# *Programming Z3*

Nikolaj Bjørner
Microsoft Research
SMT workshop
July 8, 2019

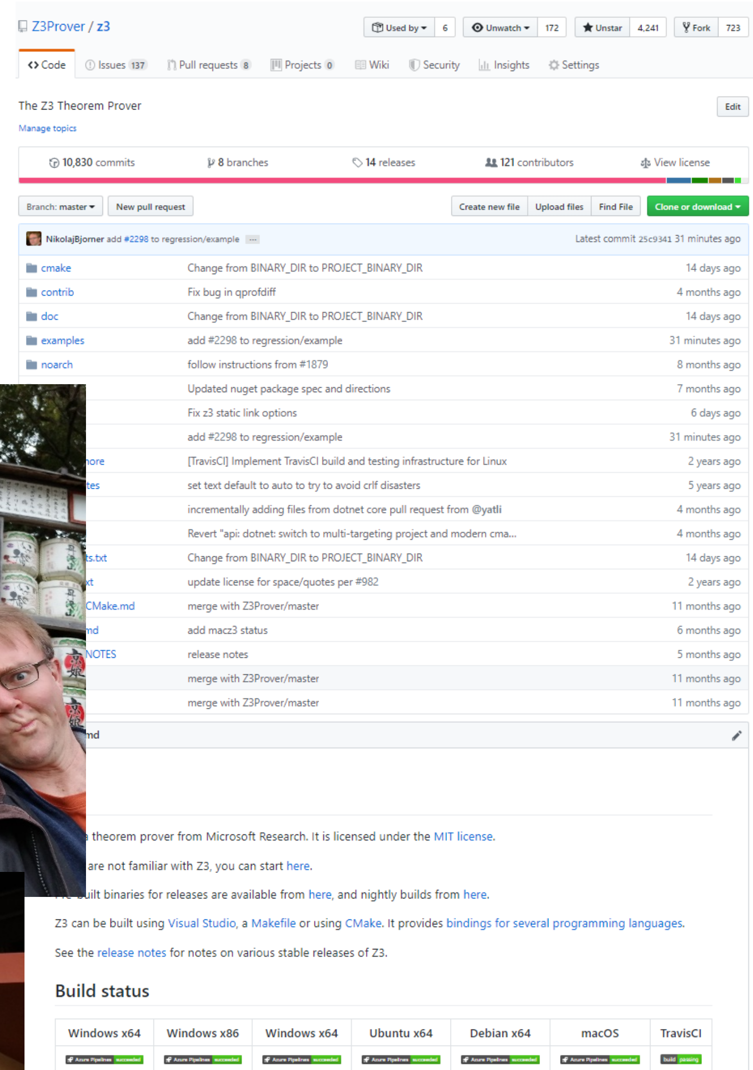
https://z3examples-nbjorner.notebooks.azure.com/j/notebooks
https://tinyurl.com/y3r67rd6

SMTWorkshop.ipynb

# Outline

- Past, present
  - An update on Z3
  - Some applications
- Some not so Secret Sauce
- Actually Programming Z3
- Active Directions

