Interactive proof-search for equational reasoning

FAVIO E. MIRANDA-PEREA*, *Departamento de Matemáticas, Facultad de Ciencias UNAM, Circuito Exterior S/N, Cd. Universitaria 04510, CdMx, México.*

LOURDES DEL CARMEN GONZÁLEZ HUESCA, *Departamento de Matemáticas, Facultad de Ciencias UNAM, Circuito Exterior S/N, Cd. Universitaria 04510, CdMx, México.*

P. SELENE LINARES-ARÉVALO, *Departamento de Matemáticas, Facultad de Ciencias UNAM, Circuito Exterior S/N, Cd. Universitaria 04510, CdMx, México.*

Abstract

Equational reasoning arises in many areas of mathematics and computer science. It is a cornerstone of algebraic reasoning and results essential in tasks of specification and verification in functional programming, where a program is mainly a set of equations. The usual manipulation of identities while conducting informal proofs obviates many intermediate steps that are neccesary while developing them using a formal system, such as the equationally complete Birkhoff calculus $\mathcal{B}$. This deductive system does not fit in the common manner of doing mathematical proofs, and it is not compatible with the mechanisms of proof assistants. The aim of this work is to provide a deductive system $\mathcal{B}^{\text{GOAL}}$ for equality, equivalent to $\mathcal{B}$ but suitable for constructing equational proofs in a backward fashion. This feature makes it adequate for interactive proof-search in the approach of proof assistants. This will be achieved by turning $\mathcal{B}^{\text{GOAL}}$ into a transition system of formal tactics in the style of Edinburgh LCF, such transformation allows us to give a direct formal definition of backward proof in equational logic.

Keywords: Equational logic, Birkhoff calculus, proof-search, rewrite, goal-oriented reasoning.

1 Introduction

Equational reasoning is the kind of algebraic inference used to manipulate equations that we learn in elementary school. This method of reasoning is quite relevant in mathematics and computer science. For instance, in abstract algebra, several structures like groups and rings, are defined by means of purely equational axioms. Therefore, proofs within their corresponding theories are achieved, to a lesser or greater extent, by means of equational reasoning. This proof method is also relevant in computer science, particularly in pure functional programming, where a program is a sequence of mathematical equations describing some intended function and the mechanism of computation consists of replacing equals by equals in an oriented way, usually by reading an equation from left to right. Due to these features, functional programming encourages the use of equational reasoning for specification, verification, debugging, refactoring and optimization in both algorithms and data structure design, see e.g. [3, 4, 16, 17]. Moreover, the importance of this methodology for functional

*E-mail: favioemp@gmail.com

Vol. 28, No. 6, © The Author(s) 2020. Published by Oxford University Press. All rights reserved.
For permissions, please e-mail: journals.permission@oup.com.
Advance Access published 29 May 2020 doi:10.1093/jigpal/jzaa013

programming has led to the development of tools, like Liquid Haskell [37], dedicated to verify equational reasoning mechanized within the Haskell programming language.

With their coming of age, interactive theorem provers, also called proof assistants, provide the computer scientist or mathematician with reliable tools for the automation of logical proofs, in particular those involving equational reasoning. A proof assistant is a computer system that consists of a domain-specific language allowing to represent logical objects, as well as definitions and theorems about them, together with a mechanism that allows for the interactive construction and validation of proofs. This proof construction process is driven through a fully interactive goal-oriented proof search. This means that the search is guided by a human, who proposes a specific reasoning method, for instance transitivity of equality or substitution of equals by equals, and leaves to the machine the task of verifying if such a choice is adequate to solve the ongoing goal. In positive case, the machine answers by presenting the new subgoals whose provability achieves also the original goal. For instance, to show the equation $s = t$ by means of transitivity, first the user must provide a term r from which the machine generates two new subgoals, namely $s = r$ and $r = t$.

The full interaction between machine and human rules out powerful automated reasoning techniques such as term rewrite systems [11] or paramodulation-based theorem provers [30]. In opposite to the usual proof development in mathematical logic, this proof-search approach corresponds mainly to the so-called backward reasoning strategy: we start with the proof sought after, say a proof of equation E from hypotheses Γ, formally a sequent $\Gamma \vdash E$, and we look for an available inference rule, whose conclusion matches this sequent, thus generating as new subgoals the sequents corresponding to its premises. The proof-search concludes successfully if all generated goals are achieved.

Proof assistants are nowadays used to perform non-trivial verification tasks and to develop sophisticated mathematical proofs, being formal logic the most important guideline for these purposes. It is then clear that a good understanding of the goal-oriented approach is mandatory for carrying out such duties successfully. However, this reasoning paradigm is not considered in standard logic books like [21, 27, 28], and although it is deeply discussed in books about proofs in mathematics like [12, 35, 38], in these sources, the representation of the goal-oriented approach is only as proof-writing techniques and it is not related with proof assistants. The absence of a formal notion of backward proof constitutes a gap between formal logic, as taught to computer science or mathematics undergraduates and practical logic as the operational mechanism of a proof assistant. For instance, the classical reference on Edinburgh LCF [31, Section 2.13] provides us with a sequent calculus for backward reasoning but not with a definition of backward proof and the same happens with the more recent handbook [19]. This notion is deeply related to the notion of tactic à la Edinburgh LCF, a concept which is defined at a low level, i.e. as an implementation, usually in a functional language, of the backward application of a specific inference rule. Nevertheless, a formal definition of backward proof is not spelled out. A remarkable exception appears in [23]. However, this is only a formal non-operational notion. In particular proofs, there are written forwards.

