Theorem Proving Modulo Based on Boolean Equational Procedures

Camilo Rocha
José Meseguer

December, 2007

Technical Report
UIUCDCS-R-2007-2922

Formal Methods and Declarative Languages Laboratory
Department of Computer Science
University of Illinois at Urbana-Champaign
201 N Goodwin Ave
Urbana, IL 61801

The moral of my story is that if we treat our formalisms with the care and respect that we pay to our other subtle artifacts, our care and respect will be more than rewarded. Calculemus!
E.W. Dijkstra, "How computer science created a new mathematical style"
EWD1073.

Table of Contents

List of Figures

Abstract

Deduction with inference rules *modulo* computation rules plays an important role in automated deduction as an effective method for scaling up. We present four equational theories that are isomorphic to the traditional Boolean theory and show that each of them gives rise to a Boolean decision procedure based on a canonical rewrite system modulo associativity and commutativity. Then, we present two modular extensions of our decision procedure for Dijkstra-Scholten propositional logic to the Sequent Calculus for First Order Logic and to the Syllogistic Logic with Complements of L. Moss. These extensions take the form of *rewrite theories* that are sound and complete for performing *deduction modulo* their equational parts and exhibit good mechanization properties. We illustrate the practical usefulness of this approach by a direct implementation of one of these theories in the Maude rewriting logic language, and automatically proving a challenge benchmark in theorem proving.

1 Introduction

The key challenge in automated deduction is *scaling up*. For the large proof efforts involved in non-toy mathematical and system verification proofs it is essential to raise the level of abstraction, so that the person performing the proofs can delegate large chunks of the effort to automated proof assistants. This need is widely felt, and approaches to meet it take different guises, such as the growing support for decision procedures, the autarkic/skeptical distinction between proofs and computations [2], and the so-called "deduction modulo" approach [6], which, as shown by Viry [27], is very closely related to the use of rewriting logic as a logical framework [18], so that the distinction between computation and deduction is captured by the corresponding distinction between equations and rules in a rewrite theory $\mathcal{R}_{\mathcal{L}}$ formalizing the inference system of the given logic $\mathcal{L}$.

Specifically, the rewrite theory $\mathcal{R}_{\mathcal{L}}$ is a triple $\mathcal{R}_{\mathcal{L}} = (\Sigma_{\mathcal{L}}, E_{\mathcal{L}} \cup A_{\mathcal{L}}, R_{\mathcal{L}})$, where: (i) $\Sigma_{\mathcal{L}}$ is a signature describing the syntax of the logic $\mathcal{L}$; (ii) $E_{\mathcal{L}}$ is a set of confluent and terminating *equations* corresponding to those parts of the deduction process that, being deterministic, can be safely automated as *computation rules* without any proof search; and (iii) $R_{\mathcal{L}}$ is a, typically small, set of rewrite rules capturing those essentially nondeterministic aspects of logical inference in $\mathcal{L}$ which require proof search. Both the computation rules $E_{\mathcal{L}}$ and the deduction rules $R_{\mathcal{L}}$ are executed by rewriting *modulo* a set $A_{\mathcal{L}}$ of equations specifying some *structural axioms* in $\mathcal{L}$ such as, for example, the associativity and commutativity of an addition operator $+$ at the level of *terms*, or of a conjunction operator at the level of *formulas*, or the similar associativity and commutativity of the formula union operator (typically denoted with the symbol $\_, \_$) in a set of formulas $\Gamma = A_1, \ldots, A_n$ at the level of *sequents*. In a traditional inference system, *all* these tasks —now delegated to either $E_{\mathcal{L}}$, or $A_{\mathcal{L}}$, or $R_{\mathcal{L}}$— would be performed as *deduction* tasks, which gets the deduction process bogged down in endless minutiae, and misses countless opportunities of making a proof much more efficient by identifying and exploiting its computational subtasks. The point, of course, is that although both $E_{\mathcal{L}}$ and $R_{\mathcal{L}}$ are executed by rewriting, $E_{\mathcal{L}}$, being confluent and terminating, has a single outcome in the form of a so-called simplified or canonical form, and can be executed as it were "blindly," without any search, and therefore also blindingly fast and with typically modest memory requirements. Furthermore, $A_{\mathcal{L}}$ provides yet one more level of computational automation, typically in the form of $A_{\mathcal{L}}$-matching or $A_{\mathcal{L}}$-unification algorithms.

By "deduction modulo" in this context, what we then mean is that the inference rules $R_{\mathcal{L}}$ are really operating not at the level of syntactic entities as in the traditional case, but *modulo* the entire equational theory $(\Sigma_{\mathcal{L}}, E_{\mathcal{L}} \cup A_{\mathcal{L}})$, comprising both the computation rules $E_{\mathcal{L}}$ and the structural axioms $A_{\mathcal{L}}$. Therefore, one step of inference with $R_{\mathcal{L}}$ modulo $E_{\mathcal{L}} \cup A_{\mathcal{L}}$ may literally correspond to millions of inference steps in a traditional inference system for $\mathcal{L}$.

These ideas have been illustrated in detail for many logics in various papers, including, for example, various sequent calculi in [18], the "sequent calculus modulo" of G. Dowek, T. Hardin and C. Kirchner in [6], Viry's rewrite theory for the sequent calculus of first-order logic in [27], and the representation of pure type systems in rewriting logic in [25]. In this paper we concentrate our attention on what we think is an interesting instance of the deduction modulo idea that combines two obvious strengths: (i) the general power of the deduction modulo framework; and (ii) the intrinsic power of equationally-based Boolean *decision procedures* operating at the level of formulas. The idea, therefore, is that the equational theory $(\Sigma_{\mathcal{L}}, E_{\mathcal{L}} \cup A_{\mathcal{L}})$ we are reasoning modulo, includes a confluent and terminating subtheory $(\Sigma_{BOOL}, E_{BOOL} \cup A_{BOOL}) \subseteq (\Sigma_{\mathcal{L}}, E_{\mathcal{L}} \cup A_{\mathcal{L}})$, where $(\Sigma_{BOOL}, E_{BOOL} \cup A_{BOOL})$ provides a decision procedure for Boolean equivalence of formulas in $\mathcal{L}$. This can be very useful, because other equations in $E_{\mathcal{L}}$ (operating, for example at the level of sequents) or some rules in $R_{\mathcal{L}}$, may immediately take advantage of the fact that we have simplified a formula to a tautology or a falsity to finish off a whole deduction subgoal.

Specifically, in Sections 4 and 5 we discuss in detail four such equationally-based Boolean decision procedures. One is the well-known procedure due to J. Hsiang, who gave a confluent and terminating set of equations for the theory of Boolean rings *modulo* associativity and commutativity in his UIUC Ph.D. thesis [13]. The other three are, to the best of our knowledge, new. We characterize their soundness and completeness by the satisfaction of two key properties: (i) they are all isomorphic to the standard Boolean theory; and (ii) they are all confluent and terminating modulo some associativity and commutativity axioms.

In this paper we give particular attention to one of these four Boolean theories, namely, a decision procedure for the propositional fragment of the Dijkstra-Scholten logic [5]. This logic has been shown by Dijkstra and Scholten to be very useful in program correctness proofs in the Dijkstra style, and has attracted a substantial following in research, teaching and programming, including [5, 10, 1, 17]. It has the same expressive power as

standard first-order logic [17]; and includes an interesting propositional fragment [11]. However, to the best of our knowledge this logic has not yet been mechanized, and no *equational* decision procedure based on confluent and terminating equations was known for it. The obvious approach to obtain a scalable mechanization of the Dijkstra-Scholten (first-order) logic in a "deduction modulo" style is then to specify it as a rewrite theory $\mathcal{R}_{\mathrm{DS}} = (\Sigma_{\mathrm{DS}}, E_{\mathrm{DS}} \cup A_{\mathrm{DS}}, R_{\mathrm{DS}})$, where $(\Sigma_{\mathrm{DS}}, E_{\mathrm{DS}} \cup A_{\mathrm{DS}})$ includes the just-mentioned equationally based decision procedure for the Boolean equivalence of formulas. We do just that, in the form of a Dijkstra-Scholten-style sequent calculus for first-order logic that we prove sound and complete in Section 6. We also show in Section 6.1 that the rewrite theory $\mathcal{R}_{\mathrm{DS}}$ satisfies all the essential requirements for being *executable* by rewriting by showing that: (i) the equational axioms A_{DS} consist only of associativity and commutativity axioms for which A_{DS}-matching and A_{DS}-unification algorithms are readily available; (ii) the equations E_{DS}, comprising not only the equations of our decision procedure but also logical equivalences at the level of sequents, are *confluent and terminating* modulo A_{DS}, and (iii) the inference rules R_{DS} are *weakly coherent* with respect to the equations E_{DS} modulo A_{DS}, which means that we can always execute the rules in R_{DS} *after* all goals have been simplified by E_{DS} without any loss in logical completeness. In Section 6.2 we illustrate the practical usefulness of this approach by a direct implementation of the rewrite theory $\mathcal{R}_{\mathrm{DS}}$ in the Maude rewriting logic language that is able to prove automatically a challenge benchmark in theorem proving, namely, Andrews' challenge [9].

As further evidence for the power of the deduction modulo approach to theorem proving supported by rewriting logic, we summarize in Section **??** another case study developed more fully in [22], namely, a Dijkstra-Scholten-style decision procedure for the Syllogistic Logic with Complements of L. Moss [20]. In this, simpler case, no proof search is involved at all, that is, all is "computation," and there is no "deduction," so that the set of rules is empty and the entire decision procedure for this logic takes the form of an *equational* theory extending that of the equational theory for proposition Dijkstra-Scholten logic. We conclude the paper with some final remarks and a discussion of future work. For detailed proofs, complete specifications, and further discussion on the results presented in this paper regarding the propositional case, we refer the reader to the technical report [21].

