Fast cut-elimination using proof terms: an empirical study

Gabriel Ebner

CL&C'18
2018-07-07

TU Wien

Theorem (special case of Herbrand 1930)

Let $\varphi(x)$ be a quantifier-free first-order formula.

Then $\exists x\, \varphi(x)$ is valid iff there exist terms $t_1, \ldots, t_n$ such that $\varphi(t_1) \vee \cdots \vee \varphi(t_n)$ is a tautology.

Theorem (Miller 1987)

Let φ be a higher-order formula.

Then φ is a theorem of elementary type theory iff there exists an expansion tree E such that $\mathrm{dp}(E)$ is a tautology and $\mathrm{sh}(E) = \varphi$.

Obtaining Herbrand disjunctions

- We can directly extract Herbrand disjunctions from cut-free proofs
- Even from proofs with quantifier-free cuts
  $\rightarrow$ do not require full cut-elimination

Uses of Herbrand’s theorem

- Computational interpretation of proofs
- Luckhardt’s proof of Roth’s theorem (1989)
- Equality of proofs
- Proof complexity

Computational proof theory

- GAPT: General Architecture for Proof Theory
  - open source, written in Scala
  - `https://github.com/gapt/gapt`
- many algorithms based on Herbrand disjunctions
  - lemma generation (cut-introduction)
  - automated inductive theorem proving
  - proof deskolemization

$\rightarrow$ need fast & reliable cut-elimination

Wishlist

- calculus close to LK
  - most of our proofs are in LK
- needs to support nonstandard inference rules
  - Skolemization
  - schematic proofs with cycles
  - ...
- higher-order logic
- induction rule
- equational reasoning

Solution

- Term calculus from Urban, Bierman 2001
    - direct term assignment for LK
- slightly extended:
    - higher-order logic
    - induction
    - equality
- fast big-step normalization

Logic

- Elementary type theory
  - no extensionality or choice built-in
- Structural induction for some base types

- Formulas in simply typed lambda calculus
  - $\wedge^{o \to o \to o}$
  - $\forall_\alpha^{(\alpha \to o) \to o}$
  - $\ldots$

Example

$$\dfrac{\dfrac{py \vdash py}{\vdash py \to py}}{\vdash \forall x\,(px \to px)}$$

Example

$$\dfrac{\dfrac{\mathrm{Ax}(h_3, h_4) \;\; ::_{[x\backslash y]} \;\; h_3\colon py \vdash \qquad\qquad h_4\colon py}{\mathrm{AndL}(h_2, h_3\colon h_4\colon \mathrm{Ax}(h_3, h_4)) \;\; ::_{[x\backslash y]} \;\; \vdash \qquad\qquad h_2\colon py \to py}}{\mathrm{AllR}(h_1, h_2\colon x\colon \mathrm{AndL}(h_2, h_3\colon h_4\colon \mathrm{Ax}(h_3, h_4))) \;\; ::_{[]} \;\; \vdash h_1\colon \forall x\,(px \to px)}$$

Example

$$\dfrac{\dfrac{\mathrm{Ax}(h_3, h_4) \;\; ::_{[x \backslash y]} \;\; h_3 \colon py \vdash h_1 \colon \forall x\, (px \to px), h_2 \colon py \to py, h_4 \colon py}{\mathrm{AndL}(h_2, h_3 \colon h_4 \colon \mathrm{Ax}(h_3, h_4)) \;\; ::_{[x \backslash y]} \;\; \vdash h_1 \colon \forall x\, (px \to px), h_2 \colon py \to py}}{\mathrm{AllR}(h_1, h_2 \colon x \colon \mathrm{AndL}(h_2, h_3 \colon h_4 \colon \mathrm{Ax}(h_3, h_4))) \;\; ::_{[]} \;\; \vdash h_1 \colon \forall x\, (px \to px)}$$

Example

$$\dfrac{\dfrac{\mathrm{Ax}(h_3, h_4) \ ::_{[x \backslash y]} \ h_3\colon py \vdash h_1\colon \forall x\,(px \to px), h_2\colon py \to py, h_4\colon py}{\mathrm{AndL}(h_2, h_3\colon h_4\colon \mathrm{Ax}(h_3, h_4)) \ ::_{[x \backslash y]} \ \vdash h_1\colon \forall x\,(px \to px), h_2\colon py \to py}}{\mathrm{AllR}(h_1, h_2\colon x\colon \mathrm{AndL}(h_2, h_3\colon h_4\colon \mathrm{Ax}(h_3, h_4))) \ ::_{[]} \ \vdash h_1\colon \forall x\,(px \to px)}$$

- Weakening and contraction are implicit

Evaluation

Three (partial?) functions:

- $\mathcal{N}(\pi)$ returns a normal form of π
  - $\rightarrow$ result of normalization
- $\mathcal{E}(\varphi, h_1 \colon \pi_1, h_2 \colon \pi_2)$ where π_1 and π_2 are normalized
  - $\rightarrow$ reduction of a top-most cut
- $\mathcal{S}(\pi_1, \varphi, h_1 := h_2 \colon \pi_2)$ where π_1 and π_2 are normalized
  - $\rightarrow$ full rank-reduction
    - "proof substitution" in Urban, Bierman 2001

All preserve typing and return normal forms.

