Introduction

Constructivization rewrite rules

Combining rewrite systems

Experimental results

Conclusion

A Rewrite System for Proof Constructivization

Raphaël Cauderlier

CNAM, Inria, LSV

LFMTP
June 23. 2016

Discharging proofs to a theorem prover

- Theorem provers are classical
- Proof assistants for type theory are constructive
- Assume the classical axiom $\longrightarrow$ no proof-as-program
- Restrict to the image of a $\neg\neg$translation $\longrightarrow$ too small

In generated proofs, classical axioms are always used but not always necessary.

Introduction
Constructivization rewrite rules
Combining rewrite systems
Experimental results
Conclusion

Introduction

Constructivization rewrite rules

Combining rewrite systems

Experimental results

Conclusion

Constructivization Problem

Given a formula φ and a classical proof π of φ, is φ constructively provable?

Constructivization Problem

Given a formula φ and a classical proof π of φ, is φ constructively provable?

- As hard as constructive provability ($\varphi := \psi \lor P \lor \neg P$)

Constructivization Problem

Given a formula φ and a classical proof π of φ, is φ constructively provable?

- As hard as constructive provability ($\varphi := \psi \vee P \vee \neg P$)
- Heuristic approach

$\lambda\Pi$-calculus modulo

Extends $\lambda\Pi$-calculus with rewriting:

- Signature Σ contains (well-typed) rewrite rules $\mathcal{R}$
- Conversion is extended

$$\frac{\Sigma;\Gamma \vdash t : A \qquad \Sigma;\Gamma \vdash B : \textbf{Type} \qquad (\text{when } A \equiv_{\mathcal{R}\beta} B)}{\Sigma;\Gamma \vdash t : B}$$

Dedukti: a logical framework based on rewriting

- Universal proof checker
  - interactive and automatic provers
- Purely functional programming language based on rewriting

Proofs are objects, we can compute on them beyond cut elimination.

Syntax of First-Order Logic

$$
\begin{array}{rcl}
A & ::= & \top \\
 & | & \bot \\
 & | & P \\
 & | & A \wedge A \\
 & | & A \vee A \\
 & | & A \Rightarrow A \\
 & | & \forall x. A(x) \\
 & | & \exists x. A(x)
\end{array}
$$

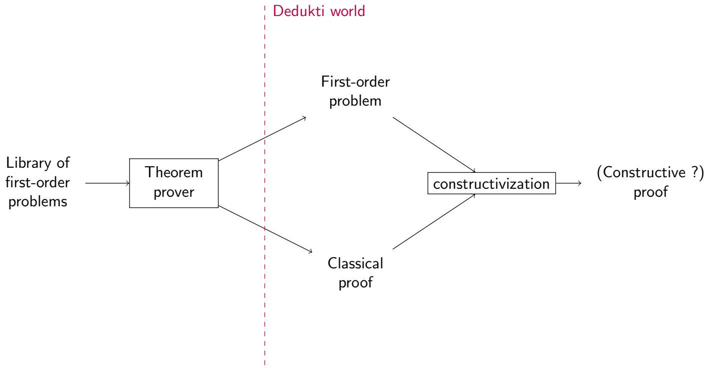
$$\neg A := A \Rightarrow \bot$$

Classical axioms

Classical logic = Intuitionistic logic + any one of those:

- Law of Double Negation: $\underline{\text{dn}}_A : \neg\neg A \Rightarrow A$
- Law of Excluded Middle: $\underline{\text{em}}_A : A \vee \neg A$
- ...

