Topic : Z3

CSE6323 - Formal Methods in Software Engineering

Ramanatha Gopireddy
Ishtiaque Hussain
Raghavendra Madala

Outline

- Introduction
  - Development of Z3
  - Applications
- Tool demo
  - Sudoku: Rules and implementation
  - How Z3 helps solving.
- Background theories for SMT Solver.
- Comparisons with other existing tools.
- Current SERC@UTA projects using Z3.
- Summary.

What is Z3?

- Z3 is a high-performance theorem prover being developed at *Microsoft Research*.
- Theorem prover: It takes logical formulas as input and checks whether the formula is correct or not.
- Z3 is a *Satisfiability Modulo Theories (SMT)* solver.
- When a set of formulas F is satisfiable, Z3 can produce a model for F.
- Z3 integrates several decision procedures.

Development of Z3

- Z3 was developed by Nikolaj Bjorner and Leonardo de Moura , researchers in the *Research in Software Engineering (RiSE)* team at Microsoft Research.

- Z3 0.1 was submitted to SMT COMP-07 on 25 June 2007 and Z3 1.0 was released on 13 Sep 2007.
- Z3 2.5 is the present version.

Applications of Z3

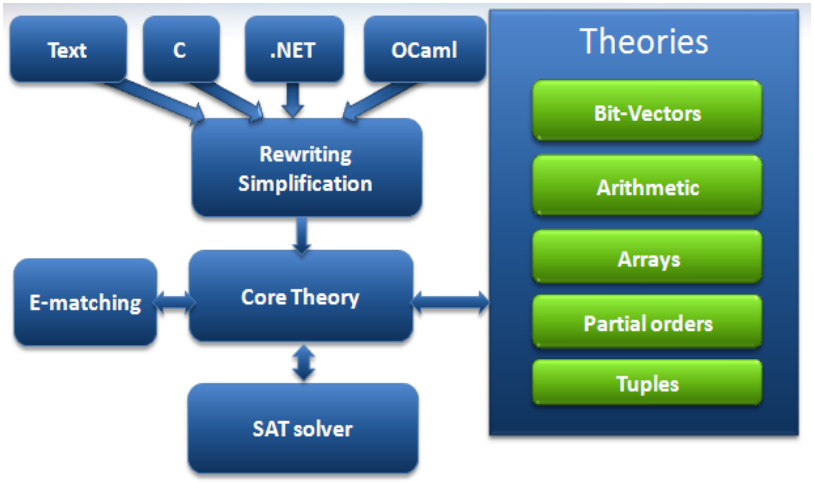
Z3 is used in many applications.

- Verifying Compiler.
  Spec#, VCC, HAVOC.
- Test-case generation.
  Pex, SAGE and Vigilante.
- Predicate Abstraction.
  SLAM/SDV and Yogi.
- Planning & Scheduling.
- Equivalence checking.

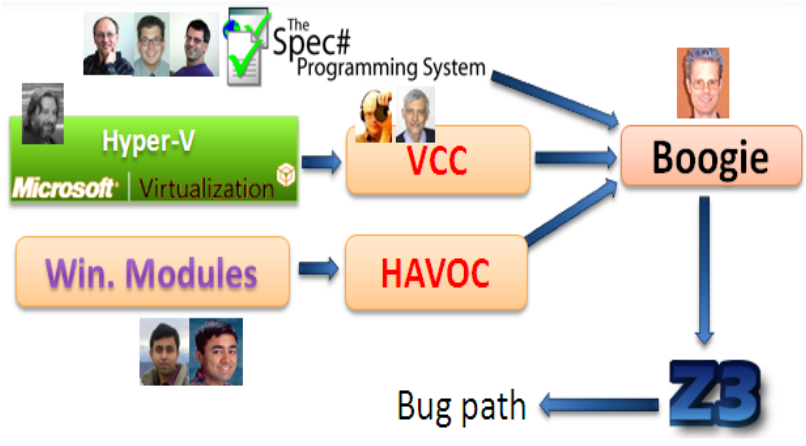
Main Features

- Supports different theories:
  → Linear real and integer arithmetic.
  → Bit-vectors.
  → Uninterpreted functions.
  → Arrays.
- Quantifiers.
- Model generation.
- Input formats:
  Text formats:
  Simplify, SMT-LIB, Z3, Dimacs.
  API formats:
  C/C++, .NET, OCaml.

