Correctness
via
Compilation to Logic
and
Automated Theorem Provers

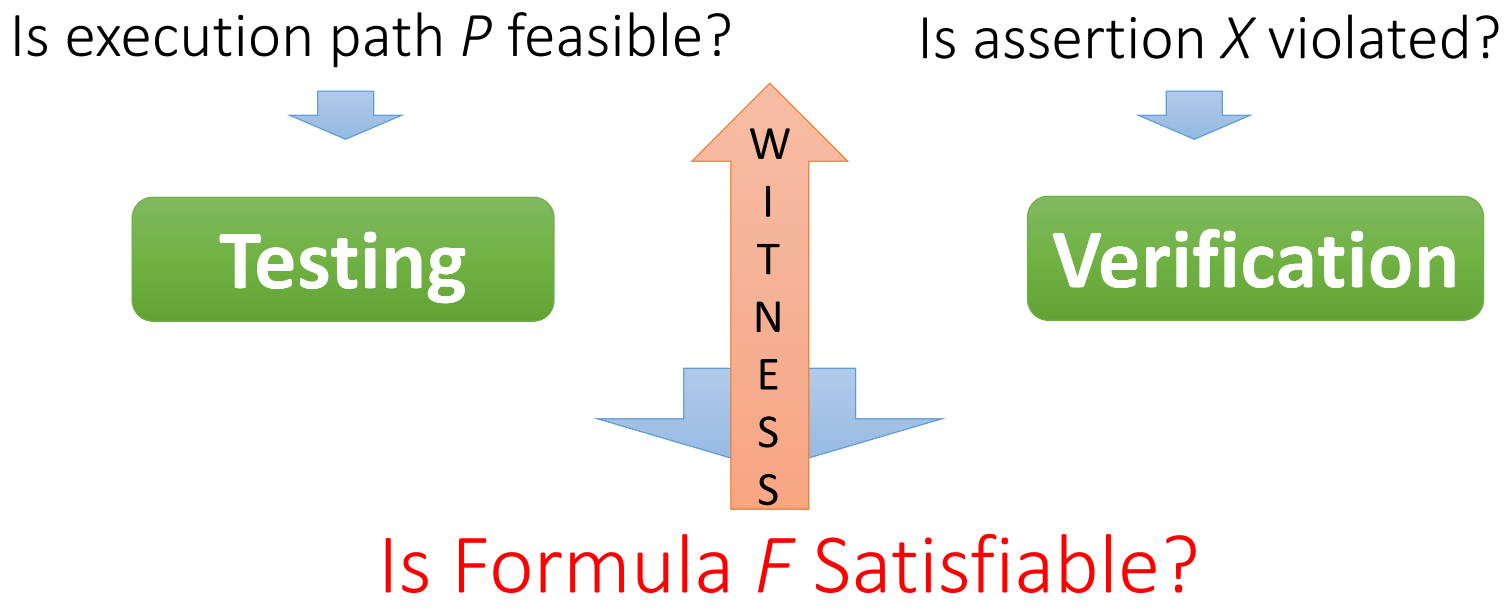
Microsoft Research

July 2014

Reduction to Logic

In theory, all problems of program correctness can be reduced to problems of logic

Reduction to Logic



Reduction to Logic

```
public static void Puzzle(int x)
{
    int res = x;
    res = res + (res << 10);
    res = res ^ (res >> 6);
    if (x > 0 && res == x + 1)
        throw new Exception("bug");
}
```

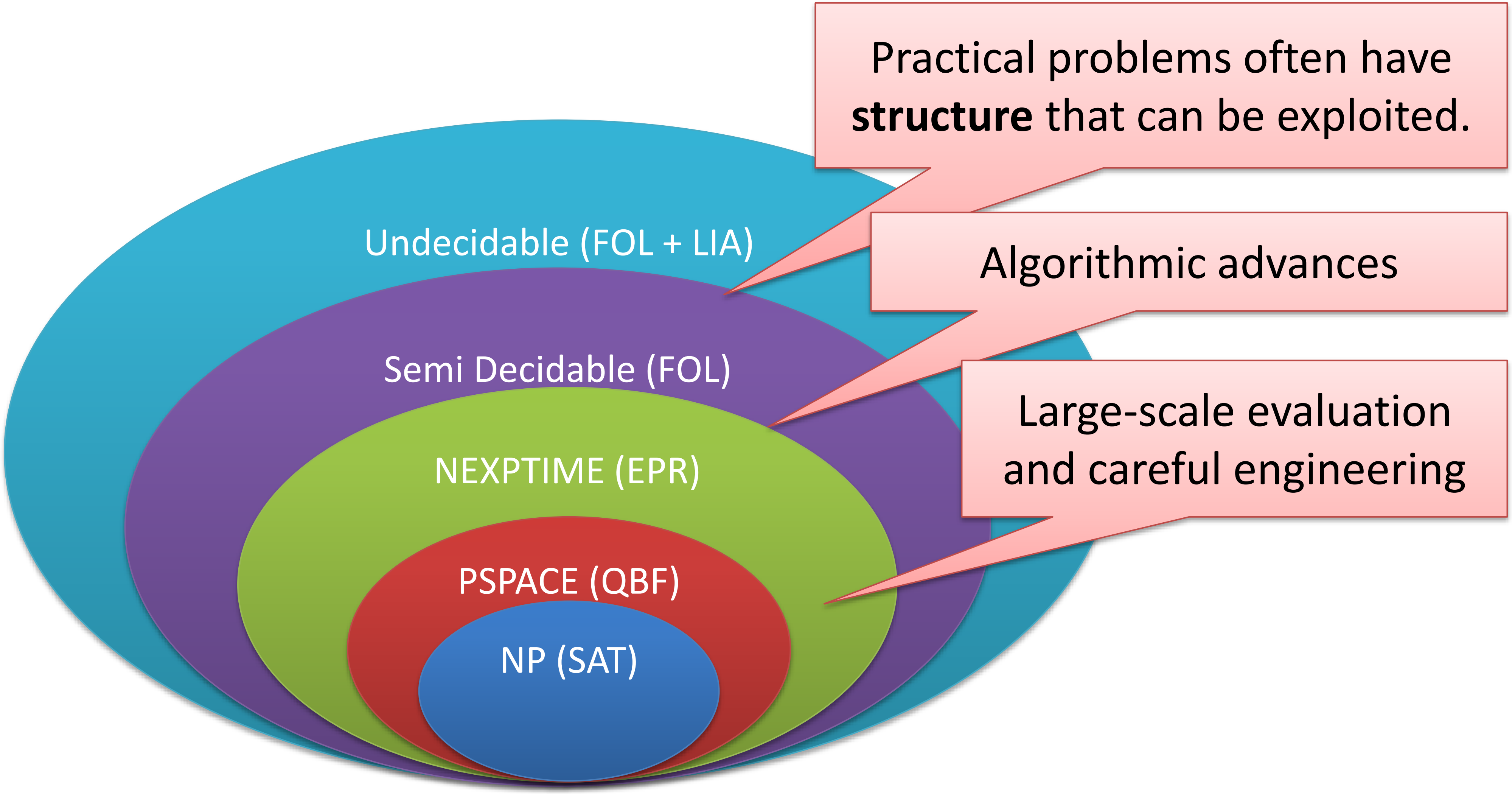
```
(declare-const x (_ BitVec 32))
(assert (bvsgt x #x00000000))
(assert (= (bvadd x #x00000001)
  (bvxor (bvadd x (bvshl x #x0000000A))
         (bvashr (bvadd x (bvshl x #x0000000A)) #x00000006))))
(check-sat)
(get-model)
```

```
sat
(model
  (define-fun x () (_ BitVec 32)
    #x1734586a)
)
```

WITNESS

http://rise4fun.com/Z3/j1BB

Logic/Complexity Classes



Reduction To Logic, Take 2

In theory, all problems of program correctness can be reduced to problems of logic