Evaluation

$$\mathcal{N}(\mathrm{NegL}(h_1, h_2 \colon \pi)) = \mathrm{NegL}^?(h_1, h_2 \colon \mathcal{N}(\pi))$$

$$\vdots$$

$$\mathcal{E}(\neg\varphi, h_1 \colon \mathrm{NegR}(h_1, h_2 \colon \pi_1), h_3 \colon \mathrm{NegL}(h_3, h_4 \colon \pi_2)) = \mathcal{E}(\varphi, h_4 \colon \pi_2', h_2 \colon \pi_1')$$

$$\vdots$$

$$\mathcal{S}(\mathrm{Ax}(h_1, h_2), \varphi, h_1 := h_3 \colon \pi) = \pi[h_3 \backslash h_2]$$

$$\mathcal{S}(\mathrm{NegL}(h_1, h_2 \colon \pi_1), \varphi, h_3 := h_4 \colon \pi_2) = \mathrm{NegL}^?(h_1, h_2 \colon \mathcal{S}(\pi_1, \varphi, h_3 := h_4 \colon \pi_2))$$

$$\vdots$$

Canonicity

Theorem

Let $\pi \ ::_\sigma \ \Gamma \vdash \Delta$ *such that* $\mathcal{N}(\pi)\downarrow$.

If π *does not contain* Rfl, Eql, *or* Ind **, then* $\mathcal{N}(\pi)$ *is cut-free.*

If π *does not contain* Ind **, and* Eql *only rewrites atoms, then* $\mathcal{N}(\pi)$ *has at most atomic cuts.*

* or definition, Skolem, link inferences

Induction-elimination

- Typically consider proofs of e.g.:

$$\begin{aligned} &\forall x\, x + 0 = x, \\ &\forall x \forall y\, x + s(y) = s(x + y) \\ &\vdash \forall x (0 + x = x) \end{aligned}$$

Induction-elimination

- Want cut-free proof of:

$$\begin{aligned} &\forall x\, x + 0 = x, \\ &\forall x \forall y\, x + s(y) = s(x + y) \\ &\vdash 0 + s^n(0) = s^n(0) \end{aligned}$$

Induction unfolding

- unfold induction inferences on constructors to cuts

$$\mathrm{Ind}(h_1, \varphi, 0, h_2\colon \pi_1, x\colon h_3\colon h_4\colon \pi_2) \ \mapsto \ \pi_1[h_2 \backslash h_1]$$

$$\mathrm{Ind}(h_1, \varphi, s(t), h_2\colon \pi_1, x\colon h_3\colon h_4\colon \pi_2) \ \mapsto$$
$$\mathrm{Cut}(\varphi(t), h_1\colon \mathrm{Ind}(h_1, \varphi, t, h_2\colon \pi_1, x\colon h_3\colon h_4\colon \pi_2), h_3\colon \pi_2[x \backslash t][h_4 \backslash h_1])$$