Z3: Core System Components



Clients: Program Verification



Tool Demo

- Sudoku:

1	9		5				6	4
	4				8	7		
	7	8	1			9		
7								
			3	7	9	2		
		9		1	4	3	5	
	5	2		3	6	4	7	
	1					6	8	
	8			9			3	

Problem

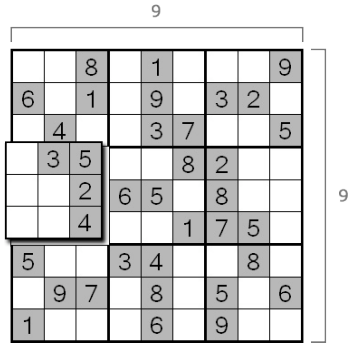
1	9	3	5	2	7	8	6	4
2	4	5	9	6	8	7	1	3
6	7	8	1	4	3	9	2	5
7	3	4	2	8	5	1	9	6
5	6	1	3	7	9	2	4	8
8	2	9	6	1	4	3	5	7
9	5	2	8	3	6	4	7	1
3	1	7	4	5	2	6	8	9
4	8	6	7	9	1	5	3	2

Solution

Source: http://www.puzzles.ca/sudoku_puzzles/sudoku_easy_005.html

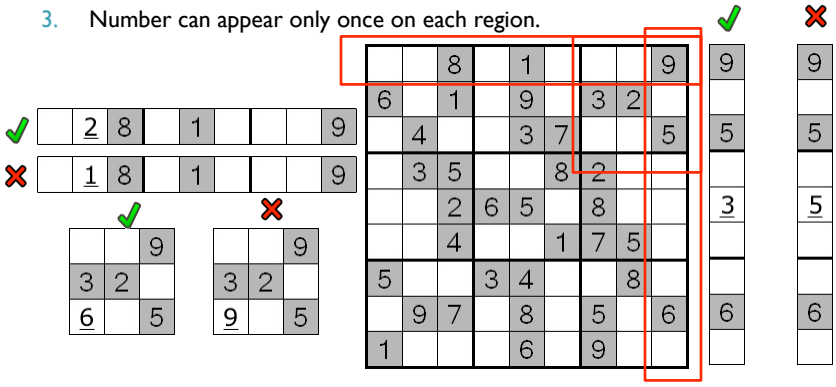
Sudoku rules:

- Sudoku is played over a 9x9 grid, divided to 3x3 sub grids called "regions".
- Sudoku begins with some of the grid cells already filled with numbers.



Continued…

- The objective of Sudoku is to fill the other empty cells with numbers between 1 and 9 (1 number only in each cell) according the following guidelines:
  1. Number can appear only once on each row.
  2. Number can appear only once on each column.
  3. Number can appear only once on each region.



Source: http://www.sudoku.name/rules/en

How do you mathematically represent the guidelines?

- Represent all the cells as variables (9X9 = 81 of them).
- Set the range of these variables from 1 to 9, inclusive.
- Assert all variables in each row as distinct.
- Assert all variables in each column as distinct.
- Assert that in a region each variable is not equal to others.
- Assign some variables values from the filled up graph (Starting Sudoku).

Background theories for SATISFIABILITY MODULO THEORIES (SMT) SOLVER

The following slides are based on Leonardo de Moura's presentation for UOregon 2008.
Source: http://research.microsoft.com/en-us/um/people/leonardo/oregon08.pdf

SMT Solver: Overview

- Decision procedures for checking satisfiability of logical formulas are crucial for many verification applications.
- Satisfiability is the problem of determining if a formula has a model.
- In the purely Boolean case, a model is a truth assignment to the Boolean variables.
- In the first-order case, a model assigns values from a domain to variables and interpretations over the domain to the function and predicate symbols.