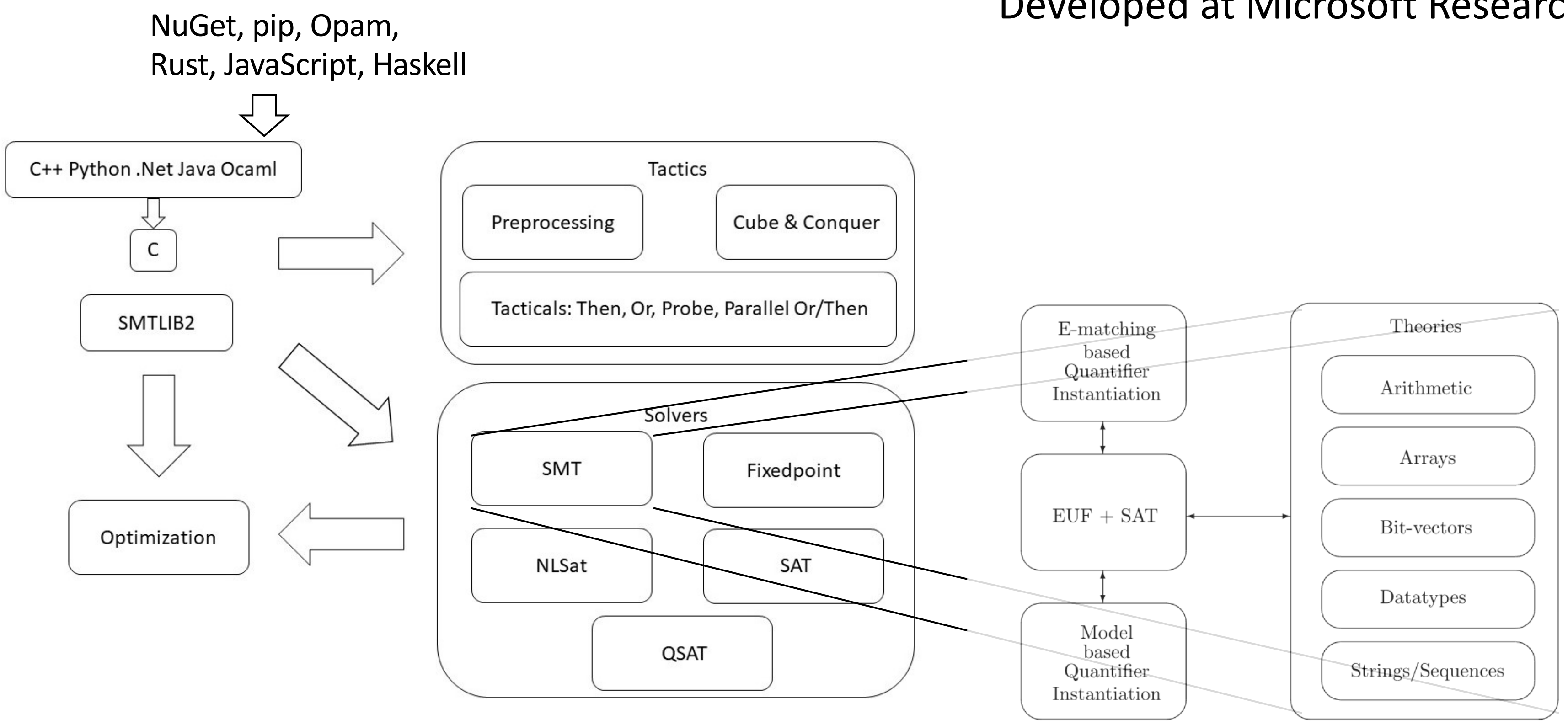
# An update


Z3Prover / z3
Used by 6
Unwatch 172
Unstar 4,241
Fork 723
Code
Issues 137
Pull requests 8
Projects 0
Wiki
Security
Insights
Settings
The Z3 Theorem Prover
Edit
Manage topics
10,830 commits
8 branches
14 releases
121 contributors
View license
Branch: master
New pull request
Create new file
Upload files
Find File
Clone or download
NikolajBjorner add #2298 to regression/example
Latest commit 25c9341 31 minutes ago
cmake Change from BINARY_DIR to PROJECT_BINARY_DIR 14 days ago
contrib Fix bug in qprofdiff 4 months ago
doc Change from BINARY_DIR to PROJECT_BINARY_DIR 14 days ago
examples add #2298 to regression/example 31 minutes ago
noarch follow instructions from #1879 8 months ago
Updated nuget package spec and directions 7 months ago
Fix z3 static link options 6 days ago
add #2298 to regression/example 31 minutes ago
[TravisCI] Implement TravisCI build and testing infrastructure for Linux 2 years ago
set text default to auto to try to avoid crlf disasters 5 years ago
incrementally adding files from dotnet core pull request from @yatli 4 months ago
Revert "api: dotnet: switch to multi-targeting project and modern cma... 4 months ago
Change from BINARY_DIR to PROJECT_BINARY_DIR 14 days ago
update license for space/quotes per #982 2 years ago
merge with Z3Prover/master 11 months ago
add macz3 status 6 months ago
release notes 5 months ago
merge with Z3Prover/master 11 months ago
merge with Z3Prover/master 11 months ago
theorem prover from Microsoft Research. It is licensed under the MIT license.
are not familiar with Z3, you can start here.
ilt binaries for releases are available from here, and nightly builds from here.
Z3 can be built using Visual Studio, a Makefile or using CMake. It provides bindings for several programming languages.
See the release notes for notes on various stable releases of Z3.
Build status
Windows x64
Windows x86
Windows x64
Ubuntu x64
Debian x64
macOS
TravisCI

https://github.com/Z3Prover/z3

Integrates a wealth of domains and solvers

Developed at Microsoft Research



# Symbolic Analysis Engines

SecGuru

Dafny

FORMULA

Modeling Foundations.

Vcc

Yogi

SLAM

SAGE

Pex

TERMINATOR

SLAYER

HAVOC

BOOGIE

SLS, floats

$\nu$Z: Opt+MaxSMT

$\mu$Z: 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

Z3 Internals

# Some Microsoft Uses of Z3

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

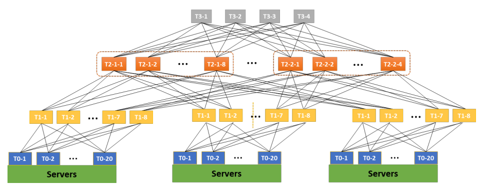
**Microsoft Security Risk Detection**

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**

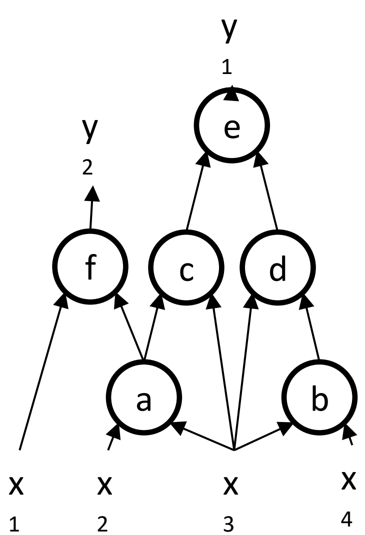

Formal proofs for WinQUIC

Program + Spec →

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

# Quantum: Reversible pebbling game

Example: find a pebbling strategy using 6 pebbles.




