Tactic, Tacticals and SMT

FBK-IRST, Trento, 2011

Leonardo de Moura and Grant Passmore

Satisfiability Modulo Theories (SMT)

A Satisfiability Checker

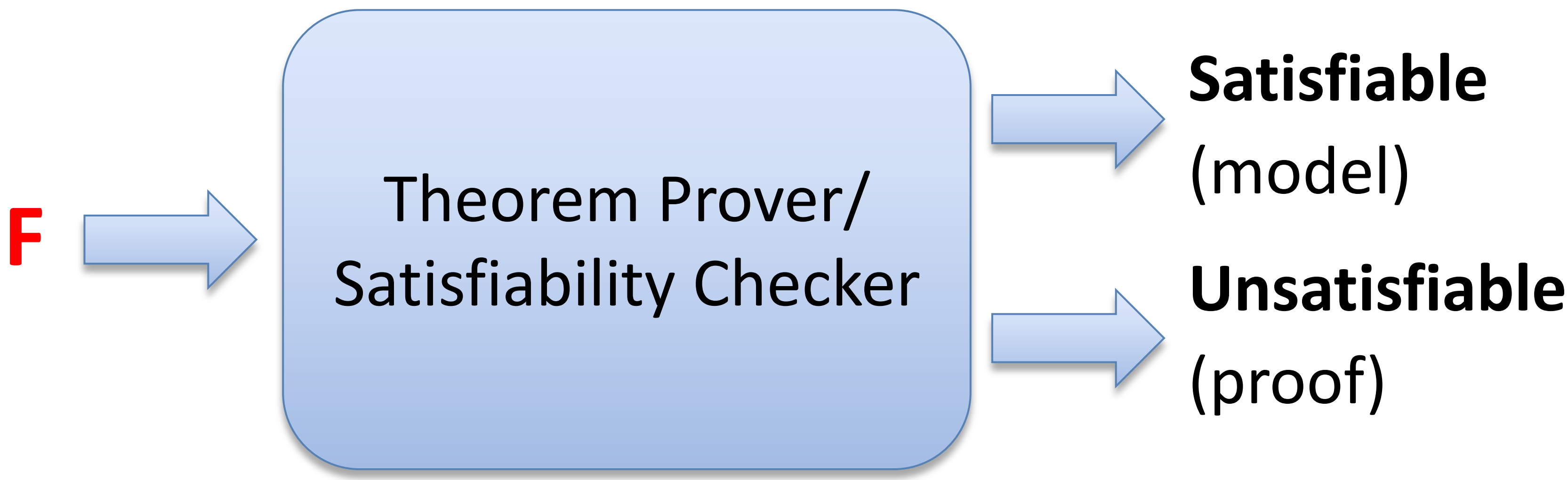
with built-in support for useful theories

Some Applications @ Microsoft

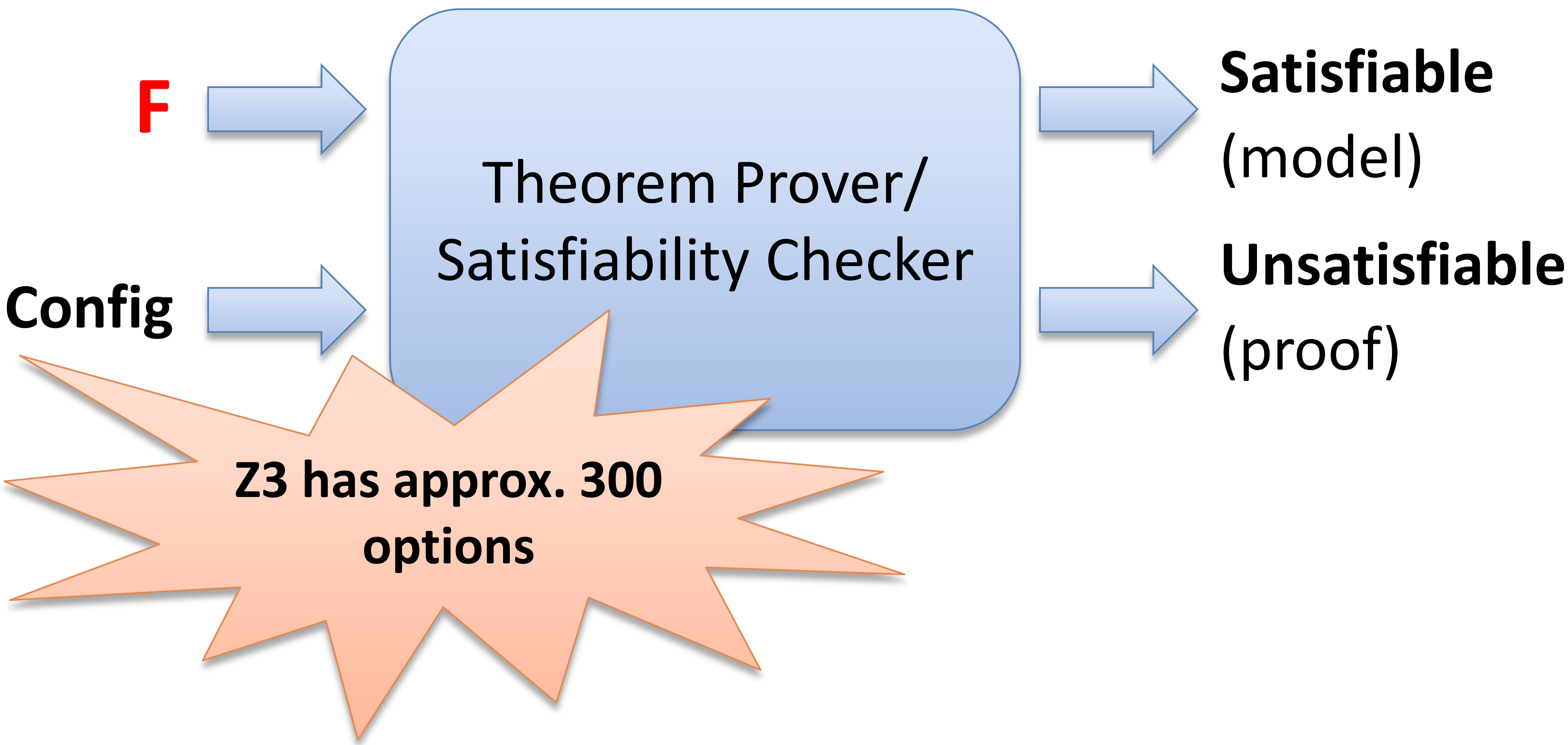
- The Spec# Programming System
- HAVOC
- ForμLa
- Hyper-V
  Microsoft | Virtualization
- Terminator T-2
- VCC
- SLAM
- Yogi
- NModel
- Vigilante
- SpecExplorer
- Pex
- F7
- SAGE