In previous work [29], we have shown how to construct backward proofs in the case of minimal propositional logic, by turning a proof-search process into a transition system inspired by the actual proof mechanisms of the Coq proof assistant [8]. This formally justifies most of the proof strategies discussed in the above-mentioned books on mathematical proofs; fills a gap between the theory and practice of constructing formal proofs and, in our experience, has advantages while teaching constructive logic to undergraduate computer science students [29]. It is important to remark that, in our approach, the proof-search is done by a proof which is constructed only in a backward direction and it is not a combination of forward and backward steps, this constitutes an important difference with related work [13, 33, 34]. This is so, for we are not interested in an automated proof-search

where the proof sought after is constructed by the software without user interaction. This way, a backward proof is for us simply a coherent sequence of tactics.

The aim of this work is to present a deductive system suitable for interactive proof-search in the case of pure first-order equational logic, which is the fragment of first-order predicate logic whose only predicate is equality and where all variables are universally quantified in an implicit way. This logic suffices to handle the usual mathematical reasoning around equality, like the development of proofs by transitivity or rewriting. With this aim in mind, it is clear that we need deductive systems adequate to model the actual reasoning processes of a human agent. Therefore, we depart from fully automated systems, for instance Satisfiability modulo theories solvers (SMT solvers), which implement powerful inference procedures like paramodulation, superposition or congruence closure. These programs allow to solve sophisticated instances of equational reasoning in a fully automatic way and can interact with a proof assistant, thus providing an automated verification process for equations. Though this is a very important topic, our work goes in an opposite direction. The goal here is to provide deductive systems that capture some specific human (mathematical) strategies that allow to interact with the proof assistant in a way that the computer-assisted proof generated by such interaction can be understood by a common user of the theorem prover, corresponding, in some sense, to a usual paper-and-pencil proof. A specific application and development of this proof approach is presented in some of our work on modal logic [10, Section 2]. The deductive systems described in this paper achieve this goal.

The paper is organized as follows: in Section 2, we recall the generalities of deductive systems à la natural deduction. Then, in Sections 3 and 4, we present respectively two common systems for equality, namely the Birkhoff calculus and a calculus with a version of the demodulation rule, i.e. the rule of replacement of equals by equals. A system with explicit rewriting, enhancing the previous ones, is presented in Section 5. The equivalence of the three deductive systems is discussed in Section 6. A suitable system for backward reasoning and interactive proof-search is defined in Section 7. The formal notion of backward proof is discussed in Section 8 where a transition system of tactics comprising the usual equational reasoning used in mathematics is provided together with the proof of its equivalence with the usual deductive systems, thus showing the equivalence of forward and backward proofs. We close with some final remarks in Section 9.

Before starting our formal discussion, let us present some elementary examples of equational reasoning. Consider the axioms defining ring theory:

$$\begin{array}{rclll} x+0 & = & x & R1 & \qquad x+y = y+x \quad R2 \\ x+(-x) & = & 0 & R3 & \qquad (x+y)+z = x+(y+z) \quad R4 \\ (x\cdot y)\cdot z & = & x\cdot(y\cdot z) & R5 & \qquad x\cdot(y+z) = x\cdot y+x\cdot z \quad R6 \end{array}$$

$$(x+y)\cdot z = x\cdot z + y\cdot z \quad R7$$

Let us show next some examples of equational reasoning within this theory

EXAMPLE 1

The following is a proof of $x\cdot 0 = 0$:

$$\begin{array}{rcll} x\cdot 0 & = & x\cdot 0+0 & R1 \\ & = & x\cdot 0+(x\cdot 0+-(x\cdot 0)) & R3 \\ & = & (x\cdot 0+x\cdot 0)+-(x\cdot 0) & R4 \\ & = & (x\cdot(0+0))+-(x\cdot 0) & R6 \\ & = & (x\cdot 0)+-(x\cdot 0) & R1 \\ & = & 0 & R3 \end{array}$$

This short proof corresponds to a routine mathematical reasoning where several formal steps are implicit, in particular the use of an equation in both directions, in this case $R1$, and the instantiation of variables. Moreover, the chain of equalities allows us to conclude the original goal by means of transitivity. Also, this is a forward proof, we start with a known valid equation, namely $x \cdot 0 = x \cdot 0 + 0$ and employ other valid equations, for instance $x \cdot 0 + -x \cdot 0 = 0$ corresponding to the last step, in order to achieve by transitivity the desired goal $x \cdot 0 = 0$.

Instead, we can construct a backward or goal-oriented derivation, which takes the original goal and generates new subgoal equations. The process ends successfully if we can solve all generated subgoals.