In practice, many problems of program correctness

can be compiled to problems of logic

and solved by automated theorem provers

Automated Theorem Prover

http://z3.codeplex.com/
http://www.rise4fun.com/z3/tutorial/

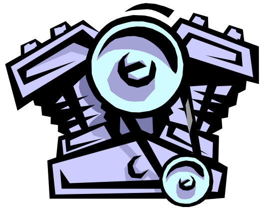
Leonardo de Moura, Nikolaj Bjorner,
Christoph Wintersteiger

Boolean Algebra

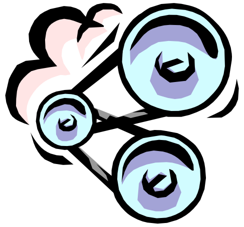
Bit Vectors

Linear Arithmetic

Floating Point



Z3 reasons over
a combination of theories



First-order Axiomitizations

Non-linear, Reals

Sets/Maps/…

Algebraic Data Types

Results and Contributions

Algorithms

Decidable Fragments

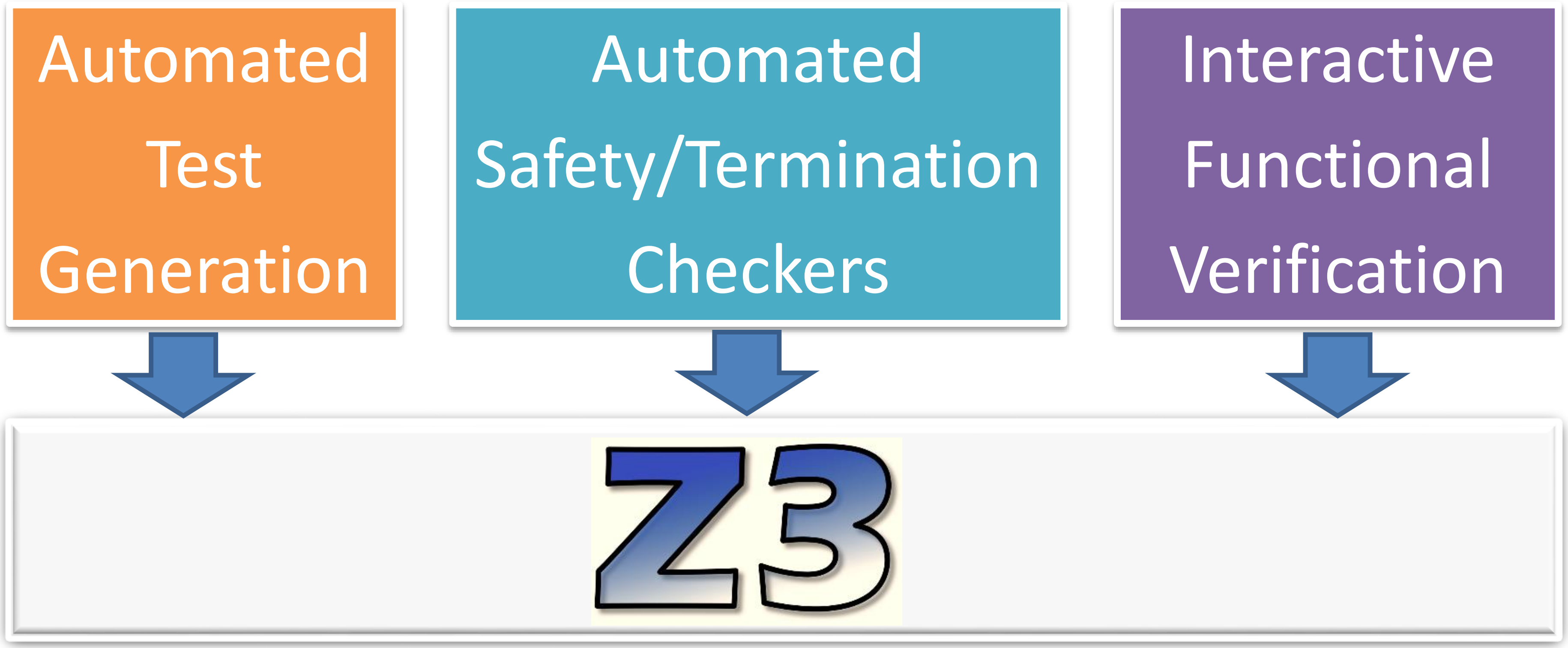
Data structures

Heuristics

Won 19/21 divisions in SMT 2011 Competition

The most influential tool paper in the first 20 years of TACAS (2014)

Program Correctness via Compilation to Logic



Automated Test Generation

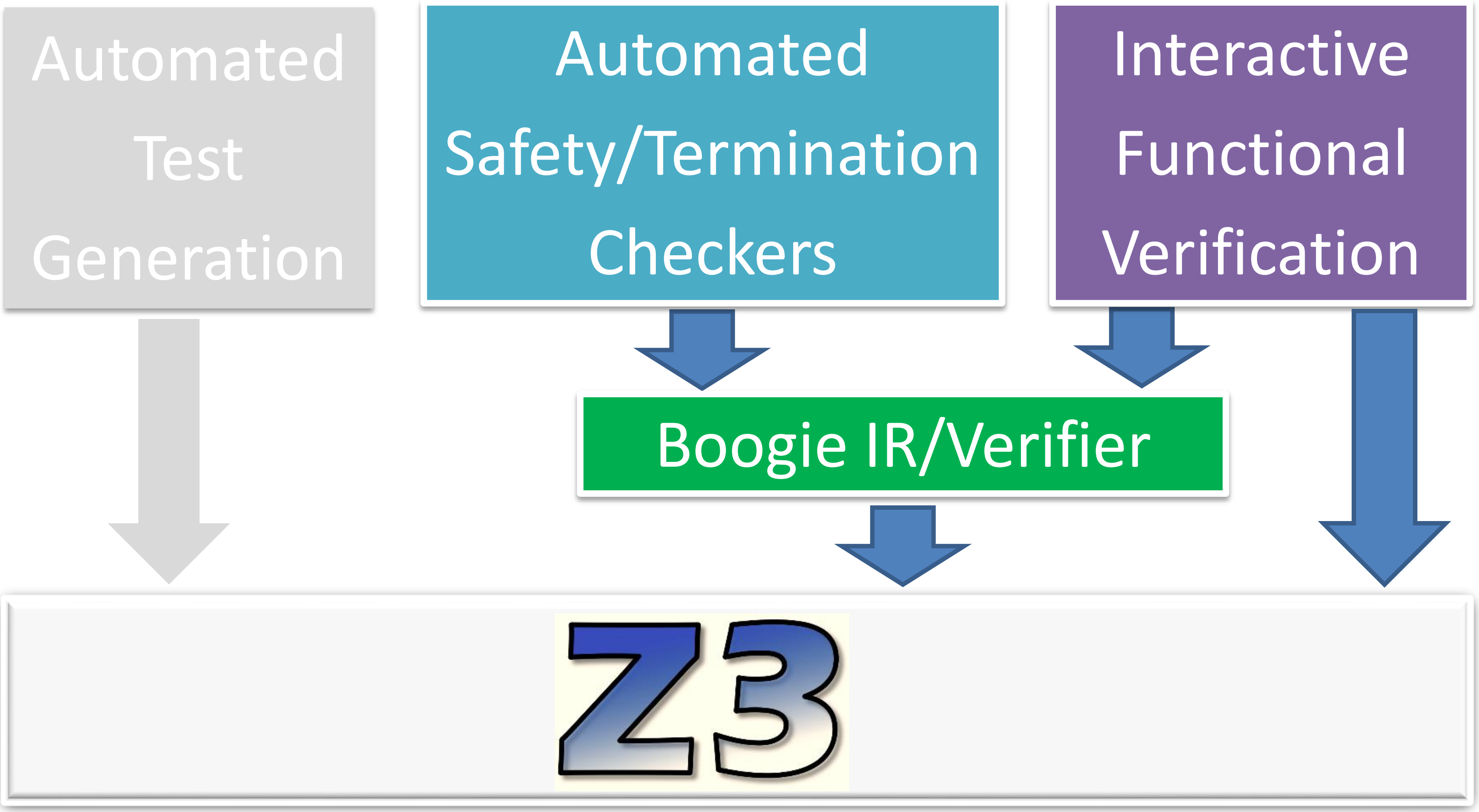
SAGE: Binary File Fuzzing

- Symbolic execution of x86 traces
- 1/3 of all file fuzzing security bugs during Windows 7
- Z3 theories: primarily bit vectors, also arrays
- Patrice Godefroid, David Molnar, Ella Bounimova

