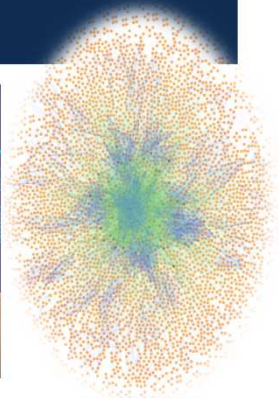
SAND2017-6338PE



Scientific Approaches to Cybersecurity: Design and Analysis of Complex Digital Systems

Robert Armstrong and Jackson Mayo

Sandia National Laboratories, Livermore, CA

May 24, 2017

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

What makes cybersecurity hard?

Sandia National Laboratories

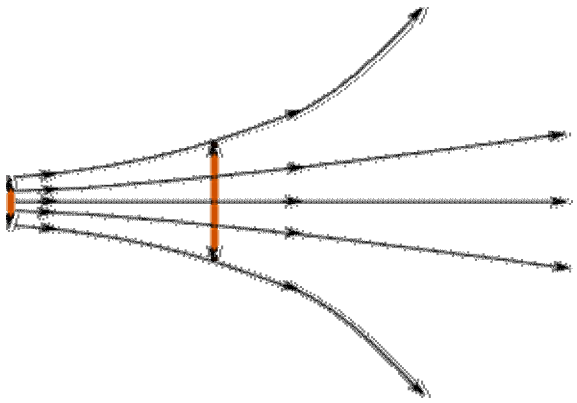
- We need to model cyber systems to say anything scientific about them, but models of digital logic behave differently from most models used in science
- Or to put it another way: Cybersecurity is a complex systems problem
- Not just digital logic – also need to model:
  - Human users and attackers
  - Supply chain (how the system came to be)
  - Physical infrastructure that the system interacts with (e.g., SCADA)
- We consider "trust" as a generalization of security that includes potential attacks during the *creation* of the system (e.g., planted vulnerabilities)

Cyber: inherently stable to < 1 bit, usually chaotic to > 1 bit perturbations

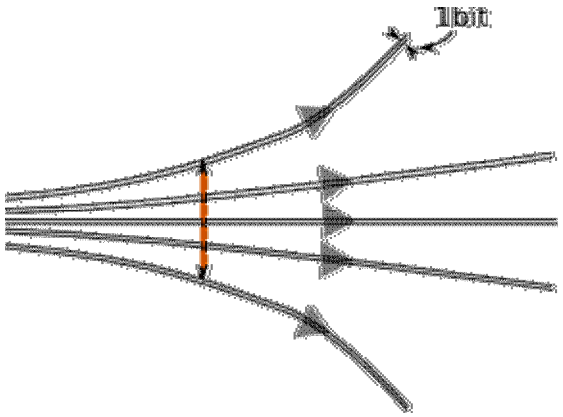
Sandia National Laboratories

- Cyber systems are both unpredictable and deterministic
  - Hard, at first, to wrap your head around
- The same input to a generic program always outputs the same result
- A one-bit variation in the program, or just the input to the program, produces wildly different results (Lyapunov exponent > 0)

Chaotic dynamical system



Generic digital system



Finding the right approach for the job

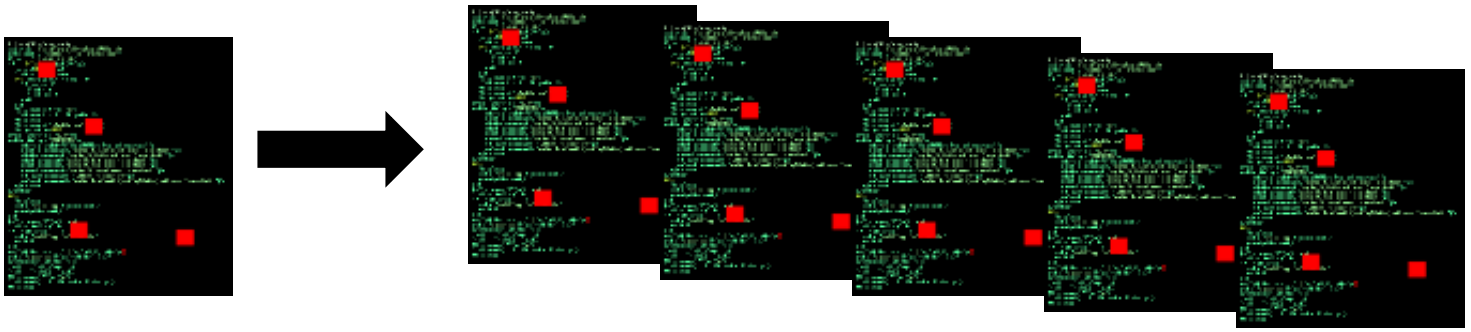
Sandia National Laboratories

- Sandia is interested in cyber systems broadly, but especially in *digital controllers for high-consequence physical systems*
- There are at least three levels of scientific inquiry on the security of cyber systems:
  - Simulation and testing of particular (existing or proposed) cyber systems
    - Explores behaviors one at a time; by itself can't assess security quantitatively
  - Mathematical analysis of particular (existing or proposed) cyber systems
    - Seeks to establish security as a quantitative global property of a model
    - Well-established example: mathematical proofs of cryptographic security
  - Design of new cyber systems that are "secure by construction"
    - Creates a system in a mathematically constrained fashion so the desired properties are built-in
- The adequacy of the model (verification and validation) is a perennial problem!
  - The most rigorous analysis may be irrelevant if an attacker can operate outside the model (see $5 wrench)

Economies of scale in computing: Friend and enemy

Sandia National Laboratories

- Enormously complex hardware and software is created at enormous cost
  - Cost is recouped by stamping out millions of identical copies



- A kid in his basement can make it do something interesting but unknown (unpredictable). He can be certain he can do the same thing to your desktop PC (deterministic)
- In the general case, all digital designs share these problems

Solution: Make the design less general, more analyzable

Digital systems are prototypical of broader complex systems

- One way of defining "complex" systems: those that behave as large-scale information networks and do not yield to traditional equation-based analysis
  - Complex systems can be engineered or evolved

Infrastructure

Computers

Societies

- Fundamental basis for their intractability: Turing's halting problem
  - Once a system (whether designed with digital logic or otherwise) is equivalent to a sufficiently powerful computer, its behavioral properties cannot be predicted in the general case
  - The digital cybersecurity problem illustrates this difficulty in its purest form

The complexity problem has its roots in theoretical computer science

Sandia National Laboratories

- Theorem (Turing 1936, Rice 1953): No algorithm exists to predict a priori the behavior of a generic information processing system
  - i.e., such a system is undecidable even if deterministic
  - Practical significance: A real system, with a finite exponentially large number of states but otherwise generic, is *effectively* undecidable – in particular, simulation or testing cannot tell us all its possible behaviors

?
>

- We need to bound all possible behaviors to quantify safety and security

The theory has direct engineering implications

Sandia National Laboratories

- A digital system *created arbitrarily* cannot be predicted or bounded
  - You have no idea how the system will respond to the vast number of inputs you *haven't* tested
  - This explains why bugs and vulnerabilities are commonplace
  - Once the state differs by even *one bit* from what you expected – due to a mistake, a natural fault, or an attack – all bets are off
- We need ways to tame this "wild west" digital complexity
- We must design digital systems specifically for analyzability and robustness
- Similar problems are faced in outside applications, but Sandia faces them in extremes
  - Extreme consequence, extreme environments, extreme scale

The solution space uses the mathematics of complex systems

Sandia National Laboratories

- Formal methods (reduced complexity)
  - Automated reasoning about all possible behaviors within a model – widely used in industry for critical digital devices
  - Model checking, theorem proving
  - Scaling limitations, though power and tractability have improved over time
- Complex systems theory (structured complexity)
  - Probabilistic analysis of response of networks to perturbations
  - Biologically inspired architectures such as diversity
  - Well suited to understand emergent system-level robustness, but only sparingly applied to engineered digital systems
- In both strategies, systems must be constrained to be analyzable
  - Ideal approach is to consciously design-in analyzability and robustness along with functionality

Formal methods reason about all possible digital behaviors

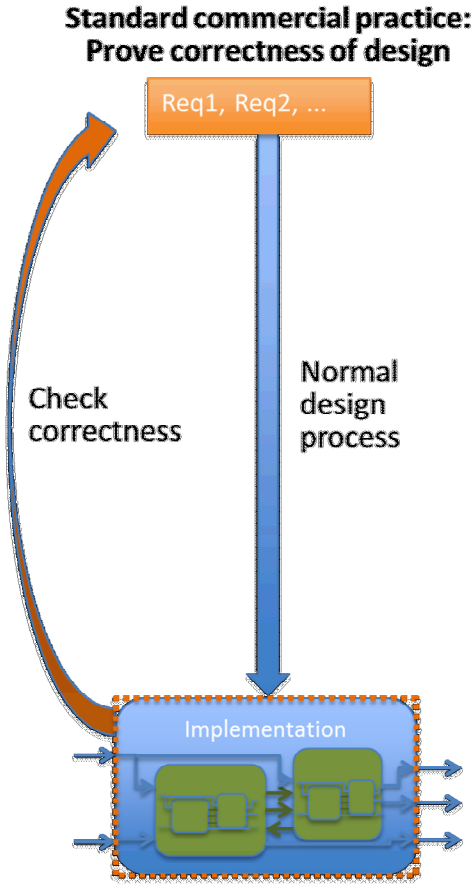
Sandia National Laboratories

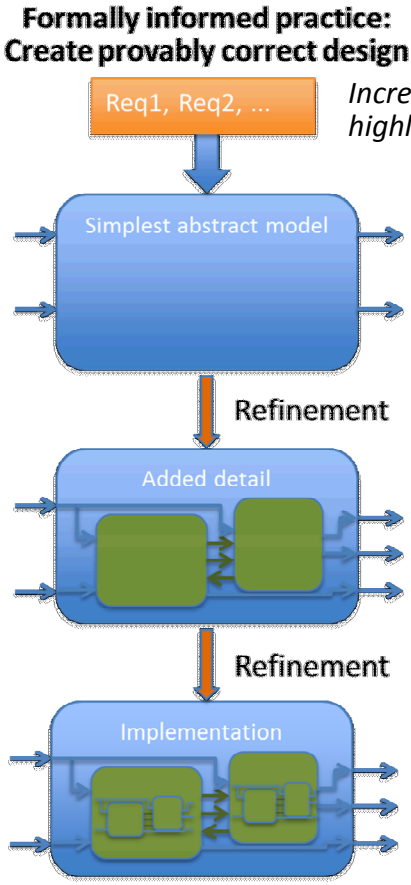
- Example: Can this 63-input logic circuit ever output TRUE?
- (b_1 XOR b_2) AND (b_2 XOR b_3) AND (b_3 XOR b_4) AND … AND (b_{62} XOR b_{63}) AND (b_{63} XOR b_1)

- Exhaustive black-box testing:
  - Try all $2^{63} \approx 10^{19}$ input values to conclude: no, never outputs TRUE

- Automated reasoning by a formal "satisfiability (SAT) solver" that seeks a TRUE output (common off-the-shelf tool):
  - Try b_1 = TRUE: then b_2 = FALSE, …, b_{63} = TRUE, b_1 = FALSE: *contradiction*
  - Try b_1 = FALSE: then b_2 = TRUE, …, b_{63} = FALSE, b_1 = TRUE: *contradiction*
  - This is a proof that *no* input can make the circuit output TRUE, based on analyzing the circuit as a mathematical object