EXAMPLE 2

The following is a backward proof of $x \cdot 0 = 0$:

$$\begin{array}{rcll} x \cdot 0 & = & 0 & \textit{original goal} \\ x \cdot 0 + 0 & = & 0 & R1 \\ x \cdot 0 + (x \cdot 0 + -(x \cdot 0)) & = & 0 & R3 \\ (x \cdot 0 + x \cdot 0) + -(x \cdot 0) & = & 0 & R4 \\ x \cdot (0 + 0) + -(x \cdot 0) & = & 0 & R6 \\ x \cdot 0 + -(x \cdot 0) & = & 0 & R1 \end{array}$$

The derivation is finished for we arrive at an equation that is previously known, namely $x \cdot 0 + -x \cdot 0 = 0$ which is an instance of the axiom $R3$. This proof can be seen as a forward proof written backwards; indeed if we read the sequence of equations from below, we have another proof that can be obtained from the one in Example 1 in a direct way. However, to say that a backward proof is just a forward proof written backwards is far from accurate. In this particular example, the former proof omits the intermediate steps of transitivity, whereas the latter does not involve transitivity at all. It is important to observe that replacing or rewriting subterms by equal subterms is present in both proofs. However, in this work we will not discuss term rewrite systems, for, although they are very powerful and important in automated reasoning, these formalisms do not correspond to usual mathematical reasoning. For instance, Example 1 uses some equations in both directions, a feature that is ruled out by term rewriting systems.

Let us show now a more elaborated example involving the common methodology of using an auxiliary lemma in a proof.

EXAMPLE 3

Using the ring axioms together with the hypothesis $x \cdot x = x$, denoted H, the following reasoning proves that $x + y = x + (y + (x \cdot y + y \cdot x))$:

$$\begin{array}{rcll} x + y & = & (x + y) \cdot (x + y) & H \\ & = & x \cdot (x + y) + y \cdot (x + y) & R6 \\ & = & (x \cdot x + x \cdot y) + (y \cdot x + y \cdot y) & R6 \\ & = & (x + x \cdot y) + (y \cdot x + y \cdot y) & H \\ & = & (x + x \cdot y) + (y \cdot x + y) & H \\ & = & (x + x \cdot y) + (y + y \cdot x) & R2 \\ & = & x + (x \cdot y + (y + y \cdot x)) & R4 \end{array}$$

At this point in the proof, we realize that we are almost done if we apply associativity, commutativity and associativity again. This seems a useful general property which is convenient to prove in the

form of the lemma equation $x + (y + z) = y + (x + z)$. This way, the original proof is finished as follows:

$$\begin{array}{lll} \vdots & & \\ = & x + (x \cdot y + (y + y \cdot x)) & \\ = & x + (y + (x \cdot y + y \cdot x)) & \textit{lemma} \end{array}$$

As we mention before, another use of equational reasoning arises in functional programming where this methodology is the main tool to reason about properties of programs and to improve their efficiency, see e.g. [4, 22].

EXAMPLE 4
The following example is taken from [4, page 81]: Define `unzip` by

$$unzip \quad = \quad fork(map\ fst, map\ snd) \quad D1$$

Here, the point (.) denotes function composition. Prove by simple equational reasoning that

$$cross(map\ f, map\ g)\ .\ unzip = unzip\ .\ map(cross(f, g)).$$

You can use the functor laws of `map` and the following rules:

$$\begin{array}{rcll} cross(f, g)\ .\ fork(h, k) & = & fork(f\ .\ h\ ,\ g\ .\ k) & Law1 \\ fork(f, g)\ .\ h & = & fork(f\ .\ h\ ,\ g\ .\ h) & Law2 \\ fst\ .\ cross(f, g) & = & f\ .\ fst & Law3 \\ snd\ .\ cross(f, g) & = & g\ .\ snd & Law4 \\ map(f.g) & = & map\ f\ .\ map\ g & Law5 \end{array}$$

Solution: $cross(map\ f, map\ g)\ .\ unzip$

$$\begin{array}{lll} = & cross(map\ f, map\ g)\ .\ fork(map\ fst, map\ snd) & D1 \\ = & fork(map\ f\ .\ map\ fst\ ,\ map\ g\ .\ map\ snd) & Law1 \\ = & fork(map\ (f\ .\ fst)\ ,\ map\ (g\ .\ snd)) & Law5 \\ = & fork(map\ (fst\ .\ cross(f, g))\ ,\ map\ (g\ .\ snd)) & Law3 \\ = & fork(map\ (fst\ .\ cross(f, g))\ ,\ map\ (snd\ .\ cross(f, g))) & Law4 \\ = & fork(map\ fst\ .\ map\ (cross(f, g))\ ,\ map\ snd\ .\ map\ (cross(f, g))) & Law5 \\ = & fork(map\ fst\ ,\ map\ snd)\ .\ map\ (cross(f, g)) & Law2 \\ = & unzip\ .\ map\ (cross(f, g)) & D1 \end{array}$$

This example finishes our introduction. Next, we discuss the generalities of the kind of deductive systems used in this work.

