# E-Matching with Free Variables

Philipp Rümmer
Uppsala University
Sweden

FATPA Workshop Belgrade
February 3rd 2012

# Context: reasoning in first-order logic (FOL)

| First-order provers | SMT solvers |
|---|---|
| Resolution, superposition, **tableaux**, etc. | **DPLL(T)**, Nelson-Oppen |
| (Free) **variables**, unification | **E-matching**, heuristics |
| **Complete** for FOL | Complete on ground fragment |
| | Many built-in **theories** |
| Great for algebra, not so much for verification | Fast, but incomplete on quantified problems |

# How about putting things together?

This is possible. Here:

- **KE-tableau/DPLL** — FOL
- **Theory procedures** — Arithmetic
- **E-matching** — Axiomatisation of theories
- **Free variables + constraints** — Quantifiers

- Interesting completeness results
- Experimental implementation: PRINCESS
- In some domains:
  Performance comparable to SMT solvers
- Some features that are rather unique

# How about putting things together?

This is possible. Here:

| | |
|---|---|
| **KE-tableau/DPLL** | FOL |
| **Theory procedures** | Arithmetic |
| **E-matching** | Axiomatisation of theories |
| **Free variables + constraints** | Quantifiers |

- Interesting completeness results
- Experimental implementation: PRINCESS
- In some domains:
  Performance comparable to SMT solvers
- Some features that are rather unique

# Outline

- The **base logic + calculus**:
  Linear integer arithmetic + uninterpreted predicates
- Positive Unit Hyper-Resolution (**PUHR**)
- Uninterpreted **functions**:
  Encoding + Axioms
- **E-matching**
- **Experiment**al results

More details: paper at LPAR 2012

# The base logic



**Linear integer arithmetic + uninterpreted predicates:**

$$t ::= \alpha \mid x \mid c \mid \alpha t + \cdots + \alpha t$$

$$\phi ::= \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \forall x.\phi \mid \exists x.\phi$$
$$\mid t \doteq 0 \mid t \dot{\geq} 0 \mid t \dot{\leq} 0 \mid \alpha \mid t \mid p(t, \ldots, t)$$

$t$ ... terms

$\phi$ ... formulae

$x$ ... variables

$c$ ... constants

$p$ ... uninterpreted predicates (fixed arity)

$\alpha$ ... integer literals ($\mathbb{Z}$)

# The base logic



**Linear integer arithmetic + uninterpreted predicates:**

$$t ::= \alpha \mid x \mid c \mid \alpha t + \cdots + \alpha t$$

$$\phi ::= \phi \land \phi \mid \phi \lor \phi \mid \neg\phi \mid \forall x.\phi \mid \exists x.\phi \mid t \doteq 0 \mid t \dot{\geq} 0 \mid t \dot{\leq} 0 \mid \alpha \mid t \mid p(t, \ldots, t)$$

- **No functions!** (more later)
- Subsumes FOL and Presburger arithmetic (PA)
- Valid formulae are not enumerable [Halpern, 1991]

# Example formula: optimisation

```
  \forall int x, y; (
    p(x, y) <-> (2*x + y <= 18 &
                 2*x + 3*y <= 42 &
                 3*x + y <= 24 &
                 x >= 0 & y >= 0)
  )
->
  \exists int x, y; (
    p(x, y) &
    \forall int x2, y2; (
      p(x2, y2) -> 3*x + 2*y >= 3*x2 + 2*y2)
  )
```

