Deduction modulo theory

Gilles Dowek

Inria, 23 avenue d'Italie, CS 81321, 75214 Paris Cedex 13, France.
gilles.dowek@inria.fr

1 Introduction

1.1 Weaker vs. stronger systems

Contemporary proof theory goes into several directions at the same time. One of them aims at analysing proofs, propositions, connectives, etc., that is at decomposing these objects into more atomic ones. This often leads to design systems that are weaker than Predicate logic, but that have better algebraic or computational properties, and to try to reconstruct part of Predicate logic on top of these systems. Propositional logic, linear logic, deep inference, equational logic, explicit substitution calculi, etc. are examples of such systems. From this view point, Predicate logic appears more as the ultimate goal of the journey, than as its starting point.

Another direction considers that very little can be expressed in pure Predicate logic and that stronger systems are needed, for instance to express genuine mathematical proofs. Axiomatic theories, modal logics, types theories, etc. are examples of such systems that are more expressive than pure Predicate logic. There, Predicate logic is the starting point of the journey.

Although both points of view coexist in many research projects, these two approaches to proof theory often lead to different systems and different problems.

Deduction modulo theory is part of the second group, as it focuses on proofs in theories. The concern of integrating theories to proof theory is that of several research groups. See, for instance, [48] and [50] for related approaches.

1.2 Logical vs. theoretical systems

To design a system stronger than pure Predicate logic, several ways are possible. One is to extend Predicate logic with new logical constants, that is to design a logic, the second is to introduce function symbols and predicate symbols within Predicate logic and state axioms expressing the meaning these symbols, that is to design a theory. The first approach can be illustrated by modal logics, the second by arithmetic or set theory. Simple type theory belongs to both groups as it can be defined either as a logic, in which case is is more often called *higher-order logic*, or as a theory in Predicate logic.

Deduction modulo theory is part of the second, theoretical rather than logical, group, as, like the axiomatic approach, it keeps Predicate logic as it is and allows to define theories within this framework.

1.3 Axioms vs. reduction rules

But, the main difference between Deduction modulo theory an the axiomatic approach is that a theory in Deduction modulo theory is not defined as a set of axioms, but as a set of reduction rules.

Indeed, axioms jeopardize most of the properties of proofs of pure Predicate logic. For instance, in pure Predicate logic, a constructive cut free proof always ends with an introduction rule, hence a constructive cut free existential proof always ends with an introduction rule of the existential quantifier. But this result does not extend to axiomatic theories, as a constructive cut free proof in a theory may also end, for instance, with the axiom rule.

In the same way, in automated theorem proving in pure Predicate logic, the search space of the proposition $\bot$ is always finite. But this result does not extend to axiomatic theories, that can generate an infinite search space for the proposition $\bot$.

To overcome these problems, theories, in Deduction modulo theory, are defined as sets of reduction rules. For instance the axioms

$$\forall y\ (0 + y = y)$$

$$\forall x \forall y\ (S(x) + y = S(x + y))$$

are replaced by the reduction rules

$$0 + y \longrightarrow y$$

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

These reduction rules define a congruence $\equiv$ on propositions, and deduction is performed modulo this congruence. For instance, with the reduction rules above the propositions $2+2 = 4$ and $4 = 4$ are congruent, hence any proof of the latter is a proof of the former. If we add rules directly rewriting atomic propositions to arbitrary propositions to define equality [1]

$$0 = 0 \longrightarrow \top$$

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

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

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

then the proposition $2 + 2 = 4$ and $\top$ are congruent, and any proof of $\top$, for instance the mere application of the introduction rule of $\top$, is a proof of the proposition $2 + 2 = 4$

$$\frac{}{\vdash 2 + 2 = 4}\ \top\text{-intro}$$

1.4 Deduction vs. computation

In the example above, the proposition $2+2=4$ is provable because it reduces to $\top$. More generally, all propositions that reduce to $\top$ are provable. But the converse is not true: not all provable propositions reduce to $\top$. Indeed, reducibility to $\top$ is often a decidable property, while provability is not.

On the opposite, the fact that the proposition $2+2=4$ has a trivial proof because it reduces to $\top$, and not a complex one that would involve the axioms of equality and the axioms addition, shows that the truth of this proposition rests on a mere computation and not on a genuine deduction.

Thus, Deduction modulo theory also permits to distinguish, in a proof, the computation part from the deduction part, while Predicate logic flattens computation and deduction.

1.5 The origins of Deduction modulo theory

Deduction modulo theory was first introduced in the area of automated theorem proving.

Indeed, in automated theorem proving, instead of using equational axioms, for instance the associativity axiom, we often replace standard unification with equational unification, for instance unification modulo associativity [53]. In the same way, in Simple type theory, instead of using the β-conversion axiom, we replace standard unification by equational unification modulo β-equivalence: higher-order unification [2, 44, 45].