2 Deductive systems for equality

In this section, we recall the basics on equational formulas and fix the notion of the deductive system we will use throughout the paper. This will be natural deduction systems [15, 32] with localized hypothesis that is presented as sequent systems. It is important to remark that we are not talking about sequent calculi, a very important kind of systems that have been deeply studied for equality reasoning (see [9]). We prefer natural deduction for these are the natural calculi, forgive the redundancy, for formalizing common mathematical reasoning. Besides, the presentation in sequent form is the right one for implementation. For example, with the use of sequents, all the current

hypotheses are available at every derivation step. This way, there is no need for the implementation of a mechanism to collect them. Let us start by recalling the basics about syntax.

DEFINITION 1 (Terms and equations).
Given a language $\mathcal{L}$, the set of terms is given by the following grammar:

$$t ::= x \mid c \mid f(t_1, \ldots, t_n),$$

where x is a variable symbol taken from an infinite countable set *Var*, $c \in \mathcal{L}$ is a constant symbol and $f \in \mathcal{L}$ is a function symbol of arity n. An equation is an expression of the form $t = s$ where t and s are terms.

Equations are the only kind of formulas available in the language. It is important to notice that, as terms do not have a binding feature, all variables occurring in terms and equations are free. However, it is useful to think they are universally quantified in an implicit way, just as it is done in usual mathematics.

Notation Constant symbols are denoted by a, b, c, functional symbols by f, g, h. The letters r, s, t, u, v denote arbitrary terms and the uppercase letters E, E' stand for equations.

An important operation while manipulating terms and equations is the substitution of a variable x by a given term s in a term t which is defined as follows:

DEFINITION 2 (Term substitution).
Consider a variable x and two terms t, s. The substitution of x by s in t, denoted $t[x := s]$, is defined as follows:

$$\begin{array}{rcl} c[x := s] & =_{def} & c \\ y[x := s] & =_{def} & \begin{cases} s & \text{if } y \text{ is the same variable as } x \\ y & \text{otherwise} \end{cases} \\ f(r_1, \ldots, r_n)[x := s] & =_{def} & f(r_1[x := s], \ldots, r_n[x := s]) \end{array}$$

As there are no bound variables, this operation is merely textual substitution. In some cases where the explicit mention to the variable x is not relevant, instead of $t[x := s]$ or $E[x := s]$, we just write $t[s]$ or $E[s]$. To simplify the reading of inference rules, sometimes we denote the components $[x := s]$ of a substitution by the letter σ. Furthermore, we can lift the substitution operation to equations in the obvious way:

$$(t = r)\sigma \;=_{def}\; t\sigma = r\sigma.$$

Next, we collect the proof-theoretic concepts needed later. As usual, a deductive system $\mathcal{D}$ over the formulas of a language $\mathcal{L}$ consists of a set $\mathcal{R}$ of inference rules, perhaps including axioms, together with a notion of formal proof. In our case, the inference rules manipulate judgments or sequents defined as follows.

DEFINITION 3 (Judgment and contexts).
A judgment or sequent $\mathcal{J}$ in a deductive system $\mathcal{D}$ is an expression of the form $\Gamma \vdash_{\mathcal{D}} E$, where E is an equation and Γ is a context of equations. This sequent is read as 'equation E follows from context Γ'. A context Γ is defined by the following grammar:

$$\Gamma ::= \cdot \mid \Gamma, E.$$

Therefore, a context is either an empty list of equations, denoted by $\cdot$, or a non-empty list Γ, E whose last element is E. Moreover, we also write Γ, Γ' for the appending of context Γ to the context Γ'.

It is important to remark that for us, contexts are neither sets nor multisets, but lists of equations. This choice is taken to facilitate the correspondence of the theoretical concepts here developed with an actual implementation in a programming language or proof assistant.

The notion of proof or derivation that we will use throughout the paper is given by the following.

DEFINITION 4 (Derivation).
A proof or derivation of a judgment $\mathcal{J}$ in a deductive system $\mathcal{D}$ is a finite sequence of judgments $\Pi = \langle \mathcal{J}_1, \ldots, \mathcal{J}_k \rangle$ such that $\mathcal{J}_k = \mathcal{J}$ and for each $\mathcal{J}_i \in \Pi$, $1 \leq i \leq k$, one of the following conditions holds:

- $\mathcal{J}_i$ is an instance of an axiom.
- $\mathcal{J}_i$ is the conclusion of an instance of an inference rule whose premises are $\mathcal{J}_{i_1}, \ldots, \mathcal{J}_{i_m} \in \Pi$ with $i_1, \ldots, i_m < i$.

Moreover, if $\mathbb{H}$ is a given set of judgments, we say that Π is a proof of $\mathcal{J}$ from hypotheses $\mathbb{H}$ if, together with the above conditions, we also allow that $\mathcal{J}_i \in \mathbb{H}$.