2 Theories, Morphisms, and Definitional Extensions

This section gathers basic notions on equational theories, theory morphisms, and definitional extensions that are needed in some sections of this work. Although the subject matter is well-known, there are some technical points that may not be so well-known. For example, the usual notion of a theory morphisms as a signature morphism has to be qualified in two important ways: (i) "signature morphims" are generalized, so that they need not map basic operations to basic operations, but can map basic operations to *terms*; and (ii) a theory morphism is *not* a signature morphism, but instead an *equivalence class* of signature morphisms. We give also some useful results about definitional extensions that we have found very helpful in cutting down the amount of things to be checked, and that we will make use of later in the work. Although *nihil novum sub sole*, the reader may find this background section of some independent interest, besides its use in subsequent sections.

Since all the Boolean theories we shall consider are unsorted, we give the whole treatment in the, simpler, unsorted setting. All ideas, however, extend naturally to typed settings. A *signature* Σ, therefore, is a countable family of sets of function symbols $\Sigma = \{\Sigma_n\}_{n\in\mathbb{N}}$. An equational *theory*[1] is a pair (Σ, E), with Σ a signature, and E a set of Σ-equations, that is, formal equalities of the form $t = t'$, with $t, t' \in T_\Sigma(X)$, where $T_\Sigma(X)$ denotes the free Σ-algebra on the set X of variables, which we assume throughout to be the countable set $X = \{x_n \mid n \in \mathbb{N} \ \wedge \ n > 0\}$. An equational theory (Σ, E) defines the full subcategory $\mathbf{Alg}_{(\Sigma,E)}$ of the category $\mathbf{Alg}_\Sigma$ of all Σ-algebras determined by all those algebras that satisfy the equations E.

Definition 1. *A* signature morphism $H : \Sigma \longrightarrow \Sigma'$ *is an assignment, for each* $n \in \mathbb{N}$*, to each* $f \in \Sigma_n$ *of a term* $H(f) \in T_{\Sigma'}(X)$ *with* $Vars(H(f)) \subseteq \{x_1, \ldots, x_n\}$*, where* $Vars(t)$ *denotes the set of variables occurring in term* t*.*

Note that H gives as a "view" of each Σ'-algebra A as a Σ-algebra $A|_H$, just by interpreting on A each operation $f \in \Sigma_n$ by means of the "derived operation" $H(f)$, where there is no ambiguity about the order

[1] More precisely, this should be called a theory *presentation*. However, since the notion of isomorphism we define will make isomorphic not only all equivalent presentations for the same Σ, but also all equivalent presentations with different signatures, this *abus de langage* will hardly matter.

of the arguments thanks to the linear order in X. Indeed, any signature morphism defines a functor $\_|_H : \mathbf{Alg}_{\Sigma'} \longrightarrow \mathbf{Alg}_{\Sigma}$.

Definition 2. *Given signature morphisms $H : \Sigma \longrightarrow \Sigma'$ and $G : \Sigma' \longrightarrow \Sigma''$ we can compose them to obtain a signature morphism $G \circ H : \Sigma \longrightarrow \Sigma''$ as follows: for each $f \in \Sigma_n$ we have $(G \circ H)(f) = \widehat{G}(H(f))$, where $\widehat{G} : T_{\Sigma'}(X) \longrightarrow T_{\Sigma''}(X)|_G$ denotes the unique Σ'-homomorphism leaving the variables X unchanged.*

It is then easy to check that composition is associative, the assignment $f \mapsto f(x_1, \ldots, x_n)$ is the identity morphism for Σ, and we have a category **Sign** of signatures and signature morphisms.

Definition 3. *A* pre-theory morphism *$H : (\Sigma, E) \longrightarrow (\Sigma', E')$ is a signature morphism $H : \Sigma \longrightarrow \Sigma'$ such that $E' \vdash \widehat{H}(E)$, where $\vdash$ denotes the equational provability relation, and $\widehat{H}$ is extended to equations in the obvious way. We say that two pre-theory morphisms $H, H' : (\Sigma, E) \longrightarrow (\Sigma', E')$ are* equivalent*, denoted $H \equiv H'$, iff for each $n \in \mathbb{N}$, and each $f \in \Sigma_n$ we have, $E' \vdash H(f) = H'(f)$. It is easy to check that this is indeed an equivalence relation. We denote each equivalence class by $[H]$, and call such an equivalence class a* theory morphism *from (Σ, E) to (Σ', E').*

Such equivalence relation is clearly a *congruence* for composition of pre-theory morphisms as signature morphisms. That is, if we have pre-theory morphisms $H, H' : (\Sigma, E) \longrightarrow (\Sigma', E')$, and $G, G' : (\Sigma', E') \longrightarrow (\Sigma'', E'')$, with $H \equiv H'$, and $G \equiv G'$, then $G \circ H \equiv G' \circ H'$. Therefore, we can *compose* theory morphisms by the rule, $[G] \circ [H] = [G \circ H]$. This defines a category **Th**, with theories as *objects* and theory morphisms as *morphisms*. It can, furthermore, be shown that **Th** is equivalent to the category of Lawvere theories [16]. We will be particularly interested in theory *isomorphisms*, so it may be worthwhile to "unpack" what they are. $[H] : (\Sigma, E) \longrightarrow (\Sigma', E')$ will be an isomorphism iff there is a $[H^{-1}] : (\Sigma', E') \longrightarrow (\Sigma, E)$ such that $[H^{-1}] \circ [H] = 1_{(\Sigma,E)}$ and $[H] \circ [H^{-1}] = 1_{(\Sigma',E')}$; iff for each $n \in \mathbb{N}$ and each $f \in \Sigma_n$ and $f' \in \Sigma'_n$ we have: (i) $E \vdash f(x_1, \ldots, x_n) = (H^{-1} \circ H)(f)$; and (ii) $E' \vdash f'(x_1, \ldots, x_n) = (H \circ H^{-1})(f')$.

One important, model-theoretic point to notice is that a theory morphism $[H] : (\Sigma, E) \longrightarrow (\Sigma', E')$ induces a functor $\_|_{[H]} : \mathbf{Alg}_{(\Sigma',E')} \longrightarrow \mathbf{Alg}_{(\Sigma,E)}$, which is just the restriction of the functor $\_|_H : \mathbf{Alg}_{\Sigma'} \longrightarrow \mathbf{Alg}_{\Sigma}$. Furthermore, since the assignment $[H] \mapsto \_|_{[H]}$ is itself a contravariant functor (see [16]), if $[H]$ is a theory isomorphism, then $\_|_{[H]} :$

$\mathbf{Alg}_{(\Sigma',E')} \longrightarrow \mathbf{Alg}_{(\Sigma,E)}$ is an isomorphism of categories that preserves the sets and functions underlying the algebras an homomorphisms. Lawvere's beautiful *Structure-Semantics Adjointness Theorem* [16] proves also the opposite direction: any isomorphism of categories $\alpha : \mathbf{Alg}_{(\Sigma',E')} \cong \mathbf{Alg}_{(\Sigma,E)}$ that preserves the sets and functions underlying the algebras an homomorphisms is of the form $\alpha = \_|_{[H]}$ for some theory isomorphism $[H]$.

Therefore, theory isomorphism give us a *presentation-independent* view of axiomatic classes of algebras. For example, the theory of groups can be presented with many different signatures and sets of axioms. What all these presentations have in common is precisely that they are *isomorphic* theories in the precise sense defined above. In this work we will be interested in theory isomorphisms for the theory of Boolean algebra. A paradigmatic example of a theory isomorphism in this case, in fact one of the isomorphisms we shall consider, is the Stone isomorphism between the theory of Boolean algebras and that of Boolean *rings*.

2.1 Definitional Extensions

Given two signatures Σ and Σ', we define their union $\Sigma \cup \Sigma'$, resp. intersection $\Sigma \cap \Sigma'$, resp. difference $\Sigma - \Sigma'$, in the obvious way: for each $n \in \mathbb{N}$ $(\Sigma \cup \Sigma')_n = \Sigma_n \cup \Sigma'_n$, $(\Sigma \cap \Sigma')_n = \Sigma_n \cap \Sigma'_n$, and $(\Sigma - \Sigma')_n = \Sigma_n - \Sigma'_n$.

Definition 4. *We call a theory morphism $[H] : (\Sigma, E) \longrightarrow (\Sigma', E')$* unambiguous *iff $[H]$ restricted to $\Sigma \cap \Sigma'$ is the identity.*

This captures the intuitive, and frequently occurring situation where $[H]$ does not change the meaning of *shared symbols*: only the function symbols that Σ does not share with Σ' are given a new interpretation by $[H]$.

Lemma 1. *If $(\Sigma, E) \xrightarrow{[H]} (\Sigma', E') \xrightarrow{[G]} (\Sigma'', E'')$ are unambiguous theory morphisms such that $\Sigma \cap \Sigma'' \subseteq \Sigma'$, then $[G \circ H]$ is unambiguous.*

Proof. Let $f \in \Sigma$. If $f \notin \Sigma \cap \Sigma''$, then the Lemma trivially holds. Assume $f \in \Sigma \cap \Sigma''$. Since $\Sigma \cap \Sigma'' \subseteq \Sigma'$, $f \in \Sigma \cap \Sigma' \cap \Sigma''$. Moreover, because $[H]$ and $[G]$ are unambiguous, $[G]([H](f)) = [G](f) = f$. Hence, $[G] \circ [H] = [G \circ H]$ is unambiguous.