pebbling configurations

$P_1 = \{\phi\}$,

$P_2 = \{a\}$,

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

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

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

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

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

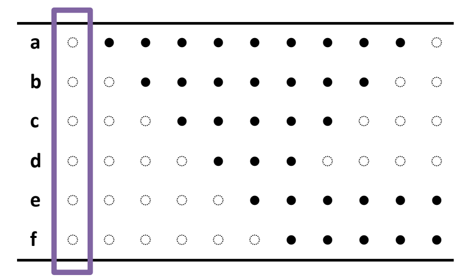
$P_8 = \{a, b, c, e, f\}$,

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

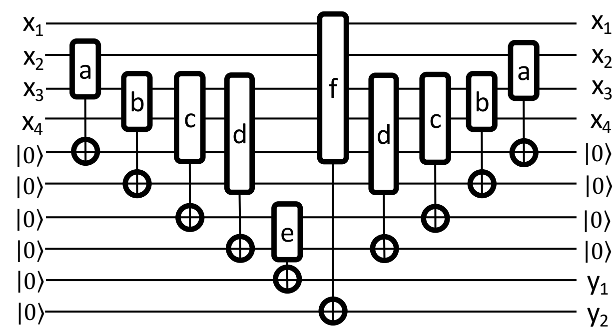
$P_{10} = \{a, e, f\}$,

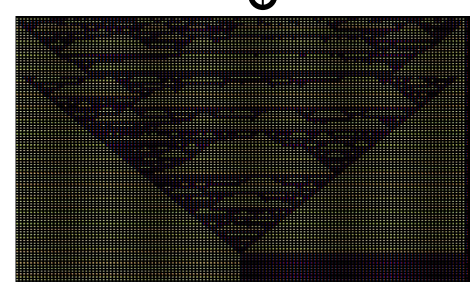
$P_m = P_{11} = \{e, f\}$

space-time trade-off




reversible circuit

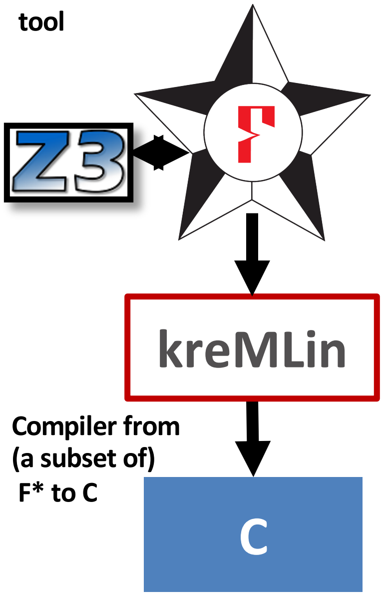


**DATE-2019. Giulia Meuli** Mathias Soeken, Giovanni De Micheli (EPFL)
Martin Roetteler, B (Microsoft)

# Everest, EverCrypt, EverParse

**F*: A general purpose programming language and verification tool**




**Compiler from (a subset of) F* to C**

**Math spec in F***

`poly1305_mac` computes a polynomial in GF($2^{130}$-5), storing the result in `tag`, and not modifying anything else

```
val poly1305_mac: tag:nbytes 16 →
                  len:u32 →
                  msg:nbytes len{disjoint tag msg} →
                  key:nbytes 32 {disjoint msg key ∧ disjoint tag
→
                  ST unit
  (requires (λ h → msg ∈ h ∧ key ∈ h ∧ tag ∈ h))
  (ensures (λ h0 _ h1 →
       let r=Spec.clamp h0.[sub key 0 16] in
       let s=h0.[sub key 16 16] in
       modifies {tag} h0 h1 ∧
       h1.[tag] == Spec.mac_1305 (encode_bytes h0.[msg]) r s))
```

**Efficient C implementation**

Verification imposes no runtime performance overhead

```
void
poly1305_mac(uint8_t *tag, uint32_t len, uint8_t *msg, uint8_t *key)
{
    uint64_t tmp [10] = { 0 };
    uint64_t *acc = tmp
    uint64_t *r = tmp + (uint32_t)5;
    uint8_t s[16] = { 0 };
    Crypto_Symmetric_Poly1305_poly1305_init(r, s, key);
    Crypto_Symmetric_Poly1305_poly1305_process(msg, len, acc, r);
    Crypto_Symmetric_Poly1305_poly1305_finish(tag, acc, s);
}
```

EverCrypt Clients: Mozilla Firefox; WireGuard VPN; Linux Kernel Zinc crypto library; MirageOS unikernel; Tezos blockchain; Microsoft QUIC.

