A Tour of CVC4

Morgan Deters
mdeters@cs.nyu.edu

Cesare Tinelli
cesare-tinelli@uiowa.edu

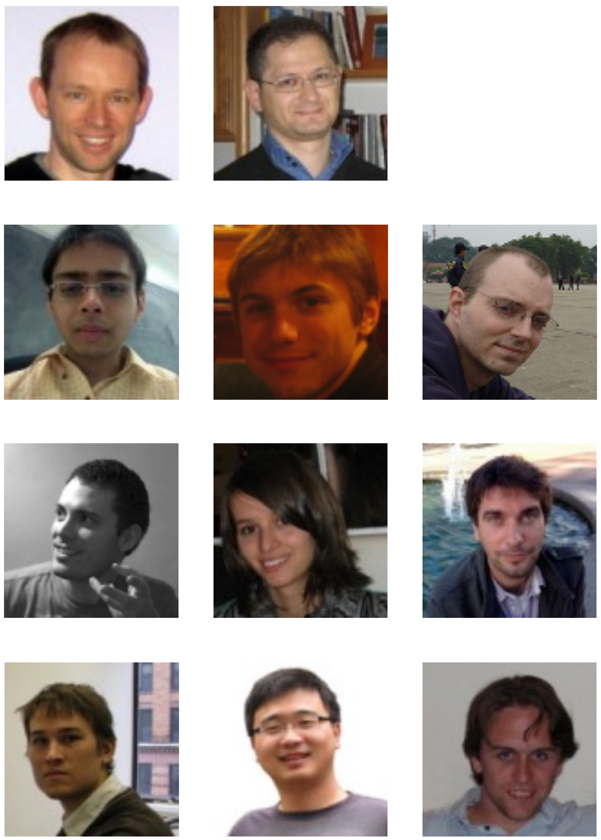
Tim King
tim.king@imag.fr

Andrew Reynolds
andrew.reynolds@epfl.ch

Clark Barrett
barrett@cs.nyu.edu

CVC4 is supported in part by the Air Force Office of Scientific Research, Google, Intel Corporation, the National Science Foundation, and Semiconductor Research Corporation

The CVC4 Team



Clark Barrett (NYU)
Cesare Tinelli (U Iowa)

Kshitij Bansal (NYU)
François Bobot (CEA)
Chris Conway (Google)

Morgan Deters (NYU)
Liana Hadarean (NYU)
Dejan Jovanović (SRI)

Tim King (Verimag)
Tianyi Liang (U Iowa)
Andrew Reynolds (EPFL)

Agenda

- Introduction and status report for CVC4

- Arithmetic
- Quantifiers (finite model finding)

- Examples/demos

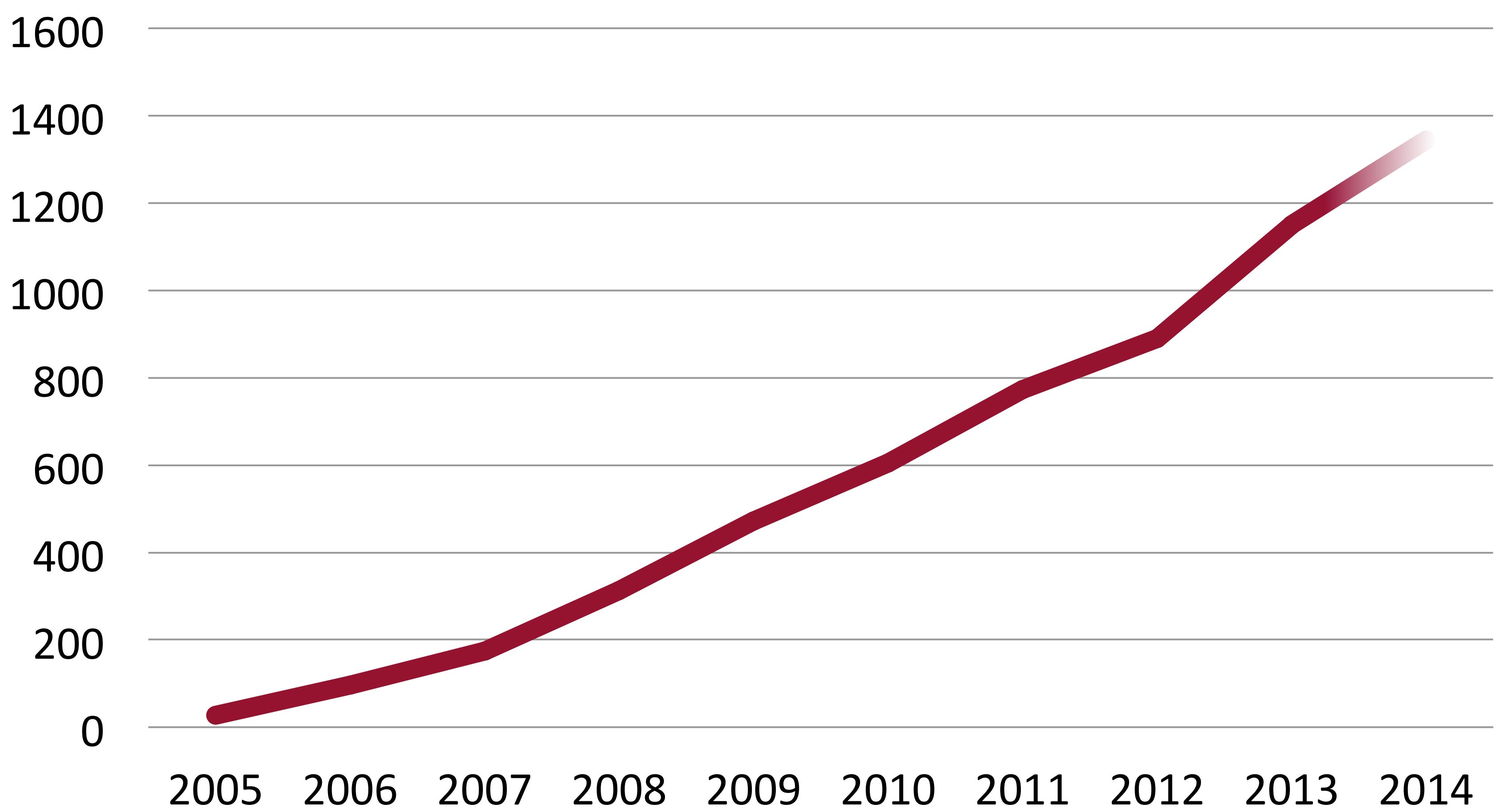
Automated Reasoning

- Historically automated reasoning meant uniform proof procedures for FOL
- More recent trend is decidable fragments
  - Domain-specific reasoning
  - Equality
  - Arithmetic
  - Data structures (arrays, lists, records)

Automated Reasoning

- Examples
  - SAT – propositional, Boolean reasoning
    - efficient
    - expressive (NP) but involved encodings
  - SMT – first order, Boolean + DS reasoning
    - loss of efficiency
    - improves expressivity and scalability

Articles mentioning SMT over time



Applications of SMT

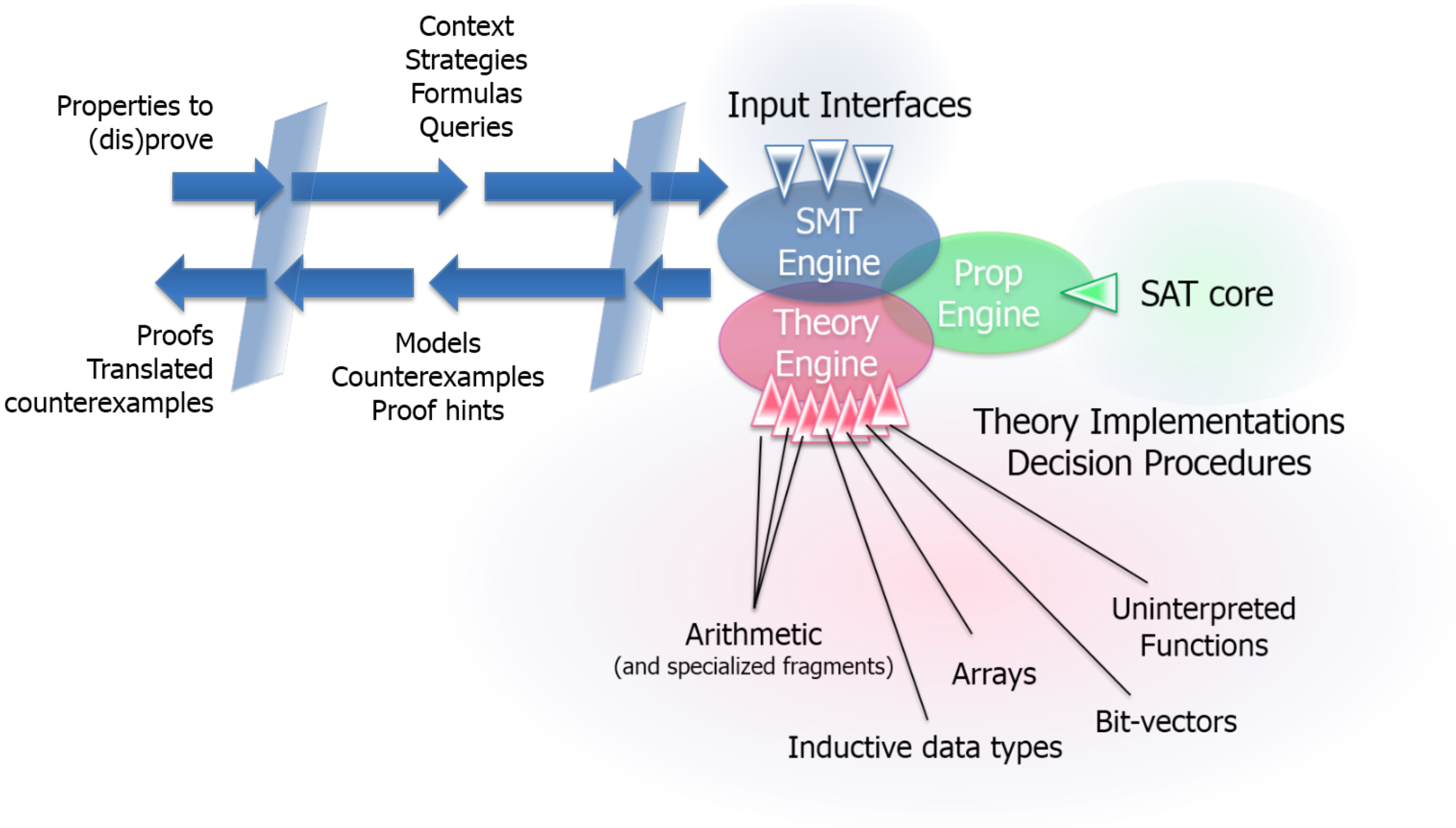
- extended static checking
- predicate abstraction
- model checking
- scheduling
- test generation
- synthesis
- (in)feasible paths
- verification

More on Expressivity

- Many theories of interest have efficient decision procedures for conjunctions of facts

- …but in practice we need arbitrary Boolean combinations
  - also combined theory constraints
  - quantifiers

Architecture of SMT



History of CVC

- SVC – 1996, own SAT solver
- CVC – Chaff, optimized internal design
- CVC Lite – 2003, rewrite to make more flexible
  - supported quantifiers
- CVC3 – major overhaul
  - better DP implementations
- CVC4 – first stable release 2012

CVC3 to CVC4

- CVC3 was very featureful…
  - support for many theories, proofs, quantifiers…

- But also suffered from serious problems
  - performance was problematic

QF_UF (100%)

Solver	Score	Time
MathSAT 5	200 / 205	9056.2 s
OpenSMT-1.0-alpha	186 / 205	7290.7 s
veriT 201007	175 / 205	36059.4 s
CVC3 2.3		
Yices 2 pr		

QF_RDL (100%)

Solver	Score	Time
OpenSMT-1.0-alpha	157 / 170	9475.5 s
veriT 201007	137 / 170	9932.2 s
CVC3 2.3	68 / 170	10386.8 s
Z3.2, 2008		
Yices 2 pr		

QF_IDL (100%)

Solver	Score	Time
OpenSMT-1.0-alpha	180 / 207	5880.7 s
veriT 201007	163 / 207	11624.2 s
CVC3 2.3	140 / 207	6930.4 s
Barcelogic		
Sateen-3.		

