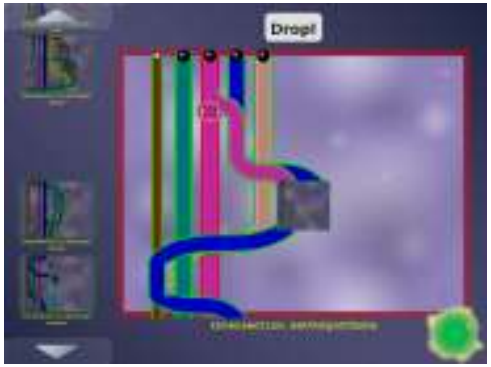


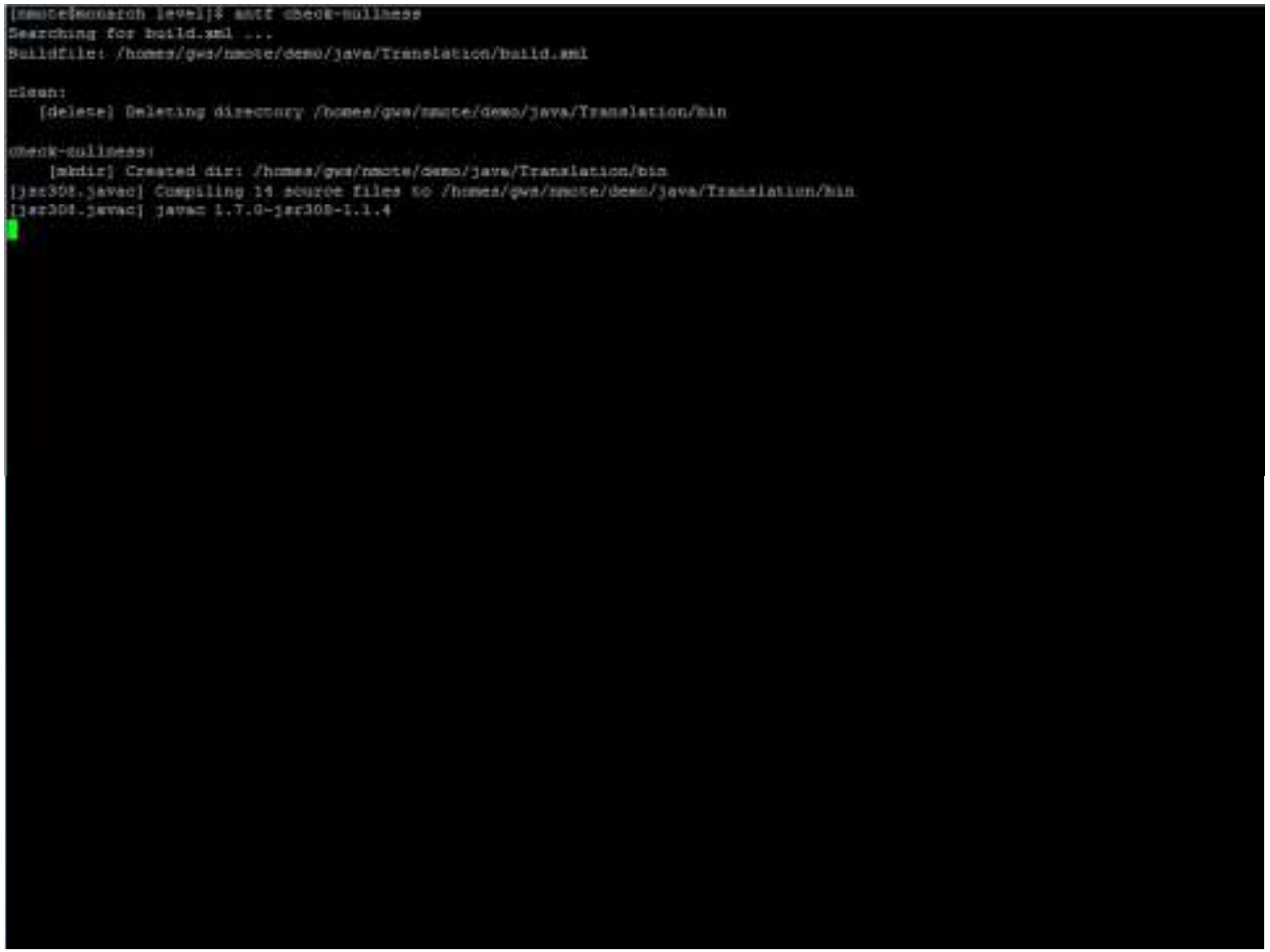
Verification games: Crowd-sourced software verification

Michael Ernst
University of Washington

Joint work with
Seth Cooper, Werner Dietl, Stephanie Dietzel, Nat Mote, Tim Pavlik, Zoran Popović, Jeanette Wing

Angry Birds

Software verification



Which is more fun?

