Quantifiers

Bruno Dutertre  Leonardo de Moura
SRI International  Microsoft Research
Verification Tools need **Quantifiers**

**Modeling the Runtime**

\[ \forall \ h,o,f: \]
\[ \text{IsHeap}(h) \land o \neq \text{null} \land \text{read}(h, o, \text{alloc}) = t \Rightarrow \]
\[ \text{read}(h,o,f) = \text{null} \lor \text{read}(h, \text{read}(h,o,f),\text{alloc}) = \]
Verification Tools need **Quantifiers**

**Frame Axioms**

\[
\forall o, f: \\
\quad o \neq \text{null} \land \text{read}(h_0, o, \text{alloc}) = t \Rightarrow \\
\quad \text{read}(h_1, o, f) = \text{read}(h_0, o, f) \lor (o, f) \in M
\]
Verification Tools need **Quantifiers**

User provided assertions

\[ \forall \; i,j : i \leq j \Rightarrow \text{read}(a,i) \leq \text{read}(b,j) \]
Verification Tools need **Quantifiers**

**Extra Theories**

\[ \forall x: p(x,x) \]

\[ \forall x,y,z: p(x,y), p(y,z) \implies p(x,z) \]

\[ \forall x,y: p(x,y), p(y,x) \implies x = y \]
Verification Tools need **Quantifiers**

**Main Challenge**

Solver must be fast is satisfiable instances
Verifying Compilers

Annotated Program $\rightarrow$ Verification Condition $F$

pre/post conditions, invariants, and other annotations
Verification Condition: Structure

∀ Axioms (non-ground)

Control & Data Flow

BIG and-or tree (ground)
VCC: Verifying C Compiler

```c
#include <vcc2.h>

typedef struct _BITMAP {
    UINT32 Size; // Number of bits
    PUINT32 Buffer; // Memory to store
    ...
    // private invariants
    invariant(Size > 0 && Size % 32 == 0)
    ...
}
```

Available at [http://vcc.codeplex.com/](http://vcc.codeplex.com/)
BAD NEWS

First-order logic (FOL) is semi-decidable

Quantifiers + EUF
BAD NEWS

FOL + Linear Integer Arithmetic is undecidable
Quantifiers + EUF + LIA
Hypervisor

Challenges:
VCs have several Megabytes
Thousands universal quantifiers
Developers are willing at most 5 min per VC
Verification Attempt Time vs. Satisfaction and Productivity

By Michal Moskal (VCC Designer and Software Verification Expert)
NNF: Negation Normal Form

\[
\begin{align*}
\text{NNF}(p) &= p \\
\text{NNF}(\neg p) &= \neg p \\
\text{NNF}(\neg \neg \phi) &= \text{NNF}(\phi) \\
\text{NNF}(\phi_0 \lor \phi_1) &= \text{NNF}(\phi_0) \lor \text{NNF}(\phi_1) \\
\text{NNF}(\neg(\phi_0 \lor \phi_1)) &= \text{NNF}(\neg \phi_0) \land \text{NNF}(\neg \phi_1) \\
\text{NNF}(\phi_0 \land \phi_1) &= \text{NNF}(\phi_0) \land \text{NNF}(\phi_1) \\
\text{NNF}(\neg(\phi_0 \land \phi_1)) &= \text{NNF}(\neg \phi_0) \lor \text{NNF}(\neg \phi_1) \\
\text{NNF}(\forall x : \phi) &= \forall x : \text{NNF}(\phi) \\
\text{NNF}(\neg(\forall x : \phi)) &= \exists x : \text{NNF}(\neg \phi) \\
\text{NNF}(\exists x : \phi) &= \exists x : \text{NNF}(\phi) \\
\text{NNF}(\neg(\exists x : \phi)) &= \forall x : \text{NNF}(\neg \phi)
\end{align*}
\]
NNF: Negation Normal Form

Theorem: \( F \leftrightarrow NNF(F) \)

Ex.: \( NNF(\neg(p \land (\neg r \lor \forall x : q(x))) \)) = \neg p \lor (r \land \exists x : \neg q(x)) \).
Skolemization

After NNF, Skolemization can be used to eliminate existential quantifiers.

$$\exists y : F[x, y] \leadsto F[x, f(x)]$$
Skolemization

The resultant formula is equisatisfiable.

Example:

\[ \forall x : p(x) \Rightarrow \exists y : q(x, y) \]

\[ \forall x : p(x) \Rightarrow q(x, f(x)) \]
∀ - Many Approaches

- Heuristic quantifier instantiation
- SMT + Saturation provers
- Complete quantifier instantiation
- Decidable fragments
- Model based quantifier instantiation
- Quantifier Elimination
Heuristic Quantifier Instantiation

E-matching (matching modulo equalities).

Example:

\[ \forall x: f(g(x)) = x \{ f(g(x)) \} \]

\( a = g(b), \)

\( b = c, \)

\( f(a) \neq c \)
Heuristic Quantifier Instantiation

E-matching (matching modulo equalities).

Example:

\[ \forall x: f(g(x)) = x \{ f(g(x)) \} \]

\[ a = g(b), \]
\[ b = c, \]
\[ f(a) \neq c \]

\[ x = b \quad f(g(b)) = b \]
E-matching problem

Input: A set of ground equations $E$, a ground term $t$, and a pattern $p$, where $p$ possibly contains variables.