Let us remark that our notion of proof is linear as opposed to the tree-style also common in proof theory. Furthermore, it corresponds to a proof constructed forwards starting from axioms. This departs from the usual mathematical practice, where backward reasoning plays an important role. Unfortunately, the literature lacks of a formal notion of backward proof. Instead, this kind of reasoning provides just a heuristic aid in proof construction. A notable exception figures in the pioneering works of Kanger [23–25] (a more accessible reference is [20]) where the notion of proof is related to the backward construction pursuit here. In essence, this definition consists in modifying Definition 4 as follows:

DEFINITION 5 (Kanger's derivation).
A proof or derivation of a judgment $\mathcal{J}$ in a deductive system $\mathcal{D}$ is a finite sequence of judgments $\Pi = \langle \mathcal{J}_1, \ldots, \mathcal{J}_k \rangle$ such that $\mathcal{J}_1 = \mathcal{J}$ and for each $\mathcal{J}_i \in \Pi$, $1 \leq i \leq k$, one of the following conditions holds:

- $\mathcal{J}_i$ is an instance of an axiom.
- $\mathcal{J}_i$ is the conclusion of an instance of an inference rule whose premises are $\mathcal{J}_{i_1}, \ldots, \mathcal{J}_{i_m} \in \Pi$ with $i_1, \ldots, i_m > i$.

Kanger's definition starts with the judgment sought after, whereas the usual notion of formal proof ends with it. Also, the justifications of any step appear after it. This gives an idea of backward proof. Regrettably, even when this intuitive notion permeates Kanger's works, the concept is not engaged.

Our purpose here is to give a formal notion of backward proof for equational logic that corresponds to the kind of techniques implemented in proof assistants. But first, we review some traditional deductive systems for equational reasoning.

3 The Birkhoff calculus $\mathcal{B}$

The first deductive system for equality we discuss is the well-known calculus $\mathcal{B}$ of Birkhoff, originated in his work on varieties of algebras [5]. Apart form its theoretical importance, this system

$$\frac{E \in \Gamma}{\Gamma \vdash_{\mathcal{B}} E}\ \textsc{Hyp} \qquad \frac{}{\Gamma \vdash_{\mathcal{B}} t = t}\ \textsc{Refl} \qquad \frac{\Gamma \vdash_{\mathcal{B}} s = t}{\Gamma \vdash_{\mathcal{B}} t = s}\ \textsc{Sym}$$

$$\frac{\Gamma \vdash_{\mathcal{B}} t = r \qquad \Gamma \vdash_{\mathcal{B}} r = s}{\Gamma \vdash_{\mathcal{B}} t = s}\ \textsc{Trans} \qquad \frac{\Gamma \vdash_{\mathcal{B}} t = s}{\Gamma \vdash_{\mathcal{B}} t\sigma = s\sigma}\ \textsc{Inst}$$

$$\frac{\Gamma \vdash_{\mathcal{B}} t_1 = s_1 \quad \ldots \quad \Gamma \vdash_{\mathcal{B}} t_n = s_n}{\Gamma \vdash_{\mathcal{B}} f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)}\ \textsc{Congr}$$

FIGURE 1 Birkhoff calculus.

is relevant in algebraic specification, for it provides a natural operational interpretation of equational specifications (e.g. [36]). The inference rules of $\mathcal{B}$ are depicted in Figure 1. Let us observe that, as we are using sequents, a starting rule HYP to derive hypothesis is mandatory.

These rules correspond to some usual reasoning strategies of equality that a human might attempt in a proof, like transitivity and symmetry. The instance rule reflects the fact that the universal quantifiers are implicitly present in equations. Furthermore, $\mathcal{B}$ is sound and complete with respect to the equational fragment of first-order logic.

THEOREM 1 (Birkhoff 1935 [5]).
Let Γ be a context. The following conditions are equivalent:

- The equation $s = t$ is derivable in $\mathcal{B}$, i.e. $\Gamma \vdash_{\mathcal{B}} s = t$.
- The equation $s = t$ is a logical consequence of Γ, i.e. $\Gamma \models s = t$.

PROOF. The theorem and its proof are an implicit consequence of the results in [5]. For an explicit proof, see for example [2, 6, 36]. □

The mentioned proofs of this important theorem manipulate contexts defined as sets instead of lists. Therefore, it could be the case that structural rules are needed for the proofs to go through in our setting. These rules are admissible in all the systems of this article (see Lemma 3).

The above theorem justifies the intuition that, if a particular equation is a logical consequence of a given set of equations Γ, then it can be verified by rewriting some of its subterms using equations in Γ. This is the common algebraic way we learn in basic school. Let us develop now our introductory Example 1 within $\mathcal{B}$.

EXAMPLE 5
Let $\mathcal{R}$ be the list of ring axioms in page 7. The following is a proof of $\mathcal{R} \vdash x \cdot 0 = 0$ in $\mathcal{B}$.