To explain why such a method works, a solution is to introduce first an inference system where propositions are identified modulo associativity, or modulo β-equivalence, to prove the equivalence with the axiomatic presentation and then, as propositions are identified modulo this congruence, everything, in particular unification, must be performed modulo this congruence.

So Deduction modulo theory comes from automated theorem proving. But it was soon understood that this idea of identifying propositions modulo a congruence was also the idea behind the notion of definitional equality in Martin-Löf's Intuitionistic type theory [49] and that Deduction modulo theory could also be seen as an extension of this notion of definitional equality to Predicate logic.

Another source of inspiration is the extension of Natural deduction with folding and unfolding rules, introduced by Prawitz [54, 21, 40, 37, 22, 25]. If it is not possible to identify an atomic proposition P with a proposition A, in this system, it is possible to introduce two non logical deduction rules

$$\frac{A}{P} \qquad\qquad \frac{P}{A}$$

2 Proof Systems

The idea of reasoning modulo a theory can be used in different formalisms: Natural deduction, Sequent calculus, λ-calculus, etc. Thus, Deduction modulo theory exists in many flavors.

2.1 Natural Deduction modulo theory

Let us start with constructive Natural deduction. The rules of constructive Natural deduction modulo theory are obtained by transforming the rules of constructive Natural deduction, to allow to use of the congruence. For instance, the rule

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow\text{-elim}$$

is transformed into

$$\frac{\Gamma \vdash C \quad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow\text{-elim} \text{ if } C \equiv (A \Rightarrow B)$$

where the proposition $A \Rightarrow B$ is replaced by any congruent proposition C. Applying the same transformation to all Natural deduction rules yields the following system.

$$\frac{}{\Gamma, A \vdash B} \text{ axiom if } A \equiv B$$

$$\frac{}{\Gamma \vdash A} \top\text{-intro if } A \equiv \top$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A} \bot\text{-elim if } B \equiv \bot$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash C} \wedge\text{-intro if } C \equiv (A \wedge B)$$

$$\frac{\Gamma \vdash C}{\Gamma \vdash A} \wedge\text{-elim if } C \equiv (A \wedge B)$$

$$\frac{\Gamma \vdash C}{\Gamma \vdash B} \wedge\text{-elim if } C \equiv (A \wedge B)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash C} \vee\text{-intro if } C \equiv (A \vee B)$$

$$\frac{\Gamma \vdash D \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \vee\text{-elim if } D \equiv (A \vee B)$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash C} \vee\text{-intro if } C \equiv (A \vee B)$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash C} \Rightarrow\text{-intro if } C \equiv (A \Rightarrow B)$$

$$\frac{\Gamma \vdash C \quad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow\text{-elim if } C \equiv (A \Rightarrow B)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash B} \langle x, A\rangle\ \forall\text{-intro if } B \equiv (\forall x\ A) \text{ and } x \notin FV(\Gamma)$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash C} \langle x, A, t\rangle\ \forall\text{-elim if } B \equiv (\forall x\ A) \text{ and } C \equiv [t/x]A$$

$$\frac{\Gamma \vdash C}{\Gamma \vdash B} \langle x, A, t\rangle\ \exists\text{-intro if } B \equiv (\exists x\ A) \text{ and } C \equiv [t/x]A$$

$$\frac{\Gamma \vdash C \quad \Gamma, A \vdash B}{\Gamma \vdash B} \langle x, A\rangle\ \exists\text{-elim if } C \equiv (\exists x\ A) \text{ and } x \notin FV(\Gamma B)$$

For instance, consider the congruence defined by the rule

$$x \subseteq y \longrightarrow \forall z\ (z \in x \Rightarrow z \in y)$$

The sequent $\vdash s \subseteq s$ has the proof

$$\dfrac{\dfrac{\dfrac{}{z \in s \vdash z \in s}\,\text{axiom}}{\vdash z \in s \Rightarrow z \in s}\,\Rightarrow\text{-intro}}{\vdash s \subseteq s}\,\langle z, z \in s \Rightarrow z \in s\rangle\ \forall\text{-intro}$$

Note that if two propositions A and B are provably equivalent, that is when $A \Leftrightarrow B$ is provable, then the proposition A has a proof if and only if the proposition B has a proof, but the propositons A and B need not have the same proofs. In contrast, if two propositions are congruent, that is when $A \equiv B$, then every proof of A is a proof of B and vice-versa, thus the propositions A and B have the same proofs.

Sequent calculus modulo theory can be defined in a similar way. See, for instance, [35].

