Universal Z3: a model finder for quantified SMT formulas

Leonardo de Moura

Introduction

Quantified SMT formulas.

Applications: synthesis, software verification, ...

forall x. f(x, x) >= x+a,

f(a, b) < a, a > 0

Models as functional programs.

f (x1, x2) = if (x1 = 1 and x2 = 2) then 0 else x1 + 1

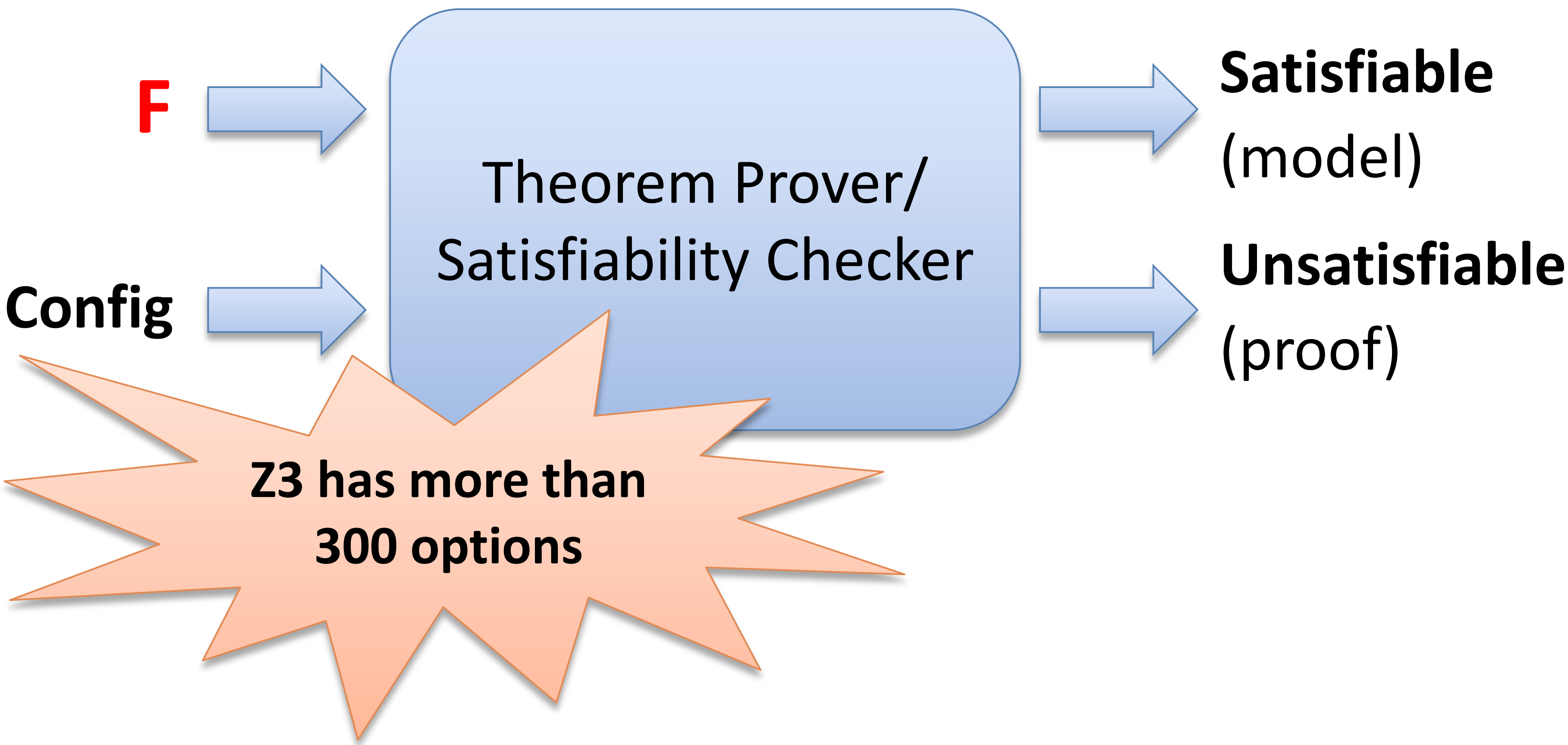
Online demo at the Z3 website.

