From North Stars to Clever Insights

On using grand challenges to drive new techniques in automated theorem proving

Nikolaj Bjørner

Microsoft Research

Aim of talk

Describe a set of *applications* that use Satisfiability Modulo Theories, SMT

Describe model-based techniques, an *insight* driving SMT architecture

Satisfiability Modulo Theories (SMT)

Is formula φ satisfiable modulo theory *T* ?

SMT solvers have specialized algorithms for *T*

Satisfiability Modulo Theories (SMT)

$$\boxed{x + 2 = y} \Rightarrow \boxed{f}(\boxed{select}(\boxed{store}(a, x, \boxed{3}), \boxed{y - 2})) = \boxed{f}(\boxed{y - x + 1})$$

Array Theory

Arithmetic

Uninterpreted Functions

$$select(store(a, i, v), i) = v$$
$$i \neq j \Rightarrow select(store(a, i, v), j) = select(a, j)$$

Z3 – An Efficient SMT Solver

What it is for:

- Program analysis tools ultimately rely on solving logical constraints

"The calculus of computation"

- A need to lower barrier of entry for program analysis tools

What it is:

- General purpose theorem prover
- Specialized algorithms for important workloads
- Open Source on GitHub

Some Microsoft Uses of Z3

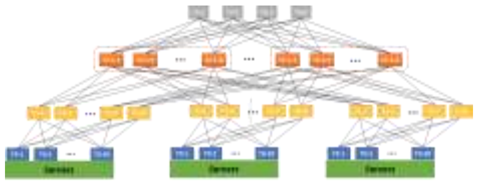
Microsoft Security Risk Detection

Solved billions of fuzzing constraints

Program paths

Bit-vector + array logic

SecGuru and FIB verifier



Online checks of $O(10^5)$ Azure routers + ACLs

Network Configurations

Bit-vector logic

Dynamics Product Configurator

Maintains design space of parameters

Production Line Configs

Enumerate Consequences

Project Everest

https://www.fstar-lang.org/tutorial/

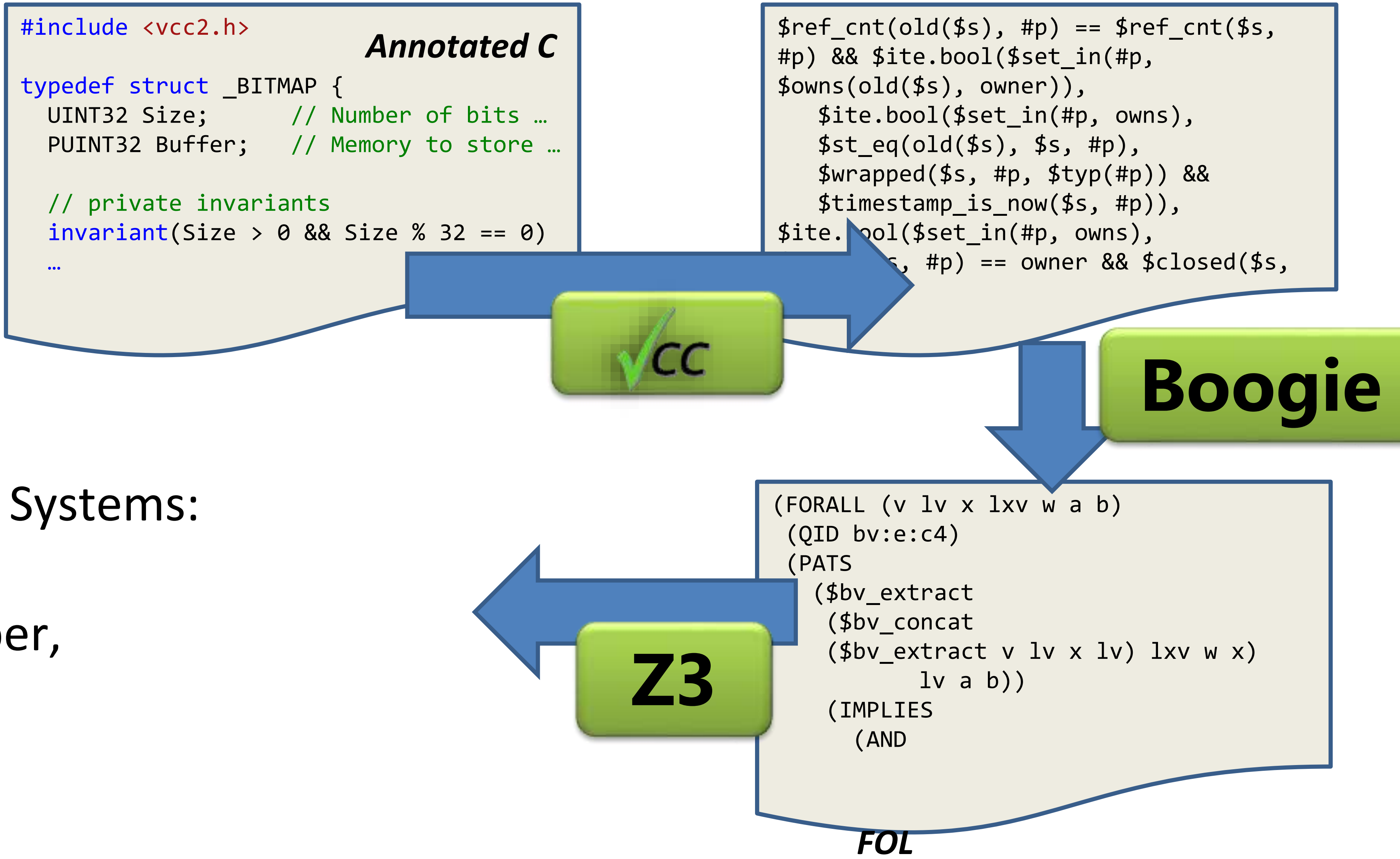
Formal proofs for Crypto Protocols

Program + Spec

$\forall x \exists y\ p(x) \rightarrow x \geq y$

also: Dynamics Tax tool, Visual Studio C++ compiler, Blockchain, Static Driver Verifier, Pex

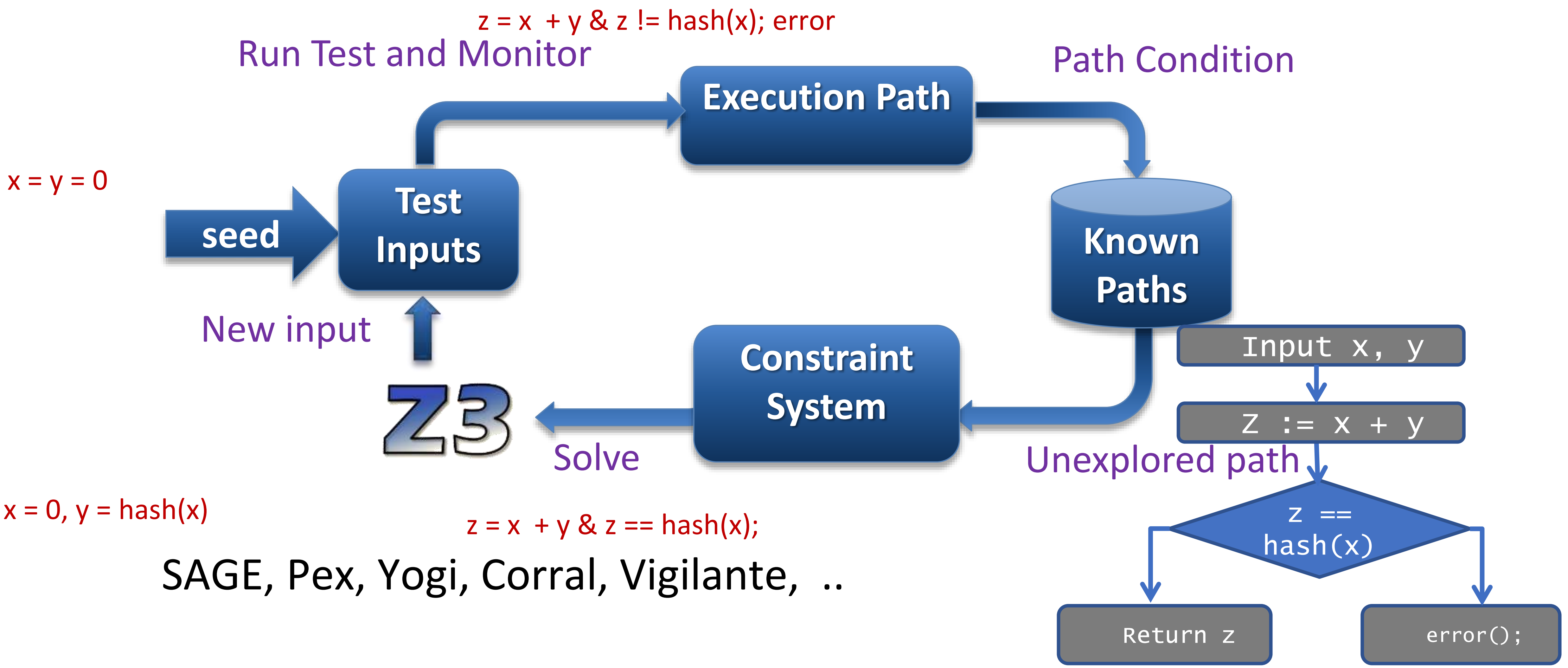
- Verified C compiler for the Microsoft Hyper-Visor 2008-2012
- Verified TLS protocols, Crypto Libraries, Parsers. Project Everest 2016-2022