Work aims at “engineered trust” via formally informed design



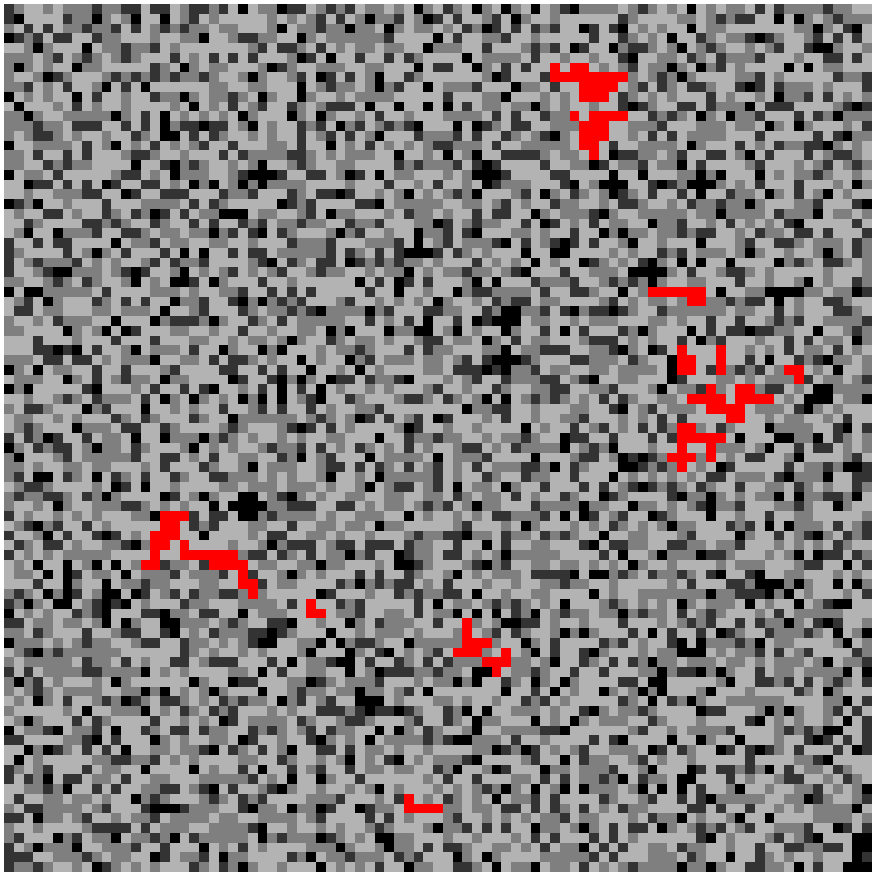


Increase proof tractability for highly stringent requirements

“Self-organized criticality” is a simple example of digital emergent behavior

Sandia National Laboratories

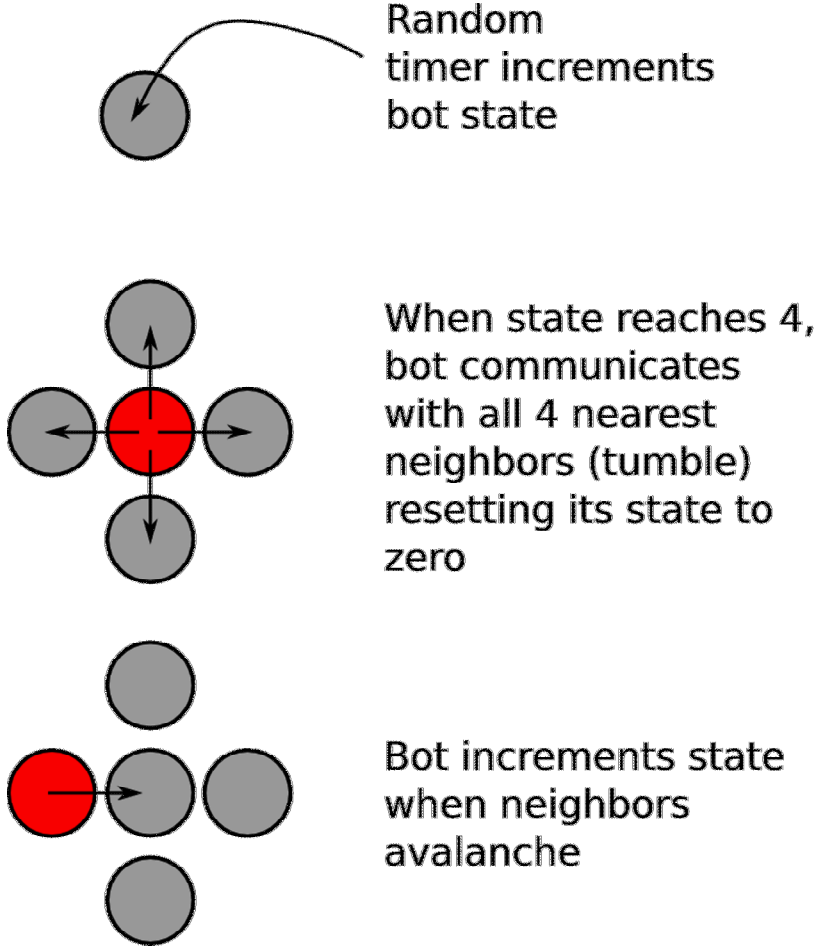
- SOC (Bak et al. 1987) is *spontaneous* development of fractal phenomena with power-law distributions
  - Similar to thermodynamic criticality but without tuning
- System shown is “Sandbot”: cyber model of coordinated malware
- Inspired by “sandpile model”: physics-like cellular automaton
  - Sand is sprinkled randomly
  - Avalanches occur at all scales



8100 machines running Sandbot: Each pixel is a single machine; network storms appear in red

“Sandbots” communicate via standard protocols and obey simple rules

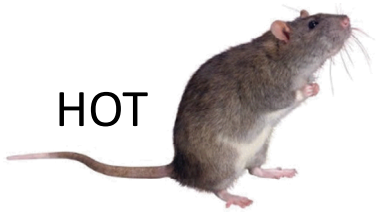
- Simple “botnet” is based on a square lattice populated by bot nodes
- Simple rules determine when a bot communicates with its 4 neighbors
- Despite the model’s simplicity, behavior at large scale is unexpected and rich
- The model has been implemented using emulation, in which a computer uses “virtual machines” to perform an extremely faithful simulation of other computers



Robustness is key to understanding systems with "organic" behavior

Sandia National Laboratories

- Highly optimized tolerance (HOT, Carlson & Doyle 1999): Systems *designed* or *selected* to perform well despite perturbations
- HOT systems exhibit power-law distributions but have organic structure (not self-similar or fractal)

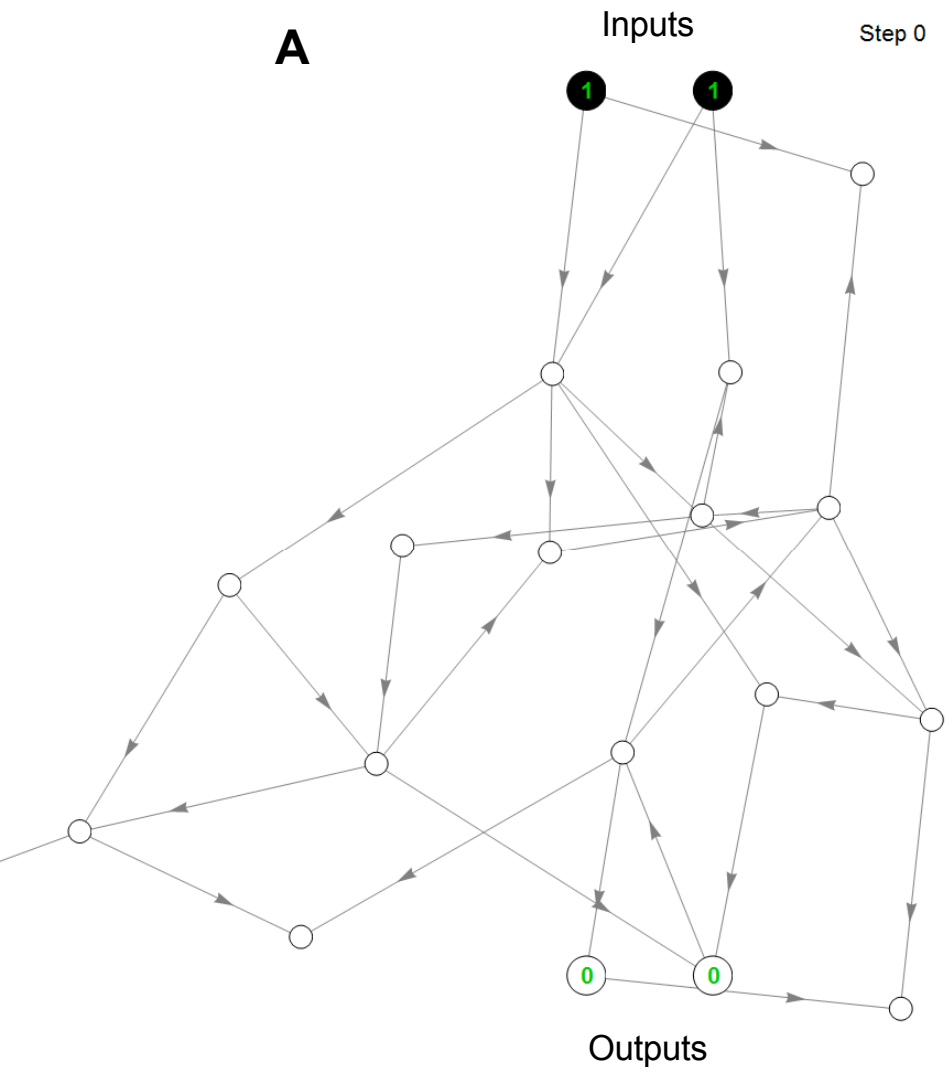
HOT  ≠  Fractal

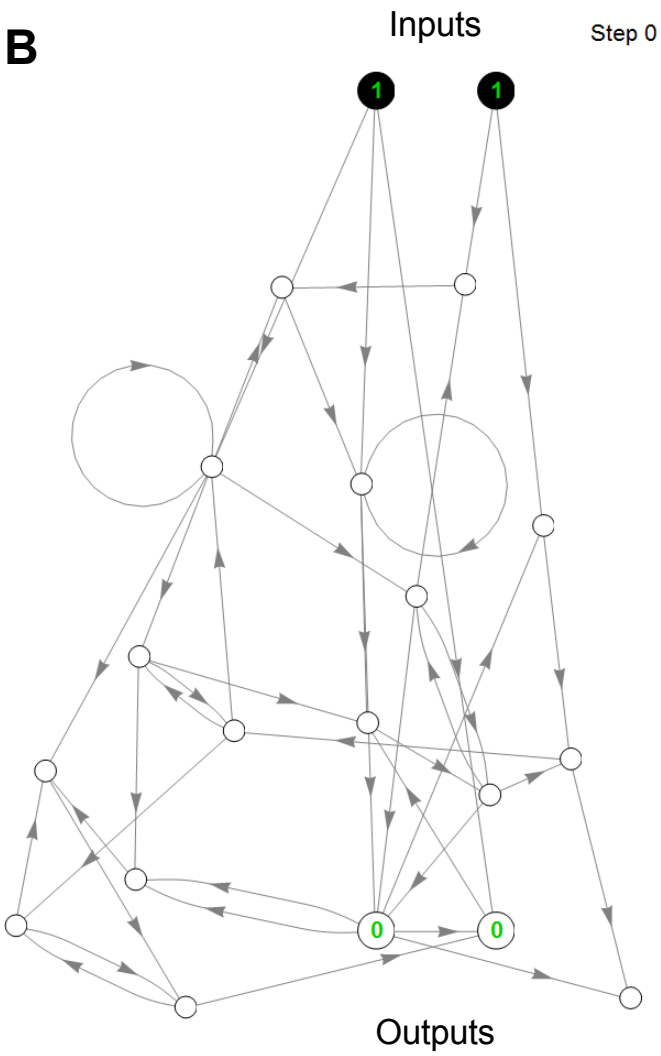
- Adapted robustness to one set of perturbations induces extra fragility to different perturbations
- Indeed, rare but catastrophic failures are seen in highly engineered/evolved systems
  - Electrical blackouts, financial panics, epidemics, massive cyber attacks, etc.