Output: The set of substitutions $\beta$ over the variables in $p$, such that:

$$E \vdash t = \beta(p)$$

Example:

$E \equiv \{a = f(b), a = f(c)\}$

$t \equiv g(a)$

$p \equiv g(f(x))$

$R \equiv \{\{x \leftarrow b\}, \{x \leftarrow c\}\}$

Applying $\beta_2$: $a = f(b), a = f(c) \vdash g(a) = g(f(c))$
E-matching Challenge

Number of matches can be exponential
It is not refutationally complete
The real challenge is finding new matches:
  Incrementally during backtracking search
  Large database of patterns
EUF Solver: Review

\[ f(g(a)) = c, c \neq f(g(b)), a = b \]

\[ F = \{ a \mapsto a, b \mapsto b, c \mapsto c, g(a) \mapsto g(a), g(b) \mapsto g(b), f(g(a)) \mapsto f(g(a)), f(g(b)) \mapsto f(g(b)) \} \]

\[ D = \{ \} \]

\[ \pi(a) = \{ g(a) \} \]

\[ \pi(b) = \{ g(b) \} \]

\[ \pi(g(a)) = \{ f(g(a)) \} \]

\[ \pi(g(b)) = \{ f(g(b)) \} \]
EUF Solver: Review

\[
f(g(a)) = c, c \neq f(g(b)), a = b
\]

\[
F = \{a \mapsto a, b \mapsto b, c \mapsto c, g(a) \mapsto g(a), g(b) \mapsto g(b), f(g(a)) \mapsto f(g(a)), f(g(b)) \mapsto f(g(b))\}
\]

\[
D = \{\}
\]

\[
\pi(a) = \{g(a)\}
\]

\[
\pi(b) = \{g(b)\}
\]

\[
\pi(g(a)) = \{f(g(a))\}
\]

\[
\pi(g(b)) = \{f(g(b))\}
\]

Merge equivalence classes of \(f(g(a))\) and \(c\).
EUF Solver: Review

\[ f(g(a)) = c, \ c \neq f(g(b)), \ a = b \]

\[
F = \{ a \leftrightarrow a, \ b \leftrightarrow b, \ c \leftrightarrow c, \ g(a) \leftrightarrow g(a), \ g(b) \leftrightarrow g(b), \ f(g(a)) \leftrightarrow c, \ f(g(b)) \leftrightarrow f(g(b)) \} \\
D = \{ \} \\
\pi(a) = \{ g(a) \} \\
\pi(b) = \{ g(b) \} \\
\pi(g(a)) = \{ f(g(a)) \} \\
\pi(g(b)) = \{ f(g(b)) \} \\
\]
EUF Solver: Review

\[ f(g(a)) = c, c \neq f(g(b)), a = b \]

\[
F = \{ a \mapsto a, b \mapsto b, c \mapsto c, g(a) \mapsto g(a), g(b) \mapsto g(b), f(g(a)) \mapsto c, f(g(b)) \mapsto f(g(b)) \}
\]

\[
D = \{
\}
\]

\[
\pi(a) = \{ g(a) \}
\]

\[
\pi(b) = \{ g(b) \}
\]

\[
\pi(g(a)) = \{ f(g(a)) \}
\]

\[
\pi(g(b)) = \{ f(g(b)) \}
\]

Add disequality
EUF Solver: Review

\[ f(g(a)) = c, c \neq f(g(b)), a = b \]

\[
\begin{align*}
F & = \{ a \mapsto a, b \mapsto b, c \mapsto c, g(a) \mapsto g(a), g(b) \mapsto g(b) \} \\
D & = \{ c \neq f(g(b)) \} \\
\pi(a) & = \{ g(a) \} \\
\pi(b) & = \{ g(b) \} \\
\pi(g(a)) & = \{ f(g(a)) \} \\
\pi(g(b)) & = \{ f(g(b)) \}
\end{align*}
\]
EUF Solver: Review

\[ f(g(a)) = c, c \neq f(g(b)), a = b \]

\[ F = \{ a \leftrightarrow a, \ b \leftrightarrow b, \ c \leftrightarrow c, \ g(a) \leftrightarrow g(a), \ g(b) \leftrightarrow g(b) \}
\]

\[ f(g(a)) \leftrightarrow c, \ f(g(b)) \leftrightarrow f(g(b)) \]

\[ D = \{ c \neq f(g(b)) \} \]

\[ \pi(a) = \{ g(a) \} \]

\[ \pi(b) = \{ g(b) \} \]

\[ \pi(g(a)) = \{ f(g(a)) \} \]

\[ \pi(g(b)) = \{ f(g(b)) \} \]

Merge equivalence classes of \( a \) and \( b \).
EUF Solver: Review

\[ f(g(a)) = c, c \neq f(g(b)), a = b, g(a) = g(b) \]

\[ F = \{a \mapsto a, b \mapsto a, c \mapsto c, g(a) \mapsto g(a), g(b) \mapsto g(b)\} \]
\[ f(g(a)) \mapsto c, f(g(b)) \mapsto f(g(b))\}\]

\[ D = \{c \neq f(g(b))\}\]

\[ \pi(a) = \{g(a), g(b)\}\]

\[ \pi(b) = \{g(b)\}\]

\[ \pi(g(a)) = \{f(g(a))\}\]

\[ \pi(g(b)) = \{f(g(b))\}\]
EUF Solver: Review

\[ f(g(a)) = c, \ c \neq f(g(b)), \ a = b, \ g(a) = g(b) \]

\[
F = \{ a \mapsto a, \ b \mapsto a, \ c \mapsto c, \ g(a) \mapsto g(a), \ g(b) \mapsto g(b) \\
    f(g(a)) \mapsto c, \ f(g(b)) \mapsto f(g(b)) \}
\]

\[
D = \{ c \neq f(g(b)) \}
\]

\[
\pi(a) = \{ g(a), g(b) \}
\]

\[
\pi(b) = \{ g(b) \}
\]

\[
\pi(g(a)) = \{ f(g(a)) \}
\]

\[
\pi(g(b)) = \{ f(g(b)) \}
\]