(1)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 = x \cdot 0$	REFL
(2)	$\mathcal{R} \vdash_{\mathcal{B}} x + -x = 0$	HYP
(3)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 + -(x \cdot 0) = 0$	INST(2)
(4)	$\mathcal{R} \vdash_{\mathcal{B}} 0 = x \cdot 0 + -(x \cdot 0)$	SYM(3)
(5)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 + 0 = x \cdot 0 + (x \cdot 0 + -(x \cdot 0))$	CONGR(1)(4)
(6)	$\mathcal{R} \vdash_{\mathcal{B}} (x + y) + z = x + (y + z)$	HYP
(7)	$\mathcal{R} \vdash_{\mathcal{B}} x + (y + z) = (x + y) + z$	SYM(6)

(8)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 + (x \cdot 0 + -(x \cdot 0)) = (x \cdot 0 + x \cdot 0) + -(x \cdot 0)$	INST(7)
(9)	$\mathcal{R} \vdash_{\mathcal{B}} x(y + z) = x \cdot y + x \cdot z$	HYP
(10)	$\mathcal{R} \vdash_{\mathcal{B}} x(0 + 0) = x \cdot 0 + x \cdot 0$	INST(9)
(11)	$\mathcal{R} \vdash_{\mathcal{B}} -(x \cdot 0) = -(x \cdot 0)$	REFL
(12)	$\mathcal{R} \vdash_{\mathcal{B}} x(0 + 0) + -(x \cdot 0) = (x \cdot 0 + x \cdot 0) + -(x \cdot 0)$	CONGR(10)(11)
(13)	$\mathcal{R} \vdash_{\mathcal{B}} (x \cdot 0 + x \cdot 0) + -(x \cdot 0) = x(0 + 0) + -(x \cdot 0)$	SYM(12)
(14)	$\mathcal{R} \vdash_{\mathcal{B}} x = x$	REFL
(15)	$\mathcal{R} \vdash_{\mathcal{B}} x + 0 = x$	HYP
(16)	$\mathcal{R} \vdash_{\mathcal{B}} 0 + 0 = 0$	INST(14)
(17)	$\mathcal{R} \vdash_{\mathcal{B}} x(0 + 0) = x \cdot 0$	CONGR(14)(16)
(18)	$\mathcal{R} \vdash_{\mathcal{B}} x(0 + 0) + -(x \cdot 0) = x \cdot 0 + -(x \cdot 0)$	CONGR(17)(11)
(19)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 + 0 = x \cdot 0$	INST(15)
(20)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 = x \cdot 0 + 0$	SYM(19)
(21)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 = x \cdot 0 + (x \cdot 0 + -(x \cdot 0))$	TRANS(20)(5)
(22)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 = (x \cdot 0 + x \cdot 0) + -(x \cdot 0)$	TRANS(21)(8)
(23)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 = x(0 + 0) + -(x \cdot 0)$	TRANS(22)(13)
(24)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 = x \cdot 0 + -(x \cdot 0)$	TRANS(23)(18)
(25)	$\mathcal{R} \vdash_{\mathcal{B}} x \cdot 0 = 0$	TRANS(24)(3)

As we can see, this formal derivation in $\mathcal{B}$ is very far from the intuitive mathematical reasoning of Example 1. Actually, that proof corresponds to steps *20* to *25* in the above proof. The remaining steps are just bureaucratic steps, corresponding mainly to the INST, CONGR and REFL rules, which are obviated by the human, but that must be explicitly derived in the formal system. From this simple example, it is clear that $\mathcal{B}$ is not friendly neither for human reasoning nor for interactive proof-search. In particular, it would be very hard to construct the above proof in a backward way.

The above and all proofs in $\mathcal{B}$ can be simplified if we allow the direct replacement of equals by equals
in any given term. This is achieved by the so-called Leibniz rule.

LEMMA 1
The following rule is admissible in $\mathcal{B}$:

PROOF. Induction on the term r. □

It is easy to see that the deductive system obtained by eliminating the CONGR rule in favor of LEIBNIZ results equivalent to system $\mathcal{B}$. However, this choice does not really simplify proofs (for instance, see [6, Section 3.8]). Therefore, we depart from the Birkhoff calculus, and in the next section, we discuss the common approach to equality reasoning in natural deduction systems.

4 $\mathcal{B}^{\text{DEM}}$: a calculus with demodulation

To the best of our knowledge, the just-discussed calculus $\mathcal{B}$ is the only human-oriented [1] deductive system for pure equational logic present in the literature. With respect to equality within first-order logic, the usual approach in natural deduction is to adopt reflexivity as introduction rule and a rule of replacement of equals by equals in any given formula as elimination rule (see e.g. [21, page 107]). These rules, together with the axiom (starting rule) of hypothesis yields the following system, denoted $\mathcal{B}^{\text{DEM}}$:

$$\frac{E \in \Gamma}{\Gamma \vdash_{\mathcal{B}^{\text{DEM}}} E}\ \text{HYP} \qquad \frac{}{\Gamma \vdash_{\mathcal{B}^{\text{DEM}}} t = t}\ \text{REFL}$$

$$\frac{\Gamma \vdash_{\mathcal{B}^{\text{DEM}}} t = s \qquad \Gamma \vdash_{\mathcal{B}^{\text{DEM}}} E[x := t\sigma]}{\Gamma \vdash_{\mathcal{B}^{\text{DEM}}} E[x := s\sigma]}\ \text{DEM}$$

FIGURE 2 System $\mathcal{B}^{\text{DEM}}$.

The elimination rule of equality DEM is also known as substitution or replacement, but, in our opinion, these names can raise ambiguities. Hence, we decide to call it demodulation, for in the case of clauses, it coincides with the rule of demodulation in [39] [2] , as a powerful mechanism in automated theorem proving, see e.g. [26].