QF_BV (100%)

Solver	Score	Time
simplifyingSTP	155 / 205	8415.3 s
SONOLAR r252	140 / 205	14033.6 s
CVC3 2.3	68 / 205	7807.8 s
MathSAT 4.3, 2009 winner	171 / 205	3638.1 s

CVC3 to CVC4

- CVC3 was very featureful…
  - support for many theories, proofs, quantifiers…

- But also suffered from serious problems
  - performance was problematic
  - very difficult to extend for research
  - could not rapidly prototype new ideas

CVC4

- Complete redesign of internal architecture
- Five years in the making
- Performance a big improvement
  - placed 1st in 14 of 32 divisions of SMT-COMP
  - performs well also in CASC
  - competitive for many common SMT uses

- …without sacrificing features

CVC4 is Expressive

- Boolean combinations of theory constraints
- Combination of theories
  - arrays of integers, functions on arrays, …
- Quantifiers
- Verification, test generation, synthesis, feasibility
- Models, proofs, unsatisfiable cores

CVC4 is Expressive

- (Linear) arithmetic over integer, rational
- Bitvectors
- Strings

- Functions
- Arrays
- Inductive datatypes
- Finite sets

CVC4 is Expressive

- Quantifiers

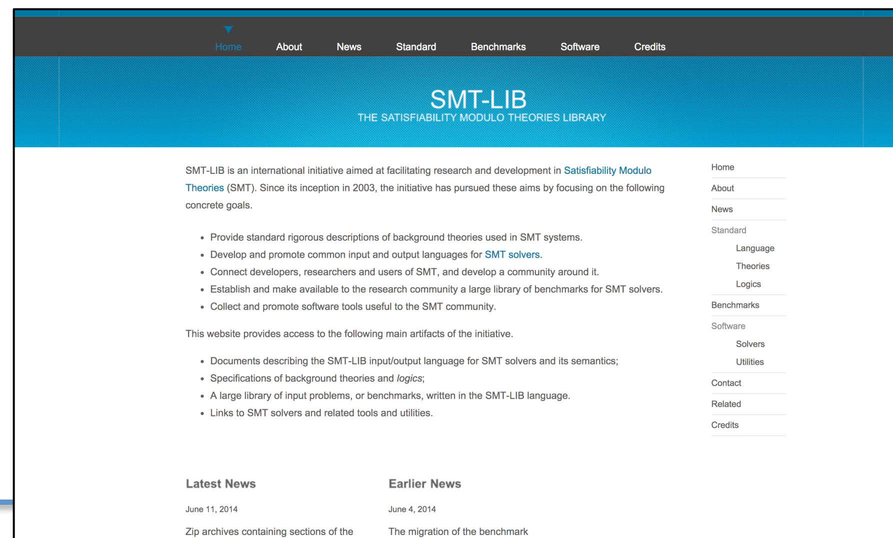
- If CVC4 doesn't have support for a theory,
  - axiomatize it

Standardization

- Fully supports SMT-LIB standard
  - v1.2, v2.0, v2.5 (draft)
  - supports much of Z3's extended command set
- Supports native CVC format
- Supports TPTP format

SMT-LIB – http://smt-lib.org

- International initiative
- Rigorously standardize descriptions of background theories for SMT
- Promote common syntax for SMT interactions
- Benchmarks
- Annual competition



SMT-LIB Command Language

- Declaring a logic

```
(set-logic QF_UF)
```

- Setting an option

```
(set-option :produce-models true)
```

- Declaring constants

```
(declare-fun p () Bool)
```

- Making assertions

```
(assert (or p q))
```

SMT-LIB Command Language

- Checking satisfiability

```
(check-sat)
```

- Extracting a model

```
(get-model)
```

SMT-LIB example

New and Upcoming Features

- Theory of strings
- Theory of finite sets
- Theory of floating point

- Unsatisfiable cores (for all theories)
- Proofs (under development, for some theories)

- Better control of preprocessing

Longer term

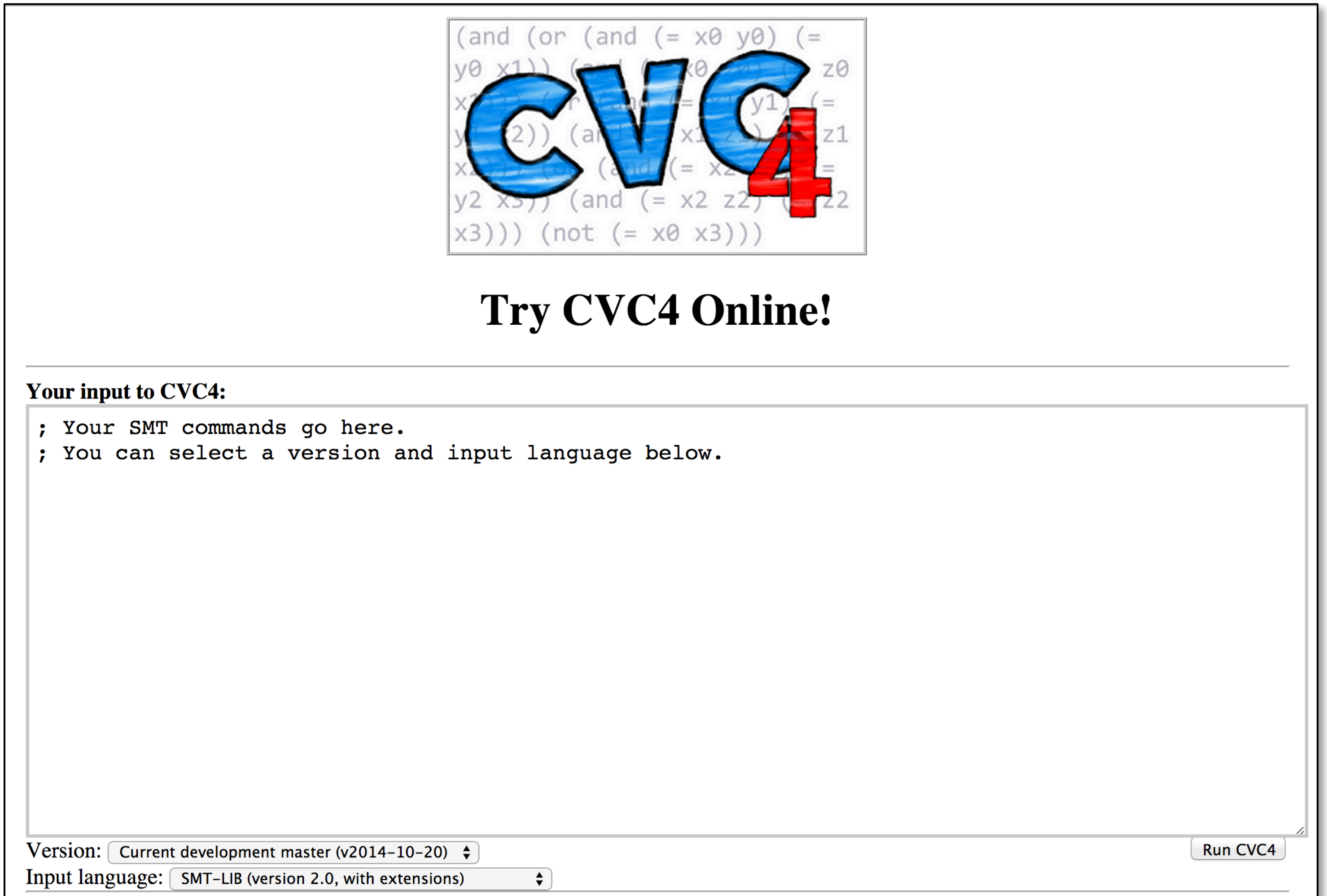
- More theories
- Increased proof support
- Automatic configuration of heuristics
- Quantifier elimination
- Optimization problems

Certificates

- *Satisfiable* comes with a satisfying model
- *Unsatisfiable* comes with a proof (or core)

- Both are fully machine-checkable
  - CVC4 need not be certified free of bugs to rely upon a result

http://cvc4.cs.nyu.edu/tryit/



Circuit example

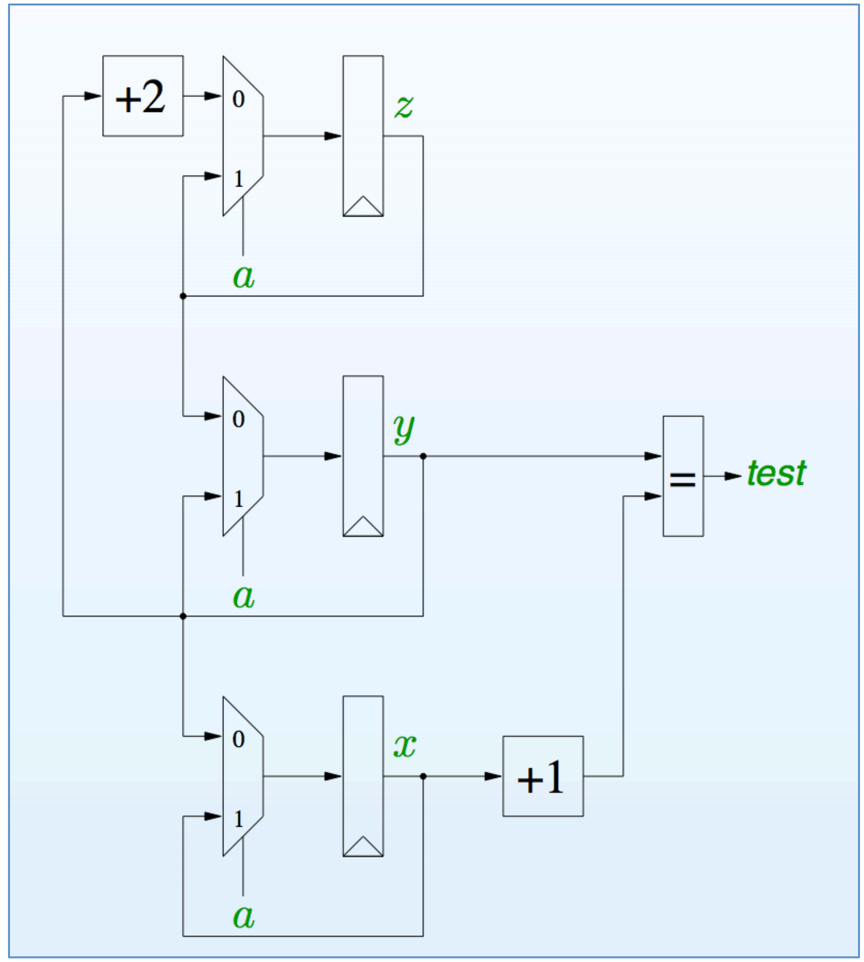
test is always supposed to be true

When does it hold?

How do we prove it?

One way: by induction on number of clock cycles

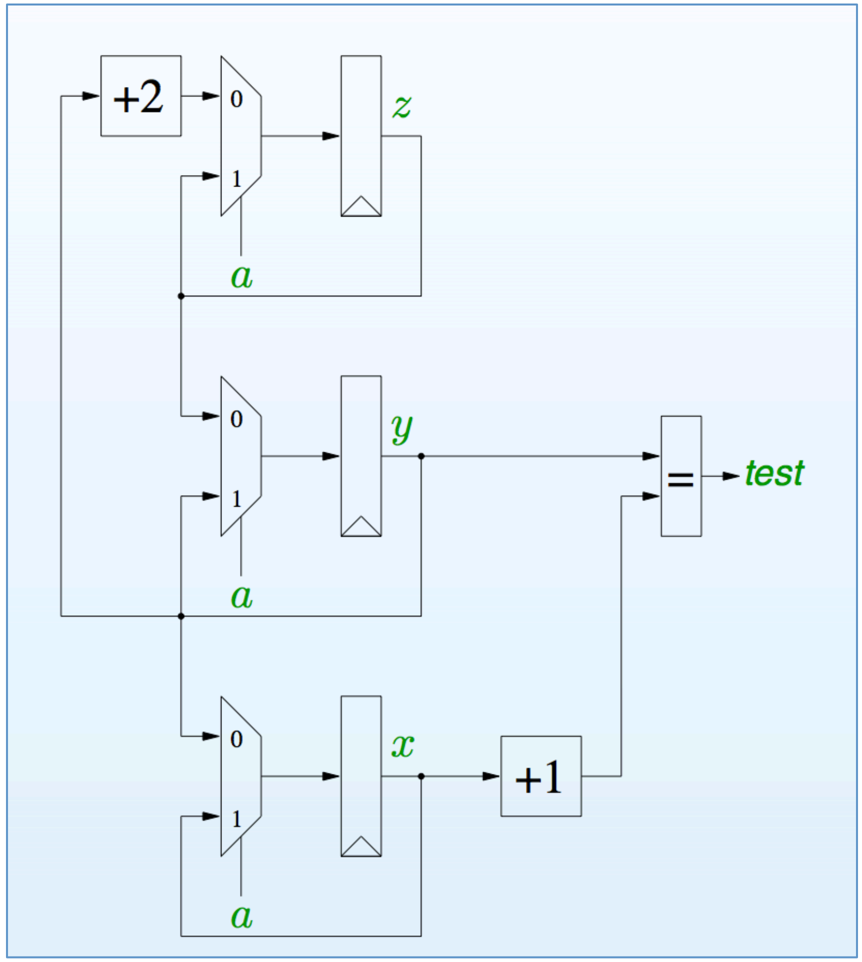
Inductive step:
If test is true, it remains so



Circuit example

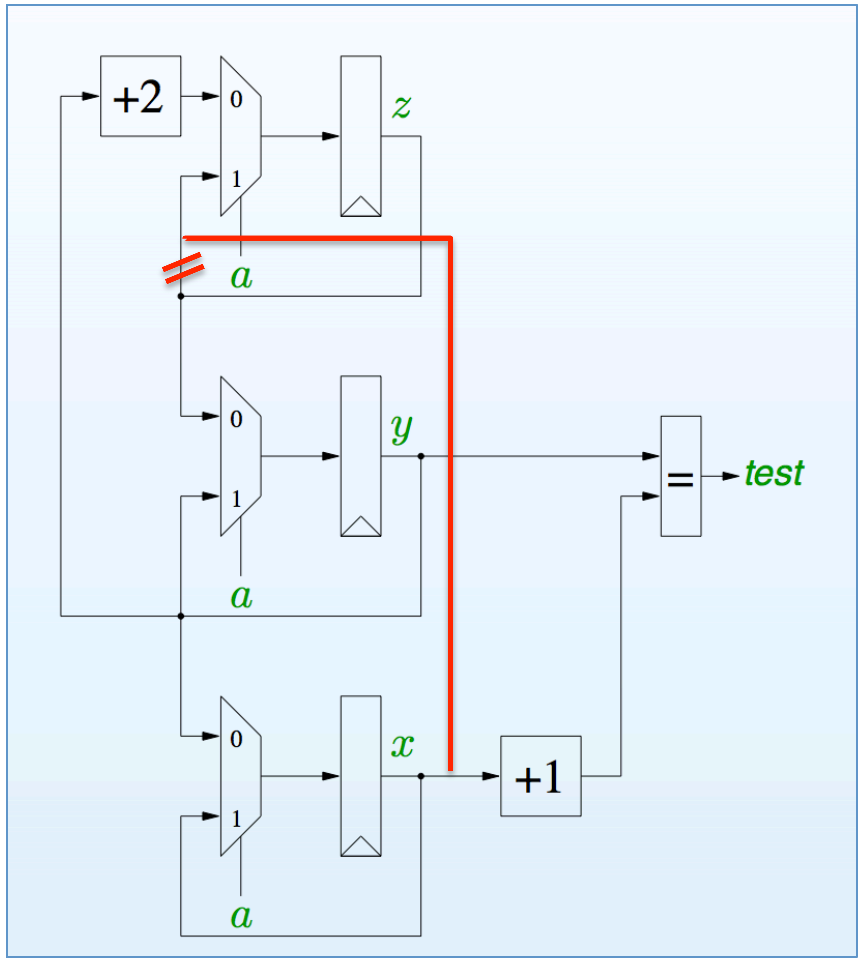
```
(y = x + 1 AND z = x + 2 AND
 x' = IF a THEN x ELSE y AND
 y' = IF a THEN y ELSE z AND
 z' = IF a THEN z ELSE y + 2) IMPLIES
 y' = x' + 1 AND z' = x' + 2
```

$$(z1 \leftrightarrow \neg x1) \wedge (z0 \leftrightarrow x0) \wedge$$
$$(y1 \leftrightarrow (x1 \oplus x0)) \wedge (y0 \leftrightarrow \neg x0) \wedge$$
$$(a \longrightarrow ((xp1 \leftrightarrow x1) \wedge (xp0 \leftrightarrow x0))) \wedge$$
$$(\neg a \longrightarrow ((xp1 \leftrightarrow y1) \wedge (xp0 \leftrightarrow y0))) \wedge$$
$$(a \longrightarrow ((yp1 \leftrightarrow y1) \wedge (yp0 \leftrightarrow y0))) \wedge$$
$$(\neg a \longrightarrow ((yp1 \leftrightarrow z1) \wedge (yp0 \leftrightarrow z0))) \wedge$$
$$(a \longrightarrow ((zp1 \leftrightarrow z1) \wedge (zp0 \leftrightarrow z0))) \wedge$$
$$(\neg a \longrightarrow ((zp1 \leftrightarrow \neg y1) \wedge (zp0 \leftrightarrow y0))) \wedge$$
$$(\neg(zp1 \leftrightarrow \neg xp1) \vee \neg(zp0 \leftrightarrow xp0) \vee$$
$$\neg(yp1 \leftrightarrow (xp1 \oplus xp0)) \wedge (yp0 \leftrightarrow \neg xp0)$$



