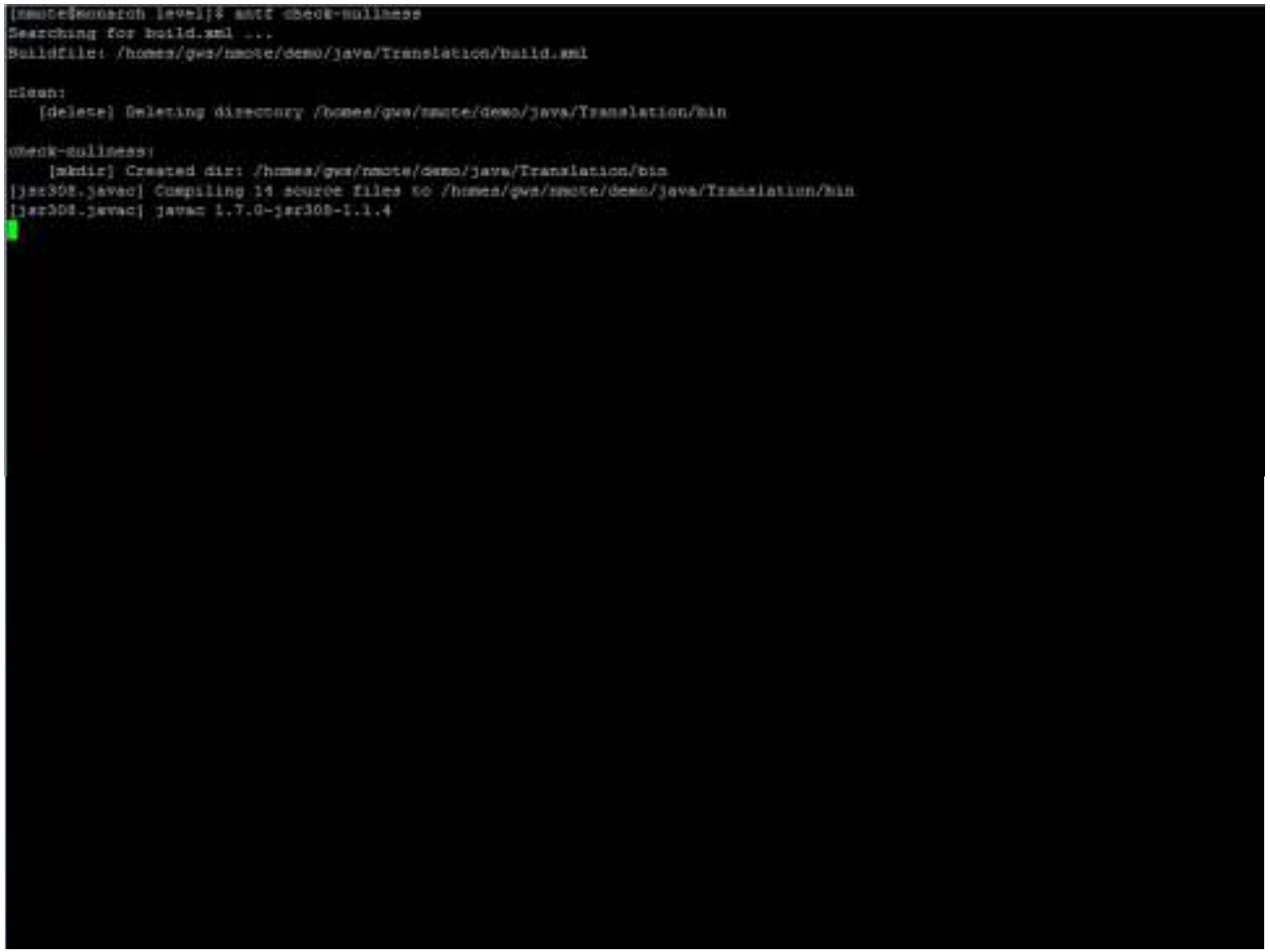
Verification games: Making verification fun

University of Washington
http://cs.washington.edu/verigames

Jonathan Burke, Matthew Burns, **Seth Cooper**, **Werner Dietl**, Stephanie Dietzel, **Michael Ernst**, Nat Mote, Tim Pavlik, **Zoran Popović**, Tyler Rigsby, Eric Spishak, Brian Walker

Angry Birds

Software verification



Which is more fun?