Complexity theory shows ways to address "whole system" robustness

Sandia National Laboratories

- Cybersecurity vision: Create high-consequence digital systems in new ways, so that they are analyzable
  - Seek to understand computers as dynamical systems
- Toy example: "Growing" a digital circuit to add two 1-bit numbers – a half adder
- There are many ways of composing logic gates to implement this functionality
- Next slide shows two such "grown" (actually randomly sampled) networks; each performs as a half adder when run for 20 steps
  - Shown correctly adding 1 + 1 to get the binary result 10
  - They also respond correctly to the other possible inputs

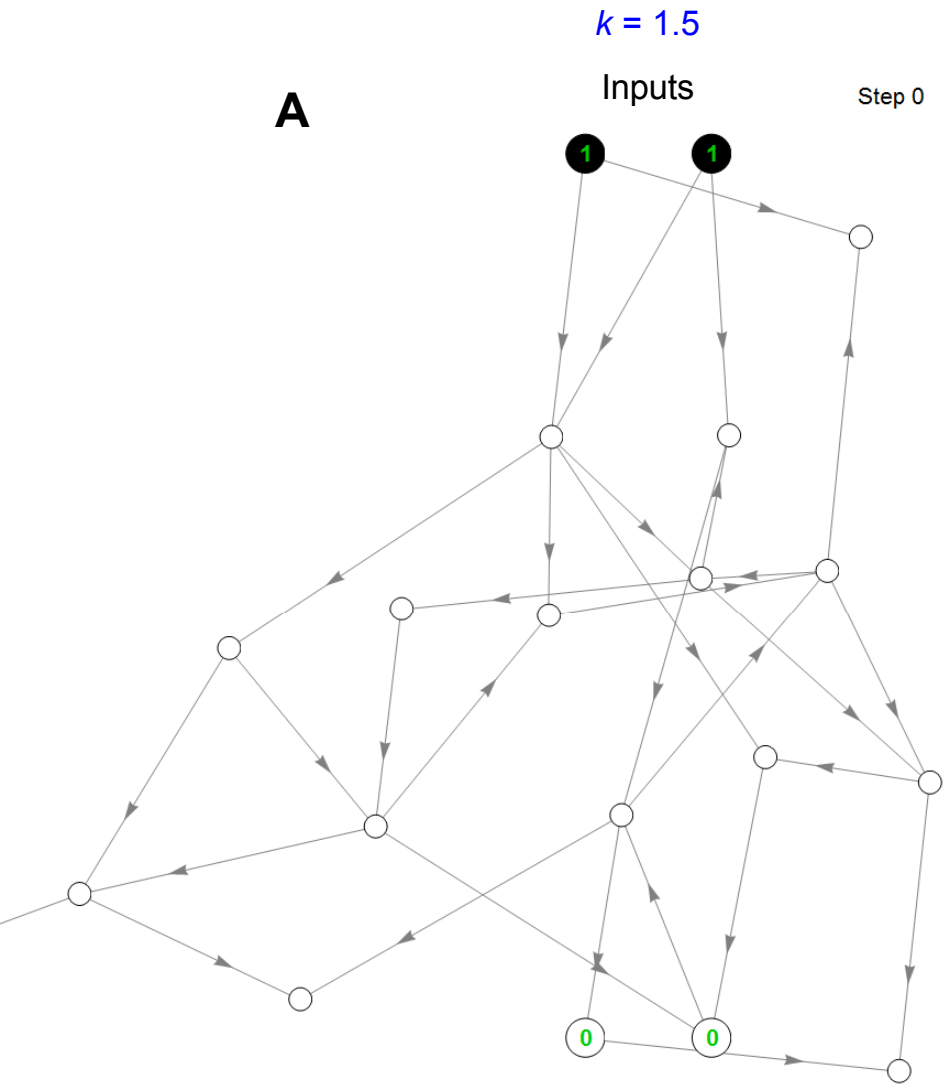

A
Inputs
Step 0
1
1
0
0
Outputs

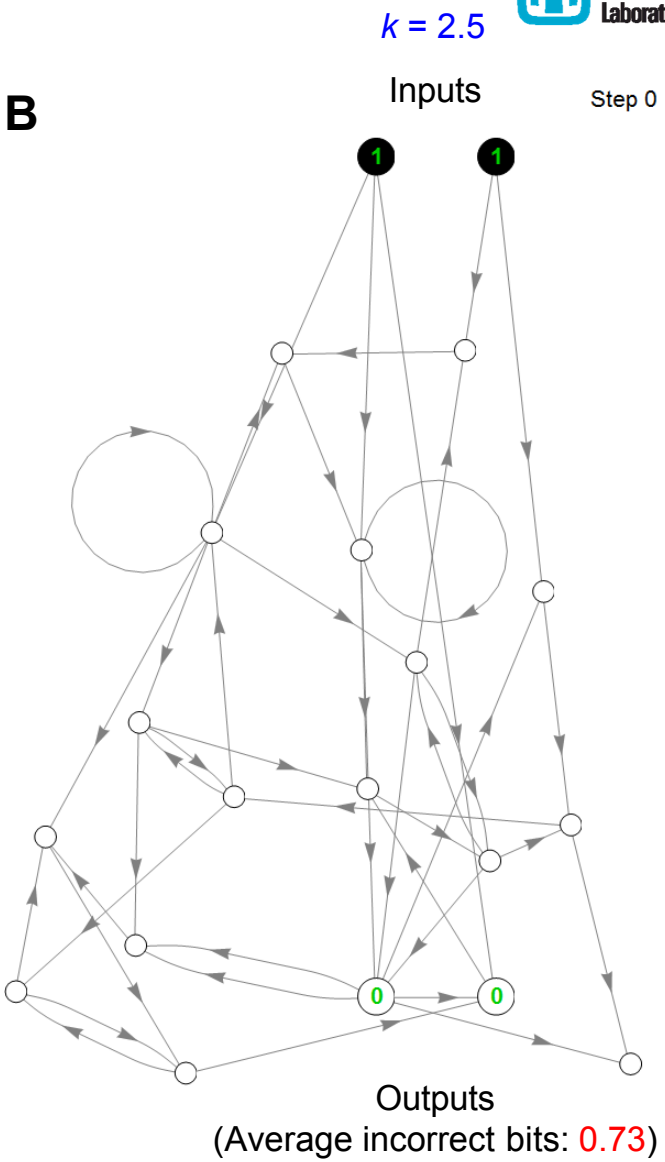

B
Inputs
Step 0
1
1
0
0
Outputs

What distinguishes the two implementations? *Resilience*

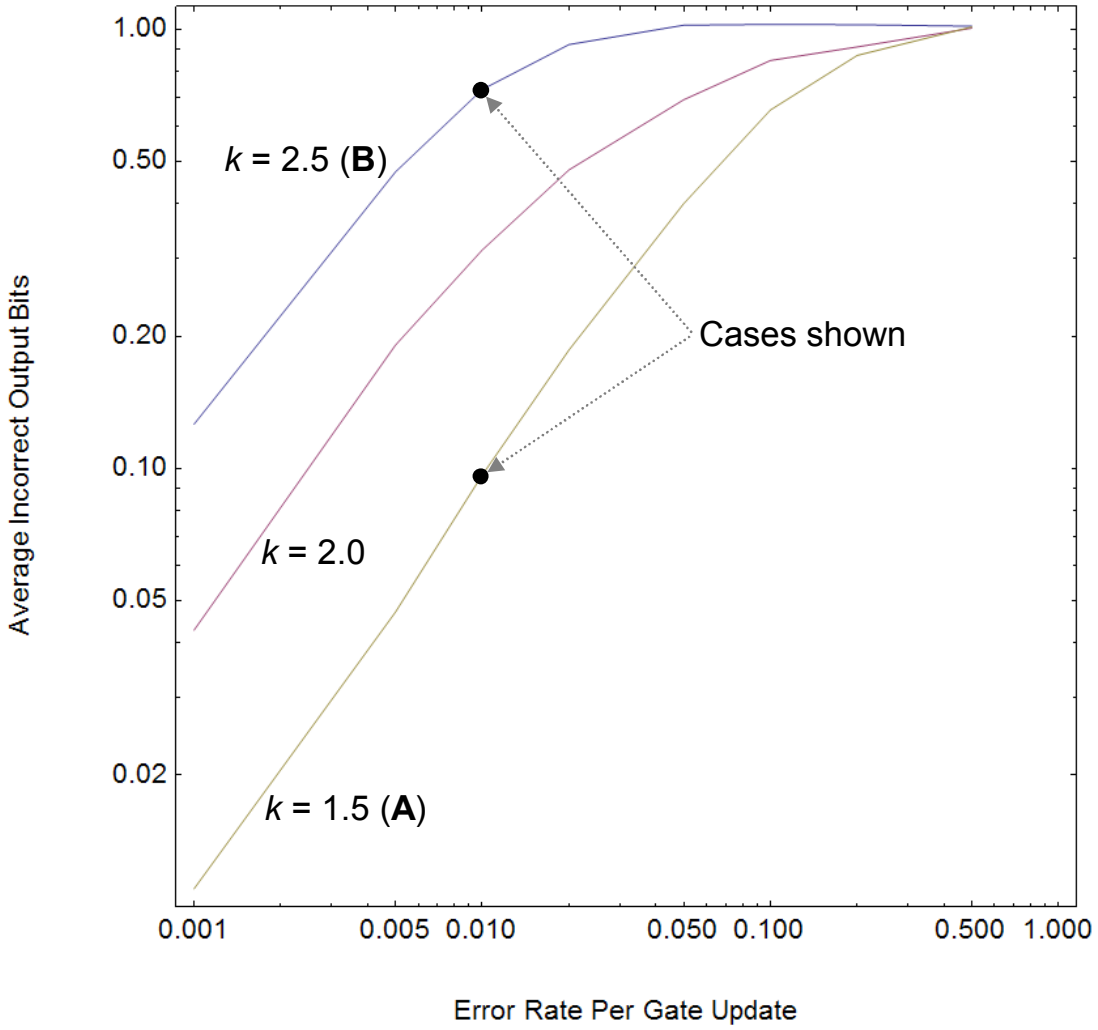
Sandia National Laboratories

- Resilience of a digital model to bit errors can be assessed via growth or damping of perturbations ("Lyapunov exponent")
  - Bit errors can represent breakdown of digital model, or effect of untested states within the digital space
  - Networks transition from stable to unstable based on connectivity and logic (generalizing Kauffman 1969)
- Next slide: runs with 1% error rate per update
  - States that deviate from the ideal run are outlined in red
- Network A has much less error in final output (greater resilience) than network B – why?
  - Here, average inputs per node (*k*) makes the difference


k = 1.5
A
Inputs
Step 0
1
1
0
0
Outputs
(Average incorrect bits: 0.10)


k = 2.5
B
Inputs
Step 0
1
1
0
0
Outputs
(Average incorrect bits: 0.73)

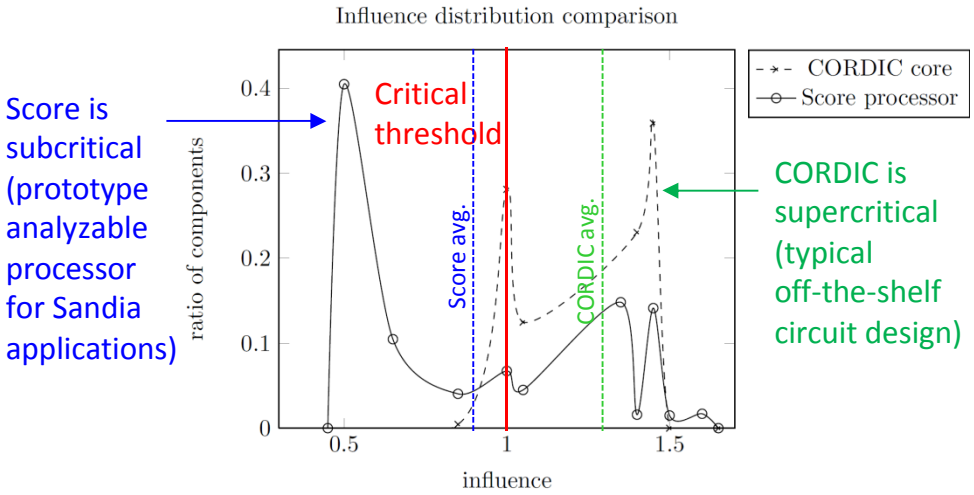
Example illustrates quantifying resilience implications of designs



- Results for these half-adder networks can be obtained by brute testing
- Systematic relations to real-world design parameters enable assessing potential catastrophic failures too rare to be found reliably through testing

Complexity theory provides insight on real-world circuits

- "Influence" measure in BNs is a more precise generalization of "inputs per node" (Seshadhri et al. 2011)
  - If Avg. Influence > 1 (supercritical), network is unstable
  - If Avg. Influence < 1 (subcritical), network is stable
- Example: Score processor shows signs of enhanced resilience – consistent with its goals of analyzability and predictability



Area for research: What makes resilient complex systems quantifiable?

