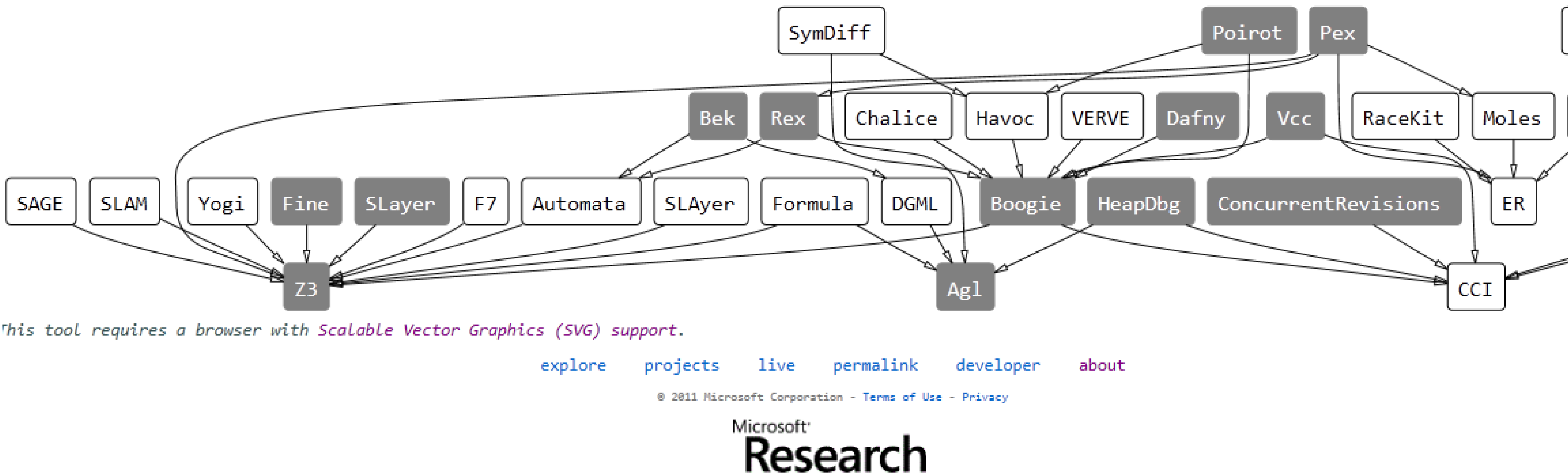
Scaling SMT Solving for Applications

Nikolaj Bjørner
Microsoft Research
Deduction at Scale, Schloβ Ringberg March 7

FSE & RiSE

Some Microsoft Engines using Z3



Try them online: http://rise4fun.com

Pex – Program Exploration

My Duels ▾ | Settings ▾ | Sign In

Pex for fun

Random Puzzle | Learn | New

300,530 clicked 'Ask Pex!'

C# | Visual Basic | F#

Does the Parameterized Unit Test pass for all input values? Click **Ask Pex!** to find out.

```
using System;
using Microsoft.Pex.Framework;
[PexClass]
public class TestClass {
  // Which values will trigger collisions in MyHashSet? Ask Pex to find out!
  [PexMethod] // this puzzle is a 'Parameterized Unit Test'
  public void TestAddContains(int x, int y) {
    var s = new MyHashSet();
    s.Add(x);
    s.Add(y);
    PexAssert.IsTrue(s.Contains(x));
    PexAssert.IsTrue(s.Contains(y));
  }
}
class MyHashSet {
```

Ask Pex! *Done. 6 interesting inputs found.* How does Pex work? Permalink

	x	y	Output/Exception	Error Message
✗	0	0	ArgumentException	'0' not allowed
✗	1	0	ArgumentException	'0' not allowed
✗	-704287306	0	IndexOutOfRangeException	Index was outside the bounds of the array.
✓	1	1		
✓	485	706		
✓	43	690	[DEBUG INFO] collision in Add at index 1	