Orchestrating Decision Engines

Deduction at Scale, Germany, 2011

Leonardo de Moura and Grant Passmore

Theorem Provers & Satisfiability Checkers

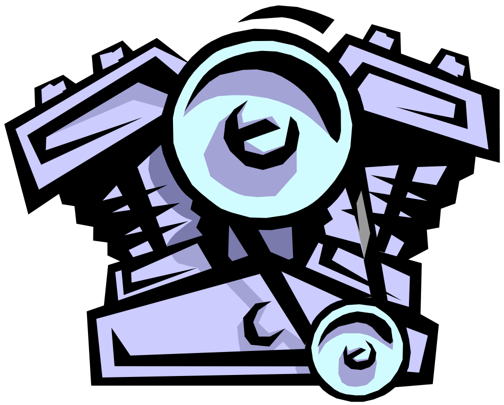


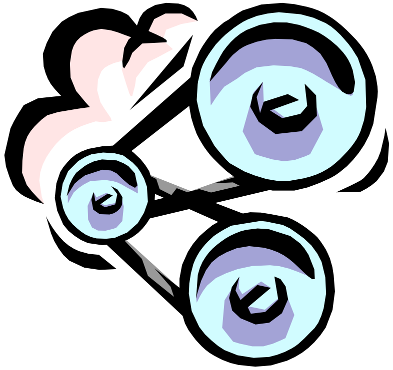
Combining Engines

Current SMT solvers provide

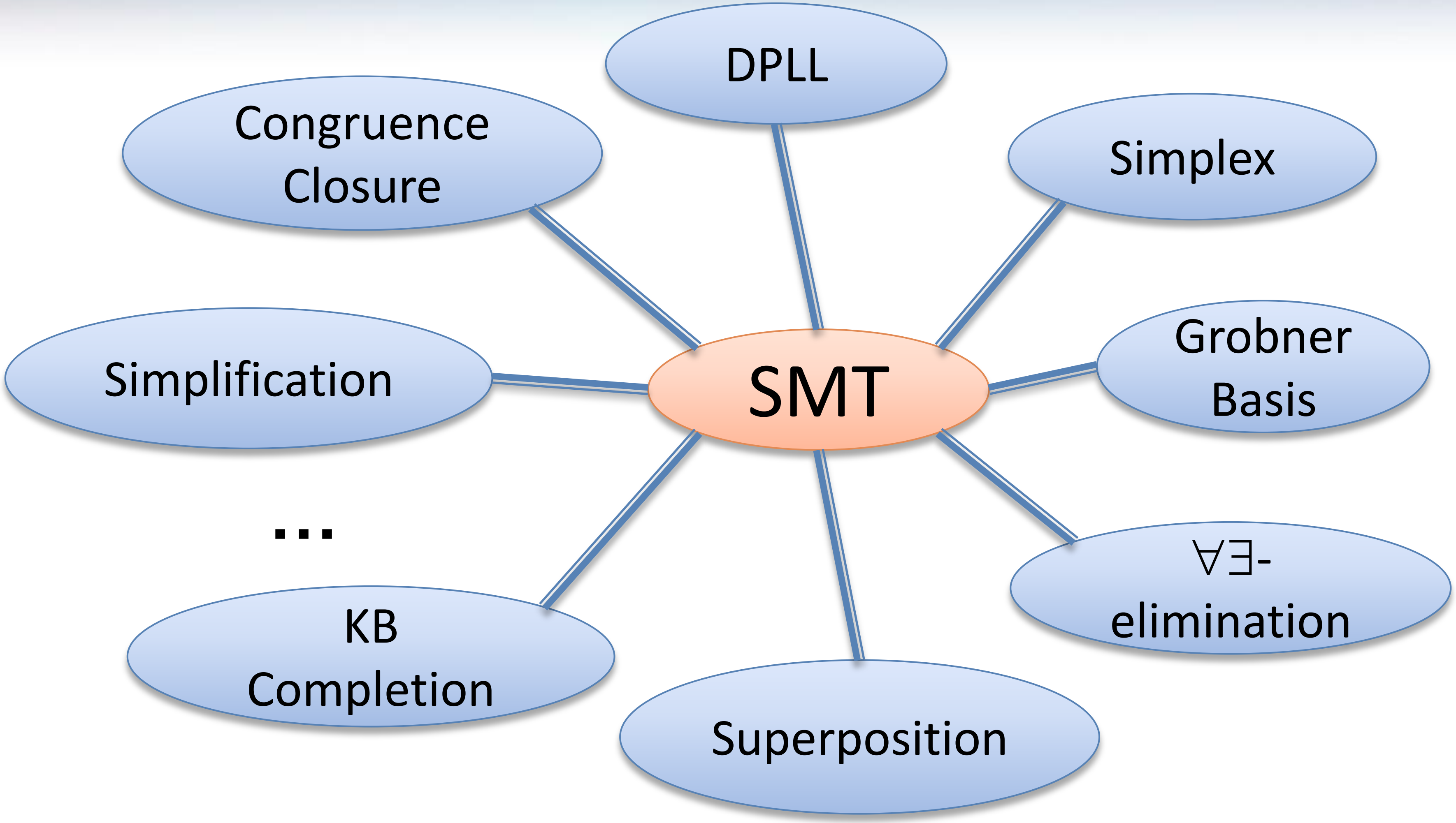
a combination

of different engines





Combining Engines



Opening the "Black Box"

Actual feedback provided by Z3 users:

"Could you send me your CNF converter?"

"I want to implement my own search strategy."

