

DTU



Verifying a Sequent Calculus Prover

Asta Halkjær From Frederik Krogsdal Jacobsen

Introduction

- A sound and complete prover for first-order logic with functions
- Based on a sequent calculus
- All proofs are formally verified in Isabelle/HOL
- Human-readable proof certificates

Background

- Formalized metatheory for non-trivial sequent calculus provers
- Formal verification of an executable prover
- Novel analytic proof technique for completeness
- Verifiable and human-readable proof certificates
- A prover for the SeCaV system

Sample SeCaV Proof Rules

$$\frac{\text{Neg } p \in z}{\Vdash p, z} \text{ BASIC}$$

$$\frac{\Vdash z \quad z \subseteq y}{\Vdash y} \text{ EXT}$$

$$\frac{\Vdash p, z}{\Vdash \text{Neg} (\text{Neg } p), z} \text{ NEGNEG}$$

$$\frac{\Vdash p, q, z}{\Vdash \text{Dis } p q, z} \text{ ALPHADIS}$$

$$\frac{\Vdash \text{Neg } p, z \quad \Vdash \text{Neg } q, z}{\Vdash \text{Neg} (\text{Dis } p q), z} \text{ BETADIS}$$

$$\frac{\Vdash p [\text{Var } 0/t], z}{\Vdash \text{Exi } p, z} \text{ GAMMAEXI}$$

$$\frac{\Vdash \text{Neg} (p [\text{Var } 0/\text{Fun } i []]), z \quad i \text{ fresh}}{\Vdash \text{Neg} (\text{Exi } p), z} \text{ DELTAEXI}$$

Prover

- SeCaV rules affect *one formula* at a time
- We affect *every applicable formula* at once
- Rules affect disjoint formulas
- By applying rules *fairly*, we never miss out on a proof
- Proof attempts are *coinductive trees* grown by applying rules
- If a tree cannot be grown further, we found a proof
- An *effect function* encodes the rules
- We export code to Haskell to obtain an executable prover

Soundness I

- If the children of a sequent all have SeCaV proofs, so does the sequent
- Framework: finite, well-formed proof trees represent SeCaV proofs
- The SeCaV proof system is sound
- ...so the prover is sound

Soundness II

- 1 Assume all child sequents have a proof
- 2 Induction on sequent:
 - 1 Case analysis on first formula in sequent
 - 2 Prove that the sequent is valid using appropriate SeCaV rules

Example:

$$\frac{\begin{array}{c} \vdots \\ \hline \text{ALPHADIS} \\ \text{⊢ Dis } P, Q, \dots \end{array}}{\vdots}$$

P, Q, \dots is a child sequent, so we can apply the ALPHADIS rule to prove the sequent using the proof of $\text{⊢ } P, Q, \dots$ (and possibly some reordering).

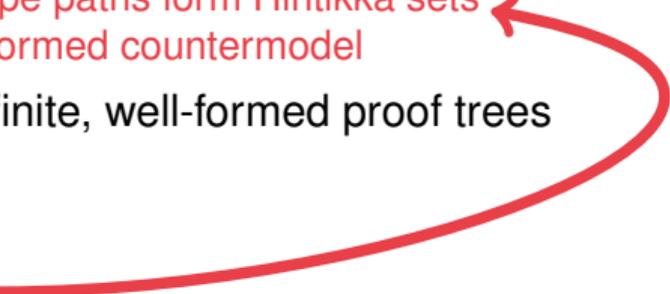
Completeness

- Framework: prover either produces a finite, well-formed proof tree or an infinite tree with a saturated escape path
- The root sequent of a saturated escape path is not valid:
 - Formulas on saturated escape paths form Hintikka sets
 - Hintikka sets induce a well-formed countermodel
- ... so valid sequents result in finite, well-formed proof trees

Completeness

- Framework: prover either produces a finite, well-formed proof tree or an infinite tree with a saturated escape path
- The root sequent of a saturated escape path is not valid:
 - Formulas on saturated escape paths form Hintikka sets
 - Hintikka sets induce a well-formed countermodel
- ... so valid sequents result in finite, well-formed proof trees

HERE BE DRAGONS



(need to control function terms since we only consider terms in the sequent)

Results and future work

- Verified soundness and completeness in Isabelle/HOL
- Verification helped find actual bugs in our implementation
- Very limited performance, but optimizations are possible
- Generation of proof certificates is not verified
- Extensions to the logic