Another variant of Natural deduction modulo theory and Sequent calculus modulo theory is *Super-deduction* [57, 16]. In Super-deduction, new deduction rules are computed from the reduction rules. For instance, the reduction rule

$$x \subseteq y \longrightarrow \forall z\ (z \in x \Rightarrow z \in y)$$

yields the deduction rules

$$\dfrac{\Gamma, z \in x \vdash z \in y}{\Gamma \vdash x \subseteq y}\, z \notin FV(\Gamma)$$

$$\dfrac{\Gamma \vdash x \subseteq y \quad \Gamma \vdash z \in x}{\Gamma \vdash z \in y}$$

These rules are very natural to use: in informal mathematics, to prove $x \subseteq y$, we often consider a generic z in x and prove that it is in y. The fact that these derived rules use atomic propositions also explains why connectives and quantifiers are almost never used in informal mathematics.

2.2 Polarized deduction modulo theory

In the design of Natural deduction modulo theory above we have transformed the rule

$$\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\,\Rightarrow\text{-elim}$$

into

$$\dfrac{\Gamma \vdash C \quad \Gamma \vdash A}{\Gamma \vdash B}\,\Rightarrow\text{-elim}\ \text{if } C \equiv (A \Rightarrow B)$$

where the proposition $A \Rightarrow B$ is replaced by any proposition C such that $C \equiv (A \Rightarrow B)$.

In a similar way, in the Sequent calculus modulo theory, we would transform the rule

$$\dfrac{\Gamma \vdash A \quad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}\,\Rightarrow\text{-left}$$

into

$$\frac{\Gamma \vdash A \quad \Gamma, B \vdash \Delta}{\Gamma, C \vdash \Delta} \Rightarrow\text{-left} \text{ if } C \equiv (A \Rightarrow B)$$

where the proposition $A \Rightarrow B$ is replaced by any proposition C such that $C \equiv (A \Rightarrow B)$.

In these deduction rules, the reduction rules are just used to define the congruence $\equiv$. In fact, this congruence does not even need to be defined with reduction rules and it could be any congruence, provided it is decidable and it does not identify non-atomic propositions with different head symbols. But we may also want to stress that computation is oriented and take, in these rules, the condition $C \longrightarrow^* (A \Rightarrow B)$ instead of $C \equiv (A \Rightarrow B)$, meaning that in the sequent $\Gamma, C \vdash \Delta$, can only reduce the proposition C.

In particular, the axiom rule

$$\frac{}{\Gamma, A \vdash B} \text{ axiom} \text{ if } A \equiv B$$

would be restated

$$\frac{}{\Gamma, A \vdash B} \text{ axiom} \text{ if } A \longrightarrow^* C \text{ and } B \longrightarrow^* C$$

If t and u are two terms such that $t \equiv u$, it is still possible to prove the sequent $P(t) \vdash P(u)$. But when t and u do not have a common reduct, the proof of $P(t) \vdash P(u)$ contains cuts. In other words, the Sequent calculus modulo theory has the cut elimination property if and only if the reduction system is confluent [27] and Newman's algorithm [51] that permits to transform an equational proof into a valley proof appears to be a cut-elimination algorithm.

This idea can be developed further: the reduction rule

$$x \subseteq y \longrightarrow \forall z \ (z \in x \Rightarrow z \in y)$$

permits to prove the equivalence

$$x \subseteq y \Leftrightarrow \forall z \ (z \in x \Rightarrow z \in y)$$

Thus, when the atomic proposition P reduces to the proposition A, P and A must be equivalent and, for instance, it is not possible to reduce $\mathit{Isosceles}(x)$ to $\mathit{Equilateral}(x)$ because a triangle may be isosceles without being equilateral.

More generally, it is easy to transform an axiom of the form $P \Leftrightarrow A$ into a reduction rule $P \longrightarrow A$, but it is not easy to transform an axiom of the form $P \Rightarrow A$ into a reduction rule. However, even if, with such an axiom, we should not be able to reduce a goal P into A, we should be able to reduce a hypothesis P into A.

This leads to an extension of Deduction modulo theory, called *Polarized deduction modulo theory* where reduction rules are classified into positive and negative, the positive rules may apply to the positive occurrences of atomic propositions and the negative ones to the negative occurrences.

For instance, in Polarized sequent calculus modulo theory, the left rule of the implication is stated

$$\frac{\Gamma \vdash A \quad \Gamma, B \vdash \Delta}{\Gamma, C \vdash \Delta} \Rightarrow\text{-left} \quad \text{if } C \longrightarrow_{-}^{*} (A \Rightarrow B)$$

and its right rule

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash C} \Rightarrow\text{-right} \quad \text{if } C \longrightarrow_{+}^{*} (A \Rightarrow B)$$

Polarized deduction modulo theory is the flavor of Deduction modulo theory that is more often used in automated theorem proving.