# Trusted Financial Software

I M A N D R A

- Recursive Function Unfolding
- Algebraic ML Datatypes
- Ground Arithmetic

## 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

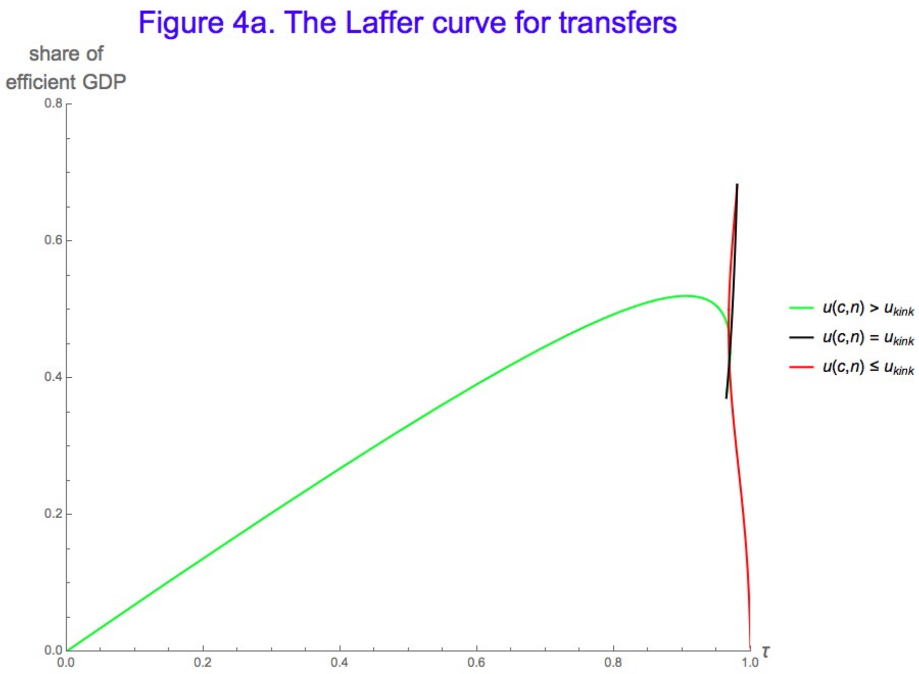
**Spotlight on an Imandra user:**

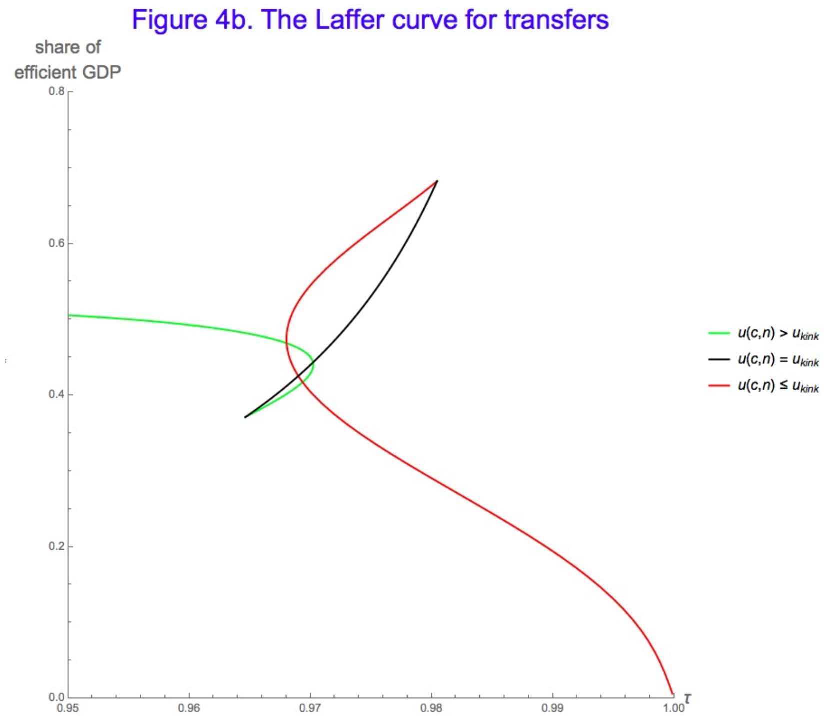
*In 2017 Aesthetic Integration partnered with Goldman Sachs to help deliver the SIGMA X MTF Auction Book, a*

https://try.imandra.ai/

# Axiomatic Economics

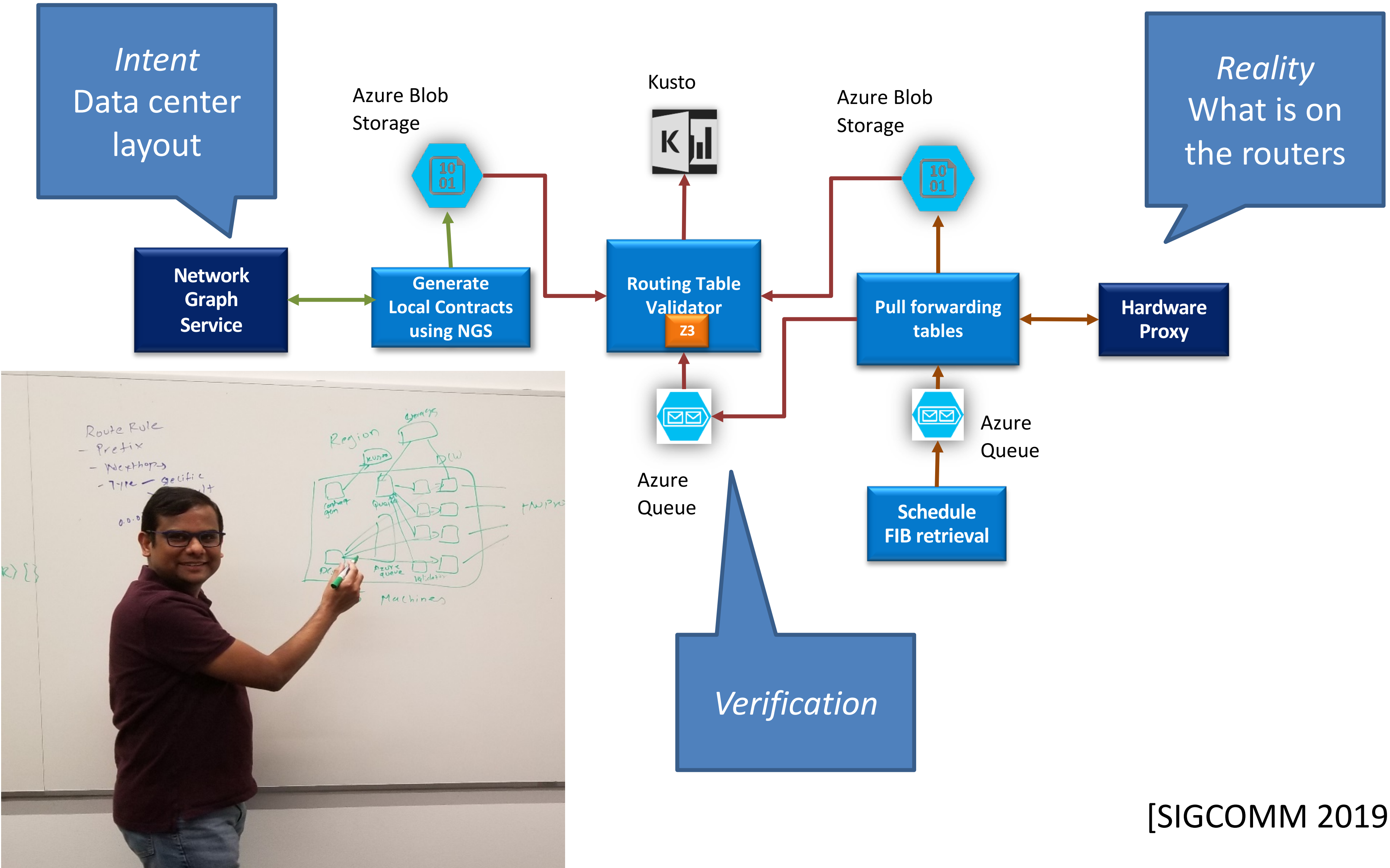
Models of economics formulated using Non-linear Real Arithmetic