- alternate between cut-elimination and induction unfolding

Termination

- Interesting question
  - conjecture termination for full calculus

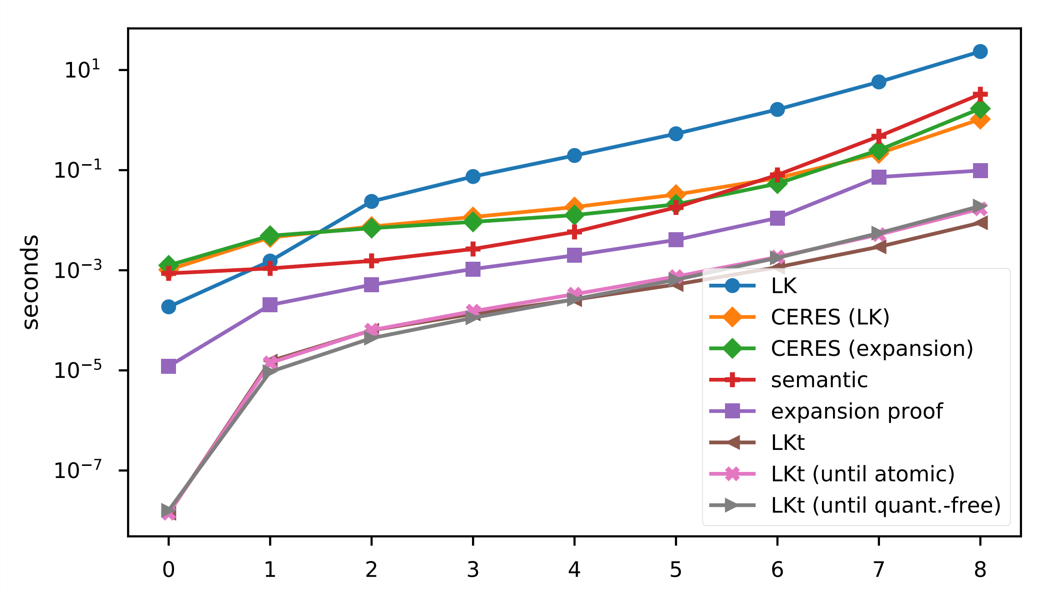
- Not so important for our applications

- First-order fragment terminates by induction on ω^2
- Urban and Bierman showed strong normalization for first-order fragment (w/o equality)
  - subtle difference: $\mathrm{NegR}^?$

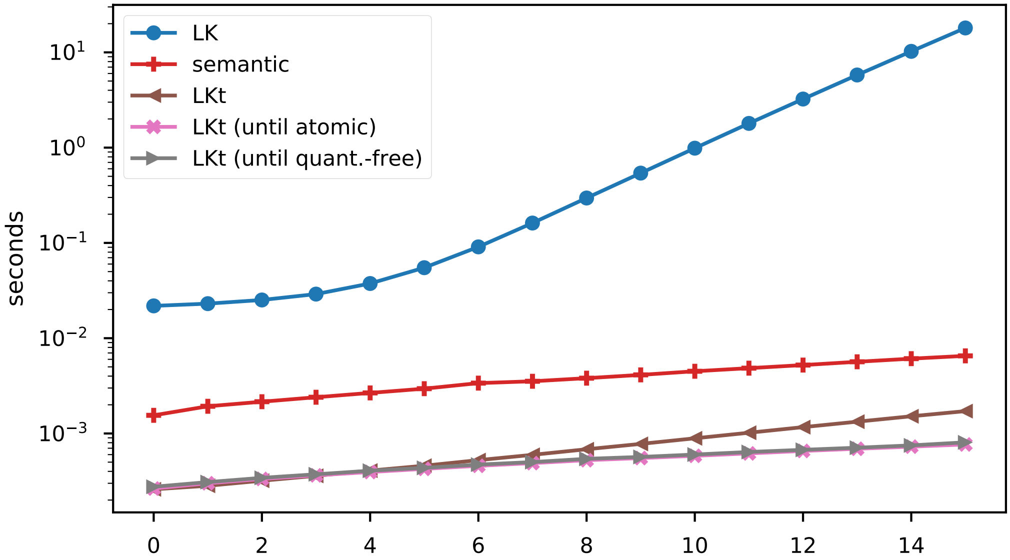
Benchmarked methods

- LK: existing Gentzen-style cut-elimination
- CERES: cut-elimination by resolution
    - also expansion proof optimization (Leitsch, Lolic 2018)
- Semantic cut-elimination
    - Forget proof, run automated theorem prover instead
- Expansion proof cut-elimination
    - like second ε-theorem, operates only on quantifier instances
- LK_t: this method

$P(0), \forall x(P(x) \to P(s(x)) \vdash P(s^{2^n}(0))$



$\forall x(x + 0 = x), \forall x \forall y(x + s(y) = s(x + y)) \vdash \forall x(0 + x = x)$



Proof of the infinitude of primes

Theorem

There are infinitely many primes.

Proof (Furstenberg 1955).

Equip $\mathbb{Z}$ with the topology generated by the arithmetic progressions... $\square$

- What is the combinatorial essence of this proof?