Merge equivalence classes of \( g(a) \) and \( g(b) \).
EUF Solver: Review

\[ f(g(a)) = c, \quad c \neq f(g(b)), \quad a = b, \quad g(a) = g(b), \quad f(g(a)) = f(g(b)) \]

\[ F = \{ a \mapsto a, \quad b \mapsto a, \quad c \mapsto c, \quad g(a) \mapsto g(b), \quad g(b) \mapsto g(b), \quad f(g(a)) \mapsto c, \quad f(g(b)) \mapsto f(g(b)) \} \]

\[ D = \{ c \neq f(g(b)) \} \]

\[ \pi(a) = \{ g(a), g(b) \} \]

\[ \pi(b) = \{ g(b) \} \]

\[ \pi(g(a)) = \{ f(g(a)) \} \]

\[ \pi(g(b)) = \{ f(g(b)), f(g(a)) \} \]
EUF Solver: Review

\[ f(g(a)) = c, c \neq f(g(b)), a = b, g(a) = g(b), f(g(a)) = f(g(b)) \]

\[ F = \{ a \mapsto a, b \mapsto a, c \mapsto c, g(a) \mapsto g(b), g(b) \mapsto g(b) \} \]
\[ f(g(a)) \mapsto c, f(g(b)) \mapsto f(g(b)) \}

\[ D = \{ c \neq f(g(b)) \} \]

\[ \pi(a) = \{ g(a), g(b) \} \]
\[ \pi(b) = \{ g(b) \} \]
\[ \pi(g(a)) = \{ f(g(a)) \} \]
\[ \pi(g(b)) = \{ f(g(b)), f(g(a)) \} \]

Merge equivalence classes of \( f(g(a)) \) and \( f(g(b)) \) \( \rightsquigarrow \) unsat.
E-matching

\[ \text{match}(x, t, S) = \{\beta \cup \{x \mapsto t\} \mid \beta \in S, x \notin \text{dom}(\beta)\} \cup \{\beta \mid \beta \in S, F^*(\beta(x)) = F^*(t)\} \]

\[ \text{match}(c, t, S) = S \text{ if } F^*(c) = F^*(t) \]

\[ \text{match}(c, t, S) = \emptyset \text{ if } F^*(c) \neq F^*(t) \]

\[ \text{match}(f(p_1, \ldots, p_n), t, S) = \bigcup_{F^*(f(t_1, \ldots, t_n)) = F^*(t)} \text{match}(p_n, t_n, \ldots, \text{match}(p_1, t_1, S) \ldots) \]

\[ \text{match}(p, t, \{\emptyset\}) \text{ returns the desired set of substitutions.} \]
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \} \]

E-match \( t \) and \( p \):

\[ t = f(c, b) \]
\[ p = f(g(x), h(x, a)) \]
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \}\]

\[ \text{match}(f(g(x), h(x, a)), f(c, b), \{ \emptyset \}) = \]
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \}\]

\[ \text{match}(f(g(x), h(x, a)), f(c, b), \{\emptyset\}) = \]
\[ \text{match}(g(x), c, \text{match}(h(x, a), b, \{\emptyset\})) \quad \text{for } f(c, b) \]
\[ \cup \]
\[ \text{match}(g(x), g(a), \text{match}(h(x, a), b, \{\emptyset\})) \quad \text{for } f(g(a), b) \]
E-matching: Example

\[ F = \{a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[
    f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \\
    g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \\
    h(a, d) \mapsto b, \ h(c, a) \mapsto b \} \]

\[
\text{match}(f(g(x), h(x, a)), f(c, b), \{\emptyset\}) = \\
\text{match}(g(x), c, \text{match}(x, a, \text{match}(a, d, \{\emptyset\}))) \quad \text{for } h(a, d) \\
\cup \\
\text{match}(x, c, \text{match}(a, a, \{\emptyset\}))) \quad \text{for } h(c, a) \\
\cup \\
\text{match}(g(x), g(a), \text{match}(h(x, a), b, \{\emptyset\}))) \]
E-matching: Example

\[ F = \{ a \mapsto c, b \mapsto b, c \mapsto c, d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \} \]

\[ \text{match}(f(g(x), h(x, a)), f(c, b), \{\emptyset\}) = \]
\[ \text{match}(g(x), c, \text{match}(x, a, \text{match}(a, d, \{\emptyset\}))) \quad \text{for } h(a, d) \]
\[ \bigcup \]
\[ \text{match}(x, c, \text{match}(a, a, \{\emptyset\})) \quad \text{for } h(c, a) \]
\[ \bigcup \]
\[ \text{match}(g(x), g(a), \text{match}(h(x, a), b, \{\emptyset\})) \]