Proof constructivization



A rewrite system for A $\vee$ $\neg$ A

The following statements are constructive theorems:

- $t_\top : \underline{\text{EM}}_\top$
- $t_\bot : \underline{\text{EM}}_\bot$
- $t_\wedge : (\underline{\text{EM}}_A \wedge \underline{\text{EM}}_B) \Rightarrow \underline{\text{EM}}_{A \wedge B}$
- $t_\vee : (\underline{\text{EM}}_A \wedge \underline{\text{EM}}_B) \Rightarrow \underline{\text{EM}}_{A \vee B}$
- $t_\Rightarrow : (\underline{\text{EM}}_A \wedge \underline{\text{EM}}_B) \Rightarrow \underline{\text{EM}}_{A \Rightarrow B}$

where $\underline{\text{EM}}_A := A \vee \neg A$
which leads to the following partial definition of $\underline{\text{em}}$:

A rewrite system for A ∨ ¬ A

$\underline{\text{EM}}_A := A \vee \neg A$
$\underline{\text{em}}_A : \underline{\text{EM}}_A$

- $\underline{\text{em}}_\top \hookrightarrow t_\top$
- $\underline{\text{em}}_\bot \hookrightarrow t_\bot$
- $\underline{\text{em}}_{A \wedge B} \hookrightarrow t_\wedge(\underline{\text{em}}_A, \underline{\text{em}}_B)$
- $\underline{\text{em}}_{A \vee B} \hookrightarrow t_\vee(\underline{\text{em}}_A, \underline{\text{em}}_B)$
- $\underline{\text{em}}_{A \Rightarrow B} \hookrightarrow t_\Rightarrow(\underline{\text{em}}_A, \underline{\text{em}}_B)$

A rewrite system for $A \vee \neg A$

- Nothing smart to do with quantifiers and atoms
- In practice, never yields a constructive proof
- We do not inspect the proof

Example: A ⇒ A

$$\underline{\mathrm{dn}}_{A\Rightarrow A} \\ (\lambda H_{\neg(A\Rightarrow A)}.H_{\neg(A\Rightarrow A)} \\ (\lambda H_A\,.E_{\perp}^{A}((\lambda H'_A.H_{\neg(A\Rightarrow A)}(\lambda H''_A.H'_A))H_A\,)))$$

Example: A ⇒ A

$$\underline{\mathrm{dn}}_{A\Rightarrow A}$$
$$(\lambda H_{\neg(A\Rightarrow A)}.H_{\neg(A\Rightarrow A)}$$
$$(\lambda H_A.E_{\perp}^{A}((\lambda H'_A.H_{\neg(A\Rightarrow A)}(\lambda H''_A.H'_A))H_A)))$$

$$\underline{\mathrm{dn}}_{A\Rightarrow B} \hookrightarrow d_{\Rightarrow}(\underline{\mathrm{dn}}_A, \underline{\mathrm{dn}}_B)$$

Example: $A \Rightarrow A$

$$
\begin{array}{l}
\lambda H_A.\underline{\mathsf{dn}}_A(\lambda H_{\neg A}. \\
\quad (\lambda H_{\neg(A\Rightarrow A)}.H_{\neg(A\Rightarrow A)} \\
\qquad (\lambda H'''_A.E^A_\bot((\lambda H'_A.H_{\neg(A\Rightarrow A)}(\lambda H''_A.H'_A))H'''_A))) \\
\quad (\lambda H_{A\Rightarrow A}.H_{\neg A}(H_{A\Rightarrow A}\ H_A))
\end{array}
$$

Example: $A \Rightarrow A$

$$\begin{array}{l}\lambda H_A.\underline{\mathrm{dn}}_A(\lambda H_{\neg A}.\\ \quad (\lambda H_{\neg(A\Rightarrow A)}.H_{\neg(A\Rightarrow A)}\\ \qquad (\lambda H'''_A.E_\bot^A((\lambda H'_A.H_{\neg(A\Rightarrow A)}(\lambda H''_A.H'_A))H'''_A)))\\ \quad (\lambda H_{A\Rightarrow A}.H_{\neg A}(H_{A\Rightarrow A}\ H_A))\end{array}$$

$$E_\bot^A(\pi_{\neg A}\ \pi_A) \hookrightarrow \pi_A$$

Example: A $\Rightarrow$ A

$$
\begin{array}{l}
\lambda H_A.\underline{\mathsf{dn}}_A(\lambda H_{\neg A}. \\
\quad (\lambda H_{\neg(A\Rightarrow A)}.H_{\neg(A\Rightarrow A)} \\
\qquad (\lambda H'''_A.H'''_A)) \\
\quad (\lambda H_{A\Rightarrow A}.H_{\neg A}(H_{A\Rightarrow A}\ H_A))
\end{array}
$$