# Abstract calculus

Input formula (with preds.): $\phi$

# Abstract calculus

Input formula (with preds.): $\phi$

$\Uparrow$

Compute PA approximation: $C_0$

# Abstract calculus

Input formula (with preds.): $\phi$

$$\Uparrow$$

Compute PA approximation: $C_0$

$$C_0 \textbf{ is valid} \implies \phi \textbf{ is valid}$$

# Abstract calculus

Input formula (with preds.): $\phi$

$\Uparrow$

Compute PA approximation: $C_0$

**$C_0$ is invalid . . . refine approximation**

# Abstract calculus

Input formula (with preds.): $\phi$

$$\Uparrow \quad \nwarrow$$

Compute PA approximation: $C_0 \Rightarrow C_1$

**$C_0$ is invalid . . . refine approximation**

# Abstract calculus

$$\begin{array}{lccccccc}
\text{Input formula (with preds.):} & \phi & & & & & \\
 & \Uparrow & \nwarrow & & & & \\
\text{Compute PA approximation:} & C_0 & \Rightarrow & C_1 & \Rightarrow & C_2 & \cdots
\end{array}$$

**$C_0$ is invalid . . . refine approximation**

# Abstract calculus

$$\begin{array}{llllllll} \text{Input formula (with preds.):} & \phi & & & & & & \\ & \Uparrow & \nwarrow & & & & & \\ \text{Compute PA approximation:} & C_0 & \Rightarrow & C_1 & \Rightarrow & C_2 & \cdots \end{array}$$

**$C_0$ is invalid . . . refine approximation**

**Any $C_i$ is valid $\implies$ $\phi$ is valid**

# Approximation? Constrained sequents!

## Notation used here:

$$\underbrace{\Gamma \vdash \Delta}_{} \underbrace{\Downarrow C}_{}$$

Antecedent, Succedent (sets of formulae)

Constraint/approximation (formula)

## Definition

$\Gamma \vdash \Delta \Downarrow C$ is *valid* if the formula $C \to \bigwedge \Gamma \to \bigvee \Delta$ is valid.

# Iterative proof construction

$$\Gamma \ \vdash \ \Delta \ \Downarrow ?$$

# Iterative proof construction

analytic reasoning
about input formula $\uparrow$

$$\Gamma \vdash \Delta \Downarrow ?$$

# Iterative proof construction

analytic reasoning
about input formula $\uparrow$

$$\begin{array}{c} \Gamma_1 \vdash \Delta_1 \Downarrow ? \\ \vdots \\ \Gamma \vdash \Delta \Downarrow ? \end{array}$$

# Iterative proof construction

analytic reasoning about input formula $\uparrow$

$$\dfrac{\Gamma_2 \vdash \Delta_2 \Downarrow ?}{\Gamma_1 \vdash \Delta_1 \Downarrow ?}$$

$$\vdots$$

$$\Gamma \vdash \Delta \Downarrow ?$$

# Iterative proof construction

analytic reasoning
about input formula ↑

$$\dfrac{\dfrac{\Gamma_3 \;\vdash\; \Delta_3 \;\Downarrow ?}{\Gamma_2 \;\vdash\; \Delta_2 \;\Downarrow ?}}{\Gamma_1 \;\vdash\; \Delta_1 \;\Downarrow ?}$$

$$\vdots$$

$$\Gamma \;\vdash\; \Delta \;\Downarrow ?$$

# Iterative proof construction

analytic reasoning about input formula $\uparrow$

$$\begin{array}{c} * \\ \vdots \\ \Gamma_3 \vdash \Delta_3 \Downarrow ? \\ \hline \Gamma_2 \vdash \Delta_2 \Downarrow ? \\ \hline \Gamma_1 \vdash \Delta_1 \Downarrow ? \\ \vdots \\ \Gamma \vdash \Delta \Downarrow ? \end{array}$$