Definition 5. *Given an unambiguous theory morphism $[H] : (\Sigma, E) \longrightarrow (\Sigma', E')$, the* definitional extension *of (Σ', E')* along *$[H]$, denoted $(\Sigma', E')^{[H]}$, is the theory $(\Sigma', E')^{[H]} = (\Sigma \cup \Sigma', E' \cup \Delta_{[H]})$, where $\Delta_{[H]} = \{f(x_1, \ldots, x_n) =$*

$H(f) \mid f \in \Sigma - \Sigma'\}$. *It is trivial to check that the obvious identity inclusion* $(\Sigma', E') \hookrightarrow (\Sigma', E')^{[H]}$ *is a theory isomorphism with inverse the identity on* Σ'*, and mapping each* $f \in \Sigma - \Sigma'$ *to* $H(f)$.

If we have two unambiguous theory morphisms

$$(\Sigma, E) \xrightarrow{[H]} (\Sigma', E') \xrightarrow{[G]} (\Sigma'', E'')$$

such that $\Sigma \cap \Sigma'' \subseteq \Sigma'$, then it is easy to prove using the above lemma that we can iterate the definitional extension process to form a "tower" of definitional extensions

$$(\Sigma'', E'') \hookrightarrow (\Sigma'', E'')^{[G]} \hookrightarrow (\Sigma', E')^{[G]\otimes[H]}$$

where $(\Sigma', E')^{[G]\otimes[H]} = (\Sigma \cup \Sigma' \cup \Sigma'', E'' \cup \Delta_{[G]} \cup \widehat{G}(\Delta_{[H]}))$. In particular we get in this way a theory isomorphism $(\Sigma'', E'') \cong (\Sigma', E')^{[G]\otimes[H]}$. This construction will be technically useful for the Boolean decision procedures we will present, based on theory isomorphisms; because it will automatically justify the correctness of each decision procedure simultaneously supporting *all* the Boolean operations of the different signatures involved.

3 Rewrite Theories and Coherence

The reason why rewriting logic directly captures the "theorem proving modulo" idea is that, given a rewrite theory of the form $\mathcal{R} = (\Sigma, E \cup A, R)$, where A is a set of "structural" equational axioms (typically associativity and/or commutativity and/or identity) such that there exists a *matching algorithm modulo* A producing a finite number of A-matching substitutions, or failing otherwise, then rewriting with rules R in $\mathcal{R}$ takes place *modulo* $E \cup A$. For example, if $\mathcal{R} = \mathcal{R}_{\mathcal{L}}$ is the rewrite theory of a sequent calculus for $\mathcal{L}$, a sequent is a term t, but the rules R in $\mathcal{R}$ do not rewrite just sequents: they rewrite $E \cup A$-equivalence class $[t]_{E\cup A}$ in the free algebra on variables X modulo $E \cup A$, denoted $T_{\Sigma/E\cup A}(X)$. More precisely, we have a one-step rewrite $[t]_{E\cup A} \longrightarrow_{\mathcal{R}} [t']_{E\cup A}$ in $\mathcal{R}$ iff we can find a term $u \in [t]_{E\cup A}$ such that u can be rewritten to v using some rule $l : q \longrightarrow r$ in R in the standard way[2], denoted $u \longrightarrow_R v$, and we furthermore have $v \in [t']_{E\cup A}$. The problem is that for arbitrary E and R,

[2] See [4] for basic notation on term rewriting. Positions in a term are denoted as strings of nonzero natural numbers and represent tree positions. $t|_p$ denotes the subterm of term t at position p, and $t[u]_p$ denotes the term obtained by replacing $t|_p$ by u at position p. The notation $u \longrightarrow_R v$ means that there is a rule $l : q \longrightarrow r$ in R, a position p in u, and a substitution θ such that $u|_p = \theta(q)$ and $v = u[\theta(r)]_p$.

whether $[t]_{E \cup A} \longrightarrow_{\mathcal{R}} [t']_{E \cup A}$ holds is in general *undecidable*, even when the equations E are confluent and terminating modulo A. Therefore, the most useful rewrite theories satisfy additional executability conditions, explained below, under which we can reduce the relation $[t]_{E \cup A} \longrightarrow_{\mathcal{R}} [t']_{E \cup A}$ to simpler forms of rewriting just modulo A, where both equality modulo A and matching modulo A are decidable.

The first condition is that E should be *ground confluent and terminating modulo* A [4]. This means that in the rewrite theory $\mathcal{R}_{E/A} = (\Sigma, A, E)$: (i) all rewrite sequences terminate, that is, there are no infinite sequences of the form

$$[t_1]_A \longrightarrow_{\mathcal{R}_{E/A}} [t_2]_A \cdots [t_n]_A \longrightarrow_{\mathcal{R}_{E/A}} [t_{n+1}]_A \cdots$$

and (ii) for each $[t]_A \in T_{\Sigma/A}$ there is a *unique* A-equivalence class $[\mathsf{can}_{E/A}(t)]_A \in T_{\Sigma/A}$ called the E-*canonical form* of $[t]_A$ modulo A such that there exists a terminating sequence of zero, one, or more steps $[t]_A \longrightarrow^*_{\mathcal{R}_{E/A}} [\mathsf{can}_{E/A}(t)]_A$.

The second condition is that the rules R should be *coherent* relative to the equations E modulo A [27]. This precisely means that, if we decompose the rewrite theory $\mathcal{R} = (\Sigma, E \cup A, R)$ into the simpler theories $\mathcal{R}_{E/A} = (\Sigma, A, E)$ and $\mathcal{R}_{R/A} = (\Sigma, A, R)$ (which have decidable rewrite relations $\longrightarrow_{\mathcal{R}_{E/A}}$ and $\longrightarrow_{\mathcal{R}_{R/A}}$ because of the assumptions on A), then for each A-equivalence class $[t]_A$ such that $[t]_A \longrightarrow_{\mathcal{R}_{R/A}} [t']_A$ we can always find a corresponding rewrite $[\mathsf{can}_{E/A}(t)]_A \longrightarrow_{\mathcal{R}_{R/A}} [t'']_A$ such that $[\mathsf{can}_{E/A}(t')]_A = [\mathsf{can}_{E/A}(t'')]_A$.

Intuitively, coherence means that we can always adopt the strategy of first simplifying a term to canonical form with E modulo A, and then apply a rule with R modulo A to achieve the effect of rewriting with R modulo $E \cup A$. The coherence condition can be relaxed to *weak coherence* of R relative to the equations E modulo A [27], where we just require that whenever $[t]_A \longrightarrow_{\mathcal{R}_{R/A}} [t']_A$ we can always find a sequence of zero, one or more rewrites $[\mathsf{can}_{E/A}(t)]_A \longrightarrow^*_{\mathcal{R}_{R \cup E/A}} [t'']_A$ such that $[\mathsf{can}_{E/A}(t')]_A = [\mathsf{can}_{E/A}(t'')]_A$.

When formalizing a logic $\mathcal{L}$ as a rewrite theory $\mathcal{R}_{\mathcal{L}}$ one has two different options (backwards or forwards) for expressing an inference rule as a rewrite rule. We will adopt the backwards reasoning option, which rewrites the goal one wants to prove to its premise subgoals. For example, a sequent rule for disjunction

$$\frac{\Gamma, B \vdash \Delta \qquad \Gamma, C \vdash \Delta}{\Gamma, B \vee C \vdash \Delta}$$

will be expressed as a rewrite rule $\Gamma, B \vee C \vdash \Delta \longrightarrow \Gamma, B \vdash \Delta \bullet \Gamma, C \vdash \Delta$, where $\bullet$ is an associative commutative operator denoting set union of sequents.

4 Five Isomorphic Boolean Theories

In this section we present five isomorphic equational theories, one of them the traditional Boolean theory. We structure each of these theories in the form $(\Sigma, E \cup A)$, with A some associativity and commutativity axioms.

The axiomatization of the traditional Boolean theory T_{BOOL} is that of a complemented distributive lattice.

Definition 6. *The equational theory* $T_{\mathrm{BOOL}} = (\Sigma_{\mathrm{BOOL}}, E_{\mathrm{BOOL}} \cup A_{\mathrm{BOOL}})$ *is given by:*

$$
\begin{aligned}
\Sigma_{\mathrm{BOOL}} &= \{\mathsf{T}^{(0)}, \mathsf{F}^{(0)}, \neg^{(1)}, \wedge^{(2)}, \vee^{(2)}\} \\
A_{\mathrm{BOOL}} &= \{P \wedge (Q \wedge R) = (P \wedge Q) \wedge R \ , \ P \wedge Q = Q \wedge P \ , \\
&\qquad P \vee (Q \vee R) = (P \vee Q) \vee R \ , \ P \vee Q = Q \vee P \} \\
E_{\mathrm{BOOL}} &= \{P \wedge P = P \ , \ P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R) \ , \\
&\qquad P \vee P = P \ , \ P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R) \ , \\
&\qquad P \wedge (P \vee Q) = P \ , \ P \vee (P \wedge Q) = P \ , \\
&\qquad P \wedge \neg P = \mathsf{F} \ , \ P \vee \neg P = \mathsf{T} \ \} \ .
\end{aligned}
$$

The axioms in A_{BOOL} express the associativity and commutativity properties (AC) of the binary operators in Σ_{BOOL}. The set of equations E_{BOOL} define both $\wedge$ and $\vee$ to be idempotent, to distribute over each other and to follow the absorption laws. The last two equations in E_{BOOL} are the well-known laws of complements, the first being the definition of contradiction and the second that of the excluded middle.