- Play games
- Prove your program correct

Crowd-sourced software verification

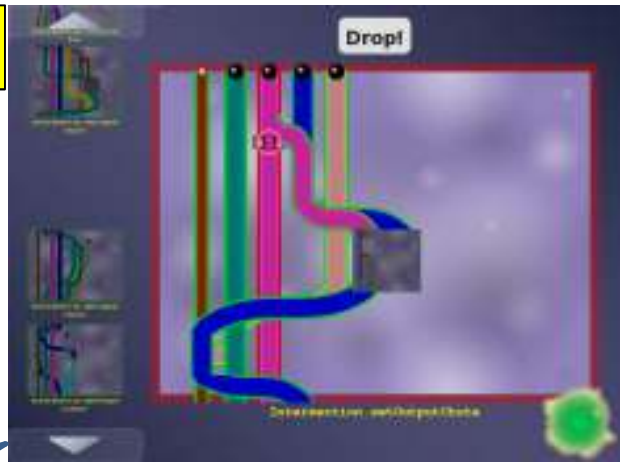
Goal: Verify software while you wait for the bus

- 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

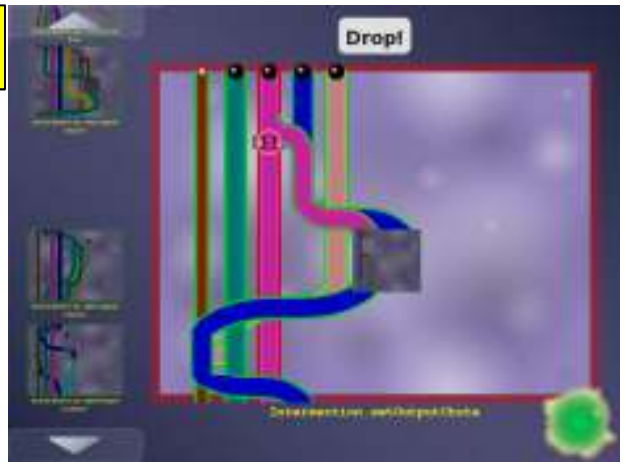
Completed game

Pipe Jam game demo

Code

Automatic translation

Game

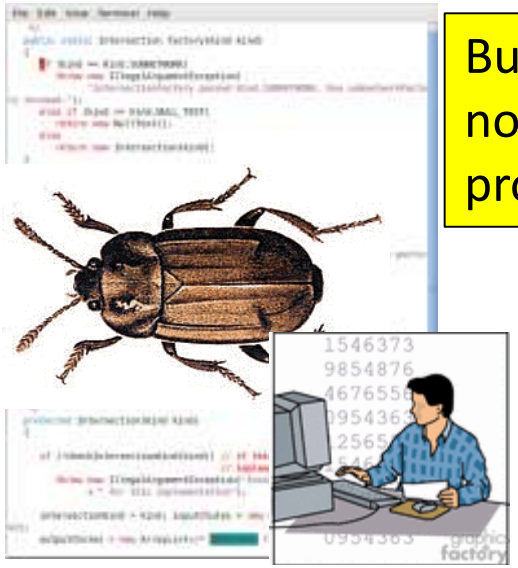


Free labor

Completed game with buzzsaws

Automatic translation

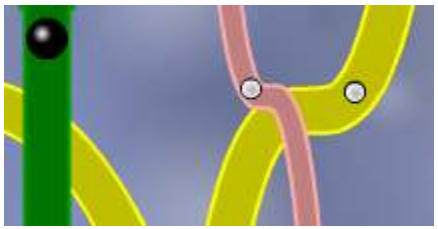
Bug detected, notify programmer



Highly-skilled, expensive labor

Program ↔ game correspondence

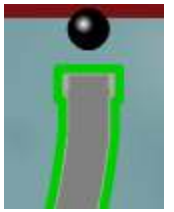
Idea: dataflow



Pipe ↔ a variable

Pipe width ↔ a property of the variable (type)

Ball ↔ a value



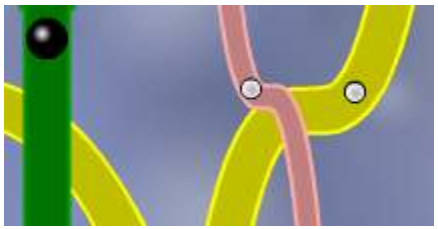
Ball size ↔ a property of the value

Pinch point ↔ requirement

Unmodifiable pipe/ball ↔ requirement

Example: encryption

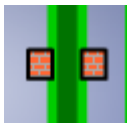
Goal: no cleartext is sent over the network



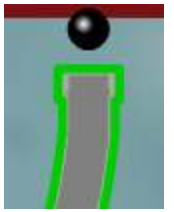
Pipe ↔ a variable

Pipe width ↔ narrow: encrypted, wide: cleartext

Ball ↔ a value



Ball size ↔ small: encrypted, large: cleartext

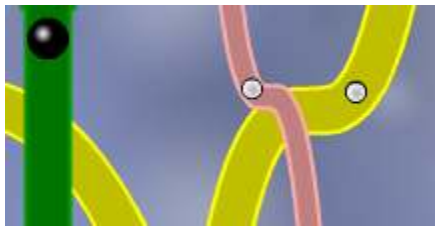


Pinch point ↔ network communication

Unmodifiable pipe/ball ↔ cleartext from user

Example: null pointer errors

Goal: no dereference of `null`

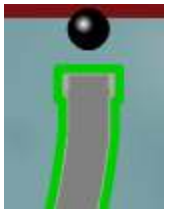


Pipe ↔ a variable

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`**

Other examples

- SQL injection
- unintended side effects
- format string and regexp validation
- incorrect equality checks
- race conditions and deadlocks
- units of measurement
- aliasing
- …

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)

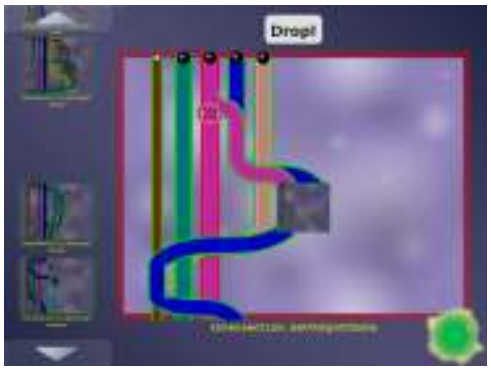
   Hypothesis: no for correct, verifiable programs,

   yes for incorrect or unverifiable programs

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

   Also see FoldIt (protein folding)

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