Microsoft Research

New and future features of Z3

MSR Cambridge, Z3 SIG meeting

Z3 Team

Leonardo de Moura,

Nikolaj Bjørner

Christoph Wintersteiger

Background

- Z3 is a theorem prover developed at MSR
- First version released in 2007
- First Clients:
  - SPEC#/Boogie, SAGE, PEX
- Main Features in 2007
  - Lazy architecture: SAT solver + theories
  - Linear (real) arithmetic, BV, UF, arrays
  - E-Matching (for Quantifiers)
  - Push/Pop (incremental solving)

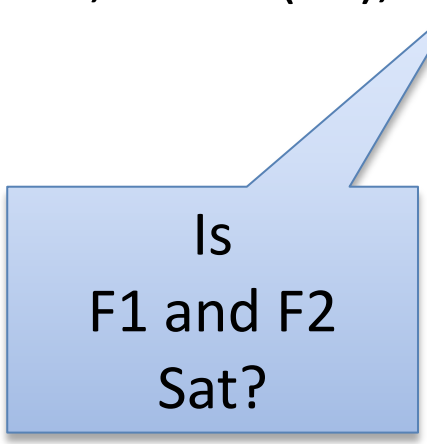
Microsoft Research

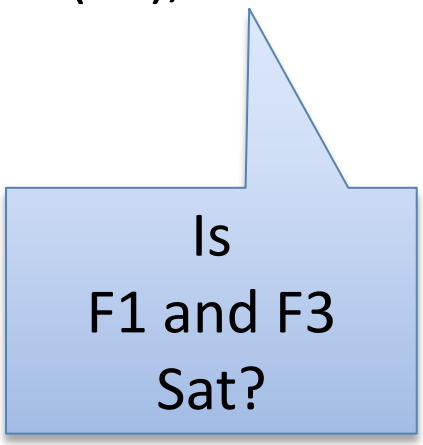
Push, Assert, Check, Pop

Popularized by SMT solvers such as: Simplify.

Part of SMT-LIB 2.0 standard.

push, assert(F1), push, assert(F2), check, pop, assert(F3), check





Supported Platforms

- Windows 32 and 64 bit versions
- .NET wrapper (C#, F#, VB)
- Linux 32 and 64 bit versions
- New: Mac OSX (starting at Z3 3.3)

Microsoft Research

Some Applications @ Microsoft

HAVOC

Terminator T-2

VCC

NModel

Vigilante

SpecExplorer

F7

SAGE

Boogie

SLAyer

Microsoft Research

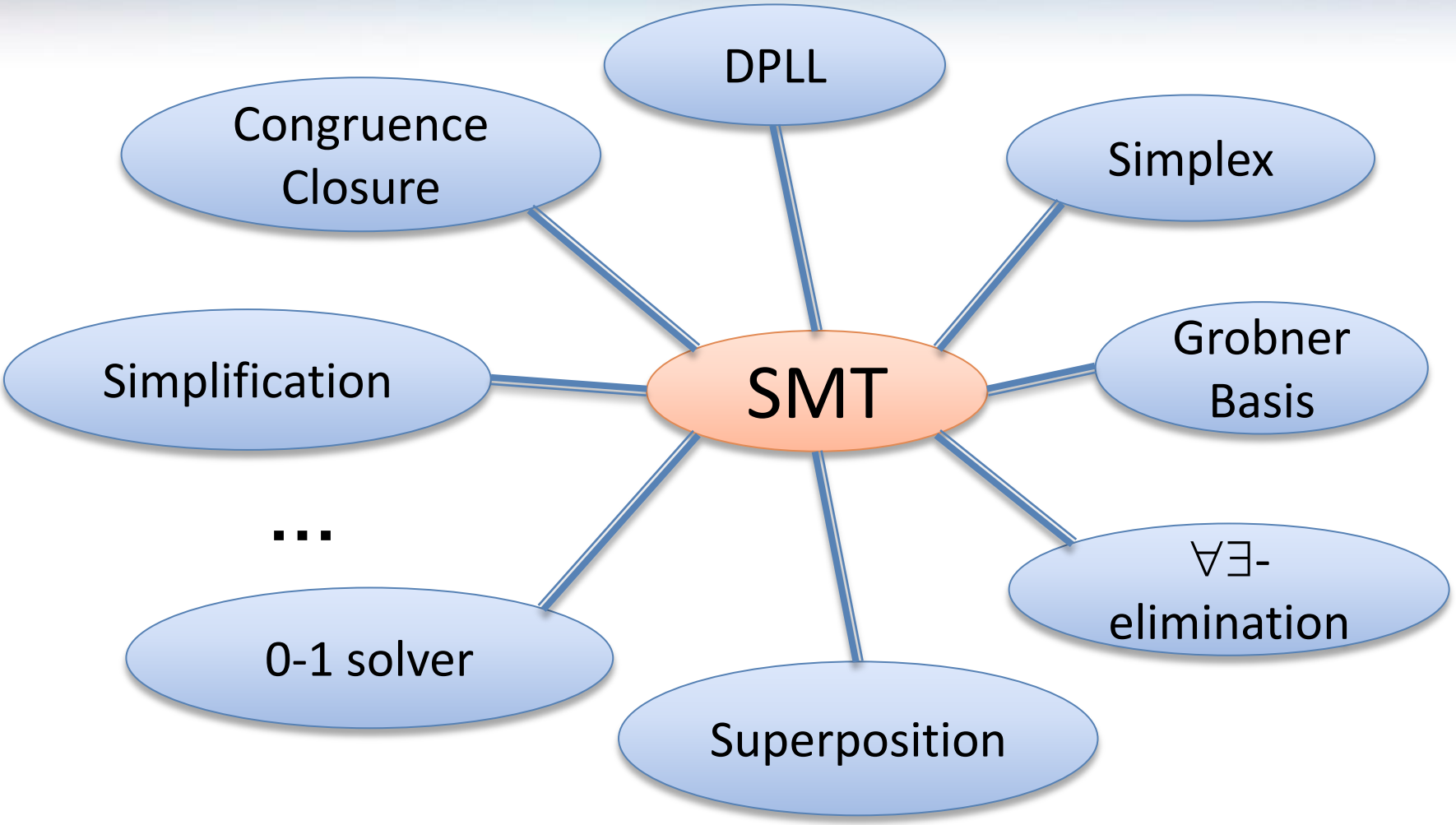
Many external users

- > 15k downloads (Windows version)