Several Significant Systems:

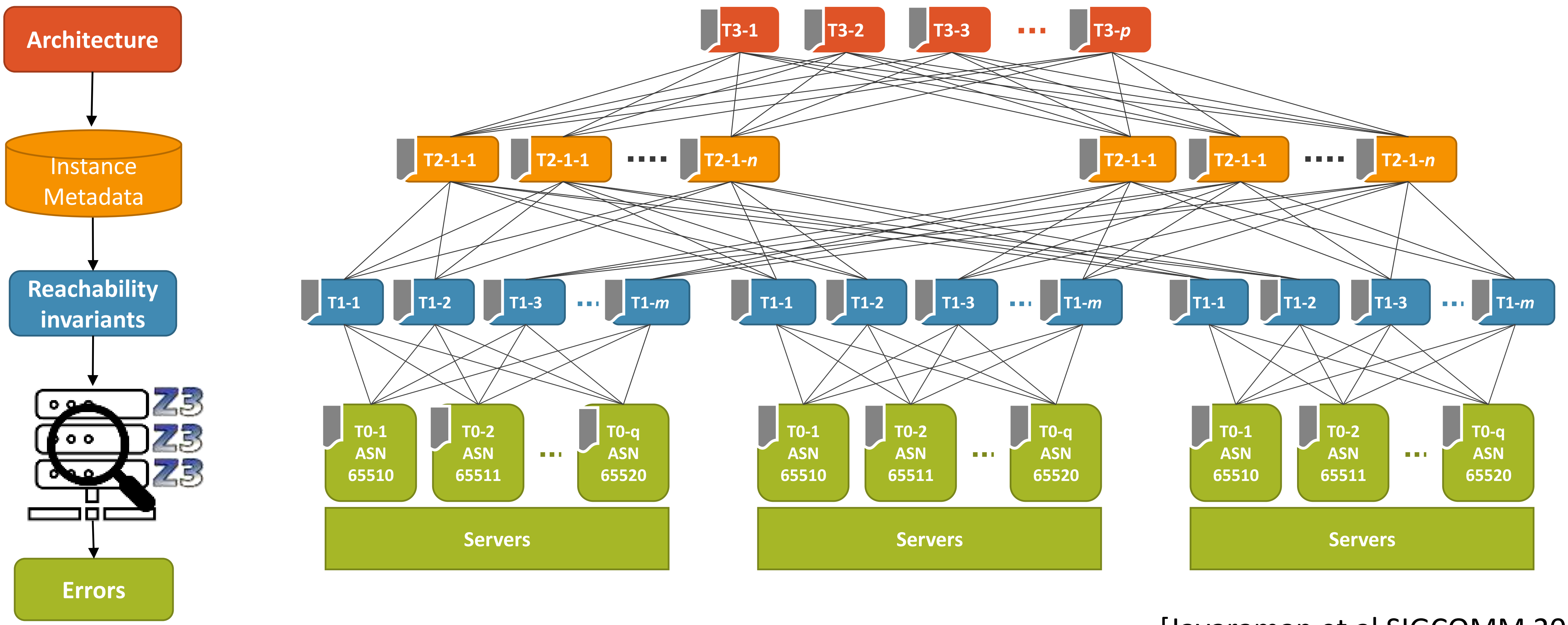
- Frama-C,
- VeriFast, Vyper,
- SeaHorn,
- K, Key

Dynamic Symbolic Execution for finding *million-dollar bugs*



SAGE, Pex, Yogi, Corral, Vigilante, ..

HyperScale Network Verification



[Jayaraman et al SIGCOMM 2019]

Connectivity Restrictions

- Host Firewalls
- Customer facing Network Security Groups
- Major refactoring of Microsoft's Edge ACL

Forwarding Policies

- Live monitoring of drift
- Pre-check before deployment
- Design validation

Local Validation: The Scalability Trick

Root Cause Complexity	Key Insight
• $O(N^3)$ • Billions of pairs of ToRs • Engineering challenge: Synchronized snapshot of FIBs	Exploit Azure network's regular structure • Each router has a fixed role for a set of addresses • Enough to verify role is enforced on each router Decompose into **local contracts** Parallelize and scale

SMT-based Algorithm

VRF name: default

Codes: C - connected, S - static, K - kernel,

O - OSPF, IA - OSPF inter area, ...

B E - eBGP, ...

...

Gateway of last resort:

B E 0.0.0.0/0 [200/0] via 30.10.192.12, ...

via ...

via ...

....

B E 10.3.129.224/28 [200/0] via 10.10.192.12, ...

via ...

...

$Define\ P, P_i\ (0 \leq i \leq n), and\ P_n$:

$P(\vec{x}) = P_1(\vec{x})$

$P_i(\vec{x}) = if\ r_i.prefix(\vec{x})\ then\ r_i.nexthops\ else\ P_{i+1}(\vec{x})$

$P_i(\vec{x}) = drop$

$r_{13}.prefix(\vec{x}) = 10.3.129.224 \leq \vec{x} \leq 10.3.129.140$

$r_{13}.nexthops = 10.10.192.12 \vee \ldots$

Check $C.range(\vec{x}) \wedge P \wedge \neg C.nexthops$

ACL Verification Engine

```
remark Isolating private addresses
deny ip 0.0.0.0/32 any
deny ip 10.0.0.0/8 any
deny ip 172.16.0.0/12 any
deny ip 192.0.2.0/24 any
...
remark Anti spoofing ACLs
deny ip 128.30.0.0/15 any
deny ip 171.64.0.0/15 any
...
remark permits for IPs without
       port and protocol blocks
permit ip any 171.64.64.0/20
....
remark standard port and protocol
       blocks
deny   tcp any any eq 445
deny   udp any any eq 445
deny   tcp any any eq 593
deny   udp any any eq 593
...
deny   53 any any
deny   55 any any
...
remark permits for IPs with
       port and protocol blocks
permit ip any 128.30.0.0/15
permit ip any 171.64.0.0/15
...
```

r_3 : $(10.0.0.0 \leq srcIp \leq 10.255.255.255) \wedge$ $protocol = 4$

$$
\begin{array}{rcll}
P(\vec{x}) & = & P_1(\vec{x}) & \\
P_i(\vec{x}) & = & r_i(\vec{x}) \vee P_{i+1}(\vec{x}) & \textbf{if } r_i.action = Allow \\
P_i(\vec{x}) & = & \neg r_i(\vec{x}) \wedge P_{i+1}(\vec{x}) & \textbf{if } r_i.action = Deny \\
P_n(\vec{x}) & = & false &
\end{array}
$$

IMANDRA

Imandra is a cloud-native automated reasoning engine.

Imandra's groundbreaking AI helps ensure the algorithms we rely on are safe, explainable and fair.