Example: A ⇒ A

$$\lambda H_A.\underline{\text{dn}}_A(\lambda H_{\neg A}.\\ (\lambda H_{\neg(A\Rightarrow A)}.H_{\neg(A\Rightarrow A)}\\ (\lambda H'''_A.H'''_A))\\ (\lambda H_{A\Rightarrow A}.H_{\neg A}(H_{A\Rightarrow A}\ H_A))$$

$$(\lambda H_A.\pi_B(H_A))\ \pi_A \hookrightarrow \pi_B(\pi_A)$$

Example: $A \Rightarrow A$

$$\lambda H_A.\underline{\mathsf{dn}}_A(\lambda H_{\neg A}.\\ (\lambda H_{A\Rightarrow A}.H_{\neg A}(H_{A\Rightarrow A}\ H_A))\\ (\lambda H'''_A.H'''_A))$$

Example: A $\Rightarrow$ A

$$\lambda H_A.\underline{\mathrm{dn}}_A(\lambda H_{\neg A}. \\ (\lambda H_{A\Rightarrow A}.H_{\neg A}(H_{A\Rightarrow A}\ H_A)) \\ (\lambda H'''_A.H'''_A))$$

$$(\lambda H_A.\pi_B(H_A))\ \pi_A \hookrightarrow \pi_B(\pi_A)$$

Example: $A \Rightarrow A$

$$\lambda H_A . \underline{\mathsf{dn}}_A (\lambda H_{\neg A} . \\ H_{\neg A} ((\lambda H'''_A . H'''_A) \; H_A))$$

Introduction

Constructivization rewrite rules

Combining rewrite systems

Experimental results

Conclusion

Example: A ⇒ A

$$\lambda H_A.\underline{\text{dn}}_A(\lambda H_{\neg A}.\; H_{\neg A}((\lambda H'''_A.H'''_A)\ H_A))$$

$$(\lambda H_A.\pi_B(H_A))\ \pi_A \hookrightarrow \pi_B(\pi_A)$$

Example: $A \Rightarrow A$

$$\lambda H_A.\underline{\mathsf{dn}}_A(\lambda H_{\neg A}.\ H_{\neg A}\ H_A)$$

Example: $A \Rightarrow A$

$$\lambda H_A.\underline{\text{dn}}_A(\lambda H_{\neg A}. H_{\neg A}\ H_A)$$

$$\underline{\text{dn}}_A(\lambda H_{\neg A}.H_{\neg A}\ \pi_A) \hookrightarrow \pi_A$$

Example: A $\Rightarrow$ A

$\lambda H_A.H_A$

A rewrite system for ¬¬ A ⇒ A

The following statements are constructive theorems:

- $d_\top : \underline{\mathrm{DN}}_\top$
- $d_\bot : \underline{\mathrm{DN}}_\bot$
- $d_\wedge : (\underline{\mathrm{DN}}_A \wedge \underline{\mathrm{DN}}_B) \Rightarrow \underline{\mathrm{DN}}_{A \wedge B}$
- $d_\Rightarrow : \underline{\mathrm{DN}}_B \Rightarrow \underline{\mathrm{DN}}_{A \Rightarrow B}$
- $d_\forall : (\forall x. \underline{\mathrm{DN}}_{P(x)}) \Rightarrow \underline{\mathrm{DN}}_{\forall x. P(x)}$

where $\underline{\mathrm{DN}}_A := \neg\neg A \Rightarrow A$
which leads to the following partial definition of $\underline{\mathrm{dn}}$:

A rewrite system for $\neg\neg A \Rightarrow A$

$\underline{\mathrm{DN}}_A := \neg\neg A \Rightarrow A$
$\underline{\mathrm{dn}}_A : \underline{\mathrm{DN}}_A$

- $\underline{\mathrm{dn}}_\top \hookrightarrow d_\top$
- $\underline{\mathrm{dn}}_\bot \hookrightarrow d_\bot$
- $\underline{\mathrm{dn}}_{A \wedge B} \hookrightarrow d_\wedge(\underline{\mathrm{dn}}_A, \underline{\mathrm{dn}}_B)$
- $\underline{\mathrm{dn}}_{A \Rightarrow B} \hookrightarrow d_\Rightarrow(\underline{\mathrm{dn}}_B)$
- $\underline{\mathrm{dn}}_{\forall x.P(x)} \hookrightarrow d_\forall(\lambda x.\underline{\mathrm{dn}}_{P(x)})$

A rewrite system for ¬¬ A ⇒ A

$\underline{\text{DN}}_A := \neg\neg A \Rightarrow A$
$\underline{\text{dn}}_A : \underline{\text{DN}}_A$

- $\underline{\text{dn}}_\top \hookrightarrow d_\top$
- $\underline{\text{dn}}_\bot \hookrightarrow d_\bot$
- $\underline{\text{dn}}_{A \wedge B} \hookrightarrow d_\wedge(\underline{\text{dn}}_A, \underline{\text{dn}}_B)$
- $\underline{\text{dn}}_{A \Rightarrow B} \hookrightarrow d_\Rightarrow(\underline{\text{dn}}_B)$
- $\underline{\text{dn}}_{\forall x.P(x)} \hookrightarrow d_\forall(\lambda x.\underline{\text{dn}}_{P(x)})$
- $\underline{\text{dn}}_A\ (\lambda H_{\neg A}.H_{\neg A}\ \pi_A) \hookrightarrow \pi_A$

A rewrite system for ¬¬ A ⇒ A

$\underline{\mathrm{DN}}_A := \neg\neg A \Rightarrow A$
$\underline{\mathrm{dn}}_A : \underline{\mathrm{DN}}_A$

- $\underline{\mathrm{dn}}_\top \hookrightarrow d_\top$
- $\underline{\mathrm{dn}}_\bot \hookrightarrow d_\bot$
- $\underline{\mathrm{dn}}_{A \wedge B} \hookrightarrow d_\wedge(\underline{\mathrm{dn}}_A, \underline{\mathrm{dn}}_B)$
- $\underline{\mathrm{dn}}_{A \Rightarrow B} \hookrightarrow d_\Rightarrow(\underline{\mathrm{dn}}_B)$
- $\underline{\mathrm{dn}}_{\forall x.P(x)} \hookrightarrow d_\forall(\lambda x.\underline{\mathrm{dn}}_{P(x)})$
- $\underline{\mathrm{dn}}_A\ (\lambda H_{\neg A}.H_{\neg A}\ \pi_A) \hookrightarrow \pi_A$
- $\underline{\mathrm{dn}}_A\ (\lambda \_.\pi_\bot) \hookrightarrow E_\bot^A(\pi_\bot)$

Elimination of negation proofs

Higher-order rewrite rules

- $\underline{\text{dn}}_A\ (\lambda H_{\neg A}.H_{\neg A}\ \pi_A) \hookrightarrow \pi_A$
- $\underline{\text{dn}}_A\ (\lambda \_.\pi_\bot) \hookrightarrow E_\bot^A(\pi_\bot)$

Elimination of negation proofs

Higher-order rewrite rules

- $\underline{\mathsf{dn}}_A\ (\lambda H_{\neg A}.H_{\neg A}\ \pi_A) \hookrightarrow \pi_A$
- $\underline{\mathsf{dn}}_A\ (\lambda\_.\pi_\perp) \hookrightarrow E_\perp^A(\pi_\perp)$
- $E_\perp^A(\pi_{\neg A}\ \pi_A) \hookrightarrow \pi_A$