New Domains → New Requirements

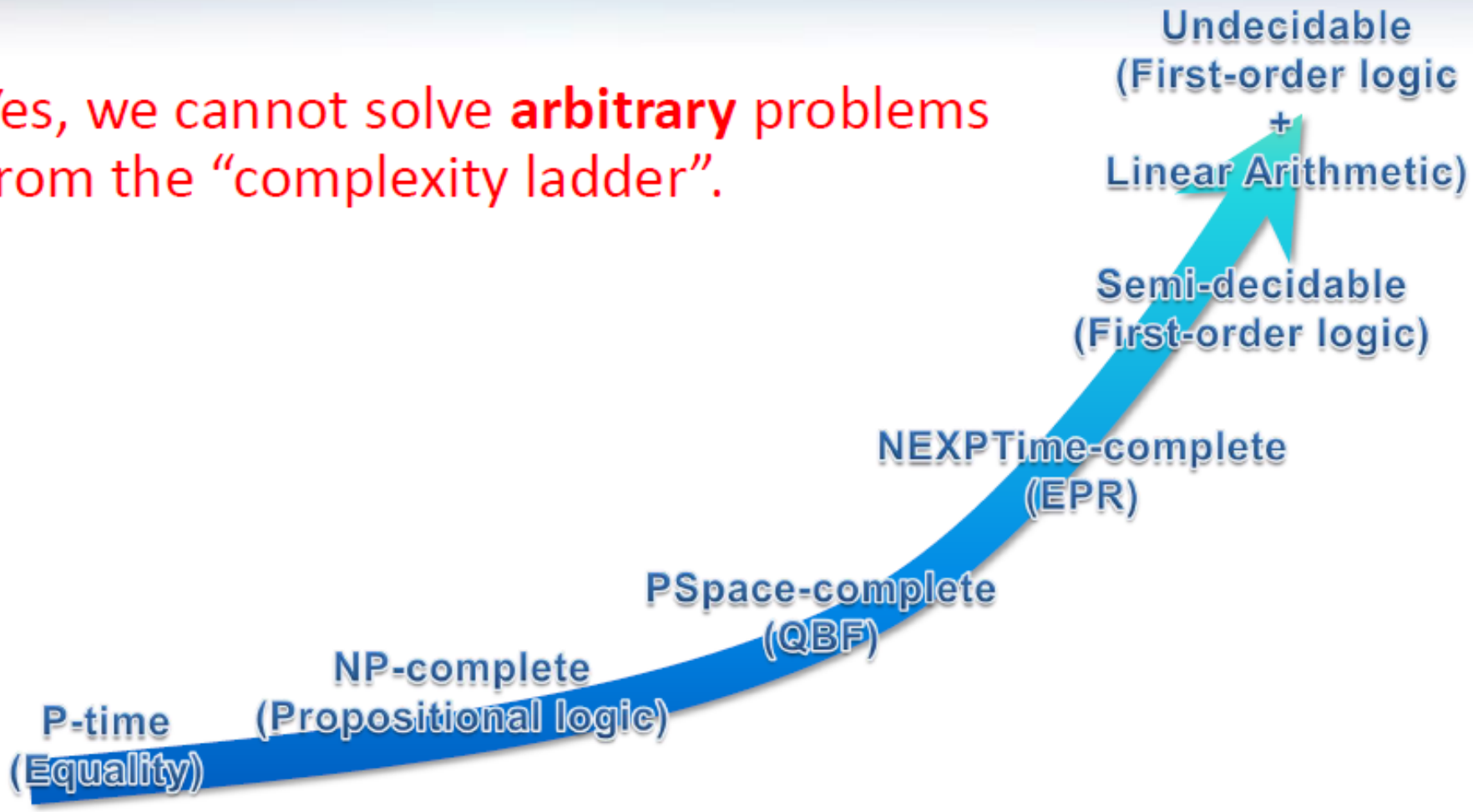
Microsoft Research

Many engines



Symbolic Reasoning

Yes, we cannot solve **arbitrary** problems from the "complexity ladder".



Microsoft Research

The Need for “Strategies”

Different Strategies for Different Domains.

The Need for "Strategies"

Different Strategies for Different Domains.

From timeout to 0.05 secs…

Example in Quantified Bit-Vector Logic (QBVF)

Join work with C. Wintersteiger and Y. Hamadi
FMCAD 2010

QBVF = Quantifiers + Bit-vectors + uninterpreted functions

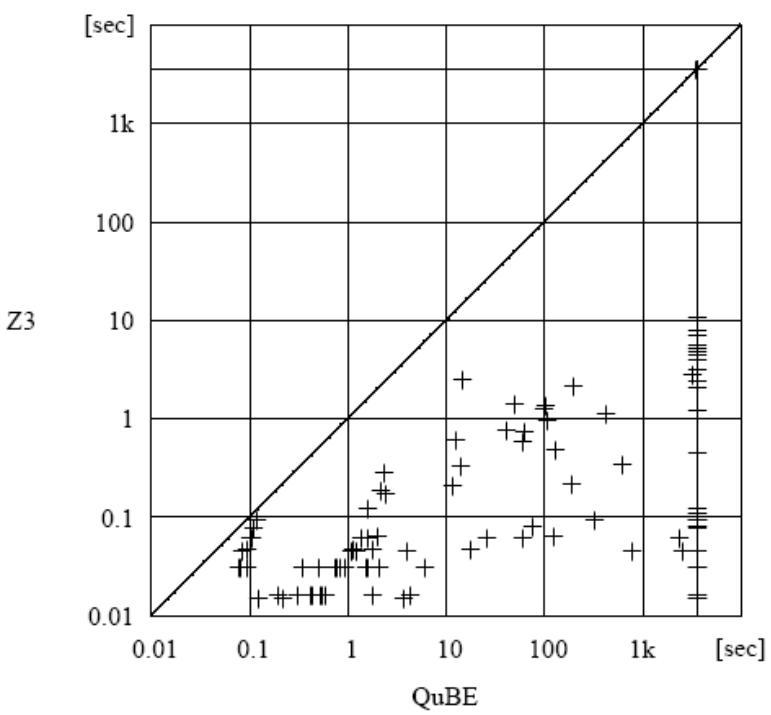
Hardware Fixpoint Checks.