Sandia National Laboratories

- Smoothness (a.k.a. stability, subcriticality) makes a system:
  - Predictable (you can extrapolate its behavior to a new situation)
  - Resilient (it tends to maintain its behavior under minor faults)
  - Evolvable (you can make small changes to it, and it remains usable)
- Smoothness *of particular observables* is common in physics
  - Amid molecular chaos, continuum equations apply when we're concerned with thermodynamic (averaged) behavior
  - Extend this to adaptive complex systems *with respect to the behaviors that are selected for?* (These are typically *not* averaged behaviors)
- Ability to bound the effect of perturbations is crucial for:
  - Inferring that a model will be predictive under conditions that *differ* from those used to test it (V&V), and inferring how much the model behavior may change due to *variations* in input parameters (UQ)

A lesson for complex systems that are *not* resilient

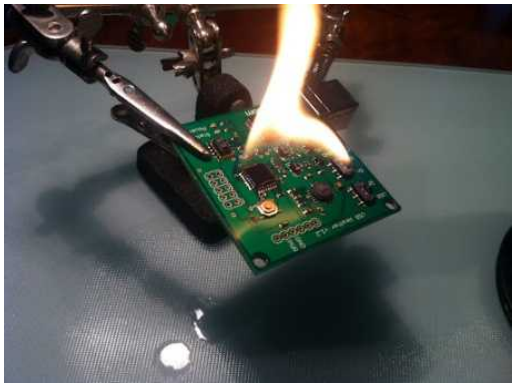
Sandia National Laboratories

- Beware of applying techniques that assume smoothness/ stability to complex system observables that have no reason to be smooth/stable
  - Could be garbage in, garbage out
- Today's complex digital systems are *not* designed in a thoroughly adaptive way, and lack inherent stability
  - Hence, highly susceptible to failures and attacks
  - By the same token, also difficult to model predictively
- The *difficulty of V&V'ing cyber(-physical) models* and the *difficulty of securing cyber(-physical) systems* are two sides of the same coin
  - The cyber V&V problem *is* the cybersecurity problem – both need to be solved together

Complexity theory enables analysis of the consequences of hardware errors

Sandia National Laboratories

- Digital hardware is generally unpredictable
  - Even when it's working as designed
    - Complexity can exceed the reach of formal methods
  - But doubly so when hardware is compromised
    - Due to out-of-spec extreme conditions
    - Radiation, thermal, intentional physical subversion



- Most tools and applications assume perfect hardware
  - Conventional formal methods prove properties for logic working as designed
- High-consequence systems require efforts to assure safe behavior in abnormal physical environments
  - Especially when coupled to vast digital state spaces in a cyber-physical system
- Example: Mixed-signal simulation can elucidate the digital imprint (e.g., bit flip pattern) of a physical insult (e.g., radiation) on a circuit
  - Using analog model for the part of the circuit subjected to the insult

Formal analysis can incorporate a digital upset model

- Exhaustively prove correct function of example half-adder networks using open-source model checker NuSMV

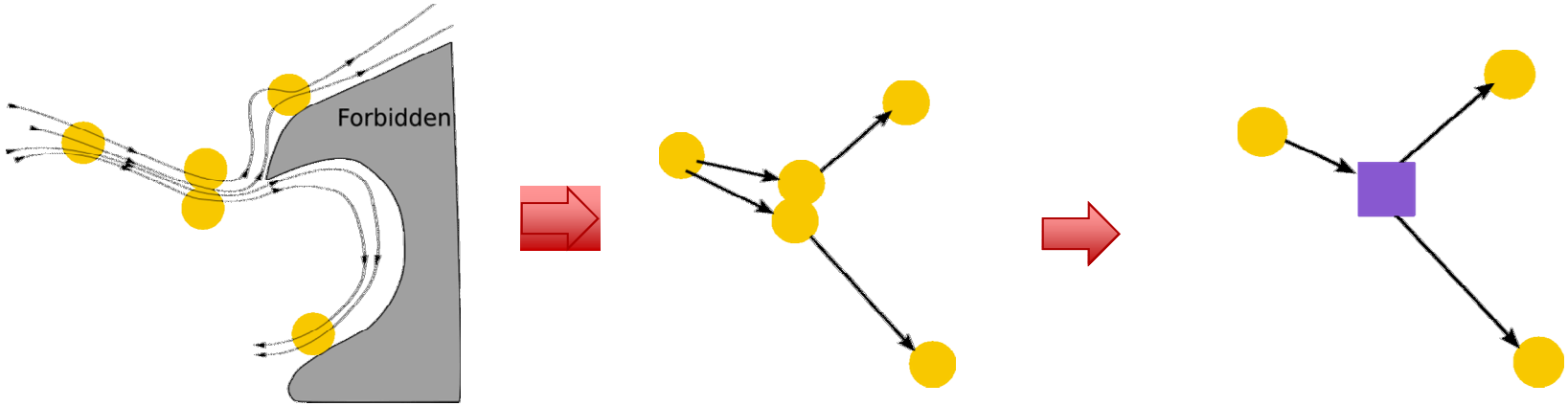
```
tfern02 := (xfer02 >> (n01 :: n03)) & 0ub4_0001;
...
LTLSPEC F  ((clock = 20) & (n18 = (n00 & n01)));
```

- Currently using simple upset assumption
  - Allow any single bit flip within a range of time steps
  - To be generalized by extracting upset models from physical simulation
- Initial formal results confirm insights from complexity theory
  - Chaotic network is susceptible: Corruption can arise from any time step
  - Quiescent network can be corrupted only if upset occurs in the last 5 of 20 time steps: self-healing otherwise

Abstraction/composition is key to analyzing large cyber-physical systems

Sandia National Laboratories

- Abstracting continuous equations into digital representation
- Abstracting detailed digital representation into an over-approximation that still permits proofs of key properties

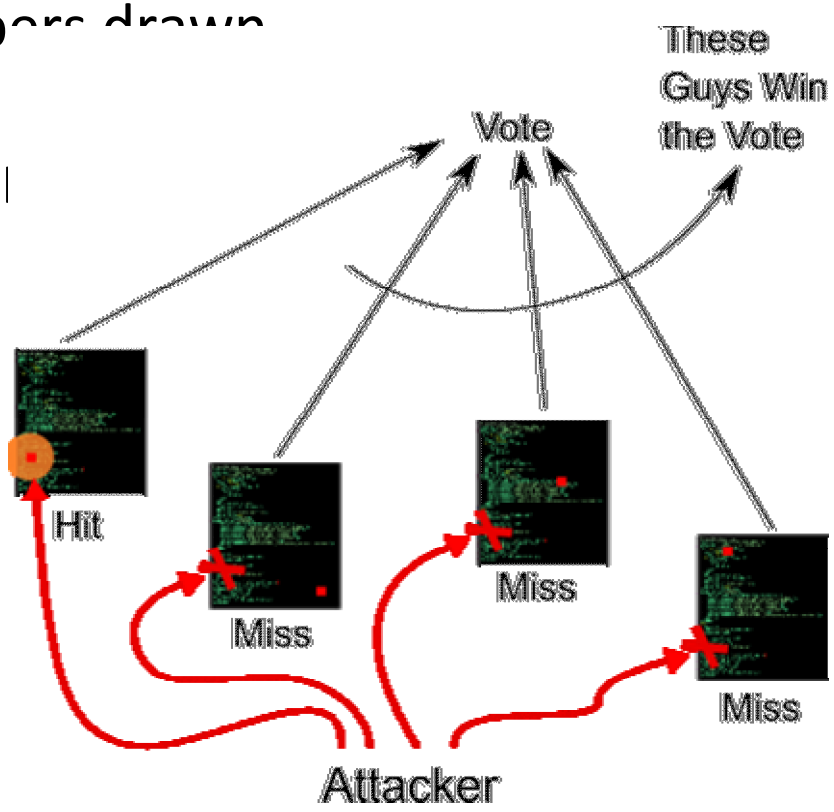


Continuous *Digitized* *Abstracted*

- Composition that preserves formal safety/security properties admits a divide-and-conquer approach

“Diverse redundancy” is a complex systems technique for trust

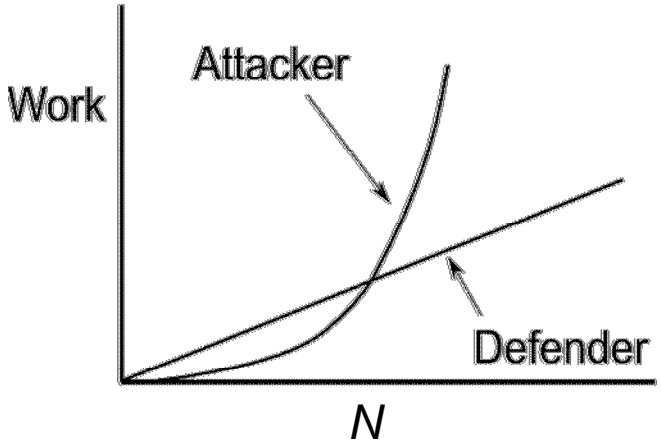
- Use a voting system with members drawn from a *set* of implementations
  - Input processed by each in parallel
  - Outputs compared to determine response
  - A type of “moving target”
- Keep intended functionality while varying vulnerabilities over space and time
- Similar to redundancy for physical fault tolerance



- Diversity leverages a simple “trust anchor” (the voting unit) for cybersecurity benefits at the complex system level

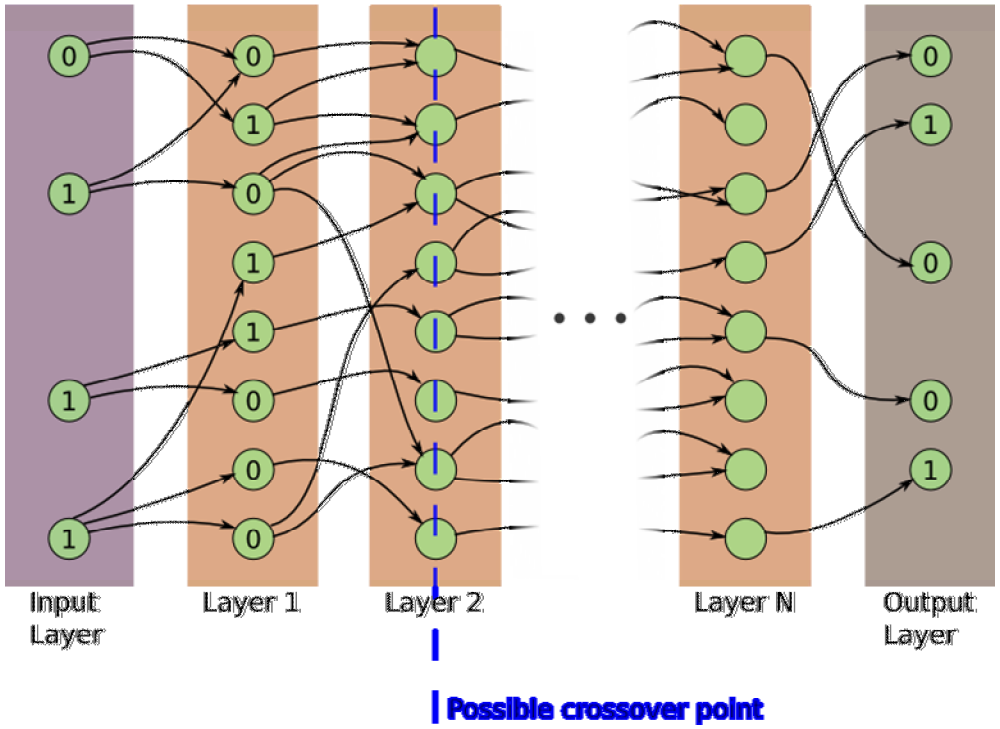
Analyzable statistics arise from an ensemble of undecidable programs

- For a specific feature set, there is a probability P_v that a particular member of the set of implementations will be susceptible to vulnerability *v*. For a voting system of size *N*:
  - The probability of success for the attacker is $(P_v)^{N/2}$
  - The attacker "work" is the expected number of tries: $(1/P_v)^{N/2}$
  - The work for defender is the cost of producing *N* implementations: $\propto N$



