

# 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, <b>tableaux</b> , etc.	<b>DPLL(T)</b> , Nelson-Oppen
(Free) <b>variables</b> , unification	<b>E-matching</b> , heuristics
<b>Complete</b> for FOL	Complete on ground fragment
	Many built-in <b>theories</b>
Great for algebra, not so much for verification	Fast, but incomplete on quantified problems

# How about putting things together?

This is possible. Here:

- |                                       |                            |
|---------------------------------------|----------------------------|
| • <b>KE-tableau/DPLL</b>              | FOL                        |
| • <b>Theory procedures</b>            | Arithmetic                 |
| • <b>E-matching</b>                   | Axiomatisation of theories |
| • <b>Free variables + constraints</b> | 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:

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

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

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

More details: paper at LPAR 2012

Linear integer arithmetic + uninterpreted predicates:

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

$$\begin{aligned} \phi ::= & \phi \wedge \phi \mid \phi \vee \phi \mid \neg \phi \mid \forall x. \phi \mid \exists x. \phi \\ & \mid t \doteq 0 \mid t \geq 0 \mid t \leq 0 \mid \alpha \mid t \mid p(t, \dots, t) \end{aligned}$$

$t$  ... terms

$\phi$  ... formulae

$x$  ... variables

$c$  ... constants

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

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

Linear integer arithmetic + uninterpreted predicates:

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

$$\begin{aligned} \phi ::= & \phi \wedge \phi \mid \phi \vee \phi \mid \neg \phi \mid \forall x. \phi \mid \exists x. \phi \\ & \mid t \doteq 0 \mid t \geq 0 \mid t \leq 0 \mid \alpha \mid t \mid p(t, \dots, t) \end{aligned}$$