Proof analysis of Furstenberg's proof

$\rightarrow$ second-order proof of $\exists q\ (\mathrm{prime}(q) \land q \notin \{p(0), \ldots, p(n-1)\})$ for $n \in \mathbb{N}$

- use cut-elimination to compute witness q

- Similar approach used by Girard 1987
  to analyze proof of van der Waerden's theorem by Furstenberg 1981

Analysis with CERES

- Previous analysis using CERES (cut-elimination by resolution) (Baaz, Hetzl, Leitsch, Richter, Spohr 2008)

- Requires resolution refutation of *characteristic clause set*
  - Automated theorem provers only managed $n = 0$

  $\rightarrow$ Manual specification of resolution proofs for $n \geq 0$

$\rightarrow$ prime divisor of $1 + p(0) * \cdots * p(n)$ as witness

  - another refutation for $n = 2$ yields a prime divisor of $p(0) + 1, p(1) + 1$, or 5

Witness obtained with $\mathrm{LK_t}$

$$\texttt{primediv\_of}(1 + 2 * p(0) * \cdots * p(n))$$

- computable in reasonable time for $n < 10$
  - (with a bit of post-processing)
- small changes in proof have big effect on witness
  - can also get factor 3

Demonstration

Conclusion

- Term assignments are an efficient implementation technique for proof transformation and analysis
  - 10^4x speedup with only small changes to the calculus

$\rightarrow$ compare with other low-bureaucracy approaches
  - functional interpretation
  - tree grammars

Backup slides

Syntax

$$
\begin{aligned}
\mathit{Hyp} &::= -\mathbb{N}^+ \mid +\mathbb{N}^+ \\
\mathit{Term} &::= \mathrm{Ax}(\mathit{Hyp}, \mathit{Hyp}) \mid \mathrm{TopR}(\mathit{Hyp}) \\
&\mid \mathrm{Cut}(\mathit{Formula}, \mathit{Hyp}\colon \mathit{Term}, \mathit{Hyp}\colon \mathit{Term}) \\
&\mid \mathrm{NegL}(\mathit{Hyp}, \mathit{Hyp}\colon \mathit{Term}) \mid \mathrm{NegR}(\mathit{Hyp}, \mathit{Hyp}\colon \mathit{Term}) \\
&\mid \mathrm{AndL}(\mathit{Hyp}, \mathit{Hyp}\colon \mathit{Hyp}\colon \mathit{Term}) \mid \mathrm{AndR}(\mathit{Hyp}, \mathit{Hyp}\colon \mathit{Term}, \mathit{Hyp}\colon \mathit{Term}) \\
&\mid \mathrm{AllL}(\mathit{Hyp}, \mathit{Expr}, \mathit{Hyp}\colon \mathit{Term}) \mid \mathrm{AllR}(\mathit{Hyp}, \mathit{Var}\colon \mathit{Hyp}\colon \mathit{Term}) \\
&\mid \mathrm{Rfl}(\mathit{Hyp}) \mid \mathrm{Eql}(\mathit{Hyp}, \mathit{Hyp}, \mathit{Bool}, \mathit{Expr}, \mathit{Hyp}\colon \mathit{Term}) \\
&\mid \mathrm{Ind}(\mathit{Hyp}, \mathit{Expr}, \mathit{Expr}, \mathit{Hyp}\colon \mathit{Term}, \mathit{Var}\colon \mathit{Hyp}\colon \mathit{Hyp}\colon \mathit{Term})
\end{aligned}
$$

Other inference rules

- Proof links $\rightarrow$ schematic proofs with cycles

$$\frac{t \text{ is a name for a proof of } \varphi_1, \dots \vdash \dots, \varphi_n}{\mathrm{Link}(t, [h_1, \dots, h_n]) \ ::_\sigma \ h_1 \colon \varphi_1, \dots, \Gamma \vdash \Delta, \dots, h_n \colon \varphi_n}$$

- Definition rules

$$\frac{\pi \ ::_\sigma \ \Gamma \vdash \Delta, h_1 \colon \varphi, h_2 \colon \varphi'}{\mathrm{Def}(h_1, \varphi', h_2 \colon \pi) \ ::_\sigma \ \Gamma \vdash \Delta, h_1 \colon \varphi}$$

- Skolem rules $\rightarrow$ Skolemized cut-free proofs in higher-order logic

$$\frac{\pi \ ::_\sigma \ \Gamma \vdash \Delta, h_1 \colon \forall x\, \varphi(x), h_2 \colon \varphi(t)}{\mathrm{AllSk}(h_1, x, h_2 \colon \pi) \ ::_\sigma \ \Gamma \vdash \Delta, h_1 \colon \forall x\, \varphi(x)} \ (t \text{ is Skolem term for } \forall x\, \varphi(x))$$

Object language

- Higher-order logic (simply-typed lambda calculus)
- Types:
  - Booleans: o
  - other base types: $a, b, c, \ldots$
  - function type: $\alpha \to \beta$
- Terms:
  - constants: c^τ
  - variables: x^τ
  - application: $t\,s$
  - abstraction: $\lambda x^\tau\ t$

Implementation

- in GAPT
- named variables as binding strategy
- cache set of free variables in each term
  $\rightarrow$ can effectively skip many branches