"I want to include these rewriting rules in Z3."

"I want to apply a substitution to term *t*."

"I want to compute the set of implied equalities."

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

Is
F1 and F2
Sat?

Is
F1 and F3
Sat?

Push, Assert, Check, Pop

Popularized by SMT solvers such as: Simplify.

Part of SMT-LIB 2.0 standard.

push, assert(F1), push, assert ... F3), check

Users need more than that!

Is
F1 and F2
Sat?

Is
F1 and F3
Sat?

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

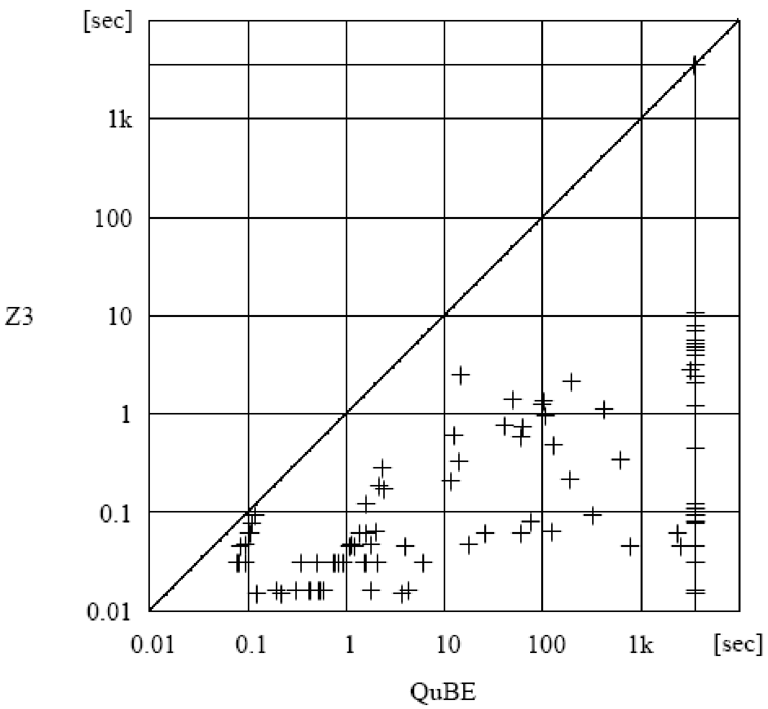
Hardware Fixpoint Checks.