- **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)  
)
```

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:  $C_0$

**$C_0$  is valid  $\implies \phi$  is valid**

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

$\Uparrow$

Compute PA approximation:  $C_0$

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

Input formula (with preds.):  $\phi$   
 $\uparrow \quad \Downarrow$   
Compute PA approximation:  $C_0 \Rightarrow C_1$

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

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

$\Uparrow \quad \Downarrow$

Compute PA approximation:  $C_0 \Rightarrow C_1 \Rightarrow C_2 \dots$

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

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

$\Uparrow \quad \Downarrow$

Compute PA approximation:  $C_0 \Rightarrow C_1 \Rightarrow C_2 \dots$

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

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

# Approximation? Constrained sequents!

Notation used here:

$$\underbrace{\Gamma \vdash \Delta}_{\text{Antecedent, Succedent}} \underbrace{\Downarrow C}_{\text{Constraint/approximation}}$$

Antecedent, Succedent  
(sets of formulae)

Constraint/approximation  
(formula)

## Definition

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

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

# Iterative proof construction

analytic reasoning  
about input formula  $\uparrow$

$\Gamma \vdash \Delta \Downarrow ?$

# Iterative proof construction

analytic reasoning  
about input formula  $\uparrow$

$\Gamma_1 \vdash \Delta_1 \Downarrow ?$

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

# Iterative proof construction

analytic reasoning  
about input formula  $\uparrow$

$$\frac{\Gamma_2 \vdash \Delta_2 \Downarrow ?}{\Gamma_1 \vdash \Delta_1 \Downarrow ?}$$
$$\vdots$$
$$\Gamma \vdash \Delta \Downarrow ?$$

# Iterative proof construction

analytic reasoning  
about input formula  $\uparrow$

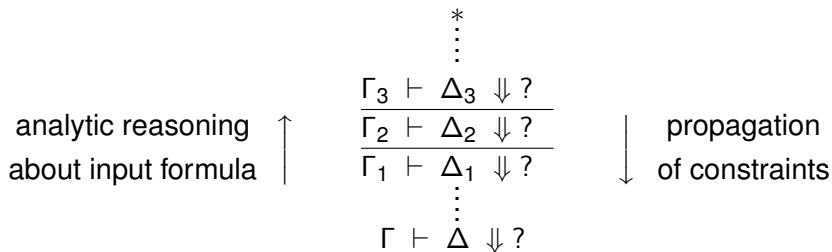
$$\frac{\frac{\frac{\Gamma_3 \vdash \Delta_3 \Downarrow ?}{\Gamma_2 \vdash \Delta_2 \Downarrow ?}}{\Gamma_1 \vdash \Delta_1 \Downarrow ?}}{\vdots}$$
$$\Gamma \vdash \Delta \Downarrow ?$$

# Iterative proof construction

analytic reasoning  
about input formula  $\uparrow$

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

# Iterative proof construction



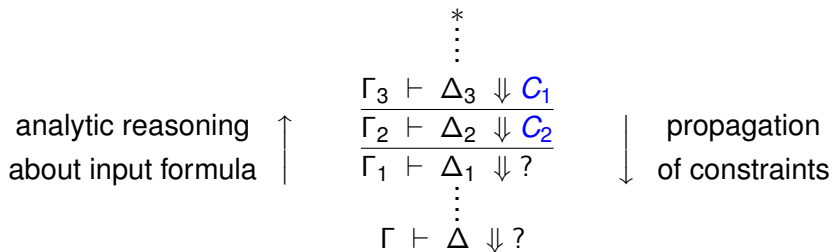
# Iterative proof construction

analytic reasoning  
about input formula  $\uparrow$

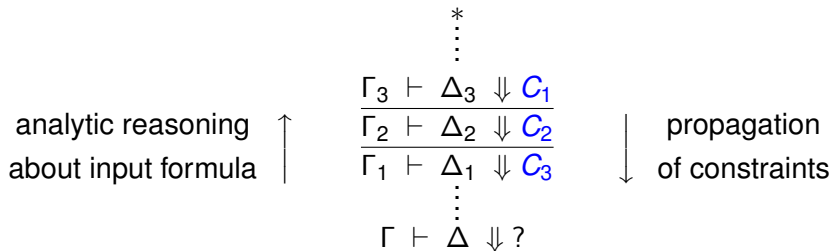
$$\begin{array}{c} * \\ \vdots \\ \hline \Gamma_3 \vdash \Delta_3 \Downarrow C_1 \\ \hline \Gamma_2 \vdash \Delta_2 \Downarrow ? \\ \hline \Gamma_1 \vdash \Delta_1 \Downarrow ? \\ \vdots \\ \Gamma \vdash \Delta \Downarrow ? \end{array}$$

$\downarrow$  propagation  
of constraints

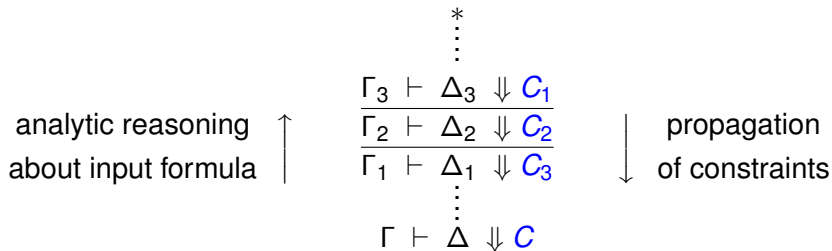
# Iterative proof construction



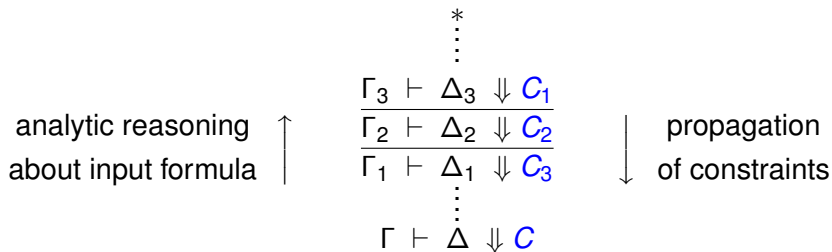
# Iterative proof construction



# Iterative proof construction



# Iterative proof construction



- 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

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

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

(c is fresh)

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

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

(selected formulae are predicate-free)

## 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  $\mathcal{ME}(LIA)$ )*



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

$$\begin{array}{c}
 \vdots \\
 \hline
 \dots, p(\bar{s}) \vdash p(\bar{t}), \bar{s} \doteq \bar{t}, \dots \\
 \hline
 \dots, p(\bar{s}) \vdash p(\bar{t}), \dots
 \end{array}
 \text{ PRED-UNIFY}$$
  

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

$$\vdots$$

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

$$\begin{array}{c}
 \vdots \\
 \hline
 \dots, p(\bar{s}) \vdash p(\bar{t}), \bar{s} \doteq \bar{t}, \dots \\
 \hline
 \dots, p(\bar{s}) \vdash p(\bar{t}), \dots
 \end{array}
 \text{ PRED-UNIFY}$$
  

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

$$\vdots$$

This corresponds to ...

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

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))
```