“Genetic programming” can produce diverse digital implementations

- Example: “Grow” realistically imperfect circuits represented as BNs
- Simple BN specification for “string recognizer”: Output 1 for a particular input bit combination (“password”), 0 for all other inputs
  - Faults are inputs other than password that produce 1
- Use “feedforward” BNs with a modular structure that can be recombined genetically, to find many circuits that perform well (but not perfectly) in meeting the specification



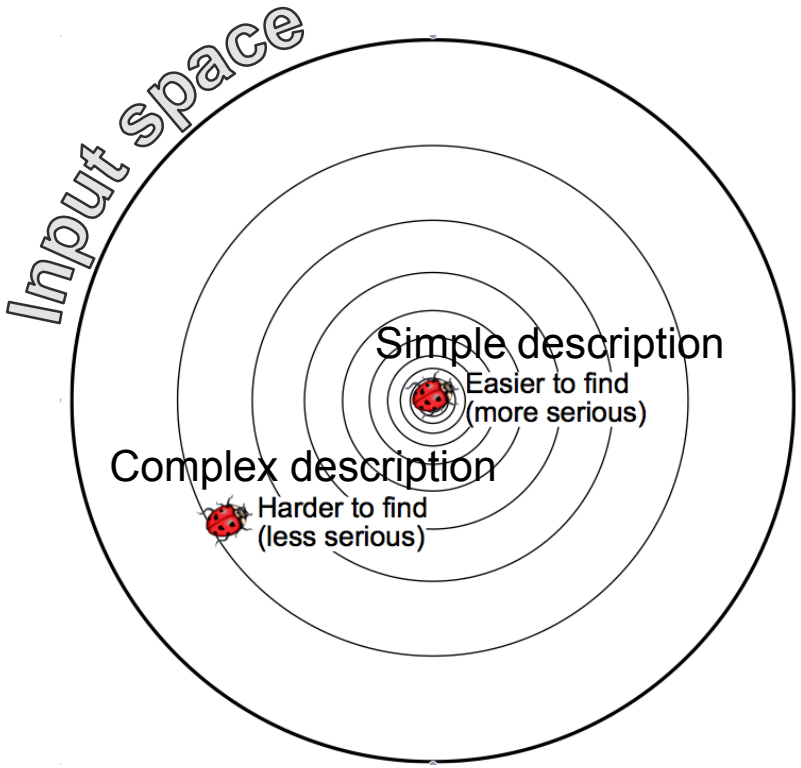
String recognizer example illustrates role of testing vs. formal methods

Sandia National Laboratories

- Most error rates for "grown" recognizers < 1%
  - Recognizing bit-strings of up to 64 bits long
  - Some low bit length (8 bit) strings have zero error rates and are proved by NuSMV to be perfect
- The NuSMV model checker itself has perfect logical consistency for every circuit tested (> 7000)
  - All recognizer circuits tested to have errors are detected faulty
    - Counterexample emitted by NuSMV proves that this is so
  - All perfect circuits (whether grown or human made) are proven perfect and have no errors in testing
  - Even circuits that have 0 tested errors are shown to be faulty by the model checker
    - Examples of 32 bit recognizers show 0 errors for 1000 random tests, yet are shown to be flawed by model checking

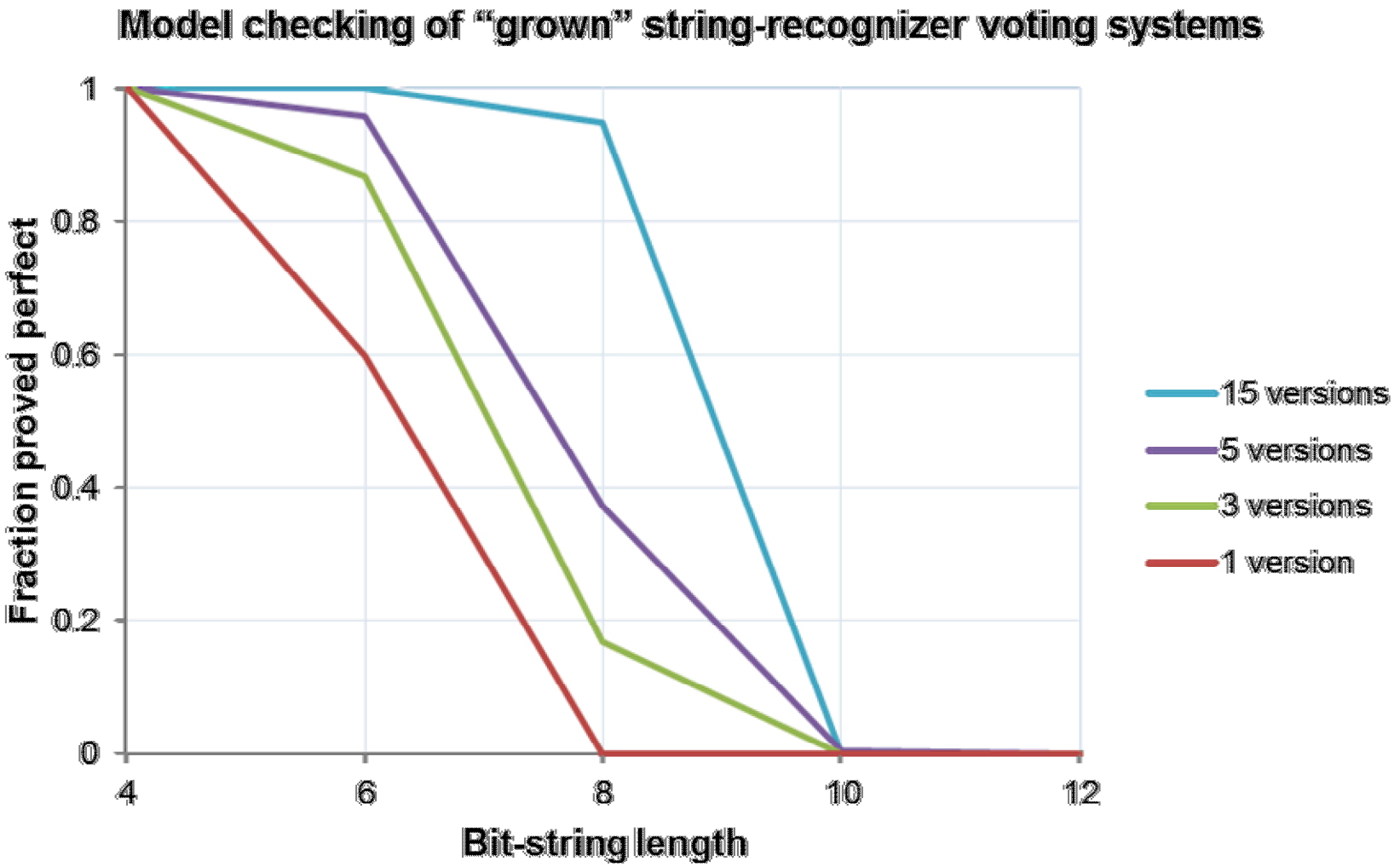
Complexity measures suggest targeted fuzzing strategies

- Evolved and designed systems have coherence that makes it useful to fuzz in "simpler" spaces
- Example: Fuzzing string recognizer with patterns *close to gold string* is more likely to find faults



- More generally: Inputs that have a *simple description* (relative to available information) should be targeted for coverage because they form a smaller "corner" space (also more attractive to attacker)

Formal analysis of diverse string recognizers exposes voting benefit

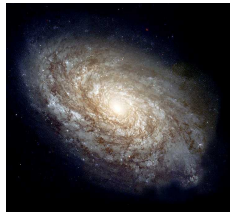


Supplemental slides

Need to assure trust for high-consequence information systems

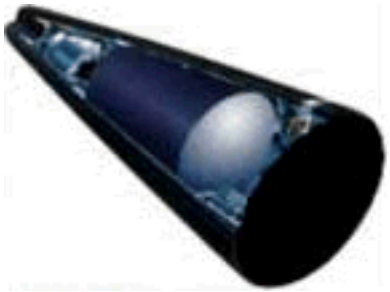
- As high-consequence systems incorporate digital components, exhaustive testing/simulation becomes infeasible

?



Simple calculator has more possible states than number of particles in known universe

- Assessing trust in such systems is vital for Sandia missions



Nuclear weapons (NW)

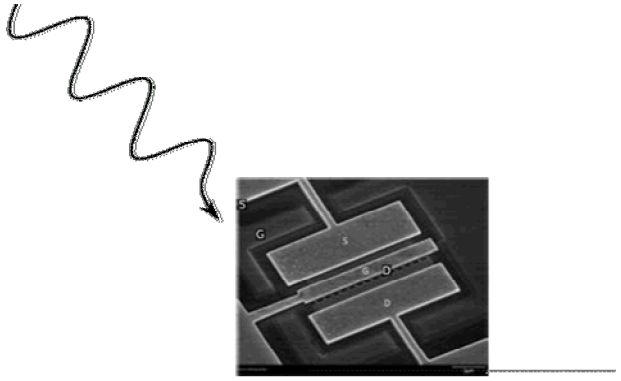
Cyber/computing

Energy infrastructure

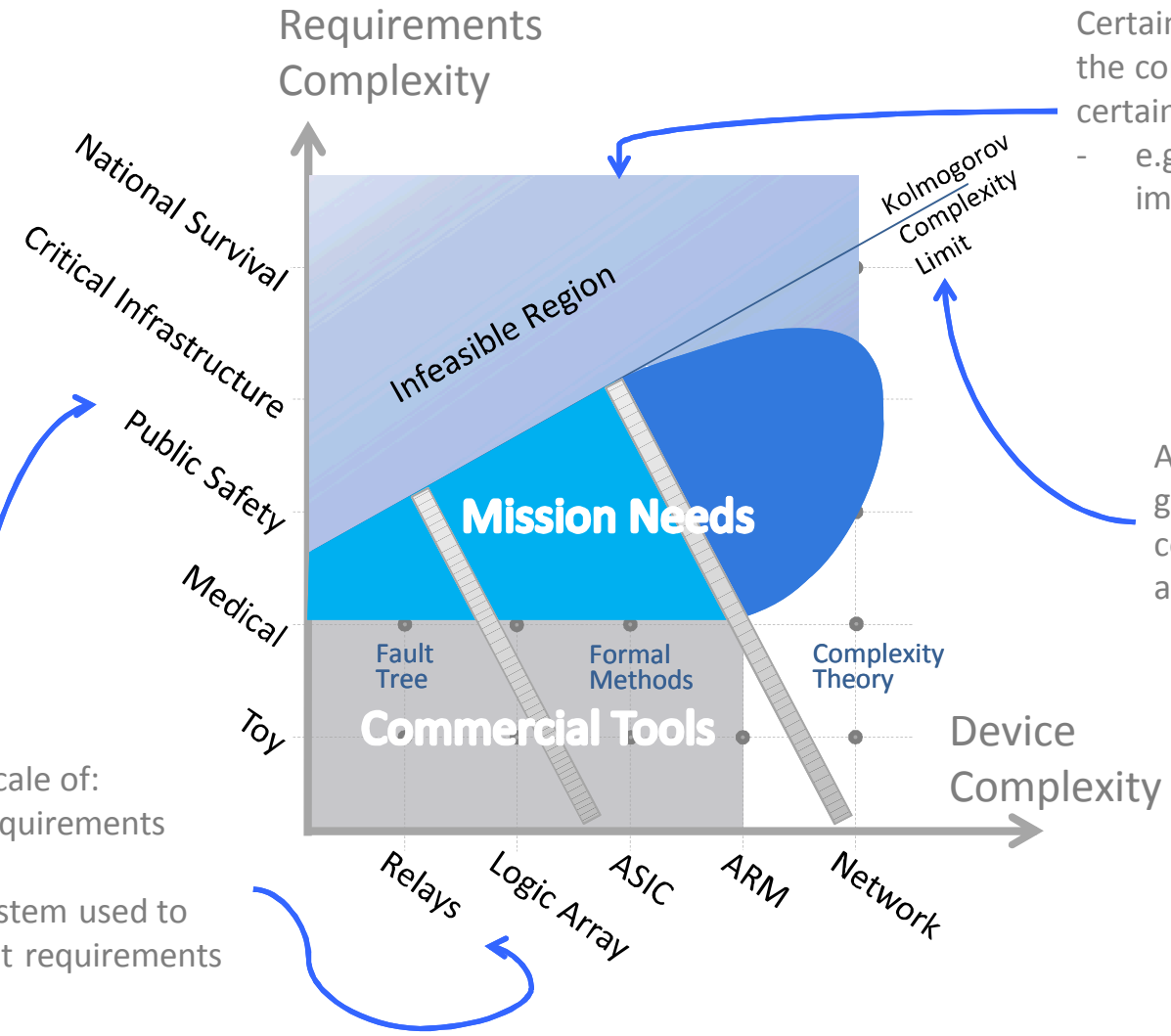
Need to verify increasingly complex behavioral requirements