We introduce the remaining four equational theories, namely, T_{DS}, T_{BR}, $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$, respectively. The theory T_{DS} is our axiomatization as a set of confluent and terminating equations modulo AC of the Dijkstra-Scholten propositional logic [5]. The theory T_{BR} is the theory of Boolean rings and is based on the isomorphism between Boolean algebras and Boolean rings discovered by M. H. Stone [14, 23]. As a rewrite system, T_{BR} was proposed by J. Hsiang [13] in the 1980's as a decision procedure for propositional logic. We are not aware of earlier equational presentations of $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$, so we use their main function symbols as acronyms.

Definition 7. *The equational theories* $T_{\mathrm{DS}} = (\Sigma_{\mathrm{DS}}, E_{\mathrm{DS}} \cup A_{\mathrm{DS}})$, $T_{\mathrm{BR}} = (\Sigma_{\mathrm{BR}}, E_{\mathrm{BR}} \cup A_{\mathrm{BR}})$, $T_{\wedge/\equiv} = (\Sigma_{\wedge/\equiv}, E_{\wedge/\equiv} \cup A_{\wedge/\equiv})$ *and* $T_{\vee/\oplus} = (\Sigma_{\vee/\oplus}, E_{\vee/\oplus} \cup$

$A_{\vee/\oplus}$) *are defined as follows:*

$$
\begin{aligned}
\Sigma_{\mathrm{DS}} &= \{\mathsf{T}^{(0)}, \mathsf{F}^{(0)}, \vee^{(2)}, \equiv^{(2)}\} \\
A_{\mathrm{DS}} &= \{P \equiv (Q \equiv R) = (P \equiv Q) \equiv R\,,\ P \equiv Q = Q \equiv P \\
&\qquad P \vee (Q \vee R) = (P \vee Q) \vee R\,,\ P \vee Q = Q \vee P\} \\
E_{\mathrm{DS}} &= \{P \equiv \mathsf{T} = P\,,\ P \equiv P = \mathsf{T}\,,\ P \vee \mathsf{T} = \mathsf{T}\,,\ P \vee \mathsf{F} = P\,,\ P \vee P = P \\
&\qquad P \vee (Q \equiv R) = (P \vee Q) \equiv (P \vee R)\}\,,
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_{\mathrm{BR}} &= \{\mathsf{T}^{(0)}, \mathsf{F}^{(0)}, \wedge^{(2)}, \oplus^{(2)}\} \\
A_{\mathrm{BR}} &= \{P \oplus (Q \oplus R) = (P \oplus Q) \oplus R\,,\ P \oplus Q = Q \oplus P\} \\
&\qquad P \wedge (Q \wedge R) = (P \wedge Q) \wedge R\,,\ P \wedge Q = Q \wedge P\} \\
E_{\mathrm{BR}} &= \{P \oplus \mathsf{F} = P\,,\ P \oplus P = \mathsf{F}\,,\ P \wedge \mathsf{F} = \mathsf{F}\,,\ P \wedge \mathsf{T} = P\,,\ P \wedge P = P \\
&\qquad P \wedge (Q \oplus R) = (P \wedge Q) \oplus (P \wedge R)\}\,,
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_{\wedge/\equiv} &= \{\mathsf{T}^{(0)}, \mathsf{F}^{(0)}, \wedge^{(2)}, \equiv^{(2)}\} \\
A_{\wedge/\equiv} &= \{P \equiv (Q \equiv R) = (P \equiv Q) \equiv R\,,\ P \equiv Q = Q \equiv P \\
&\qquad P \wedge (Q \wedge R) = (P \wedge Q) \wedge R\,,\ P \wedge Q = Q \wedge P\} \\
E_{\wedge/\equiv} &= \{P \equiv \mathsf{T} = P\,,\ P \equiv P = \mathsf{T}\,,\ P \wedge \mathsf{T} = P\,,\ P \wedge \mathsf{F} = \mathsf{F}\,,\ P \wedge P = P \\
&\qquad P \wedge (Q \equiv R) = (P \wedge Q) \equiv (P \wedge R) \equiv P\}\,,
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_{\vee/\oplus} &= \{\mathsf{T}^{(0)}, \mathsf{F}^{(0)}, \vee^{(2)}, \oplus^{(2)}\} \\
A_{\vee/\oplus} &= \{P \oplus (Q \oplus R) = (P \oplus Q) \oplus R\,,\ P \oplus Q = Q \oplus P \\
&\qquad P \vee (Q \vee R) = (P \vee Q) \vee R\,,\ P \vee Q = Q \vee P\} \\
E_{\vee/\oplus} &= \{P \oplus \mathsf{F} = P\,,\ P \oplus P = \mathsf{F}\,,\ P \vee \mathsf{T} = \mathsf{T}\,,\ P \vee \mathsf{F} = \mathsf{F}\,,\ P \vee P = P \\
&\qquad P \vee (Q \oplus R) = (P \vee Q) \oplus (P \vee R) \oplus P\}\,.
\end{aligned}
$$

The function symbols $\equiv$ and $\oplus$ denote equivalence and discrepancy, respectively, and have less binding power than any other function symbol. Both symbols are associative and commutative in the theories where they are defined. The other function symbols correspond to those of Σ_{BOOL}; we have chosen not to change their notation in order to keep the definitions and proofs as compact as possible. The symbol $\oplus$ is sometimes denoted by $\not\equiv$ and it is known as either the symmetric difference operator in algebra or as the exclusive or operator in switching theory.

To show that the theories T_{BOOL}, T_{DS}, T_{BR}, $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$ are all isomorphic, we have to make precise the notion of equational *theory isomorphism*, and more generally, that of *theory morphism* in the category **Th** of equational theories. Appendix 2 gives the precise definition. Here we just summarize the basic idea by pointing out that a theory morphism

$H : (\Sigma, E) \longrightarrow (\Sigma', E')$ maps each $f \in \Sigma_n$ to a Σ'-term with n variables and satisfies the property that if $u = v \in E$, then $E' \vdash H(u) = H(v)$.

Definition 8. *The nine morphisms appearing in Fig. 1 are defined as follows:*

- *G maps identically T, F and $\vee$. For $\neg$ and $\wedge$ we have: $G(\neg P) = P \equiv \mathsf{F}$ and $G(P \wedge Q) = P \equiv Q \equiv P \vee Q$.*
- *G^{-1} maps identically T, F and $\vee$. For $\equiv$ we have $G^{-1}(P \equiv Q) = (P \vee \neg Q) \wedge (\neg P \vee Q)$.*
- *H maps identically T, F and $\wedge$. For $\neg$ and $\vee$ we have: $H(\neg P) = P \oplus \mathsf{T}$ and $H(P \vee Q) = P \oplus Q \oplus P \wedge Q$.*
- *H^{-1} maps identically T, F and $\wedge$. For $\oplus$ we have $H^{-1}(P \oplus Q) = (P \vee Q) \wedge (\neg P \vee \neg Q)$.*
- *K maps identically T, F and $\wedge$. For $\neg$ and $\vee$ we have: $K(\neg P) = P \equiv \mathsf{F}$ and $K(P \vee Q) = P \equiv Q \equiv P \wedge Q$.*
- *K^{-1} maps identically T, F and $\wedge$. For $\equiv$ we have $K^{-1}(P \equiv Q) = (P \wedge Q) \vee (\neg P \wedge \neg Q)$.*
- *L maps identically T, F and $\vee$. For $\neg$ and $\wedge$ we have: $L(\neg P) = P \oplus \mathsf{T}$ and $L(P \wedge Q) = P \oplus Q \oplus P \vee Q$.*
- *L^{-1} maps identically T, F and $\vee$. For $\oplus$ we have $L^{-1}(P \oplus Q) = (P \wedge \neg Q) \vee (\neg P \wedge Q)$.*
- *op is the duality morphism for Boolean algebras, mapping T to F, F to T, $\neg$ to $\neg$, $\wedge$ to $\vee$ and $\vee$ to $\wedge$.*

Theorem 1. *The morphisms op, G, H, K and L are theory isomorphisms between the corresponding theories.*

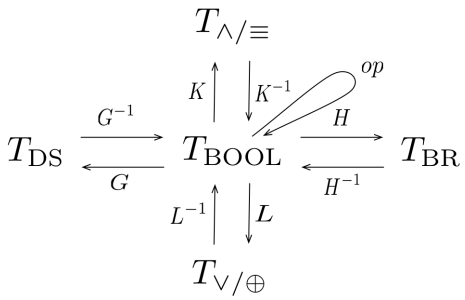

Fig. 1. Isomorphisms between the Boolean theory and the other four theories.

We call these isomorphisms *Boolean isomorphisms*. They give rise to new Boolean isomorphisms by composition among them.

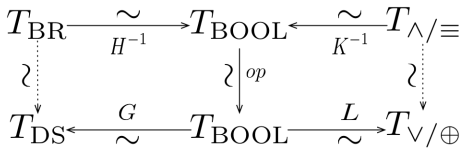

Fig. 2. Commutation and composition of Boolean isomorphisms.

Figure 2 highlights two particular ones, namely $G \circ op \circ H^{-1}$ and $L \circ op \circ K^{-1}$, which show that the theories T_{BR} and T_{DS}, and the theories $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$ are pairs of *dual theories*. These morphisms are used in the next section to build decision procedures for propositional logic by term rewriting using the Boolean theories T_{DS}, T_{BR}, $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$.

5 Four Equational Decision Procedures

In this section we explain in more detail a decision procedure for propositional logic for the equational theory T_{DS}. The exact same construction applies to T_{BR} (where it is well-known since [13]), to $T_{\wedge/\equiv}$ and to $T_{\vee/\oplus}$. The complete set of four decision procedures for propositional logic we have studied using this approach, each containing equations for all other Boolean connectives as definitional extensions, can be found in [21].