The first reason is that, in clause based theorem proving, a reduction rule of the form

$$x \in y \cup z \longrightarrow x \in y \vee x \in z$$

can be used to reduce a positive literal in a clause but not a negative one. For instance, the clause $L_1 \vee L_2 \vee a \in b \cup c$ reduces to the clause $L_1 \vee L_2 \vee a \in b \vee a \in c$, but the clause $L_1 \vee L_2 \vee \neg a \in b \cup c$ reduces to the proposition $L_1 \vee L_2 \vee \neg(a \in b \vee a \in c)$ that is not a clause. In contrast, if we replace this reduction rule by the polarized rules

$$x \in y \cup z \longrightarrow_{-} x \in y \vee x \in z$$

$$x \in y \cup z \longrightarrow_{+} x \in y$$

$$x \in y \cup z \longrightarrow_{+} x \in z$$

then the clause $L_1 \vee L_2 \vee \neg a \in b \cup c$ reduces to the clauses $L_1 \vee L_2 \vee \neg a \in b$ and to $L_1 \vee L_2 \vee \neg a \in c$. More generally, any reduction system can be transformed this way to a clausal one [39].

The second reason is that any consistent set of axioms can be transformed into a Polarized reduction system that is classically equivalent [26, 12]. Interestingly, this result has been proved with applications to automated theorem proving in mind, it uses automated theorem proving methods, but it is a purely proof-theoretical result.

2.3 Expressing theories in Deduction modulo theory

The early work on expressing theories in Deduction modulo theory was focused on specific theories: Simple type theory [30], Arithmetic [36, 1], set theory [34], etc.

Then, as already said, systematic ways of transforming sets of axioms into sets of reduction rules have been investigated [26, 12].

2.4 The $\lambda\Pi$-calculus modulo theory

The early developments of Deduction modulo theory were independent of the proofs-as-algorithms paradigm: in Deduction modulo theory, like in Predicate logic, terms, propositions, and proofs belong to three different languages, and proofs are not terms. But we have mentioned that one of the origins of Deduction

modulo theory was the definitional equality of Martin-Löf's Intuitionistic type theory. This suggests that this idea of identifying congruent propositions can also be useful in systems based on the proofs-as-algorithms paradigm.

The simplest system to express proofs of Predicate logic as algorithms is the λ-calculus with dependent types [43], also know as the $\lambda\Pi$-calculus. This leads to the development of an extension of the $\lambda\Pi$-calculus, called the $\lambda\Pi$-calculus modulo theory [20]. This system is closely related to Martin-Löf's logical framework [52].

Any theory that can be expressed in minimal Deduction modulo theory can be expressed in the $\lambda\Pi$-calculus modulo theory, in particular Simple type theory. An interesting point here is that the Calculus of Constructions [18] has been designed to express proofs of Simple type theory as algorithms. It happens that $\lambda\Pi$-calculus modulo theory also can express those proofs as algorithms. This suggests that the Calculus of Constructions itself could be expressed in the $\lambda\Pi$-calculus modulo theory, and this is indeed the case [20]. The embedding of the Calculus of Constructions into the $\lambda\Pi$-calculus modulo theory follows closely the expression of Simple type theory in Deduction modulo theory.

It happens *a posteriori* that this embedding of the Calculus of Constructions into the $\lambda\Pi$-calculus modulo theory can be seen as an extension of the $\lambda\Pi$-calculus with an impredicative universe *à la* Tarski [3] and thus that there is a strong link between the expression of Simple type theory in Predicate logic and the notion of universe *à la* Tarski.

3 Properties

3.1 Models

The usual models of classical Predicate logic, valued in $\{0,1\}$, can be used for Deduction modulo theory. A congruence $\equiv$ is said to be valid in a model when $A \equiv B$ implies $[\![A]\!]_\phi = [\![B]\!]_\phi$ for all valuation ϕ, and a soundness and completeness theorem can be proved using standard methods.

Like for Predicate logic, the set of truth values $\{0,1\}$ can be extended to any Boolean algebra, allowing to prove a stronger completeness theorem: given a theory, there exists a model such that the propositions valid in this model are exactly the propositions provable in this theory.

Boolean algebras can be extended to Heyting algebras to define a sound and complete semantics for constructive logic.

