Understanding Resolution Proofs through Herbrand's Theorem

TBILLC 2013

Stefan Hetzl, Tomer Libal, Martin Riener, Mikheil Rukhaia

September 28, 2013

Introduction

A typical sequence for resolution theorem proving

- Formulate a problem as first-order logic formula
- Call a resolution prover: negate, skolemize, clausify, find refutation
- Result: Yes (+ proof object) / No / Timeout

Introduction

Problem

- Resolution Proofs are hard to read for humans
- Reason: Information is implicit

Approach

- Transform the resolution proof into sequent calculus
- Extract an Expansion Tree from the sequent calculus proof
- Interactive navigation (via Display Expansion Tree)

Example

Wolf, Goat & Cabbage Riddle

$$p(south, south, south, south, start),$$

$$\forall T.(p(south, north, south, north, T) \rightarrow p(north, north, south, north, go\_alone(T))),$$
$$\forall T1.(p(north, north, south, north, T1) \rightarrow p(south, north, south, north, go\_alone(T1))),$$
$$\forall T2.(p(south, south, north, south, T2) \rightarrow p(north, south, north, south, go\_alone(T2))),$$
$$\forall T3.(p(north, south, north, south, T3) \rightarrow p(south, south, north, south, go\_alone(T3))),$$

$$\forall T4.(p(south, south, south, north, T4) \rightarrow p(north, north, south, north, take\_wolf(T4))),$$
$$\forall T5.(p(north, north, south, north, T5) \rightarrow p(south, south, south, north, take\_wolf(T5))),$$
$$\forall T6.(p(south, south, north, south, T6) \rightarrow p(north, north, north, south, take\_wolf(T6))),$$
$$\forall T7.(p(north, north, north, south, T7) \rightarrow p(south, south, north, south, take\_wolf(T7))),$$

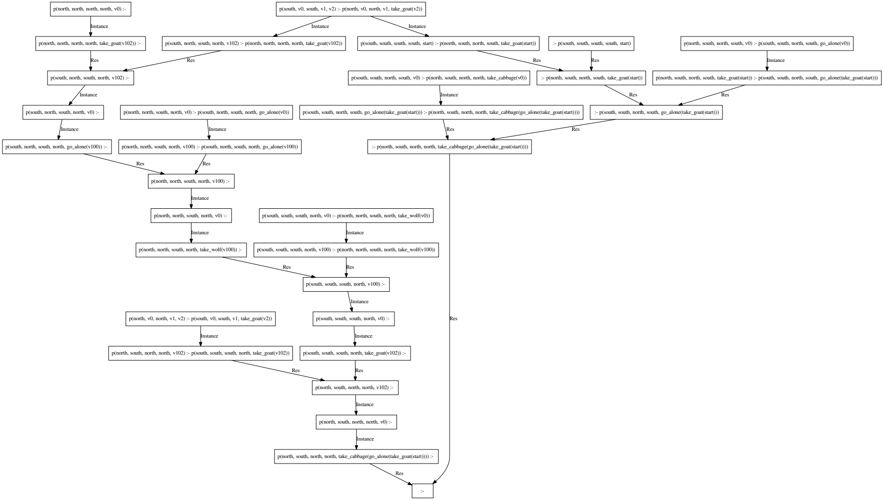
$$\forall X.\forall Y.\forall U.(p(south, X, south, Y, U) \rightarrow p(north, X, north, Y, take\_goat(U))),$$
$$\forall X1.\forall Y1.\forall V.(p(north, X1, north, Y1, V) \rightarrow p(south, X1, south, Y1, take\_goat(V))),$$

$$\forall T8.(p(south, north, south, south, T8) \rightarrow p(north, north, south, north, take\_cabbage(T8))),$$
$$\forall T9.(p(north, north, south, north, T9) \rightarrow p(south, north, south, south, take\_cabbage(T9))),$$
$$\forall U1.(p(south, south, north, south, U1) \rightarrow p(north, south, north, north, take\_cabbage(U1))),$$
$$\forall V1.(p(north, south, north, north, V1) \rightarrow p(south, south, north, south, take\_cabbage(V1)))$$