- Play games
- Prove your program correct

Crowd-sourced software verification

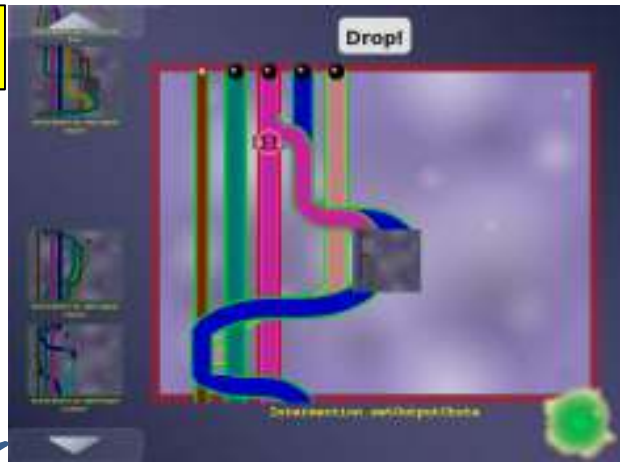
Goal: Verify software while you wait for the bus

- Make software verification **easy** and **fun**
- Make the game **accessible** to everyone
- Harness the power of the **crowd**

Code

Automatic translation

Game



Encodes a constraint system

Highly-skilled, expensive labor

Free labor

Verified software (with proof/ annotations)

Automatic translation

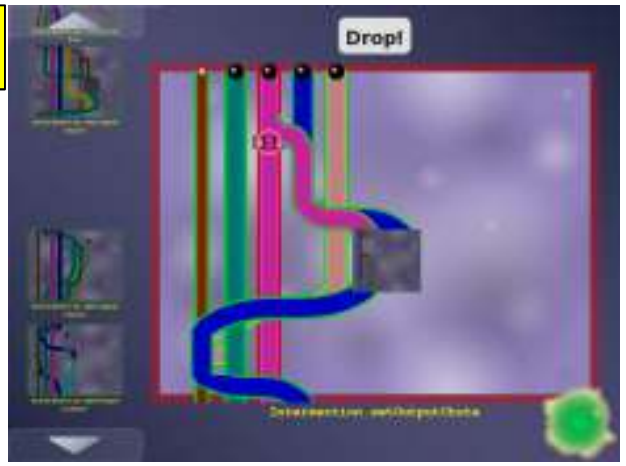
Completed game


SAVE
BACK
SUBMIT
World 1 Map
SCORE
2727
DROP!
Connection-vulture--GET
Connection-vulture--SET
Connection-in--GET
Connection--init--Ljava-net-Socket-Ljava-lang-ThreadGroup-ILVulture---V
Connection-run---V
Connection
Connection-vulture--GET
Connection-vulture--SET
Connection-client--GET

Code

Automatic translation

Game

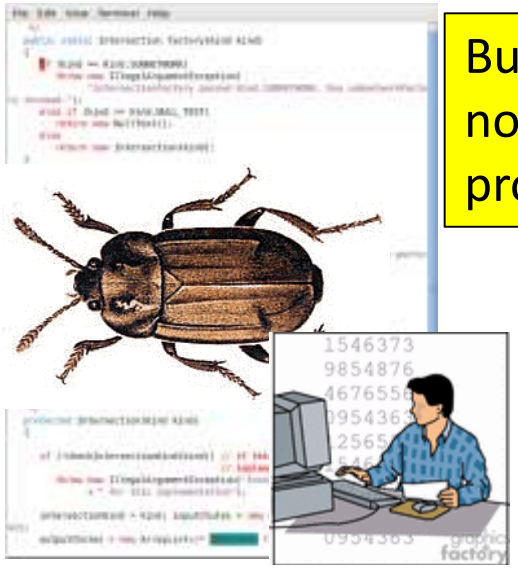


Free labor

Completed game with buzzsaws

Automatic translation

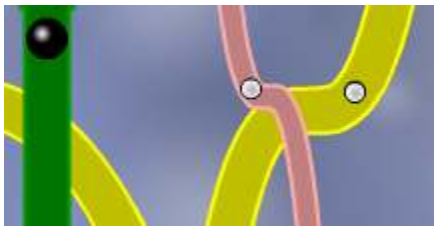
Bug detected, notify programmer



Highly-skilled, expensive labor

Program ↔ game correspondence

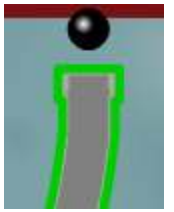
Intuition: dataflow



Pipe ↔ a variable

Pipe width ↔ a property of the variable (type)

Ball ↔ a value



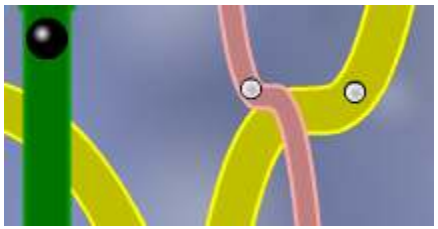
Ball size ↔ a property of the value

Pinch point ↔ requirement

Unmodifiable pipe/ball ↔ requirement

Example: encryption

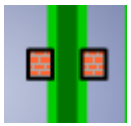
Goal: no cleartext is sent over the network