Theorem 2. *The equations E_{DS} in T_{DS} are confluent and terminating modulo A_{DS}. Similarly, the equations in $E_{\wedge/\equiv}$ and $E_{\vee/\oplus}$, in $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$, are confluent and terminating modulo $A_{\wedge/\equiv}$ and $A_{\vee/\oplus}$, respectively.*

We focus on T_{DS} and refer to [21] for $T_{\wedge/\equiv}$ and $T_{\vee/\oplus}$. Termination and confluence modulo A_{DS} can be established mechanically by using formal tools that: (i) find a well-founded ordering $\succ$ on A_{DS}-equivalence classes of terms such that $[t]_{A_{\mathrm{DS}}} \rightarrow_{E_{\mathrm{DS}}/A_{\mathrm{DS}}} [t']_{A_{\mathrm{DS}}}$ implies $[t]_{A_{\mathrm{DS}}} \succ [t']_{A_{\mathrm{DS}}}$, and (ii) check confluence of E_{DS} modulo A_{DS} by computing all so-called "critical-pairs" modulo A_{DS} and showing they are all confluent. We have used the CiME tool [15] to check termination and confluence of E_{DS} modulo A_{DS}. Furthermore, it can be shown using Maude's Sufficient Completeness Checker [12] that the canonical form of any term is either T, F or $t_0 \equiv \ldots \equiv t_n$, where all t_i are distinct disjunctions (modulo AC) of propositional variables (see [21] for the sufficient completeness proof).

As a consequence, we can use T_{DS} as a *decision procedure for propositional logic*. That is, we have the following equivalences for any propositional expressions t and t':

$$T_{\mathrm{DS}} \vdash t = t' \;\Leftrightarrow\; T_{\mathrm{DS}} \vdash t \equiv t' = \mathsf{T} \;\Leftrightarrow\; \mathsf{can}_{E_{\mathrm{DS}}/A_{\mathrm{DS}}}[t] = \mathsf{can}_{E_{\mathrm{DS}}/A_{\mathrm{DS}}}[t'] \;.$$

In particular, since T and F are both in $E_{\mathrm{DS}}/A_{\mathrm{DS}}$-canonical form, we have:

$$T_{\mathrm{DS}} \vdash t \equiv t' = \mathsf{T} \;\Leftrightarrow\; \mathsf{can}_{E_{\mathrm{DS}}/A_{\mathrm{DS}}}[t \equiv t'] = [\mathsf{T}]$$

and

$$T_{\mathrm{DS}} \vdash t \equiv t' = \mathsf{F} \;\Leftrightarrow\; \mathsf{can}_{E_{\mathrm{DS}}/A_{\mathrm{DS}}}[t \equiv t'] = [\mathsf{F}].$$

We call a proposition t a *tautology* iff $\mathsf{can}_{E_{\mathrm{DS}}/A_{\mathrm{DS}}}[t] = [\mathsf{T}]$ and a *falsity* iff $\mathsf{can}_{E_{\mathrm{DS}}/A_{\mathrm{DS}}}[t] = [\mathsf{F}]$. We call t *satisfiable* iff $\mathsf{can}_{E_{\mathrm{DS}}/A_{\mathrm{DS}}}[t] \neq [\mathsf{F}]$. Therefore, our decision procedure gives also a decision procedure for checking satisfiability of any proposition t.

6 A Rewriting Modulo View of the Sequent Calculus

We present a rewrite theory $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}} = (\Sigma^{\mathrm{DS}}_{\mathrm{SEQ}}, E^{\mathrm{DS}}_{\mathrm{SEQ}} \cup A^{\mathrm{DS}}_{\mathrm{SEQ}}, R^{\mathrm{DS}}_{\mathrm{SEQ}})$ modular with respect to the equational theory T_{DS}, directly inspired by the definition of the sequent calculus in [24, 8]. A rewrite theory $\mathcal{R}^{\mathrm{BOOL}}_{\mathrm{SEQ}}$ for the sequent calculus based on the traditional connectives $\vee$, $\wedge$ and $\neg$ has been previously presented by P. Viry [26, 27]. Although Viry's equations for the formula part are executable, they fall short of being a decision procedure for Boolean equivalence of formulas. Therefore his $\mathcal{R}^{\mathrm{BOOL}}_{\mathrm{SEQ}}$ seems to have somewhat limited power in his "modulo" part. By contrast, our approach, by including E_{DS} in $E^{\mathrm{DS}}_{\mathrm{SEQ}}$, and A_{DS} in $A^{\mathrm{DS}}_{\mathrm{SEQ}}$, besides being readily implementable as we explain in Sections 6.1 and 6.2, has substantially more inference power in its modulo part, since any first-order formula that is a tautology or a falsity based on its Boolean structure will be automatically reduced to T or F by the E_{DS} equations, and this can be then used by the remaining equations in $E^{\mathrm{DS}}_{\mathrm{SEQ}}$ to automatically prove some sequents. We furthermore show in Section 6.2 the practical usefulness of our approach by reporting on experiments with an implementation of $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ in the Maude rewriting logic language.

We focus on $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ because, although first-order logic reasoning based on the Dijkstra-Scholten axiomatization has been extensively used in teaching, programming and research (see for instance [5, 10, 1, 17]), to the best of our knowledge no mechanization of Dijkstra-Scholten-style first order logic reasoning has been developed so far, so that the implementation of $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ is the first such mechanization we are aware of. However, for users interested in reasoning based on the connectives of T_{BR}, $T_{\wedge/\equiv}$

or $T_{\vee/\oplus}$, the same approach we present here for $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ can be developed in rewrite theories $\mathcal{R}^{\mathrm{BR}}_{\mathrm{SEQ}}$, $\mathcal{R}^{\wedge/\equiv}_{\mathrm{SEQ}}$ and $\mathcal{R}^{\vee/\oplus}_{\mathrm{SEQ}}$, and with the same rewriting modulo advantages.

The order-sorted signature $\Sigma^{\mathrm{DS}}_{\mathrm{SEQ}}$ that we will use for representing terms of the sequent calculus is that of figure 3:

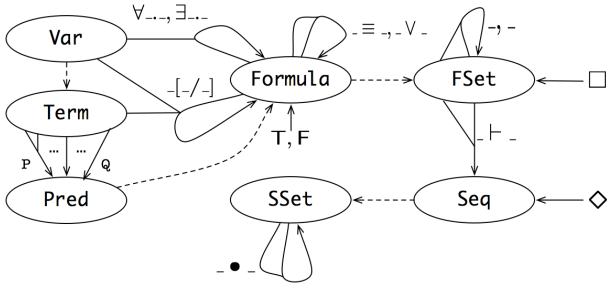

Fig. 3. Signature $\Sigma^{\mathrm{DS}}_{\mathrm{SEQ}}$ of the theory $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$.

The sort Formula corresponds to first-order formulas built from the constants T and F, the binary operators $\equiv$ and $\vee$, and universal and existential quantification. The atomic building blocks for formulas are predicates of sort Pred ranging over first order terms Term, and constructed by predicate symbols P, Q, *etc.* of different arities. The sort Var corresponds to names of bound variables. The operator $\_[\_/\_]$ stands for explicit substitution of a variable by a term in a formula. The sort FSet corresponds to sets of formulas, with the constant $\Box$ denoting the empty set of formulas. The sorts Seq and SSet represent first-order sequents and sets of first-order sequents, respectively. We denote the trivial sequent with the constant symbol $\diamond$. Dashed lines represent sort inclusions.

In the rest of this section we use the variables $B, C, \ldots$, to represent formulas, $\Gamma, \Delta, \ldots$, to represent sets of formulas, $S, S', \ldots$ to represent sequents, and $SS, SS', \ldots$, to represent sets of sequents.

Definition 9. *The rewrite theory* $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}} = (\Sigma^{\mathrm{DS}}_{\mathrm{SEQ}}, E^{\mathrm{DS}}_{\mathrm{SEQ}} \cup A^{\mathrm{DS}}_{\mathrm{SEQ}}, R^{\mathrm{DS}}_{\mathrm{SEQ}})$ *is defined as follows:*

$$
\begin{aligned}
A^{\mathrm{DS}}_{\mathrm{SEQ}} \quad &= A_{\mathrm{FORM}} \cup \{\Gamma, (\Delta, \Pi) = (\Gamma, \Delta), \Pi \ ,\ \Gamma, \Delta = \Delta, \Gamma \ ,\ \Gamma, \Box = \Gamma \ , \\
&\qquad SS \bullet (SS' \bullet SS'') = (SS \bullet SS') \bullet SS'' \ ,\ SS \bullet SS' = SS' \bullet SS \ , \\
&\qquad SS \bullet \diamond = SS \ \}
\end{aligned}
$$

$$
\begin{aligned}
E^{\mathrm{DS}}_{\mathrm{SEQ}} \quad &= E_{\mathrm{SUBS}} \cup E_{\mathrm{FORM}} \cup \{\forall x.\mathsf{T} = \mathsf{T}\ ,\ \forall x.\mathsf{F} = \mathsf{F}\ ,\ \exists x.B = \forall x.(B \equiv \mathsf{F}) \equiv \mathsf{F}\ , \\
&\quad \Gamma, \mathsf{F} \vdash \Delta = \diamond\ ,\ \Gamma \vdash \mathsf{T}, \Delta = \diamond\ ,\ \Gamma, \mathsf{T} \vdash \Delta = \Gamma \vdash \Delta\ , \\
&\quad \Gamma \vdash \mathsf{F}, \Delta = \Gamma \vdash \Delta\ , \Gamma, \Gamma = \Gamma\ ,\ (\Gamma \vdash \Delta) \bullet (\Gamma \vdash \Delta) = \Gamma \vdash \Delta\ \}
\end{aligned}
$$