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

# Verifying 100Ks Routers in Azure




[SIGCOMM 2019]

# BlockChain

Rigorous Methods for
Smart Contracts
Dagstuhl, June 1-5 2020

Org: Christakis, B, Maffeis, Rosu

- Solidity
  - Static Analysis, Leonardo Alt et al
  - Augur - Symbolic Execution
  - SPACER
- runtimeverification.com [Rosu + 30]
  - Verify byte code, ground truth
  - Arithmetic over 256-bit bit-vectors (use linear arithmetic)
- synthetic-minds.com [Srivastava]
  - Recursive functions (modeling a heap library)
- www.certora.com [Grossmann, Katz, Sagiv, Taube]
  - Quantifier Free Arrays + Bit-Vectors (QF_ABV)
  - Quantifier Free Functions + linear arithmetic
  - EPR + linear arithmetic

# What SMT features are used by applications?

- QF_ABV: symex with heap

When can policies **not** be sensitive to strings?

- QF_S: XML configurations, policies
- UFLIA: Boogie, Everest, Viper, ..
- QF_UFLIA, QF_ABV: Smart contracts

Formulas with 256 bit arithmetic

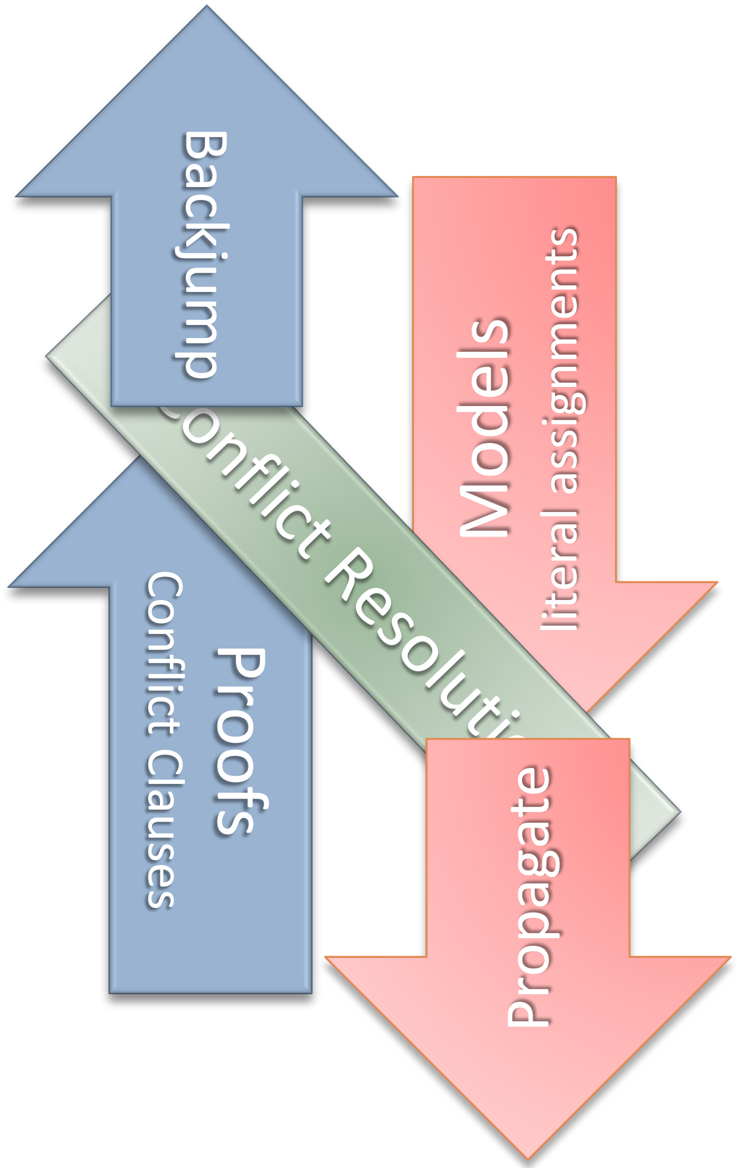
- ALL (Kitchen sink): Pex, Haskell
- Boolean Theories: Operations research applications, approximate counting (Xor)

# SOME (NOT SO) SECRET SAUCE

# Guiding inferences using models

| | |
|---|---|
| Model-based Theory Combination | [M,B 07] |
| Model-based Quantifier Instantiation | [G,M (B) 09] |
| Generalized Property Directed Reachability | [H,B 12] |
| Model-constructing Satisfiability | [J,M 11,12,13] |
| Model-based Quantifier Elimination | [G 14] |

# Mile High: Modern SAT/SMT search



# On Constructing Models

- Build a partial interpretation $M$ by setting an unconstrained variable

- Propagate with $M$

- Backtrack if propagation leads to a conflict

Role models

# On Constructing Models - variant

- Build a partial interpretation $M$ by setting an unconstrained variable
- Extend $M$ by solving sub-formula
- Propagate with $M$ globally
- Backtrack (locally or globally) if propagation leads to a conflict

# Model-based Theory Combination

Use a candidate model $M_i$ for a theory $T_i$ and propagate all equalities implied by $M_i$.