# Iterative proof construction

$$\dfrac{\dfrac{\dfrac{\vdots^{*}}{\Gamma_3 \vdash \Delta_3 \Downarrow ?}}{\Gamma_2 \vdash \Delta_2 \Downarrow ?}}{\Gamma_1 \vdash \Delta_1 \Downarrow ?}$$

$$\vdots$$

$$\Gamma \vdash \Delta \Downarrow ?$$

analytic reasoning about input formula $\uparrow$

$\downarrow$ propagation of constraints

# Iterative proof construction

$$\begin{array}{c} * \\ \vdots \\ \hline \Gamma_3 \vdash \Delta_3 \Downarrow C_1 \\ \hline \Gamma_2 \vdash \Delta_2 \Downarrow ? \\ \hline \Gamma_1 \vdash \Delta_1 \Downarrow ? \\ \vdots \\ \Gamma \vdash \Delta \Downarrow ? \end{array}$$

analytic reasoning about input formula $\uparrow$

$\downarrow$ propagation of constraints

# Iterative proof construction

$$\dfrac{\dfrac{\begin{array}{c}*\\ \vdots\\ \Gamma_3 \vdash \Delta_3 \Downarrow C_1\end{array}}{\Gamma_2 \vdash \Delta_2 \Downarrow C_2}}{\begin{array}{c}\Gamma_1 \vdash \Delta_1 \Downarrow ?\\ \vdots\\ \Gamma \vdash \Delta \Downarrow ?\end{array}}$$

analytic reasoning about input formula $\uparrow$

$\downarrow$ propagation of constraints

# Iterative proof construction

analytic reasoning about input formula $\uparrow$

$$
\begin{array}{c}
* \\
\vdots \\
\Gamma_3 \vdash \Delta_3 \Downarrow C_1 \\
\hline
\Gamma_2 \vdash \Delta_2 \Downarrow C_2 \\
\hline
\Gamma_1 \vdash \Delta_1 \Downarrow C_3 \\
\vdots \\
\Gamma \vdash \Delta \Downarrow ?
\end{array}
$$

$\downarrow$ propagation of constraints

# Iterative proof construction

$$\cfrac{\cfrac{\vdots^{*}}{\Gamma_3 \vdash \Delta_3 \Downarrow C_1}}{\cfrac{\Gamma_2 \vdash \Delta_2 \Downarrow C_2}{\cfrac{\Gamma_1 \vdash \Delta_1 \Downarrow C_3}{\vdots \atop \Gamma \vdash \Delta \Downarrow C}}}$$

analytic reasoning about input formula $\uparrow$

$\downarrow$ propagation of constraints

# Iterative proof construction

analytic reasoning about input formula $\uparrow$

$$
\begin{array}{c}
*\\
\vdots\\
\Gamma_3 \vdash \Delta_3 \Downarrow {\color{blue}C_1}\\
\hline
\Gamma_2 \vdash \Delta_2 \Downarrow {\color{blue}C_2}\\
\hline
\Gamma_1 \vdash \Delta_1 \Downarrow {\color{blue}C_3}\\
\vdots\\
\Gamma \vdash \Delta \Downarrow {\color{blue}C}
\end{array}
$$

$\downarrow$ propagation of constraints

- Constraints are **simplified** during propagation
- If $C$ is **valid**, then so is $\Gamma \vdash \Delta$
- If $C$ is **satisfiable**, it describes a solution for $\Gamma \vdash \Delta$
- If $C$ is unsatisfiable, expand the proof tree further ...

# A few proof rules

$$\frac{\Gamma \vdash \phi, \Delta \Downarrow C \qquad \Gamma \vdash \psi, \Delta \Downarrow D}{\Gamma \vdash \phi \wedge \psi, \Delta \Downarrow C \wedge D} \text{ AND-RIGHT}$$

$$\frac{\Gamma, [x/c]\phi, \forall x.\phi \vdash \Delta \Downarrow [x/c]C}{\Gamma, \forall x.\phi \vdash \Delta \Downarrow \exists x.C} \text{ ALL-LEFT}$$

($c$ is fresh)

$$\frac{\Gamma, p(\bar{s}) \vdash p(\bar{t}), \bar{s} \doteq \bar{t}, \Delta \Downarrow C}{\Gamma, p(\bar{s}) \vdash p(\bar{t}), \Delta \Downarrow C} \text{ PRED-UNIFY}$$

$$\frac{*}{\Gamma, \phi_1, \ldots, \phi_n \vdash \psi_1, \ldots, \psi_m, \Delta \Downarrow \neg\phi_1 \vee \cdots \vee \neg\phi_n \vee \psi_1 \vee \cdots \vee \psi_m} \text{ CLOSE}$$