Rex – Regular Expression Exploration

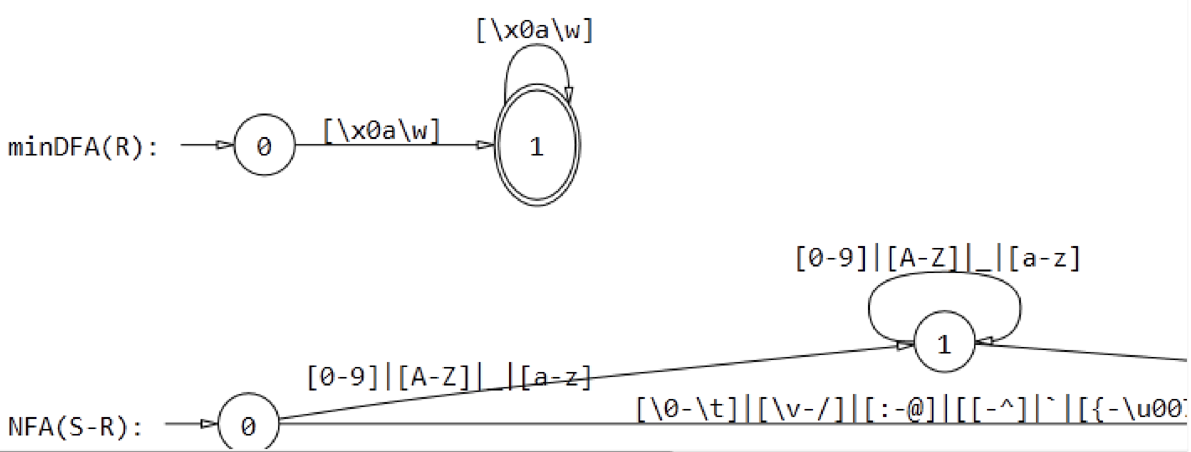
poirot pex rex spec# vcc z3

```
^(\w|\n)+$
```

ask rex *Can you discover the secret regex?* ***Click 'ask Rex'!*** *Read more or watch the video.*

You Missed! Your regex gave different matches than the secret regex. Try modifying it and Ask Rex again!

	string	your regex R	secret regex S	result
✓	"r"	match	match	
✓	"H89i"	match	match	
✗	"F_5\n"	match	no match	Your match is different from the secret regex match.
✗	"Q\n\n"	match	no match	Your match is different from the secret regex match.
✗	"@"	no match	match	Your match is different from the secret regex match.
✗	"95` "	no match	match	Your match is different from the secret regex match.
✓	""	no match	no match	
✓	")\n7|m"	no match	no match	



Margus Veanes

Bek: Regular Symbolic Transducers

Click on a tool to load a sample then ask!

agl | **bek** | boogie | code contracts | concurrent revisions | dafny | esm | fine | heapdbg | poirot | pex | rex | slayer | spec# | vcc | z3

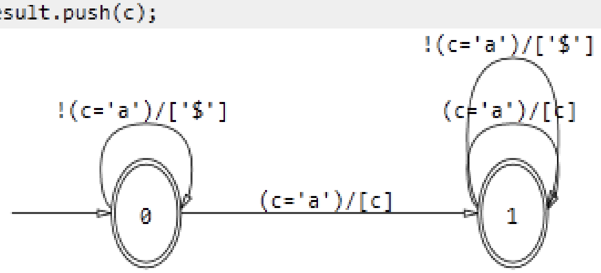
```
program boolAssignmentDemo(t);
string s;
s := iter(c in t){b := false;} {
        case ((c == 'a')) :
                b := !(b) && b;
                b := b || b;
                b := !(b);
                yield (c);
        case (true) :
                yield ('$');
        };
return s;
```

ask bek

Is this sanitizer idempotent? Click 'ask bek'! Read more or watch the video.

```
// BEK says : boolAssignmentDemo is idempotent
// BEK says : boolAssignmentDemo is not reversible.

// The following JavaScript is equivalent to the BEK program:
function boolAssignmentDemo(t)
{
var s =
function ($){
var result = new Array();
for(i=0;i<$.length; i++){
var c =$[i];
if ((c == String.fromCharCode(97)))
{
b := (~(b) && b);
b := (b || b);
b := ~(b);
result.push(c);
```



Margus Veanes
David Molnar

SAGE by the numbers

Slide shamelessly stolen and adapted from [Patrice Godefroid, ISSTA 2010]

100+ CPU-years - largest dedicated fuzz lab in the world

100s apps - fuzzed using SAGE

100s previously unknown bugs found

1,000,000,000+ computers updated with bug fixes

Millions of **$** saved for Users and Microsoft

10s of related tools (incl. Pex), 100s DART citations

100,000,000+ constraints - largest usage for any SMT solver

PREfix [Moy, B., Sielaff]

3(INT_MAX+1)/4 +
(INT_MAX+1)/4
= INT_MIN

```
int binary_se
while (low <= hig
  {
    // Find middle value
    int mid = (low + high) / 2;
    int val = arr[mid];
    if (val == key) return mid;
    if (val < key) low = mid+1;
    else high = mid-1;
  }
```