Sandia National Laboratories

- Analyses often must address system reliability, safety, and security in both nominal and extreme physical environments
- Most effort in **physical** NW design is for extreme environments
  - All the more need to address this issue when it's coupled to a vast digital state space – yielding a cyber-physical system
- Formal methods and complexity theory can help address the design space of cyber-physical information systems, verifying requirements infeasible to cover by testing/simulation
  - Example coming up: Modeling the digital response of electronics to radiation-induced upsets



Need to verify increasingly complex behavioral requirements



Certain devices inherently lack the complexity needed for certain tasks

- e.g., a relay cannot implement a power grid

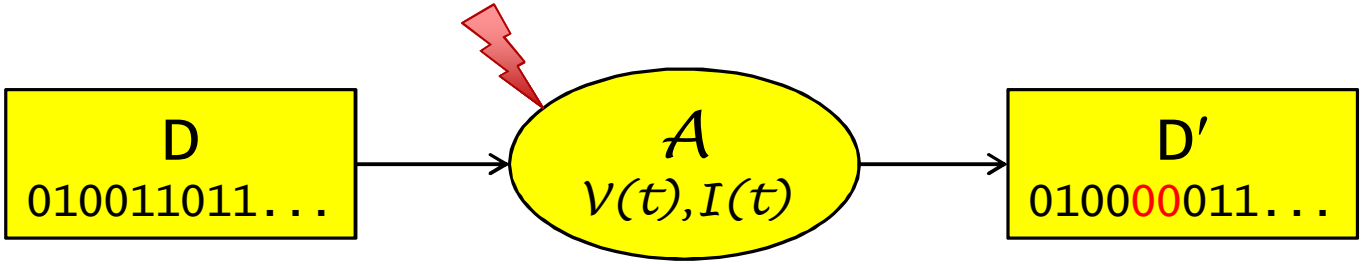
A device needs only a given amount of design complexity to implement a set of requirements

Logarithmic scale of:

- System requirements (y-axis)
- Device/system used to implement requirements (x-axis)

Analysis scalability is enhanced by coupling analog and digital models

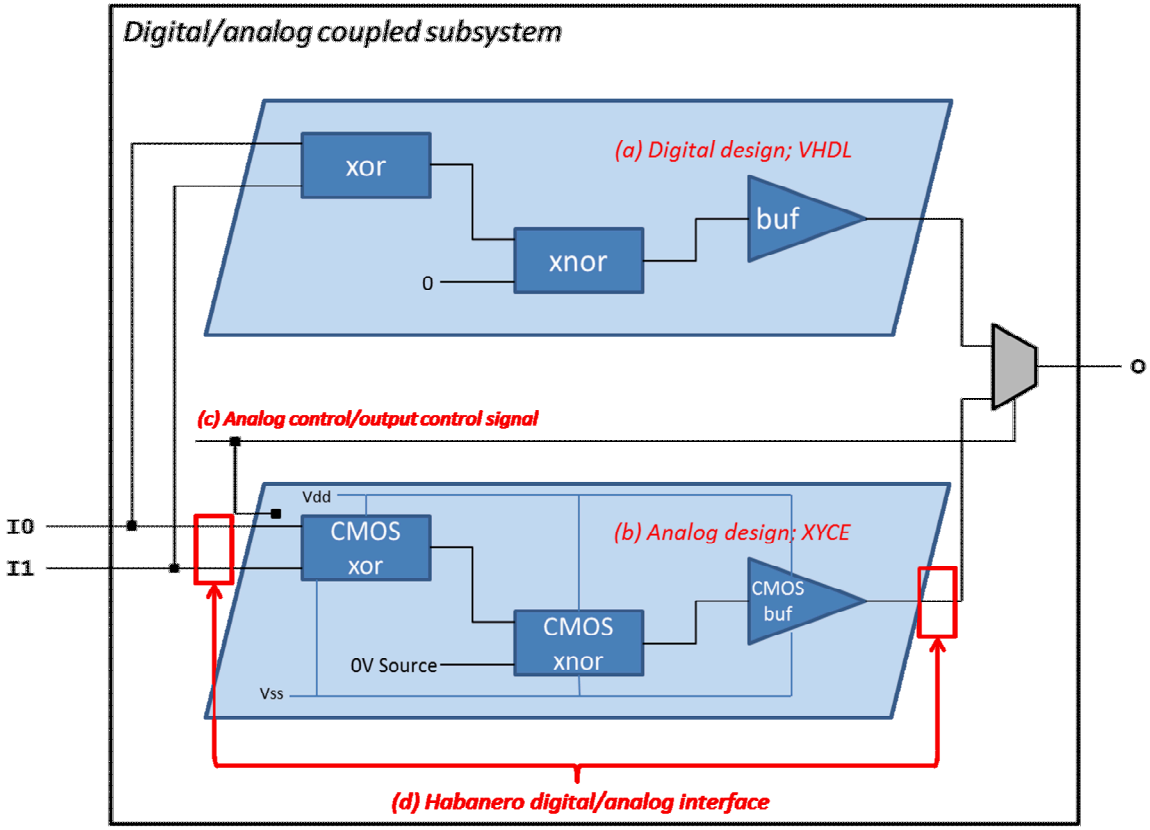
- Goal: Understand the net effect of an out-of-nominal event as an abnormal transition (upset) in a digital state space



- Simulate this transition – then use, e.g., a digital formal model to evaluate its consequences exhaustively
- Analog simulations of many different upsets can leverage HPC (embarrassingly parallel)

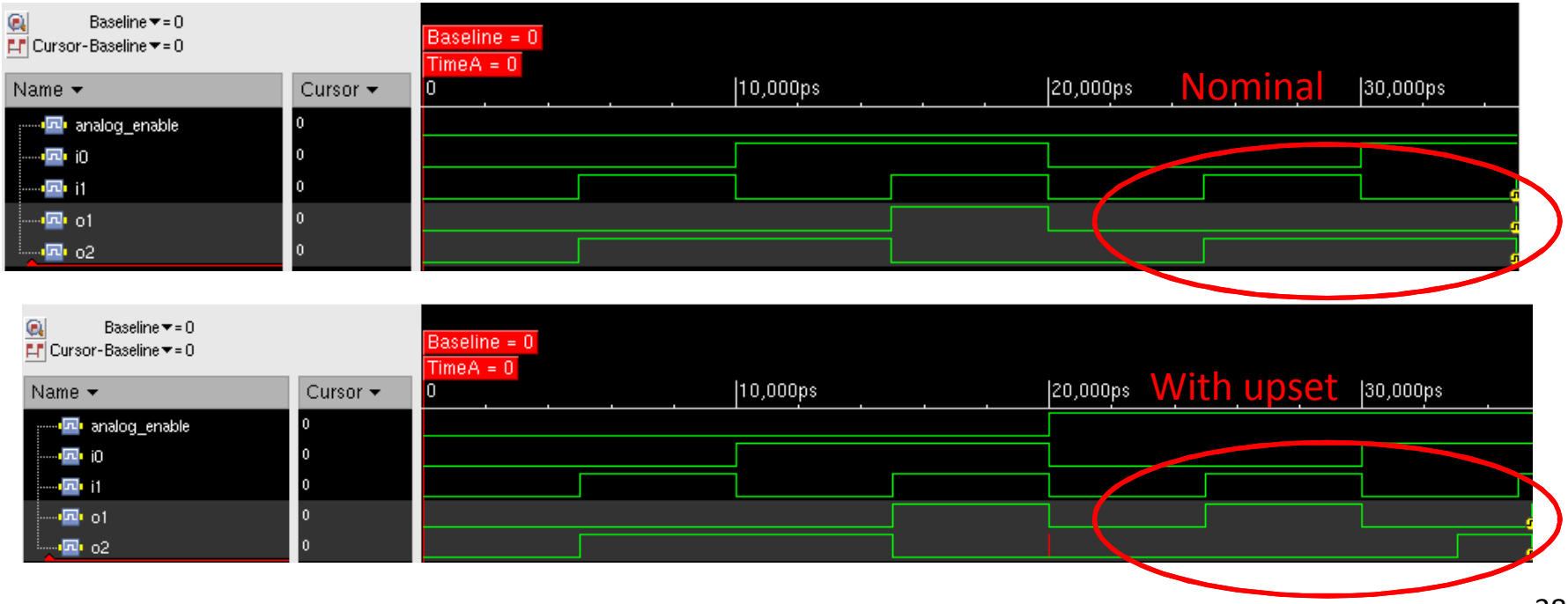
Digital upsets can be generated from physics

- Proof of concept uses a toy “half adder” circuit
- Introduce “swappable” analog model for one logic element



A toy “half adder” circuit illustrates generating digital upsets from physics

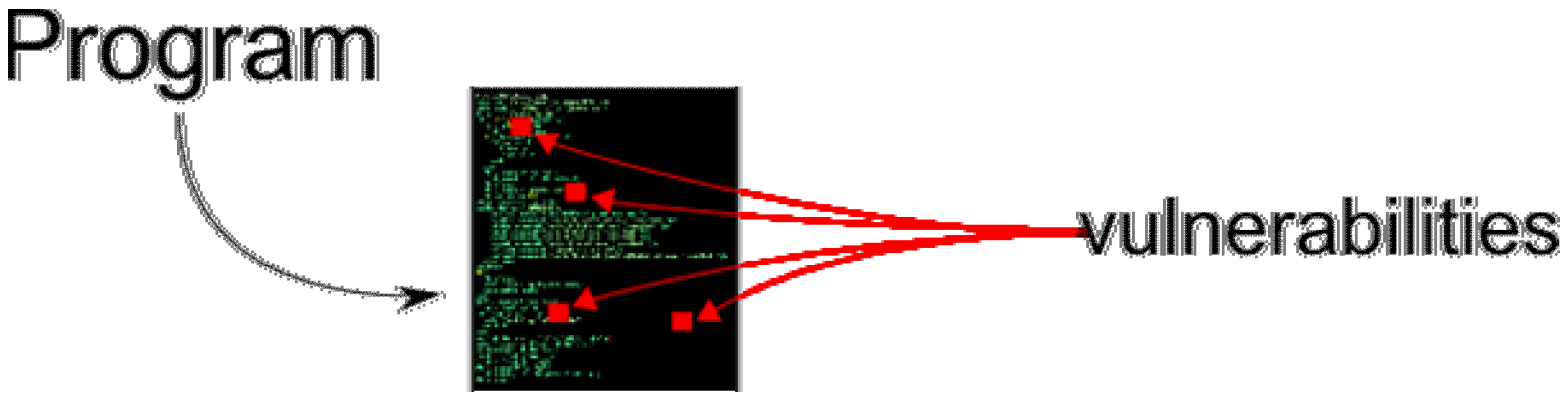
- In Habanero, simulate a single-event upset by:
  - Swapping in an analog model for a selected logic element
  - Applying a photocurrent (using an available perturbation in Xyce)
  - Re-digitizing the resulting state – output errors are seen



Securing an arbitrary code is not just hard; it's impossible

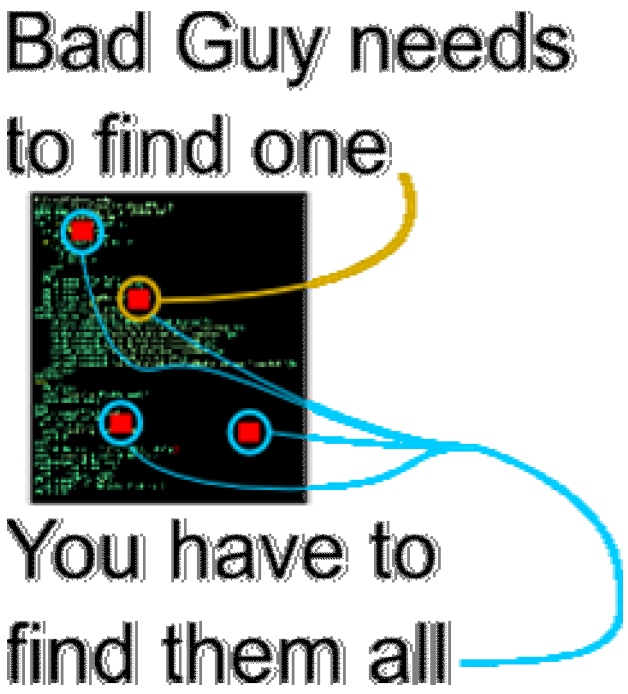
Sandia National Laboratories

- Restated: Generic code has vulnerabilities that are unprovable and unknowable
  - *Not* statistical, even in principle
  - Turing completeness demands that a generic code is undecidable



- So now what?