Pex: Parameterized Unit Testing

- Symbolic execution of .NET traces
- Many external users via VS; powers www.codehunt.com
- Z3 theories: automata, bit vectors, maps, algebraic data types, …
- Nikolai Tillman, Peli de Halleux

Program Correctness via Compilation to Logic



Automated Safety/Termination Checkers

SymDiff

- Modular comparison of procedures for behavioral differences
- Used to test many versions of .NET JIT
- Implemented at **Boogie** level, supports C/C++, x86, …
- Shuvendu Lahiri, Chris Hawblitzel

Corral

- Whole program analysis engine based on stratified inlining
- Powers Static Driver Verifier
- Implemented at **Boogie** level, supports C/C++ and .NET
- Akask Lal (MSRI) , Shaz Qadeer

Interactive Functional Verification

Dafny

- Object-oriented language with specification language and verifier
- Extensive use in education and MS
- Implemented using **Boogie**
- Rustan Leino, Michal Moskal

http://www.rise4fun.com/dafny/tutorial/

F*

- ML-like functional language with powerful type system and verifier
- Certified TLS implementation
- Combines type checking with Z3
- N. Swamy, C. Fournet (MSRC), + MSR-INRIA, INRIA and IMDEA colleagues

http://www.rise4fun.com/fstar/tutorial/

Correctness via Compilation to Logic

- F*
  - Formalize programming language semantics via logic
- Z3: *The Next Generation*
  - Program analysis to logic
- Network verification
  - Eliminate datacenter misconfiguration errors

BUGS ACROSS THE BOARD!

Arguably, all are language design failures

Heartbleed OpenSSL
Internet Explorer 1776
…

- *Buffer overrun*
- *Use-after-free*

Facebook API Oauth
OWASP CSRFGuard
…

- *Dynamic type-safety violation*
- *Dynamic type-safety violation*

Coq

ACM Software Systems Award 2013
The most advanced and robust program logic in the world
Widely recognized as the gold standard for reliability in PL academic circles
Proofs of 4-color theorem, Feit-Thompson theorem, CompCert C compiler, …
But, even Coq is flawed: 2 soundness bugs in termination checker in the last 6 months!

- *A flaw in the logic lurking for the past 15+ years*

SECURING THE SPECTRUM OF PROGRAMMING LANGUAGES WITH F*

N. Swamy, C. Fournet, et al.
RiSE, PPT, MSR-INRIA

F*: a semantic framework in which to

- model,
- implement,
- and certify software

across the spectrum of programming languages

http://research.microsoft.com/fstar
http://rise4fun.com/fstar/

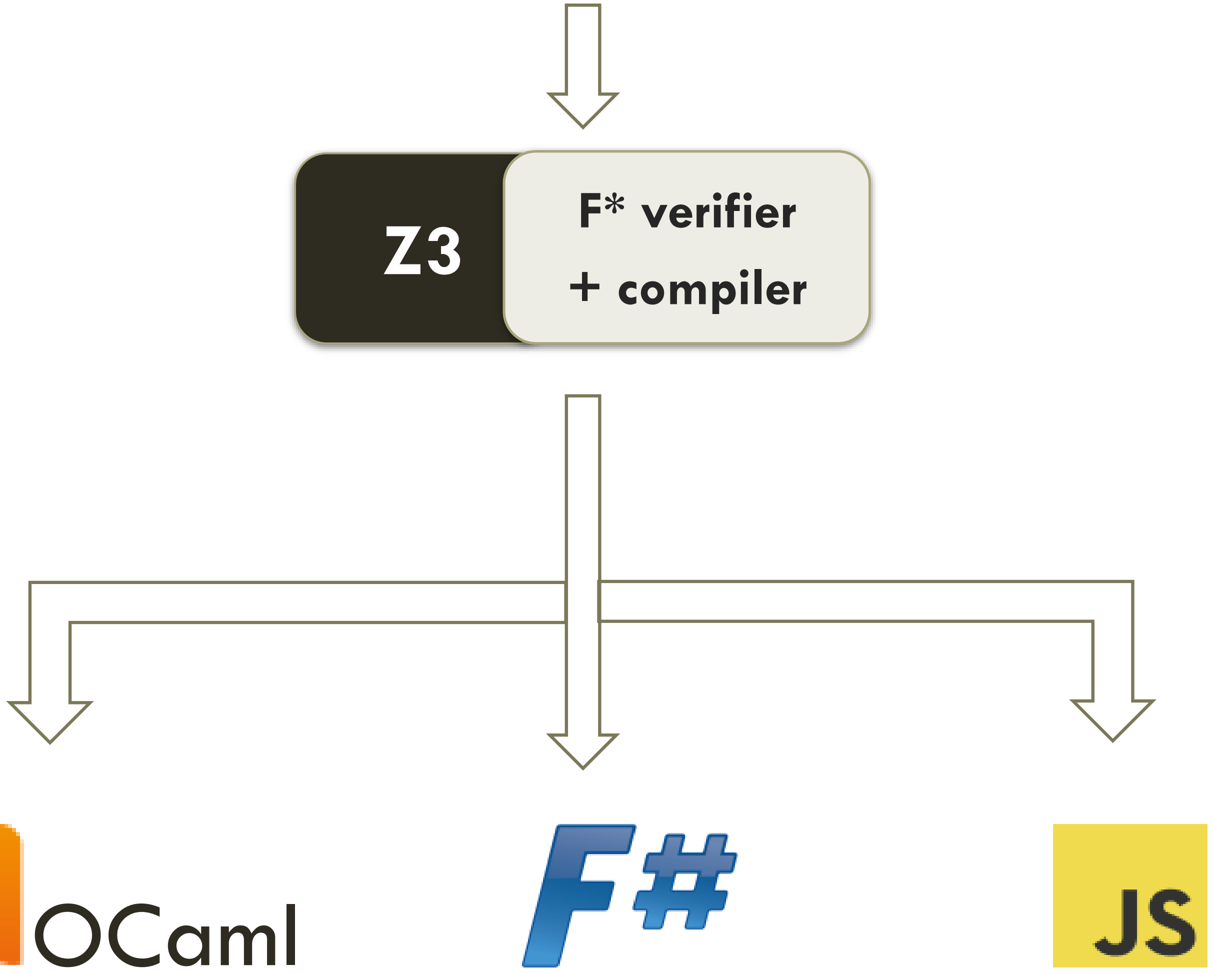
Developed collaboratively by
RiSE (Redmond), PPT (Cambridge)
and MSR-INRIA since 2008

F* source resembles F#, but with richer specifications via types

```
val counter: unit -> Writer x:int{x >= 0}
let counter = let c = ref 0 in
              fun () -> c := !c + 1; !c
```