Package: java.util.Arrays
Function: binary_search

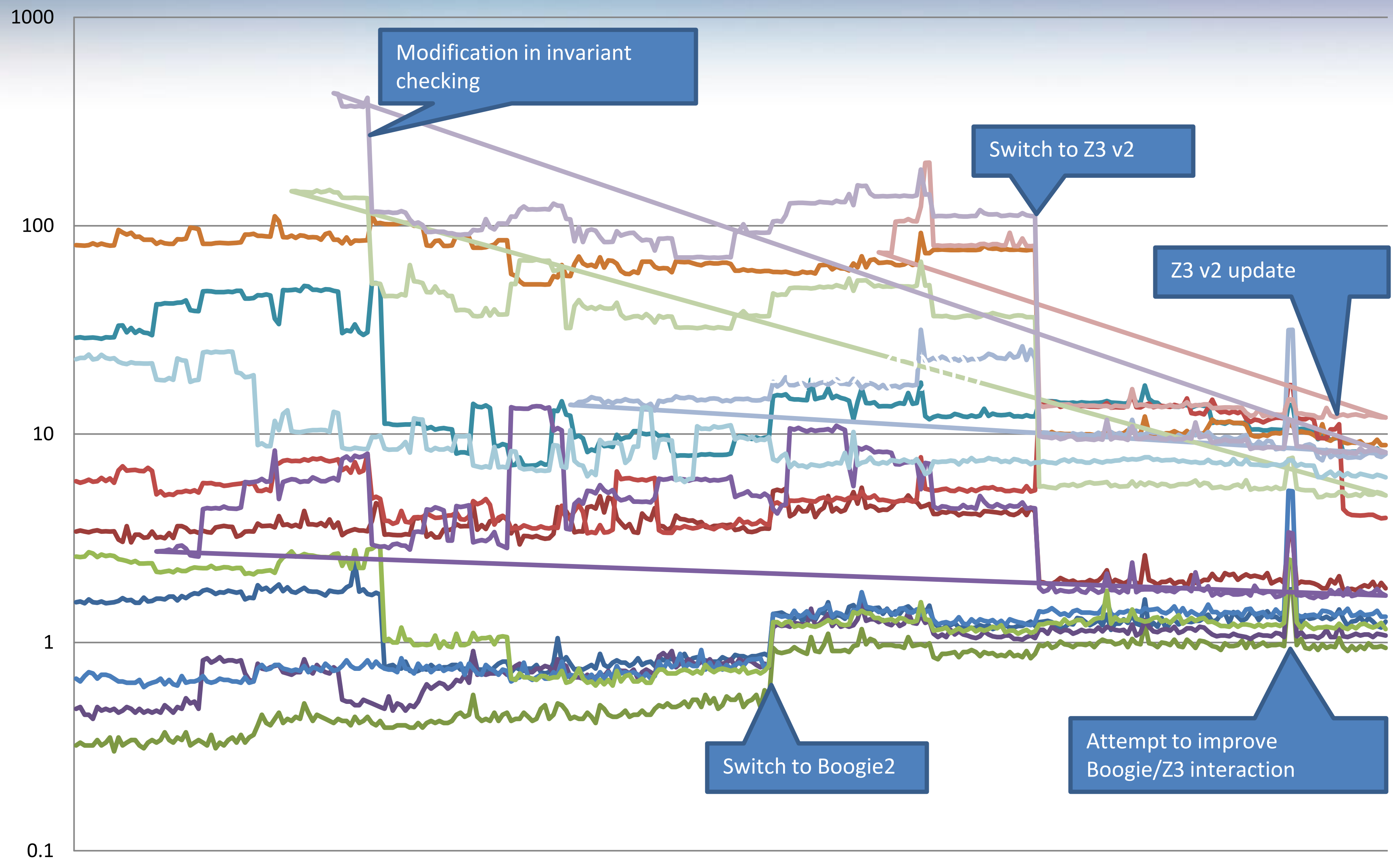
-INT_MIN=
INT_MIN

```
id itoa(int n, char*
  if (n < 0) {
     *s++ = '-';
     n = -n;
  }
  // Add digits to s
  ....
```

Book: Kernighan and Ritchie
Function: itoa (integer to ascii)