Let us now reproduce our Example 1 in $\mathcal{B}^{\text{DEM}}$.

EXAMPLE 6

Let $\mathcal{R}$ be the list of ring axioms defined in the Introduction (1). The following is a proof of $\mathcal{R} \vdash x \cdot 0 = 0$ in $\mathcal{B}^{\text{DEM}}$

(1)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x + -x = 0$	HYP
(2)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x \cdot 0 + -x \cdot 0 = x \cdot 0 + -x \cdot 0$	REFL
(3)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x \cdot 0 + -(x \cdot 0) = 0$	DEM (1), (2) $[x := x \cdot 0]$
(4)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x + 0 = x$	HYP
(5)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x \cdot (0 + 0) = x \cdot (0 + 0)$	REFL
(6)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x \cdot 0 = x \cdot (0 + 0)$	DEM (4), (5) $[x := 0]$
(7)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x \cdot (0 + 0) + -(x \cdot 0) = 0$	DEM (6), (3)
(8)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x \cdot (y + z) = x \cdot y + x \cdot z$	HYP
(9)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} (x \cdot 0 + x \cdot 0) + -(x \cdot 0) = 0$	DEM (8), (7) $[y, z := 0, 0]$
(10)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} (x + y) + z = x + (y + z)$	HYP
(11)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x \cdot 0 + (x \cdot 0 + -(x \cdot 0)) = 0$	DEM (10), (9) $[x, y, z := x \cdot 0, x \cdot 0, x \cdot 0]$
(12)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x \cdot 0 + 0 = 0$	DEM (3), (11)
(13)	$\mathcal{R} \vdash_{\mathcal{B}^{\text{DEM}}} x \cdot 0 = 0$	DEM (4), (12) $[x := x \cdot 0]$

This shows a clear enhancement with respect to the proof of Example 5. Moreover, if we take the step sequence $13, 12, 11, 9, 7, 3$, we recover the backward proof given in Example 2. This means that $\mathcal{B}^{\text{DEM}}$ is a much better candidate to pursue backward proofs than $\mathcal{B}$. Nevertheless, we still have some bureaucratic steps that forbid a backward proof construction, namely those corresponding to the application of the REFL and HYP rules.

We solve this inconvenience with yet another deductive system for equational logic in the next section.

5 $\mathcal{B}^{\text{REW}}$: a calculus with explicit rewrite

Analyzing the proof of Example 6, we realize that one third of the steps correspond to an instance of the HYP rule. The reason is that an application of the DEM rule demands a proof of the equation to perform the substitution, an equation that in several cases, as in the example, is already part of the context. Nevertheless, the shape of the inference rules in $\mathcal{B}^{\text{DEM}}$ do not let us take advantage of that

$$\frac{}{\Gamma \vdash t = t}\ \textsc{Refl} \qquad \frac{\Gamma,\, t = s,\, \Gamma' \vdash E[x := t\sigma]}{\Gamma,\, t = s,\, \Gamma' \vdash E[x := s\sigma]}\ \textsc{Rew}$$

FIGURE 3 System $\mathcal{B}^{\textsc{Rew}}$.

fact and it is then mandatory to apply the HYP rule to derive the desired equation. Thus, the only way to use a context equation is by first explicitly deriving it using the HYP rule. This is unpleasant in comparison with the Fitch's approach where the proofs involve just formulas not sequents. In that style, the assumptions are directly available and it is harmless to start a forward proof by just listing them. In contrast, in a sequent system it is difficult to know a priori which context Γ, having E as a member, is the more adequate to add an instance of the HYP rule to a proof. This is a problem for forward proofs, but as we will see later we would like to eliminate this rule for forward proof construction, due to the fact that we do not know a priori the adequate context to derive a hypothesis.

The solution to this inconvenience is provided by allowing the direct use of a hypothesis in the context. This is exactly what is done in the left rules of sequent calculi. Such inference schemes are what we call hypothesis-aware rules. More examples of this kind of rules appear in [1, 31]. We present now a deductive system $\mathcal{B}^{\textsc{Rew}}$ for equational logic with a hypothesis-aware rule for rewriting and with no HYP rule:

We reproduce now Example 1 in $\mathcal{B}^{\textsc{Rew}}$.

EXAMPLE 7

Let Γ be the following list of ring axioms:

$$\Gamma = [R1,\, R1^{Sym},\, R3,\, R3^{Sym},\, R4,\, R4^{Sym},\, R6^{Sym}],$$

where E^{Sym} is the symmetric equation of E.