Logic: Basics

- Logic studies the trinity between language, interpretation and proof.
- Language circumscribes the syntax that is used to construct sensible assertions.
- Interpretation ascribes an intended sense to these assertions by fixing the meaning of certain symbols.

  An assertion is valid if it holds in all interpretations.
- Checking validity through interpretations is typically not feasible, so proofs in the form axioms and inference rules are used to demonstrate the validity of assertions.

Language: Signature

- A signature $\sum$ is a finite set of:
- Function symbols: $\sum_F = \{f, g, \ldots\}$
- Predicate symbols: $\sum_P = \{p, q, \ldots\}$
- and an arity function: $\sum \mapsto N$
- Function symbols with arity 0 are called constants.
- A countable set V of variables disjoint of $\sum$.

Language: Terms

- The set $T(\sum, V)$ of terms is the smallest set such that:
- $V \subset T(\sum, V)$
- $f(t_1, \ldots, t_n) \in T(\sum, V)$ whenever
- $f \in \sum_F, t_1, \ldots, t_n \in T(\sum, V)$ and *arity(f) = n.*
- The set of ground terms is defined as $T(\sum, \emptyset)$.

Z3- An Efficient SMT Solver 17

Language: Atomic Formula

- $p(t_1, \ldots, t_n)$ is an atomic formula whenever

  $p \in \sum_P$, *arity(p) = n, and* $t_1, \ldots, t_n \in T(\sum, V)$.
- true and false are atomic formulas.
- If $t_1, \ldots, t_n$ are ground terms, then $p(t_1, \ldots, t_n)$ is called a ground (atomic) formula.
- A literal is an atomic formula or its negation.

Z3- An Efficient SMT Solver 18

Language: Quantifier Free Formulas

- The set *QFF(∑, V) of quantifier free formulas is the smallest set* such that:
- Every atomic formulas is in *QFF(∑, V)*.
- If $\Phi \in QFF(\sum, V)$, *then* $\neg\Phi \in QFF(\sum, V)$.
- If $\Phi_1, \Phi_2 \in QFF(\sum, V)$, *then*
  $\Phi_1 \wedge \Phi_2 \in QFF(\sum, V)$
  $\Phi_1 \vee \Phi_2 \in QFF(\sum, V)$
  $\Phi_1 \Rightarrow \Phi_2 \in QFF(\sum, V)$
  $\Phi_1 \Leftrightarrow \Phi_2 \in QFF(\sum, V)$

Model

- A model M is defined as:
  Domain |M|: set of elements.
- A formula is true in a model M if it evaluates to true under the given interpretations over the domain |M|.

Interpreting Terms

- M[[x]] = M(x)
- M[[f(a_1, . . . , a_n)]] = M(f)
  (M[[a_1]], . . . ,M[[a_n]])

Interpretation Example

- ∑ = {0,+,<}, *and M such that* |M| = {*a, b, c*}
- M(0) = a,
- M(+) = {<a, a ↦ a>, <a, b ↦ b>, <a, c ↦ c>,
  <b, a ↦ b>, <b, b ↦ c>, <b, c ↦ a>,
  <c, a ↦ c>, <c, b ↦ a>, <c, c ↦ b>}
- M(<) = {<a, b>, <a, c>, <b, c>}
- If M(x) = a, M(y) = b, M(z) = c, then
  M[+(+(x, y), z)] =
  M(+)(M(+)(M(x),M(y)),M(z)) =
  M(+)(M(+)(a, b), c) =
  M(+)(b, c) = a

Interpretation Example

- ∑ = {0,+,<}, *and M such that* |M| = {*a, b, c*}
- M(0) = a,
- M(+) = {<a, a ↦ a>, <a, b ↦ b>, <a, c ↦ c>,
  <b, a ↦ b>, <b, b ↦ c>, <b, c ↦ a>,
  <c, a ↦ c>, <c, b ↦ a>, <c, c ↦ b>}
- M(<) = {<a, b>, <a, c>, <b, c>}

$M \models (\forall x : (\exists y : +(x,y) = 0))$

CORRECT

$M \models (\forall x : (\exists y : x < y))$

WRONG

$M \models (\forall x : (\exists y : +(x,y) = x))$

CORRECT

Validity

- A formula F is satisfiable if there is an interpretation M such that M ⊨ F.
- Otherwise, the formula F is unsatisfiable.