Analysis of millions of lines of Microsoft Code base

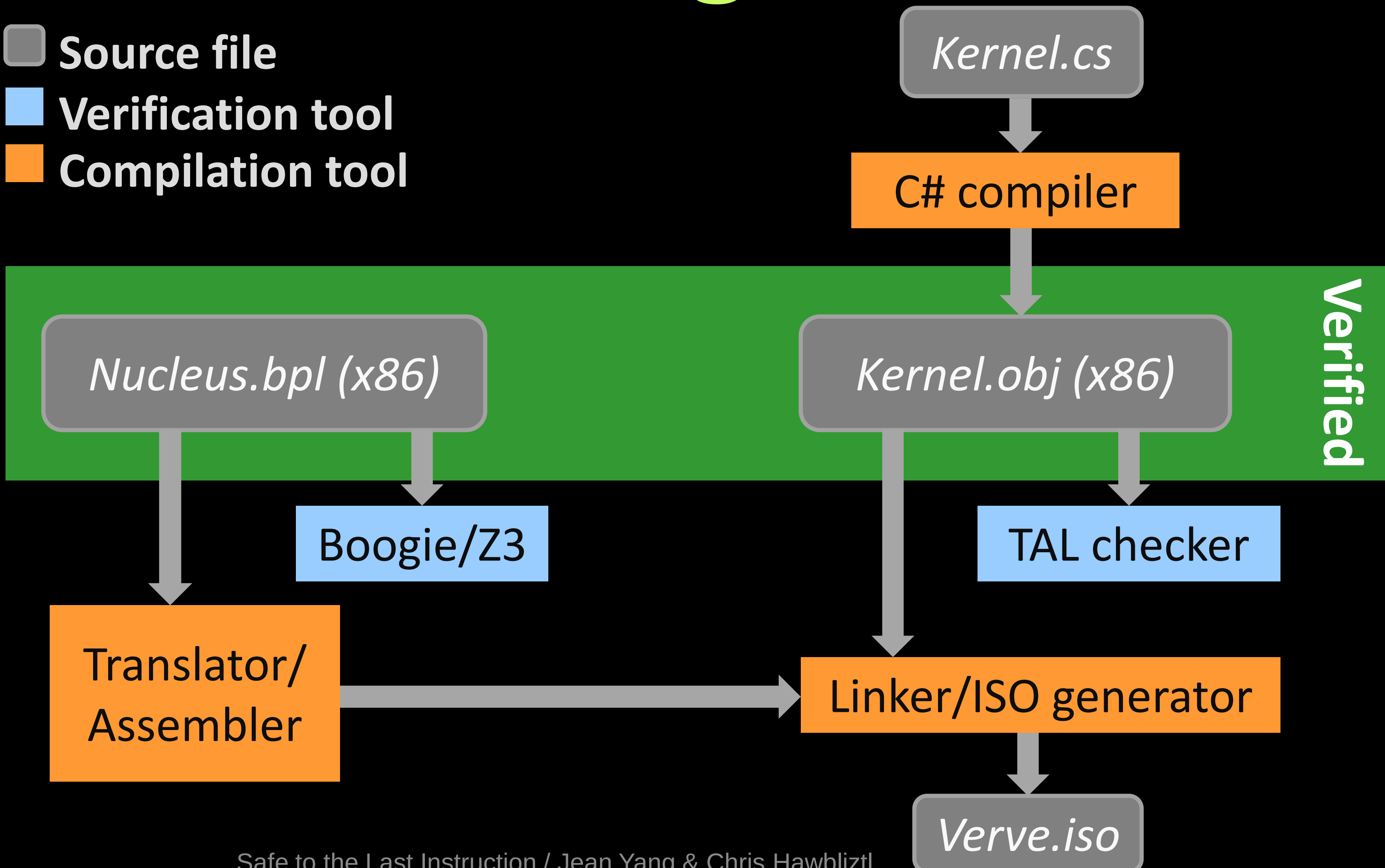
VCC Performance Trends Nov 08 – Mar 09



Building Verve

9 person-months

Source file
Verification tool
Compilation tool



Scale: what is important - for applications?

Claim (as I see it):

- *Simplification* - lots of junk
- *Structural* - not random, (symmetry?)
- *Shallow* - unsat core
- *Repertoire* - cooperating methods
- *Decomposable* - solve simpler problems
- *Abstraction* - SAT < SMT

Are we there yet?

- Improve search methods and solvers,
- extend expressiveness, *tactics*,
- precise answers.

Scale: what is important - for applications?

Claim (as I see it):

- *Simplification* - lots of junk
- *Structural* - not random, (symmetry?)
- *Shallow* - unsat core
- *Repertoire* - cooperating methods
- *Decomposable* - solve simpler problems
- *Abstraction* - SAT < SMT

Are we there yet?