- If $M_i \vDash T_i \cup \Gamma_i \cup \{ u = v \}$ then propagate $u = v$

Hedging that other theories agree.

Backtrack if some other theory disagrees with $u = v$.

It is cheaper to enumerate equalities for a specific model.

Trick: instrument solver to create as few equalities as possible.

# Model-based Quantifier Instantiation

Assume we are given $\psi \wedge \forall x\ \varphi[x]$,
then use model for $\psi$ 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 $\varphi[t]$

# Model-based Quantifier Elimination

*project* can use $M_0$ to identify a finite set of solutions to $v$ that cover all of *e*

```
def qe(∃v⃗ . F):
    e, a = Solver(), Solver()
    e.add(F)
    a.add(¬F)
    G = False
    while sat == e.check():
        M₀ = e.model()
        M₁ = [ lit for lit in literals(F) if is_true(M₀.eval(lit)) ]
        # assume F is in negation normal form
        assert unsat == a.check(M₁)
        M₂ = a.unsat_core()
        π = project(M₂, v⃗)
        G = G ∨ π
        e.add(¬π)
    return G
```

Example: project $x$ from $(y_1 \leq x \ldots y_{10} \leq x \wedge x \leq z_1, \ldots, z_{10}) =$

$$y_1, \ldots, y_9 \leq y_{10}, y_{10} \leq z_1, \ldots, z_{10}$$

assuming $M(y_1), \ldots, M(y_9) \leq M(y_{10})$, corresponds to instantiating *x* by $y_{10}$

# Symbolic model checking as Satisfiability of Horn Clauses

**mc(x) = x-10** **if x > 100**

**mc(x) = mc(mc(x+11))** **if x ≤ 100**

**assert (x ≤ 101 ⇒ mc(x) = 91)**

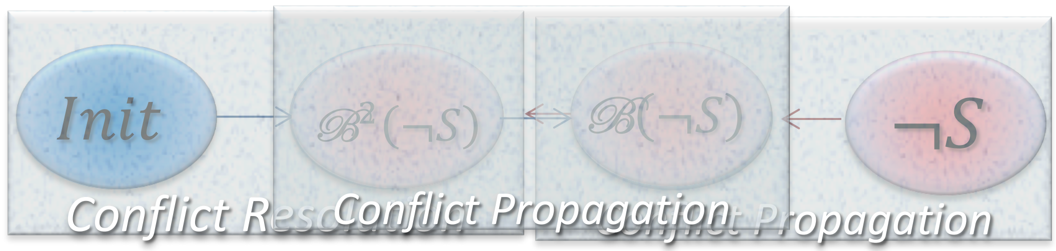
$$\forall X.\ X > 100 \rightarrow \mathrm{mc}(X, X-10)$$

$$\forall X, Y, R.\ X \leq 100 \wedge \mathrm{mc}(X+11, Y) \wedge \mathrm{mc}(Y, R) \rightarrow \mathrm{mc}(X, R)$$