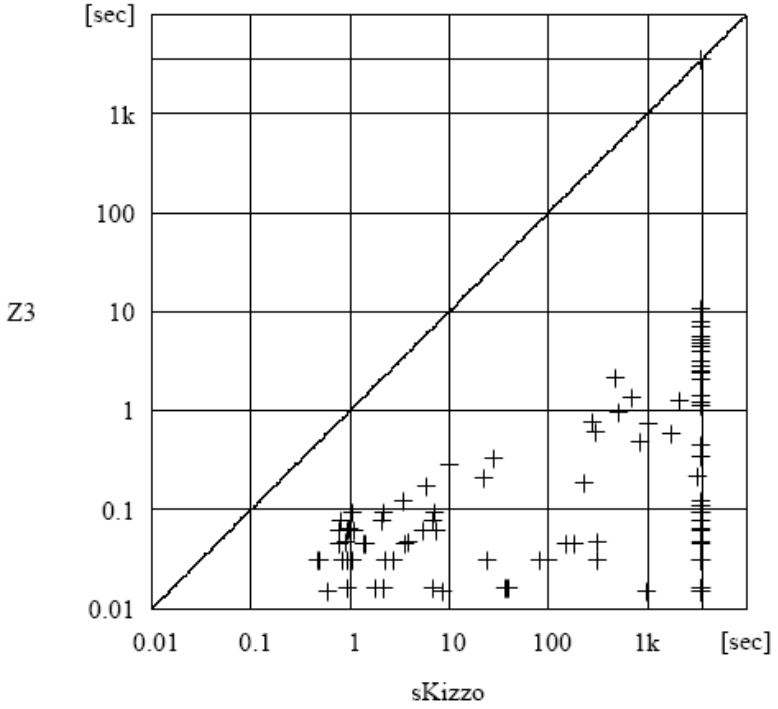
Given: $I[x]$ and $T[x, x']$

$$\forall x, x' \ . \ I[x] \wedge T^k[x, x'] \rightarrow \exists y, y' \ . I[y] \wedge T^{k-1}[y, y']$$

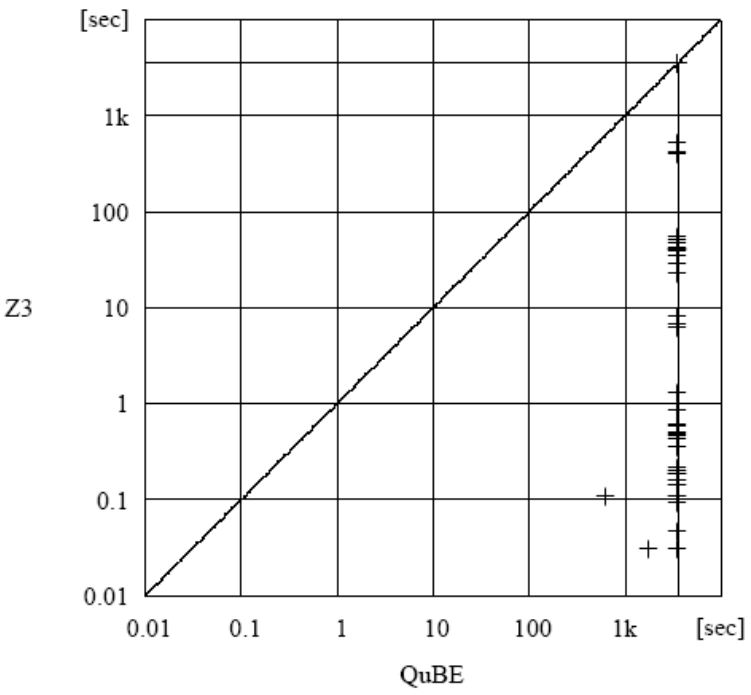
Ranking function synthesis.

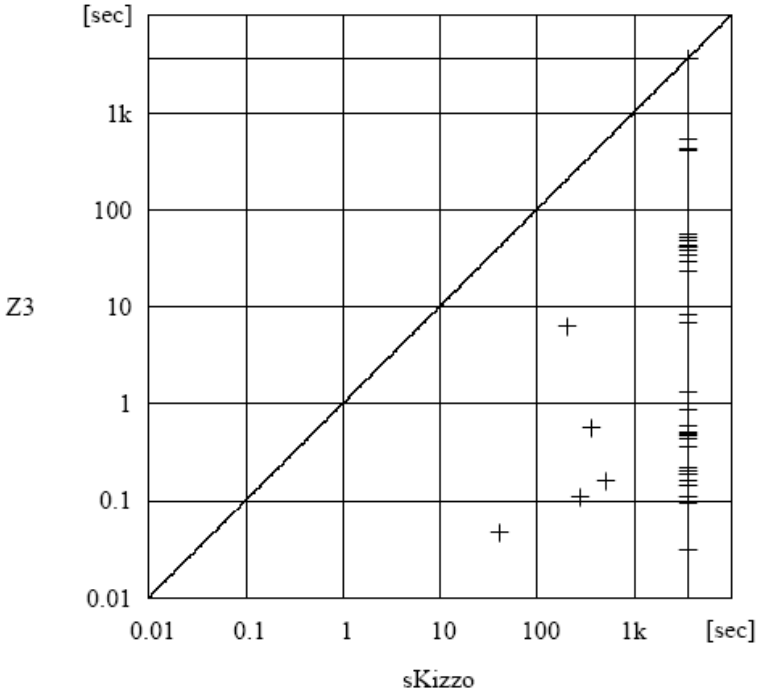
Hardware Fixpoint Checks





Ranking Function Synthesis





Why is Z3 so fast in these benchmarks?

Z3 is using different engines:

rewriting, simplification, model checking, SAT, …

Z3 is using a customized **strategy**.

We could do it because

we have access to the source code.

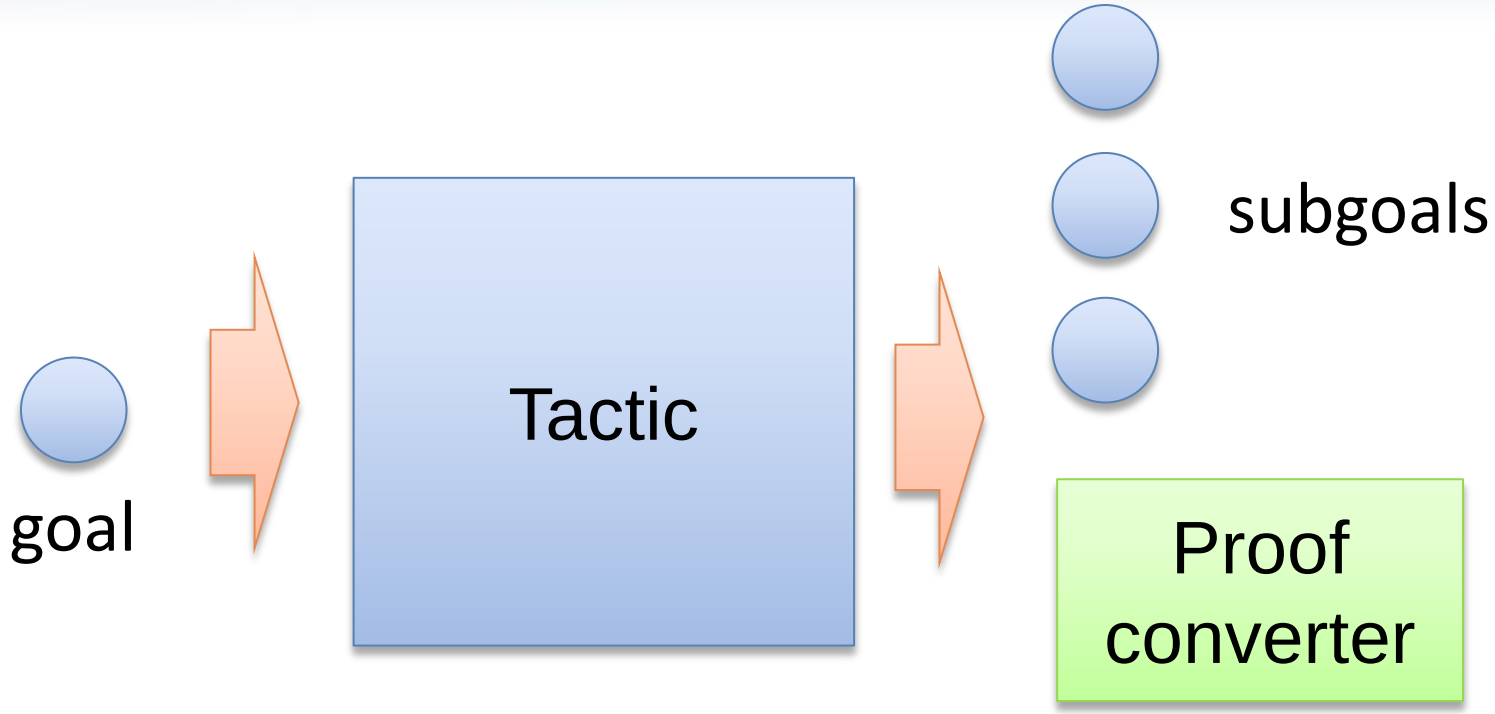
The "Message"