TRY IMANDRA ONLINE

INSTALL IMANDRA LOCALLY

Verifying ReasonReact component logic — ReasonML & Imandra

4 September 2018

Spotlight on an Imandra user:

In 2017 Aesthetic Integration partnered with Goldman Sachs to help deliver the SIGMA X MTF Auction Book, a new orderbook venue implementing periodic auctions. Aesthetic Integration used Imandra, our own automated

Recursive Function Unfolding

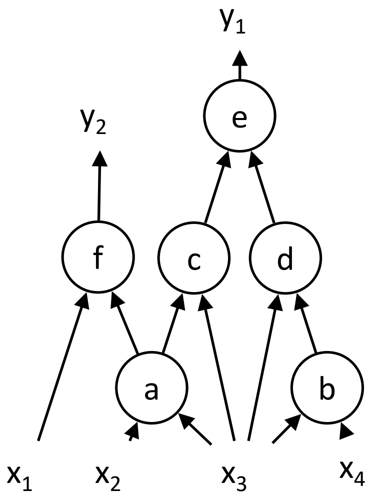
Algebraic ML Datatypes

Ground Arithmetic

https://try.imandra.ai/

Quantum: Reversible pebbling game

Example: find a pebbling strategy using 6 pebbles.



pebbling configurations

$P_1 = \{\phi\}$,

$P_2 = \{a\}$,

$P_3 = \{a, b\}$,

$P_4 = \{a , b, c\}$,

$P_5 = \{a, b, c, d\}$,

$P_6 = \{a, b, c, d, e\}$,

$P_7 = \{a, b, c, d, e, f\}$,

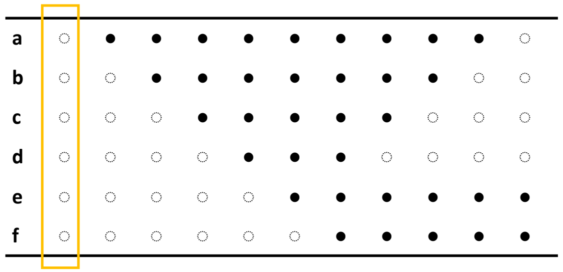
$P_8 = \{a, b, c, e, f\}$,

$P_9 = \{a, b, e, f\}$,

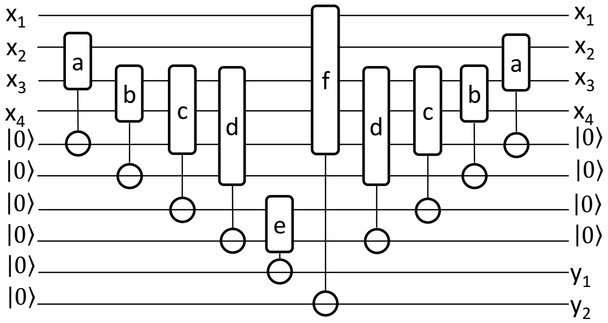
$P_{10} = \{a, e, f\}$,

$P_m = P_{11} = \{e, f\}$

space-time trade-off



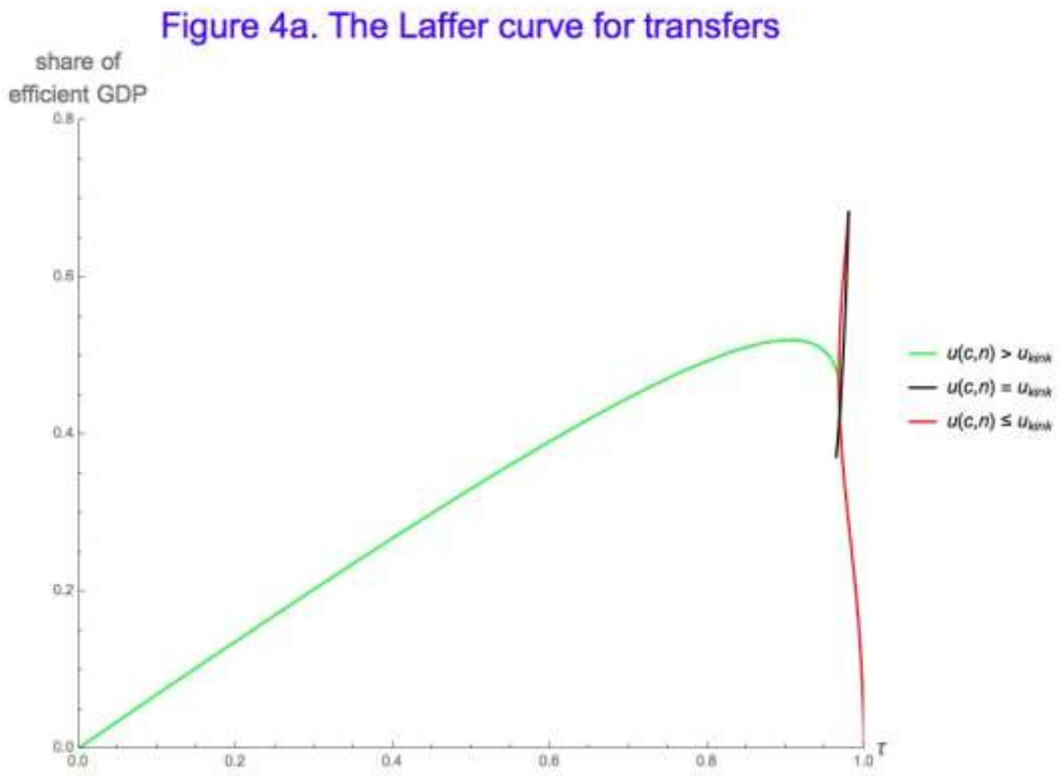
reversible circuit

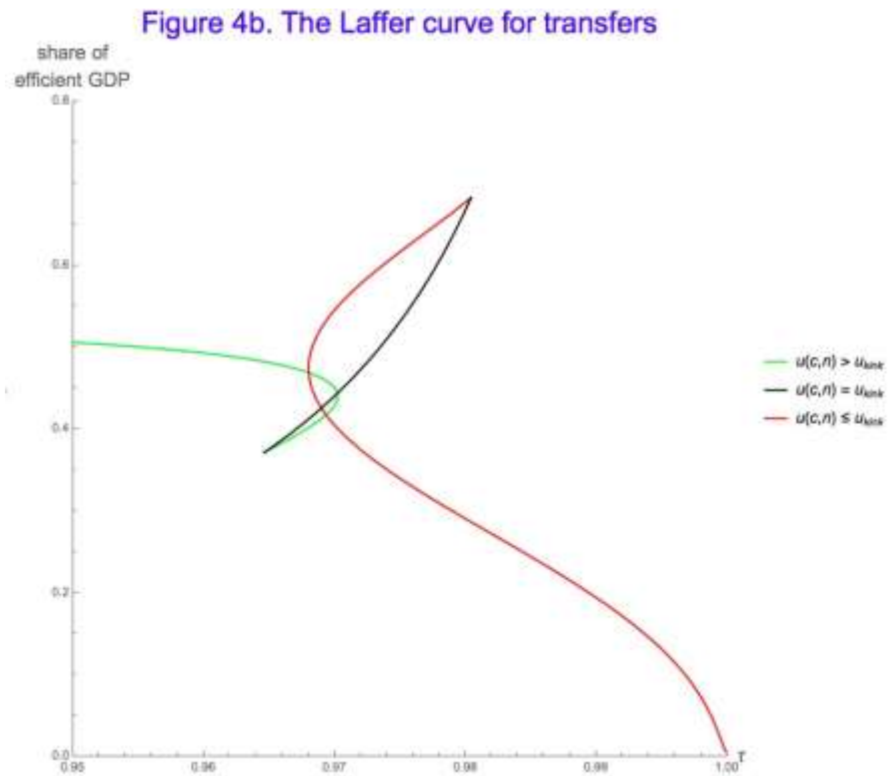


Example in Tutorial

Axiomatic Economics

Models of economics formulated using Non-linear Real Arithmetic





Casey Mulligan, University of Chicago, School of Economics uses Mathematica, Redlog, Z3

Example in Tutorial

Symbolic Analysis Engines

SLS, floats

νZ: Opt+MaxSMT

μZ: Datalog

Generalized PDR

Existential Reals

Model Constructing SAT

CutSAT: Linear Integer Formulas

Quantified Bit-Vectors

Linear Quantifier Elimination

Model Based Quantifier Instantiation

Generalized, Efficient Array Decision Procedures

Engineering DPLL(T) + Saturation

Effectively Propositional Logic

Model-based Theory Combination.

