florida
 - Amy did this as a third-year undergraduate.

High School Musical

- Example: “Johnny Cope”

More samples and comparisons: <http://eplex.cs.ucf.edu/neatmusic>



I'm Not A Musician (But I Play One On TV)

- Barry Taylor (the original musician) comments on the results:

“Those samples were amazing! ... I have heard some ‘auto drum’ software in the past and the result, at least when applied to old traditional tunes, is usually appalling. I must confess that I was a little wary of listening to the results of your application applied to *my* sequences, but your work gives me renewed hope that great quality can indeed be achieved! Keep up the superb work! I wish you continued success in your project...and I'm thrilled to contribute in this small way!”

Motivation

- Games such as Guitar Hero and Rock Band, as well as programs like Garage Band, are amazingly popular these days.
 - Why not harness some of that enthusiasm for class projects?



- Non-musical approaches:

- <http://jite.org/documents/Vol6/JITEv6p249-261Venables263.pdf> 65

You Survived!



Questions? Anyone? Bueller?

I will answer questions on any topic until we run out of time.