Elimination of negation proofs

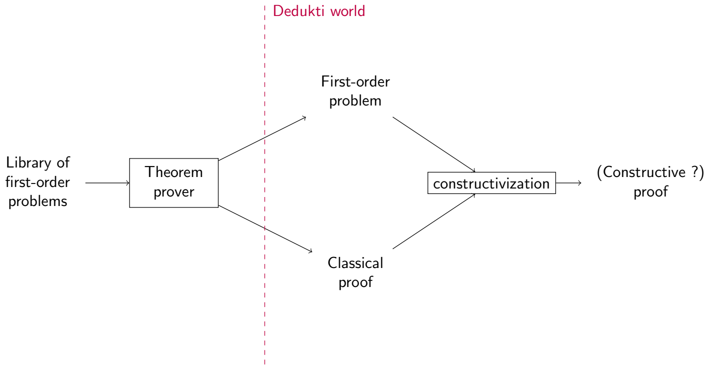
Higher-order rewrite rules

- $\underline{\text{dn}}_A\ (\lambda H_{\neg A}.H_{\neg A}\ \pi_A) \hookrightarrow \pi_A$
- $\underline{\text{dn}}_A\ (\lambda\_.\pi_\bot) \hookrightarrow E_\bot^A(\pi_\bot)$
- $E_\bot^A(\pi_{\neg A}\ \pi_A) \hookrightarrow \pi_A$
- $E_\bot^{A\Rightarrow B}(\pi_\bot)\ \pi_A \hookrightarrow E_\bot^B(\pi_\bot)$
- $\lambda H_A.E_\bot^B(\pi_\bot) \hookrightarrow E_\bot^{A\Rightarrow B}(\pi_\bot)$

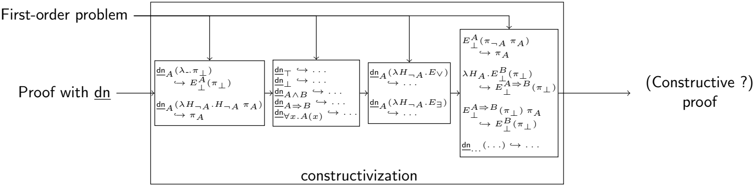
Confluence

- $\lambda H_\top . H_\top \hookleftarrow \lambda H_\top . \underline{\mathsf{dn}}_\top \; (\lambda H_{\neg\top} . H_{\neg\top} \; H_\top) \hookrightarrow \lambda H_\top . I_\top$

The constructivizatoin pipeline



Zooming at the constructivization box



Meta-Dedukti

Dedukti is good at writing Dedukti code.

- Normalize the same term with respect to several successive rewrite systems
- Rewrite system = meta-programming stage
- Confluence of the last

Results

Timeout: 10s (Zenon) and 10min (Dedukti)
Memory limit: 2GB
Library: TPTP v6.3.0

Problems	6528
Classical proofs	1258
Normalized proofs	1240
Constructive proofs	856
Constructivization rate	68%

Related work

- Zenonide
  - Constructivization in sequent calculus
  - Specific to Zenon
  - Similar performances (66.8%)
  - Complementary (76% combined)
- iLeanCop
  - Best intuitionistic theorem prover according to ILTP
  - No proof output
  - More ambitious: complete for first-order intuitionistic logic
  - Surprise: we are competitive

Conclusion

- Simple heuristics for proof constructivization
- Dedukti as a meta-language for transforming proofs
- Constructivizators can be chained
- Most classical proofs generated by Zenon are constructive
- Theorems = total functions, Axioms = partial functions

Future work

- Syntax for the meta-language
- Try other provers (iProver Modulo, VeriT)
- Deduction modulo
- Higher-order
- Intermediate logics
- Other axioms (extensionality, univalence, choice)

Future work

- Syntax for the meta-language
- Try other provers (iProver Modulo, VeriT)
- Deduction modulo
- Higher-order
- Intermediate logics
- Other axioms (extensionality, univalence, choice)

Thank you for your attention!