Microsoft Research

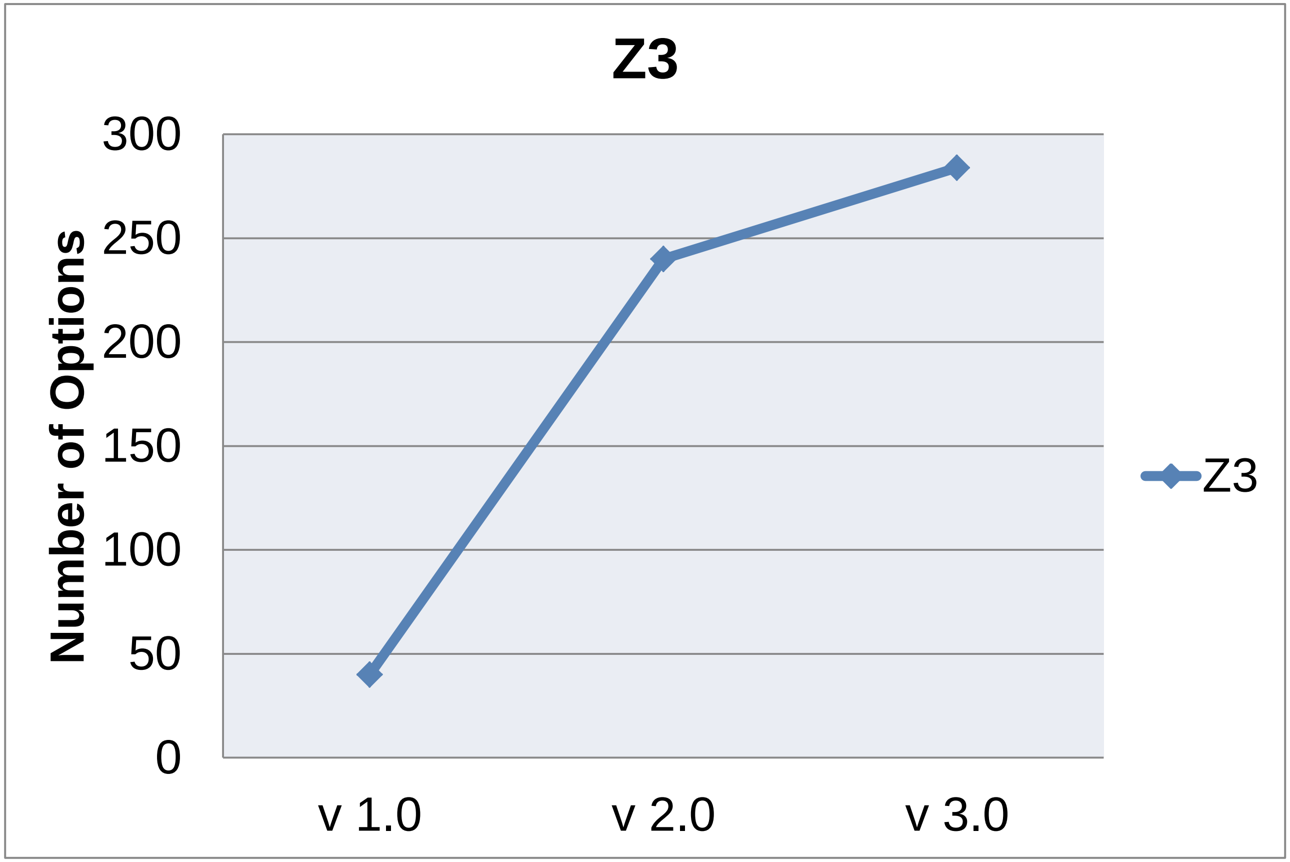
Theorem Provers & Satisfiability Checkers



Theorem Provers & Satisfiability Checkers



Z3 number of options evolution

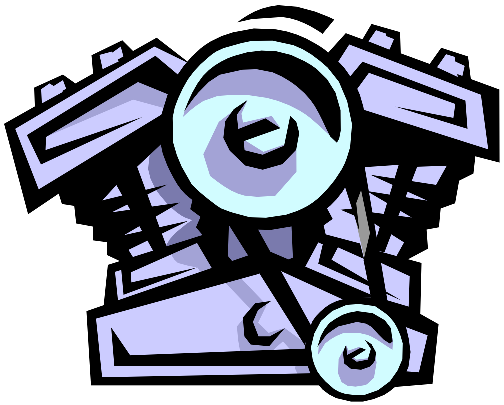


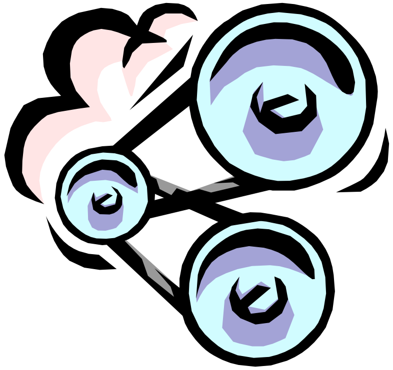
Combining Engines

Current SMT solvers provide

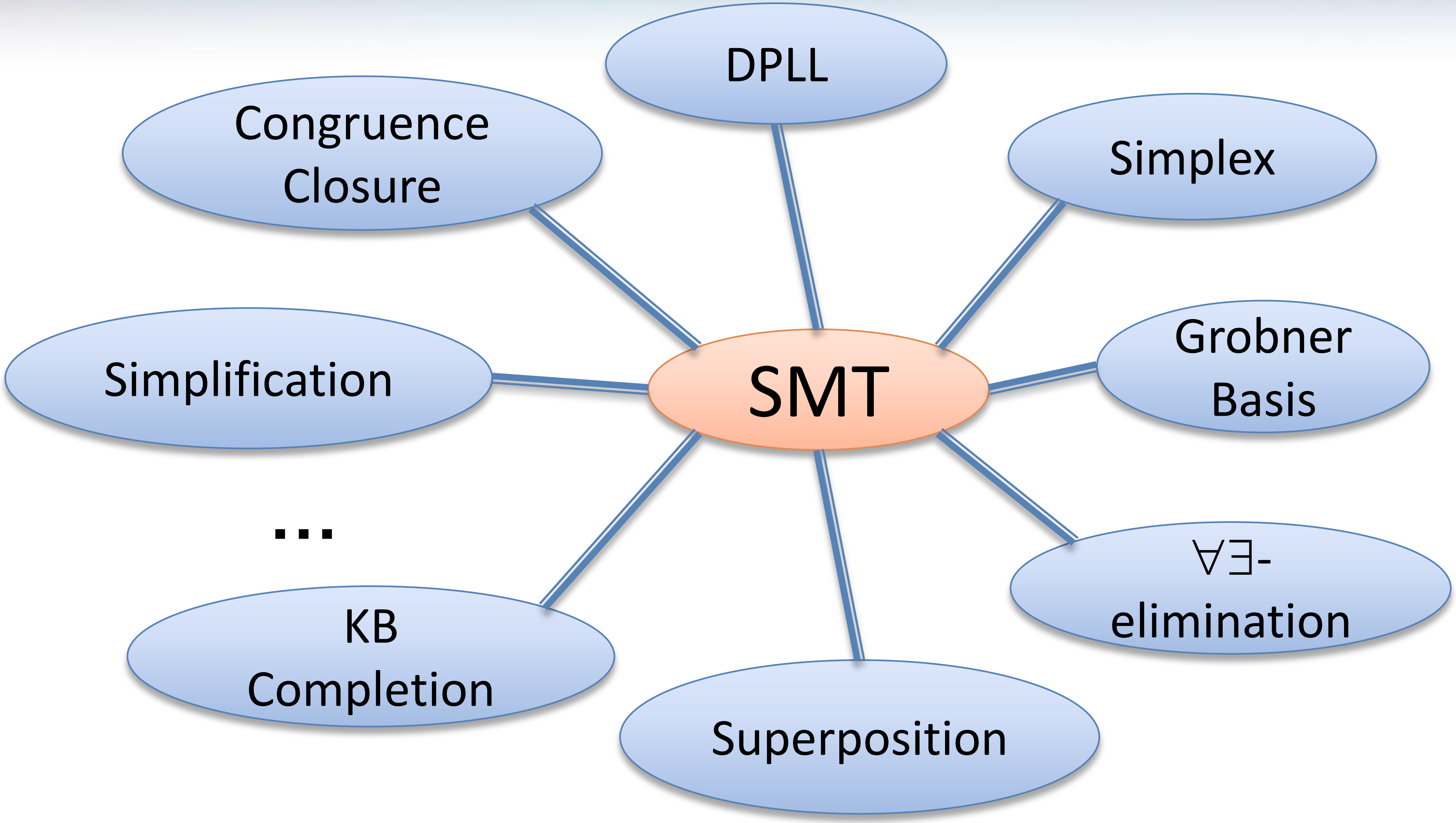
a combination

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."