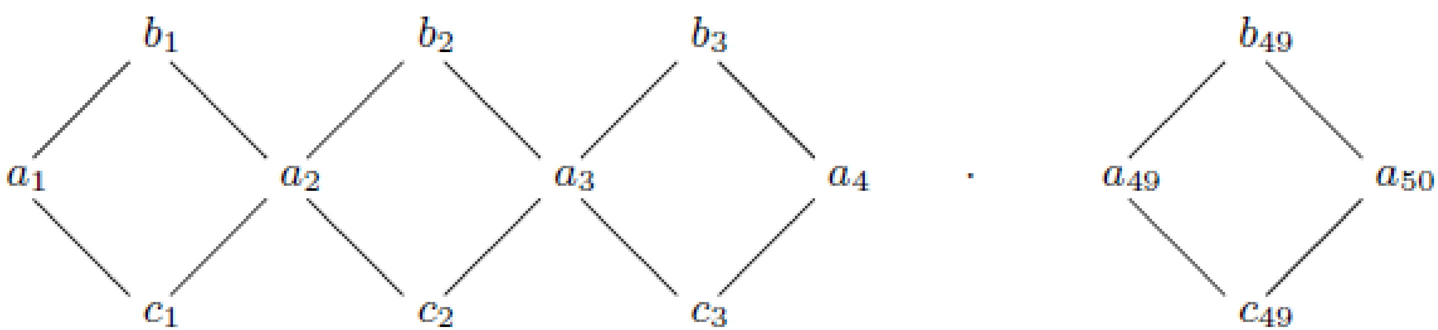
- **Improve search methods** and solvers,
- extend expressiveness, *tactics*,
- precise answers.

DPLL(T) misses short proofs

resolution

The **Black Diamonds** of DPLL(T)

$$\neg(a_1 \simeq a_{50}) \;\wedge\; \bigwedge_{i=1}^{49} [(a_i \simeq b_i \wedge b_i \simeq a_{i+1}) \vee (a_i \simeq c_i \wedge c_i \simeq a_{i+1})]$$



Has no short DPLL(T) proof.

Has short DPLL(T) proof when using $a_1 \simeq a_2, a_2 \simeq a_3, a_3 \simeq a_4, \ldots, a_{49} \simeq a_{50}$

Example from [Rozanov, Strichman, SMT 07]

DPLL(T) in a nutshell

T- Propagate $\quad M \mid F, C \vee \ell \implies M, \ell^{C \vee \ell} \mid F, C \vee \ell \quad$ $C \text{ is } false \text{ under } T + M$

T- Conflict $\quad M \mid F \implies M \mid F \mid \neg M' \quad$ $M' \subseteq M \text{ and } M' \text{is } false \text{ under } T$

T- Propagate $\quad a > b, b > c \mid F, a \leq c \vee b \leq d \implies$

$$a > b, b > c, b \leq d^{a \leq c \vee b \leq d} \mid F, a \leq c \vee b \leq d$$

T- Conflict $\quad M \mid F \implies M \mid F,\ a \leq b \vee b \leq c \vee c < a$

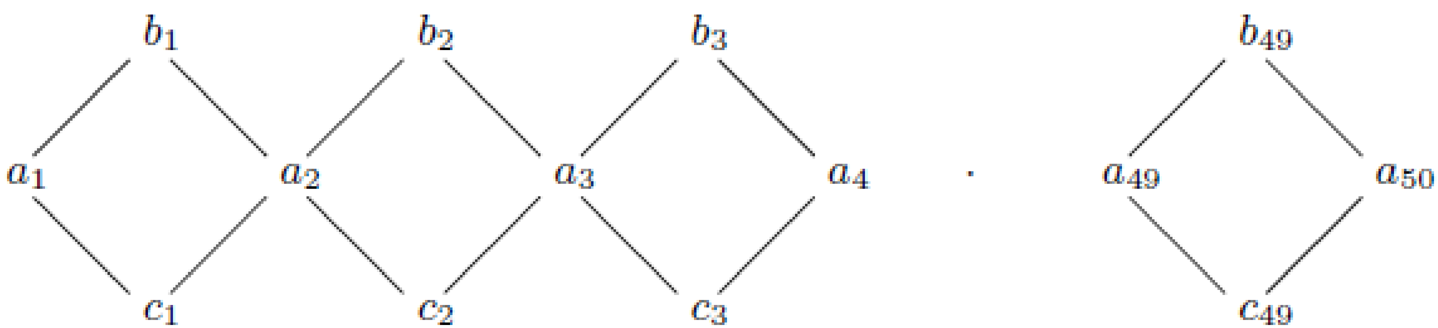
$$where\ a > b, b > c, a \leq c \subseteq M$$