SMT solvers are collections of engines.

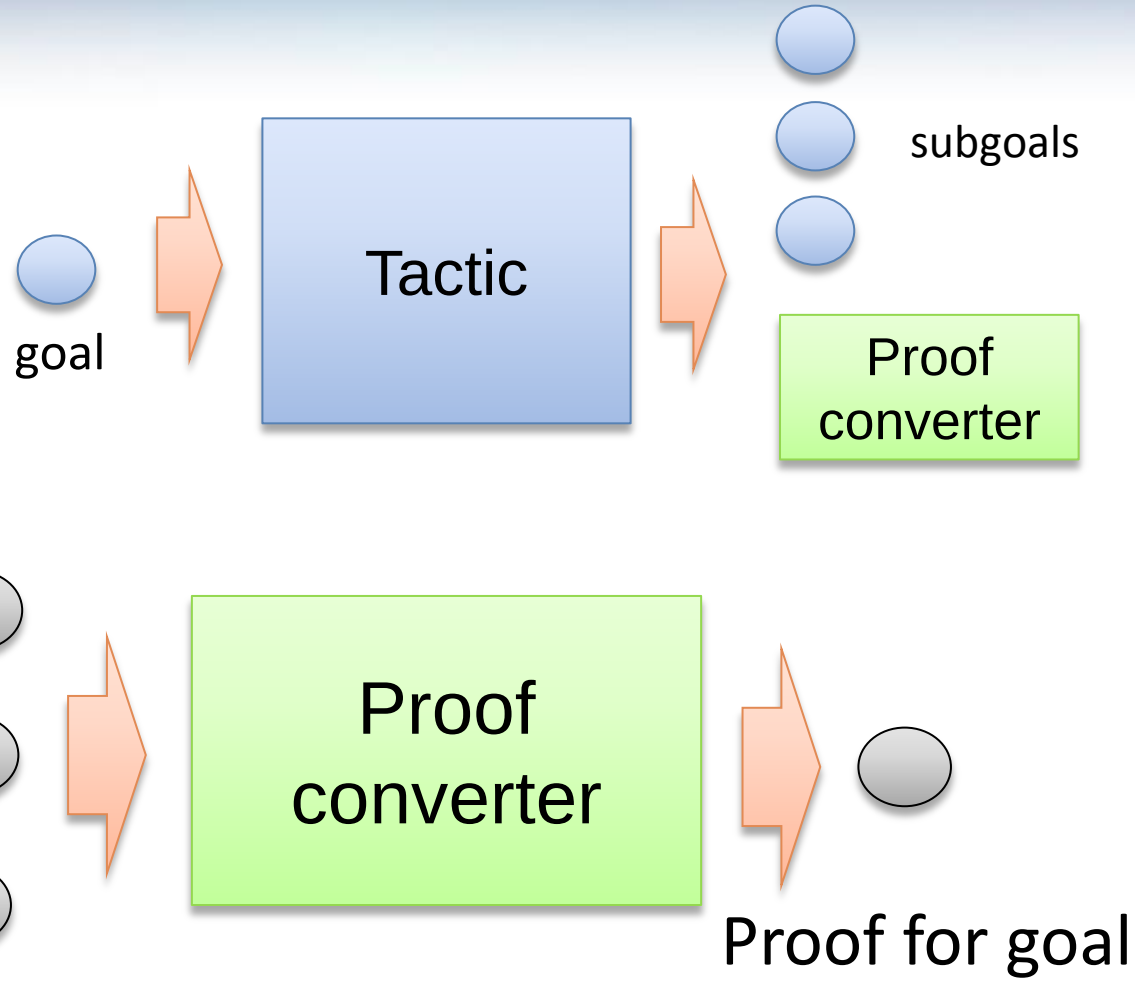
They should provide access to these engines.

Users should be able to define their own strategies.

