Deduction modulo theory

Gilles Dowek

Joint work with ∞

I. Comparing research projects in proof theory

Weaker vs. stronger systems

Several directions at the same time

Decomposing proofs, propositions, connectives, etc., into more atomic objects

Weaker than Predicate logic: propositional logic, linear logic, deep inference, equational logic, explicit substitution, etc.

Very little can be expressed in pure Predicate logic

Stronger than Predicate logic: axiomatic theories, modal logics, types theories, Deduction modulo theory, etc.

Logical vs. theoretical systems

Stronger than pure Predicate logic

New logical constants, new rules: modal logics, simple type theory, etc.

Function symbols and predicate symbols within Predicate logic, axioms: arithmetic, set theory, simple type theory, etc.

Deduction modulo theory: theoretical rather than logical

A framework in which it is possible to define many theories

Axioms vs. reduction rules

A theory: a set of ~~axioms~~ reduction rules

Axioms jeopardize: cut free proofs end with an introduction rule, witness property, search space of $\bot$ empty, etc.

$$0 = 0 \longrightarrow \top$$

$$S(x) = 0 \longrightarrow \bot$$

$$0 = S(y) \longrightarrow \bot$$

$$S(x) = S(y) \longrightarrow x = y$$

Prove 4 = 4, Peano third and fourth axiom

Deduction vs. computation

if $A \longrightarrow^* \top$, then A provable

Not the converse

Indeed, reducibility to $\top$ decidable, not provability

On the opposite

If $A \longrightarrow^* \top$, proof of A just a computation (not a genuine deduction)

The origins of Deduction modulo theory

Automated theorem proving: equational unification (A, β)

Definitional equality in Martin-Löf's type theory

Prawitz' Folding and unfolding rules

II. Problems and results: an overview

Expressing theories in Deduction modulo theories

Specific theories: Simple type theory, Arithmetic, Set theory, ...

General method for propositional logic, predicate logic: consistency implies cut elimination (classical case), but not optimal efficiency

Partial methods for constructive logic (consistency not enough, what about consistency + witness?)

Automated theorem proving

Resolution modulo theory: too complex: clauses rewrite to non-clausal propositions

Polarized resolution modulo theory (and as a restriction of Resolution, SOS, SR)

Ordered polarized resolution modulo theory (iProver modulo)

Tableaux modulo theory: very good results for class theory (second-order logic, B-set theory)

Super Zenon and Zenon modulo

Models

Very close to Predicate logic: same models

Validity of rewrite rules: $A \equiv B$ implies $[\![A]\!]_\phi = [\![B]\!]_\phi$

Extension to models valued in Boolean / Heyting algebras

But: if $\vdash A \Leftrightarrow B$, then $[\![A]\!]_\phi = [\![B]\!]_\phi$ as well

Too extensional, drop antisymmetry

if $\vdash A \Leftrightarrow B$, then $[\![A]\!]_\phi \leq [\![B]\!]_\phi$ and $[\![A]\!]_\phi \geq [\![B]\!]_\phi$

if $A \equiv B$, then $[\![A]\!]_\phi = [\![B]\!]_\phi$

Many theories have a model in any pre-Heyting algebra

Cut elimination

Depends on the theory: $P \longrightarrow P \Rightarrow Q$ no,
$P \longrightarrow Q \Rightarrow P$ yes

General criterion: a model valued in the (pre-Heyting) algebra of Reducibility candidates

Only the construction of the model is specific

Dependent types

Algorithmic interpretation of proofs (Curry-de Bruijn-Howard isomorphism): usually for specific theories ($\lambda\Pi$-calculus, Gödel's system T, Martin-Löf's type theory, Girard's system F, Calculus of (Inductive) Constructions, ...)

$\lambda\Pi$-calculus + rewriting: all theories ($\varnothing$, Arithmetic, Simple type theory, Set theory, ...)

Decouple algorithmic interpretation of proofs ($\lambda\Pi$-calculus) from the choice of a theory (rewriting)

Embedding Pure Type Systems in the $\lambda\Pi$-calculus modulo theory

III. Focus on Dedukti

An proof-checker for $\lambda\Pi$-modulo

Just a proof-checker (no tactics, program extraction, user interface, ...)

A suite of programs rather than a monolithic system

Difficult to implement : compile reduction (lambda-calculus + arbitrary rewrite rules), but now an efficient implementation

Download it and play with it

Why is it called Dedukti?

$\lambda\Pi$-modulo theory: A logical framework (STT, PTS, etc.)

Importing proofs from other systems

Full library of HOL

Coq, Focalize: under progress

First-order proofs and proofs in Deduction modulo theory (iProver, Zenon, etc.): represent classical proofs

PVS: future work

Do your own

Future work: interoperability

If $A \Rightarrow B$ proved in $\mathcal{T}$ and A proved in $\mathcal{T}'$

prove B in $\mathcal{T} \cup \mathcal{T}'$

$\mathcal{T} \cup \mathcal{T}'$ consistent? Cut elimination?

The HTML of proofs?

Future work: reverse mathematics

A proof of $0 + x = x$ in a strong system (CIC, Z)

What rules are actually used?

What is the minimal theory where we can prove this?

To which system can we export this proof?

Future work: tactics

A formalization of the Cubical model of HoTT

Would be great if we had rewrite rules at the level of tactics

Can we design a better tactic language if rewriting is primitive?