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))
```

## E-Matching

Heuristic → **incomplete**

Good for “simple” instances

**User guidance** possible

→ Triggers

Quite **fast**

→ Only **ground** formulae

## Free variables + unification

**Systematic**

Can find “difficult” instances

Quite **expensive**

→ Very **nondeterministic**

# Comparison

## E-Matching

Heuristic → **incomplete**

Good for “simple” instances

**User guidance** possible

→ Triggers

Quite **fast**

→ Only **ground** formulae

## Free variables + unification

**Systematic**

Can find “difficult” instances

Quite **expensive**

→ Very **nondeterministic**

Combination?

## E-Matching

Heuristic → **incomplete**

Good for “simple” instances

**User guidance** possible

→ Triggers

Quite **fast**

→ Only **ground** formulae

## Free variables + unification

**Systematic**

Can find “difficult” instances

Quite **expensive**

→ Very **nondeterministic**

## Combination!

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

Directed instantiation of formulae:

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

Directed instantiation of formulae:

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

---

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

Directed instantiation of formulae:

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

---

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

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)

$$\frac{\frac{\dots, p(X) \vdash}{\forall x.p(x), \forall x.(p(x) \rightarrow q(x) \vee r(x+1)), \forall x.\neg r(x) \vdash q(a)}}$$

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)

$$\frac{\overline{\dots, p(X) \vdash}}{\forall x.p(x), \forall x.(p(x) \rightarrow q(x) \vee r(x+1)), \forall x.\neg r(x) \vdash q(a)}$$

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)

$$\frac{\frac{q(X) \vee r(X+1) \vdash}{\dots, p(X) \vdash}}{\forall x.p(x), \forall x.(p(x) \rightarrow q(x) \vee r(x+1)), \forall x.\neg r(x) \vdash q(a)}$$

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)

$$\frac{
 \frac{
 \frac{q(X) \vdash}{\quad} \quad \frac{r(X+1) \vdash}{\quad}
 }{q(X) \vee r(X+1) \vdash}
 }{\dots, p(X) \vdash}
 }{\forall x.p(x), \forall x.(p(x) \rightarrow q(x) \vee r(x+1)), \forall x.\neg r(x) \vdash q(a)}$$

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)

$$\frac{
 \frac{
 \frac{q(X) \vdash}{\quad} \quad \frac{r(X+1) \vdash}{\quad}
 }{q(X) \vee r(X+1) \vdash}
 }{\dots, p(X) \vdash}
 }{\forall x.p(x), \forall x.(p(x) \rightarrow q(x) \vee r(x+1)), \forall x.\neg r(x) \vdash q(a)}$$

- Formulae with **negative literals**:

- Formulae **without negative literals**:

$$\frac{\frac{q(X) \vdash}{\frac{q(X) \vee r(X+1) \vdash}{\dots, p(X) \vdash}} \quad \frac{\frac{*}{false \vdash}}{r(X+1) \vdash}}{\forall x. (p(x) \rightarrow q(x) \vee r(x+1)), \forall x. \neg r(x) \vdash q(a)}$$

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

$$\frac{\frac{q(X) \vdash}{\frac{q(X) \vee r(X+1) \vdash}{\dots, p(X) \vdash}} \quad \frac{\frac{*}{false \vdash}}{r(X+1) \vdash}}{\forall x.p(x), \forall x.(p(x) \rightarrow q(x) \vee r(x+1)), \forall x.\neg r(x) \vdash q(a)}$$

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

15/25

## 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

Functions almost like in SMT:

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

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

Functions almost like in SMT:

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

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

- Axioms necessary: **Totality + Functionality**

$$\forall \bar{x}. \exists y. f_p(\bar{x}, y)$$

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

Functions almost like in SMT:

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

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

- 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**

**Two ways** to encode function applications:

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

**Two ways** to encode function applications:

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

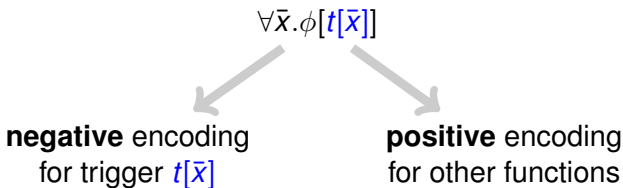
$\Rightarrow$  **Useful: PUHR** only matches on **negative** literals

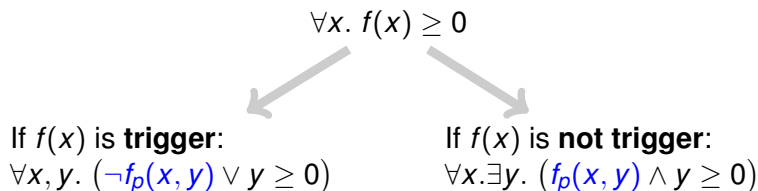
# E-Matching through PUHR

**Two ways** to encode function applications:

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

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





# The highlight: relative completeness

In **SMT solvers**:

- Choice of triggers determines **provability**
- Bad triggers → 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

	<b>AUFLIA+p</b> (193)	<b>AUFLIA-p</b> (193)
Z3	191	191
<b>PRINCESS</b>	<b>145</b>	<b>137</b>
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>

- **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!

- $\mathcal{ME}(\text{LIA})$ : 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

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