$$
\begin{aligned}
R^{\mathrm{DS}}_{\mathrm{SEQ}} \quad &= \{\Gamma, B \vdash B, \Delta\ \longrightarrow\ \diamond\ , \\
&\quad \Gamma, B \equiv C \vdash \Delta\ \longrightarrow\ \Gamma, B, C \vdash \Delta \bullet \Gamma \vdash B, C, \Delta\ , \\
&\quad \Gamma \vdash B \equiv C, \Delta\ \longrightarrow\ \Gamma, B \vdash C, \Delta \bullet \Gamma, C \vdash B, \Delta\ , \\
&\quad \Gamma, B \vee C \vdash, \Delta\ \longrightarrow\ \Gamma, B \vdash \Delta \bullet \Gamma, C \vdash \Delta\ , \\
&\quad \Gamma \vdash B \vee C, \Delta\ \longrightarrow\ \Gamma \vdash B, C, \Delta\ , \\
&\quad \Gamma, \forall x.B \vdash \Delta\ \longrightarrow\ \Gamma, B[t/x] \vdash \Delta\ , \\
&\quad \Gamma \vdash \forall x.B, \Delta\ \longrightarrow\ \Gamma \vdash B[y/x], \Delta\ \},
\end{aligned}
$$

where A_{FORM} *and* E_{FORM} *correspond to the equations* A_{DS} *and* E_{DS} *defined over the sort* Formula*,* E_{SUBS} *to the equations for explicit substitution,* t *is any first order term free for* x *and* y *is a variable not occurring free in* Γ, B, Δ *.*

Equations in $A^{\mathrm{DS}}_{\mathrm{SEQ}}$ specify associativity, commutativity and the existence of an identity element for sets of formulas and sequents, in addition to those equations extended from A_{DS}. New equations in $E^{\mathrm{DS}}_{\mathrm{SEQ}}$ express different well-known logical equivalences between both formulas and sequents. The rewrite rules in $R^{\mathrm{DS}}_{\mathrm{SEQ}}$ correspond to a deductively complete subset of the sequent calculus rules presented in [24].

A proof of a sequent S *modulo* $E^{\mathrm{DS}}_{\mathrm{SEQ}} \cup A^{\mathrm{DS}}_{\mathrm{SEQ}}$ is then represented as a rewriting logic proof in $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ of the form:

$$
[S]_{E^{\mathrm{DS}}_{\mathrm{SEQ}} \cup A^{\mathrm{DS}}_{\mathrm{SEQ}}} \longrightarrow^{*}_{R^{\mathrm{DS}}_{\mathrm{SEQ}}} [\diamond]_{E^{\mathrm{DS}}_{\mathrm{SEQ}} \cup A^{\mathrm{DS}}_{\mathrm{SEQ}}}\ ,
$$

which we abbreviate as $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}} \vdash S \longrightarrow^{*} \diamond$.

Theorem 3. *The rewrite theory* $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ *is sound, that is, a sequent* S *is provable in the sequent calculus if there is a derivation* $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}} \vdash S \longrightarrow^{*} \diamond$ *.*

Proof. We assume that the theory $\mathcal{R}^{\mathrm{BOOL}}_{\mathrm{SEQ}}$ is sound. For the theory $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ to be sound, it means that every one of its equations and rules can be 'mimicked' with an $\mathcal{R}^{\mathrm{BOOL}}_{\mathrm{SEQ}}$ valid rewriting sequence. The equations in $A^{\mathrm{DS}}_{\mathrm{SEQ}}$ have an identical counterpart in that system, therefore they are sound. For the equations in $E^{\mathrm{DS}}_{\mathrm{SEQ}}$ the situation is straightforward: $\forall x.\mathsf{T} = \mathsf{T}$ and $\forall x.\mathsf{F} = \mathsf{F}$ are particular cases of the rules dealing with universal quantification for constant terms; we deal with a similar situation for

the equations $\Gamma, \mathsf{F} \vdash \Delta = \diamond$, $\Gamma \vdash \mathsf{T}, \Delta = \diamond$, $\Gamma, \mathsf{T} \vdash \Delta = \Gamma \vdash \Delta$ and $\Gamma \vdash \mathsf{F}, \Delta = \Gamma \vdash \Delta$. Let us illustrate the procedure with the following case, assuming we have that $\widehat{\mathsf{T}} = 1$ and $\widehat{\mathsf{F}} = 0$, where $\widehat{t}$ is the respective translation of the term $t \in T_{\Sigma^{\mathrm{DS}}_{\mathrm{SEQ}}}(X)$ to the term $\widehat{t} \in t \in T_{\Sigma^{\mathrm{BOOL}}_{\mathrm{SEQ}}}(X)$:

$\widehat{\Gamma}, \widehat{\mathsf{F}} \vdash \widehat{\Delta}$
$=$ $\langle$ $\widehat{F} = 0$ and $0 = B \wedge \neg B$, for any formula B $\rangle$
$\widehat{\Gamma}, B \wedge \neg B \vdash \widehat{\Delta}$
$=$ $\langle$ Rule for conjunction at the left hand side of the sequent $\rangle$
$\widehat{\Gamma}, B, \neg B \vdash \widehat{\Delta}$
$=$ $\langle$ Trading of formulas $\rangle$
$\widehat{\Gamma}, B \vdash B, \widehat{\Delta}$
$\longrightarrow$ $\langle$ Axiom $\rangle$
$\widehat{\diamond}$.

For the rules one has to observe that all the rules have identical counterparts in $\mathcal{R}^{\mathrm{BOOL}}_{\mathrm{SEQ}}$ or in the standard Sequent Calculus, thus sound.

In the sequel, we will use the fact that in the presence of the rule $\Gamma, B \vdash B, \Delta \longrightarrow \diamond$ the weakening rules $\Gamma, B \vdash \Delta \longrightarrow \Gamma \vdash \Delta$ and $\Gamma \vdash B, \Delta \longrightarrow \Gamma \vdash \Delta$ are redundant [8] .

Theorem 4. *The rewrite theory $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ is complete, that is, if a sequent S is provable in the sequent calculus then there is a derivation $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}} \vdash S \longrightarrow^* \diamond$.*

Proof. The proof for the axioms and structural equations not involving sequents is straightforward. We turn our attention to the equations and rules involving sequents and deduction steps. For those equations, only involving the connectives $\vee$ or $\equiv$ the property trivially holds. For those involving negation the translation $\neg\widehat{B} = B \equiv \mathsf{F}$ and applying the corresponding equivalence rule suffices. The interesting cases are the ones involving conjunction. Let us show the proof for one of them; the proof for the other one follows similarly. We want to prove that the Viry's rule $\widehat{\Gamma} \vdash \widehat{B \wedge C}, \widehat{\Delta} \longrightarrow \widehat{\Gamma} \vdash \widehat{B}, \widehat{\Delta} \bullet \widehat{\Gamma} \vdash \widehat{C}, \widehat{\Delta}$, has a counterpart derivation in $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$:

$\Gamma \vdash B \equiv C \equiv B \vee C, \Delta$
$\longrightarrow$ $\langle$ Rule for equivalence $\rangle$
$\Gamma, B \vdash C \equiv B \vee C, \Delta \bullet \Gamma, C \equiv B \vee C \vdash B, \Delta$
$\longrightarrow$ $\langle$ Weakening $\rangle$

$\Gamma, B \vdash C \equiv B \vee C, \Delta \bullet \Gamma \vdash B, \Delta$
$\longrightarrow$ $\langle$ Rule for equivalence $\rangle$
$\Gamma, B, C \vdash B \vee C, \Delta \bullet \Gamma, B, B \vee C \vdash C, \Delta \bullet \Gamma \vdash B, \Delta$
$\longrightarrow$ $\langle$ Rule for disjunction and weakening twice $\rangle$
$\Gamma, B, C \vdash B, C, \Delta \bullet \Gamma \vdash C, \Delta \bullet \Gamma \vdash B, \Delta$
$\longrightarrow$ $\langle$ Axiom $\rangle$
$\diamond \bullet \Gamma \vdash C, \Delta \bullet \Gamma \vdash B, \Delta$
$=$ $\langle$ Identity and commutativity of $\bullet$ $\rangle$
$\Gamma \vdash B, \Delta \bullet \Gamma \vdash C, \Delta$.

From the previous two theorems we have that the rewrite theory $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ is sound and complete, that is, a sequent S is provable in the sequent calculus iff there is a derivation $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}} \vdash S \longrightarrow^* \diamond$.

6.1 $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ is Weakly Coherent

As mentioned in Section 3, for a rewrite theory $\mathcal{R} = (\Sigma, E \cup A, R)$ to be efficiently executable it is very important to show that its equational theory E is confluent and terminating modulo A, and its rewrite rules R are weakly coherent [27] relative to its equations E modulo the given equational axioms A. We can then *execute* both the rules R and the equations E by rewriting modulo A without losing completeness. Therefore, for our theory $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}} = (\Sigma^{\mathrm{DS}}_{\mathrm{SEQ}}, E^{\mathrm{DS}}_{\mathrm{SEQ}} \cup A^{\mathrm{DS}}_{\mathrm{SEQ}}, R^{\mathrm{DS}}_{\mathrm{SEQ}})$, proofs of confluence and termination of $E^{\mathrm{DS}}_{\mathrm{SEQ}}$ modulo $A^{\mathrm{DS}}_{\mathrm{SEQ}}$, and coherence of $R^{\mathrm{DS}}_{\mathrm{SEQ}}$ with respect to $E^{\mathrm{DS}}_{\mathrm{SEQ}}$ modulo $A^{\mathrm{DS}}_{\mathrm{SEQ}}$, mean that $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ provides a *mechanization* of the sequent calculus *modulo* $E^{\mathrm{DS}}_{\mathrm{SEQ}} \cup A^{\mathrm{DS}}_{\mathrm{SEQ}}$ by rewriting. Section 6.2 discusses our experience with the mechanization of $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$. Here we focus on the proofs of confluence, termination and weak coherence.