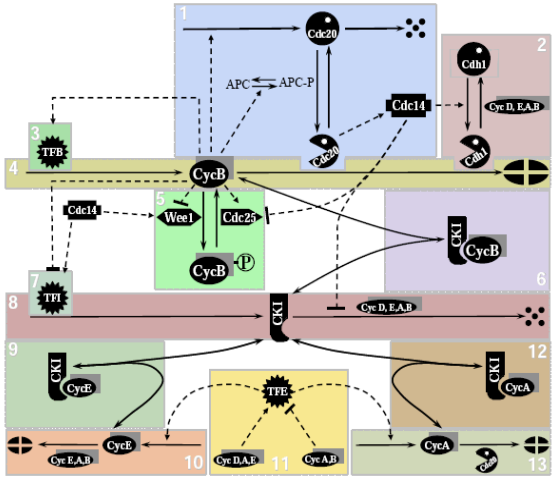
Complexity makes cyber threats *asymmetric*



- Developer, user, *and attacker* all don't know where the vulnerabilities are (*undecidable*)
- Worse, attacker may have planted a vulnerability
- Asymmetry: One vulnerability compromises the whole code
  - Developer has to find all of them (impossible in general)
- No one can guarantee "this code is clean" or even quantify improvement

Complexity is a fact of “life”

- Biological phenomena are a prototype and inspiration for many complex domains
  - Life involves a large chemical regulatory network

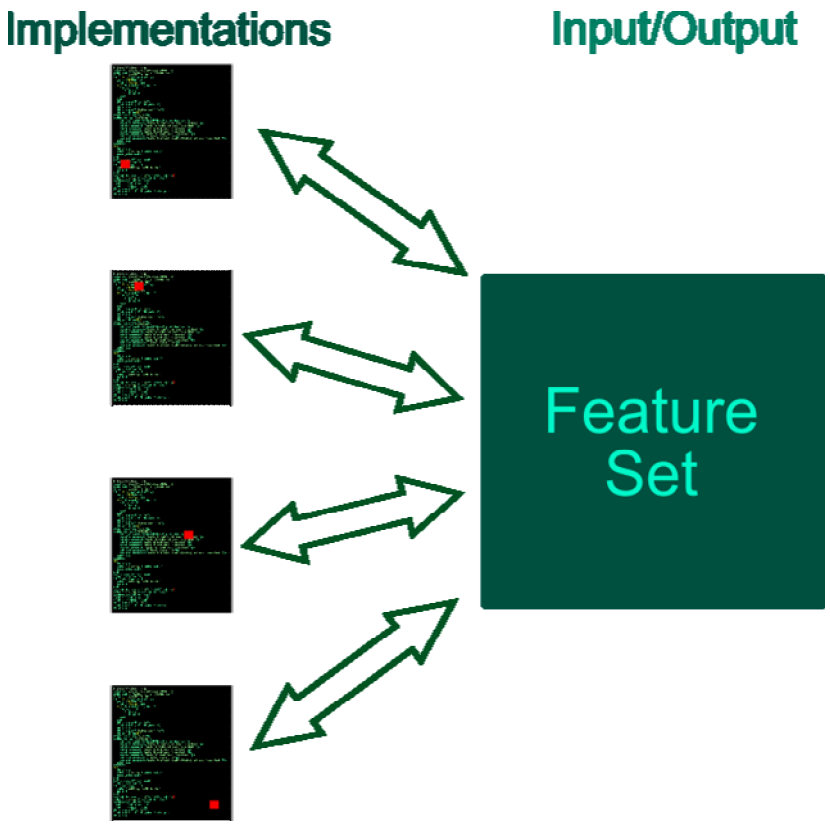


Eukaryotic cell-cycle regulation

  - “Game of Life” model is based on population dynamics
  - Bio concepts pervade computing (viruses, mutations)
- Biology typifies complex couplings of manmade systems – economy, energy, cybersecurity

Observation #1: A program's feature set has many implementations

Sandia National Laboratories

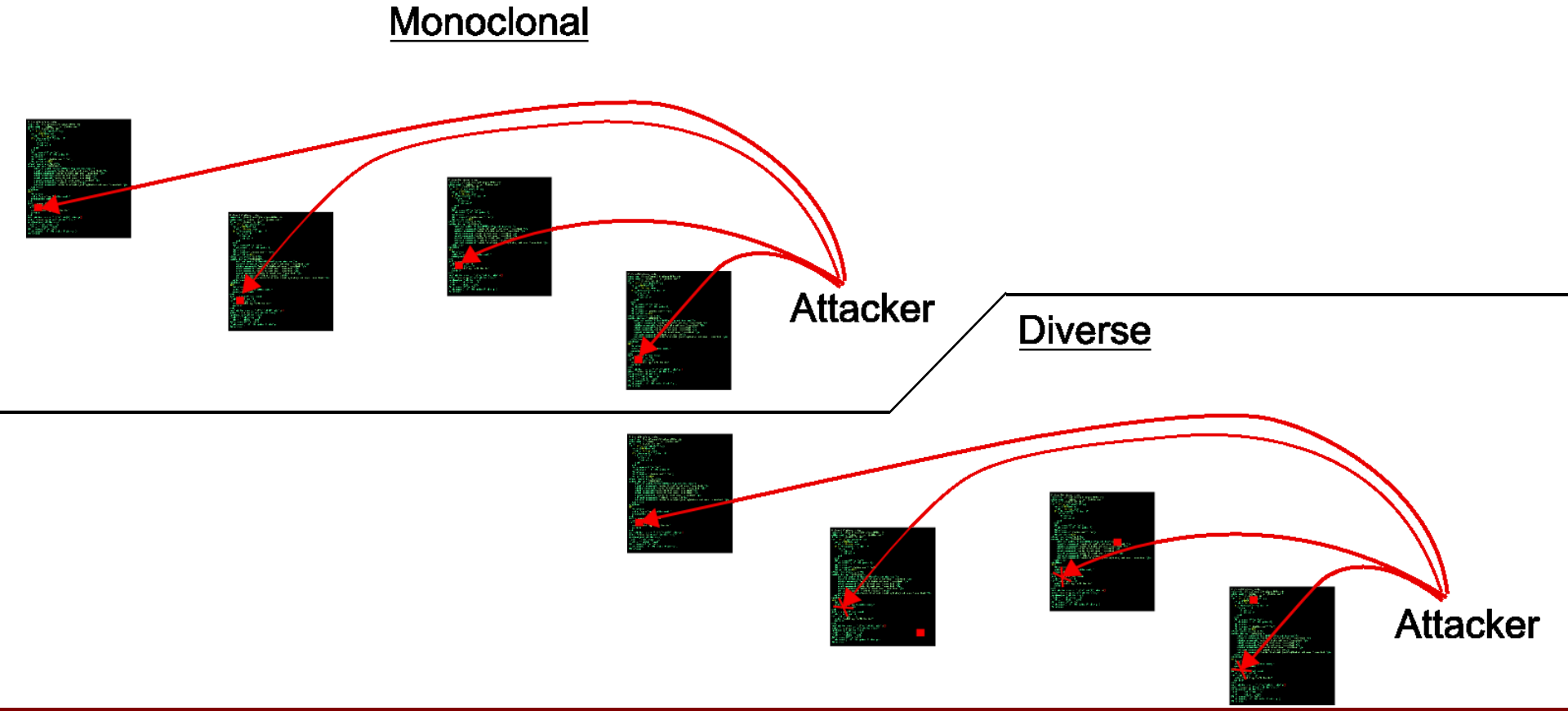


- Feature set is defined by a test suite
- Test suite verifies that an implementation conforms to desired functionality
- Test suite is a sample; cannot realistically cover all possible input/outputs
- Vulnerabilities arise from untested input/outputs
- Any feature set has infinitely many implementations
  - Finite large number if size is bounded

Observation #2: Ensemble of instances permits the formulation of statistics

Sandia National Laboratories

- Assume: Multiple implementations randomize security holes
- Ensemble of multiple-version, “randomized” undecidable codes allows formation of security *improvement* statistics



High-reliability systems can be constructed from "*N*-version software"

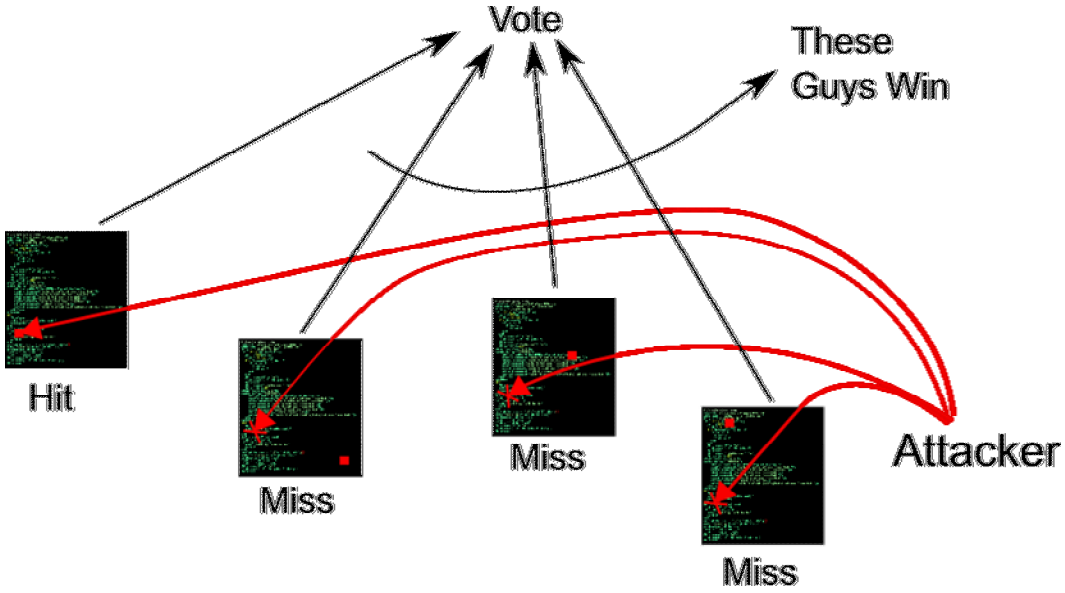
Sandia National Laboratories

- Space Shuttle: 4 computers, identical software, different hardware, same design
  - Focus is on hardware faults
- Similarly, software redundancy used mostly for control systems up to now
  - *N*-version software: Multiple versions implemented to the same feature set by different developers
- Models of *N*-version software view the control system as a stochastic process that walks the code graph of the software
  - Control system takes the place of a "fuzzer"