Clausal(CNF) Form

- In clausal form, the formula is a set (conjunction) of clauses $\bigvee_i C_i$, and each clause C_i is a disjunction of literals.
- A literal is an atom or the negation of an atom.
- $p1 \vee \neg p2, \neg p1 \vee p2 \vee p3, p3$
- Most SAT solvers assume the formula is in CNF.

The DPLL Procedure

- David – Putnam – Logemann - loveland algorithm is a complete backtracking- based algorithm for deciding the satisfiability of a logical formulae CNF.
- DPLL tries to build incrementally a model M for a CNF formula F.
- M is grown by:
  deducing the truth value of a literal from M and F, (or)
  guessing a truth value.
- If a wrong guess leads to an inconsistency, the procedure backtracks and tries the opposite one.

Breakthrough In SAT Solving

- Modern SAT solvers are based on the DPLL algorithm.
- Modern implementations add several sophisticated search techniques.
  - Backjumping
  - Learning
  - Restarts
  - Indexing

Abstract DPLL: Example

$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$

Abstract DPLL: Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$
$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Abstract DPLL: Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$
$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$
$$1\ 2_{\overline{1} \vee 2} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Abstract DPLL: Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Abstract DPLL: Example

$$\| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(Decide)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3 \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow \quad \text{(UnitProp)}$$

$$1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4} \ \| \quad \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}$$

Abstract DPLL: Example

$$\begin{array}{rcll}
 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2} & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1\ 2_{\overline{1} \vee 2}\ 3 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4} & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} &
\end{array}$$

Abstract DPLL: Example

$$\begin{array}{rcll}
 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2} & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1\ 2_{\overline{1} \vee 2}\ 3 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4} & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5 & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5\ \overline{6}_{\overline{5} \vee \overline{6}} & \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} &
\end{array}$$

Abstract DPLL: Example

$$\begin{array}{rcl}
\| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1 \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2} \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1\ 2_{\overline{1} \vee 2}\ 3 \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4} \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5 \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5\ \overline{6}_{\overline{5} \vee \overline{6}} \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Conflict)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5\ \overline{6}_{\overline{5} \vee \overline{6}} \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} \| & 6 \vee \overline{5} \vee \overline{2}
\end{array}$$

Abstract DPLL: Example

$$\begin{array}{rcl}
\| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1 \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2} \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1\ 2_{\overline{1} \vee 2}\ 3 \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4} \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Decide)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5 \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(UnitProp)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5\ \overline{6}_{\overline{5} \vee \overline{6}} \| & \overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2} & \Rightarrow \quad \text{(Conflict)} \\
1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5\ \overline{6}_{\overline{5} \vee \overline{6}} \| & \underbrace{\overline{1} \vee 2,\ \overline{3} \vee 4,\ \overline{5} \vee \overline{6},\ 6 \vee \overline{5} \vee \overline{2}}_{F} \| & 6 \vee \overline{5} \vee \overline{2}
\end{array}$$

Abstract DPLL: Example

$1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5\ \overline{6}_{\overline{5} \vee \overline{6}} \parallel \quad F \quad \parallel \quad 6 \vee \overline{5} \vee \overline{2}$

Abstract DPLL: Example

$1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5\ \overline{6}_{\overline{5} \vee \overline{6}} \parallel \quad F \quad \parallel \quad 6 \vee \overline{5} \vee \overline{2} \quad \Rightarrow$ (Resolve)