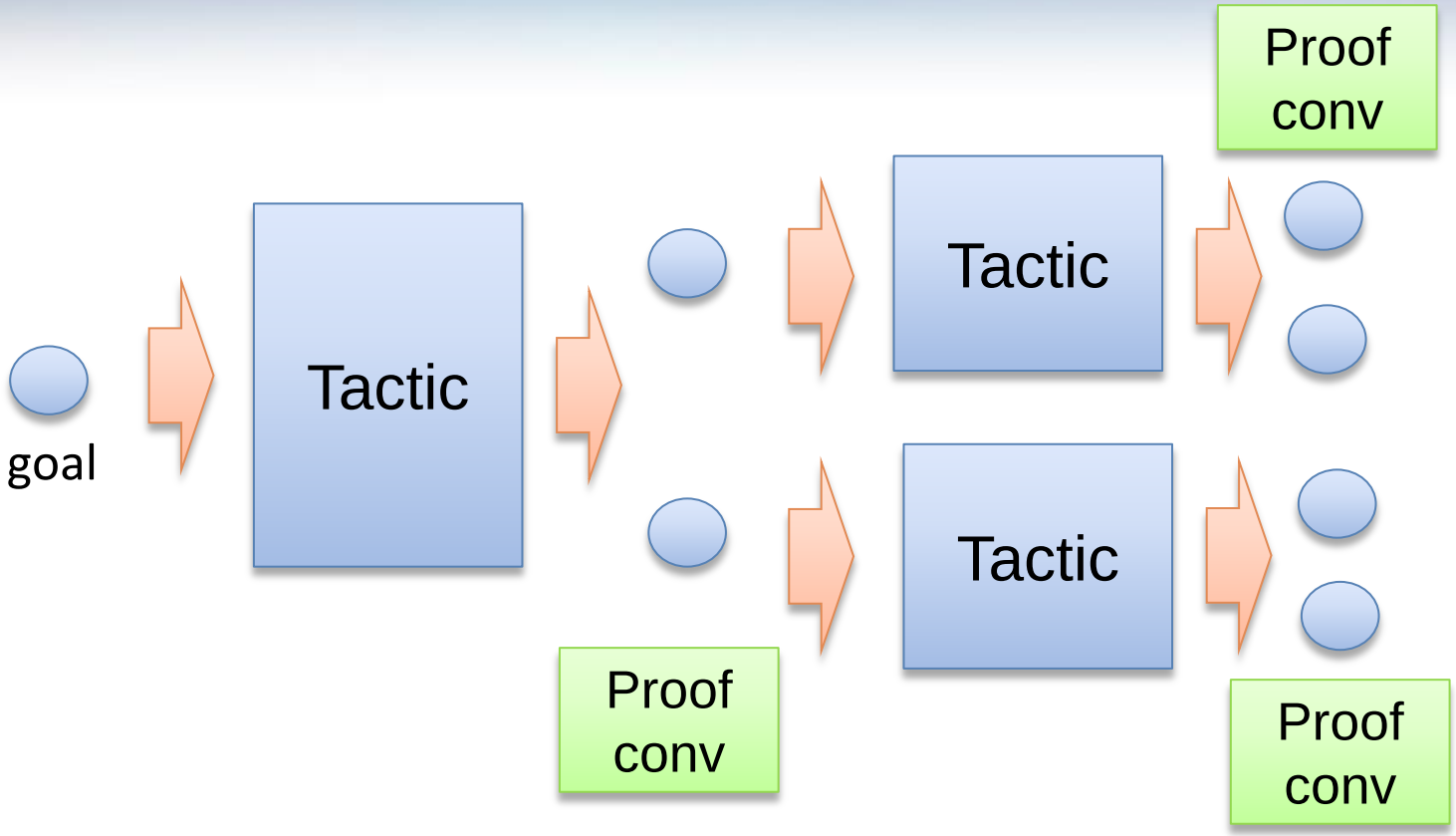
Main inspiration: LCF-approach



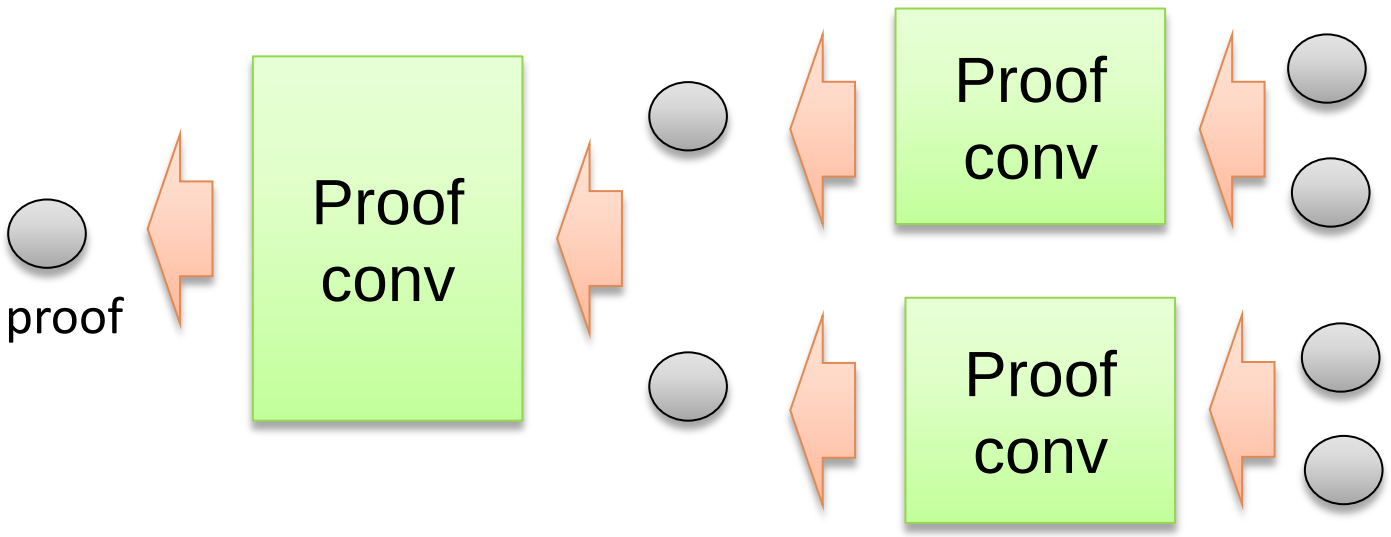
Main inspiration: LCF-approach



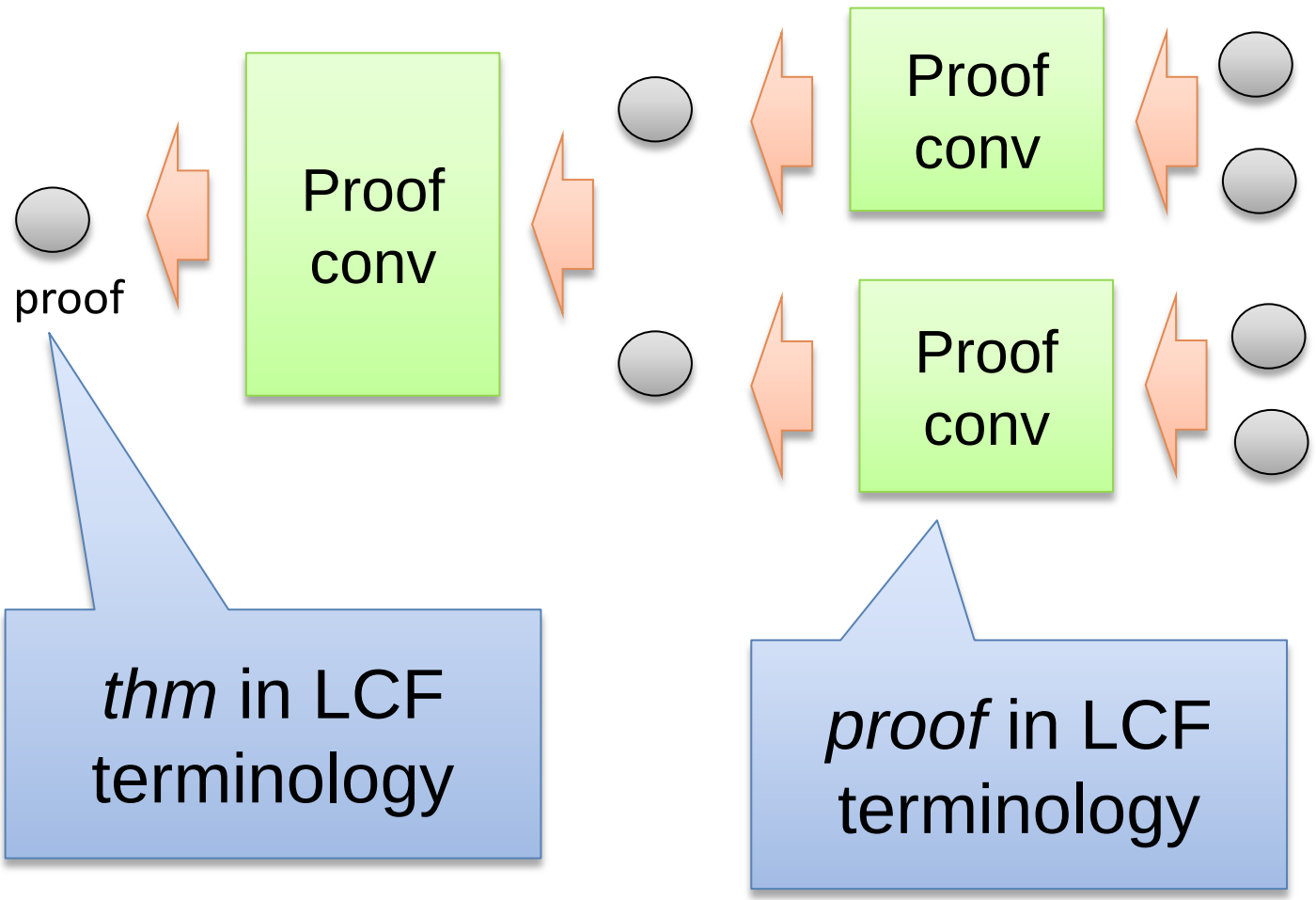
Main inspiration: LCF-approach



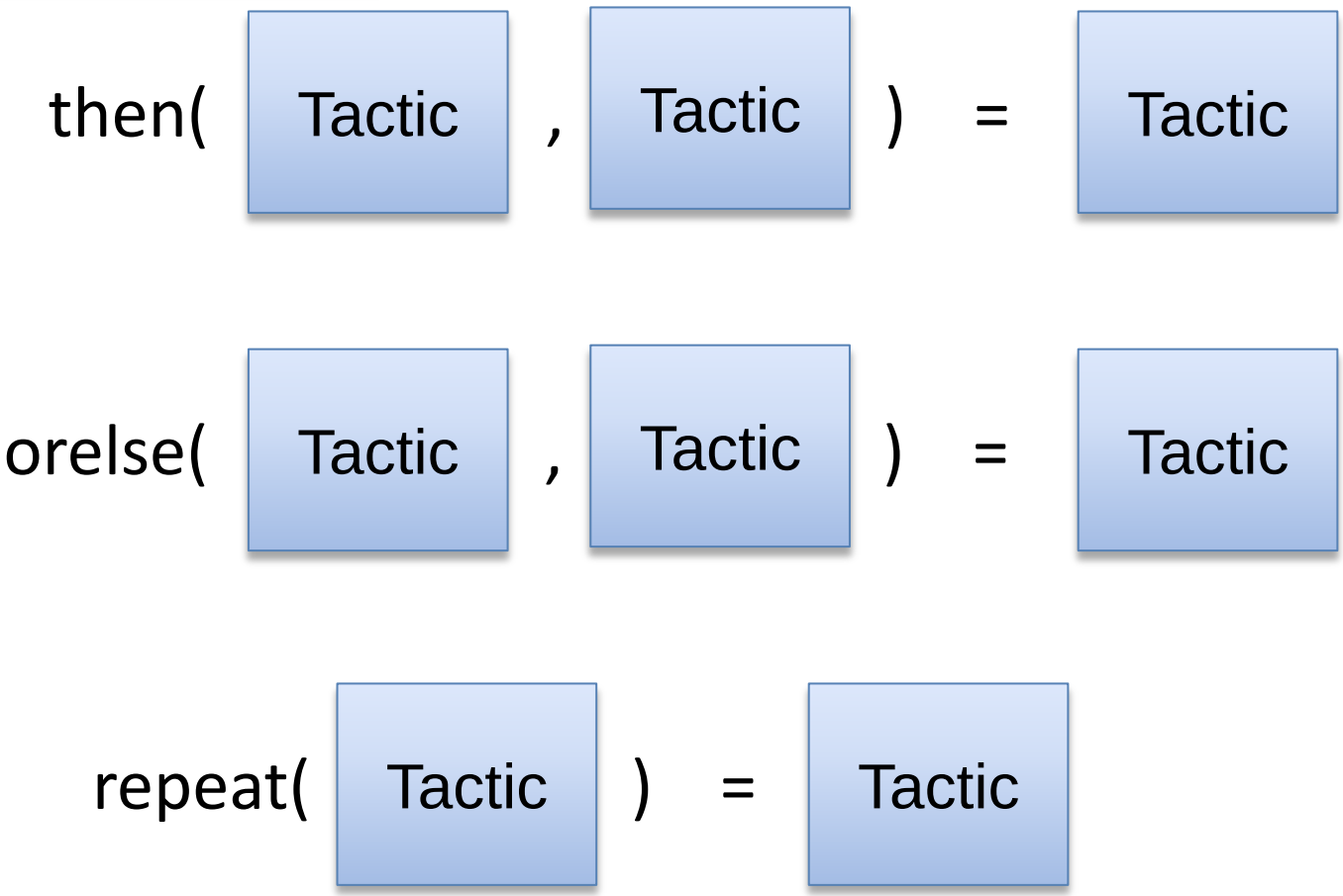
Main inspiration: LCF-approach



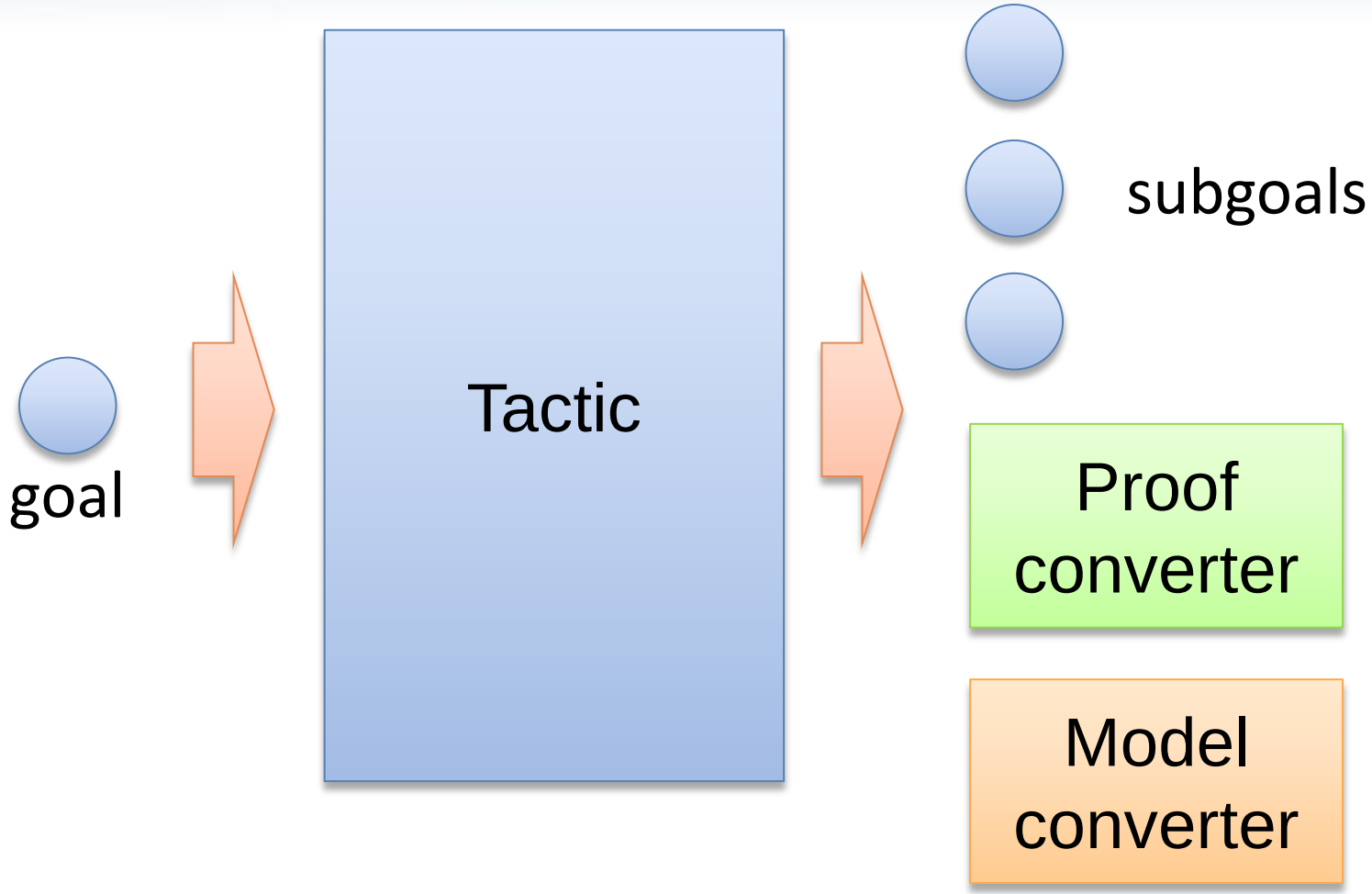
Main inspiration: LCF-approach



Tacticals aka Combinators



SMT Tactic



SMT Tactic example

$$[\ a = b + 1,\ (a < 0 \vee a > 0),\ b > 3\ ]$$

Tactic:
elim-vars

Proof converter

$$[\ (b + 1 < 0 \vee b + 1 > 0),\ b > 3\ ]$$

Model converter

SMT Tactic example

$$[\ a = b + 1,\ (a < 0 \vee a > 0),\ b > 3\ ]$$

Tactic:
elim-vars

$$[\ (b + 1 < 0 \vee b + 1 > 0),\ b > 3\ ]$$

Proof
converter

M, M(a) = M(b) + 1

Model
converter

M

SMT Tactic example

$$[\ a = b + 1,\ (a < 0 \vee a > 0),\ b > 3\ ]$$