The Strategy Challenge

To build theoretical and practical tools allowing users to exert strategic control over core heuristic aspects of high performance SMT solvers.

What a strategy is?

Theorem proving as an exercise of combinatorial search

Strategy are adaptations of general search mechanisms which reduce the search space by tailoring its exploration to a particular class of formulas.

Even though one could illustrate how much more effective partial strategies can be if we had only a very dreadful general algorithm, it would appear desirable to postpone such considerations till we encounter a more realistic case where there is no general algorithm or nor efficient general algorithm, e.g., in the whole predicate calculus or in number theory. As the interest is presumably in seeing how well a particular procedure can enable us to prove theorems on a machine, it would seem preferable to spend more effort on choosing the more efficient methods rather than on enunciating more or less familiar generalities.

Hao Wang, 1958

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

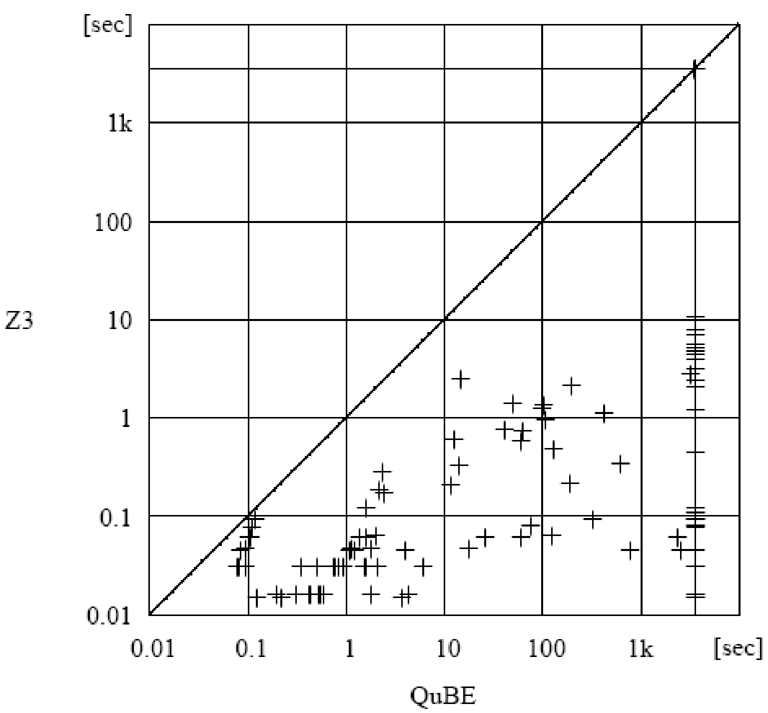
Hardware Fixpoint Checks.

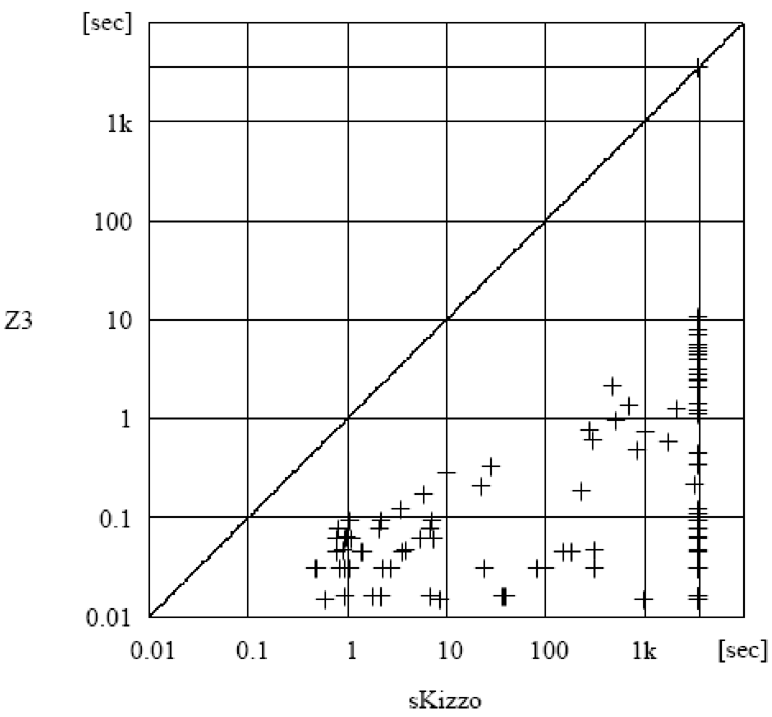
Given: $I[x]$ and $T[x, x']$

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

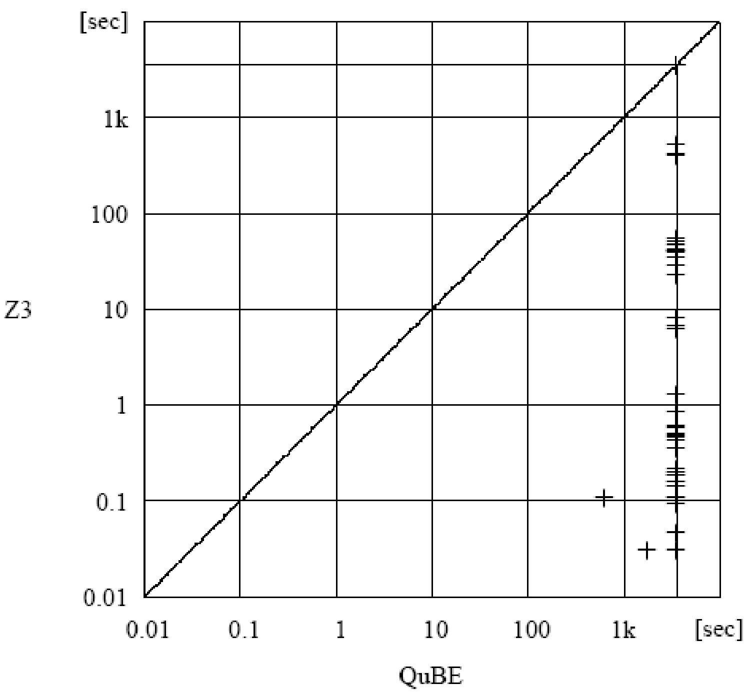
Ranking function synthesis.