(selected formulae are predicate-free)

# Correctness

**Lemma (Soundness)**

*It's sound!*

**Lemma (Completeness)**

*Complete for fragments:*

- *FOL*
- *PA*
- *Purely existential formulae*
- *Purely universal formulae*
- *Universal formulae with finite parametrisation (same as $\mathcal{ME}$(LIA))*

# Practicality

# Practicality

So far: **quantifier instantiation** is always **delayed**:

$$\dfrac{\dfrac{\dfrac{\vdots}{\ldots, p(\bar{s}) \vdash p(\bar{t}),\ \bar{s} \doteq \bar{t}, \ldots}}{\ldots, p(\bar{s}) \vdash p(\bar{t}), \ldots}\ \text{PRED-UNIFY}}{}$$

$$\vdots$$

$$\dfrac{\dfrac{\vdots}{\Gamma, [x/c]\phi, \forall x.\phi \vdash \Delta}}{\Gamma, \forall x.\phi \vdash \Delta}\ \text{ALL-LEFT}$$

$$\vdots$$

# Practicality

So far: **quantifier instantiation** is always **delayed**:

$$\dfrac{\dfrac{\vdots}{\ldots, p(\bar{s}) \ \vdash \ p(\bar{t}),\ \bar{s} \doteq \bar{t}, \ldots}}{\ldots, p(\bar{s}) \ \vdash \ p(\bar{t}), \ldots} \ \textsc{Pred-Unify}$$

$$\dfrac{\dfrac{\vdots}{\Gamma, [x/c]\phi, \forall x.\phi \ \vdash \ \Delta}}{\Gamma, \forall x.\phi \ \vdash \ \Delta} \ \textsc{All-Left}$$

$$\vdots$$

This corresponds to . . .

- **Free variables + Unification**
- Standard approach in **FOL provers**

# Alternative: E-Matching, standard in **SMT solvers**

Matching of **triggers** (modulo equations):

$$\frac{\Gamma, \forall \bar{x}.\phi[t[\bar{x}]], [\bar{x}/\bar{s}]\phi[t[\bar{x}]] \ \vdash \ \psi[t[\bar{s}]], \Delta}{\Gamma, \forall \bar{x}.\phi[t[\bar{x}]] \ \vdash \ \psi[t[\bar{s}]], \Delta}$$

# Alternative: E-Matching, standard in **SMT solvers**

Matching of **triggers** (modulo equations):

$$\frac{\Gamma, \forall \bar{x}.\phi[t[\bar{x}]], [\bar{x}/\bar{s}]\phi[t[\bar{x}]] \;\vdash\; \psi[t[\bar{s}]], \Delta}{\Gamma, \forall \bar{x}.\phi[t[\bar{x}]] \;\vdash\; \psi[t[\bar{s}]], \Delta}$$

```
\forall int a, i, v;
   select(store(a, i, v), i) = v

\forall int a, i1, i2, v;
   (i1 != i2 ->
   select(store(a, i1, v), i2) = select(a, i2))
```

# Alternative: E-Matching, standard in **SMT solvers**

Matching of **triggers** (modulo equations):

$$\frac{\Gamma, \forall \bar{x}.\phi[t[\bar{x}]], [\bar{x}/\bar{s}]\phi[t[\bar{x}]] \ \vdash \ \psi[t[\bar{s}]], \Delta}{\Gamma, \forall \bar{x}.\phi[t[\bar{x}]] \ \vdash \ \psi[t[\bar{s}]], \Delta}$$

```
\forall int a, i, v;
   select(store(a, i, v), i) = v

\forall int a, i1, i2, v;
   (i1 != i2 ->
   select(store(a, i1, v), i2) = select(a, i2))
```