\( a \) and \( d \) are not in the same equivalence class.
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \}\]

\[ \text{match}(f(g(x), h(x, a)), f(c, b), \{\emptyset\}) = \]
\[ \text{match}(g(x), c, \text{match}(x, a, \emptyset)) \]
\[ \cup \]
\[ \text{match}(x, c, \text{match}(a, a, \{\emptyset\}))) \]
\[ \cup \]
\[ \text{match}(g(x), g(a), \text{match}(h(x, a), b, \{\emptyset\}))) \]
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c,|a) \mapsto b \} \]

\[ \text{match}(f(g(x), h(x, a)), f(c, b), \{\emptyset\}) = \]

\[ \text{match}(g(x), c, \emptyset) \]
\[ \quad \cup \]
\[ \quad \text{match}(x, c, \text{match}(a, a, \{\emptyset\})) \]
\[ \cup \]
\[ \text{match}(g(x), g(a), \text{match}(h(x, a), b, \{\emptyset\})) \]
E-matching: Example

\[ F = \{a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c,b) \mapsto f(c,b), \ f(g(a),b) \mapsto f(c,b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a,d) \mapsto b, \ h(c,a) \mapsto b \} \]

\[
match(f(g(x), h(x,a)), f(c,b), \{\emptyset\}) =
\]

\[
match(g(x), c, \emptyset)
\]

\[
\cup
\]

\[
match(x, c, match(a, a, \{\emptyset\})))
\]

\[
\cup
\]

\[
match(g(x), g(a), match(h(x,a), b, \{\emptyset\})))
\]

\]
E-matching: Example

\[
F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \ f(c,b) \mapsto f(c,b), \ f(g(a),b) \mapsto f(c,b), \ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \ h(a,d) \mapsto b, \ h(c,a) \mapsto b \} \\
match(f(g(x), h(x,a)), f(c,b), \{\emptyset\}) = \\
match(g(x), c, \emptyset) \cup \\
match(x, c, match(a, a, \{\emptyset\})) \cup \\
match(g(x), g(a), match(h(x,a), b, \{\emptyset\})) \\
F^*(a) = F^*(a)
\]
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \} \]

\[ \text{match}(f(g(x), h(x, a)), f(c, b), \{\emptyset\}) = \]
\[ \text{match}(g(x), c, \emptyset) \]
\[ \cup \]
\[ \text{match}(x, c, \{\emptyset\})) \]
\[ \cup \]
\[ \text{match}(g(x), g(a), \text{match}(h(x, a), b, \{\emptyset\})) \]
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \} \]

\[
\text{match}(f(g(x), h(x, a)), f(c, b), \{\emptyset\}) = \\
\text{match}(g(x), c, \emptyset) \\
\bigcup \\
\{\{x \mapsto c\}\}) \\
\bigcup \\
\text{match}(g(x), g(a), \text{match}(h(x, a), b, \{\emptyset\}))
\]
E-matching: Example

\[ F = \{ a \mapsto c, \, b \mapsto b, \, c \mapsto c, \, d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \quad f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \quad g(b) \mapsto g(b), \quad g(c) \mapsto c, \quad g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \quad h(c, a) \mapsto b \} \]

\[ \text{match}(f(g(x), h(x, a)), f(c, b), \{\emptyset\}) = \]
\[ \text{match}(g(x), c, \{\{x \mapsto c\}\}) \]
\[ \cup \]
\[ \text{match}(g(x), g(a), \text{match}(h(x, a), b, \{\emptyset\})) \]
E-matching: Example

\[
F = \{a \mapsto c, b \mapsto b, c \mapsto c, d \mapsto d, \\
f(c, b) \mapsto f(c, b), f(g(a), b) \mapsto f(c, b), \\
g(a) \mapsto c, g(b) \mapsto g(b), g(c) \mapsto c, g(d) \mapsto c, \\
h(a, d) \mapsto b, h(c, a) \mapsto b\}
\]

\[
match(f(g(x), h(x, a)), f(c, b), \{\emptyset\}) =
\]

\[
match(x, a, \{\{x \mapsto c\}\}) \cup \quad \text{for } g(a)
\]

\[
match(x, c, \{\{x \mapsto c\}\}) \cup \quad \text{for } g(c)
\]

\[
match(x, d, \{\{x \mapsto c\}\}) \cup \quad \text{for } g(d)
\]

\[
match(g(x), g(a), match(h(x, a), b, \{\emptyset\}))
\]
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \} \]

\[ \text{match}(f(g(x), h(x, a)), f(c, b), \{\emptyset\}) = \]
\[ \{\{x \mapsto c\}\} \cup \]
\[ \{\{x \mapsto c\}\} \cup \]
\[ \emptyset \cup \]
\[ \text{match}(g(x), g(a), \text{match}(h(x, a), b, \{\emptyset\})) \]
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \} \]

match\((f(g(x), h(x, a)), f(c, b), \{\emptyset\}) = \]
\[ \{ \{ x \mapsto c \} \} \cup \]
\[ \text{match}(g(x), g(a), \text{match}(h(x, a), b, \{\emptyset\})) \]
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \}\]

\[ \text{match}(f(g(x), h(x, a)), f(c, b), \{ \emptyset \}) = \]
\[ \{ \{ x \mapsto c \} \} \cup \]
\[ \{ \{ x \mapsto c \} \} \]
E-matching: Example

\[ F = \{ a \mapsto c, \ b \mapsto b, \ c \mapsto c, \ d \mapsto d, \]
\[ f(c, b) \mapsto f(c, b), \ f(g(a), b) \mapsto f(c, b), \]
\[ g(a) \mapsto c, \ g(b) \mapsto g(b), \ g(c) \mapsto c, \ g(d) \mapsto c, \]
\[ h(a, d) \mapsto b, \ h(c, a) \mapsto b \}\]

\[ \text{match}(f(g(x), h(x, a)), f(c, b), \{ \emptyset \}) = \]
\[ \{ \{ x \mapsto c \} \} \]
## Efficient E-matching

<table>
<thead>
<tr>
<th>Problem</th>
<th>Indexing Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast retrieval</td>
<td>E-matching code trees</td>
</tr>
<tr>
<td>Incremental E-Matching</td>
<td>Inverted path index</td>
</tr>
</tbody>
</table>
E-matching: code trees

Trigger:
\[ f(x1, g(x1, a), h(x2), b) \]

Similar triggers share several instructions.