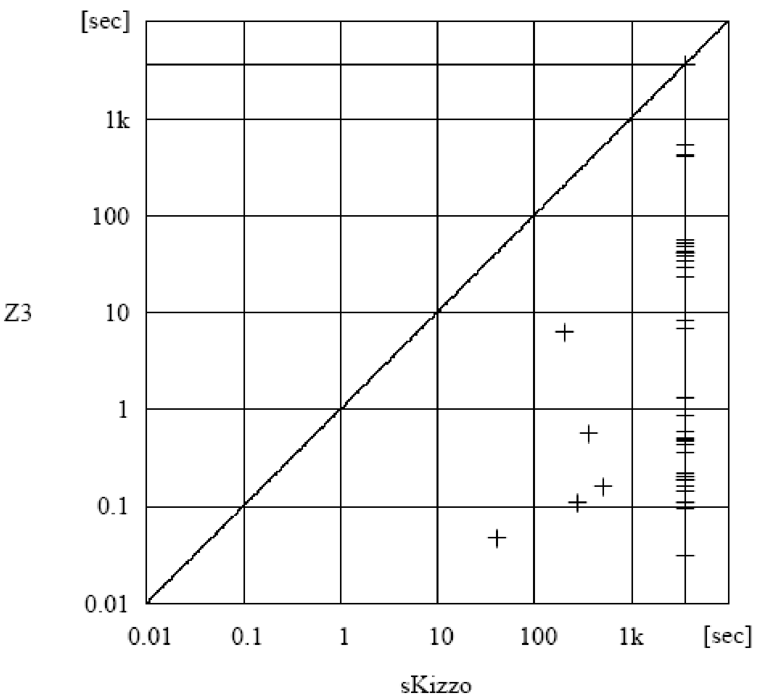
Hardware Fixpoint Checks





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.

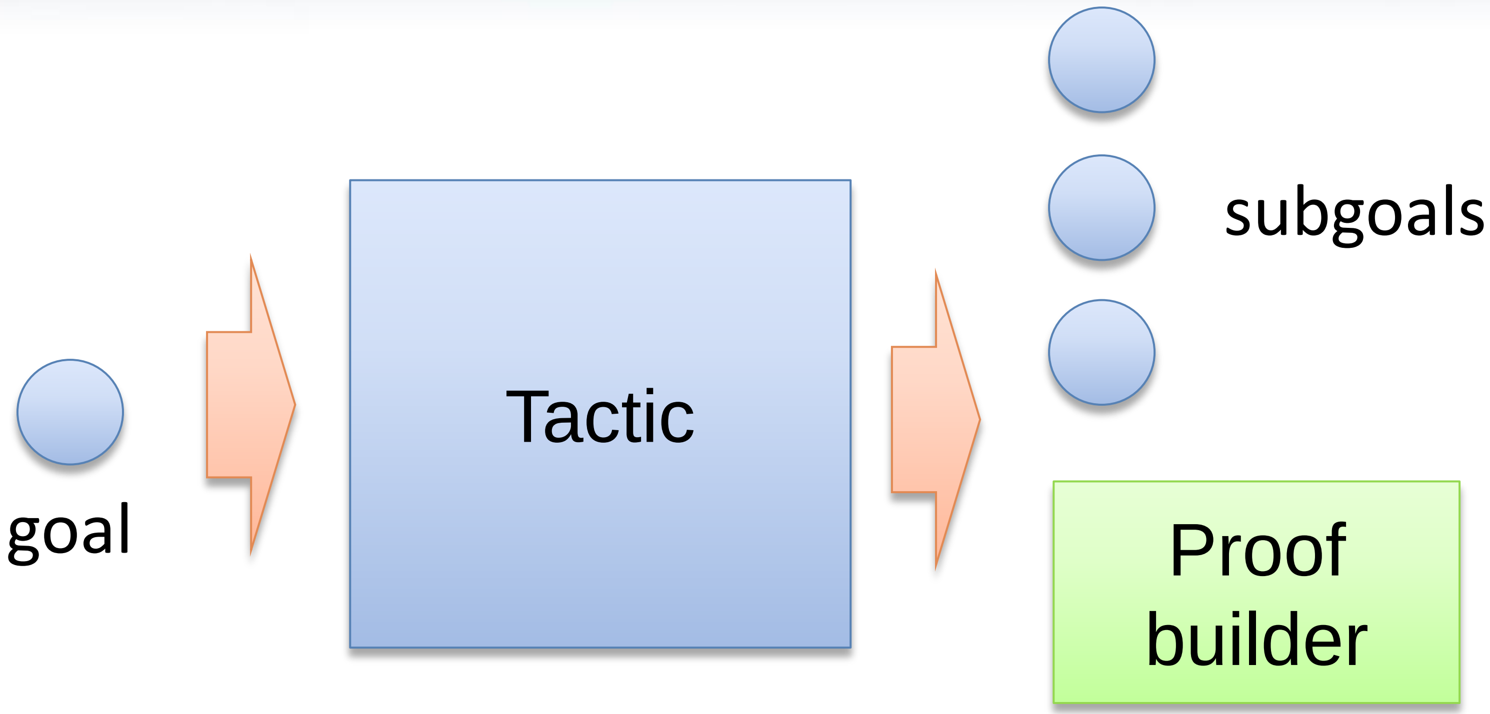
The "Message"

SMT solvers are collections of little engines.

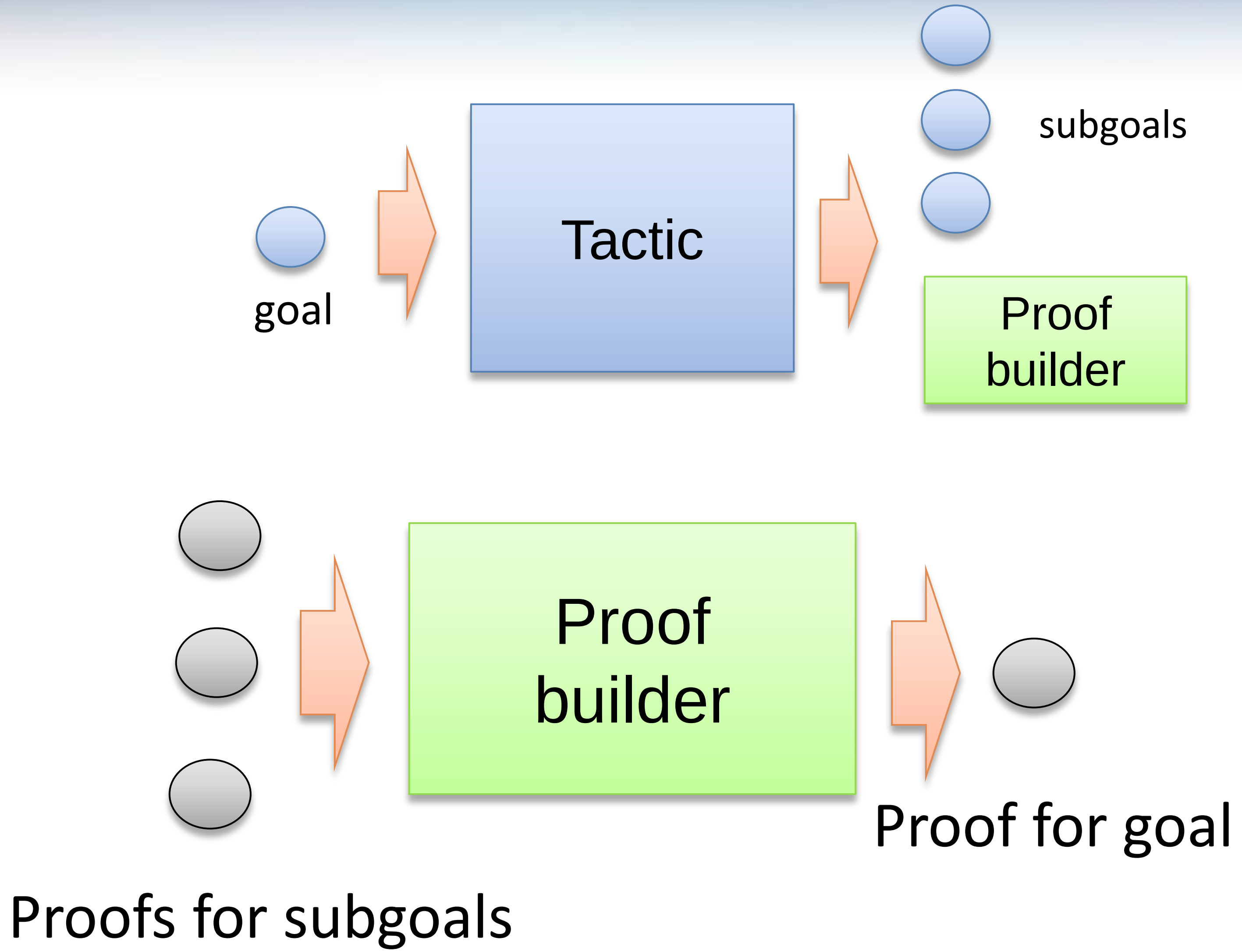
They should provide access to these engines.