Tactic:
split-clause

$$[\ a = b + 1,\ a < 0,\ b > 3\ ]$$

$$[\ a = b + 1,\ a > 0,\ b > 3\ ]$$

Proof converter

Model converter

SMT Tactics

simplify
nnf
cnf
tseitin
lift-if
bitblast
gb
vts

propagate-bounds
propagate-values
split-ineqs
split-eqs
rewrite
p-cad
sat
solve-eqs

SMT Tacticals

then : $(tactic \times tactic) \rightarrow tactic$

then(t_1, t_2) applies t_1 to the given goal and t_2 to every subgoal produced by t_1.

then$*$: $(tactic \times tactic\ sequence) \rightarrow tactic$

then$*(t_1, [t_{2_1}, ..., t_{2_n}])$ applies t_1 to the given goal, producing subgoals $g_1, ..., g_m$. If $n \neq m$, the tactic fails. Otherwise, it applies t_{2_i} to every goal g_i.

orelse : $(tactic \times tactic) \rightarrow tactic$

orelse(t_1, t_2) first applies t_1 to the given goal, if it fails then returns the result of t_2 applied to the given goal.

par : $(tactic \times tactic) \rightarrow tactic$

par(t_1, t_2) excutes t_1 and t_2 in parallel.

SMT Tacticals

$$\mathsf{then}(\mathsf{skip}, t) = \mathsf{then}(t, \mathsf{skip}) = t$$

$$\mathsf{orelse}(\mathsf{fail}, t) = \mathsf{orelse}(t, \mathsf{fail}) = t$$

SMT Tacticals

repeat : $tactic \rightarrow tactic$

Keep applying the given tactic until no subgoal is modified by it.

repeatupto : $tactic \times nat \rightarrow tactic$

Keep applying the given tactic until no subgoal is modified by it, or the maximum number of iterations is reached.

tryfor : $tactic \times seconds \rightarrow tactic$

tryfor(t, k) returns the value computed by tactic t applied to the given goal if this value is computed within k seconds, otherwise it fails.

Feature / Measures

Probing structural features of formulas.