# Comparison

| **E-Matching** | **Free variables + unification** |
|---|---|
| Heuristic $\rightarrow$ **incomplete** | **Systematic** |
| Good for "simple" instances | Can find "difficult" instances |
| **User guidance** possible<br>$\rightarrow$ Triggers | |
| Quite **fast**<br>$\rightarrow$ Only **ground** formulae | Quite **expensive**<br>$\rightarrow$ Very **nondeterministic** |

# Comparison

| **E-Matching** | **Free variables + unification** |
|---|---|
| Heuristic → **incomplete** | **Systematic** |
| Good for “simple” instances | Can find “difficult” instances |
| **User guidance** possible<br>→ Triggers | |
| Quite **fast**<br>→ Only **ground** formulae | Quite **expensive**<br>→ Very **nondeterministic** |

Combination?

# Comparison

| **E-Matching** | **Free variables + unification** |
|---|---|
| Heuristic → **incomplete** | **Systematic** |
| Good for “simple” instances | Can find “difficult” instances |
| **User guidance** possible<br>→ Triggers | |
| Quite **fast**<br>→ Only **ground** formulae | Quite **expensive**<br>→ Very **nondeterministic** |

Combination!

1. For **predicates**:
   Positive unit hyper-resolution (PUHR)
2. Lifted to **functions** using encoding

# Positive Unit Hyper-Resolution [Manthey, Bry]

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

$$\overline{\forall x.p(x), \forall x.\big(p(x) \to q(x) \vee r(x+1)\big), \forall x.\neg r(x) \;\vdash\; q(a)}$$

# Positive Unit Hyper-Resolution [Manthey, Bry]

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

$$\forall x.p(x), \forall x.\big(p(x) \to q(x) \vee r(x+1)\big), \forall x.\neg r(x) \ \vdash \ q(a)$$

# Positive Unit Hyper-Resolution [Manthey, Bry]

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

$$\dfrac{\dfrac{}{\ldots, p(X) \ \vdash}}{\forall x.p(x), \forall x.\big(p(x) \to q(x) \lor r(x+1)\big), \forall x.\neg r(x) \ \vdash \ q(a)}$$

# Positive Unit Hyper-Resolution [Manthey, Bry]

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

$$\cfrac{\cfrac{}{\ldots, p(X) \;\vdash}}{\forall x.p(x), \forall x.(p(x) \to q(x) \vee r(x+1)), \forall x.\neg r(x) \;\vdash\; q(a)}$$

# Positive Unit Hyper-Resolution [Manthey, Bry]

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

$$\dfrac{\dfrac{\overline{\qquad q(X) \vee r(X+1) \;\vdash \qquad}}{\ldots, p(X) \;\vdash}}{\forall x.p(x), \forall x.\big(p(x) \rightarrow q(x) \vee r(x+1)\big), \forall x.\neg r(x) \;\vdash\; q(a)}$$

# Positive Unit Hyper-Resolution [Manthey, Bry]

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{q(X) \ \vdash} \qquad \dfrac{}{r(X+1) \ \vdash}}{q(X) \vee r(X+1) \ \vdash}}{\ldots, p(X) \ \vdash}}{\forall x.p(x), \forall x.\big(p(x) \rightarrow q(x) \vee r(x+1)\big), \forall x.\neg r(x) \ \vdash \ q(a)}
$$

# Positive Unit Hyper-Resolution [Manthey, Bry]

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{q(X) \vdash} \qquad \dfrac{}{r(X+1) \vdash}}{q(X) \lor r(X+1) \vdash}}{\ldots, p(X) \vdash}}{\forall x.p(x), \forall x.(p(x) \to q(x) \lor r(x+1)), \forall x.\neg r(x) \vdash q(a)}$$

# Positive Unit Hyper-Resolution [Manthey, Bry]

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{q(X)\ \vdash} \qquad \dfrac{\dfrac{*}{\mathit{false}\ \vdash}}{r(X+1)\ \vdash}}{q(X) \lor r(X+1)\ \vdash}}{\ldots, p(X)\ \vdash}}{\forall x.p(x), \forall x.\big(p(x) \rightarrow q(x) \lor r(x+1)\big), \forall x.\neg r(x)\ \vdash\ q(a)}$$

