E-Matching and the Hypertableau Rule in the Theorem Prover Princess

Philipp Rümmer
philipp@chalmers.se

Wintermeeting of the SET Division
January 13th 2009

Outline

- The theorem prover Princess
- E-Matching
- Hypertableaux

- How to simulate e-matching using hypertableaux

1. The Princess theorem prover

Prover for first-order logic with linear integer arithmetic:

- Tailored to program verification
- Complete for first-order logic, Presburger arithmetic, etc.

- Classical sequent calculus, non-clausal
- No uninterpreted functions $\rightarrow$ relational encoding
- Free variables + unification + constraints

More information, implementation, paper:
`http://www.cse.chalmers.se/~philipp/princess/`

1. The Princess theorem prover

Prover for first-order logic with linear integer arithmetic:

- Tailored to program verification
- Complete for first-order logic, Presburger arithmetic, etc.

- Classical sequent calculus, non-clausal
- No uninterpreted functions $\rightarrow$ relational encoding
- Free variables + unification + constraints

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

More information, implementation, paper:
`http://www.cse.chalmers.se/~philipp/princess/`

2. E-Matching

Standard quantifier handling in SMT solvers:

- Matching of “triggers” (modulo equations):

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

- Triggers $t[\bar{x}]$ are often provided by user

Standard quantifier handling in SMT solvers:

- Matching of "triggers" (modulo equations):

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

- Triggers $t[\bar{x}]$ are often provided by user

```
\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))
```

Standard quantifier handling in SMT solvers:

- Matching of “triggers” (modulo equations):

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

- Triggers $t[\bar{x}]$ are often provided by user

```
\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
Quite cheap	Quite expensive $\rightarrow$ Very nondeterministic

Comparison

E-Matching	Free variables + unification
Heuristic $\longrightarrow$ incomplete	Systematic
Good for "simple" instances	Can find "difficult" instances
Quite cheap	Quite expensive $\longrightarrow$ Very nondeterministic

$\Rightarrow$ Combination?

3. Hypertableaux (aka “Model Generation”)

Derive models of clause sets by fixed-point iteration:

- Clauses without negative literals:
  $\Rightarrow$ Instantiate with free variables
- Clauses with negative literals:
  $\Rightarrow$ Discharge negative literals with unit resolution

3. Hypertableaux (aka "Model Generation")

Derive models of clause sets by fixed-point iteration:

- Clauses without negative literals:
  $\Rightarrow$ Instantiate with free variables
- Clauses with negative literals:
  $\Rightarrow$ Discharge negative literals with unit resolution

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

3. Hypertableaux (aka "Model Generation")

Derive models of clause sets by fixed-point iteration:

- Clauses without negative literals:
  $\Rightarrow$ Instantiate with free variables
- Clauses with negative literals:
  $\Rightarrow$ Discharge negative literals with unit resolution

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

3. Hypertableaux (aka “Model Generation”)

Derive models of clause sets by fixed-point iteration:

- Clauses without negative literals:
  $\Rightarrow$ Instantiate with free variables
- Clauses with negative literals:
  $\Rightarrow$ Discharge negative literals with unit resolution

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

3. Hypertableaux (aka “Model Generation”)

Derive models of clause sets by fixed-point iteration:

- Clauses without negative literals:
  $\Rightarrow$ Instantiate with free variables
- Clauses with negative literals:
  $\Rightarrow$ Discharge negative literals with unit resolution

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

3. Hypertableaux (aka “Model Generation”)

Derive models of clause sets by fixed-point iteration:

- Clauses without negative literals:
  $\Rightarrow$ Instantiate with free variables
- Clauses with negative literals:
  $\Rightarrow$ Discharge negative literals with unit resolution

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

3. Hypertableaux (aka “Model Generation”)

Derive models of clause sets by fixed-point iteration:

- Clauses without negative literals:
  $\Rightarrow$ Instantiate with free variables
- Clauses with negative literals:
  $\Rightarrow$ Discharge negative literals with unit resolution

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

3. Hypertableaux (aka "Model Generation")

Derive models of clause sets by fixed-point iteration:

- Clauses without negative literals:
  $\Rightarrow$ Instantiate with free variables
- Clauses with negative literals:
  $\Rightarrow$ Discharge negative literals with unit resolution

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

Derive models of clause sets by fixed-point iteration:

- Clauses without negative literals:
  $\Rightarrow$ Instantiate with free variables
- Clauses with negative literals:
  $\Rightarrow$ Discharge negative literals with unit resolution

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

3. Hypertableaux (aka "Model Generation")

Derive models of clause sets by fixed-point iteration:

- Clauses without negative literals:
  $\Rightarrow$ Instantiate with free variables
- Clauses with negative literals:
  $\Rightarrow$ Discharge negative literals with unit resolution

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

Completeness (Conjecture)

If $\Gamma\ \vdash\ \Delta$ is provable in the ordinary Princess calculus, then it is also provable with the Hypertableau rule.

Relational encoding of functions

n-ary function f becomes $(n+1)$-ary predicate f_p:

- Axioms: Totality + Functionality

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

- Two equivalent ways to encode function applications:

$$\begin{array}{lcll} \phi[f(\bar{t})] & \leadsto & \forall y.(f_p(\bar{t}, y) \rightarrow \phi[y]) & \text{(negative)} \\ & \leadsto & \exists y.(f_p(\bar{t}, y) \wedge \phi[y]) & \text{(positive)} \end{array}$$

- All function applications become literals

E-Matching using the Hypertableau rule

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

$\leadsto$ negative function encoding for $t[\bar{x}]$

$\leadsto$ positive encoding for other function appl.

- E-Matching (almost) like in SMT-solvers
- But: Choice of triggers has no effect on completeness!

E-Matching using the Hypertableau rule

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

$\leadsto$ negative function encoding for $t[\bar{x}]$

$\leadsto$ positive encoding for other function appl.

- E-Matching (almost) like in SMT-solvers
- But: Choice of triggers has no effect on completeness!

Example:

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

$$\forall x, y.\ \big(f_p(x, y) \rightarrow y \geq 0\big) \qquad \forall x.\exists y.\ \big(f_p(x, y) \wedge y \geq 0\big)$$

Conclusion

- Relational function encoding + hypertableau = E-Matching
- Implementation in progress ...
- E-Matching made respectable?

Future work, open questions:

- Formal completeness proof for Princess hypertableau rule
- When to use e-matching, when to use free variables?
- Relational encoding vs. native functions
- Partial functions vs. total functions

Thanks for your attention!