Relevancy Propagation

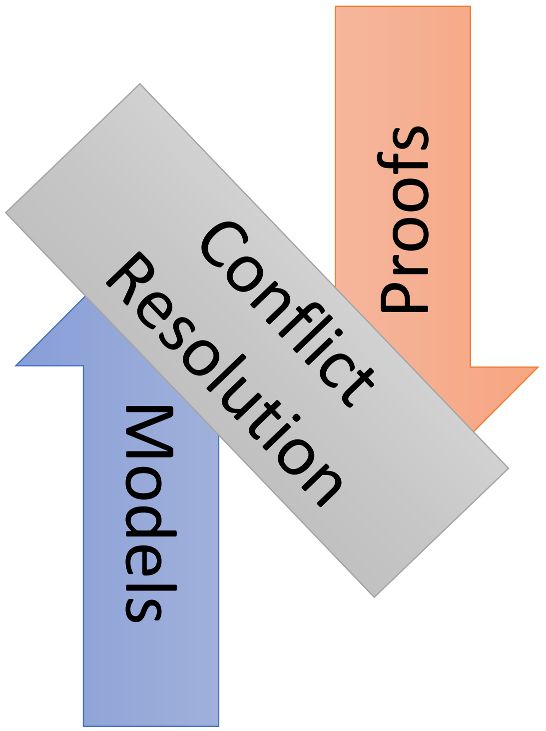
Efficient E-matching for SMT solvers

Model-based techniques in Automated Theorem Proving

Saturation x Search

Proof-finding

Model-finding



Two procedures

Resolution	DPLL
Proof-finder	Model-finder
Saturation	Search

Saturation: successful instances

Polynomial time procedures

Gaussian Elimination

Congruence Closure

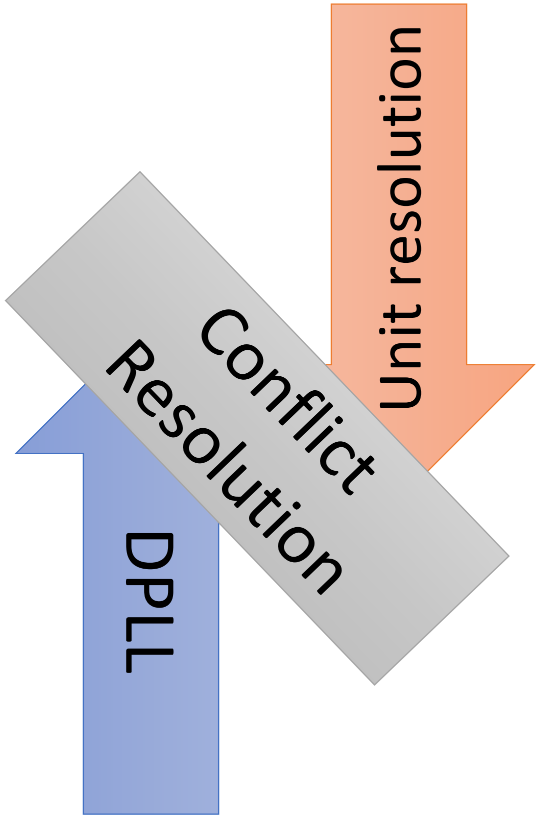
Search: successful instances

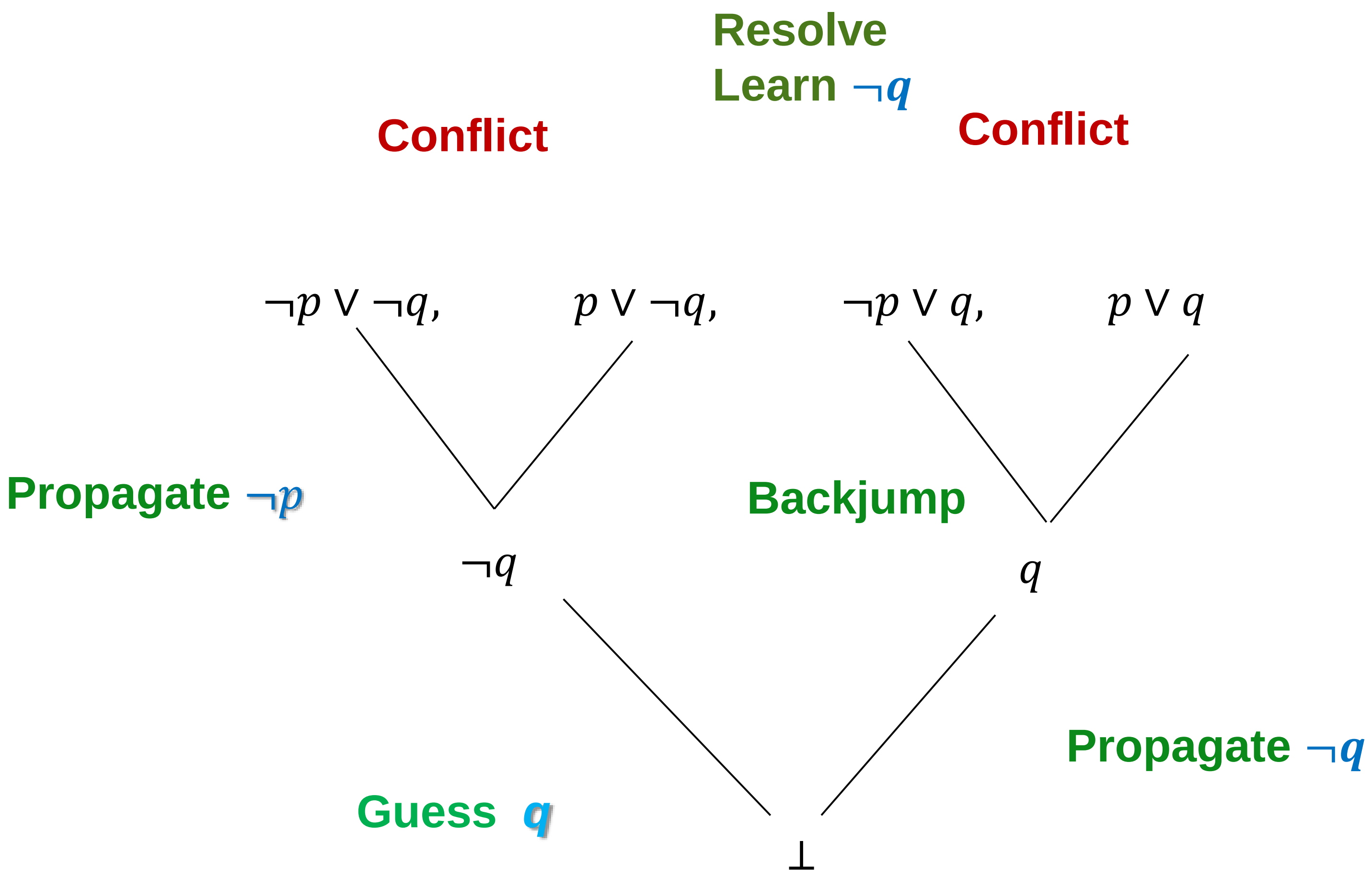
Decomposable Search Spaces

The “Cube” in “Cube & Conquer”

Some instances of model finding

CDCL: Conflict Driven Clause Learning





Linear Arithmetic

Fourier-Motzkin	Simplex
Proof-finder	Model-finder
Saturation	Search

Linear Arithmetic

Saturation:

$$\frac{a \le x, b \le x, c \le x, x \le d, x \le e}{a \le d, a \le e, b \le d, b \le e, c \le d, c \le d}$$

Model Finding:

$$\frac{a \le x, b \le x, c \le x, x \le d, x \le e}{a \le d, b \le d, c \le d, d \le e \quad a \le e, b \le e, c \le e, e \le d}$$

For models d = 2, e = 3

For models d = 4, e = 3

Other examples
(for linear arithmetic)

Generalizing DPLL to richer logics

[McMillan 2009]

Fourier-Motzkin **X** Conflict Resolution

[Korovin et al 2009]