$1\ 2_{\overline{1} \vee 2}\ 3\ 4_{\overline{3} \vee 4}\ 5\ \overline{6}_{\overline{5} \vee \overline{6}} \parallel \quad F \quad \parallel \quad \overline{5} \vee \overline{2}$

Abstract DPLL: Example

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F \parallel 6 \vee \overline{5} \vee \overline{2} \Rightarrow \text{(Resolve)}$$

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F \parallel \overline{5} \vee \overline{2} \Rightarrow \text{(Learn)}$$

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F,\ \overline{5} \vee \overline{2} \parallel \overline{5} \vee \overline{2}$$

Abstract DPLL: Example

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F \parallel 6 \vee \overline{5} \vee \overline{2} \Rightarrow \text{(Resolve)}$$

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F \parallel \overline{5} \vee \overline{2} \Rightarrow \text{(Learn)}$$

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F,\ \overline{5} \vee \overline{2} \parallel \overline{5} \vee \overline{2} \Rightarrow \text{(Resolve)}$$

$$1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel F,\ \overline{5} \vee \overline{2} \parallel \overline{5} \vee \overline{1}$$

Abstract DPLL: Example

$$
\begin{array}{rlll}
1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel & F & \parallel\ 6\vee\overline{5}\vee\overline{2} & \Rightarrow \text{ (Resolve)} \\
1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel & F & \parallel\ \overline{5}\vee\overline{2} & \Rightarrow \text{ (Learn)} \\
1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel & F,\ \overline{5}\vee\overline{2} & \parallel\ \overline{5}\vee\overline{2} & \Rightarrow \text{ (Resolve)} \\
1\ 2_{\overline{1}\vee 2}\ 3\ 4_{\overline{3}\vee 4}\ 5\ \overline{6}_{\overline{5}\vee\overline{6}} \parallel & F,\ \overline{5}\vee\overline{2} & \parallel\ \overline{5}\vee\overline{1} & \Rightarrow \text{ (Backjump)} \\
1\ 2_{\overline{1}\vee 2}\ \overline{5}_{\overline{5}\vee\overline{1}} \parallel & F,\ \overline{5}\vee\overline{2} & &
\end{array}
$$

Abstract DPLL

$$
\begin{array}{lcll}
M \parallel F & \Longrightarrow & M\ l \parallel F & \text{if } \begin{cases} l \text{ or } \bar{l} \text{ occurs in } F, \\ l \text{ is undefined in } M \end{cases} \quad \text{(Decide)} \\
M \parallel F, C \vee l & \Longrightarrow & M\ l_{C\vee l} \parallel F, C \vee l & \text{if } \begin{cases} M \models \neg C, \\ l \text{ is undefined in } M \end{cases} \quad \text{(UnitPropagate)} \\
M \parallel F, C & \Longrightarrow & M \parallel F, C \parallel C & \text{if } M \models \neg C \quad \text{(Conflict)} \\
M \parallel F \parallel C \vee \bar{l} & \Longrightarrow & M \parallel F \parallel D \vee C & \text{if } l_{D\vee l} \in M, \quad \text{(Resolve)} \\
M \parallel F \parallel C & \Longrightarrow & M \parallel F, C \parallel C & \text{if } C \notin F \quad \text{(Learn)} \\
M\ l'\ M' \parallel F \parallel C \vee l & \Longrightarrow & M\ l_{C\vee l} \parallel F & \text{if } \begin{cases} M \models \neg C, \\ l \text{ is undefined in } M \end{cases} \quad \text{(Backjump)}
\end{array}
$$