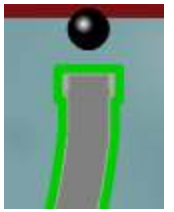
Pipe ↔ a variable

Pipe width ↔ narrow: encrypted, wide: cleartext

Ball ↔ a value



Ball size ↔ small: encrypted, large: cleartext

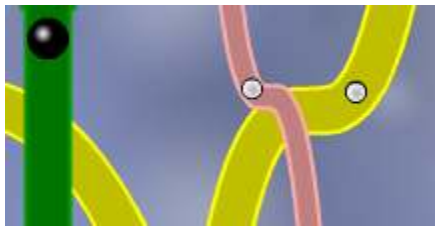


Pinch point ↔ network communication

Unmodifiable pipe/ball ↔ cleartext from user

Example: null pointer errors

Goal: no dereference of `null`

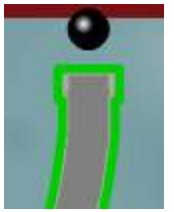


Pipe ↔ a variable

Pipe width ↔ narrow: non-null, wide: maybe null

Ball ↔ a value

Ball size ↔ small: non-null, large: null



Pinch point ↔ dereference

Unmodifiable pipe/ball ↔ literal **`null`**

Type flow vs. dataflow

- Multiple flows per variable
  - A variable's type may have multiple qualifiers

    @Immutable Map<@English String, @NonNegative Integer>
- Some variables are not represented at all
  - primitives (int, …) when analyzing null pointer errors
- No loops
  - If program is verifiable, solvable in polynomial time
  - Human leverage: high-level pattern matching

More accurate intuition: type constraints

- Building a new type inference framework

Other examples

- SQL injection
- unintended side effects
- format string and regexp validation
- incorrect equality checks
- race conditions and deadlocks
- units of measurement
- aliasing
- CWE/SANS Top 25 Most Dangerous Software Errors
- …

What verification challenges?

- Type system approach
- Modular; local reasoning & understanding
- Equally powerful as any other verification technology (theorem proving, model checking, …)
- Less effective for correctness of numerical computations
- Not good for full functional correctness
- Not good for temporal properties (focus on data)

Challenges

1. Can we build the system? Yes!

   End-to-end: game ↔ program verification

2. Will the game be fun? Maybe

   Better than waiting for the bus

3. Do people outperform verification algorithms?

   Inference is undecidable (human experts ≫ algorithms)

   Hypothesis: no for correct, verifiable programs,

   yes for incorrect or unverifiable programs

   Location of buzzsaws is key to the whole approach

   Game players only have to reduce overall verification cost, not fully verify the program

   Also see FoldIt (protein folding)

Scoring

Score is influenced by:

- Collisions (verifiability)
- Use of buzzsaws (trusted assumptions)
- Pipe widths, distinguishing input and output pipes (re-usability of modules)

Score is a proxy for quality of verification result

Collaboration and competition

- High-score boards
- Collaborative teams solve challenges
- Share scripts
- Interaction: chats, forums, …

3-way collaboration: machines, players, verification experts

1. Machines: Automatic inference and optimizations
   - Brute force is not feasible for large programs
   - Error messages from type inference systems are poor
2. Players do work that automated tools cannot
   - Leave a few easy challenges to encourage players
3. Verification experts do work that players cannot
   - Classify un-verifiable code as safe or insecure

Scalability

- Programs have natural modularity, created by the programmer
  - World = program
  - Level = class
  - Board = method
- Crowdsourcing scalability:
  - Distribute games to humans
    - Angry Birds: 5 million hours of play time *per day*
  - Reconfigure games to adjust difficulty
  - Redundancy

Status

- Prototype exists
  - Scales to small programs (hundreds of lines)
  - Players say it is “kind of fun”
- Many challenges remain
  - Scale to larger programs
  - Create tests (example failures, or counterexamples)
  - Scale to multiple players (parallelism, social aspects)
  - Make the game more fun
  - … many others