Users should be able to define their own strategies.

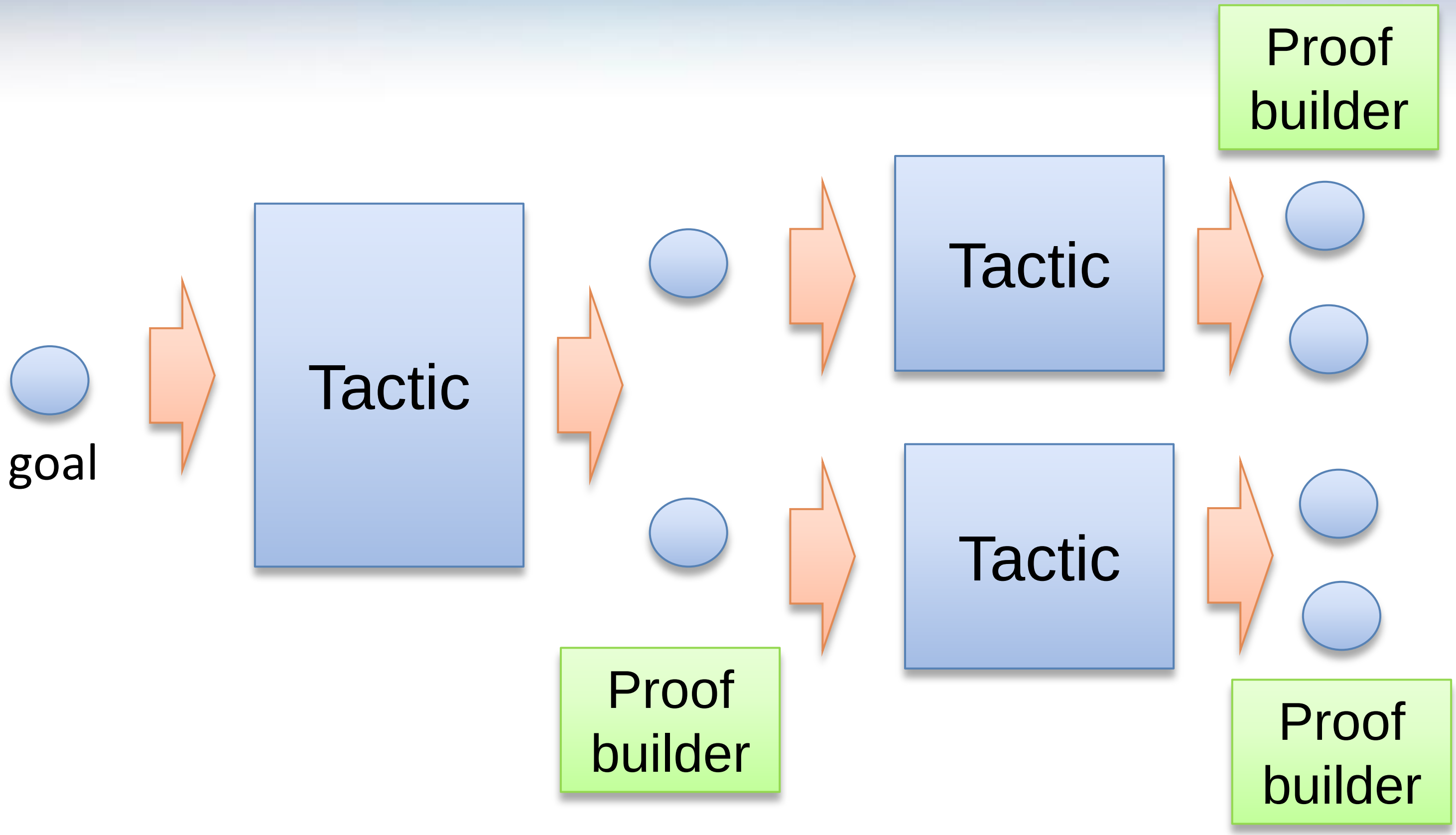
Main inspiration: LCF-approach



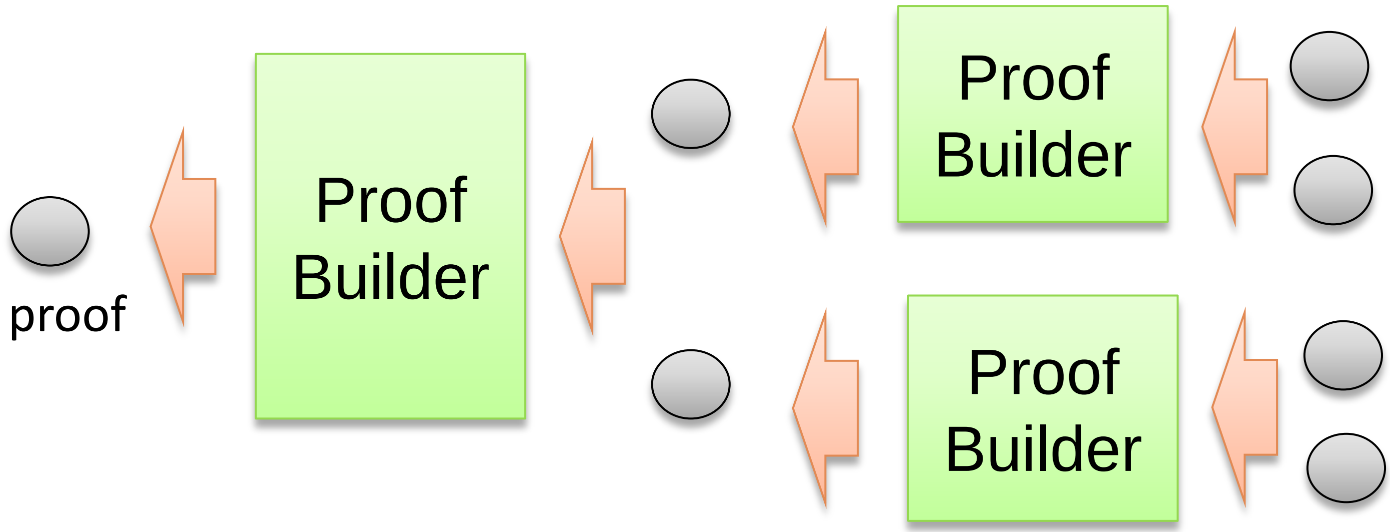
Main inspiration: LCF-approach



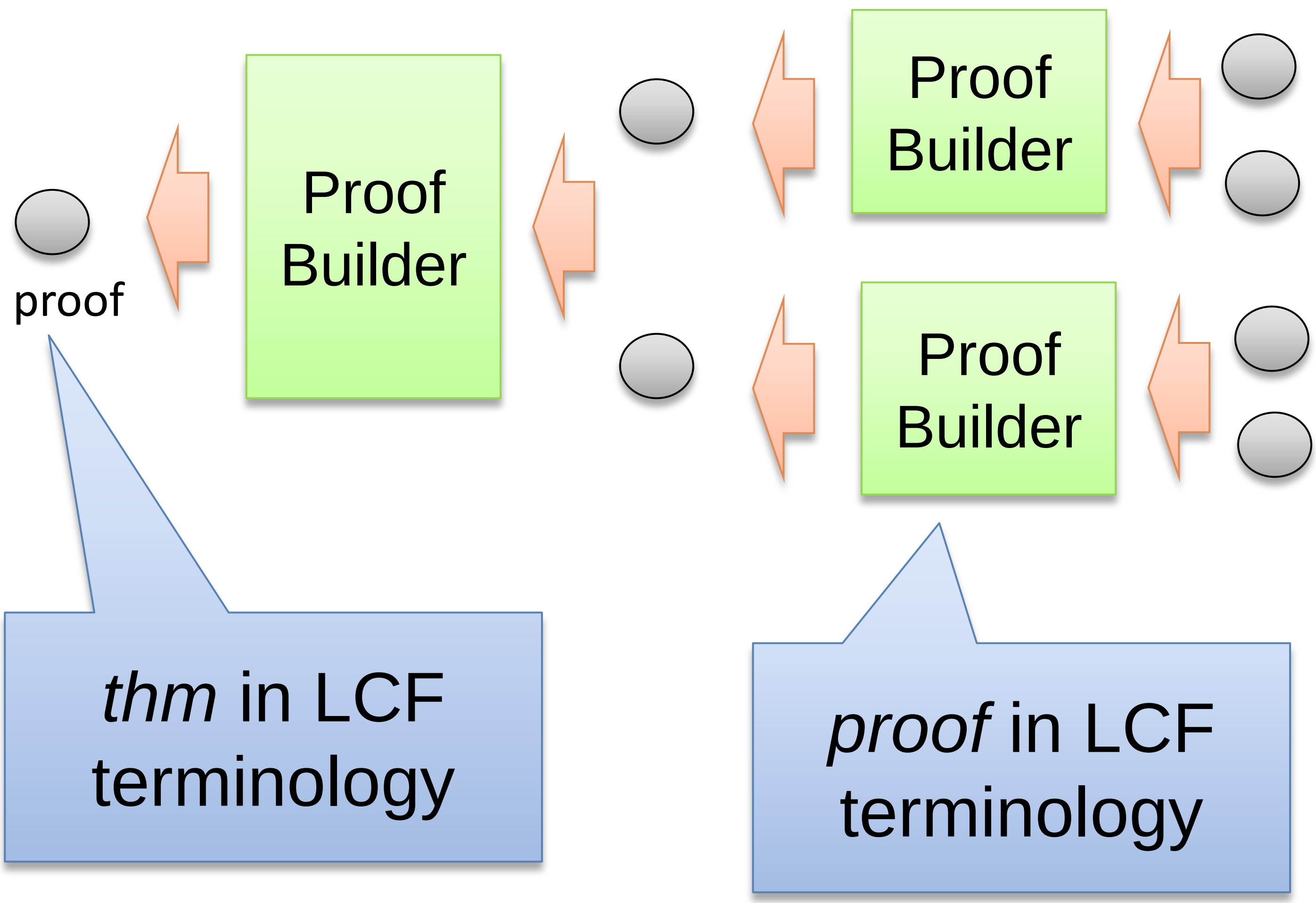
Main inspiration: LCF-approach



Main inspiration: LCF-approach



Main inspiration: LCF-approach



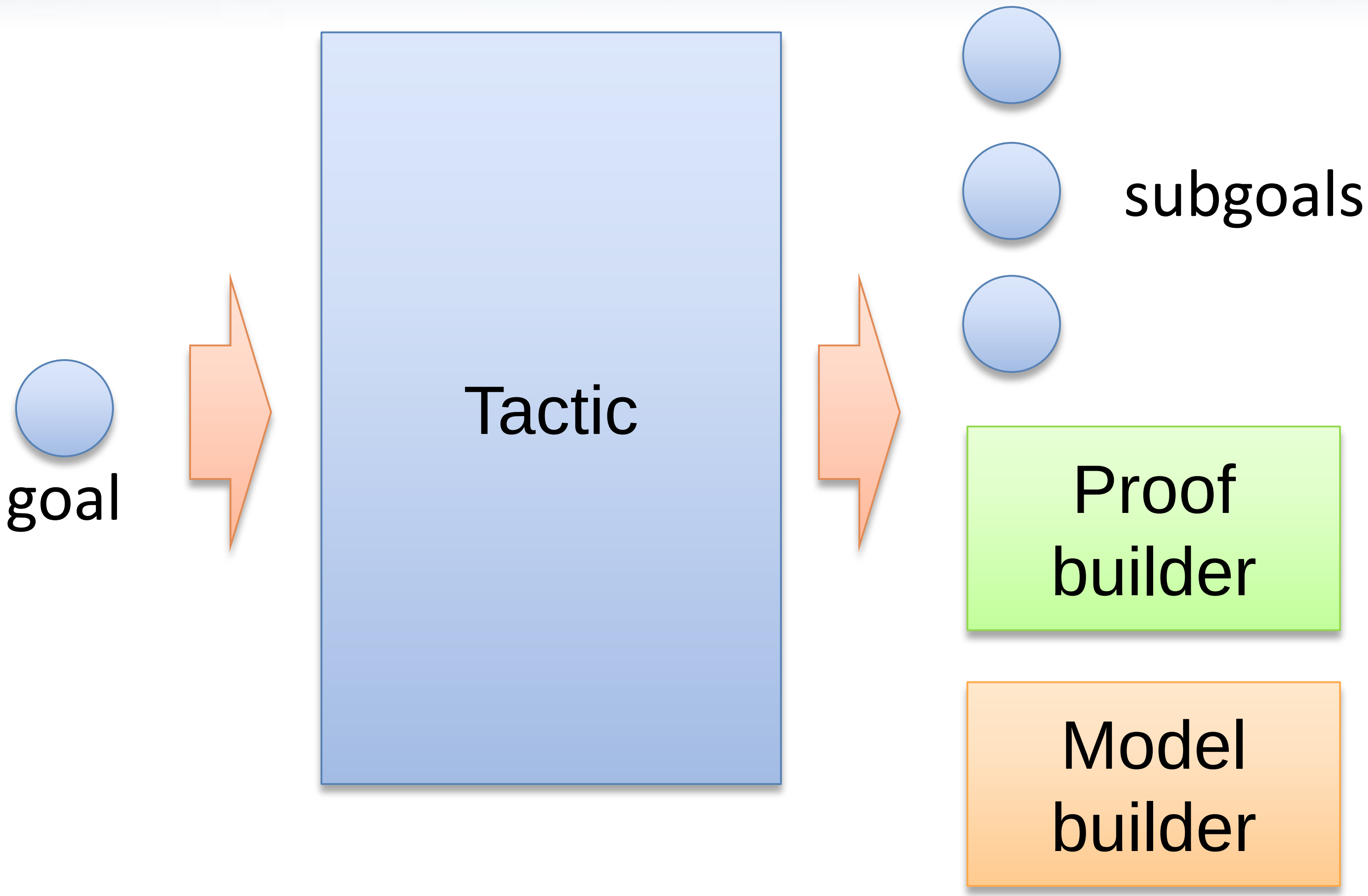
Tacticals aka Combinators

then(Tactic , Tactic) = Tactic

orelse(Tactic , Tactic) = Tactic

repeat(Tactic) = Tactic

SMT Tactic



SMT Tactic

$$
\begin{array}{lcl}
goal & = & formula\ sequence \times attribute\ sequence \\
\\
proofconv & = & proof\ sequence \rightarrow proof \\
modelconv & = & model \times nat \rightarrow model \\
trt & = & \mathsf{sat}\ model \\
 & | & \mathsf{unsat}\ proof \\
 & | & \mathsf{unknown}\ goal\ sequence \times modelconv \times proofconv \\
 & | & \mathsf{fail} \\
tactic & = & goal \rightarrow trt
\end{array}
$$

SMT Tactic

$$
\begin{array}{lcl}
goal & = & formula\ sequence \times attribute\ sequence \\
 & & \\
proofconv & = & proof\ sequence \longrightarrow proof \\
modelconv & = & model \times nat \longrightarrow model \\
trt & = & \mathsf{sat}\ model \\
 & \mid & \mathsf{unsat}\ proof \\
 & \mid & \mathsf{unknown}\ goal\ sequence \times modelconv \times proofconv \\