# Positive Unit Hyper-Resolution [Manthey, Bry]

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{q(X) \vdash} \qquad \dfrac{\dfrac{*}{\mathit{false} \vdash}}{r(X+1) \vdash}}{q(X) \vee r(X+1) \vdash}}{\ldots, p(X) \vdash}}{\forall x.p(x), \forall x.(p(x) \rightarrow q(x) \vee r(x+1)), \forall x.\neg r(x) \vdash q(a)}$$

Directed instantiation of formulae:

- Formulae with **negative literals**:
  $\Rightarrow$ Discharge with **unit resolution**
- Formulae **without negative literals**:
  $\Rightarrow$ Instantiate with **free variables**
  (or: enumerate ground terms)

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{q(X) \vdash \quad \Downarrow X \doteq a} \qquad \dfrac{\dfrac{*}{\mathit{false} \vdash}}{r(X+1) \vdash}}{q(X) \lor r(X+1) \vdash}}{\ldots, p(X) \vdash}}{\forall x.p(x), \forall x.(p(x) \to q(x) \lor r(x+1)), \forall x.\neg r(x) \vdash q(a)}$$

# PUHR in our calculus

**Theorem (Completeness)**

*Suppose $\Gamma \vdash \Delta \Downarrow C$ is provable in the calculus without PUHR, where $C$ is valid. Then there is a valid constraint $C'$ so that the calculus with PUHR can prove $\Gamma \vdash \Delta \Downarrow C'$.*

In PRINCESS:

- PUHR normally yields **drastic speed-up**
- (but not always)

# Lifting to functions

# Lifting to functions

Functions almost like in SMT:

- Terms are always **flattened**
- $n$-ary **function** $f$ becomes $(n+1)$-ary **predicate** $f_p$
  E.g.

$$
\begin{aligned}
g(f(x), a) \quad &\rightsquigarrow \quad f(x) = c \wedge g(c, a) = d \\
&\rightsquigarrow \quad f_p(x, c) \wedge g_p(c, a, d)
\end{aligned}
$$

# Lifting to functions

Functions almost like in SMT:

- Terms are always **flattened**
- $n$-ary **function** $f$ becomes $(n+1)$-ary **predicate** $f_p$
  E.g.

$$
\begin{aligned}
g(f(x), a) \quad & \rightsquigarrow \quad f(x) = c \land g(c, a) = d \\
& \rightsquigarrow \quad f_p(x, c) \land g_p(c, a, d)
\end{aligned}
$$

- Axioms necessary: **Totality + Functionality**

$$
\begin{aligned}
& \forall \bar{x}. \exists y.\ f_p(\bar{x}, y) \\
& \forall \bar{x}, y_1, y_2.\ (f_p(\bar{x}, y_1) \to f_p(\bar{x}, y_2) \to y_1 \doteq y_2)
\end{aligned}
$$

# Lifting to functions

Functions almost like in SMT:

- Terms are always **flattened**
- *n*-ary **function** $f$ becomes $(n+1)$-ary **predicate** $f_p$
  E.g.

$$g(f(x), a) \quad \rightsquigarrow \quad f(x) = c \wedge g(c, a) = d$$
$$\rightsquigarrow \quad f_p(x, c) \wedge g_p(c, a, d)$$

- Axioms necessary: **Totality + Functionality**

$$\forall \bar{x}. \exists y.\ f_p(\bar{x}, y)$$
$$\forall \bar{x}, y_1, y_2.\ (f_p(\bar{x}, y_1) \rightarrow f_p(\bar{x}, y_2) \rightarrow y_1 \doteq y_2)$$