Instructions:
1. init(f, 2)
2. check(r4, b, 3)
3. bind(r2, g, r5, 4)
4. compare(r1, r5, 5)
5. check(r6, a, 6)
6. bind(r3, h, r7, 7)
7. yield(r1, r7)

Combine code sequences in a code tree
E-matching limitations

E-matching needs ground seeds.

∀x: p(x),
∀x: not p(x)
E-matching limitations

Bad user provided triggers:
\[ \forall x: f(g(x)) = x \{ f(g(x)) \} \]
\[ g(a) = c, \]
\[ g(b) = c, \]
\[ a \neq b \]

Trigger is too restrictive
E-matching limitations

Bad user provided triggers:
\[ \forall x: \ f(g(x)) = x \{ g(x) \} \]
\[ g(a) = c, \]
\[ g(b) = c, \]
\[ a \neq b \]

More “liberal” trigger
E-matching limitations

Bad user provided triggers:

\[ \forall x: f(g(x)) = x \{ g(x) \} \]

\[ g(a) = c, \]
\[ g(b) = c, \]
\[ a \neq b, \]
\[ f(g(a)) = a, \]
\[ f(g(b)) = b \]

\[ a=b \]
E-matching **limitations**

It is not refutationally complete

**False positives**
E-matching: why do we use it?

Integrates smoothly with current SMT Solvers design.

Proof finding.

Software verification problems are big & shallow.
Decidable Fragments
&
Complete Quantifier Instatiation
∀ + theories

There is no sound and refutationally complete procedure for linear arithmetic + unintepreted function symbols
Model Generation

How to represent the model of satisfiable formulas?

Functor:

Given a model $M$ for $T$
Generate a model $M'$ for $F$ (modulo $T$)

Example:

$F$: $f(a) = 0$ and $a > b$ and $f(b) > f(a) + 1$

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>$\text{ite}(x=1, 0, 2)$</td>
</tr>
</tbody>
</table>
Model Generation

How to represent the model of satisfiable formulas?

Functor:

Given a model $M$ for $T$

Generate a model $M'$ for $F$ (modulo $T$)

Example:

$F$: \[ f(a) = 0 \text{ and } a > b \text{ and } f(b) > f(a) + 1 \]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
</tr>
<tr>
<td>$f$</td>
<td>$\text{ite}(x=1, 0, 2)$</td>
</tr>
</tbody>
</table>

Interpretation is given using $T$-symbols
Model Generation

How to represent the model of satisfiable formulas?

Functor:

Given a model $M$ for $T$

Generate a model $M'$ for $F$ (modulo $T$)

Example:

$F: f(a) = 0$ and $a > b$ and $f(b) > f(a) + 1$

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
</tr>
<tr>
<td>$f$</td>
<td>$ite(x=1, 0, 2)$</td>
</tr>
</tbody>
</table>
Models as Functional Programs

```
(declare-fun f (Int Int) Int)
(declare-const a Int)
(declare-const b Int)

(assert (forall ((x Int)) (>= (f x x) (+ x a))))

(assert (< (f a b) a))
(assert (> a 0))
(check-sat)
(get-model)

(echo "evaluating (f (+ a 10) 20)...")
(eval (f (+ a 10) 20))
```

```
sat
(model
  (define-fun b () Int 2)
  (define-fun a () Int 1)
  (define-fun f ((x!1 Int) (x!2 Int)) Int
    (ite (and (= x!1 1) (= x!2 2)) 0 (+ 1 x!1)))
)
  evaluating (f (+ a 10) 20)...
  12
```

ask z3
Model Checking

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>ite(x=1, 0, 2)</td>
</tr>
</tbody>
</table>

Is \( \forall x: f(x) \geq 0 \) satisfied by \( M' \)?

Yes,

not (ite(k=1,0,2) \geq 0) is unsatisfiable
Model Checking

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>(\text{ite}(x=1, 0, 2))</td>
</tr>
</tbody>
</table>

\[ M' : \]

Is \(\forall x: f(x) \geq 0\) satisfied by \(M'\)?

Yes,

not \((\text{ite}(k=1,0,2) \geq 0)\) is unsatisfiable

Negated quantifier

Replaced \(f\) by its interpretation

Replaced \(x\) by fresh constant \(k\)
Essentially uninterpreted fragment

Variables appear only as arguments of uninterpreted symbols.

\[ f(g(x_1) + a) < g(x_1) \lor h(f(x_1), x_2) = 0 \checkmark \]

\[ f(x_1 + x_2) \leq f(x_1) + f(x_2) \xmark \]
Basic Idea

Given a set of formulas $F$, build an equisatisfiable set of quantifier-free formulas $F^*$

“Domain” of $f$ is the set of ground terms $A_f$

$t \in A_f$ if there is a ground term $f(t)$

Suppose

1. We have a clause $C[f(x)]$ containing $f(x)$.
2. We have $f(t)$.

$\Rightarrow$

Instantiate $x$ with $t$: $C[f(t)]$. 
Example

\begin{align*}
F & \\
g(x_1, x_2) = 0 \lor h(x_2) = 0, \\
g(f(x_1), b) + 1 & \leq f(x_1), \\
h(c) & = 1, \\
f(a) & = 0 \\
F^* & 
\end{align*}
Example

\[ F \]
\[ g(x_1, x_2) = 0 \lor h(x_2) = 0, \]
\[ g(f(x_1), b) + 1 \leq f(x_1), \]
\[ h(c) = 1, \]
\[ f(a) = 0 \]