In order to achieve our goal, we first enrich $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ with new axioms, corresponding to meta-theorems of the sequent calculus. These axioms will be used as oriented equations in the proof of week coherence, but will not alter the soundness of the theory by obvious reasons.

Lemma 2. *The following equivalences are meta-theorems of the sequent calculus:*

1. $\Gamma, B \vdash \Delta \bullet \Gamma \vdash \Delta$ *iff* $\Gamma \vdash \Delta$
2. $\Gamma \vdash B, \Delta \bullet \Gamma \vdash \Delta$ *iff* $\Gamma \vdash \Delta$
3. $\Gamma, B \vdash \Delta \bullet \Gamma \vdash B, \Delta$ *iff* $\Gamma \vdash \Delta$

Proof. The proofs are very similar. Let us prove the first equivalence: the if part follows from the fact that if there is a proof of Δ from Γ, clearly

there is a proof of Δ from Γ, B; the only-if part follows from the weakening rules and $\bullet$ idempotent.

In the sequel, as aforementioned, we assume that the previous equivalences are part of the simplifying equations $E^{\mathrm{DS}}_{\mathrm{SEQ}}$ oriented from left to right.

Theorem 5. *$E^{\mathrm{DS}}_{\mathrm{SEQ}}$ is confluent and terminating modulo $A^{\mathrm{DS}}_{\mathrm{SEQ}}$.*

Proof. Termination and confluence of $E^{\mathrm{DS}}_{\mathrm{SEQ}}$ have been mechanically checked with the CiME system, assuming that the explicit substitution calculus we use is totally defined over formulas and does not generate any overlapping with the remaining equations and rules.

Theorem 6. *$R^{\mathrm{DS}}_{\mathrm{SEQ}}$ is weakly coherent with respect to $E^{\mathrm{DS}}_{\mathrm{SEQ}}$ modulo $A^{\mathrm{DS}}_{\mathrm{SEQ}}$.*

Proof. We check that all critical pairs between $R^{\mathrm{DS}}_{\mathrm{SEQ}}$ and $E^{\mathrm{DS}}_{\mathrm{SEQ}}$ are properly joinable. First observe that we have critical pairs at the level of sets of formulas: we have equations for $\equiv$ and $\vee$ (which are the non-constant constructors of T_{DS}) and the rules for $\equiv$ and $\vee$, respectively. In each case, we have to prove that two critical pairs are joinable: one corresponds to the obvious critical pair, and the other to the critical pair obtained by augmenting by one the number of argument of $\equiv$ and $\vee$, since the operators do not have structural axioms for identity. For this matter, the complete set of critical pairs (the first argument corresponds to the term obtained by the equation, while the second to the one obtained by the rule) we have to check is the following:

- $(\Gamma, \mathsf{T} \vdash \Delta \quad , \quad \Gamma, B, B \vdash \Delta \bullet \Gamma \vdash B, B, \Delta)$
- $(\Gamma, \mathsf{T} \equiv X \vdash \Delta \quad , \quad \Gamma, B \equiv B, X \vdash \Delta \bullet \Gamma \vdash B \equiv B, X, \Delta)$
- $(\Gamma, B \vdash \Delta \quad , \quad \Gamma, \mathsf{T}, B \vdash \Delta \bullet \Gamma \vdash \mathsf{T}, B, \Delta)$
- $(\Gamma, B \equiv X \vdash \Delta \quad , \quad \Gamma, B \equiv \mathsf{T}, X \vdash \Delta \bullet \Gamma \vdash B \equiv \mathsf{T}, X, \Delta)$
- $(\Gamma \vdash \mathsf{T}, \Delta \quad , \quad \Gamma, B \vdash B, \Delta \bullet \Gamma, B \vdash B, \Delta)$
- $(\Gamma \vdash \mathsf{T} \equiv X, \Delta \quad , \quad \Gamma, B \equiv B \vdash X, \Delta \bullet \Gamma, X \vdash B \equiv B, \Delta)$
- $(\Gamma \vdash B, \Delta \quad , \quad \Gamma, B \vdash \mathsf{T}, \Delta \bullet \Gamma, \mathsf{T} \vdash B, \Delta)$
- $(\Gamma \vdash B \equiv X, \Delta \quad , \quad \Gamma, B \equiv \mathsf{T} \vdash X, \Delta \bullet \Gamma, X \vdash B \equiv \mathsf{T}, \Delta)$
- $(\Gamma, B \vdash \Delta \quad , \quad \Gamma, B \vdash \Delta \bullet \Gamma, B \vdash \Delta)$
- $(\Gamma, B \vee X \vdash \Delta \quad , \quad \Gamma, B \vee B \vdash \Delta \bullet \Gamma, X \vdash \Delta)$
- $(\Gamma, B \vdash \Delta \quad , \quad \Gamma, B \vdash \Delta \bullet \Gamma, \mathsf{F} \vdash \Delta)$
- $(\Gamma, B \vee X \vdash \Delta \quad , \quad \Gamma, B \vee \mathsf{F} \vdash \Delta \bullet \Gamma, X \vdash \Delta)$
- $(\Gamma, \mathsf{T} \vdash \Delta \quad , \quad \Gamma, B \vdash \Delta \bullet \Gamma, \mathsf{T} \vdash \Delta)$
- $(\Gamma, \mathsf{T} \vee X \vdash \Delta \quad , \quad \Gamma, B \vee \mathsf{T} \vdash \Delta \bullet \Gamma, X \vdash \Delta)$

– $(\Gamma \vdash B, \Delta \quad , \quad \Gamma \vdash B, B, \Delta)$
– $(\Gamma \vdash B \vee X, \Delta \quad , \quad \Gamma \vdash B \vee B, X, \Delta)$
– $(\Gamma \vdash B, \Delta \quad , \quad \Gamma \vdash B, \mathsf{F}, \Delta)$
– $(\Gamma \vdash B \vee X, \Delta \quad , \quad \Gamma \vdash B \vee \mathsf{F}, X, \Delta)$
– $(\Gamma \vdash \mathsf{T}, \Delta \quad , \quad \Gamma \vdash B, \mathsf{T}, \Delta)$
– $(\Gamma \vdash \mathsf{T} \vee X, \Delta \quad , \quad \Gamma \vdash B \vee \mathsf{T}, X, \Delta)$

where X is a variable representing the mentioned additional argument. So, if (c_1, c_2) is one of the critical pairs, for weak coherence we need to prove that the term c_2 (modulo equations) can be reached from the term c_1 via equations and rules. We present the proof of joinability of $(\Gamma, \mathsf{T} \vdash \Delta \quad , \quad \Gamma, B, B \vdash \Delta \bullet \Gamma \vdash B, B, \Delta)$:

$\Gamma, \mathsf{T} \vdash \Delta$
$=$ $\langle$ Axiom for T on the left hand side $\rangle$
$\Gamma \vdash \Delta$

On the right-hand side:

$\Gamma, B, B \vdash \Delta \bullet \Gamma \vdash B, B, \Delta$
$=$ $\langle$ Idempotency of sets $\rangle$
$\Gamma, B \vdash \Delta \bullet \Gamma \vdash B, \Delta$
$=$ $\langle$ Previous Lemma $\rangle$
$\Gamma \vdash \Delta$

It is easy to check that all these critical pairs are joinable, leading us to conclude that $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ is weakly coherent, since it is terminating and confluent.

6.2 An Executable Specification in Maude

We present part of the specification of the rewrite theory $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ in Maude. Maude is a high-performance logical framework based on rewriting logic [3]. We only give the fragment corresponding to the sequent rewrite rules in $R^{\mathrm{DS}}_{\mathrm{SEQ}}$. The key point is that, since Maude modules *are* rewrite theories, the Maude specification of $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ is just a transcript in typewriter font notation of $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$, plus a few auxiliary functions to handle variables and substitutions.

```
mod SEQ is
   ...
   vars FSB FSC : FSet .   vars B C : Formula   var S : Seq .
```

```
   rl FSB,B |- B,FSC  =>  mts .
   rl FSB,B equ C |- FSC  =>  FSB,B,C |- FSC * FSB |- B,C,FSC .
   rl FSB |- B equ C,FSC  =>  FSB,B |- C,FSC * FSB,C |- B,FSC .
   rl FSB,B or C |- FSC   =>  FSB,B |- FSC * FSB,C |- FSC .
   rl FSB |- B or C,FSC   =>  FSB |- B,C,FSC .
   rl FSB,[x : B] |- FSC  =>  FSB,B[t/x] |- FSC  [nonexec] .
   rl FSB |- [x : B],FSC  => FSB |- B[newVar(FSB,B,FSC)/x],FSB .
endm
```

Universal quantification is represented with square brackets. We use `mts` to represent $\diamond$, `_equ_` for $\_ \equiv \_$, `_or_` for $\_ \vee \_$, and `_*_` for $\_ \bullet \_$. Both `_,_` and `_*_` are declared as ACU operators, that is, as associative and commutative, and having an identity element. Maude efficiently implements matching and unification modulo AC and ACU. The last two rules deserve special attention. The next-to-last rule is declared not executable (`nonexec`) because there is an extra-variable in its right-hand side, and thus the derivation tree may have infinite branching. The key observation is that the presence of extra variables in a rule's right-hand side, while making rewriting with it problematic, is unproblematic for *narrowing* with the rules of a coherent or weak coherent rewrite theory $\mathcal{R}$ modulo its equational axioms, under the assumption that its rewrite rules are *topmost*[3]. This makes narrowing with the rules of the rewrite theory a sound and complete deduction process [19] for solving existential queries of the form

$$\exists \overrightarrow{x} \,.\, t(\overrightarrow{x}) \longrightarrow^* t'(\overrightarrow{x}) \,.$$

In our case, the existential queries in question are of the form

$$\square \vdash B \longrightarrow^* \diamond \,,$$

where B is the FOL sentence we want to prove. Although B is a sentence and therefore has no free variables, the above next-to-last rule introduces new variables, which are then incrementally instantiated as new rules are used to narrow the current set of sequents at each step. We can perform such narrowing by exploiting the efficient AC and ACU unification algorithms available in the current version of Maude and the fact that it is a reflective language [3]. The last rule makes explicit the need for the auxiliary function `newVar` to generate fresh variables not occurring in the given formulas.