$$\vdash$$

$$\exists Z.p(north, north, north, north, Z)$$

Example: Resolution Refutation



Example: Instantiation Terms needed for Solution

Extracted Substitutions

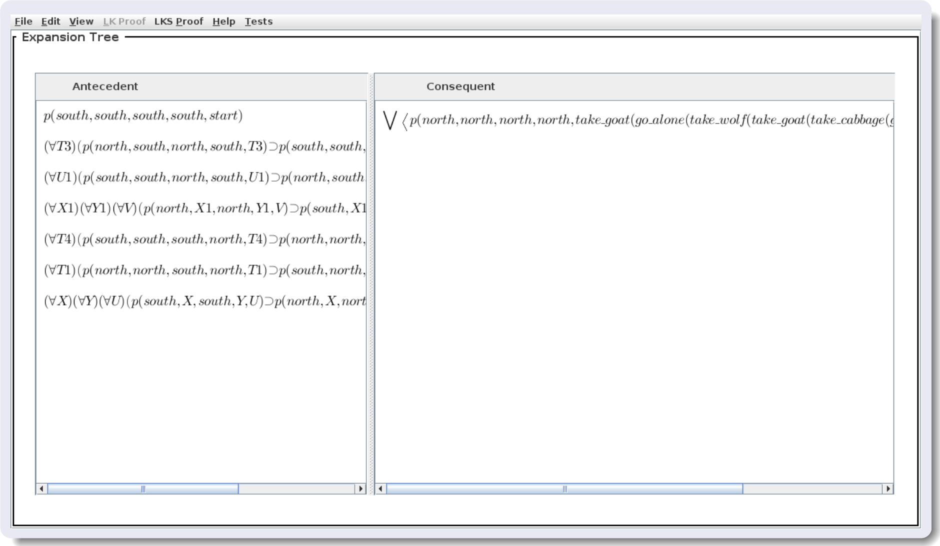
$\sigma_1 = \{\text{v0} \leftarrow \text{go\_alone(take\_goat(start))}\}$
$\sigma_2 = \{\text{v0} \leftarrow \text{go\_alone(v100)}\}$
$\sigma_3 = \{\text{v0} \leftarrow \text{take\_cabbage(go\_alone(take\_goat(start)))}\}$
$\sigma_4 = \{\text{v0} \leftarrow \text{take\_goat(start)}\}$
$\sigma_5 = \{\text{v0} \leftarrow \text{take\_goat(v102)}\}$
$\sigma_6 = \{\text{v0} \leftarrow \text{take\_wolf(v100)}\}$
$\sigma_7 = \{\text{v0} \leftarrow \text{v100}\}$
$\sigma_8 = \{\text{v100} \leftarrow \text{v0}\}$
$\sigma_9 = \{\text{v102} \leftarrow \text{v0}\}$
$\sigma_{10} = \{\text{v1} \leftarrow \text{north}, \text{v0} \leftarrow \text{north}, \text{v2} \leftarrow \text{v102}\}$
$\sigma_{11} = \{\text{v1} \leftarrow \text{north}, \text{v0} \leftarrow \text{south}, \text{v2} \leftarrow \text{v102}\}$
$\sigma_{12} = \{\text{v1} \leftarrow \text{south}, \text{v0} \leftarrow \text{south}, \text{v2} \leftarrow \text{start}\}$

Solution ...

$$\sigma = \{Z \leftarrow take\_goat(go\_alone(take\_wolf(take\_goat(take\_cabbage(go\_alone(take\_goat(start)))))))\}$$

... not easily extracted.

Example: Expansion Tree in Prooftool



Resolution to Sequent Calculus Transformation

Clause Set

$$\left\{ \begin{array}{l} \vdash F_1, F_2; \\ F_1 \vdash; \\ F_2, F_2 \vdash \end{array} \right\}$$

Ground Refutation in Sequent Notation

$$\dfrac{\dfrac{\vdash F_1, F_2 \qquad F_1 \vdash}{\vdash F_2}\,res \qquad \dfrac{F_2, F_2 \vdash}{F_2 \vdash}\,factor}{\vdash}\,res$$

CNF Transformation

CNF Projections

- Let $CNF(\Gamma \vdash \Delta) = \{\Pi_1 \vdash \Lambda_1; \ldots; \Pi_n \vdash \Lambda_n\}$
- If $\Gamma \vdash \Delta$ does not contain strong quantifiers, then a proof $\Gamma, \Pi_i \vdash \Lambda_i, \Delta$ is constructed for each clause i:

$$L_1 \vdash L_1 \qquad\qquad L_n \vdash L_n$$

$$\vdots$$

(CNF Transformation)

$$\vdots$$

$$\Gamma, \Pi_i \vdash \Lambda_i, \Delta$$

From formulas

End-Sequent without strong Quantifiers

$$\dfrac{\dfrac{\dfrac{\dfrac{F_1 \vdash F_1 \qquad F_2 \vdash F_2}{\vdots}}{\text{(CNF Transformation)}}}{\Gamma \vdash F_1, F_2, \Delta} \qquad \dfrac{\dfrac{\dfrac{F_1 \vdash F_1}{\vdots}}{\text{(CNF Transformation)}}}{\Gamma, F_1 \vdash \Delta}}{\Gamma \vdash F_2, \Delta}\,cut \qquad \dfrac{\dfrac{\dfrac{\dfrac{F_2, F_2 \vdash F_2, F_2}{\vdots}}{\text{(CNF Transformation)}}}{\Gamma, F_2, F_2 \vdash \Delta}}{\Gamma, F_2 \vdash \Delta}\,c:r$$

$$\dfrac{\Gamma \vdash F_2, \Delta \qquad \Gamma, F_2 \vdash \Delta}{\Gamma \vdash \Delta}\,res$$