Copy quantifier-free formulas

“Domains”:
\[ A_f: \{ a \} \]
\[ A_g: \{ \} \]
\[ A_h: \{ c \} \]
Example

\[ g(x_1, x_2) = 0 \lor h(x_2) = 0, \]
\[ g(f(x_1), b) + 1 \leq f(x_1), \]
\[ h(c) = 1, \]
\[ f(a) = 0 \]

“Domains”:

\[ A_f : \{ a \} \]
\[ A_g : \{ \} \]
\[ A_h : \{ c \} \]
Example

\[ g(x_1, x_2) = 0 \lor h(x_2) = 0, \]
\[ g(f(x_1), b) + 1 \leq f(x_1), \]
\[ h(c) = 1, \]
\[ f(a) = 0 \]

\[ F* \]
\[ h(c) = 1, \]
\[ f(a) = 0, \]
\[ g(f(a), b) + 1 \leq f(a) \]

“Domains”:
\[ A_f : \{ a \} \]
\[ A_g : \{ [f(a), b] \} \]
\[ A_h : \{ c \} \]
Example

F

\[ g(x_1, x_2) = 0 \lor h(x_2) = 0, \]
\[ g(f(x_1), b) + 1 \leq f(x_1), \]
\[ h(c) = 1, \]
\[ f(a) = 0 \]

F*

\[ h(c) = 1, \]
\[ f(a) = 0, \]
\[ g(f(a), b) + 1 \leq f(a), \]

“Domains”:

\[ A_f : \{ a \} \]
\[ A_g : \{ [f(a), b] \} \]
\[ A_h : \{ c \} \]
Example

F

\[ g(x_1, x_2) = 0 \lor h(x_2) = 0, \]
\[ g(f(x_1), b) + 1 \leq f(x_1), \]
\[ h(c) = 1, \]
\[ f(a) = 0 \]

F*

\[ h(c) = 1, \]
\[ f(a) = 0, \]
\[ g(f(a), b) + 1 \leq f(a), \]
\[ g(f(a), b) = 0 \lor h(b) = 0 \]

“Domains”:
\[ A_f : \{ a \} \]
\[ A_g : \{ \{f(a), b\} \} \]
\[ A_h : \{ c, b \} \]
Example

$g(x_1, x_2) = 0 \lor h(x_2) = 0,$
$g(f(x_1),b) + 1 \leq f(x_1),$
$h(c) = 1,$
$f(a) = 0$

“Domains”:
$A_f : \{ a \}$
$A_g : \{ [f(a), b] \}$
$A_h : \{ c, b \}$

$F$

$F^*$

$h(c) = 1,$
$f(a) = 0,$
$g(f(a),b) + 1 \leq f(a),$
$g(f(a), b) = 0 \lor h(b) = 0$
Example

\[ F \]
\[
g(x_1, x_2) = 0 \lor h(x_2) = 0, \\
g(f(x_1), b) + 1 \leq f(x_1), \\
h(c) = 1, \\
f(a) = 0
\]

\[ F^* \]
\[
h(c) = 1, \\
f(a) = 0, \\
g(f(a), b) + 1 \leq f(a), \\
g(f(a), b) = 0 \lor h(b) = 0, \\
g(f(a), c) = 0 \lor h(c) = 0
\]

“Domains”:
\[ A_f : \{ a \} \]
\[ A_g : \{ [f(a), b], [f(a), c] \} \]
\[ A_h : \{ c, b \} \]
Example

\[ g(x_1, x_2) = 0 \lor h(x_2) = 0, \]
\[ g(f(x_1), b) + 1 \leq f(x_1), \]
\[ h(c) = 1, \]
\[ f(a) = 0 \]

F

\[ h(c) = 1, \]
\[ f(a) = 0, \]
\[ g(f(a), b) + 1 \leq f(a), \]
\[ g(f(a), b) = 0 \lor h(b) = 0, \]
\[ g(f(a), c) = 0 \lor h(c) = 0 \]

F*

\[ a \rightarrow 2, \ b \rightarrow 2, \ c \rightarrow 3 \]
\[ f \rightarrow \{ 2 \rightarrow 0, \ldots \} \]
\[ h \rightarrow \{ 2 \rightarrow 0, \ 3 \rightarrow 1, \ \ldots \} \]
\[ g \rightarrow \{ [0,2] \rightarrow -1, \ [0,3] \rightarrow 0, \ \ldots \} \]
Basic Idea

Given a model $M$ for $F^*$, Build a model $M^\pi$ for $F$

Define a projection function $\pi_f$ s.t.
range of $\pi_f$ is $M(A_f)$, and
$\pi_f(v) = v \quad \text{if} \quad v \in M(A_f)$