Circuit example

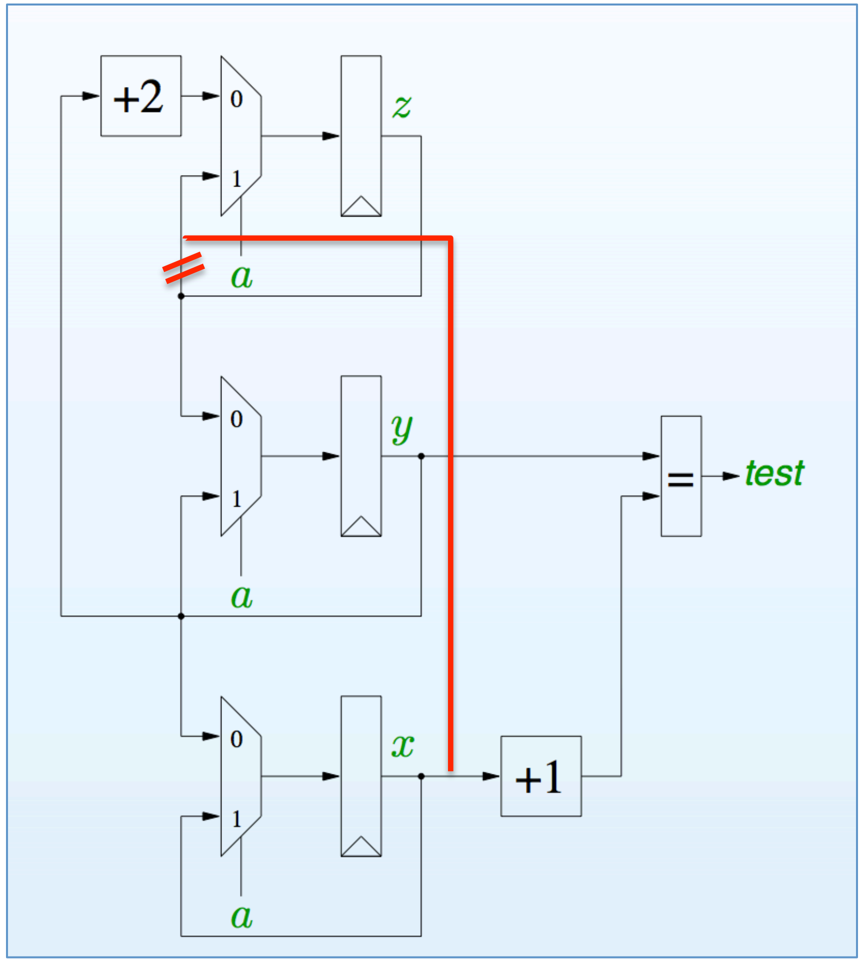
```
(y = x + 1 AND z = x + 2 AND
 x' = IF a THEN x ELSE y AND
 y' = IF a THEN y ELSE z AND
 z' = IF a THEN z ELSE y + 2) IMPLIES
 y' = x' + 1 AND z' = x' + 2
```



Circuit example

```
(y = x + 1 AND z = x + 2 AND
 x' = IF a THEN x ELSE y AND
 y' = IF a THEN y ELSE z AND
 z' = IF a THEN z ELSE y + 2) IMPLIES
 y' = x' + 1 AND z' = x' + 2
```

```
(model
(define-fun x () Int (- 2))
(define-fun y () Int (- 1))
(define-fun z () Int 0)
(define-fun |x'| () Int (- 2))
(define-fun |y'| () Int (- 1))
(define-fun |z'| () Int (- 2))
(define-fun a () Bool true)
)
```



Arithmetic

Arithmetic in CVC4

- Quantifier-free linear real and integer arithmetic

  QF_LRA, QF_LIA, QF_LIRA

- Constraints of the form:

  $x - y \geq -1$, $y \leq 4$, $x \mathrel{!}= 5$, $x + y \geq 6$, $x < 5$ …

- Supports efficient theory combination:

  UF, Arrays, Sets, Datatypes

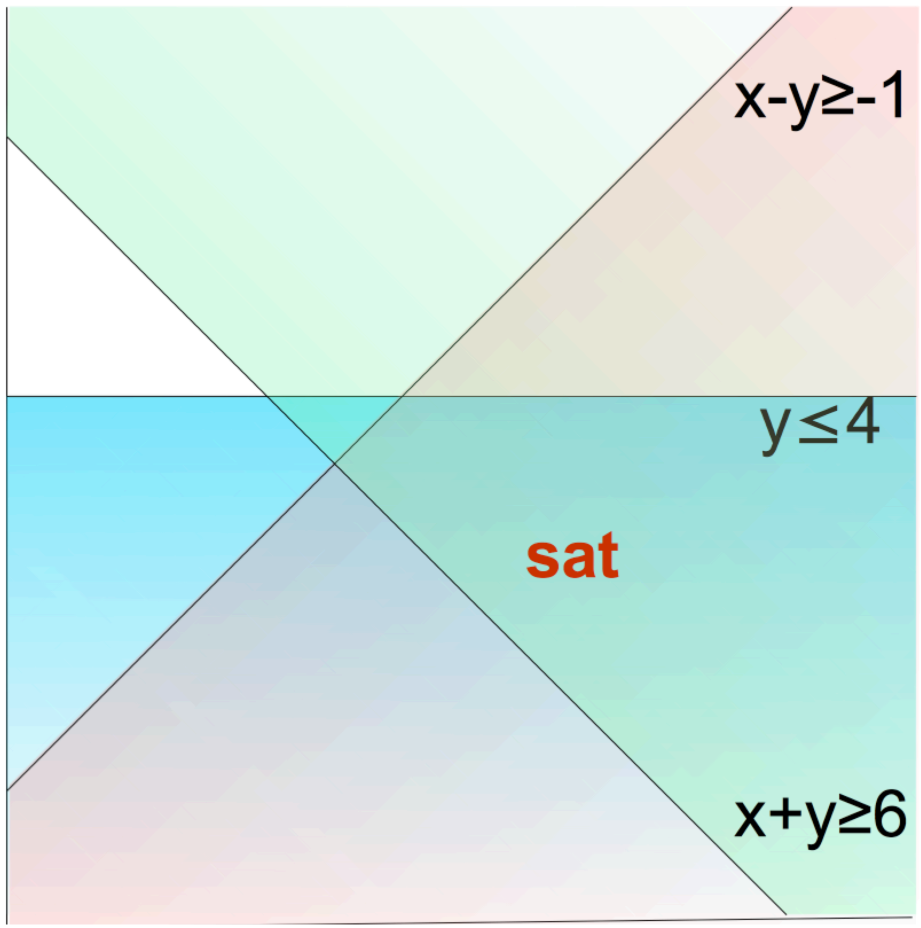
Linear Real Arithmetic

- Given the linear inequalities

  $\{x - y \geq -1, y \leq 4, x + y \geq 6\}$

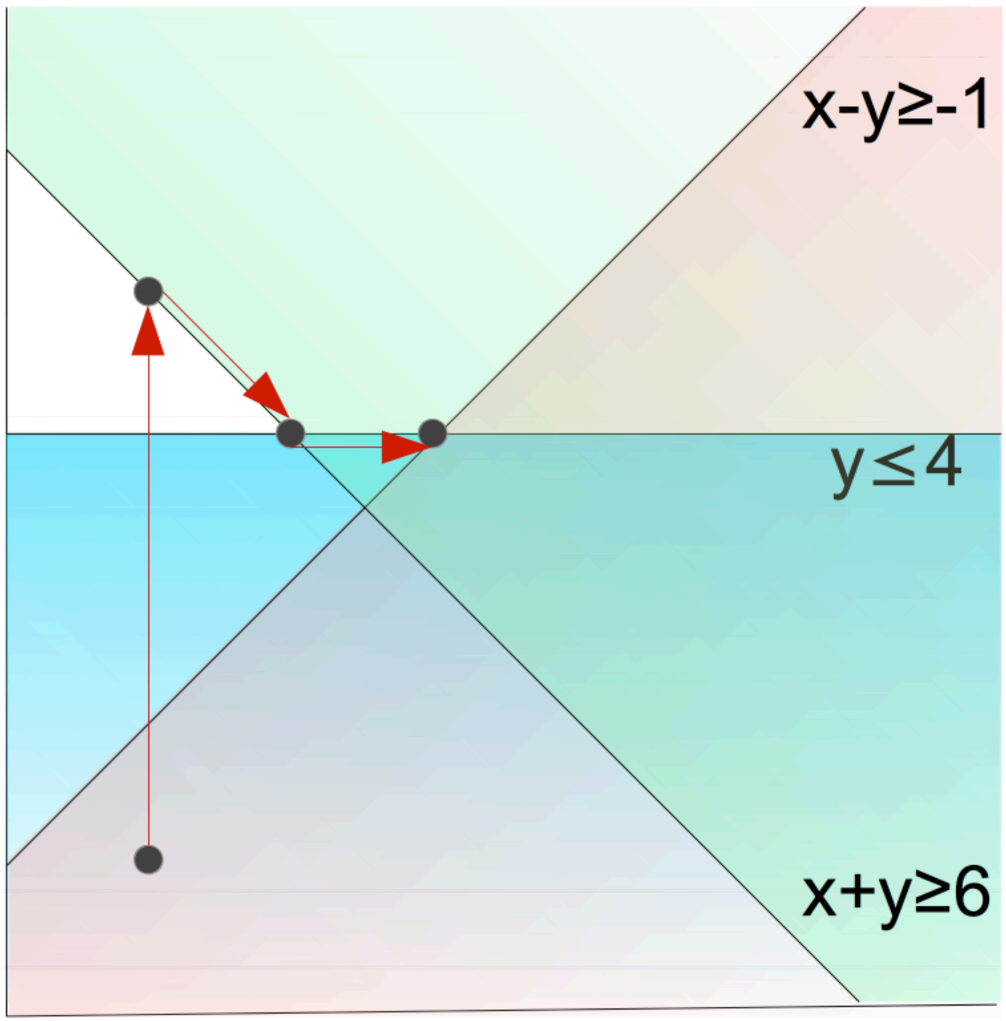
  is there an assignment to x and y that makes all of the inequalities true?
- Solve using simplex based approaches

Visually



Is an intersection of half planes empty?