Unate Lemmas

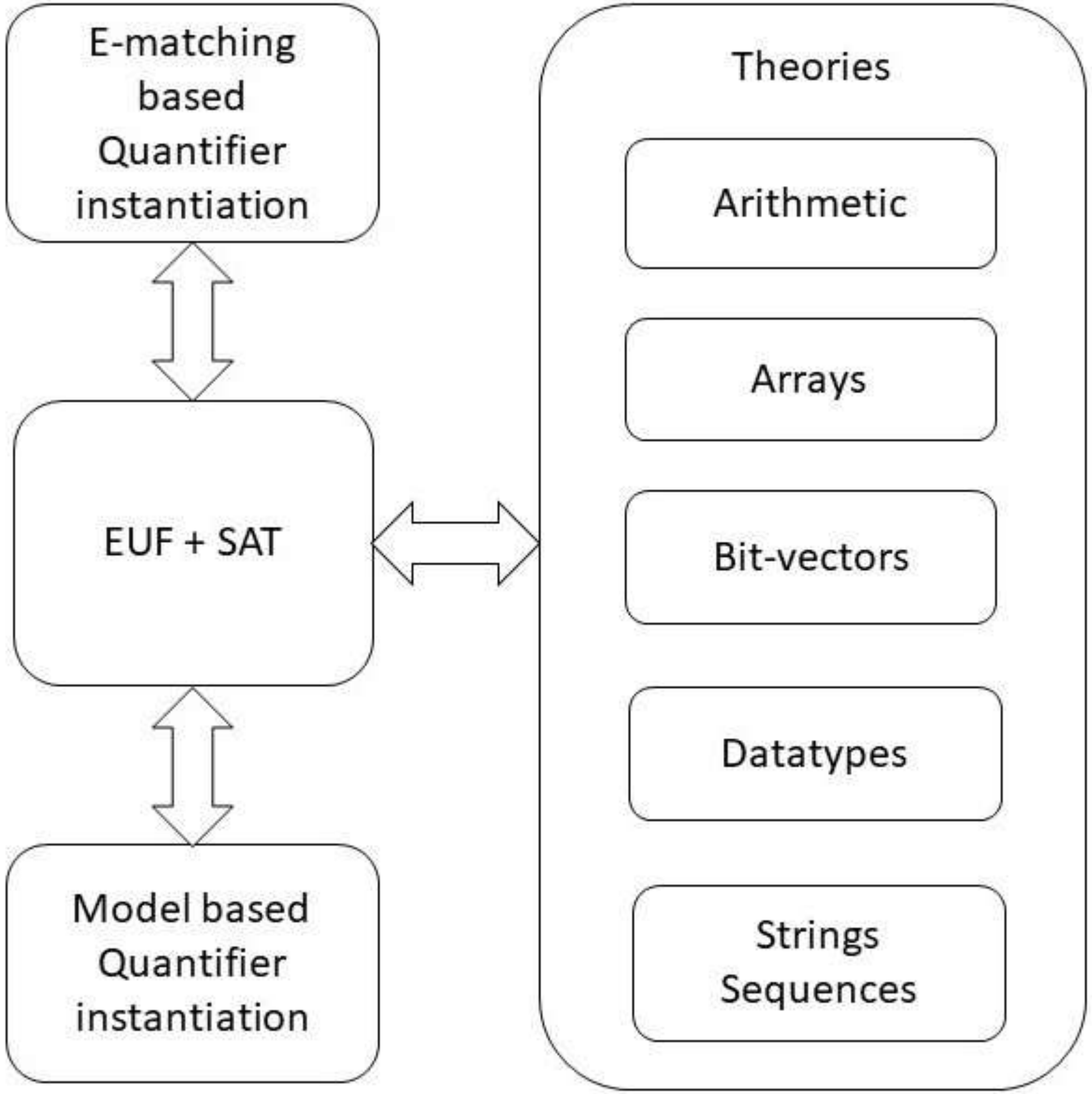
[Coton 2009]

Little engines of proof

Z3 Architecture

SMT = SAT + Theories

- SAT Solver handles search
- Theory Solvers handle theory reasoning
- Integration through equality sharing

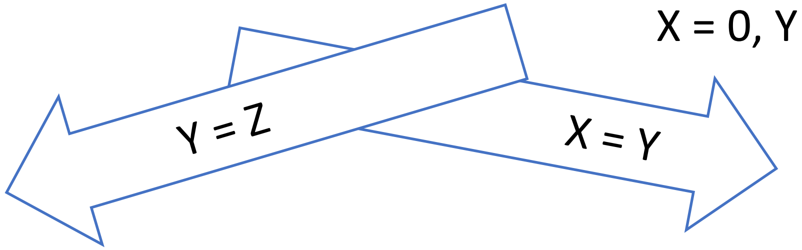


Model-based theory combination

- Each theory constructs a candidate model
- Each model implies some equalities
- Propagate equalities implied by candidate model
- Use backtracking if theories cannot reconcile equalities

Theory Solver 1 found solution
X + Y + Z = 1
X = Y = 0, Z = 1

Theory Solver 2 found solution
X + Y + Z = 2
X = 0, Y = Z = 1



Search for solution where X = Y = Z

[Moura & B, SMT 2007]

Model-based Quantifier Instantiation

Assume we are given $\psi \wedge \forall x\ \varphi[x]$,
then use model for ψ as starting point
for search of instantiations of $\forall x\ \varphi[x]$

```
(declare-fun f (Int) Int)
(declare-const a Int)
(declare-const b Int)

(assert (forall ((x Int)) (> (f x) (f a))))

(assert (> (f b) (f a)))

(check-sat)
```

ψ: $\quad f(b) > f(a)$

$\varphi[x]$: $\quad f(x) > f(a)$

Candidate model:

$a := 0, b := 1, f(x) := x = 0 ? 1 : 2$

Model check:

is $\underbrace{f(x)}_{x=0?1:2} \leq \underbrace{f(a)}_{=1}$ SAT?

Yes, set $x = a = 0$

Model-based Quantifier Instantiation

Assume we are given $\psi \land \forall x\ \varphi[x]$,
then use model for ψ as starting point
for search of instantiations of $\forall x\ \varphi[x]$

```
s.add(ψ)
while True:
    if unsat == s.check():
        return unsat
    M = s.model()
    checker = Solver()
    checker.add(¬φ^M[x])
    if unsat == checker.check():
        return sat
    M = checker.model()
    find t, such that x ∉ t, t^M = x^M.
    s.add(φ[t])
```

$t^M = x^M$ is not a strict requirement.

It is sufficient to use *M* to mine for a term *t* that still satisfies $\varphi[t]$

[Ge, de Moura CAV 2009, ..]

Generalized, Efficient Array Decision Procedures

Array store and read operations (`a[i]`), and

$$K(v)[i] = v$$
$$map_f(a_1, \ldots, a_n)[i] = f(a_1[i], \ldots, a_n[i])$$

Rules such as:

$$\text{idx}\ \frac{a \equiv store(b,i,v)}{a[i] \simeq v}$$

$$\Downarrow\ \frac{a \equiv store(b,i,v), \quad w \equiv a'[j], \quad a \sim a'}{i \simeq j \vee a[j] \simeq b[j]}$$

$$\Uparrow\ \frac{a \equiv store(b,i,v), \quad w \equiv b'[j], \quad b \sim b'}{i \simeq j \vee a[j] \simeq b[j]}$$

$$\text{ext}\ \frac{a{:}(\sigma \Rightarrow \tau), \quad b{:}(\sigma \Rightarrow \tau)}{a \simeq b \vee a[k_{a,b}] \not\simeq b[k_{a,b}]}$$

Model-based filters for restricting the application of these rules while retaining completeness.

[de Moura & B, FMCAD 2009]