Then,
$M^\pi(f)(v) = M(f)(\pi_f(v))$
Basic Idea

\[ M(f) \]

\[ M(A_f) \rightarrow M(f(A_f)) \]

\[ M^\pi(f) \]

\[ \pi_f \]

\[ M(A_f) \rightarrow M(f(A_f)) \]
Basic Idea

Given a model $\mathbf{M}$ for $F^*$,
Build a model $\mathbf{M}^\pi$ for $F$

In our example, we have: $h(b)$ and $h(c)$

$A_h = \{ b, c \}$, and $M(A_h) = \{ 2, 3 \}$

$\pi_h = \{ 2 \rightarrow 2, 3 \rightarrow 3, \text{else} \rightarrow 3 \}$

$M(h) = \{ 2 \rightarrow 0, 3 \rightarrow 1, \ldots \}$

$M^\pi(h) = \lambda x. \text{if}(x=2, 0, 1)$
Example

\[ g(x_1, x_2) = 0 \lor h(x_2) = 0, \]
\[ g(f(x_1), b) + 1 \leq f(x_1), \]
\[ h(c) = 1, \]
\[ f(a) = 0 \]

\[ F \]

\[ F^* \]

\[ h(c) = 1, \]
\[ f(a) = 0, \]
\[ g(f(a), b) + 1 \leq f(a), \]
\[ g(f(a), b) = 0 \lor h(b) = 0, \]
\[ g(f(a), c) = 0 \lor h(c) = 0 \]

\[ M^\pi \]

\[ a \rightarrow 2, \ b \rightarrow 2, \ c \rightarrow 3 \]
\[ f \rightarrow \lambda x. \ 2 \]
\[ h \rightarrow \lambda x. \ \text{if}(x=2, \ 0, \ 1) \]
\[ g \rightarrow \lambda x,y. \ \text{if}(x=0 \land y=2, -1, \ 0) \]

\[ M \]

\[ a \rightarrow 2, \ b \rightarrow 2, \ c \rightarrow 3 \]
\[ f \rightarrow \{ \ 2 \rightarrow 0, \ \ldots \} \]
\[ h \rightarrow \{ \ 2 \rightarrow 0, \ 3 \rightarrow 1, \ \ldots \} \]
\[ g \rightarrow \{ \ [0,2] \rightarrow -1, \ [0,3] \rightarrow 0, \ \ldots \} \]
Example: Model Checking

$M^\pi$

$a \rightarrow 2, b \rightarrow 2, c \rightarrow 3$

$f \rightarrow \lambda x. 2$

$h \rightarrow \lambda x. \text{if}(x=2, 0, 1)$

$g \rightarrow \lambda x,y. \text{if}(x=0 \land y=2,-1,0)$

Does $M^\pi$ satisfies?

$\forall x_1, x_2 : g(x_1, x_2) = 0 \lor h(x_2) = 0$

$\forall x_1, x_2 : \text{if}(x_1=0 \land x_2=2,-1,0) = 0 \lor \text{if}(x_2=2,0,1) = 0$ is valid

$\exists x_1, x_2 : \text{if}(x_1=0 \land x_2=2,-1,0) \neq 0 \land \text{if}(x_2=2,0,1) \neq 0$ is unsat

$\text{if}(s_1=0 \land s_2=2,-1,0) \neq 0 \land \text{if}(s_2=2,0,1) \neq 0$ is unsat
Why does it work?

Suppose $M^\pi$ does not satisfy $C[f(x)]$.

Then for some value $v$,
$M^\pi\{x \rightarrow v\}$ falsifies $C[f(x)]$.

$M^\pi\{x \rightarrow \pi_f(v)\}$ also falsifies $C[f(x)]$.

But, there is a term $t \in A_f$ s.t. $M(t) = \pi_f(v)$
Moreover, we instantiated $C[f(x)]$ with $t$.

So, $M$ must not satisfy $C[f(t)]$.
Contradiction: $M$ is a model for $F^\ast$. 
Refinement: Lazy construction

$F^*$ may be very big (or infinite).

Lazy-construction

Build $F^*$ incrementally, $F^*$ is the limit of the sequence $F^0 \subset F^1 \subset \ldots \subset F^k \subset \ldots$

If $F^k$ is unsat then $F$ is unsat.

If $F^k$ is sat, then build (candidate) $M^\pi$

If $M^\pi$ satisfies all quantifiers in $F$ then return sat.
Refinement: Model-based instantiation

Suppose $M^\pi$ does not satisfy a clause $C[f(x)]$ in $F$.

Add an instance $C[f(t)]$ which “blocks” this spurious model. Issue: how to find $t$?

Use model checking, and the “inverse” mapping $\pi_f^{-1}$ from values to terms (in $A_f$).

$\pi_f^{-1}(v) = t$ if $M^\pi(t) = \pi_f(v)$
Example: Model-based instantiation

\[ F \]
\[ \forall x_1: f(x_1) < 0, \]
\[ f(a) = 1, \]
\[ f(b) = -1 \]

\[ F^0 \]
\[ f(a) = 1, \]
\[ f(b) = -1 \]

\[ M^\pi \]
\[ a \mapsto 2, b \mapsto 3 \]
\[ f \mapsto \lambda x. \text{if}(x = 2, 1, -1) \]

---

**Model Checking**
\[ \forall x_1: f(x_1) < 0 \]
\[ \text{not if}(s_1 = 2, 1, -1) < 0 \]

\[ F^1 \]
\[ f(a) = 1, \]
\[ f(b) = -1 \]
\[ f(a) < 0 \]

unsat

\[ s_1 \mapsto 2 \]
\[ \pi_f^{-1}(2) = a \]
Infinite F*

Is refutationally complete?

FOL Compactness

A set of sentences is unsatisfiable iff it contains an unsatisfiable finite subset.

A theory $T$ is a set of sentences, then apply compactness to $F* \cup T$
Infinite F* : Example

F
∀x₁: f(x₁) < f(f(x₁)),
∀x₁: f(x₁) < a,
1 < f(0).

F*

f(0) < f(f(0)), f(f(0)) < f(f(f(0))), ...
f(0) < a, f(f(0)) < a, ...
1 < f(0)

Every finite subset of F* is satisfiable.