Similarly, *N*-version software can quantifiably improve cybersecurity

Sandia National Laboratories

- Clear generalization of *N*-version reliability to cybersecurity …

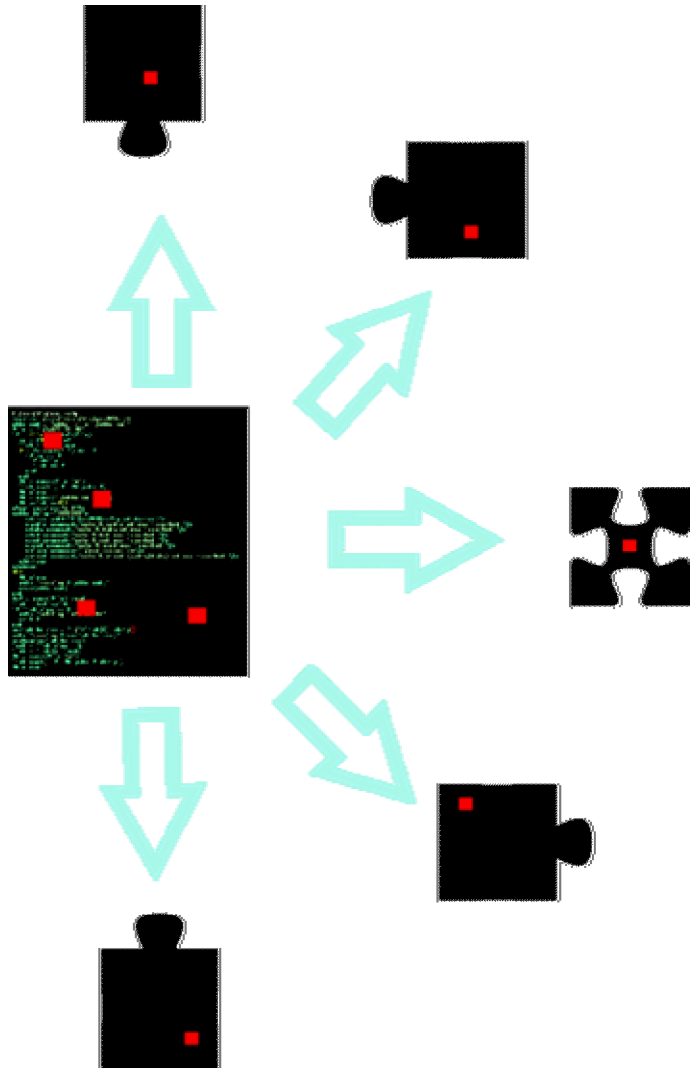


- … but there are important differences requiring enabling technology
  - Compromised versions must be removed and replaced
  - Hand-made new versions are time-consuming and expensive
    - May repeat previous mistakes

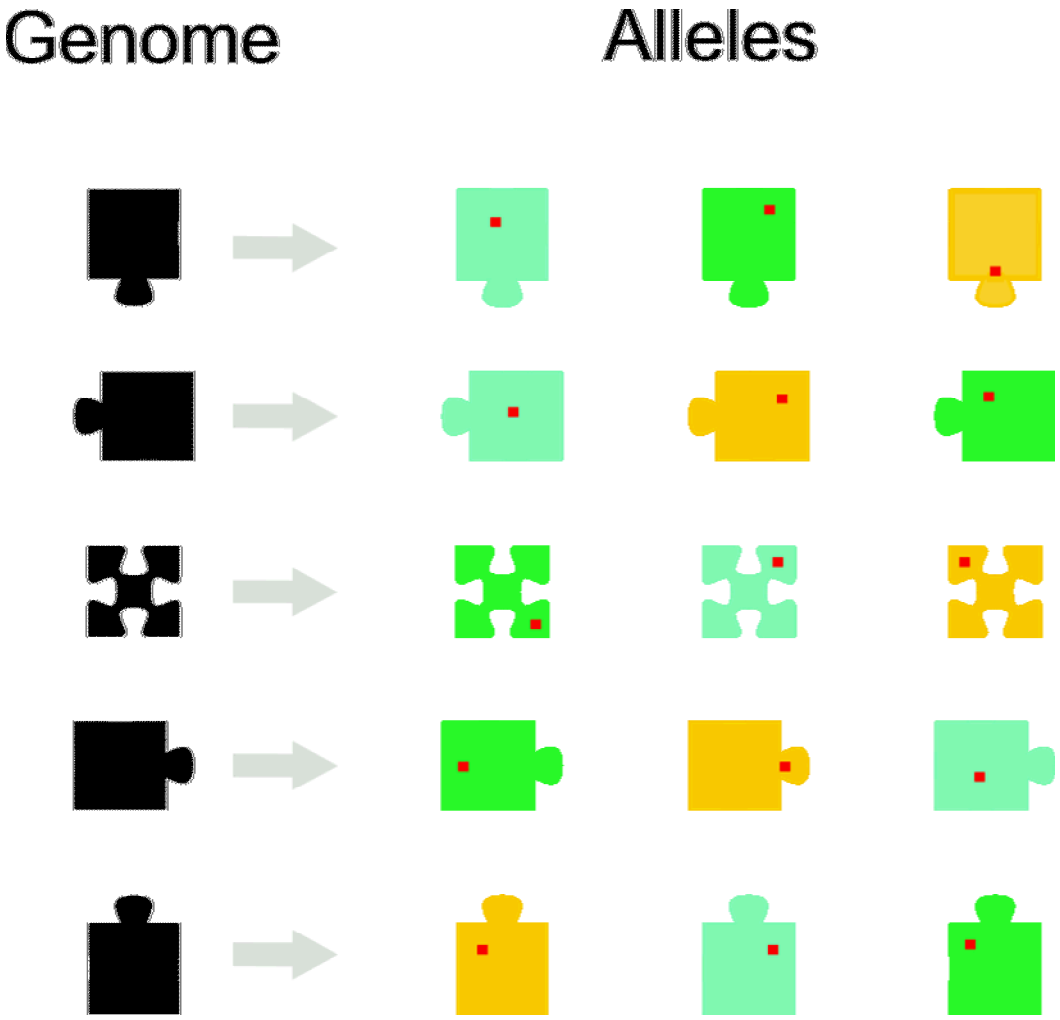
A simple example: Diverse software can be constructed from components

Sandia National Laboratories

- Component-based codes automatically conform to a feature set if the constituent components conform to their individual feature sets (semantic interfaces)
  - Multiple implementations of the code amount to multiple versions of components
  - Components can be mixed and matched to form a combinatorial number of code implementations

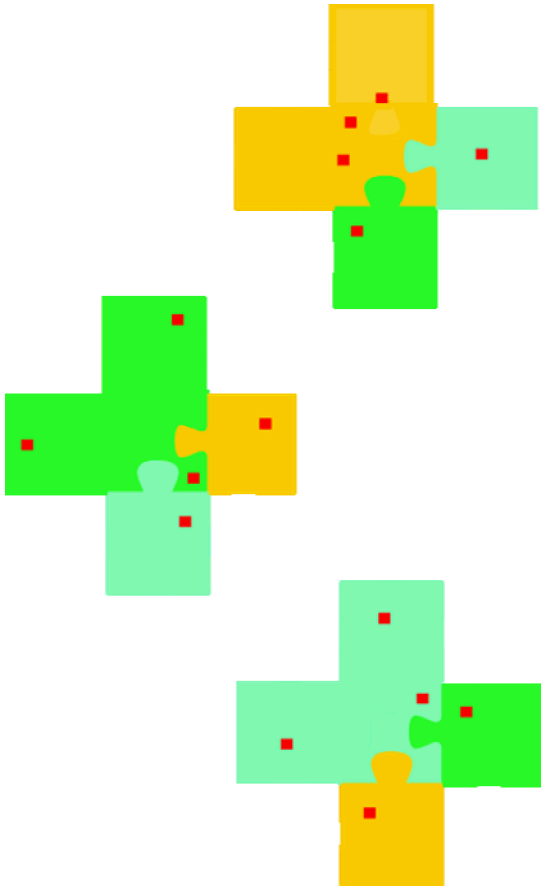


Living systems adapt to cope with unknowable attacks



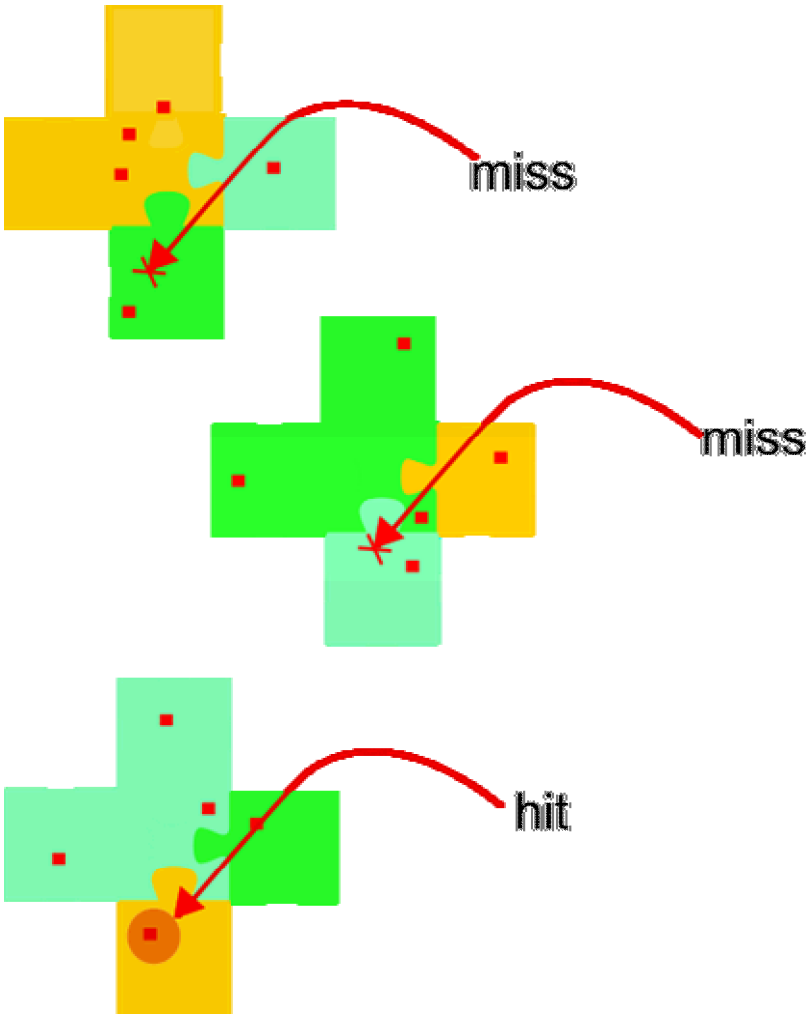
- A component type is similar to a gene; component implementations are similar to alleles of a gene

Reassemble alleles into individuals



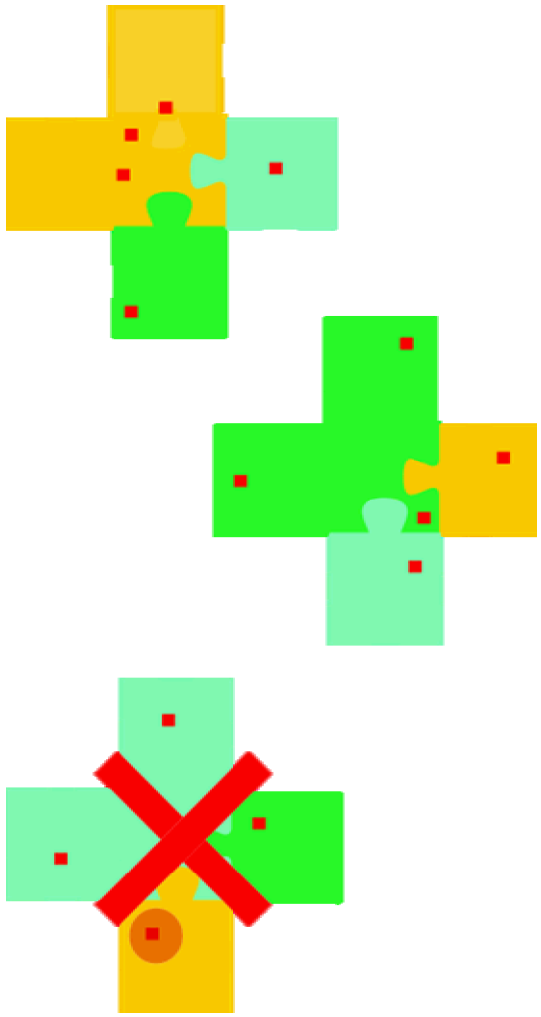
- Different alleles can be assembled into new individuals that have “randomized” security holes
- New individuals are differently vulnerable and potentially adaptive
- Excess functionality and planted vulnerabilities can be “annealed” away

Compare responses from individuals



- Now different individuals will produce the same feature set but react differently to attacks

Evolve new and more robust individuals



- Eliminate the one with the differentiated response