Pipe Jam

Contributions

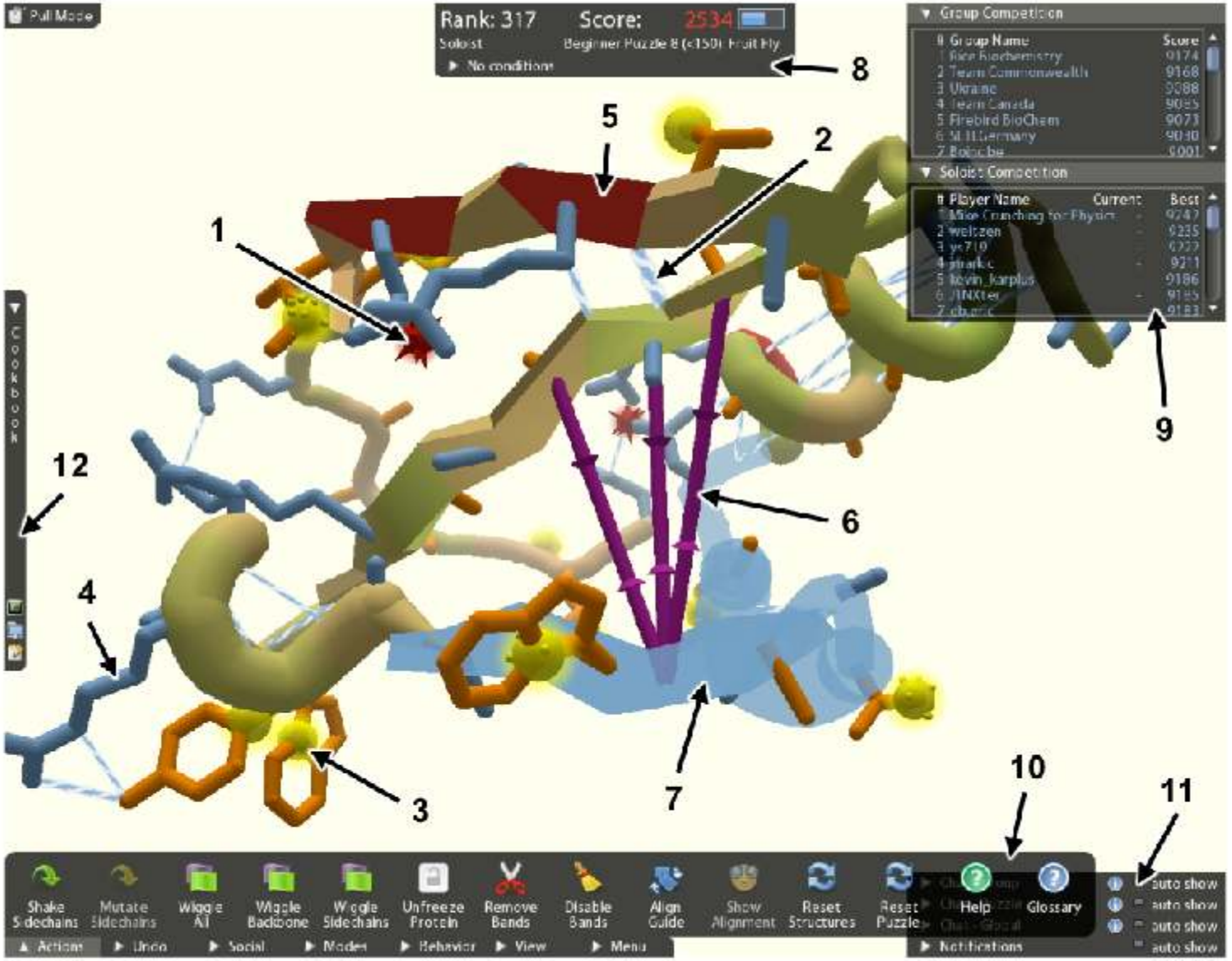
- Gamification of program verification
- Game corresponds to correctness condition
- Game utilizes physical intuition
- Game is playable by anyone
- Game allows application of human insight
- Goal: cheaper verification $\Rightarrow$ more verification

http://cs.washington.edu/verigames

Play it now:
http://games.cs.washington.edu/verigame/pipejam-20120709/

FoldIt

- Proteomics game at UW
- Effectively created the genre of games that solve hard problems
- Three Nature papers in under 2 years
- Over 240,000 players, 200+ new per day


Pull Mode
Rank: 317
Score: 2534
Soloist
Beginner Puzzle 8 (<150): Fruit Fly
No conditions
Group Competition
Soloist Competition
Player Name
Current
Best
Cookbook
1
2
3
4
5
6
7
8
9
10
11
12
Shake Sidechains
Mutate Sidechains
Wiggle All
Wiggle Backbone
Wiggle Sidechains
Unfreeze Protein
Remove Bands
Disable Bands
Align Guide
Show Alignment
Reset Structures
Reset Puzzle
Help
Glossary
Actions
Undo
Social
Modes
Behavior
View
Menu
Notifications
auto show