Uses an SMT solver to automatically prove user-provided specifications

Multiple backends for cross-platform support

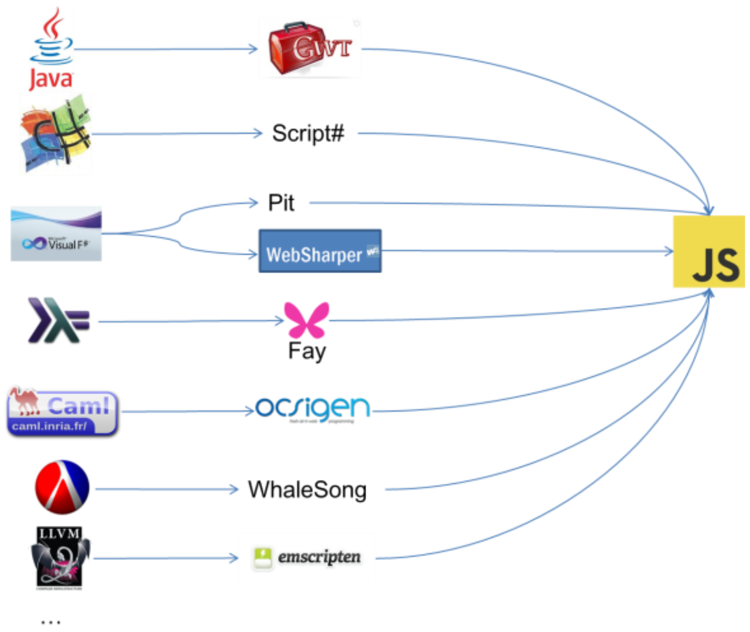


THE 1ST (SELF-)CERTIFIED PROGRAM VERIFIER

- Why trust a program verifier?
- In 2012, we programmed the F* verifier in F*, and proved it correct
- Bootstrapped the correctness proof using Coq (avoid the termination bug)
- Involved checking the largest known Coq proof
  - 8GB proof
  - verified by Coq in 24 machine-days

A PERFECTLY SECURE COMPILER FROM F* TO JS

- Increasingly, JavaScript is the target for many compilers
- But, the semantics of JS is wildly different from these languages



In 2013, **beyond JavaScript:**

- Developed a formal semantics of JavaScript within F*
- Proved a compiler from F* to JavaScript "fully abstract"
  - Full abstraction is the *semantically perfect property* for a translation

A VERIFIED IMPLEMENTATION OF TLS

miTLS-1.0:

- A full reference implementation of TLS (SSL) implemented in F7 (a subset of F*)
- A proof of its security: TLS establishes a secure channel between its endpoints
- But, performance overhead of 10x

miTLS-2.0: Currently underway

- A high-performance variant, using verified low-level memory management
  - **Beyond C++:** using F* for safe, performant low-level code
- Goal: A drop-in replacement for OpenSSL *with certified security*

FOUNDATIONAL RESEARCH IN PL SEMANTICS

Impacts theory:

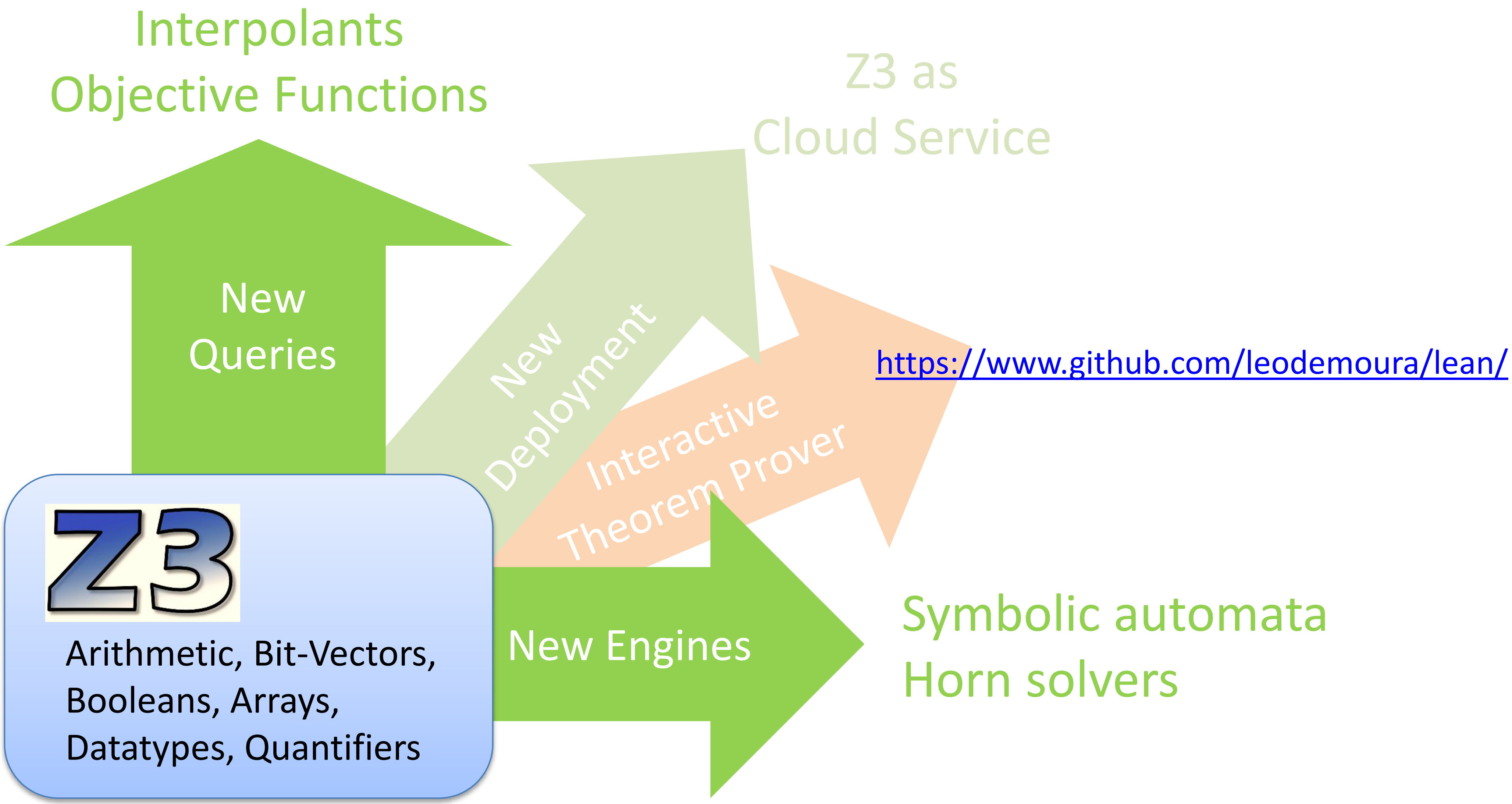
- Robust logics and tools for program verification and mathematical proofs

And practice, through certified security for

- Key elements of internet infrastructure (TLS) and
- Web-programming (JavaScript)