Unsatisfiable
Infinite $F^*$: What is wrong?

Theory of linear arithmetic $T_Z$ is the set of all first-order sentences that are true in the standard structure $Z$. $T_Z$ has non-standard models. $F$ and $F^*$ are satisfiable in a non-standard model.

Alternative: a theory is a class of structures. Compactness does not hold. $F$ and $F^*$ are still equisatisfiable.
Extensions

Shifting

\neg(0 \leq x_1) \lor \neg(x_1 \leq n) \lor f(x_1) = g(x_1+2)
Extensions

Many-sorted logic
Pseudo-Macros

\[ 0 \leq g(x_1) \lor f(g(x_1)) = x_1, \]
\[ 0 \leq g(x_1) \lor h(g(x_1)) = 2x_1, \]
\[ g(a) < 0 \]
Extensions

Online tutorial at:
http://rise4fun.com/z3/tutorial
Extensions

Online tutorial at:
http://rise4fun.com/z3/tutorial
Related work

Bernays-Schönfinkel class.
Stratified Many-Sorted Logic.
Array Property Fragment.
Local theory extensions.
SMT + Saturation
CDCL/DPLL: Review

- Partial model
- Set of clauses

M | F
Guessing

\[ p \mid p \lor q, \neg q \lor r \]

\[ p, \neg q \mid p \lor q, \neg q \lor r \]
Deducing

\[ p \mid p \lor q, \neg p \lor s \]

\[ p, s \mid p \lor q, \neg p \lor s \]
CDCL/DPLL : Review

Backtracking

\[ p, \neg s, q \mid p \lor q, s \lor q, \neg p \lor \neg q \]

\[ p, s \mid p \lor q, s \lor q, \neg p \lor \neg q \]
DPLL(Γ)

Tight integration: DPLL + Saturation solver.
DPLL(Γ)

Inference rule:

\[
\begin{array}{c}
C_1 \\ \vdots \\ C_n \\
C
\end{array}
\]

DPLL(Γ) is parametric.

Examples:

Resolution

Superposition calculus

...
DPLL(\(\Gamma\))

- Partial model
- Set of clauses

M | F
DPLL(Γ) : Deduce I

\[ p(a) \mid p(a) \lor q(a), \ \forall x: \neg p(x) \lor r(x), \ \forall x: p(x) \lor s(x) \]
DPLL(\( \Gamma \)) : Deduce I

\begin{align*}
p(a) \mid p(a) \lor q(a), & \quad \neg p(x) \lor r(x), \quad p(x) \lor s(x)
\end{align*}
DPLL(Γ) : Deduce I

\[ p(a) \mid p(a) \lor q(a), \neg p(x) \lor r(x), p(x) \lor s(x) \]

Resolution

\[ p(a) \mid p(a) \lor q(a), \neg p(x) \lor r(x), p(x) \lor s(x), r(x) \lor s(x) \]
DPLL(Γ) : Deduce II

Using ground atoms from M:

\[ M \mid F \]

Main issue: backtracking.

Hypothetical clauses:

\[ H \models C \]

Track literals from M used to derive C

(hypothesis) Ground literals

(regular) Clause
DPLL(Γ) : Deduce II

\[ p(a) \mid p(a) \lor q(a), \neg p(x) \lor r(x) \]

\[ p(a), \neg p(x) \lor r(x) \]

\[ r(a) \]

\[ p(a) \mid p(a) \lor q(a), \neg p(x) \lor r(x), p(a) \triangleright r(a) \]
DPLL(Γ) : Backtracking

\( p(a), r(a) \mid p(a) \lor q(a), \neg p(a) \lor \neg r(a), p(a) \triangleright r(a), \ldots \)
DPLL(\(\Gamma\)) : Backtracking

\[ p(a), \neg r(a) \mid p(a) \lor q(a), \neg p(a) \lor \neg r(a), p(a) \not\triangleright r(a), \ldots \]

\[ \neg p(a) \mid p(a) \lor q(a), \neg p(a) \lor \neg r(a), \ldots \]

\(p(a)\) is removed from \(M\)
DPLL(Γ) : Improvement

Saturation solver ignores non-unit ground clauses.

\[ p(a) | p(b) \lor c(a), \neg p(x) \lor r(x) \]
DPLL(Γ) : Improvement

Saturation solver ignores non-unit ground clauses. It is still refutationally complete if:

Γ has the reduction property.
DPLL(Γ) : Improvement

Saturation solver ignores non-unit ground clauses. It is still refutationally complete if:

- Γ has the reduction property.
DPLL(Γ) : Problem

Interpreted symbols
\( \neg(f(a) > 2), \quad f(x) > 5 \)

It is refutationally complete if

Interpreted symbols only occur in ground clauses
Non ground clauses are variable inactive
“Good” ordering is used
Summary

E-matching

proof finding

fast

shallow proofs in big formulas

not refutationally complete

regularly solves VCs with more than 5 Mb
Summary

Complete instantiation + MBQI
decides several useful fragments
model & proof finding
slow
complements E-matching
Summary

SMT + Saturation
refutationally complete for pure first-order proof finding
slow
Not covered

Quantifier elimination

Fourier-Motzkin (Linear Real Arithmetic)

Cooper (Linear Integer Arithmetic)

CAD (Nonlinear Real Arithmetic)

Algebraic Datatypes (Hodges)

Finite model finding

Many Decidable Fragments
Challenge

New and efficient procedures capable of producing models for satisfiable instances.