However in all these models valued in $\{0,1\}$, Boolean algebras and Heyting algebras, two provably equivalent propositions always have the same truth value: if $A \Leftrightarrow B$ is valid, then $A \Rightarrow B$ and $B \Rightarrow A$ are valid, hence $[\![A]\!]_\phi \leq [\![B]\!]_\phi$ and $[\![B]\!]_\phi \leq [\![A]\!]_\phi$ and by antisymmetry $[\![A]\!]_\phi = [\![B]\!]_\phi$. Thus, there is no way to make a difference, in the model, between provable equivalence and congruence: whether A and B are equiprovable or have the same proofs, they have the same truth value.

A way to overcome this is to extend Boolean algebras and Heyting algebras by dropping the antisymmetry condition on the relation $\leq$. This relation is then

a pre-order and the algebras defined this way can be called *pre-Boolean* algebras [9] and *pre-Heying* algebras [28]. The soundness theorem is proved exactly the same way—antisymmetry is never used in this proof—, and the completeness is simpler as the class of models larger. This corresponds to the intuition that the relation $\leq$, defined by $A \leq B$ if $A \Rightarrow B$ is provable, is reflexive and transitive, but not antisymmetric.

This way, two provably equivalent propositions may be interpreted by distinct truth values, unlike two congruent propositions that must be interpreted by the same truth value, and it is possible to define models where a proposition A is interpreted by the set of its proofs.

When a theory has a model valued in some pre-Heyting algebra it is consistent, when it has a model valued in all pre-Heyting algebras it is said to be *super-consistent*.

3.2 Cut-elimination

Proof-reduction is defined in Deduction modulo theory in the same way as in Predicate logic, but the difference is that it does not always terminate. Indeed, if we define a theory with the reduction rule $P \longrightarrow (P \Rightarrow Q)$ the sequent $\vdash Q$ has the following proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P \vdash P \Rightarrow Q}\,\text{axiom}\quad\dfrac{}{P \vdash P}\,\text{axiom}}{P \vdash Q}\,\Rightarrow\text{-elim}}{\vdash P \Rightarrow Q}\,\Rightarrow\text{-intro}\qquad\dfrac{\dfrac{\dfrac{}{P \vdash P \Rightarrow Q}\,\text{axiom}\quad\dfrac{}{P \vdash P}\,\text{axiom}}{P \vdash Q}\,\Rightarrow\text{-elim}}{\vdash P}\,\Rightarrow\text{-intro}}{\vdash Q}\,\Rightarrow\text{-elim}$$

that contains a cut and that reduces to itself.

Moreover, it is possible to prove that all cut free, that is irreducible, proofs end with an introduction rule, thus not only this proof does not terminate, but the sequent $\vdash Q$ has no cut free proof.

And a similar example can be built with a terminating reduction system [35].

Thus, unlike for axiomatic theories, the notion of proof-reduction can be defined in a generic, theory independent, way, and the properties of cut free proofs, such as the property that the last rule of a cut free proof is an introduction rule can be proved in a generic way. But, the proof-termination theorem itself must be proved for each theory.

Using a method introduced by to prove the termination of proof reduction in Simple type theory [38], we can prove that proof-reduction terminates in some theory, if a reducibility candidate $[\![A]\!]$ can be associated to each proposition A, in such a way that two congruent propositions are associated with the same reducibility candidate [35]

$$A \equiv B \text{ implies } [\![A]\!] = [\![B]\!]$$

This association of a reducibility candidate to each proposition is thus a model valued in the algebra of the reducibility candidates and the condition that

two congruent propositions are associated with the same reducibility candidate is the validity of this congruence in this model.

This way we get that if a theory has a model valued in the algebra of reducibility candidates, then all proofs strongly terminate.

The algebra of reducibility candidates is a pre-Heyting algebra—but not a Heyting algebra—thus we also get that proof-reduction terminates in super-consistent theories.

This semantic view on termination of proof reduction theorems also permits to relate these termination proofs to the so called *semantic* cut-elimination proofs that proceed by proving a completeness result for cut free provability. First, without proving the termination of proof-reduction, it is possible to prove directly that, in a super-consistent theory, each provable proposition has a cut free proof [32]. This completeness proof does not use the pre-Heyting algebra of reducibility candidates but a simpler pre-Heyting algebra.

Then, in some non super-consistent theories, proof reduction does not terminate, but each provable proposition has a cut free proof [41]. An example is obtained by replacing the proposition Q by $\top$ in the example above. This proof still fails to terminate but the sequent $\vdash \top$ has another proof, that is cut free. Such cut-elimination theorems can only be proved via a completeness theorem and their constructive content is a proof transformation algorithm, that is not related to proof-reduction.

Finally, some theories do not have the cut elimination property, but they can be extended to theories that have this property by adding derivable reduction rules [15, 13]. This saturation process can be compared to Knuth-Bendix method [47]—remember that confluence is a cut-elimination property—that does not prove that all reduction systems are confluent, but that, in some cases, it is possible to extend a reduction system with derivable rules, to make it confluent.