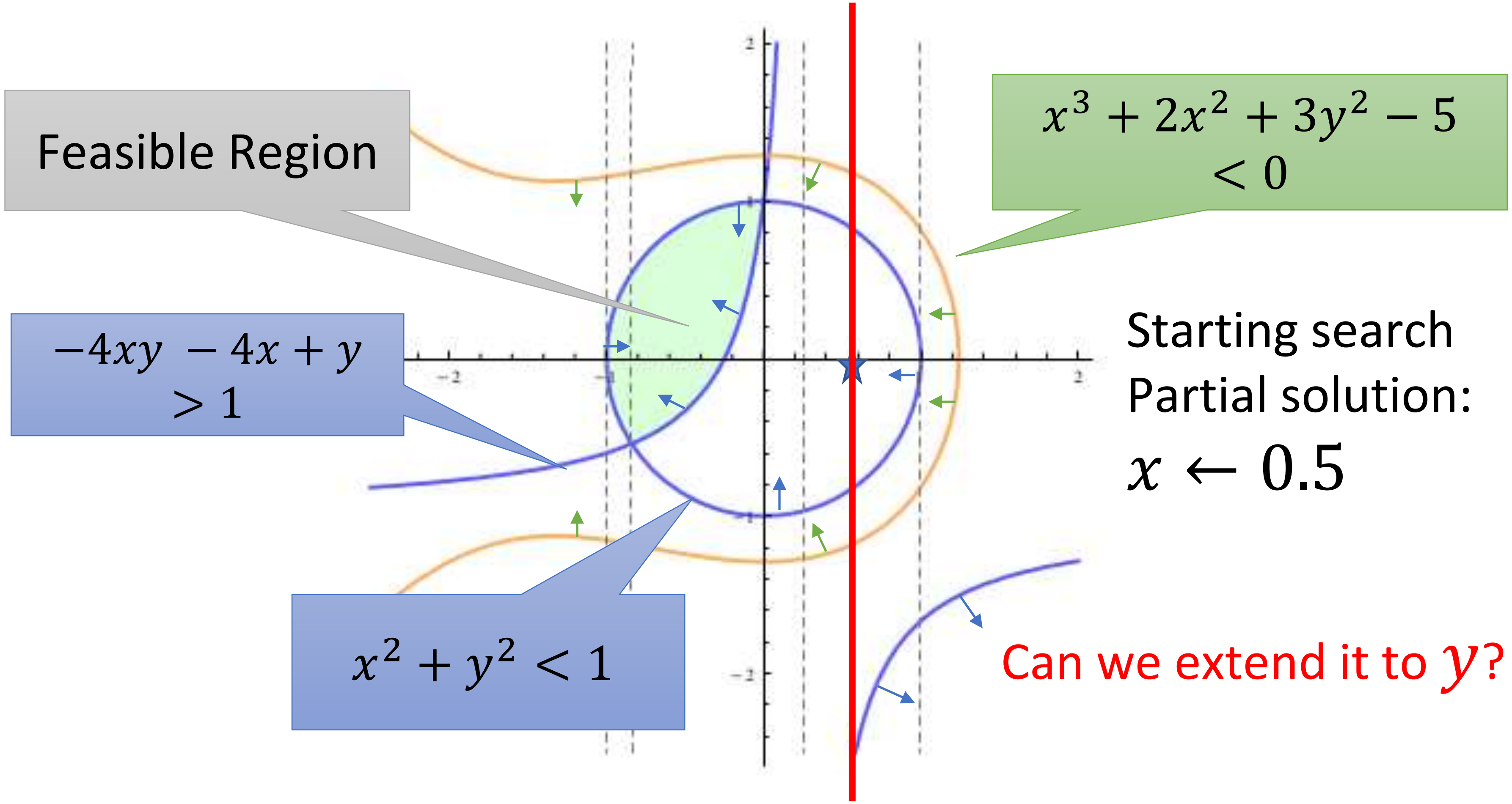
Polynomial Constraints

AKA
Existential Theory of the Reals

$$x^2 - 4x + y^2 - y + 8 < 1$$
$$xy - 2x - 2y + 4 > 1$$

NLSAT

Key ideas: Use partial solution to guide the search



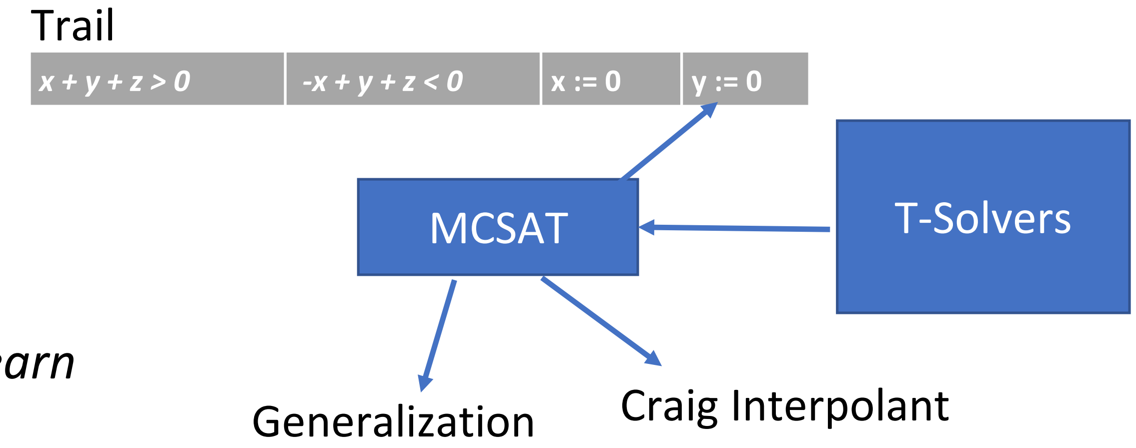
Starting search

Partial solution:

$x \leftarrow 0.5$

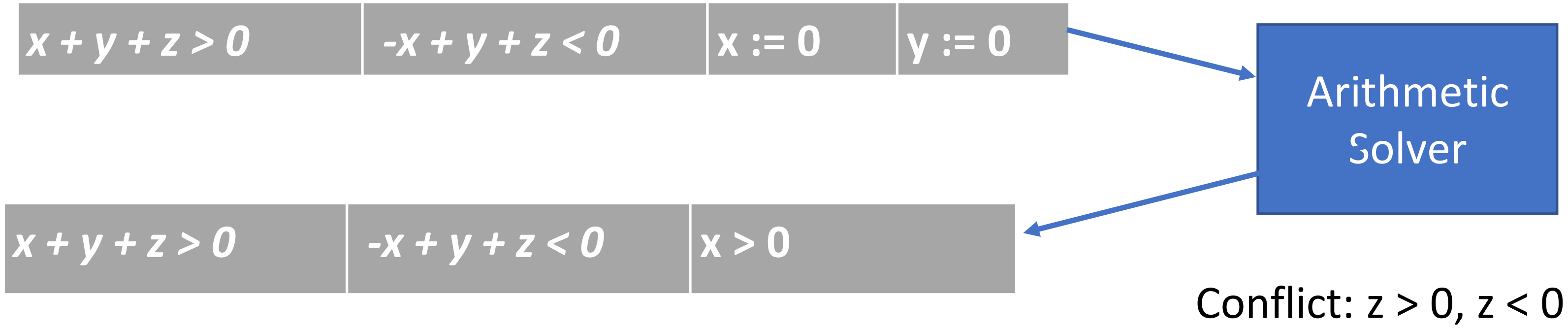
Can we extend it to y?

MCSat



Search

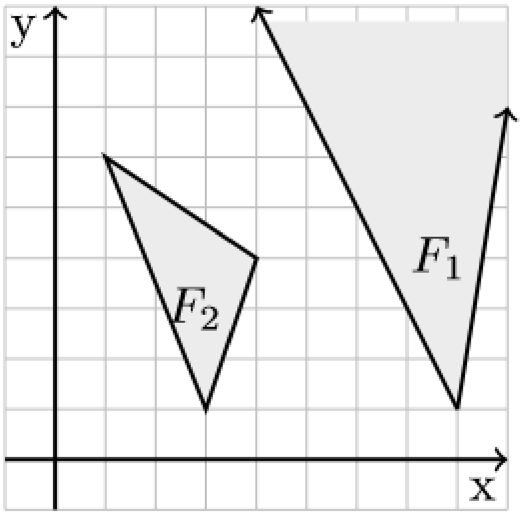
- *Trail: values guessed for sub-terms*
- *Propagate values*, derive consequences
- *Conflict resolution: Detect, backjump, learn*
- *Forget, restart, indexing,…*

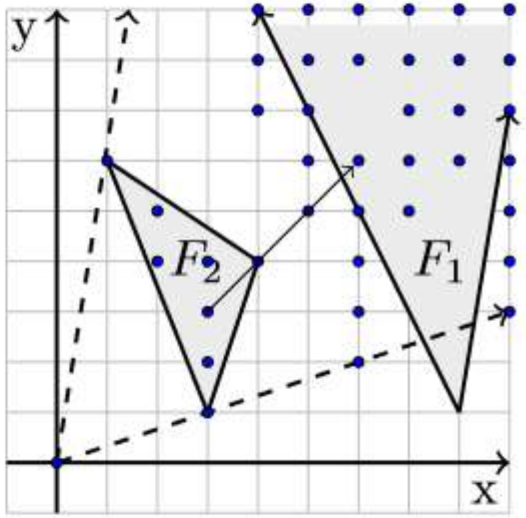


x > 0 is “explained” by the clause $x + y + z > 0 \wedge -x + y + z < 0 \Rightarrow x > 0$

[Jovanovich, Barret, de Moura, VMCAI 2013]

Solving LIA* using approximations - models and interpolants





F_1: $y + 2x \geq 17 \wedge 6x \ - y \leq 47$

F_2: $5x + 2y \geq 17 \wedge 3x \ - y \leq 8 \wedge 2x + 3y \leq 20$

$F_1 \wedge F_2$ is UNSAT

F_2^* : $\exists x_1, y_1 x_2, y_2, \ldots$
$F_2(x_1, y_1) \wedge F_2(x_2, y_2) \wedge \cdots \wedge F_2(x_k, y_k) \wedge$
$x = \sum x_i \wedge y = \sum y_i$

$F_1 \wedge F_2^*$ is SAT

[Levatich, B, Piskac, Shoham, to appear VMCAI 2020]

Solving LIA* using Approximations

Claim: F_2^* can be expressed in LIA

Claim: F_2 can be expressed as $\vec{\boldsymbol{x}} \in \bigcup_i a_i + B_i^*$
i.e., every LIA formula is a finite union of semi-linear sets.