Z3: The Next Generation

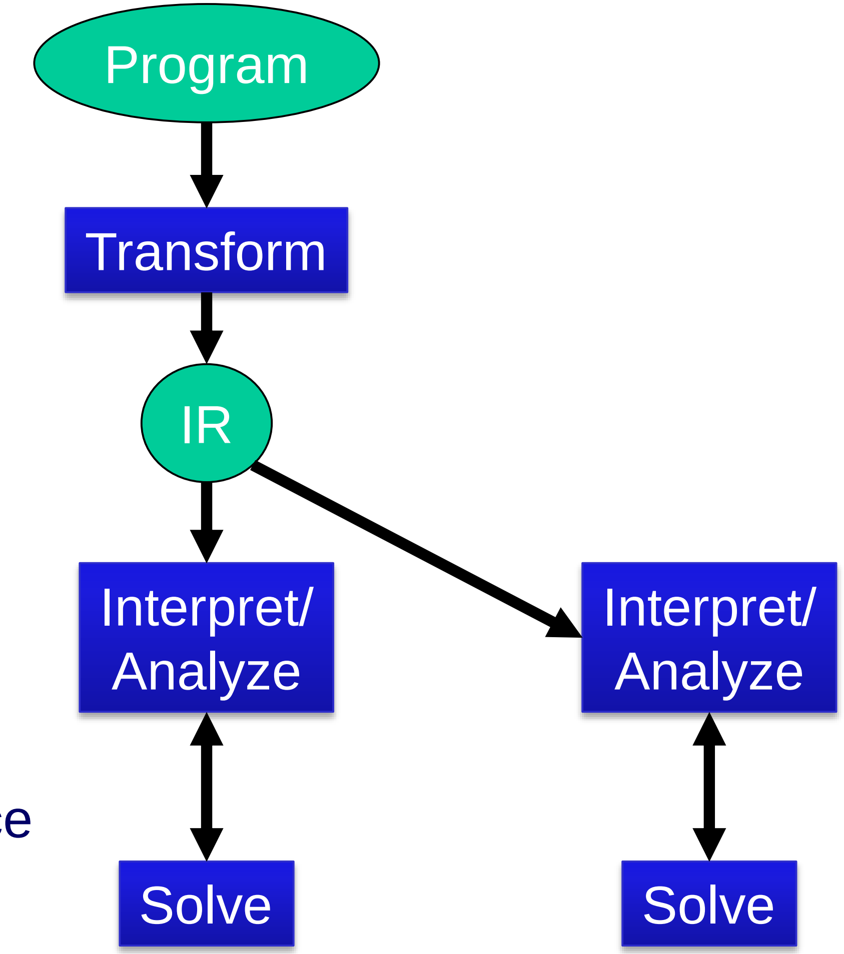
Leonardo de Moura, Nikolaj Bjorner,
Christoph Wintersteiger,
Ken McMillan, Margus Veanes,
Andrey Rybalchenko



Horn Clause
Satisfiability Modulo Theories
(Program Analysis to Logic)

Ken McMillan, Nikolaj Bjørner, Andrey Rybalchenko

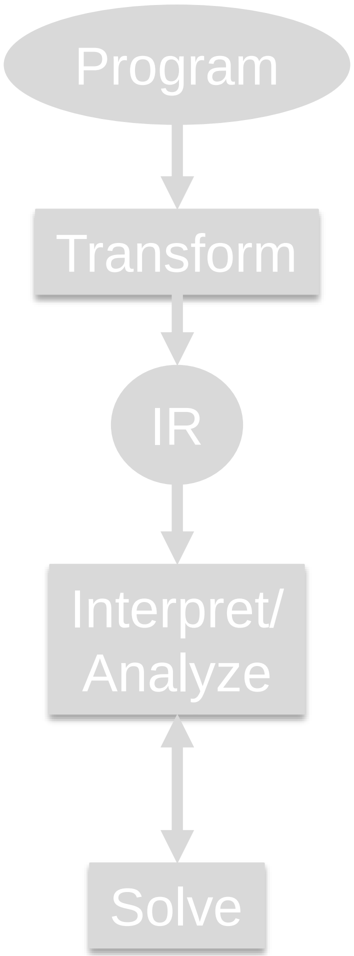
Program Analysis Architecture

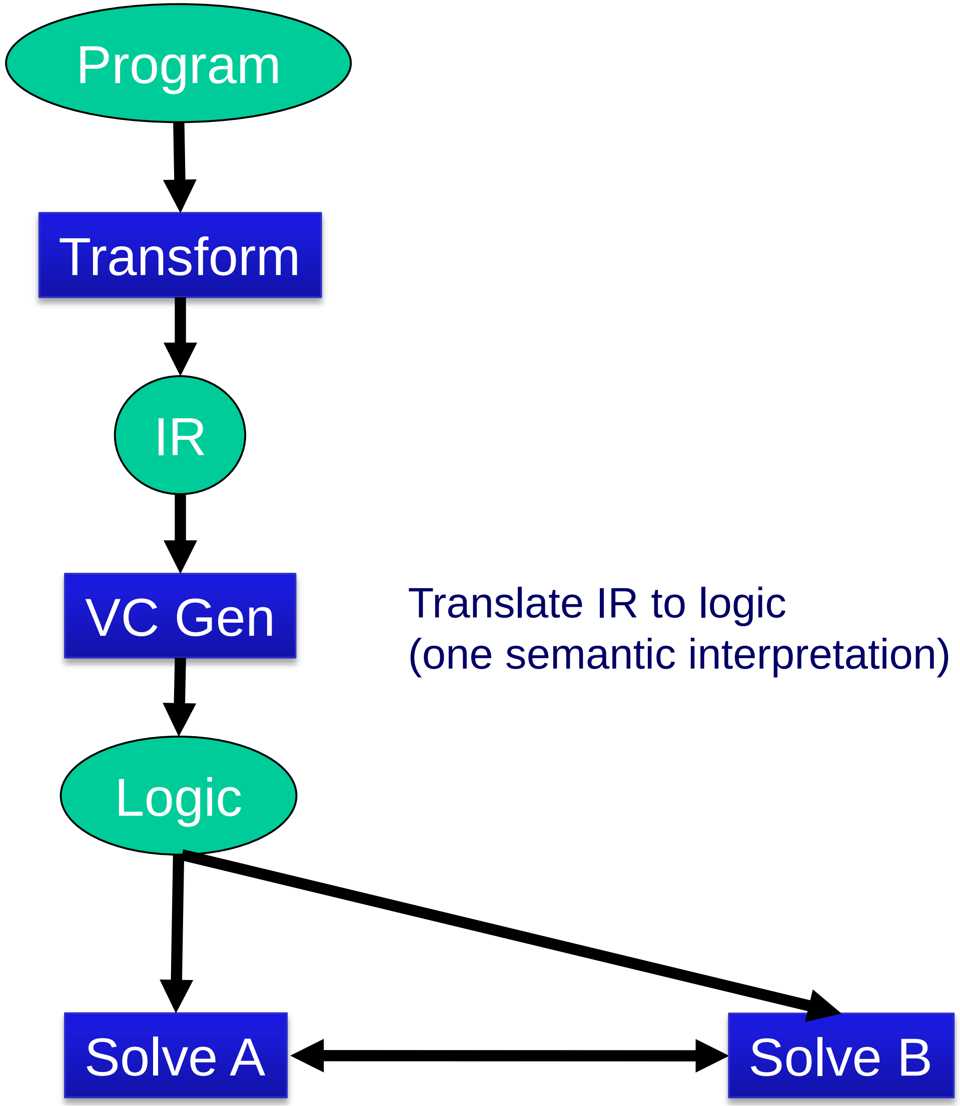


Many queries
Inefficient interface

Different analyzers interpret
IR differently

The Logic Alternative





Translate IR to logic
(one semantic interpretation)

Easy to share information

Program Analysis as Higher-order Verification

Verification

- Requires decorating a program with auxiliary assertions, such as
  - Loop invariants
  - Procedure summaries
- VC generation of decorated program yields
  - first-order logic formulae

- Leaving the auxiliary assertions as unknowns in VC generation yields
  - second-order logic formulae
- Automatically discovering sufficient auxiliary assertions is the problem of

Program Analysis

.