$\Gamma \vdash \Delta$: arbitrary sequent without strong quantifiers

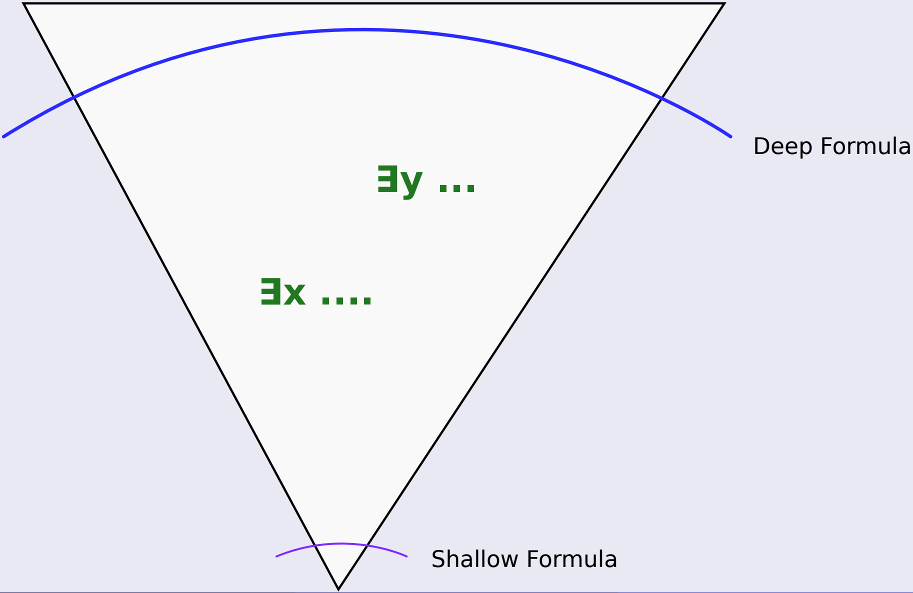
Expansion Trees

Expansion Trees

- Introduced for HOL by D. Miller
- Generalization of Herbrand Disjunction:
  - Formula written as tree with logical operators as nodes
  - Children of Weak Quantifiers have one child per instantiation needed
  - Shallow Formula: Original formula containing weak quantifiers
  - Deep Formula: Weak quantifiers are replaced by a disjunction of instances

    i.e. common parts of formulas in Herbrand Disjunction are merged
- Expansion Proof:
  - Deep Formula is a tautology
  - Straightforward extraction from Sequent Calculus proofs without quantified cuts

Expansion Tree

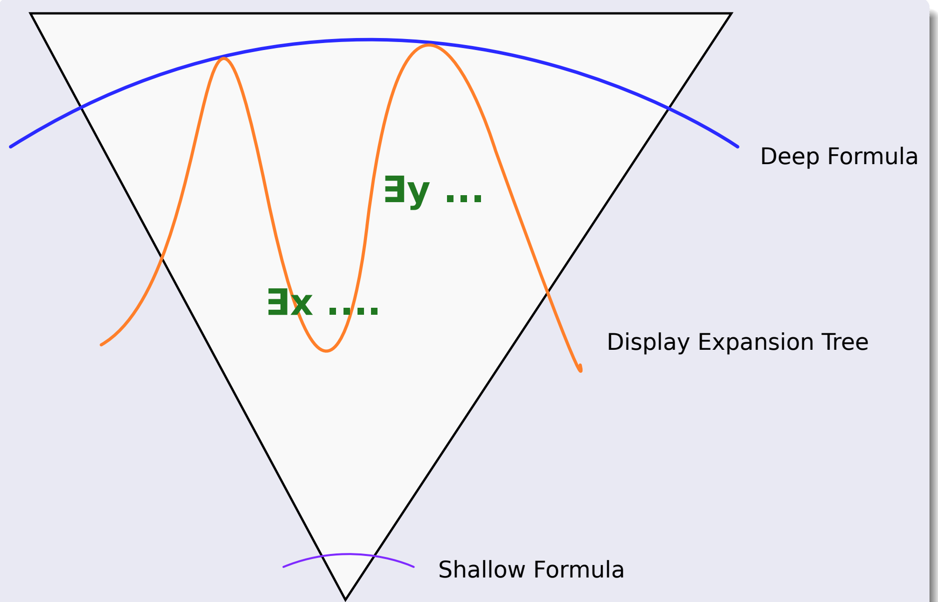


Display Expansion Tree

Display Expansion Tree

- Modified Expansion Tree for interactive exploration
- Weak Quantifier Nodes have three states:
  - Closed: $\exists x F[x]$
  - Open: $\exists x < t_1, \ldots, t_n > F[x]$
  - Expanded: $\bigvee \left\langle \begin{array}{c} F(t_1) \\ \ldots \\ F(t_n) \end{array} \right\rangle$
- Open/expanded nodes require their ancestor nodes to be open/expanded

Display Expansion Tree



Demo

(Change to Prooftool)

Conclusion & Future Work

Conclusion

- Display Expansion Trees show information not visible in Resolution refutation
- Interactive focus on instances relevant to the user
- Implementation at `http://www.logic.at/gapt`

Future Work

- Calculate instances on node expansion
- Extend to higher-order Refutations

The End

Thanks for your attention!