3.3 Automated theorem proving methods

Deduction modulo theory has been introduced to design and study automated theorem proving methods. The first method introduced was a variant of Resolution [31] that was too complicated because rules were not polarized. Thus, clauses could rewrite to non clausal proposition that needed to be handled. Polarization permitted to simplify the method [29] and also to understand better its relation to other methods. This method is complete if and only if the theory has the cut-elimination property [42].

Imagine we have a clause

$$L_1 \vee L_2 \vee a \in b \cup c$$

and a negative reduction rule

$$x \in y \cup z \longrightarrow_{-} x \in y \vee x \in z$$

then applying this rule to this clause yields the clause

$$L_1 \vee L_2 \vee a \in b \vee a \in c$$

But instead of this reduction rule, we could have taken a clause

$$\underline{\neg x \in y \cup z} \vee x \in y \vee x \in z$$

and Resolution, applied to the literal $a \in b \cup c$ and the underlined literal in the new clause, would have yielded the same result. Thus, there is no need to extend Resolution to handle reduction rules, but reduction rules can just be seen as special clauses, called *one-way clauses*. The Resolution rule cannot be applied to two one-way clauses and when it is applied to a one-way clause and an ordinary one, only the literal corresponding to the left-hand side of the reduction rule can be used. Thus Polarized resolution modulo theory is just another variant of Equational resolution with selection. It is, in fact, an improvement of Resolution with the Set of support strategy [58] and of the Semantic resolution [56]. But, unlike others variants of Resolution with selection, its completeness is equivalent to a cut-elimination theorem. For instance the completeness of Polarized resolution modulo the rules of Simple type theory cannot be proved in Simple type theory [14].

These remarks also showed the way to combine this method with other selection strategies in Resolution. In particular, it has been shown that this restriction is compatible with Ordered resolution [10], which is surprising as the Set of support strategy and the Semantic resolution strategy is not.

Beside Resolution, other proof-search methods have been investigated, in particular direct search in cut free sequent calculus modulo theory, also known as the *tableaux* method [8].

4 Implementations

The early work on Deduction modulo theory only led to experimental implementations. But more mature systems have been developed in the recent years.

4.1 Dedukti

Dedukti [5, 7, 55] is an implementation of the $\lambda\Pi$-calculus modulo theory. It is thus based on the proofs-as-algorithms paradigm and proof-checking is reduced to type-checking. But type-checking itself may require an arbitrary amount of computation to check the congruence of two propositions.

Dedukti is a parametric system: by changing the reduction rules, we change the theory in which the proofs are checked. Thus Dedukti is a logical framework [43]. As the proofs of many different systems can be expressed in this framework Dedukti is mostly used to check proofs developed in other systems—hence its name: "to deduce" in Esperanto—: HOL [4], Focalize [17], Coq [6, 3], etc. as well as proofs produced by automated theorem proving systems, such as iProver modulo and Zenon modulo described below. The current goal of the project is to be able to interface proofs developed in different systems.

4.2 iProver modulo, Super Zenon and Zenon modulo

iProver modulo [11] is an implementation of Ordered polarized resolution modulo theory. It is developed as an extension of iProver.

Super Zenon [46] is an implementation of Tableaux modulo theory specifically designed for a variant of Class theory—Second order logic—called *B set theory*, and using Super-deduction instead of the original Deduction modulo theory.

Zenon modulo [23] is a generic implementation of Tableaux modulo theory method.

5 Trends and Open Problems

In recent years, the effort in Deduction modulo theory has been put on the development of implementations. In particular, we do not know how far we can go in interfacing proof systems using a logical framework such as Dedukti. We also need to investigate how having user defined reduction rule can impact tactic based proof development.

In automated theorem proving we do not understand yet how to mix Resolution modulo theory with equality specific methods such as superposition.

Some theories, such as alternating push-down systems, can be proved decidable through an encoding in Deduction modulo theory [33]. But we do not know yet how far we can go in using Deduction modulo theory for proving decidability results and incorporating decision methods in theorem provers.

On the more proof-theoretical side, we know that super-consistency is a sufficient condition for the strong termination of proof reduction but we do not know if it is a necessary condition. As suggested in [19], the notion of super-consistency may require some adjustment so that we can prove that it is a necessary and sufficient condition for proof termination. Finally, some extension of Deduction modulo theories allow congruences that identify non-atomic propositions with different head-symbols [24], in particular isomorphic types such as $A \Rightarrow (B \wedge C)$ and $(A \Rightarrow B) \wedge (A \Rightarrow C)$, but we do not know yet how far we can go in this direction.