Example Simplex Search



Simplex Solvers in CVC4

- 3 exact precision DPs
  - Simplex for DPLL(T)
  - Sum-Of-Infeasibilities (SOI) Simplex [FMCAD'13]
  - FCSimplex (variant of SOI simplex)
- External floating point solver GLPK

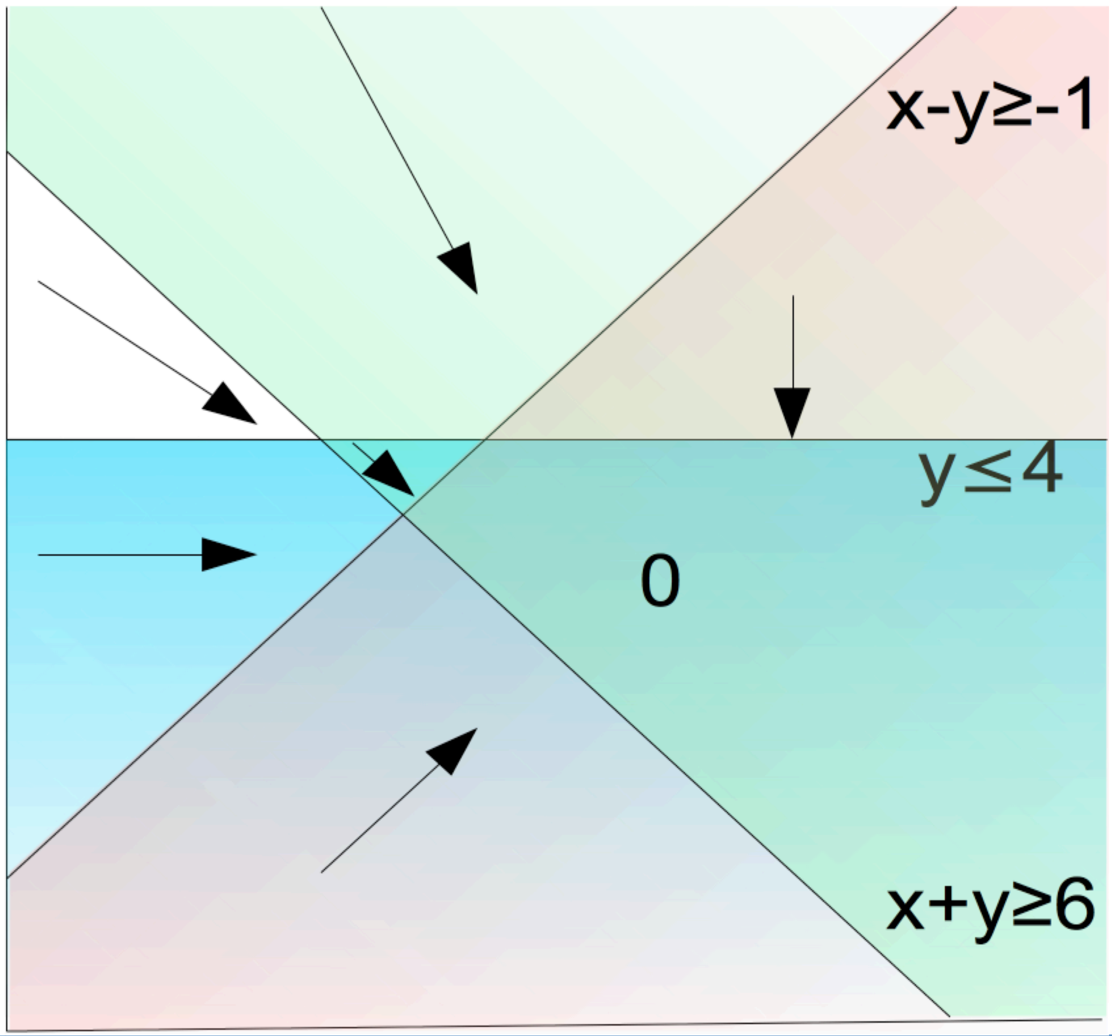
Simplex for DPLL(T)

- [CAV'06 Dutertre & de Moura]
- Highly incremental
  - Resumes from previous assignment
  - No backtracking the tableau
- Run after each new constraint
- Good on verification problems

Sum-of-infeasibilities Simplex

- Adds an optimization function V(X)
- V(X) is the total amount bounds are violated
- Minimizes V(X) using primal simplex
- "--use-soi"
- Heavy hammer

Direction of Vio(X)



Leveraging LP & MIP for SMT

- GPLK is an LP/MIP Solver
- ‘Call GLPK if the problem seems hard’
- “--use-approx”
- Very heavy hammer
- See my talk Thursday
- Compile against: https://github.com/timothy-king/glpk-cut-log

From Reals to Mixed Integers

- Add IsInt(x) constraints
- First solve real relaxation
  - Ignore IsInt(x) constraints
- If real relaxation is sat:
  - check if assignment from Simplex a(x) satisfies IsInt(x) constraints
  - Refine by branching:

    $x \geq 2$ or $x \leq 1$

    Cuts from Proofs [Dillig'06]

ITE Preprocessing

- ITE cofactoring [Kim et al. '09]
- Lifting sums out of ites
  (ite c (+ x s) (+ x t)) →x + (ite c s t)
- GCD factorization
  (ite c 0 (ite d 1024 2048)) →
  1024 * (ite c 0 (ite d 1 2))

Non-linear support in CVC4

- Extremely rudimentary support
  - Parsing, rewriting, solving
  - No models
- Rewrites terms into sum of monomials form
  (x+y)(x-y)→ x*x - y*y
- Abstracts each monomial as a fresh variable
  - x*x, x*y, x*x*y are all new variables
- Usable if you instantiate axioms manually
- (may improve in the future)

Optimization

- (Coming soon)
- Primal Simplex is implemented
- Not yet user accessible
- Will implement
  - Linear Search [Sebastiani '12]
  - Symba [Li et al. '14]

CVC4's Arithmetic Module

- Optimized for challenging QF_LRA and QF_LIA non-incremental benchmarks
- On the lookout for collaborations to motivate improvements
- For hard linear problems, try: "--use-soi" or "--use-approx"

Quantifiers

Quantified Formulas in CVC4

- CVC4 supports multiple techniques :
  - E-matching
  - Conflict-based instantiation [FMCAD 2014]
  - Rewrite rules
  - Induction
  - Finite Model Finding

    $\Rightarrow$ *Focus in this tutorial*

Overview : Finite Model Finding

- Finite Model Finding in SMT
  - Reduction from quantified → ground constraints
- Two techniques for scalability:
  - Minimizing model sizes
  - Model-based quantifier instantiation
- Variants of approach/Examples

Finite Model Finding in SMT

- To determine the satisfiability of:

$$G \quad \cup \quad \forall xy{:}S.\, Q(x,y)$$

For all x, y of sort S

Finite Model Finding in SMT

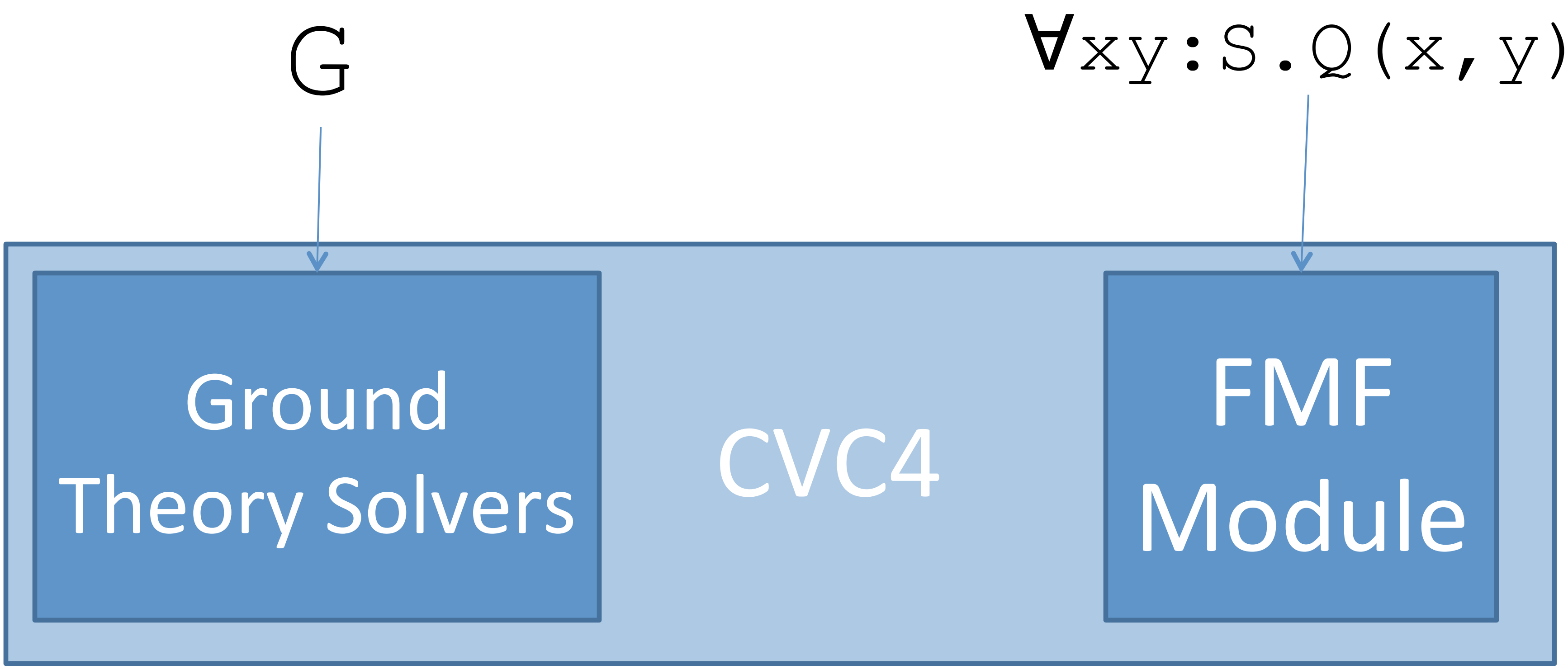
- To determine the satisfiability of:

$$G \quad \cup \quad \forall xy{:}S.\,Q(x,y)$$

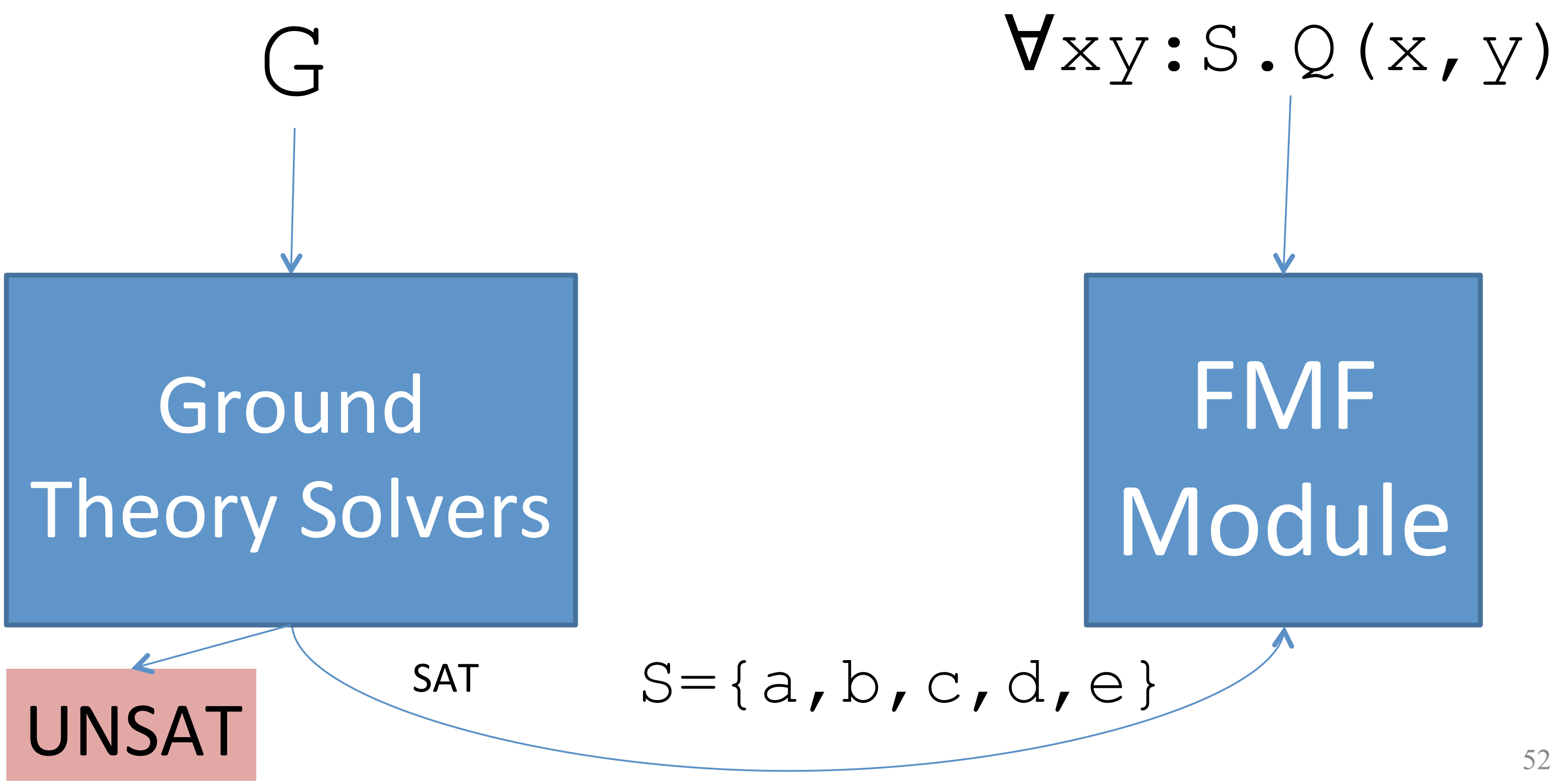
For all x, y of sort S

⇒ *If S has finite interpretation,*
- *use finite model finding*

Finite Model Finding in SMT

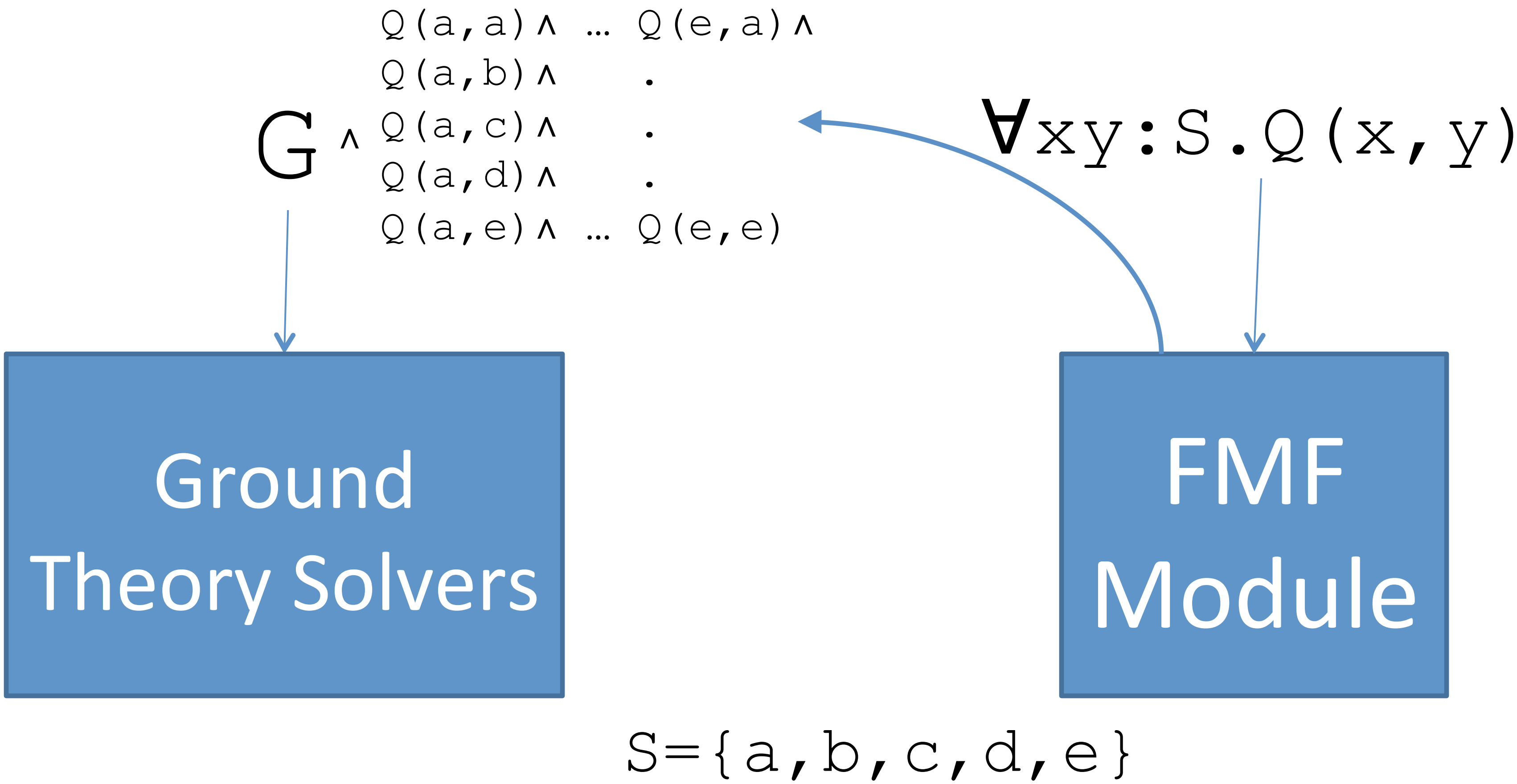


Finite Model Finding in SMT



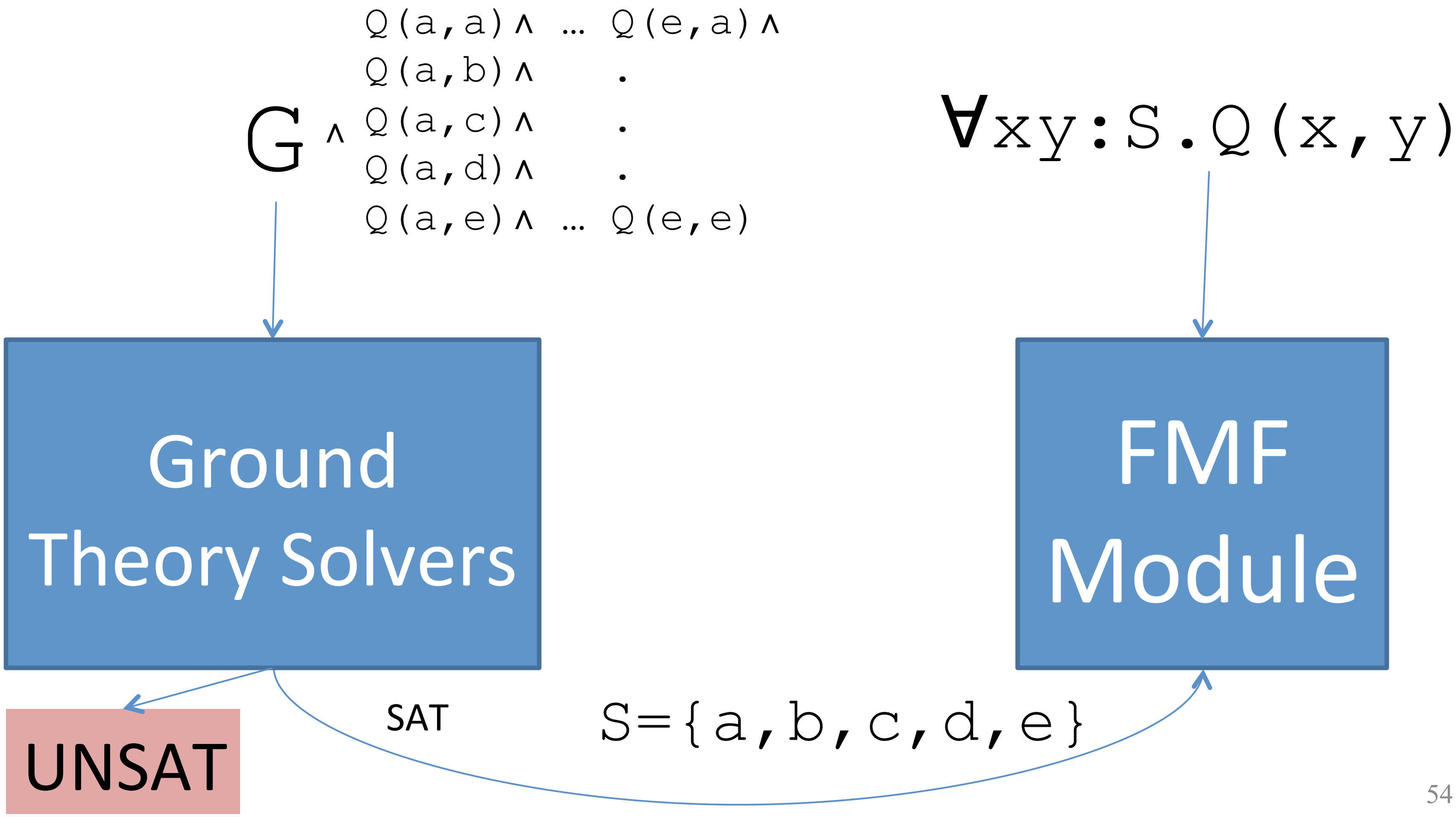
Finite Model Finding in SMT

- Reduce quantified formula to ground formula(s)

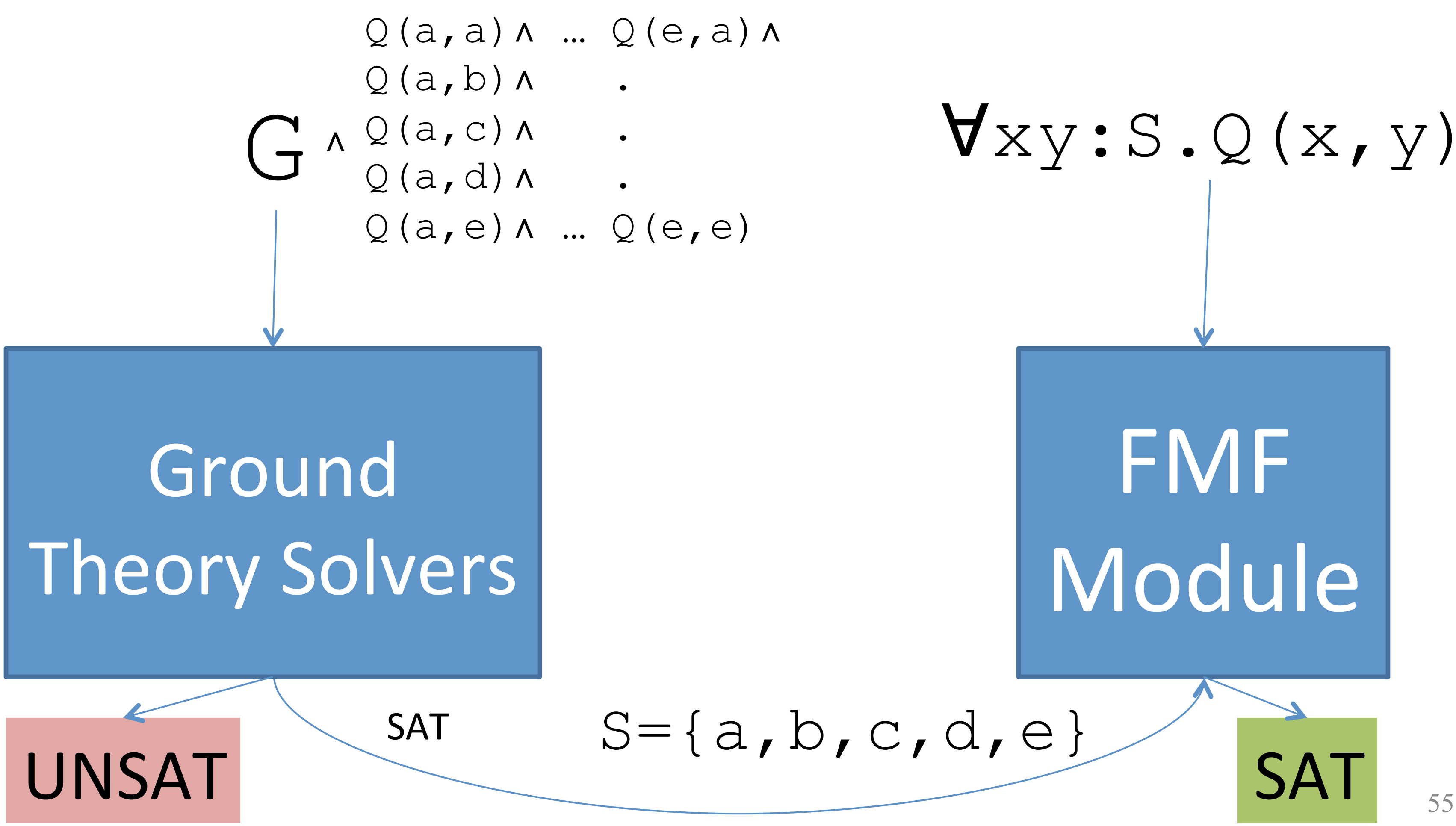


Finite Model Finding in SMT

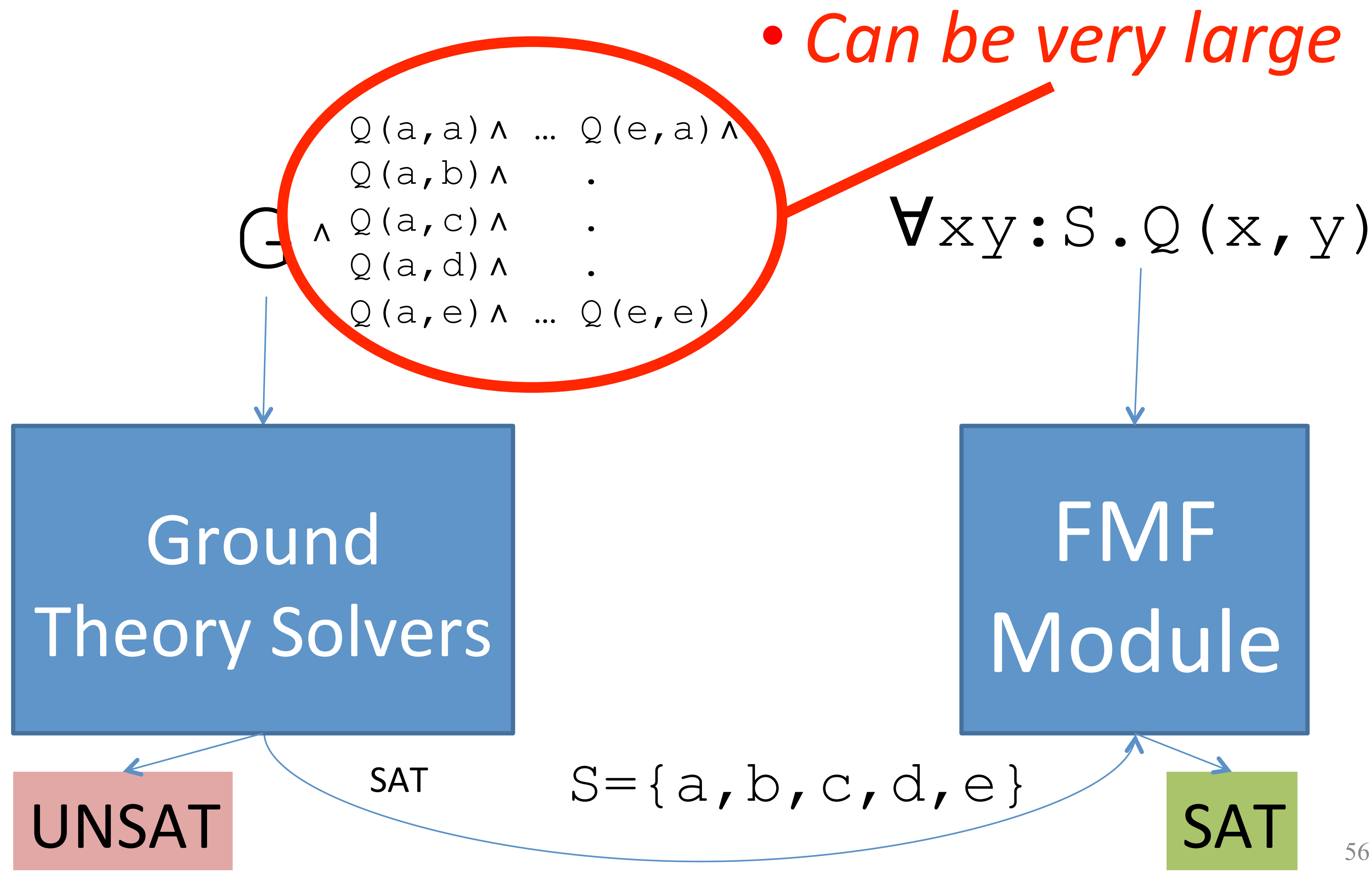
- Check satisfiability of G∧Q(a,a)∧…∧Q(e,e)