Under/Over-Approximations

Under-approximation

unsat answers cannot be trusted

Over-approximation

sat answers cannot be trusted

Under/Over-Approximations

Under-approximation

model finders

Over-approximation

proof finders

Under/Over-Approximations

Under-approximation

Example: QF_NIA model finders

add bounds to unbounded variables (and blast)

Over-approximation

Example: Boolean abstraction

Tactics: supported features

Models

Proofs

Unsat cores

No support for incremental solving

Workaround:

use “all-purpose” incremental solver with timeout

switch to “domain-specific” tactic if can’t solve in *k* ms

Microsoft Research

New Arithmetic Solvers

- Current technology
  - Linear Real: Simplex based
  - Linear Integer: Gomory-Cuts, Branching, **Model Finder**, **0-1**
  - Nonlinear:

    Grobner Basis, Branching, Interval Propagation,

    Model Finder, **VTS**

Microsoft Research

New Arithmetic Solvers

- New
  - Linear Real: New Simplex based on state-of-the-art techniques from MIP
  - Linear Integer: cutsat algorithm (complete)
  - Nonlinear: cad-like (complete) algorithm, algebraic numbers

    x^5 + 2x + 2, [-0.8175, -0.8174]

Microsoft Research

Floating-Point Arithmetic

- New solver combining ideas from:
  - Interval Propagation
  - Reduction to arithmetic
  - Bit-blasting
- Main target: verification problems
  - No overflows
  - No NaN
  - Error estimation

Microsoft Research

Interpolants

- No plans to support them in Z3
- News:

  Ken McMillan built a new interpolating prover on top of Z3

  Very good experimental results (FMCAD'11)

Microsoft Research

Z3 API: an update

- Memory management
  - Old: scoped memory management

    Hack: Z3_persist_ast
  - New: ref-count

    Z3_mk_context_rc, Z3_inc_ref, Z3_dec_ref

- Logging
  - Old: Z3_open_log (.z3 file), Z3_trace_to_file (.c file)
  - New: global Z3_open_log

Microsoft Research

Z3 API: an update

- Multiple Solvers & Tactics
  - Many solvers per Context
  - Incremental & Non-incremental Solvers
  - Solvers based on user-provided Tactics
  - We will continue to support the old API

Microsoft Research

Z3 Guide

http://rise4fun.com/z3/tutorial/guide

New features:

Models as “functional programs”

Quantifiers and Modeling with Quantifiers

Model Based Quantifier Instantiation

Microsoft Research

Microsoft Research

BV NEWS

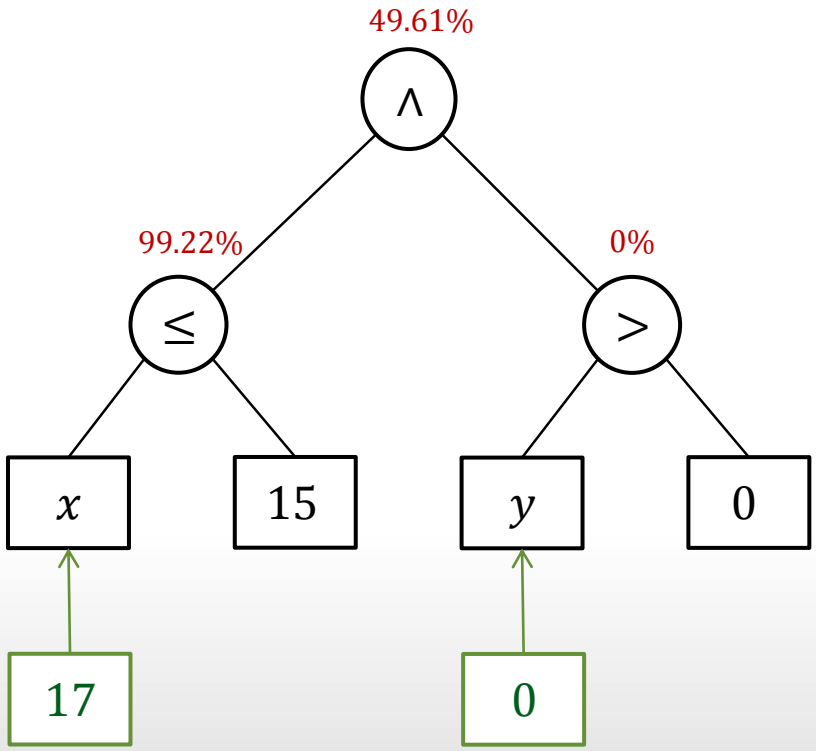
Microsoft

News about Bit-Vectors

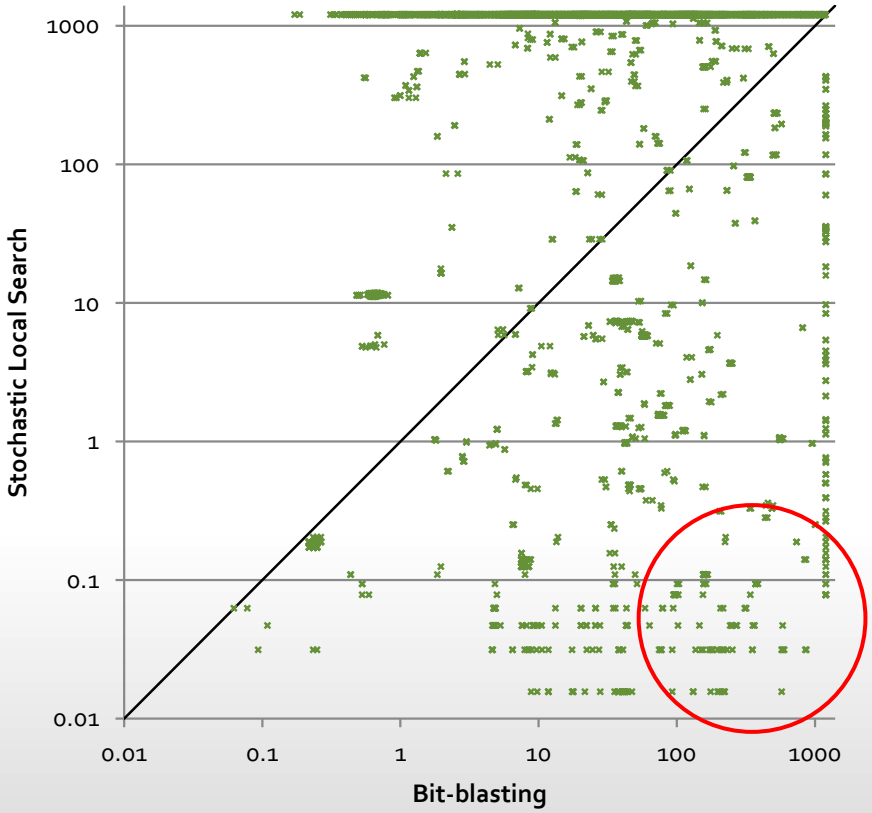
- Bit-blasted formulas are large
  - Sometimes unnecessary
- What we're looking into
  - Stochastic Local Search
  - Custom Propagators

Stochastic Local Search

- Random model
- Improve locally
  - Bit-flips
  - ± 1
  - Negation
  - Et cetera
- Scoring system
  - Relative Satisfaction



First Results



Quantification

- MBQI for (UF)BV
  - Complete and efficient method
  - Logic being added to SMT-LIB
  - `(set-logic UFBV)`

- Finite-state problems
  - Model Checking
  - Synthesis