References

1. L. Allali. Algorithmic equality in Heyting arithmetic modulo. *Higher Order Rewriting*, 2007.
2. P.B. Andrews. Resolution in type theory. *The Journal of Symbolic Logic*, 36, 1971, pp. 414-432.
3. A. Assaf. A Calculus of Constructions with explicit subtyping. Manuscript, 2014.
4. A. Assaf. *Traduction de HOL en Dedukti.* Master thesis, 2012.
5. M. Boespflug. *Conception d'un noyau de vérification de preuves pour le lambda-Pi-calcul modulo.* Doctoral thesis, École polytechnique, 2011.
6. M. Boespflug and G. Burel. CoqInE: translating the calculus of inductive constructions into the lambda-pi-calculus modulo. *Proceedings of the Second International Workshop on Proof Exchange for Theorem Proving*, CEUR Workshop Proceedings 878, 2012, pp. 44-50.

7. M. Boespflug, Q. Carbonneaux, and O. Hermant. The $\lambda\Pi$-calculus Modulo as a Universal Proof Language. *Proceedings of the Second International Workshop on Proof Exchange for Theorem Proving*, CEUR Workshop Proceedings 878, 2012, pp. 28-43.
8. R. Bonichon and O. Hermant. A semantic completeness proof for TaMeD. *LPAR*, Lecture Notes in Computer Science 4246, Springer, 2006, pp. 167-181.
9. A. Brunel, O. Hermant, and C. Houtmann. Orthogonality and Boolean Algebras for Deduction Modulo. *Typed Lambda Calculus and Applications*, Lecture Notes in Computer Science 6990, Springer, 2011, pp. 76-90.
10. G. Burel. Embedding deduction modulo into a prover. A. Dawar and H. Veith (eds.), *CSL*, Lecture Notes in Computer Science, 6247, Springer, 2010, pp. 155-169.
11. G. Burel. Experimenting with deduction modulo. V. Sofronie-Stokkermans and N. Bjørner (eds.), *CADE*, Lecture Notes in Computer Science 6803, Springer, 2011, pp. 162-176.
12. G. Burel. From Axioms to Rewriting Rules. Manuscript.
13. G. Burel. Cut Admissibility by Saturation. *RTA-TLCA*, Lecture Notes in Computer Science 8560, Springer, 2014.
14. G. Burel and G. Dowek. How can we prove that a proof search method is not an instance of another? *Fourth International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice*. ACM International Conference Proceeding Series, 2009.
15. G. Burel and C. Kirchner. Regaining cut admissibility in deduction modulo using abstract completion. *Information and Computation*, 208(2), 2010, pp. 140-164.
16. P. Brauner, C. Houtmann, and C. Kirchner. Superdeduction at work. *Rewriting, Computation and Proof, Essays dedicated to Jean-Pierre Jouannaud on the occasion of his 60th birthday*, Lectures Notes in Computer Science 4600, Springer, 2007, pp. 132-166.
17. R. Cauderlier. *Traits orients objet en $\lambda\Pi$-calcul modulo : Compilation de FoCaLize vers Dedukti.* Master thesis, 2012.
18. T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76, 1988, pp. 95-120.
19. D. Cousineau. On completeness of reducibility candidates as a semantics of strong normalization. *Logical Methods in Computer Science*, 8(1), 2012.
20. D. Cousineau and G. Dowek. Embedding pure type systems in the lambda-Pi-calculus modulo. S. Ronchi Della Rocca, *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science 4583, Springer, 2007, pp. 102-117.
21. M. Crabbé. Non-normalisation de la théorie de Zermelo. Manuscript, 1974.
22. M. Crabbé. Stratification and cut-elimination. *The Journal of Symbolic Logic*, 56(1), 1991, pp. 213-226.
23. D. Delahaye, D. Doligez 2, F. Gilbert, P. Halmagrand, O. Hermant. Proof Certication in Zenon Modulo: When Achilles Uses Deduction Modulo to Outrun the Tortoise with Shorter Steps. *International Workshop on the Implementation of Logics*, 2013.
24. A. Díaz-Caro and G. Dowek. Simply Typed Lambda-Calculus Modulo Type Isomorphism. Manuscript, 2014.
25. G. Dowek. About folding-unfolding cuts and cuts modulo. *Journal of Logic and Computation* 11(3), 2001, pp. 419-429.
26. G. Dowek. What is a theory? H. Alt, A. Ferreira (Eds.), *Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 2285, Springer, 2002, pp. 50-64.