 & \mid & \mathsf{fail} \\
tactic & = & goal \longrightarrow trt
\end{array}
$$

end-game tactics:

never return unknown(sb, mc, pc)

SMT Tactic

$$
\begin{array}{lcl}
\mathit{goal} & = & \mathit{formula\ sequence} \times \mathit{attribute\ sequence} \\
\\
\mathit{proofconv} & = & \mathit{proof\ sequence} \rightarrow \mathit{proof} \\
\mathit{modelconv} & = & \mathit{model} \times \mathit{nat} \rightarrow \mathit{model} \\
\mathit{trt} & = & \mathsf{sat}\ \mathit{model} \\
 & | & \mathsf{unsat}\ \mathit{proof} \\
 & | & \mathsf{unknown}\ \mathit{goal\ sequence} \times \mathit{modelconv} \times \mathit{proofconv} \\
 & | & \mathsf{fail} \\
\mathit{tactic} & = & \mathit{goal} \rightarrow \mathit{trt}
\end{array}
$$

non-branching tactics:
sb is a sigleton in
unknown(sb, mc, pc)

Trivial goals

Empty goal [ ] is trivially satisfiable

False goal […, false, …] is trivially unsatisfiable

basic : tactic

SMT Tactic example

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

Tactic:
elim-vars

Proof
builder

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

Model
builder

SMT Tactic example

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

Tactic:
elim-vars

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

Proof
builder

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