Example

var $i : int$:= 0;
while $i < N$ invariant $R(i)$ do
  $i := i + 2$;
done
assert $i \neq 41$;

unknown (symbolic) assertion

- The verification conditions are:

$(i = 0) \Rightarrow R(i)$ invariant holds on entry

$R(i) \wedge (i < N) \Rightarrow R(i+2)$ loop preserves invariant

$R(i) \wedge \neg(i < N) \Rightarrow (i \neq 41)$ assertion holds on loop exit

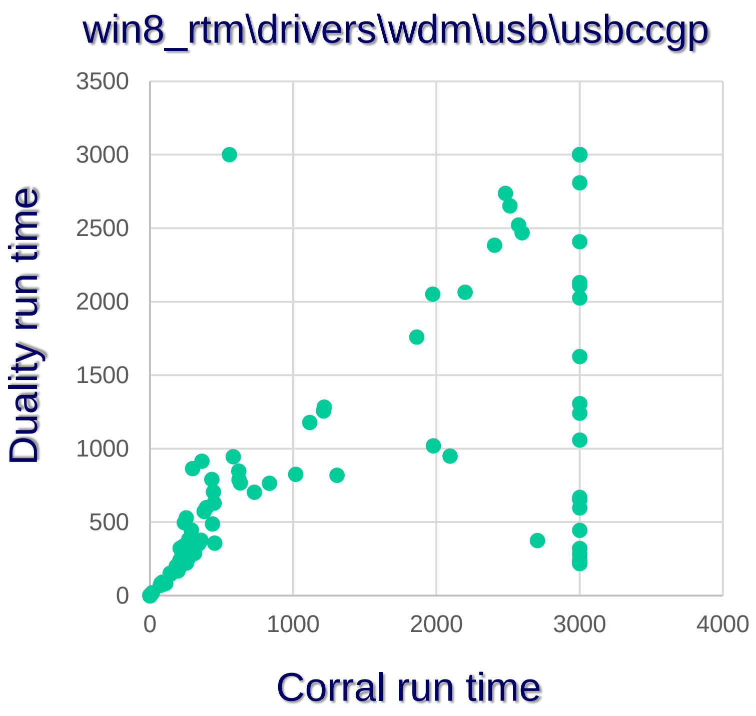
- To analyze the program, we *solve* for R:

$$R(i) \Leftrightarrow (i \bmod 2) = 0$$

Horn Solvers

- Apply to various program analysis scenarios
  - Interprocedural analysis
  - Concurrent programs
- Use standard logics and formats
  - Many applications
- Now implemented in Z3 using
  - Property-driven reachability (PDR)
  - Interpolation methods (Duality)

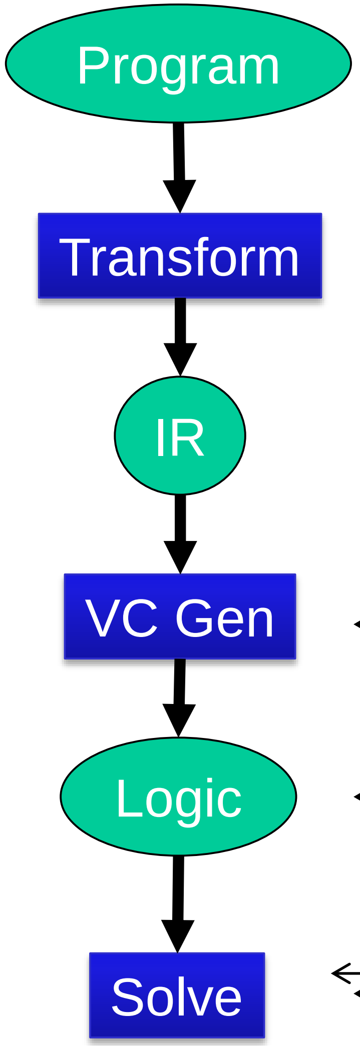
Performance



- Corral is currently the most effective analyzer in SDV, using stratified inlining and many calls to Z3
- Duality is an interpolation-based Horn solver, integrated inside Z3
- On this hard example, Duality dominates Corral in performance.

	Timeouts	Wins	Defects	Warnings	Passes
Corral	38	1	28	8	49
Duality	14	23	29	9	69

Advantages of Reduction to Logic



1) Separation of concerns
2) Simplified tools
3) Allows interoperation
4) Established standards