$$\forall X, R.\ \mathrm{mc}(X, R) \wedge X \leq 101 \rightarrow R = 91$$

***Finds solution for mc***

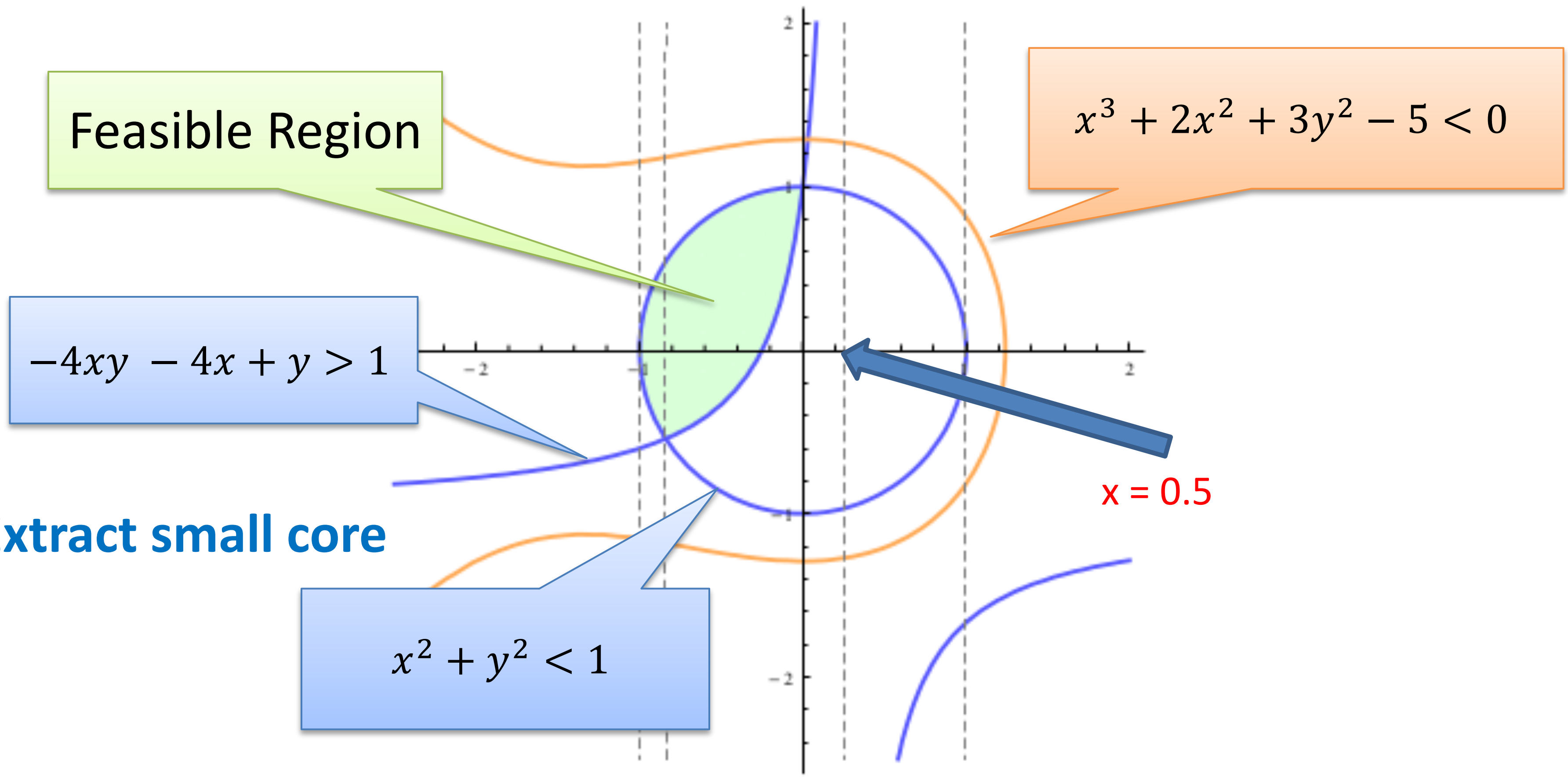
# Reachability: Mile-high perspective




- Over-approximate reachable states from Init.
  - Model of Invariant “from above”
- Under-approximate states that can reach $\neg S$
  - Model of trace “from below”
- If over and under-approximations are separated at bound *k,* produce certificate for separation
  - Strengthens over-approximation of reachable states
  - [SPACER: + under-approximations reachable states]

# Solving ∃R Efficiently

A key idea: Use partial solution to guide the search




**Extract small core**

Dejan Jovanovich & Leonardo de Moura, IJCAR 2012

# Strategies

Model-based methods:

- used with back-jumping
- guide inferences

Contrast with

Strategies:

- prune state space of choices
- Limit required inferences

# A Basic Quantifier Strategy

```
def strategy(M,j): return $\bigwedge_{M \neq null, a \in Atoms, level(j,a) < j} sign(M,a)$
def tailv(j): return $x_{j-1}, x_j, x_{j+1}, \ldots$

j = 1
M = null
while True:
  if $F_j$ ∧ strategy(M, j) is unsat:
    if j == 1:
      return F is unsat
    if j == 2:
      return F is sat
    C = Core($F_j$, strategy(M, j))
    J = Mbp(tailv(j), C)
    j = index of max variable in J ∪ {1,2} of same parity as j
    $F_j$ = $F_j$ ∧ ¬J
    M = null
  else:
    M = current model
    j = j + 1
```

[better ones for QBF by Rabe; Janota, Klieber, B]

# Branching Strategies

Can we *train* a branching strategy from benchmarks?

NeuroCore [Daniel Selsam, B. SAT 2019]:

- Train DNN about variables likely to be in core.
- *Periodic refocus* to recalibrate variable splitting queue.
- Claim: DNN provides good precision/recall. Speedups.
- Caveat: DNNs are likely not essential for *refocusing*

Solving formulas

Models, cores

Optimization, Fixed-points

# PROGRAMMING Z3 - REALLY

https://z3examples-nbjorner.notebooks.azure.com/j/notebooks
https://tinyurl.com/y3r67rd6

# SOME ACTIVE DIRECTIONS

# SM(Boolean Algebras)

- Plugin in SAT core for
  - Cardinality constraints
  - Pseudo Booleans
  - Xor
- SMT over other Boolean Functions?
  - propagators
  - In-processing
  - conflict analysis

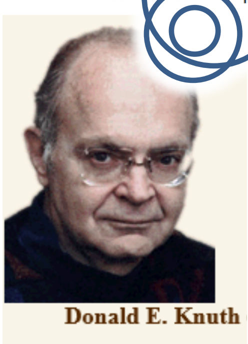
# Other recent/in-progress Solvers

- Monoids (strings) and Sequences
- Special Relations (partial, linear orders)
- Transitive Closure as a combinator
- Boolean Algebra and Presburger Arithmetic
- A theory for Job Scheduling