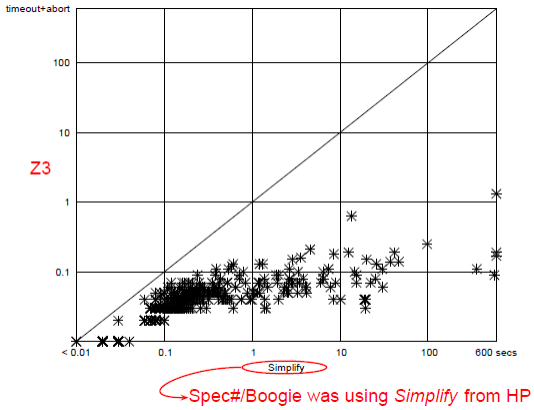
DPLL: Strategy

- Abstract DPLL is very flexible.
- Basic Strategy:
  - Only apply **Decide if UnitPropagate and Conflict cannot be** applied.
- Conflict Resolution:
  - Learn only one clause per conflict (the clause used in **Backjump**).
  - Use **Backjump as soon as possible**.
  - Use the rightmost (applicable) literal in M when applying **Resolve**.

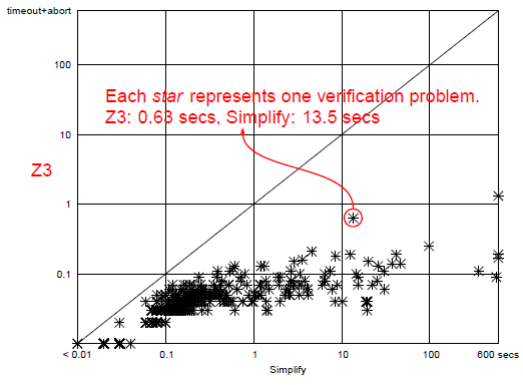
Other SMT Solvers

- Barcelogic, Beaver, CVC, CVC3, MathSMT, Simplify, Yices.
- Spec# replaced Simplify theorem prover by Z3 as the default reasoning engine in May 2007 resulting in substantial performance improvements.

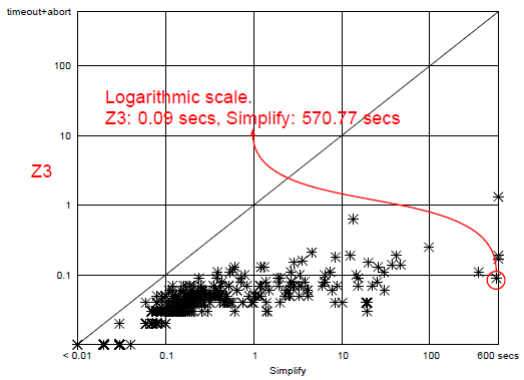
Performance(Spec#/Boogie): Z3× Simplify



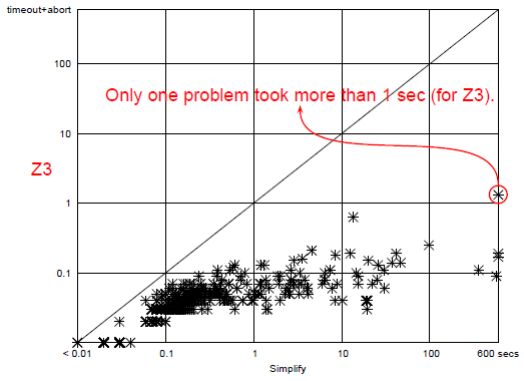
Performance(Spec#/Boogie): Z3× Simplify (Contd..)



Performance(Spec#/Boogie): Z3× Simplify (Contd..)



Performance(Spec#/Boogie): Z3× Simplify (Contd..)



Summary

- Powerful, mature, and versatile tools like SMT solvers can now be exploited in very useful ways.
- The construction and application of satisfiability procedures is an active research area with exciting challenges.
- SMT is hot at Microsoft.
- Z3 is a new SMT solver.
- Main applications:
  - Test-case generation.
  - Verifying compiler.
  - Model Checking & Predicate Abstraction.

References

- http://research.microsoft.com/en-us/um/redmond/projects/z3/index.html
- http://www.cs.uoregon.edu/research/summerschool/summer08/
- http://channel9.msdn.com/posts/Peli/The-Z3-Constraint-Solver/
- http://research.microsoft.com/en-us/um/people/leonardo/oregon08.pdf