interpreting program semantics

SMTLIB

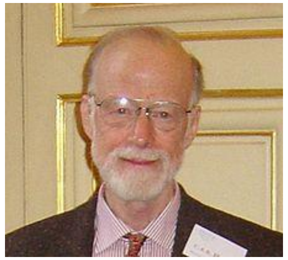
common language and model

proofs search

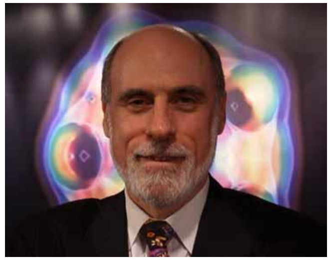
uses logic, not IR

Network Verification

PROGRAM VERIFICATION NETWORKS USEFUL **SDN** TOOLS



Tony Hoare



Vincent Cerf

Symbolic Analysis via Z3

SecGuru Validation Tool for Network ACLs

Flow Analysis Solver for Network Reachability

…

SecGuru: Validating Network Connectivity Restrictions

Windows Azure

Karthick Jayaraman, Charlie Kaufman

Microsoft Research

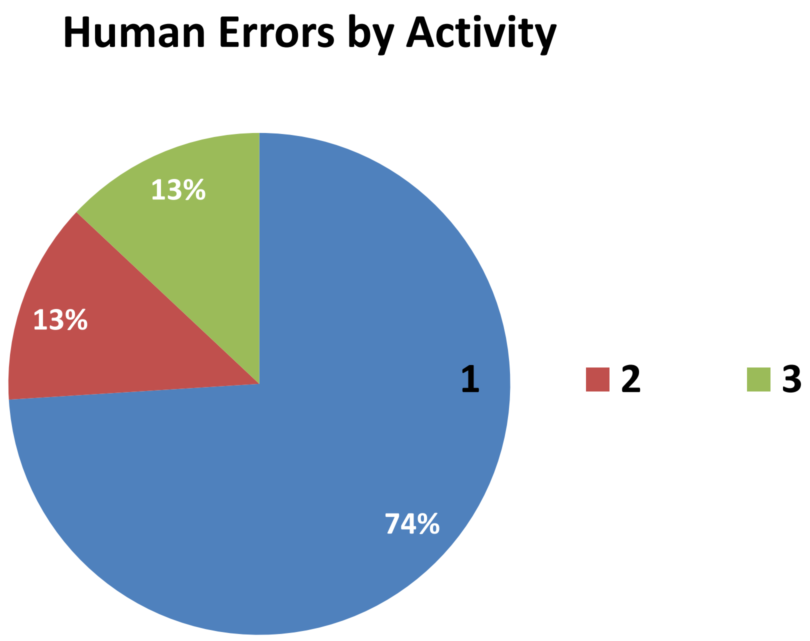
Nikolaj Bjørner

Network Policies:
Complexity, Challenge and Opportunity

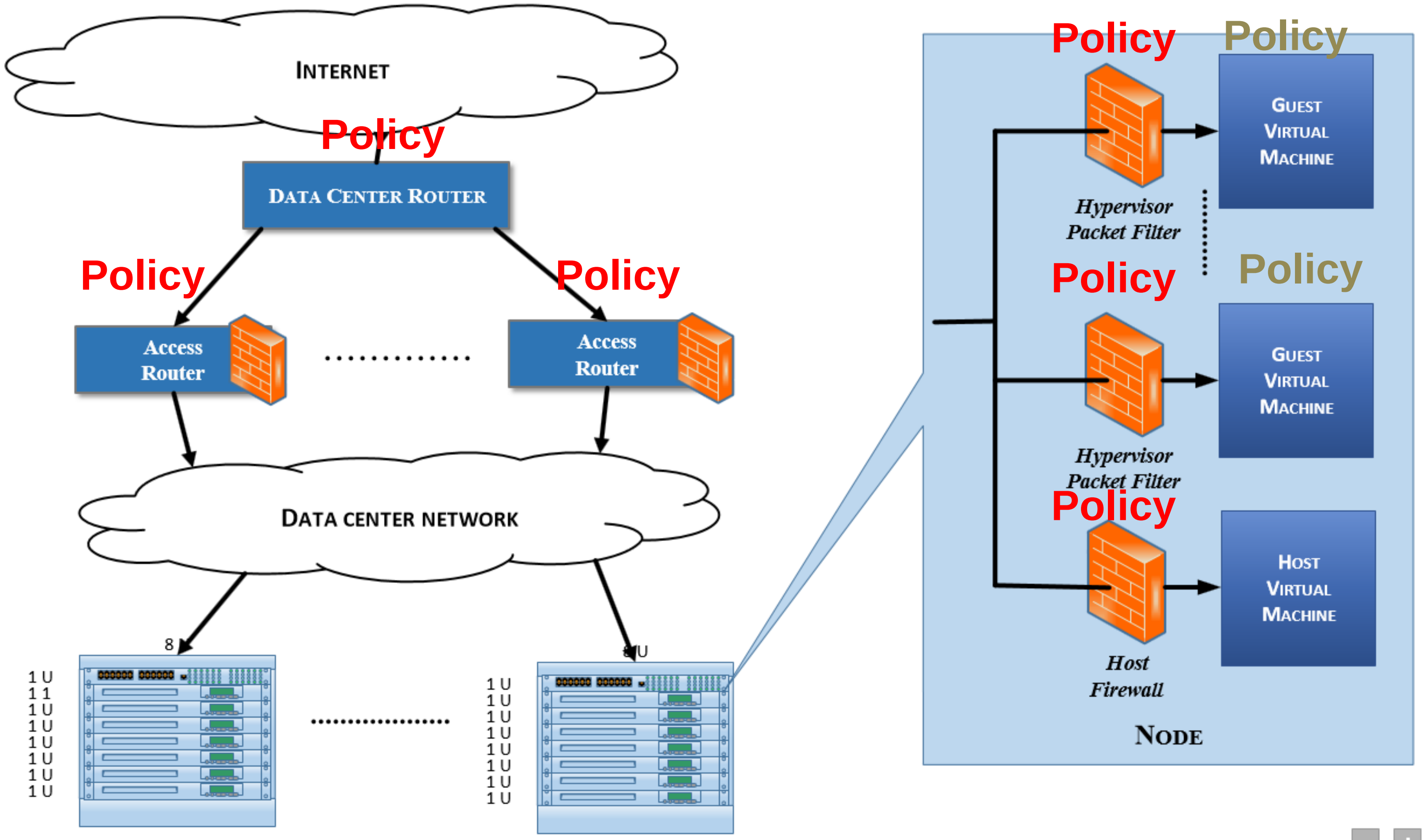
Several devices, vendors, formats
- Net filters
- Firewalls
- Routers

Challenge in the field
- Do devices enforce policy?
- Ripple effect of policy changes

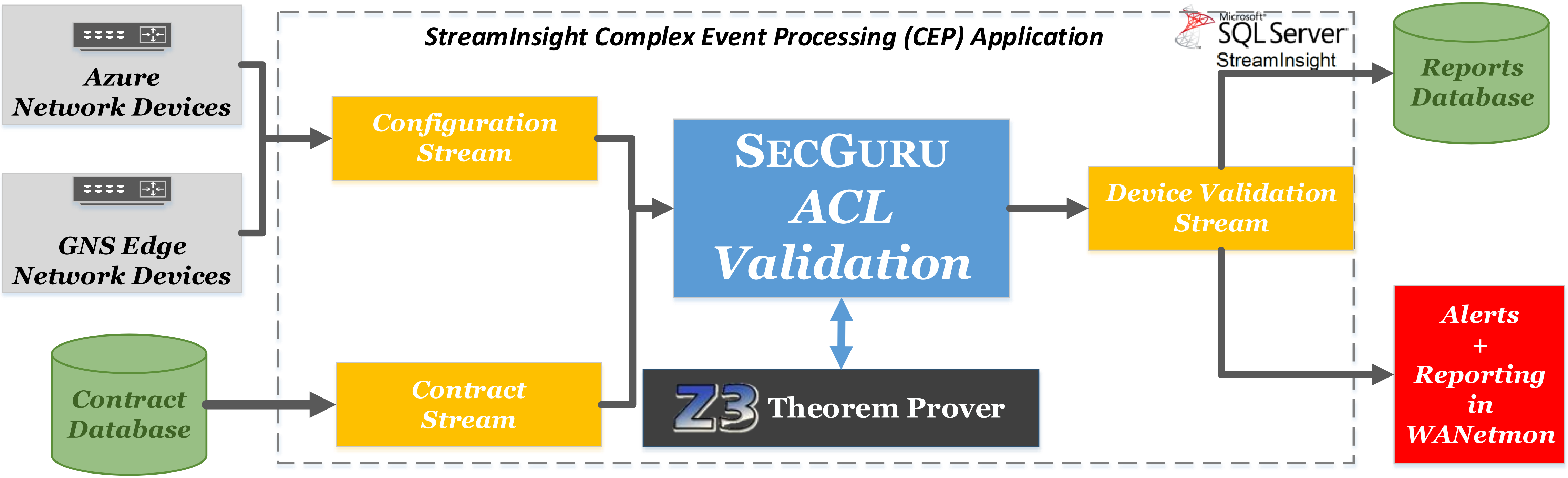
Arcane
- Low-level configuration files
- Mostly manual effort



A Data-center Architecture



SecGuru Workflow



Windows Azure Network Monitoring Infrastructure

Contracts/Policies

Status	Source Address	Src Port	Destination Address	Dst Port	Protocol
Permit	10.20.0.0/19	*Any*	157.55.252.0/30	*Any*	6
Deny	*Any*	*Any*	65.52.244.0/27	*Any*	4

Policies as Logical Formulas

Traditional Low level of Configuration network managers use

```
es: interface FastEthernet0/0------------
    e FastEthernet0/1
||  des  iption +++ LAN +++
||  ip  ddress 192.168.255.10 255.255.255.248
||  speed 10
||  full-duplex
|  !
|+ +---  3 lines: interface ATM0/0/0-------------------
|  !
|+ +---  8 lines: interface ATM0/0/0.40 point-to-point--
|  !
+  +-- 13 lines: router eigrp 301-----------------------
   !
   ip forward-protocol nd
-  ip route 0.0.0.0 0.0.0.0 192.168.255.202 250
|  ip route 81.000.00.000 255.255.255.255 87.00.00.0
|  ip route 172.16.0.0 255.255.0.0 192.168.255.11
|  ip route 192.168.255.252 255.255.255.255 ATM0/0/0.40
```

Precise Semantics as formulas

$$\mathit{Allow}: \quad \begin{array}{c} (10.20.0.0 \leq \mathit{srcIp}\ 10.20.31.255) \wedge \\ (157.55.252.0 \leq \mathit{dstIp} \leq 157.55.252.255) \wedge \\ (\mathit{protocol} = 6) \end{array}$$

$$\mathit{Deny}: \quad \begin{array}{c} (65.52.244.0 \leq \mathit{dstIp} \leq 65.52.247.255) \wedge \\ (\text{protocol} = 4) \end{array}$$

Combining semantics

$$\left(\bigvee_i \mathit{Allow}_i\right) \wedge \left(\bigwedge_j \neg \mathit{Deny}_j\right)$$

Contracts/ Policies

Z3

Semantic Diffs

Policies as Logical Formulas

Traditional Low level of Configuration network managers use

```
es: interface FastEthernet0/0-------------

|     e FastEthernet0/1
||  des  iption +++ LAN +++
||  ip  ddress 192.168.255.10 255.255.255.248
||  speed 10
||  full-duplex
|  !
|+ +---  3 lines: interface ATM0/0/0--------------------
|  !
|+ +---  8 lines: interface ATM0/0/0.40 point-to-point--
   !
+  +-- 13 lines: router eigrp 301-----------------------
   !
   ip forward-protocol nd
-  ip route 0.0.0.0 0.0.0.0 192.168.255.202 250
|  ip route 81.000.00.000 255.255.255.255 87.00.00.0
|  ip route 172.16.0.0 255.255.0.0 192.168.255.11
|  ip route 192.168.255.252 255.255.255.255 ATM0/0/0.40
```