Finite Model Finding in SMT



Finite Model Finding in SMT

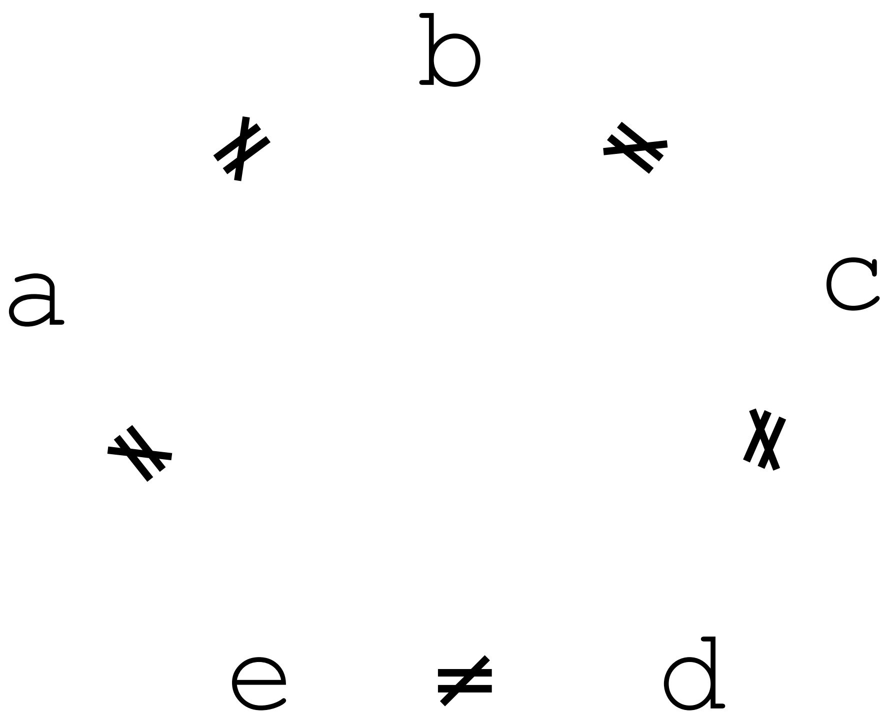


Scalable Approach to FMF

- Address large # instantiations by:
  1. Minimizing model sizes
  2. Only consider instantiations that refine model
     - Model-based quantifier instantiation
       – [Ge/deMoura CAV 2009]

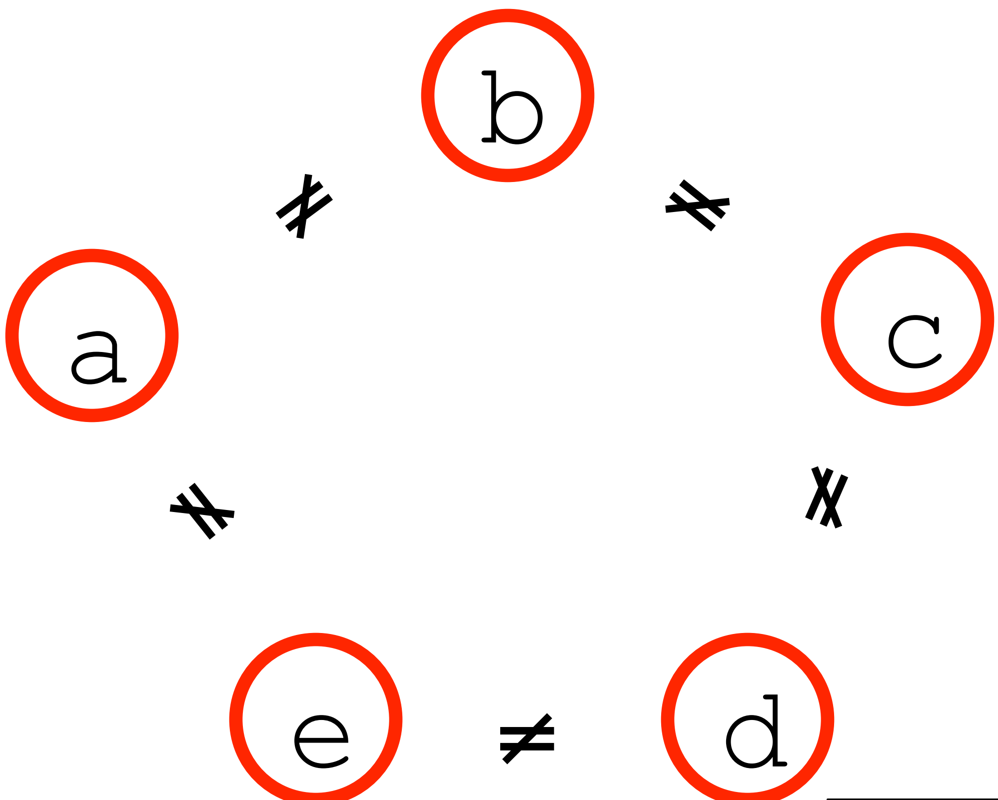
1. Minimize model size (Example)

- For `G={a≠b,b≠c,c≠d,d≠e,e≠a}`



1. Minimize model size (Example)

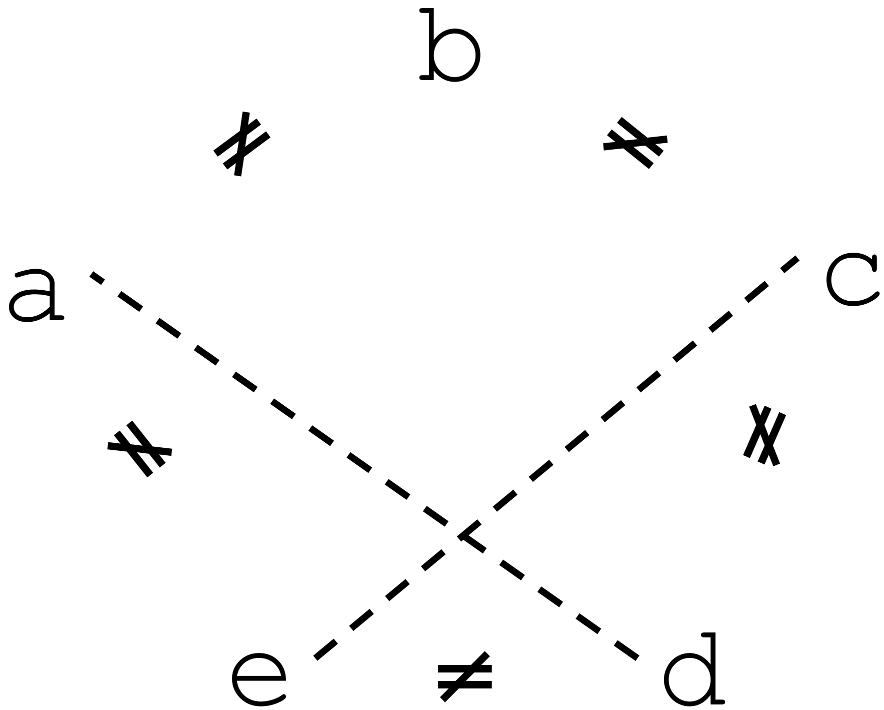
- Has model of size 5:



$\Rightarrow$ S={a,b,c,d,e}

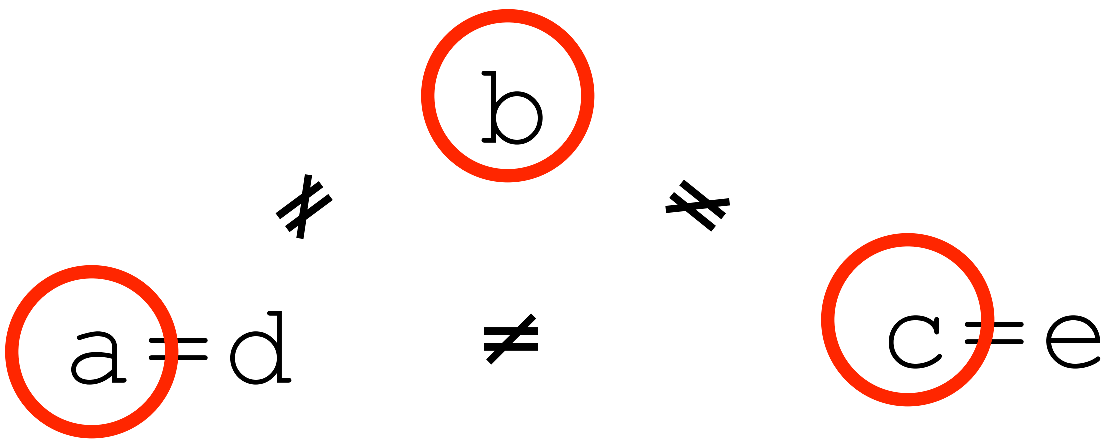
1. Minimize model size (Example)

- Can identify `a=d, c=e`



1. Minimize model size (Example)

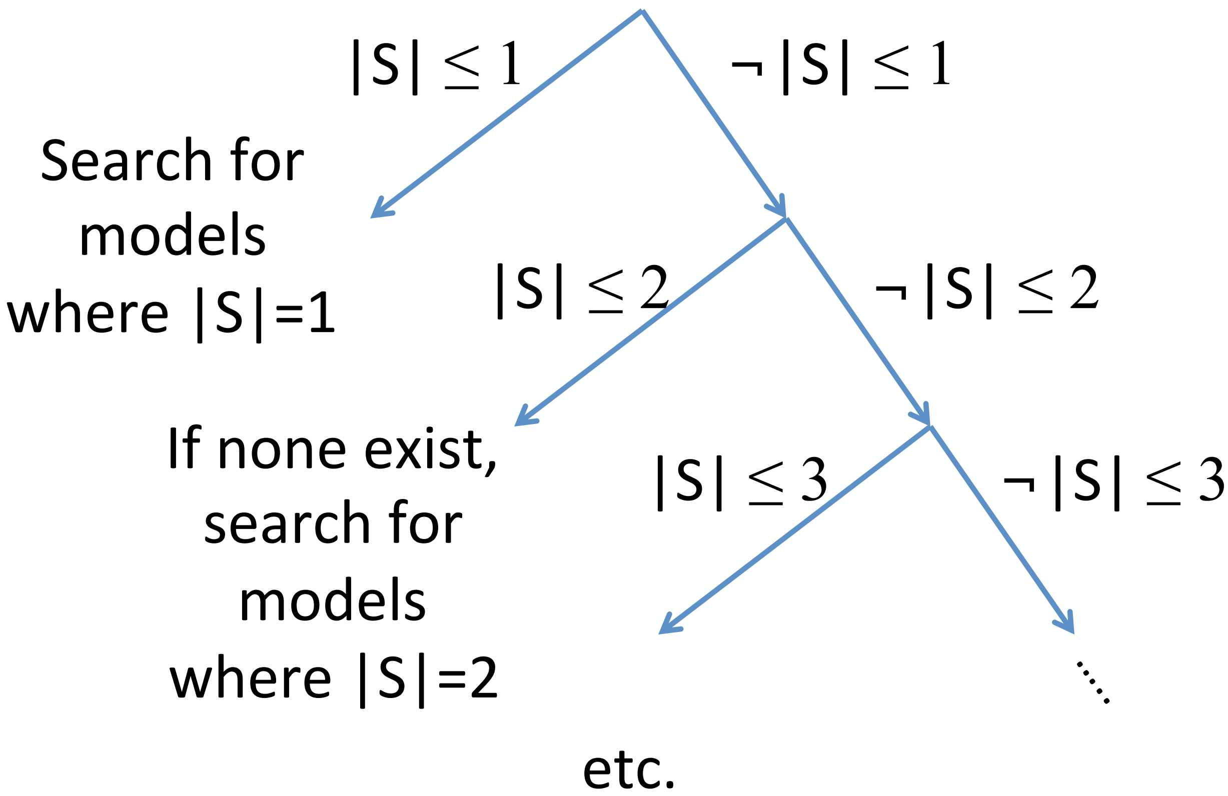
- Also has model of size 3:



$\Rightarrow$ S={a,b,c}

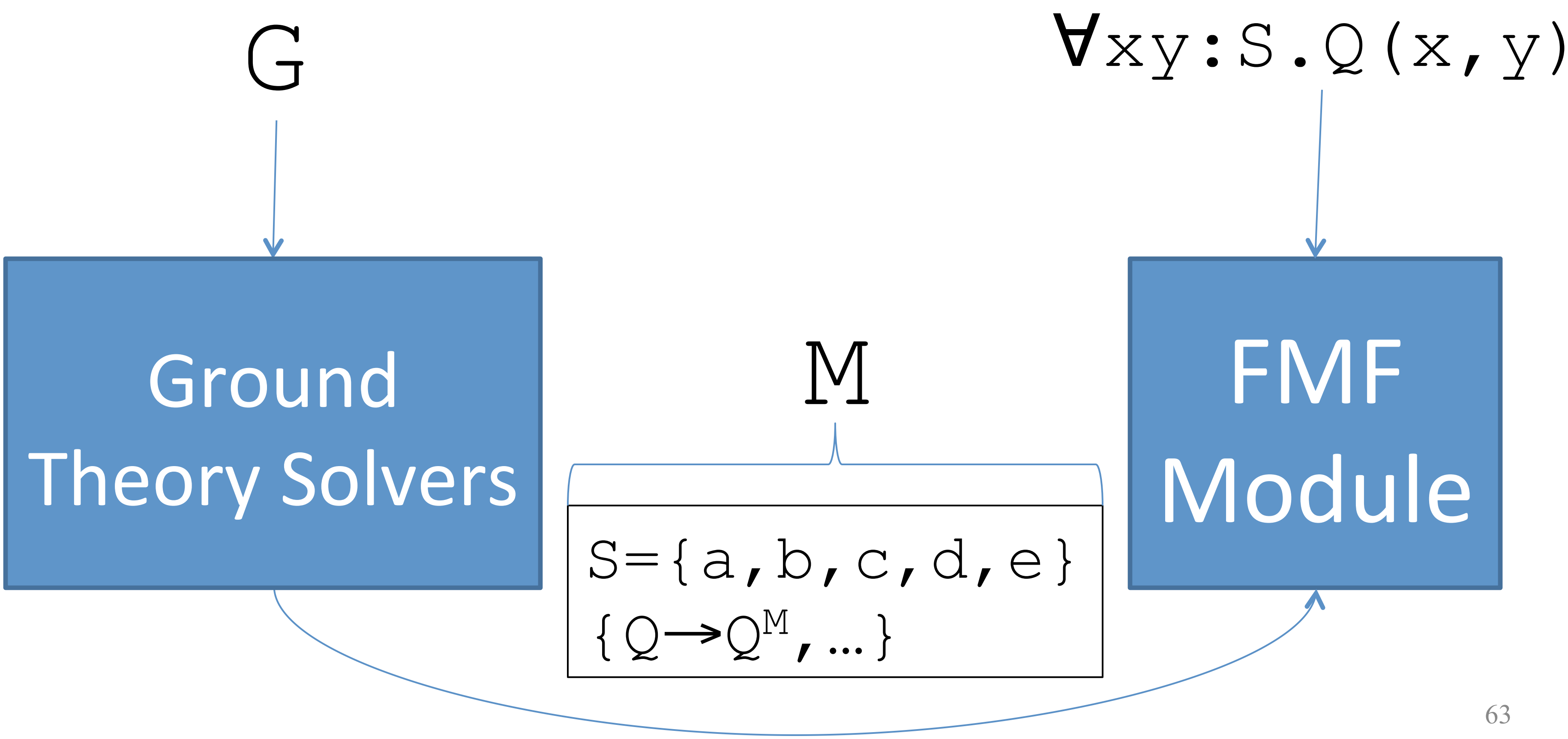
1. Minimize model size

- UF+cardinality constraints [CAV 2013]