Justification: $F_2^*(\vec{x}) \coloneqq \exists \boldsymbol{\mu}\boldsymbol{\lambda}. (\vec{x} = \sum_i \mu_i a_i + \lambda_i B_i) \wedge \bigwedge_i (\mu_i = 0 \rightarrow \lambda_i = 0)$

Brute force solver: express F_2^* using semi-linear sets, then use LIA solver

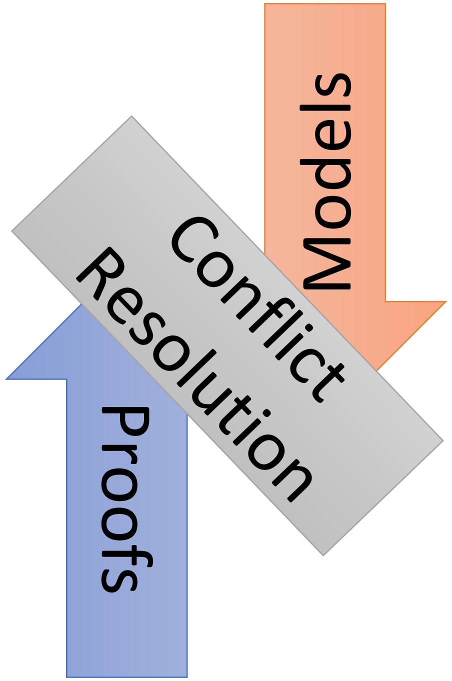
Catch: completely impractical

Solving LIA* using Approximations

Establish under-approximation $U^* \rightarrow F_2^*$ such that $U^* \wedge F_1$ is SAT

Solve $F_1 \wedge F_2^*$

Establish over-approximation $F_2^* \rightarrow O^*$ such that $O^* \wedge F_1$ is UNSAT



Under-approx $U^* \to F_2^*$ such that $U^* \wedge F_1$ is SAT

Initially, $U := \emptyset$, $U^* := (x, y) = (0,0)$

Maintain, $U = \bigcup_i a_i + \lambda B_i$ under-approximates F_2
and set $U^*(\vec{x}) := \exists \boldsymbol{\mu}\boldsymbol{\lambda}. (\vec{x} = \sum_i \mu_i a_i + \lambda_i B_i) \wedge \bigwedge_i (\mu_i = 0 \to \lambda_i = 0)$

Find x, y: $U^*(x_0, y_0) \wedge F_2(x, y) \wedge \neg U^*(x_0 + x, y_0 + y)$

Add (x, y) to U, reduce vectors using new element

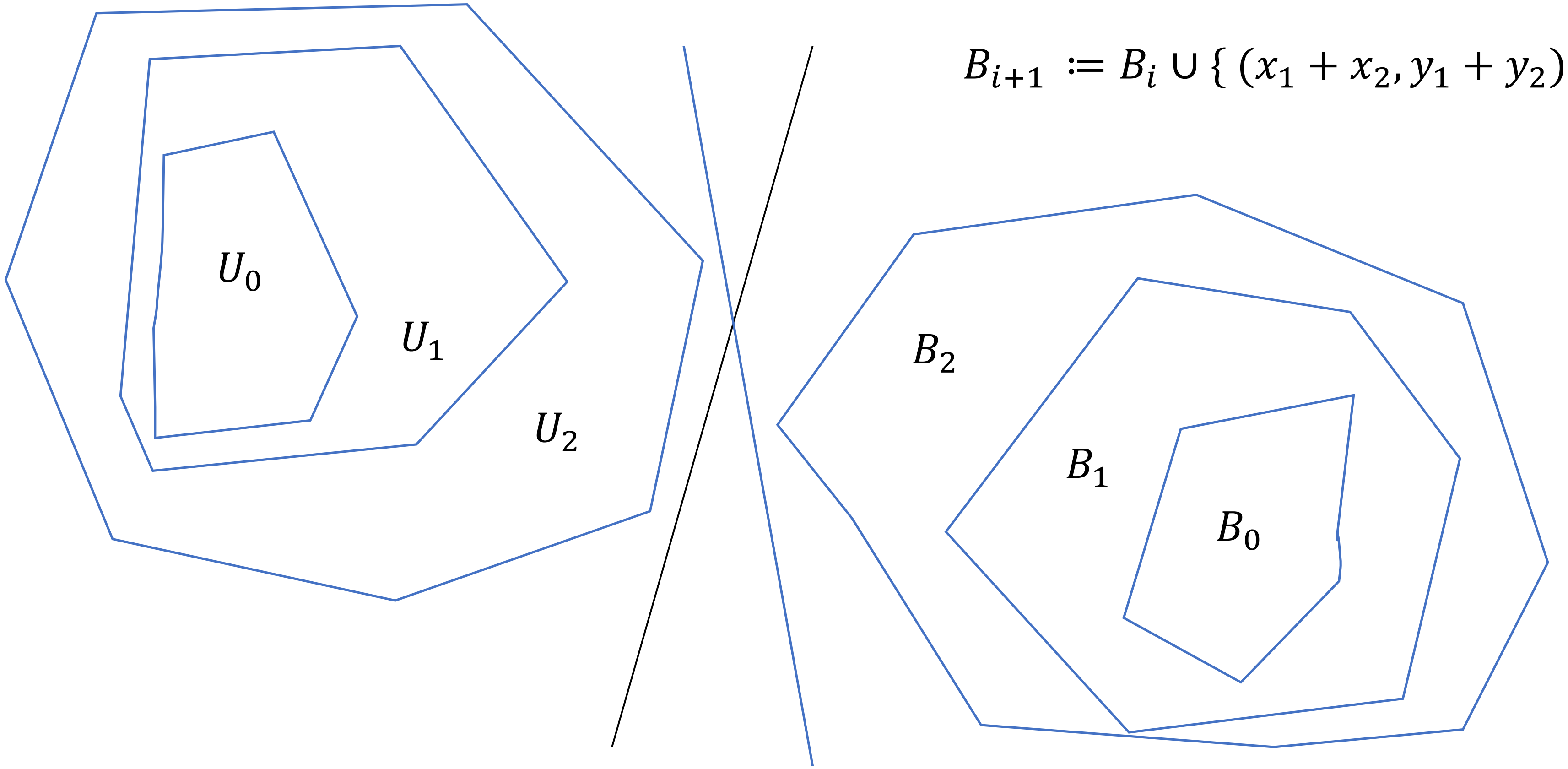
Over-approx F_2^* $\rightarrow O^*$ such that $O^* \wedge F_1$ is UNSAT

$U_0 \coloneqq \{\, (x, y) |\, U^*(x, y)\}$

$U_{i+1} \coloneqq U_i \cup \{\, (x_1 + x_2, y_1 + y_2) \mid U_{i+1}(x_1, y_1) \wedge\ F_2(x_2, y_2)\}$

$B_0 \coloneqq \{\, (x, y) |\, F_1(x, y)\}$

$B_{i+1} \coloneqq B_i \cup \{\, (x_1 + x_2, y_1 + y_2) \mid B_{i+1}(x_1, y_1) \wedge\ F_2(x_2, y_2)\}$



Over-approx F_2^* $\rightarrow O^*$ such that $O^* \wedge F_1$ is UNSAT

Initially O^* := true

Interpolate

$$U^*(x_0, y_0) \wedge F_2(x_1, y_1) \rightarrow I(x_0 + x_1, y_0 + y_1),$$
$$I(x, y) \rightarrow (F_2(x_2, y_2) \rightarrow \neg F_1(x_2 + x, y_2 + y))$$

Add conjunctions from I to O^* that are inductive, that is:

$$O^*(x, y) \wedge F_2(x_1, y_1) \rightarrow O^*(x + x_1, y + y_1)$$

Solving LIA* using Approximations

Q:

Can we leverage duality fully?