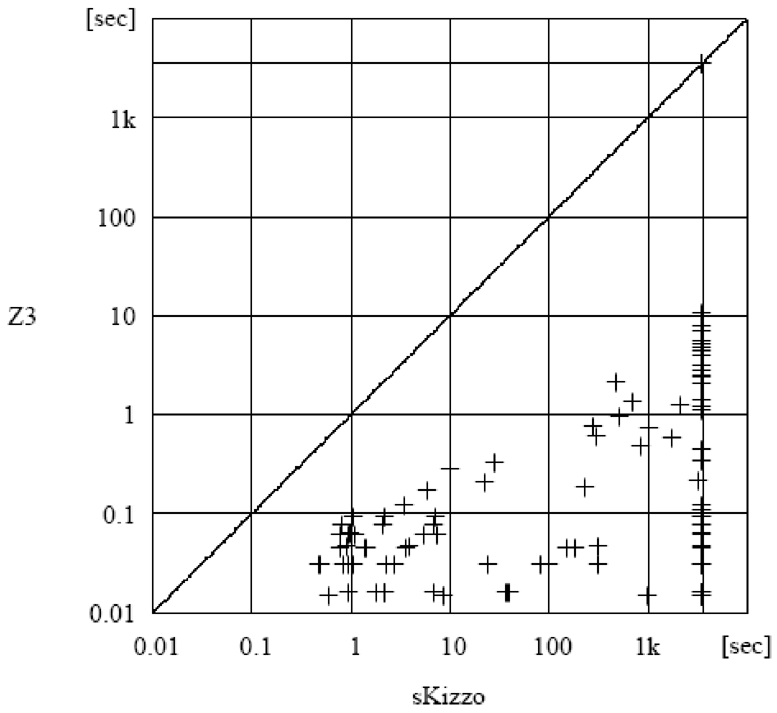
Given: $I[x]$ and $T[x, x']$

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

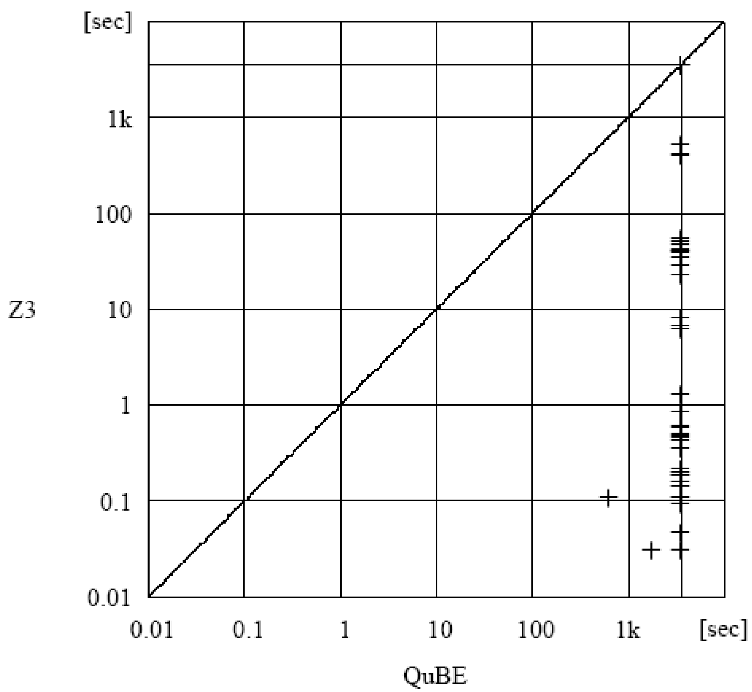
Ranking function synthesis.