Model
builder

M

SMT Tactic example

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

Tactic: split-or

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

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

Proof builder

Model builder

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

$\mathsf{then} : (\mathit{tactic} \times \mathit{tactic}) \to \mathit{tactic}$

$\mathsf{then}(t_1, t_2)$ applies t_1 to the given goal and t_2 to every subgoal produced by t_1.

$\mathsf{then}{*} : (\mathit{tactic} \times \mathit{tactic\ sequence}) \to \mathit{tactic}$

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

$\mathsf{orelse} : (\mathit{tactic} \times \mathit{tactic}) \to \mathit{tactic}$

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

$\mathsf{par} : (\mathit{tactic} \times \mathit{tactic}) \to \mathit{tactic}$

$\mathsf{par}(t_1, t_2)$ 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 : $tactic \longrightarrow tactic$

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

repeatupto : $tactic \times nat \longrightarrow tactic$

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

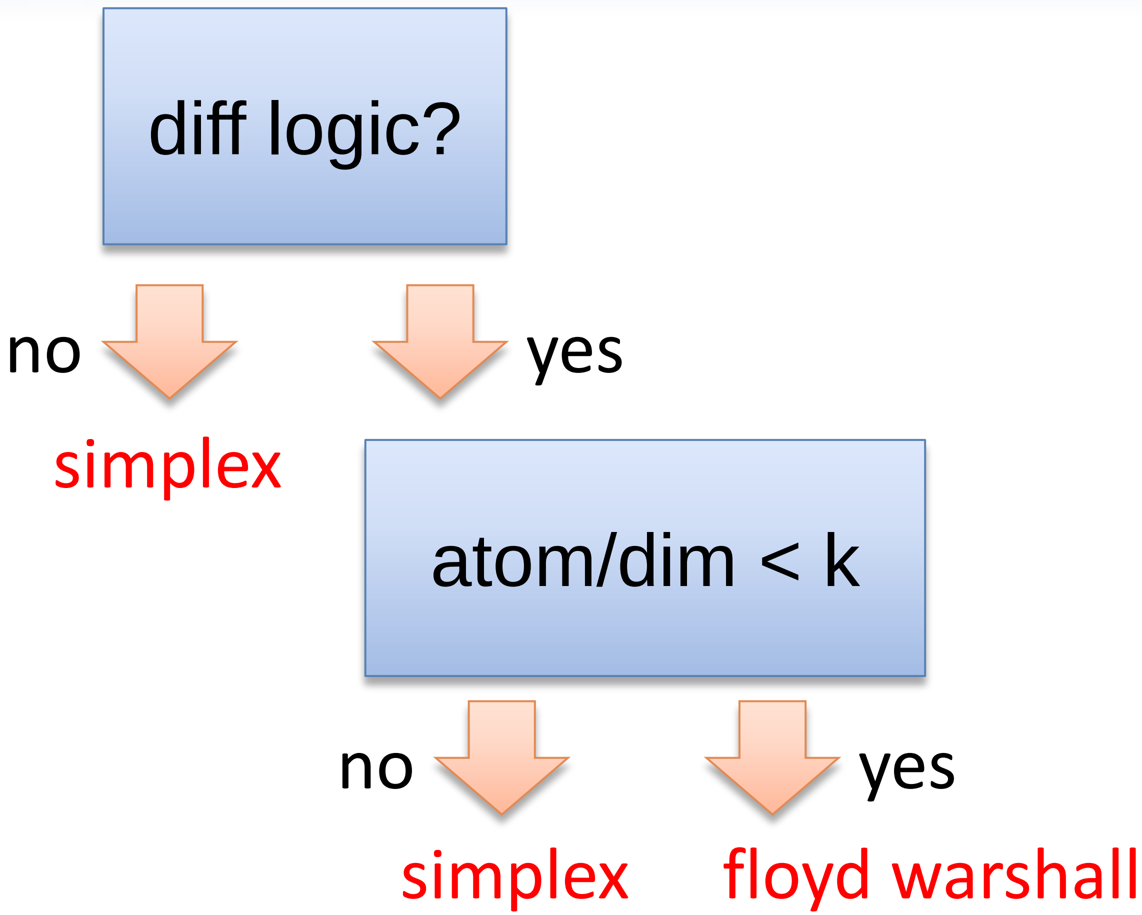
tryfor : $tactic \times seconds \longrightarrow tactic$

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.