We were only exploiting the duality in one direction:

Under-approximation U* used to strengthen O*

But O* was not used to weaken U*

QSAT – Playing with Models and Cores

Instantiated to theories

- *Linear real arithmetic*
- *Linear integer arithmetic*
- *Algebraic datatypes*
- *Non-linear real arithmetic*
- (Bit-vectors)

Q: What is a good approach to learn strategies?

Q: Mixing theories and beyond theories that admit QE?

QSAT – Playing with Models and Cores

Two players

- $\exists$: $\exists x_1 \forall y_2 \exists x_3 \forall y_4 F,$ $\quad F_1 \leftarrow F_3 \leftarrow F$
- $\forall$: $\forall x_1 \exists y_2 \forall x_3 \exists y_4 \neg F$ $\quad F_2 \leftarrow F_4 \leftarrow \neg F$

State:

- A *model*, **M**,
  - for oponents solution
- A *strategy*, **S**,
  - function declaring how opponent would assign its variables in response
- Example
  - It is x_3's turn
  - **M** says $x_1 = 5, y_2 = 3$
  - **S** says $y_4 = x_3 + 2$

Example move:

- $F_3 \wedge S \wedge M$ is UNSAT
- $Core \leftarrow Some\ UNSAT\ Core\ of\ F_3 \wedge M \wedge S$
- $\exists C \leftarrow$ Model-based projection of $\exists y_2\ Core$
- $F_1 \leftarrow F_1 \wedge \neg \exists C$
- *Play game at level 1*

[Janota, B LPAR 2015 short]

Summary

- SMT solvers have come into quite wide-spread use in the past decade
  - Thanks to a large span of applications and technical advances

- Many solving techniques exploit duality of model search and deduction
  - Harnessing the interplay remains a throve of future opportunities
  - Beyond model-based techniques:
    - “Cubing”: Establish problem decomposition
    - “Strategies”: Prune search space that is no more likely to produce solutions

Research Question: Guiding Search

Problem: Tuned engines are prone to *overfitting*

State of art: Tune input parameters (using ML) and code back-off schemes

Opportunity: Use data-driven techniques to re-direct search

Learning cubes using DNNs

Goal: Choose most important case split

Train DNN using unsat formulas:

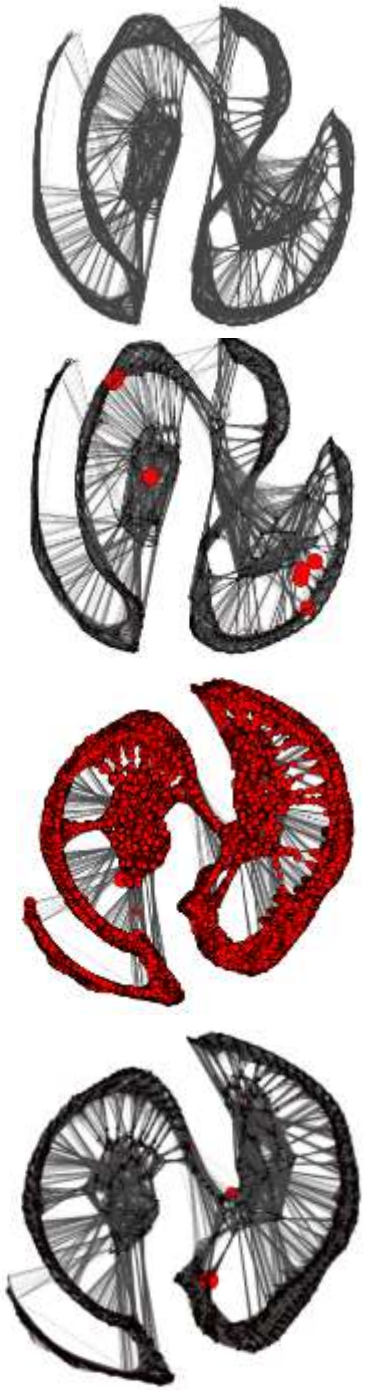
- Log conflict clauses
- Use DRAT-Trim to extract unsat core
- Score(v) := if v in core then 1 else 0.

Idea: only variables in a core are useful to case split on

DNN architecture: NeuroSAT (a graphical Neural Network)

Experiment:

- Generated 100,000 unsat problems from SAT competition 2014-2017
- Trained network with cores from the training set
- Integrated in SAT solvers glucose, MiniSAT, Z3 by *periodically refocusing* case split queue
- Evaluated on SAT2018
- Solved +10%/+20% more

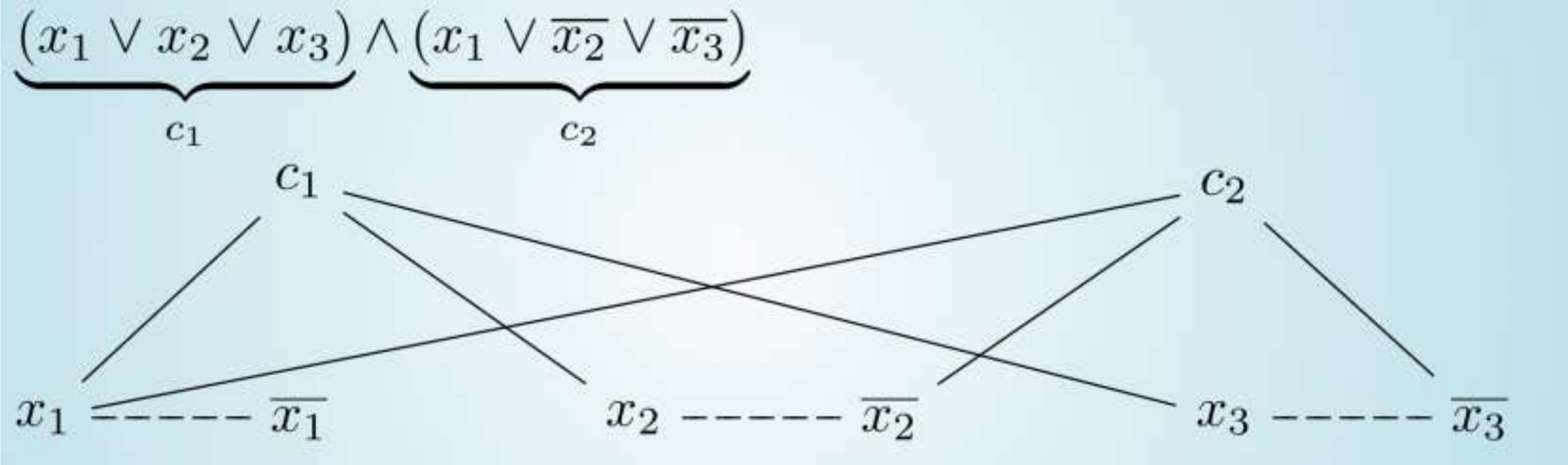


[Selsam, B. SAT 2019]

Background and Learnings

- Clauses:

- Graphical Network:



- DRAT proof Trail:

```
D:\z3\release>z3 core.cnf sat.drat.file=f.out

D:\z3\release>type f.out | more
-8783 9288 8770 8619 0
-8785 -2031 8873 9813 9765 8618 0
73 8747 0
-1777 -3258 -3564 -3558 9802 -3553 0
1532 9802 -3558 -3564 10024 0
9802 9809 -3558 10025 -3564 10024 0
-8520 10242 0
10128 8520 -1777 0
-1796 -1770 8967 1797 0
-7935 2432 4302 0
9999 2575 8141 8381 8132 0
-4945 9795 0
-8583 0
8751 0
```

- **Learning**: Access to DRAT proof trail enables 20/20 hindsight for optimization. Makes RL less relevant.

- **Future**: We could explore space of objective functions much more and instance specific uses.

Research Question: Scaling Search

Problem: How to use cloud resources to solve really-hard problems?

State of art: Cube & Conquer in SAT solvers, Branch & Bound in MIP

Opportunity: Use Azure infrastructure for scalable Cube & Conquer for SMT

Cube, Cloud and Z3

Rahul Kumar (MSR)
Miguel Neves (U Lisboa)