Introduces no new literals - terminates

DPLL(T) misses short proofs

Idea: DPLL(⊔)

[B, Dutertre, de Moura 08]



Try branch $a_1 \simeq b_1 \wedge b_1 \simeq a_2$
Implies $a_1 \simeq b_1 \simeq a_2$
Collect implied equalities

Try branch $\neg(a_1 \simeq b_1 \wedge b_1 \simeq a_2)$
Implies $a_1 \simeq c_1 \simeq a_2$
Collect implied equalities

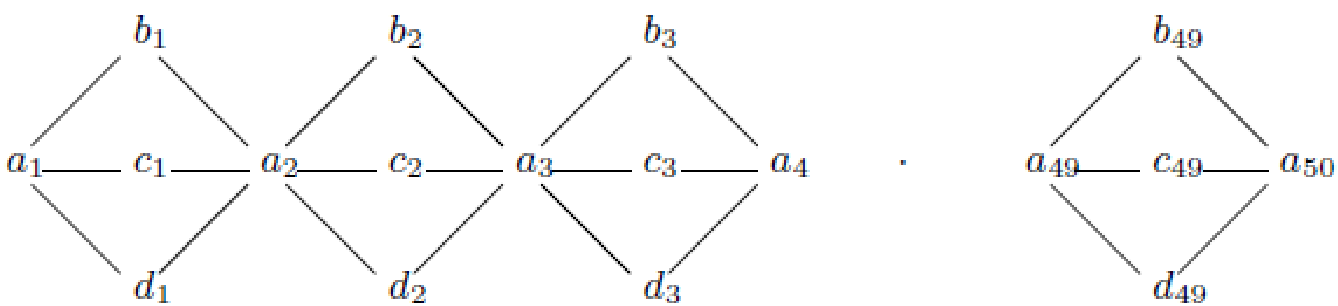
Compute the ***join*** ⊔ of the two equalities – common equalities are learned

Still potentially O(n^2) rounds just at ***base*** level of search.

DPLL(⊔ base) misses short proofs

- Single case splits don't suffice

$$a_1 \not\simeq a_{50} \;\wedge\; \bigwedge_{i=1}^{49} \left[\begin{array}{l} \quad (a_i \simeq b_i \wedge b_i \simeq a_{i+1}) \\ \vee\ (a_i \simeq c_i \wedge c_i \simeq a_{i+1}) \\ \vee\ (a_i \simeq d_i \wedge d_i \simeq a_{i+1}) \end{array} \right]$$



Requires 2 case splits to collect implied equalities

Conflict Directed Theory Resolution

Method: *resolve* literals in conflict clauses

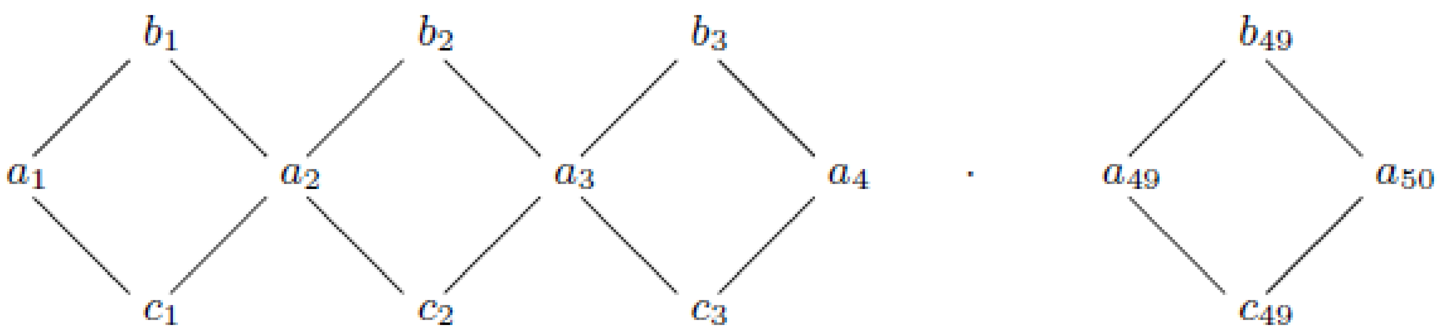
Theorem (for EUF): DPLL + CD**E**R + Restart $\equiv_p$ **E-**Resolution
Informal Claim: DPLL + CD**T**R + Restart $\equiv_p$ Resolution

Practical?