Precise Semantics as formulas

Allow: $(10.20.0.0 \leq srcIp\ 10.20.31.255) \wedge (157.55.252.0 \leq dstIp \leq 157.55.252.255) \wedge (protocol = 6)$

Deny: $(65.52.244.0 \leq dstIp \leq 65.52.247.255) \wedge (\text{protocol} = 4)$

Combining semantics

$$\boldsymbol{Policy_0} = false$$

$$\boldsymbol{Policy_{i+1}} = \boldsymbol{if}\ IsAllow_i\ \boldsymbol{then}\ Rule_i \vee \boldsymbol{Policy_i}\ \boldsymbol{else}\ \neg Rule_i \wedge \boldsymbol{Policy_i}$$

$$\boldsymbol{Policy} \equiv \boldsymbol{Policy_{N+1}}$$

Contracts/ Policies

Z3

Semantic Diffs

Beyond Z3: a *new* idea to go from one violation to all violations

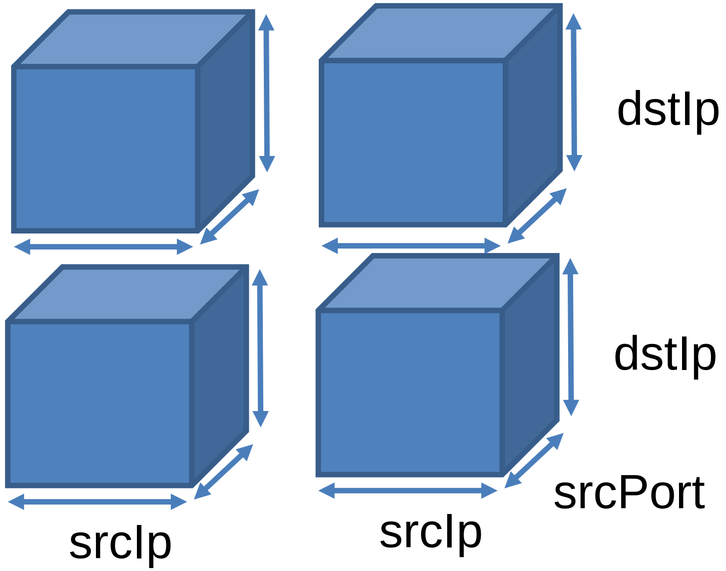
$$\left(\bigvee_i Allow_i\right) \wedge \left(\bigwedge_j \neg Deny_j\right) \longrightarrow \text{Z3} \longleftarrow \neg\left[\left(\bigvee_m Allow_m\right) \wedge \left(\bigwedge_n \neg Deny_n\right)\right]$$

Semantic Diffs

$$srcIp = 10.20.0.0/16, 10.22.0.0/16$$
$$dstIp = 157.55.252.000/24, 157.56.252.000/24$$
$$port = 80, 443$$

Representing solutions

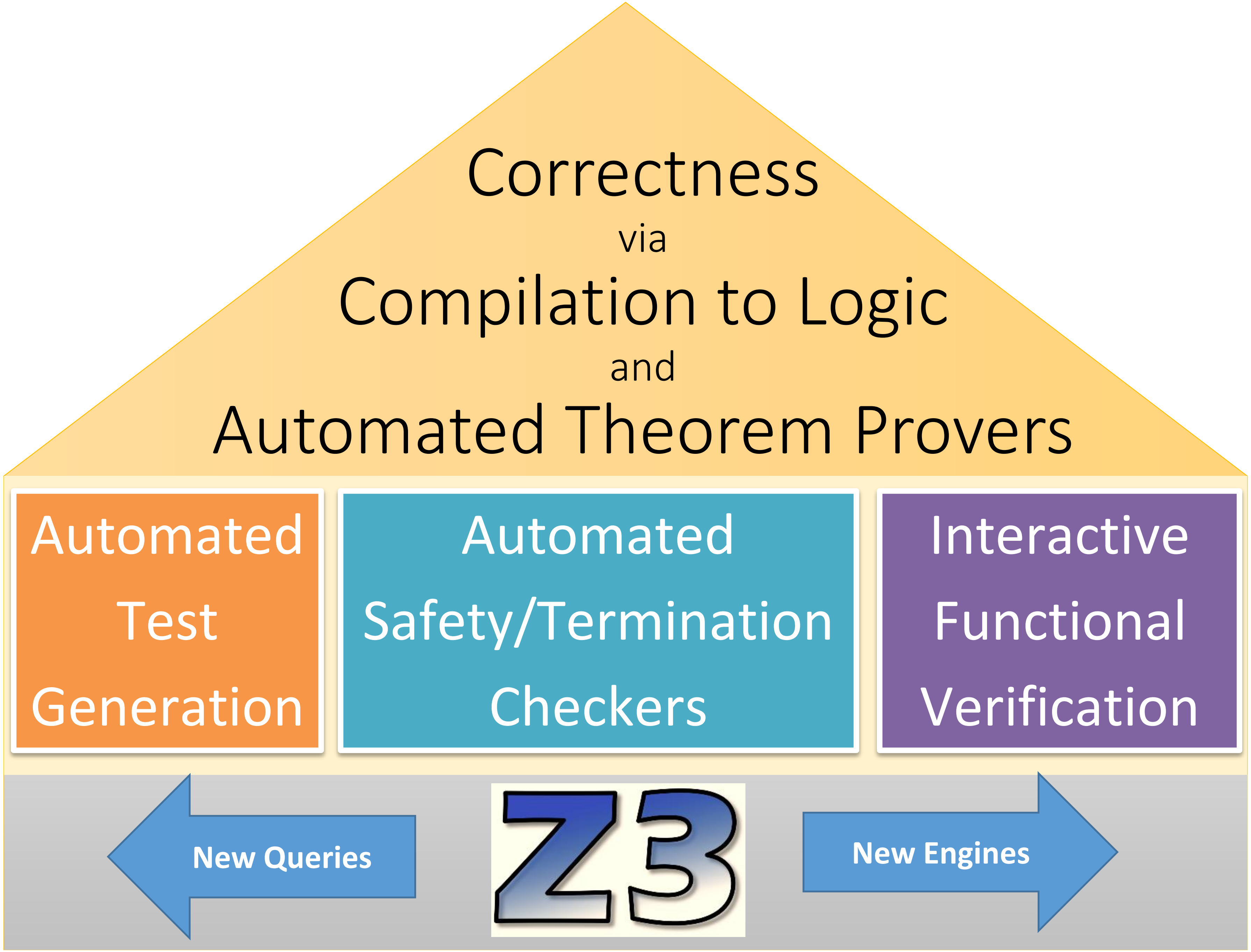
- $2 * 2^{16} * 2 * 2^{8} * 2 \;=\; 2^{27}$ single solutions, or
- 8 products of contiguous ranges, or
- A single product of ranges



SecGuru contains optimized algorithm for turning single solutions into all (product of ranges)

Other Domains Powered by Z3

Analog circuit design	Verifying global convergence for a digital phase-locked loop	FMCAD 2013	UBC
Model-based testing of OS performance	Designing scalable software for multicore processors	SOSP 2014	MIT
Program optimization	Consolidation of Queries with User-Defined Functions	PLDI 2014	UT Austin, U. Oxford, MSR
Computational biology	Analyzing and synthesizing genomic logic functions	CAV 2014	U. Oxford, MSR


Correctness
via
Compilation to Logic
and
Automated Theorem Provers
Automated Test Generation
Automated Safety/Termination Checkers
Interactive Functional Verification
New Queries
Z3
New Engines