27. G. Dowek. Confluence as a cut elimination property. R. Nieuwenhuis (Ed.), *Rewriting Technique and Applications*, Lecture Notes in Computer Science 2706, Springer, 2003, pp 2-13.
28. G. Dowek. Truth values algebras and proof normalization. *TYPES 2006*, Lectures Notes in Computer Science 4502, Springer, 2007.
29. G. Dowek. Polarized resolution modulo. *IFIP Theoretical Computer Science*, 2010.
30. G. Dowek, Th. Hardin, and C.Kirchner. HOL-lambda-sigma: an intentional first-order expression of higher-order logic. *Mathematical Structures in Computer Science*, 11, 2001, pp. 1-25.
31. G. Dowek, Th. Hardin, and C. Kirchner. Theorem proving modulo. *Journal of Automated Reasoning*, 31(1), 2003, pp. 33-72.
32. G. Dowek, O. Hermant. A Simple Proof that Super-Consistency Implies Cut Elimination. *Notre Dame Journal of Formal Logic*, 53(4), 2012, pp. 439-456.
33. G. Dowek and Y. Jiang. Pushdown systems in Polarized deduction modulo. Manuscript.
34. G. Dowek and A. Miquel. Cut elimination for Zermelo's set theory. Manuscript.
35. G. Dowek and B. Werner. Proof normalization modulo. *The Journal of Symbolic Logic*, 68(4), 2003, pp. 1289-1316.
36. G. Dowek and B. Werner. Arithmetic as a theory modulo. J. Giesel (Ed.), *Term rewriting and applications*, Lecture Notes in Computer Science 3467, Springer, 2005, pp. 423-437.
37. J. Ekman. *Normal proofs in set theory.* Doctoral thesis, Chalmers university of technology and University of Göteborg, 1994.
38. J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse et son application à l'élimination des coupures dans l'analyse et la théorie des types. J.E. Fenstad (Ed.) *Second Scandinavian Logic Symposium*, North-Holland, 1970.
39. Jianhua Gao. Clausal Presentation of Theories in Deduction Modulo. *International Workshop on Proof-Search in Axiomatic Theories and Type Theories*, 2011.
40. L. Hallnäs. *On normalization of proofs in set theory.* Doctoral thesis, University of Stockholm, 1983.
41. O. Hermant. Semantic cut elimination in the Intuitionnistic Sequent Calculus. *Typed Lambda Calculus and Applications*, Lecture Notes in Computer Science 3461, Springer, 2005, pp. 221-233.
42. O. Hermant. Resolution is cut-free. *Journal of Automated Reasoning*, 44(3), 2010, pp. 245-276.
43. R. Harper, F. Honsell, G. Plotkin. A Framework for Defining Logics. *Proceedings of Logic in Computer Science*, 1987, pp. 194-204.
44. G. Huet. A mechanisation of Type Theory. *Third International Joint Conference on Artificial Intelligence*, 1973, pp. 139-146.
45. G. Huet. A Unification Algorithm for Typed λ-calculus. *Theoretical Computer Science*, 1, 1975, pp. 27-57.
46. M. Jacquel, K. Berkani, D. Delahaye, and C. Dubois. Tableaux Modulo Theories Using Superdeduction - An Application to the Verification of B Proof Rules with the Zenon Automated Theorem Prover. IJCAR, 2012, pp. 332-338.
47. D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. J. Leech (Ed.), *Computational Problems in Abstract Algebra*, Pergamon Press, 1970, pp. 263-297.
48. S. Negri and J. Von Plato. Cut elimination in the presence of axioms. *Bulletin of Symbolic Logic*, 4(4), pp. 418-435.
49. P. Martin-Löf. *Intuitionistic type theory.* Bibliopolis, 1984.

50. A. Naibo. *Le statut dynamique des axiomes: Des preuves aux modèles.* Doctoral thesis, 2013.
51. M.H.A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43, 2, 1942, pp. 223-243.
52. B. Nordström, K. Petersson, and J.M. Smith. Martin-Löf's type theory. S. Abramsky, D. Gabbay,and T. Maibaum (eds.) *Handbook of Logic in Computer Science*, Clarendon Press, 2000, pp. 1-37.
53. G. Plotkin. Building-in equational theories. *Machine Intelligence*, 7, 1972, pp. 73-90.
54. D. Prawitz. *Natural Deduction, a Proof-theoretical Study.* 1965.
55. R.Saillard. Towards explicit rewrite rules in the -calculus modulo. *International Workshop on the Implementation of Logics*, 2013.
56. J.R. Slagle. Automatic theorem proving with renamable and semantic resolution. *J. ACM*, 14, 1967, pp. 687-697.
57. B. Wack. *Typage et déduction dans le calcul de réécriture.* Doctoral Thesis, Université Henri Poincaré Nancy 1, 2005.
58. L. Wos, G.A. Robinson, D.F. Carson. Efficiency and completeness of the set of support strategy in theorem proving. *J. ACM*, 12, 1965, pp. 536-541.