Method introduces extra literals (= junk)
→ *Throttle* resolution dynamically based on activity.

Th(Equality) - Example

$$\neg(a_1 \simeq a_{50}) \wedge \bigwedge_{i=1}^{49} [(a_i \simeq b_i \wedge b_i \simeq a_{i+1}) \vee (a_i \simeq c_i \wedge c_i \simeq a_{i+1})]$$



Eventually, many conflicts contain: $a_1 \simeq b_1 \wedge b_1 \simeq a_2$

Use E-resolution, add clause: $a_1 \simeq b_1 \wedge b_1 \simeq a_2 \rightarrow a_1 \simeq a_2$

Then DPLL(T) learns by itself: $a_1 \simeq a_2$

Th(Equality) - Example

$$\bigwedge_{i=1}^{N} (p_i \vee x_i \simeq v_0) \wedge (\neg p_i \vee x_i \simeq v_1) \wedge (p_i \vee y_i \simeq v_0) \wedge (\neg p_i \vee y_i \simeq v_1) \wedge$$
$$\neg(f(x_N, \ldots, f(x_2, x_1) \ldots) \simeq f(y_N, \ldots, f(y_2, y_1) \ldots))$$

Eventually, many conflicts contain:

$$x_i \simeq u_i \wedge y_i \simeq u_i \quad u_i = v_0 \; or \; u_i = v_1 \quad for \; i = 1..N$$
$$\neg(f(x_N, \ldots, f(x_2, x_1) \ldots) \simeq f(y_N, \ldots, f(y_2, y_1) \ldots))$$

Add:

$$(\bigwedge_{i=1}^{N} x_i \simeq y_i) \rightarrow f(x_N, \ldots, f(x_2, x_1) \ldots) \simeq f(y_N, \ldots, f(y_2, y_1) \ldots)$$

CDTR for Th(Equalities)

Dynamic Ackermann Reduction

If *Congruence Rule* repeatedly learns

$$f(v, v') \sim f(w, w')$$

Then add clause for SAT core to use

$$v \simeq w \land v' \simeq w' \rightarrow f(v, v') \simeq f(w, w')$$

Dynamic Ackermann Reduction with Transitivity

If *Equality Transitivity* repeatedly learns

$$u \sim w \qquad\qquad from\, u \sim v\; and\; v \sim w$$

Then add clause for SAT core to use

$$u \simeq v \land v \simeq w \rightarrow v \ \simeq w$$

CDTR for Th(Equalities)

Dynamic Ackermann Reduction

If *Congruence Rule* <u>repeatedly</u> learns

$f(v, v') \sim f(w, w')$ for literal $f(v, v') \simeq f(w, w')$

Then add clause for SAT core to use

$$v \simeq w \wedge v' \simeq w' \rightarrow f(v, v') \simeq f(w, w')$$

Dynamic Ackermann Reduction with Transitivity

If *Equality Transitivity* <u>repeatedly</u> learns

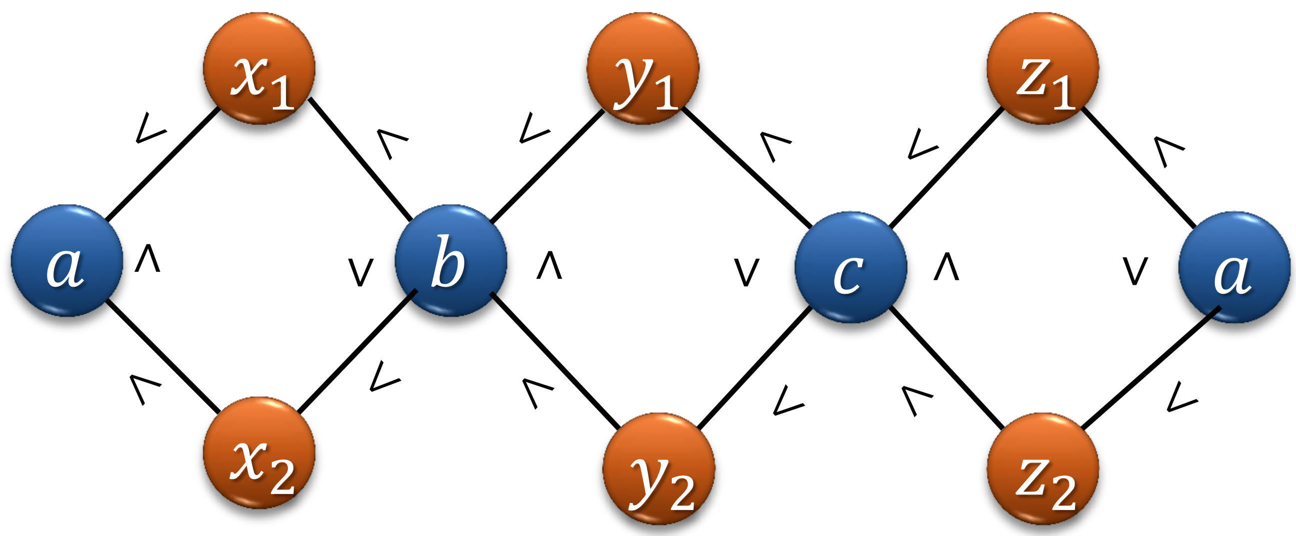
$u \sim w \qquad from\ u \sim v\ and\ v \sim w$