- Very closely resembles **congruence closure**

# E-Matching through PUHR

**Two ways** to encode function applications:

$$\phi[f(\bar{t})] \quad \rightsquigarrow \quad \forall y.(\neg f_p(\bar{t}, y) \lor \phi[y]) \qquad \text{(negative)}$$

$$\rightsquigarrow \quad \exists y.(f_p(\bar{t}, y) \land \phi[y]) \qquad \text{(positive)}$$

# E-Matching through PUHR

**Two ways** to encode function applications:

$$\phi[f(\bar{t})] \quad \rightsquigarrow \quad \forall y.(\neg f_p(\bar{t}, y) \vee \phi[y]) \qquad \text{(negative)}$$

$$\rightsquigarrow \quad \exists y.(f_p(\bar{t}, y) \wedge \phi[y]) \qquad \text{(positive)}$$

$\Rightarrow$ **Useful: PUHR** only matches on **negative** literals

# E-Matching through PUHR

**Two ways** to encode function applications:

$$
\begin{aligned}
\phi[f(\bar{t})] \quad &\rightsquigarrow \quad \forall y.(\neg f_p(\bar{t}, y) \vee \phi[y]) \qquad &\text{(negative)} \\
&\rightsquigarrow \quad \exists y.(f_p(\bar{t}, y) \wedge \phi[y]) \qquad &\text{(positive)}
\end{aligned}
$$

$\Rightarrow$ **Useful: PUHR** only matches on **negative** literals

$$\forall \bar{x}.\phi[t[\bar{x}]]$$

**negative** encoding
for trigger $t[\bar{x}]$

**positive** encoding
for other functions

# Example

$$\forall x.\ f(x) \geq 0$$

If $f(x)$ is **trigger**:
$\forall x, y.\ \left(\neg f_p(x, y) \lor y \geq 0\right)$

If $f(x)$ is **not trigger**:
$\forall x. \exists y.\ \left(f_p(x, y) \land y \geq 0\right)$

# The highlight: relative completeness

In **SMT solvers**:

- Choice of triggers determines **provability**
- Bad triggers $\rightarrow$ bad luck

In the **PUHR calculus**:

- Choice of triggers determines **performance**
- Regardless of triggers, **the same formulae are provable**
- E-matching is complemented by
  **free variables + unification**

# Where are we? Experimental evaluation

| | **AUFLIA+p** (193) | **AUFLIA-p** (193) |
|---|---|---|
| Z3 | 191 | 191 |
| **PRINCESS** | **145** | **137** |
| CVC3 | 132 | 128 |

- Implementation of our calculus in PRINCESS
- Unsatisfiable AUFLIA benchmarks from SMT-comp 2011
- Intel Core i5 2-core, 3.2GHz, timeout 1200s, 4Gb
- `http://www.philipp.ruemmer.org/princess.shtml`

# Conclusion

- **E-Matching** = **Relational function encoding** + **PUHR**
- Overall goal:
  Tools that provide the **performance** of SMT solvers,
  but **completeness** as common in FOL provers
- Presented work is one step on this way

There is more to say, e.g.:

- Connection to **constraint programming**
- Theory of **arrays**, **sets**
- Handling of **bit-vectors**
- **Craig interpolation**

Thanks for your attention!

# Related work

- $\mathcal{ME}$(LIA): model evolution modulo linear integer arithmetic, [Baumgartner, Tinelli, Fuchs, 08]
- SPASS+T [Prevosto, Waldmann, ESCoR'06]
- DPLL($\mathcal{SP}$) [de Moura, Bjørner, IJCAR'08]

- Various approaches to integrate theories in saturation calculi, e.g. [Stickel, JAR'85], [Bürchert, CADE'90], [Korovin, Voronkov, CSL'07]
- Constraint logic programming
- Various SMT solvers

# Open PhD Position at Uppsala University

I’m looking to hire a **PhD student**:

- Subject areas:
  **SMT, floating-point arithmetic, Craig interpolation**;
  Application in embedded systems analysis
- Contact me for more information
- Pass on to students that might be interested