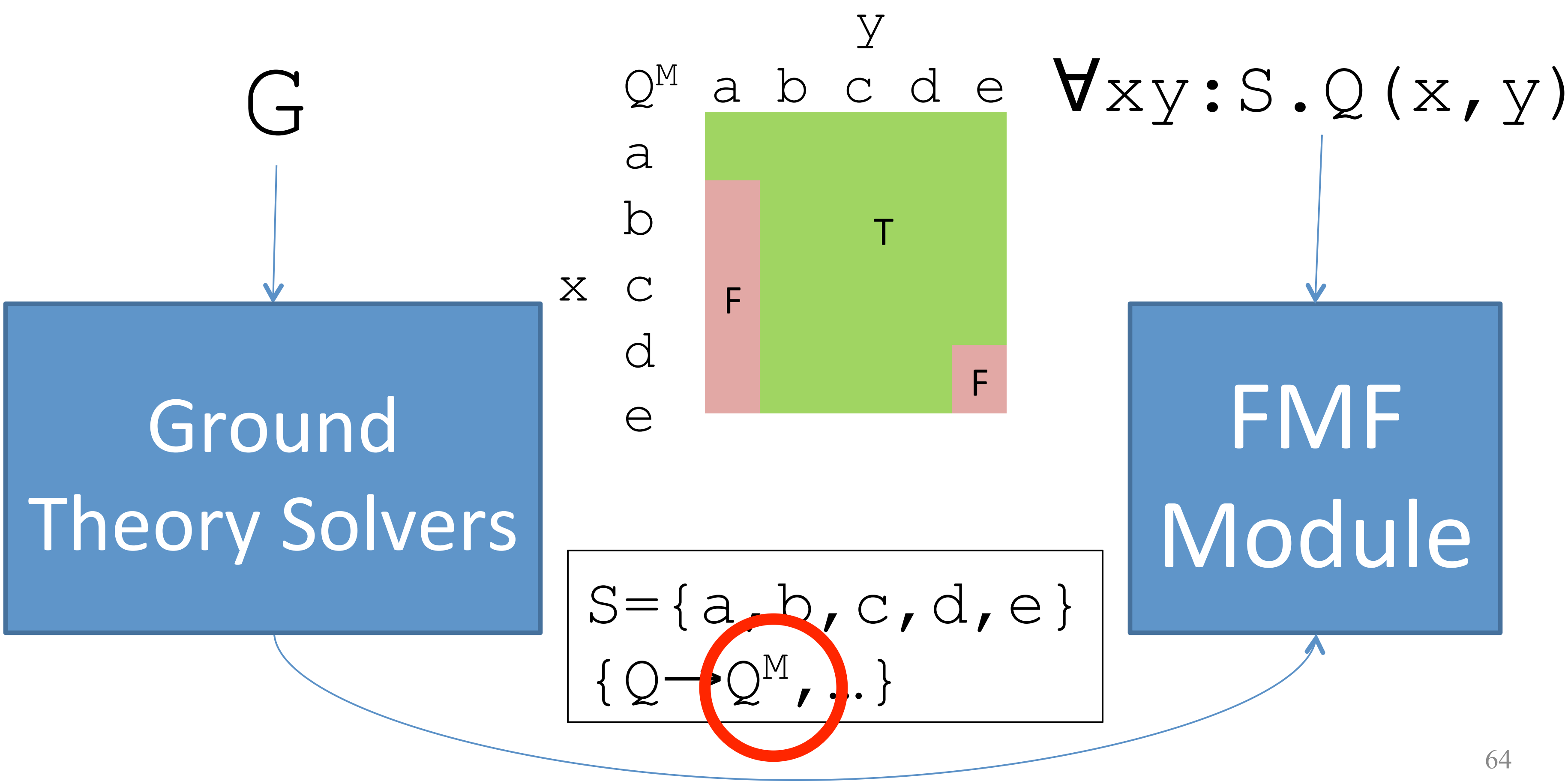
2. Model-Based Quantifier Instantiation

- Construct *candidate model* `M`



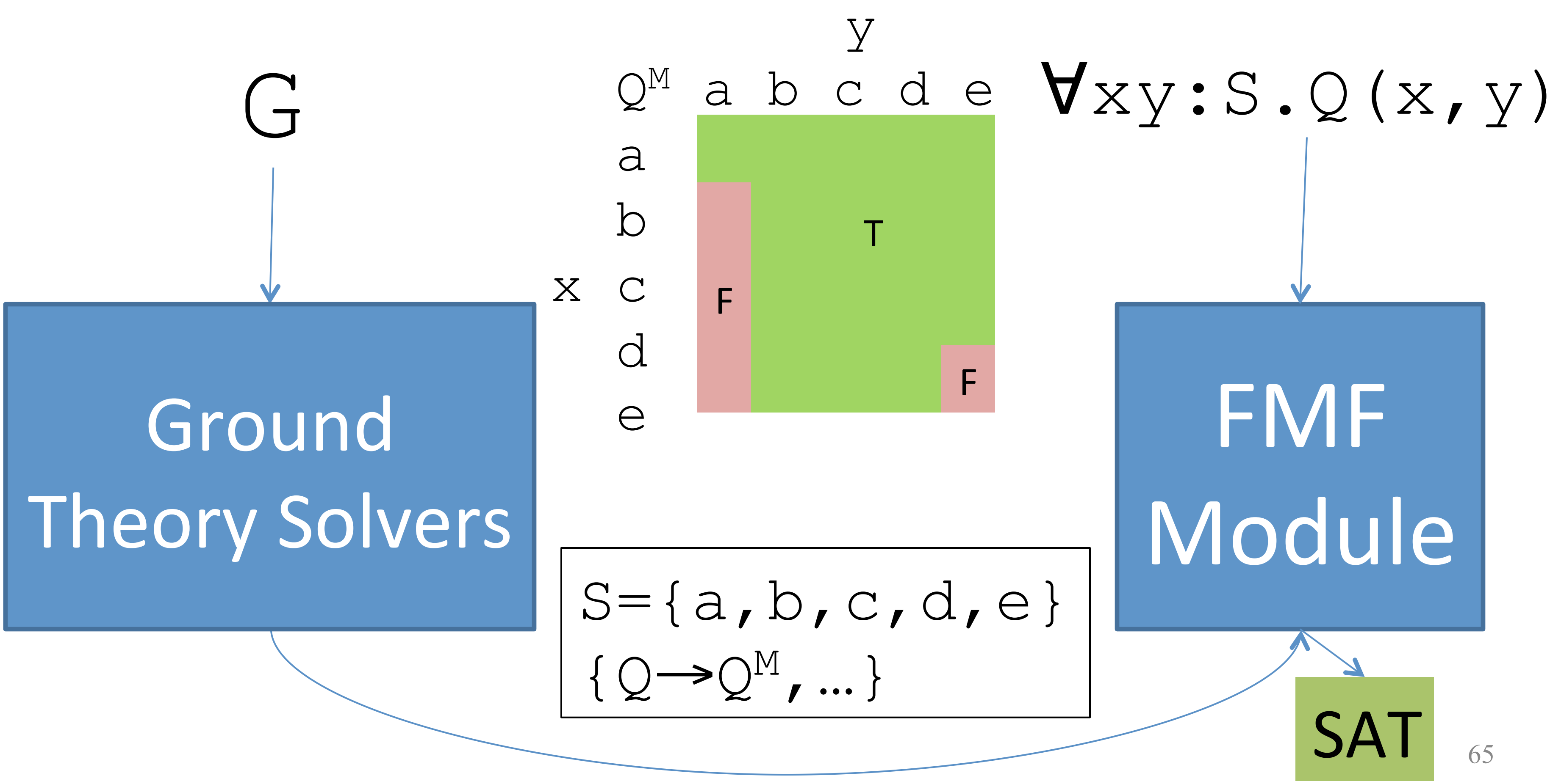
2. Model-Based Quantifier Instantiation

- Using model, evaluate $Q^M(x, y)$ [CADE 2013]



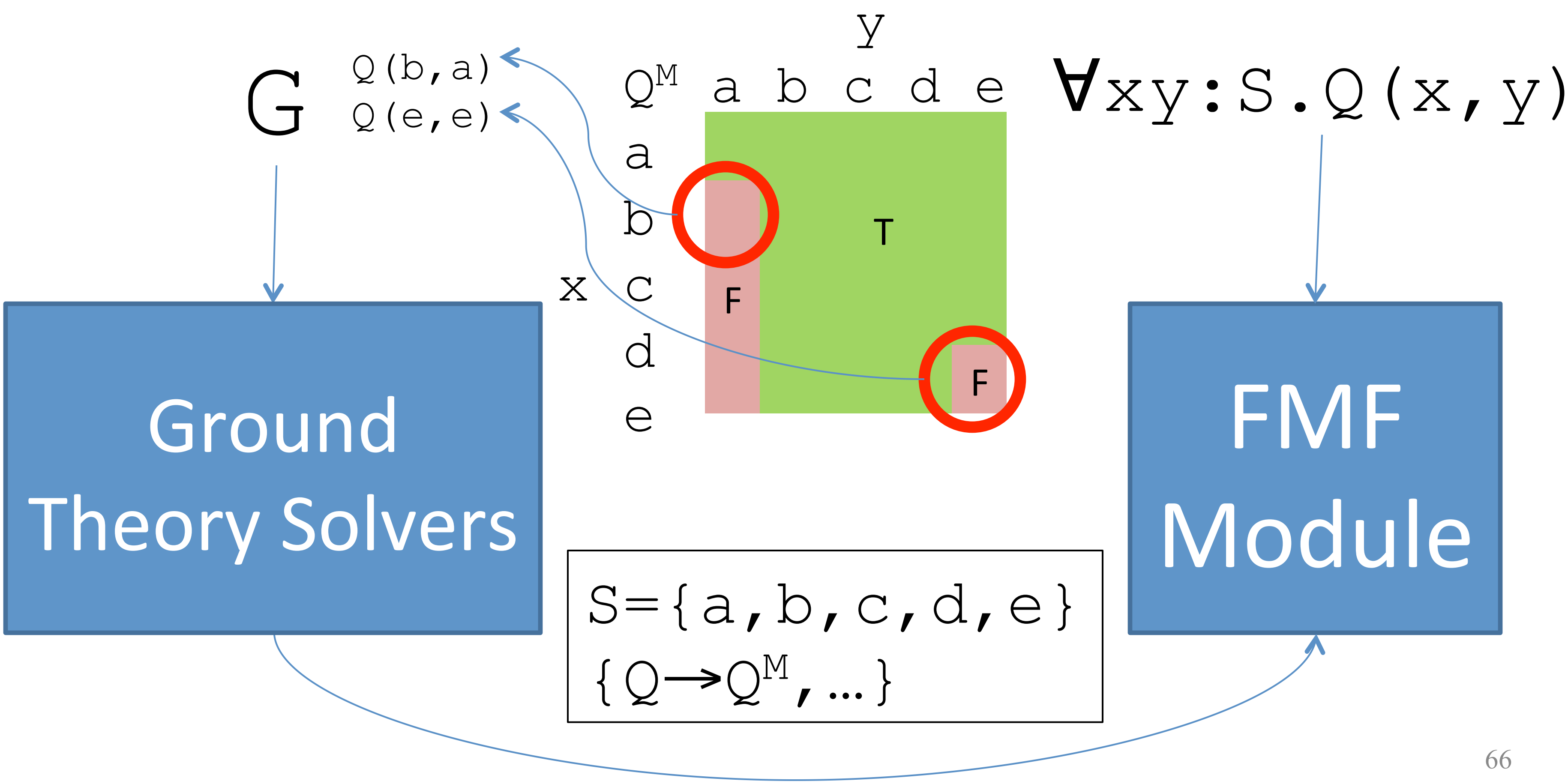
2. Model-Based Quantifier Instantiation

- If all T instances, M is a model, answer “SAT”



2. Model-Based Quantifier Instantiation

- If all T instances, M is a model, answer "SAT"
- Else, add (subset of) F instances to G



Application : DVF at Intel

- CVC4+FMF used as a backend for:
  - DVF tool at Intel Research
    - Verifying software/hardware architectures

  ⇒ Required:
    - Scalable methods for finding counterexamples (FMF)
    - Support for many theories (arith, BV, arrays, …)