- New full arithmetic solver replacement by Lev Nachmanson

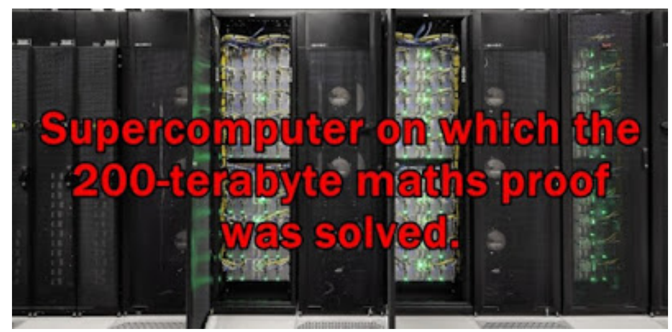
# Scaling SAT/SMT with lookaheads

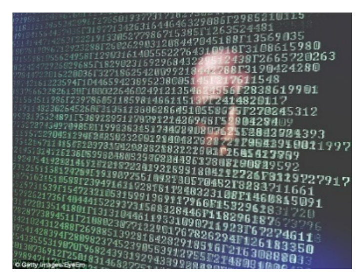
```
@<Construct the |look| table@>=
o,u=lmem[root].child,j=k=v=0;
while (1) {
  oo,look[k].lit=lmem[u].vcomp;
  o,lmem[u].rank=k++; /* |k| advances in preorder */
  if (o,lmem[u].child) {
    o,lmem[u].parent=v; /* fix parent temporarily for traversal */
    v=u,u=lmem[u].child; /* descend to |u|'s descendants */
  }@+else {
post: o,i=lmem[u].rank;
    o,look[i].offset=j,j+=2; /* |j| advances in postorder */
    if (v) oo,lmem[u].parent=lmem[v].vcomp; /* fix parent for lookahead
    else o,lmem[u].parent=0;
    if (o,lmem[u].link) u=lmem[u].link; /* move to |u|'s next sibling *
    else if (v) {
      o,u=v,v=lmem[u].parent; /* after the last sibling, move to |u|'s
      goto post;
    }@+else break;
  }
}
looks=k;
if (j!=k+k) confusion("looks");

@*Looking ahead. The lookahead process has much in common with what
we do when making a decision at a branch node, except that we
don't make drastic changes to the data structures. We don't
assign any truth values at levels higher than |proto_truth|; and
that level is reserved for literals that will be forced true if the
lookahead procedure finds no contradictions. We don't create
new binary implications when a ternary clause gets a false literal;
we estimate the potential benefit of such binary implications instead.
```




Sat11.w
TAoCP
Vol 4B sec 7.2.2.2

CDCL + lookahead
Actively pursued for scaling SAT



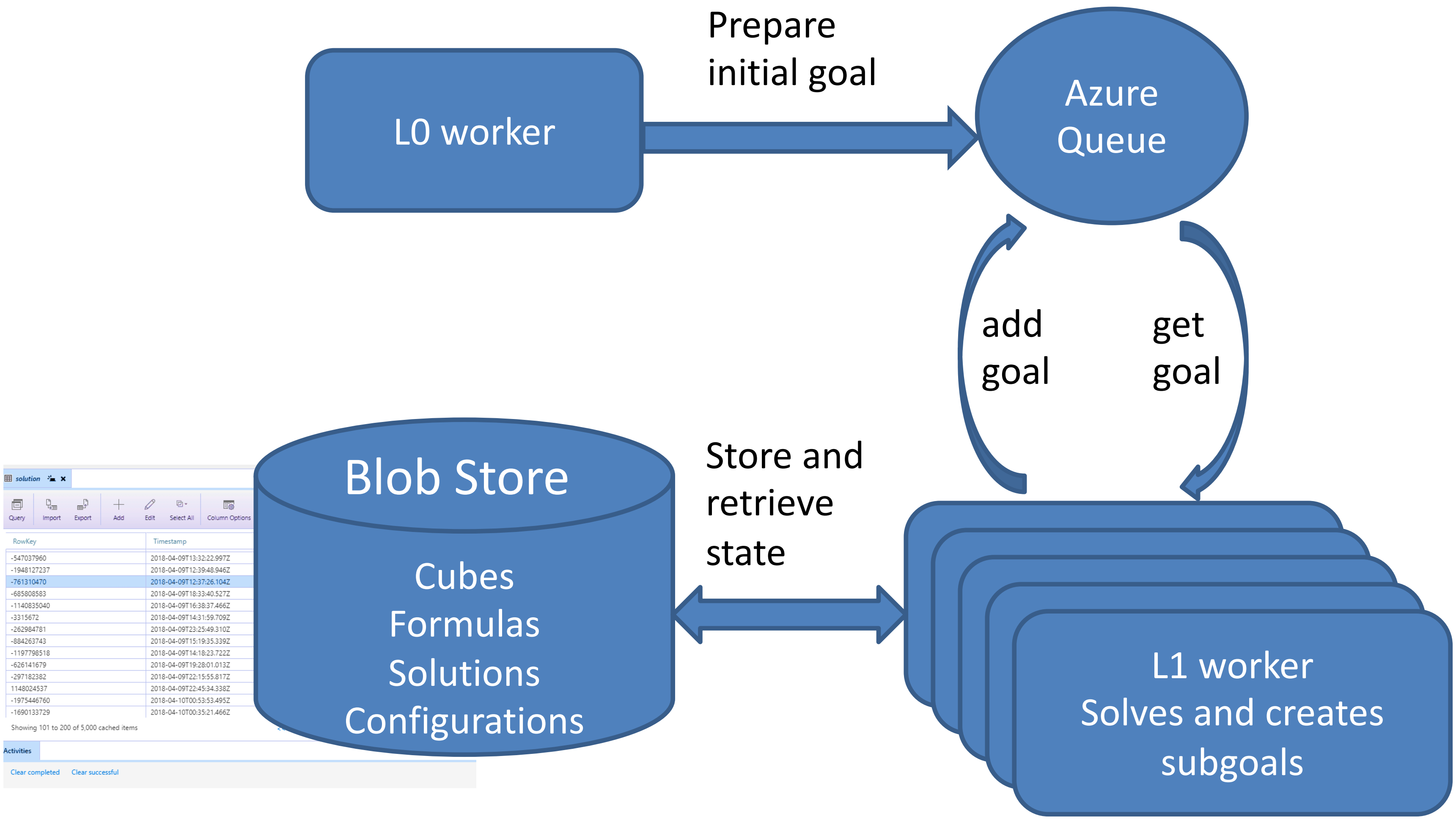



Oliver Kullman

Marijn Heule

# The Cube, the Cloud and Z3

Rahul Kumar (MSR)
Miguel Neves (U Lisboa)



# Summary

- SMT applications and the Gartner Hype curve
- Models and Strategies in SMT search
- Programming Z3: BMC, MaxSAT, AllSAT, Cubes
- Directions: Scaling SAT/SMT, CP theories

Thanks: Arie Gurfinkel, Marijn Heule, Rahul Kumar, Leonardo de Moura, Lev Nachmanson, Nina Narodytska, Miguel Angelo Da Terra Neves, Daniel Selsam, and Christoph Wintersteiger.