Feature / Measures: Yices Strategy



Feature / Measures: Yices Strategy

$$\mathsf{orelse}\left(\mathsf{then}\left(\mathsf{failif}\left(\mathsf{diff} \wedge \frac{\mathsf{atom}}{\mathsf{dim}} > \mathsf{k}\right),\ \mathsf{simplex}\right),\ \mathsf{floydwarshall}\right)$$

Fail if condition is not satisfied.
Otherwise, do nothing.

Feature / Measures: Examples

bw: Sum total bit-width of all rational coefficients of polynomials in case.
diff: True if the formula is in the difference logic fragment.
linear: True if all polynomials are linear.
dim: Number of arithmetic constants.
atoms: Number of atoms.
degree: Maximal total multivariate degree of polynomials.
size: Total formula size.

Tacticals: syntax sugar

$$\mathsf{if}(c,\ t_1,\ t_2) = \mathsf{orelse}(\mathsf{then}(\mathsf{failif}(\neg c), t_1), t_2)$$

$$\mathsf{when}(c,\ t) = \mathsf{if}(c,\ t,\ \mathsf{skip})$$

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

$S \rightarrow S \cup S'$

Over-approximation

$S \rightarrow S \setminus S'$

Under/Over-Approximations

Under-approximation

Example: QF_NIA model finders

add bounds to unbounded variables (and blast)

Over-approximation

Example: Boolean abstraction

Under/Over-Approximations

Combining under and over is bad!

sat and unsat answers cannot be trusted.

Tracking: under/over-approximations

In principle, proof and model converters can check if the resultant models and proofs are valid.

Tracking: under/over-approximations

In principle, proof and model converters can check if the resultant models and proofs are valid.

Problem: if it fails what do we do?

Tracking: under/over-approximations

In principle, proof and model converters can check if the resultant models and proofs are valid.

Problem: if it fails what do we do?

We want to write tactics that can check whether a goal is the result of an abstraction or not.

Tracking: under/over-approximations

Solution

Associate an precision attribute to each goal.

Goal Attributes

Store extra logical information

Examples:

precision markers

goal depth

polynomial factorizations

Decision Engines as Tacticals

AP-CAD (tactic) = tactic

Decision Engines as Tacticals

$$\mathsf{then}(\mathsf{preprocess},\ \mathsf{smt}(\mathsf{finalcheck}))$$

Strategy: Example

$$\mathsf{then}(\mathsf{then}(\mathsf{simplify},\ \mathsf{gaussian}),\ \mathsf{orelse}(\mathsf{modelfinder},\ \mathsf{smt}(\mathsf{apcad}(\mathsf{icp}))))$$

RAHD Calculemus Strategy

	dim	deg	div	calc-0	calc-1	calc-2	qepcad-b	redlog/rlqe	redlog/rlcad
P0	5	4	N	.91	1.59	1.7	416.45*	40.4	-
P1	6	4	N	1.69	3.08	3.42	-*	-	-
P2	5	4	N	1.34	2.41	2.62	-*	-	-
P3	5	4	N	1.52	2.56	2.75	-*	-	-
P4	5	4	N	1.14	2.02	2.16	-*	-	-
P5	14	2	N	.25	.26	.27	-*	97.4	-
P6	11	5	N	147.4	.07	.06	-*	<.01	<.01
P7	8	2	N	.05	<.01	<.01	.08	<.01	<.01
P8	7	32	N	4.5	.1	<.01	8.38	<.01	-
P9	7	16	N	4.51	.15	<.01	.29	.01	6.7
P10	7	12	N	100.74	20.76	8.85	-*	-	-
P11	6	2	Y	1.6	.5	.53	.01	.01	.05
P12	5	3	N	.78	.3	.36	.02	.01	.07
P13	4	10	N	3.83	3.95	4.02	-*	-	-
P14	2	2	N	4.55	1.67	.07	.01	-	-
P15	4	3	Y	.177	.2	.12	.01	<.01	<.01
P16	4	2	N	9.99	2.17	2.1	.02	<.01	<.01
P17	4	2	N	.62	.59	.65	.28	.02	.61
P18	4	2	N	1.25	1.28	1.27	.01	<.01	<.01
P19	3	6	Y	3.34	1.72	2.08	.02	.01	.7
P20	3	4	N	1.18	.65	.65	.01	<.01	.3
P21	3	2	N	.02	.03	<.01	.02	.01	.1
P22	2	4	N	<.01	<.01	<.01	.01	<.01	<.01
P23	2	2	Y	<.01	<.01	<.01	<.01	<.01	<.01

Z3 QF_LIA Strategy

then(preamble, orelse(mf, pb, bounded, smt)

Simplification
Constant propagation
Interval propagation
Contextual simplification
If-then-else elimination
Gaussian elimination
Unconstrained terms

Challenge: small step configuration

proof procedure as a transition system

Abstract DPLL, DPLL(T), Abstract GB, cutsat, …

UnitPropagate :

$$M \parallel F, C \vee l \quad \Longrightarrow \quad M\ l \parallel F, C \vee l \quad \text{if} \begin{cases} M \models \neg C \\ l \text{ is undefined in } M \end{cases}$$

PureLiteral :

$$M \parallel F \quad \Longrightarrow \quad M\ l \parallel F \quad \text{if} \begin{cases} l \text{ occurs in some clause of } F \\ \neg l \text{ occurs in no clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Decide :

$$M \parallel F \quad \Longrightarrow \quad M\ l^{\mathsf{d}} \parallel F \quad \text{if} \begin{cases} l \text{ or } \neg l \text{ occurs in a clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Fail :

$$M \parallel F, C \quad \Longrightarrow \quad \mathit{FailState} \quad \text{if} \begin{cases} M \models \neg C \\ M \text{ contains no decision literals} \end{cases}$$

Backtrack :

$$M\ l^{\mathsf{d}}\ N \parallel F, C \quad \Longrightarrow \quad M\ \neg l \parallel F, C \quad \text{if} \begin{cases} M\ l^{\mathsf{d}}\ N \models \neg C \\ N \text{ contains no decision literals} \end{cases}$$

Challenge: small step configuration

proof procedure as a transition system

Abstract DPLL, DPLL(T), Abstract GB, cutsat, …

Challenge:

Efficient strategic control

Decide :

$$M \parallel F \implies M\, l^{\mathsf{d}} \parallel F \quad \text{if } \begin{cases} l \text{ or } \neg l \text{ occurs in a clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Fail :

$$M \parallel F, C \implies \mathit{FailState} \quad \text{if } \begin{cases} M \models \neg C \\ M \text{ contains no decision literals} \end{cases}$$

Backtrack :

$$M\, l^{\mathsf{d}}\, N \parallel F, C \implies M\, \neg l \parallel F, C \quad \text{if } \begin{cases} M\, l^{\mathsf{d}}\, N \models \neg C \\ N \text{ contains no decision literals} \end{cases}$$

Conclusion

Different domains need different strategies.

We must expose the little engines in SMT solvers.

Interaction between different engines is a must.

Tactic and Tacticals: big step approach.

More transparency.