[3] In our case we can easily make $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ topmost by enclosing the set of sequents under an angle bracket operator and adding an extra variable for the remaining sequents to each rule.That is, for narrowing purposes, we can transform a sequent rule such as for example $\Gamma \vdash B \vee C, \Delta \longrightarrow \Gamma \vdash B, C, \Delta$ into a topmost rule $\langle \Gamma \vdash B \vee C, \Delta \bullet SS \rangle \longrightarrow \langle \Gamma \vdash B, C, \Delta \bullet SS \rangle$, where SS is a variable of sort SSet.

We have used the complete specification in Maude of $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ to mechanically prove several FOL theorems. Here, we present the case study of Andrew's challenge [9], a theorem that is quite difficult to prove for many theorem provers and is used as a benchmark. Andrew's challenge is to prove the following theorem:

$$(\exists x.\forall y.(P(x) \equiv P(y)) \equiv ((\exists z.Q(z)) \equiv (\forall w.P(w)))) \equiv \\ (\exists x.\forall y.(Q(x) \equiv Q(y)) \equiv ((\exists z.P(z)) \equiv (\forall w.Q(w)))) \ .$$

Since $\equiv$ is both associative and commutative, we can rephrase Andrew's challenge as $B \equiv C$, where:

$$\begin{array}{rcl} B & : & \exists x.\forall y.(P(x) \equiv P(y)) \equiv \exists z.P(z) \equiv \forall w.P(w) \\ C & : & \exists x.\forall y.(Q(x) \equiv Q(y)) \equiv \exists z.Q(z) \equiv \forall w.Q(w) \ , \end{array}$$

and it is assumed that the formula is closed. Observe that B is valid regardless of P, and the same applies to C. Hence, it is enough to prove B or C. Here, we choose to prove the former, whose translation corresponds to the $\Sigma^{\mathrm{DS}}_{\mathrm{SEQ}}$-term $\widehat{B}$:

```
{ v(0) : [ v(1) : P(v(1)) equ P(v(2)) ] } equ { v(3) : P(v(3)) } equ [ v(4) : P(v(4)) ]
```

where `P` is of sort `Pred` . The proof search in Maude using narrowing modulo the $A^{\mathrm{DS}}_{\mathrm{SEQ}}$ axioms is shown below:

```
Maude> red narrowSearch( mtf |- B̂ , mts , full ACU-unify E-simplify ) .
==========================================
reduce in SEQ : narrowSearch( mtf |- B̂ , mts , full ACU-unify E-simplify ) .
rewrites: 56664622 in 149750ms cpu (1193116ms real) (378394 rewrites/second)
...
```

We have used the auxiliary function `narrowSearch` which calls the narrowing strategy we use. The first argument corresponds to the sequent we want to prove, the second to the empty sequent (i.e., to the term where there is nothing left to prove) and the third to a list of parameters for the narrowing algorithm; in this case we use ACU unification and simplification with the equations before and after any narrowing step. Upon termination, the narrowing strategy returns the substitution found, meaning that the initial sequent can be transformed into the empty one and the time taken for the search. Thus, our implementation of $\mathcal{R}^{\mathrm{DS}}_{\mathrm{SEQ}}$ was able to automatically solve Andrew's challenge. We have omitted the details of the resulting substitution.

7 Conclusions

We have explained the general idea of how logics can be specified as rewrite theories to obtain "theorem proving modulo" proof systems that can substantially raise the level of abstraction at which a user interacts with a theorem prover and make deduction considerably more scalable. We have then focused on building in decision procedures for Boolean equivalence of formulas, and have shown how they can be seamlessly integrated within the theorem proving modulo paradigm. Specifically, we have presented three new such equationally-based procedures, and have used one of them, deciding the Dijkstra-Scholten propositional logic, to obtain an executable rewrite theory for a sequent calculus version of Dijkstra-Scholten first-order logic that can be directly used to prove nontrivial theorems. A similar "theorem proving modulo" approach to obtain a decision procedure for the Syllogistic Logic with Complements of L. Moss [20] has also been presented. We have also shown how the decision procedures can be further sped up by the use of optimizing equations. We have also presented experimental results suggesting that these procedures, when implemented on a high-performance rewrite engine, have very good efficiency and outperform a DPLL(T)-based SAT-solver.

We view this work as a step forward in bringing the theorem proving modulo ideas closer to practice. However, more research is needed, both in terms of developing other compelling case studies for other logics and proof systems, and in terms of developing a body of generic techniques that should make it straightforward to obtain an efficient mechanization of a logic directly from a rewriting logic specification of its inference system. Such techniques should include, for example, more efficient implementations of narrowing modulo axioms, and generic libraries of tactics expressed as generic rewriting strategies in the sense of [7].

References

1. R. Backhouse. *Program Construction: : Calculating Implementations from Specifications*. Willey, 2003.
2. H. Barendregt and E. Barendsen. Autarkic computations and formal proofs. *Journal of Automated Reasoning*, 28(3):321–336, 2002.
3. M. Clavel, F. Durán, S. Eker, J. Meseguer, P. Lincoln, N. Martí-Oliet, and C. Talcott. *All About Maude - A High-Performance Logical Framework*. Springer LNCS Vol. 4350, 1st edition, 2007.
4. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 243–320. 1990.
5. E. W. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics*. Springer-Verlag, 1990.

6. G. Dowek, T. Hardin, and C. Kirchner. Theorem proving modulo. *J. Autom. Reasoning*, 31(1):33–72, 2003.
7. S. Eker, N. Martí-Oliet, J. Meseguer, and A. Verdejo. Deduction, strategies, and rewriting. In N. Martí-Oliet, editor, *Proc. Strategies 2006*, pages 417–441. ENTCS, Elsevier, 2007. To appear.
8. J.-Y. Girard. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1989.
9. D. Gries. A calculational proof of Andrews's challenge. Technical Report TR96-1602, Cornell University, Computer Science, Aug.28, 1996.
10. D. Gries and F. B. Schneider. *A Logical Approach to Discrete Math*. Texts and Monographs in Computer Science. Springer Verlag, 1993.
11. D. Gries and F. B. Schneider. Equational propositional logic. *Inf. Process. Lett.*, 53(3):145–152, 1995.
12. J. Hendrix, H. Ohsaki, and J. Meseguer. Sufficient completeness checking with propositional tree automata. Technical Report UIUCDCS-R-2005-2635, University of Illinois Urbana-Champaign, 2005.
13. J. Hsiang. *Topics in automated theorem proving and program generation*. PhD thesis, University of Illinois at Urbana-Champaign, 1982.
14. N. Jacobson. *Basic algebra. I*. W. H. Freeman and Co., San Francisco, Calif., 1974.
15. Laboratoire de Recherche en Informatique. The CiME 2.0 System (`http://cime.lri.fr/`).
16. F. W. Lawvere. Functorial semantics of algebraic theories. *Proceedings, National Academy of Sciences*, 50:869–873, 1963. Summary of Ph.D. Thesis, Columbia University.
17. V. Lifschitz. On calculational proofs. *Ann. Pure Appl. Logic*, 113(1-3):207–224, 2001.
18. N. Martí-Oliet and J. Meseguer. Rewriting logic as a logical and semantic framework. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd. Edition*, pages 1–87. Kluwer Academic Publishers, 2002. First published as SRI Tech. Report SRI-CSL-93-05, August 1993.
19. J. Meseguer and P. Thati. Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols. *Higher-Order and Symbolic Computation*, 20(1–2):123–160, June 2007.
20. L. S. Moss. Syllogistic logic with complements. Draft, 2007.
21. C. Rocha and J. Meseguer. Five isomorphic Boolean theories and four equational decision procedures. Technical Report 2007-2818, University of Illinois at Urbana-Champaign, 2007.
22. C. Rocha and J. Meseguer. A rewriting decision procedure for Dijkstra-Scholten's syllogistic logic with complements. *Revista Colombiana de Computación*, 2007.
23. G. F. Simmons. *Introduction to topology and modern analysis*. McGraw-Hill Book Co., Inc., New York, 1963.
24. R. Socher-Ambrosius and P. Johann. *Deduction Systems*. Springer-Verlag, Berlin, 1997.
25. M.-O. Stehr and J. Meseguer. Pure type systems in rewriting logic: Specifying typed higher-order languages in a first-order logical framework. In O. Owe, S. Krogdahl, and T. Lyche, editors, *Essays in Memory of Ole-Johan Dahl*, volume 2635 of *Lecture Notes in Computer Science*, pages 334–375. Springer LNCS Vol. 2635, 2004.
26. P. Viry. Adventures in sequent calculus modulo equations. In *Electr. Notes Theor. Comput. Sci*, volume 15, 1998.
27. P. Viry. Equational rules for rewriting logic. *Theoretical Computer Science*, 285:487–517, 2002.