Then add clause for SAT core to use

$$u \simeq v \wedge v \simeq w \rightarrow v \ \simeq w$$

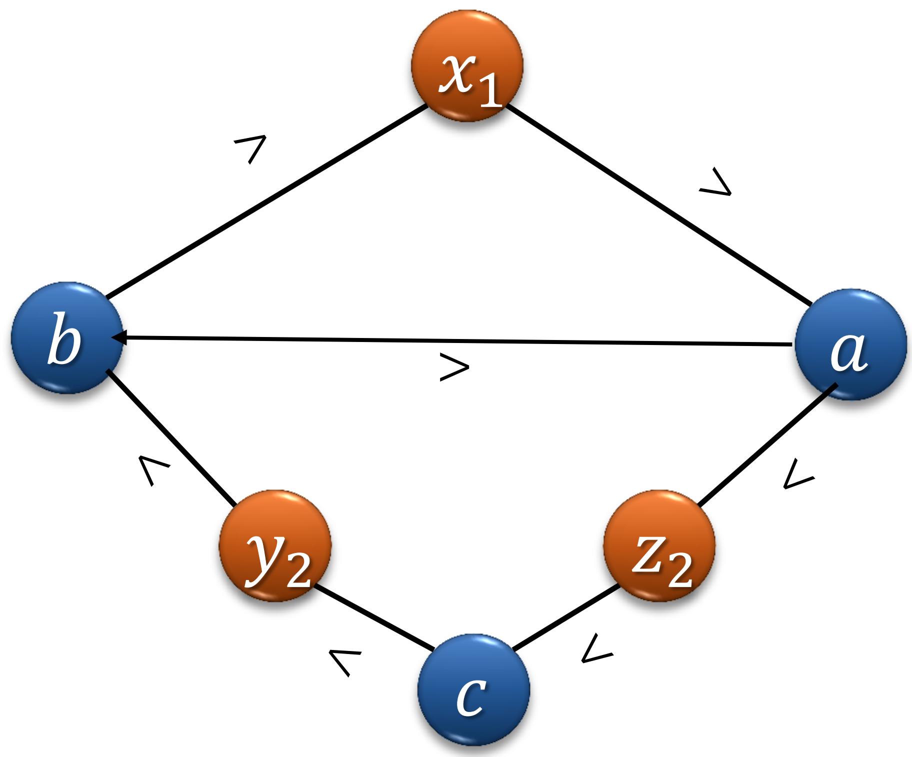
CDTR: A cottage industry?

$$a < x_1 \wedge a < x_2 \wedge (x_1 < b \vee x_2 < b) \wedge$$
$$b < y_1 \wedge b < y_2 \wedge (y_1 < c \vee y_2 < c) \wedge$$
$$c < z_1 \wedge c < z_2 \wedge (z_1 < a \vee z_2 < a)$$



CDTR: *Linear Difference Arithmetic*

Top Two Most Active vertices



Add clause

$a < x_1 < b \rightarrow a < b$

Summary

- Modern SMT solvers find resolution proofs
  - unlike SAT solvers: SMT $>_p$ RES
  - Gap is real enough
- Presented a technique for equalities
  - Based on applying **Resolution** to conflicts.
  - **Dynamic** - to address literal introduction junk.
- Just one of many possible optimizations.
  - e.g. cutting plane proofs, arbitrary cuts (Frege)
  - The devil is in the theory