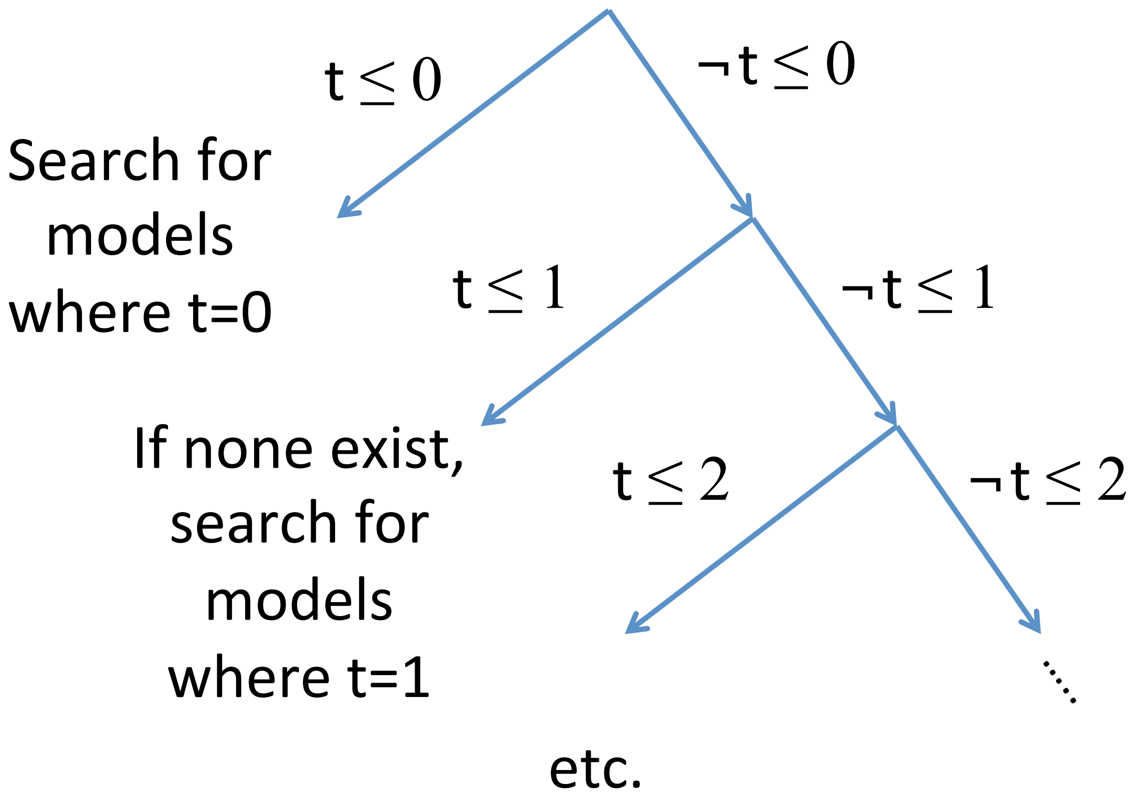
Variants of Approach

- CVC4 has techniques for handling:
  - Bounded Integer Quantification
  - Strings of bounded length
  - Syntax-Guided Synthesis (in progress)

$\Rightarrow$ *Each follow similar pattern*

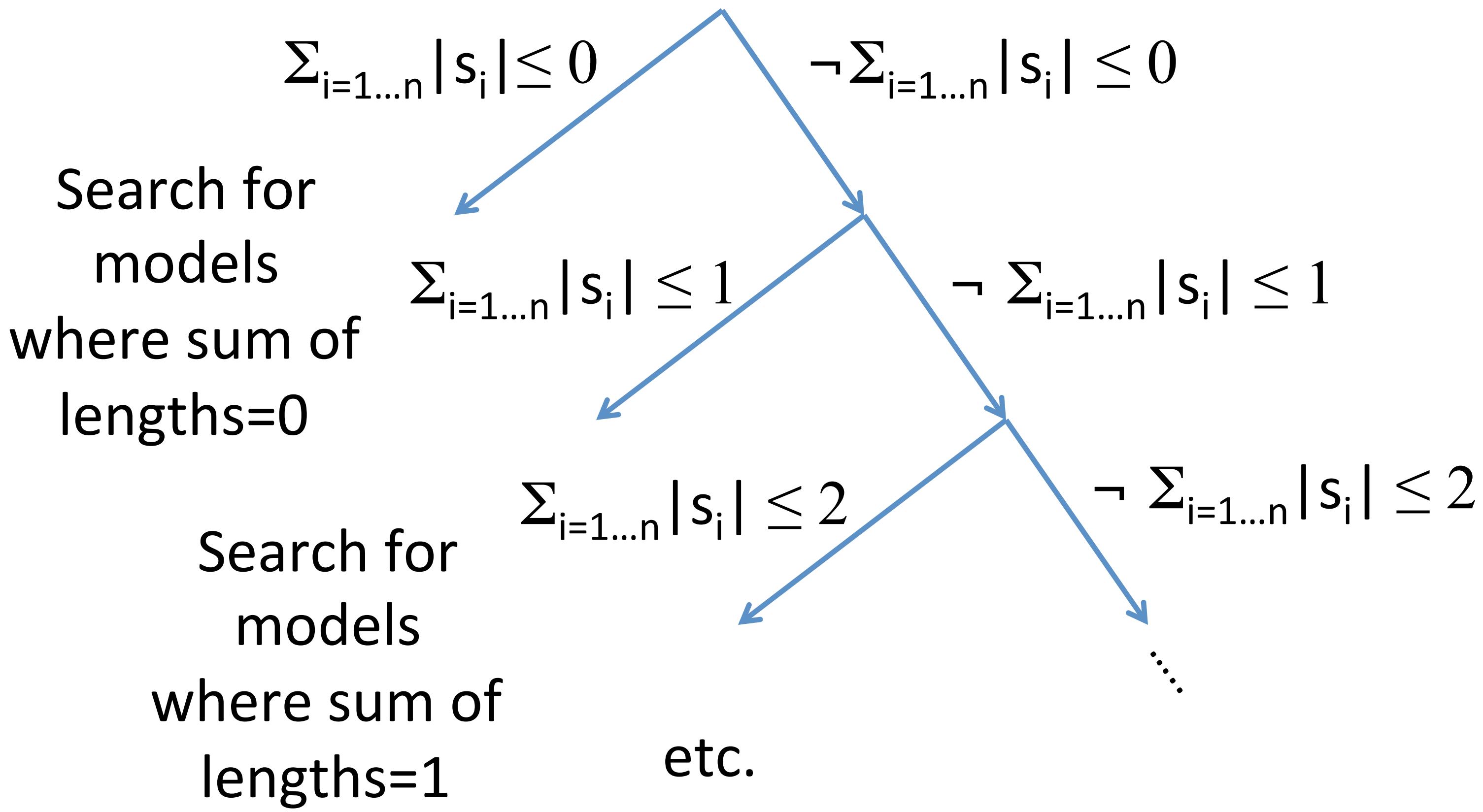
Variant #1 : Bounded Integers

- $\forall$`x:Int. 0 ≤ x < t ⇒P(x)`



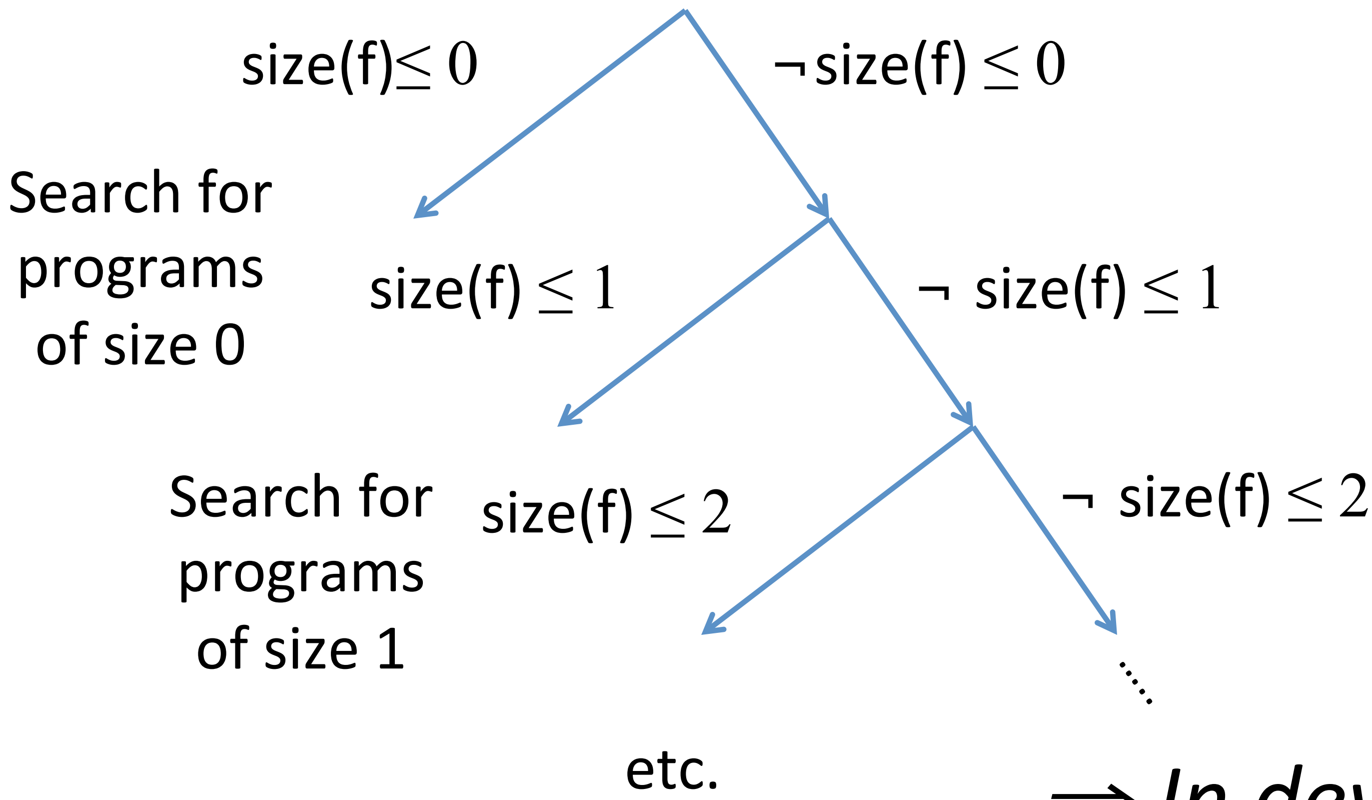
Variant #2 : Bounded Length Strings

- Given input $F[s_1,\ldots,s_n]$ for strings $s_1 \ldots s_n$:



Variant #3 : Syntax-Guided Synthesis

- `(¬)∃f:Program.∀i.P(f,i)`



⇒ *In development*

Wrap-up

Use in Research

Barrett, Conway, Deters, Hadarean, Jovanović, King, Reynolds, and Tinelli. **CVC4**. *Computer Aided Verification (CAV),* 2011.

Hadarean, Barrett, Jovanović, Tinelli, and Bansal. **A tale of two solvers: Eager and lazy approaches to bit-vectors**. *Computer Aided Verification (CAV)*, 2014.

Jovanović and Barrett. **Being careful about theory combination.** *Formal Methods in System Design (FMSD)*, vol. 42, 2013.

Jovanović and Barrett. **Sharing is caring: combination of theories**. *Frontiers of Combining Systems (FroCoS),* 2011.

Jovanović, Barrett, and de Moura. **The design and implementation of the model constructing satisfiability calculus**. *FMCAD*, 2013.

King and Barrett. **Exploring and categorizing error spaces using BMC and SMT**. *Satisfiability Modulo Theories (SMT),* 2011.

King, Barrett, and Dutertre. **Simplex with sum of infeasibilities for SMT**. *Formal Methods in Computer-Aided Design (FMCAD)*, 2013.

King, Barrett, and Tinelli. **Leveraging linear and mixed integer programming for SMT**. *FMCAD,* 2014.

Liang, Reynolds, Tinelli, Barrett, and Deters. **A DPLL(T) theory solver for a theory of strings and regular expressions**. *Computer Aided Verification (CAV)*, 2014.

Reynolds, Tinelli, and de Moura. **Finding conflicting instances of quantified formulas in SMT**. *FMCAD,* 2014.

Reynolds, Tinelli, Goel, and Krstić. **Finite model finding in SMT**. *Computer Aided Verification (CAV),* 2013.

Reynolds, Tinelli, Goel, Krstić, Deters, and Barrett. **Quantifier instantiation techniques for finite model finding in SMT**. *CADE,* 2013.

Wang, Barrett, and Wies. **Cascade 2.0**. *Verification, Model Checking, and Abstract Interpretation (VMCAI),* 2014.

This week at FMCAD

King, Barrett, and Tinelli. **Leveraging Linear and Mixed Integer Programming for SMT.**

Thursday 1:30pm (just after lunch).

Reynolds, Tinelli, and de Moura. **Finding Conflicting Instances of Quantified Formulas in SMT.**

Thursday 3:45pm (final session of the day).

We use CVC4, too

cascade http://cascade.cims.nyu.edu/

Verification platform for C programs. Supports a range of data representation and memory models for precise or abstract analysis.

NYU

{ KIND } http://clc.cs.uiowa.edu/Kind/

K-induction based model checker for Lustre programs. Novel invariant generation techniques. In use at Rockwell-Collins.

The University of Iowa

Practicalities of using CVC4

- Open-source (BSD-licensed)

- Accepted into MacPorts
- Already in Fedora
- Debian package built (forthcoming in Debian)

Easy to use

- Straightforward API
  - follows SMT-LIB command format
  - C++, Java, …
- Compatibility support for CVC3's APIs
  - C, C++, and Java

Well-supported

- GitHub, Travis-CI
- Users' mailing list
- StackOverflow
- Bugzilla
- Nightly development builds

- …under active development

The CVC4 Team

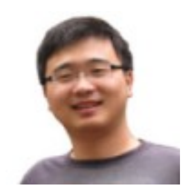

Clark Barrett (NYU)
Cesare Tinelli (U Iowa)



Kshitij Bansal (NYU)
François Bobot (CEA)
Chris Conway (Google)



Morgan Deters (NYU)
Liana Hadarean (NYU)
Dejan Jovanović (SRI)


Tim King (Verimag)
Tianyi Liang (U Iowa)
Andrew Reynolds (EPFL)

Calling all users

- Always looking for users
- Always looking for collaborations

`http://cvc4.cs.nyu.edu/`

Thank you!

Morgan Deters
mdeters@cs.nyu.edu

Tim King
tim.king@imag.fr

Andrew Reynolds
andrew.reynolds@epfl.ch

Cesare Tinelli
cesare-tinelli@uiowa.edu

Clark Barrett
barrett@cs.nyu.edu

THE UNIVERSITY OF IOWA

Verimag

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

NYU

CVC4 is supported in part by the Air Force Office of Scientific Research, Google, Intel Corporation, the National Science Foundation, and Semiconductor Research Corporation