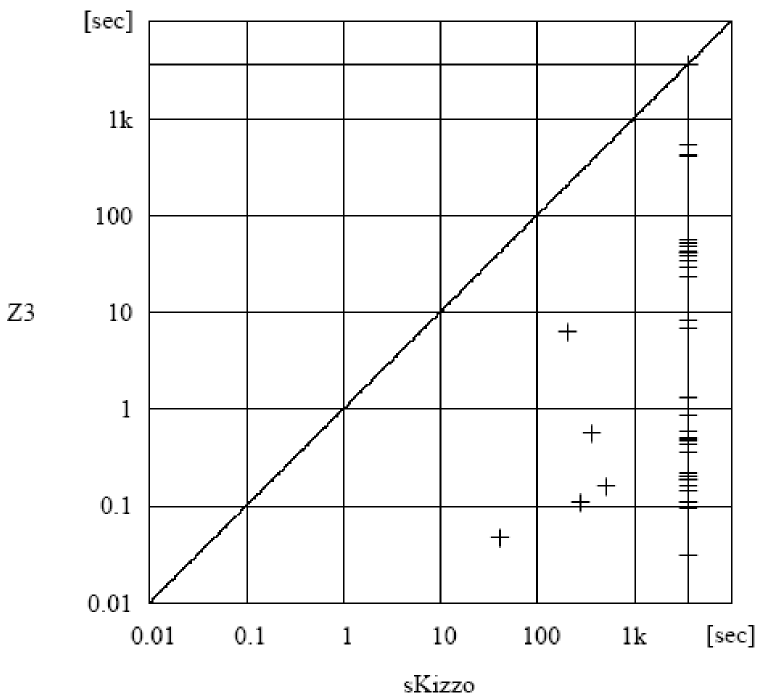
Hardware Fixpoint Checks





Ranking Function Synthesis





Why is Z3 so fast in these examples?

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 little engines.

They should provide access to these engines.

Users should be able to define their own strategies.

Our Proposal

Inspired by ideas from:

Interactive Theorem Proving: Tactics, Goals, …

Rushby's Tool Bus.

Exposing “Little” engines

Simplifier

Rewriter

CNF, NNF, SKNF converters

Procedures for:

Quantifier Elimination

Gaussian Elimination

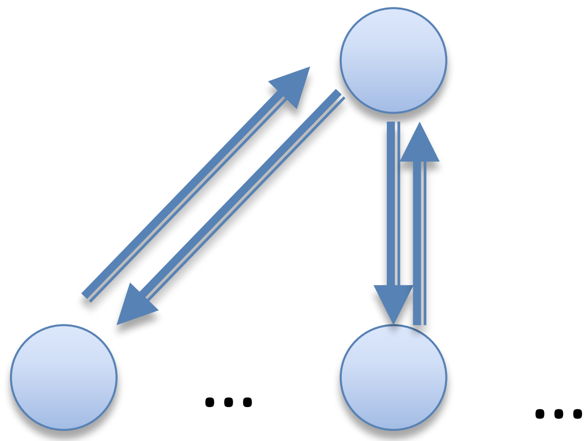
Grobner Basis

Polynomial Factorization

….

Goal & Subgoals

Goal = set of formulas.



A tactic splits a goal in sub-goals.

It also provides a model-builder and a proof-builder.

"Streams" of sub-goals & Approximation

A tactic splits a goal in a "stream" of sub-goals.

The "stream" may be produced on-demand.

It is easy to support over/under approximations.

Probes

In most cases it is not feasible to manually inspect the state of a goal.

Probes provide statistics or abstract views of goals.

High-order Tactics (aka tacticals)

Or tactics that receive other tactics as arguments.

It opens so many possibilities.

Example: Abstract Partial CAD in RAHD
More about that in Paul Jackson’s talk.

Lazy SMT as a strategy

It is based on the "Boolean-Abstraction" Tactic.

AKA (Lazy DNF converter)

$$(a < 2 \vee a > 3) \wedge (\text{not } (a < 2)) \wedge b = a \wedge (b < 2 \vee b > 4)$$

produces the "stream":

$$a > 3 \wedge (\text{not } (a < 2)) \wedge b = a \wedge b < 2$$

$$a > 3 \wedge (\text{not } (a < 2)) \wedge b = a \wedge b > 4$$

DPLL(T) as a strategy

A common idiom in SMT is:

Perform “cheap” theory reasoning during the search.

Perform “expensive” theory reasoning after a full Boolean assignment is produced.

These should be parameters to a more general strategy.

Decision Engines as Web Services

Communication based on SMT-LIB 2.0 format.

+ extensions

Basic capability:

"naming" of formulas, goals, tactics, ... (any entity)

Working in progress: Z3 $\leftrightarrow$ RAHD demo.

Conclusion

Different domains need different strategies.

We must expose the little engines in SMT solvers.

Interaction between different engines is a must.

Users can try their little engines in the context of a much bigger infrastructure.

More transparency.