The following are two proofs of $\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 = 0$:

(1)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 = x \cdot 0$	REFL
(2)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 = x \cdot 0 + 0$	REW $(R1^{Sym}, [x := x \cdot 0])$ (1)
(3)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 = x \cdot 0 + (x \cdot 0 + -(x \cdot 0))$	REW $(R3^{Sym}, [x := x \cdot 0])$ (2)
(4)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 = (x \cdot 0 + x \cdot 0) + -(x \cdot 0)$	REW $(R4, [x, y, z := x \cdot 0, x \cdot 0, -(x \cdot 0)])$ (3)
(5)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 = (x \cdot (0 + 0)) + -(x \cdot 0)$	REW $(R6^{Sym}, [y, z := 0, 0])$ (4)
(6)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 = (x \cdot 0) + -(x \cdot 0)$	REW $(R1^{Sym}, [x := 0])$ (5)
(7)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 = 0$	REW $(R3, [x := x \cdot 0])$ (6)

(1)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 + -x \cdot 0 = x \cdot 0 + -x \cdot 0$	REFL
(2)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 + -x \cdot 0 = 0$	REW $(R3^{Sym}, [x := x \cdot 0])$ (1)
(3)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot (0 + 0) + -x \cdot 0 = 0$	REW $(R1^{Sym}, [x := 0])$ (2)
(4)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} (x \cdot 0 + x \cdot 0) + -x \cdot 0 = 0$	REW $(R6^{Sym}, [y, z := 0, 0])$ (3)
(5)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 + (x \cdot 0 + -x \cdot 0) = 0$	REW $(R3, [x, y, z := x \cdot 0, x \cdot 0, -x \cdot 0])$ (4)
(6)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 + 0 = 0$	REW $(R3, [x := x \cdot 0])$ (5)
(7)	$\Gamma \vdash_{\mathcal{B}^{\textsc{Rew}}} x \cdot 0 = 0$	REW $(R1, [x := x \cdot 0])$ (6)

This example shows that the current system $\mathcal{B}^{\textsc{Rew}}$ is closer to the informal equality reasoning. The former proof corresponds to the forward proof of Example 1, whereas the latter, when read backwards, is essentially the proof of Example 2. Moreover, there is no need of explicitly prove a

hypothesis and therefore the proofs are considerably shorter. The hypotheses-awareness of the system allows to have the HYP rule as an admissible rule.

PROPOSITION 1
The HYP rule

$$\frac{E \in \Gamma}{\Gamma \vdash_{\mathcal{B}^{\mathrm{REW}}} E}\mathrm{HYP}$$

is admissible in $\mathcal{B}^{\mathrm{REW}}$.

PROOF. Let $E =_{def} t = s$ and assume $E \in \Gamma$. In such case, there are contexts Γ' and Γ'' such that Γ is the same as Γ', E, Γ''. The following is a proof of $\Gamma \vdash_{\mathcal{B}^{\mathrm{REW}}} E$:

$$\begin{array}{lll} (1) & \Gamma',\, t = s,\, \Gamma'' \vdash_{\mathcal{B}^{\mathrm{REW}}} t = t & \textsc{Refl} \\ (2) & \Gamma',\, t = s,\, \Gamma'' \vdash_{\mathcal{B}^{\mathrm{REW}}} t = s & \textsc{Rew}\ (t = s,\, [x := s])\ (1) \end{array}$$

□

In spite of the benefits of this system, we still have an issue: the need to use an equation in both directions. The example above uses the sum identity and inverse axioms. This is so, for the REW rule only allows the rewriting from left to right. To avoid this concern, we can add either the rule of symmetry or a rule that allows the rewriting in the other direction. Both rules are admissible in $\mathcal{B}^{\mathrm{REW}}$ but the second choice follows closer the spirit of this system and is a direct consequence of the structural rules of weakening and cut (see Lemma 3 and Theorem 2).

LEMMA 2
The following rule is admissible in $\mathcal{B}^{\mathrm{REW}}$:

$$\frac{\Gamma,\, t = s, \Gamma' \vdash_{\mathcal{B}^{\mathrm{REW}}} E[x := s\sigma]}{\Gamma,\, t = s, \Gamma' \vdash_{\mathcal{B}^{\mathrm{REW}}} E[x := t\sigma]}\mathrm{REWL}$$

PROOF.

$$\begin{array}{lll} (1) & \Gamma,\, t = s, \Gamma' \vdash_{\mathcal{B}^{\mathrm{REW}}} t = t & \textsc{Refl} \\ (2) & \Gamma,\, t = s, \Gamma' \vdash_{\mathcal{B}^{\mathrm{REW}}} s = t & \textsc{Rew}\ (t = s\,,\, [x := s])\ (1) \\ (3) & \Gamma,\, t = s, \Gamma' \vdash_{\mathcal{B}^{\mathrm{REW}}} E[x := s\sigma] & \textit{Given} \\ (4) & \Gamma,\, t = s, \Gamma',\, s = t \vdash_{\mathcal{B}^{\mathrm{REW}}} E[x := s\sigma] & \textsc{Weak}\ (3) \\ (5) & \Gamma,\, t = s, \Gamma',\, s = t \vdash_{\mathcal{B}^{\mathrm{REW}}} E[x := t\sigma] & \textsc{Rew}\ (s = t\,,\, [\,])\ (4) \\ (6) & \Gamma,\, t = s, \Gamma' \vdash_{\mathcal{B}^{\mathrm{REW}}} E[x := t\sigma] & \textsc{Cut}\ (2), (5) \end{array}$$

□

With the rule REWL available, the proof of Example 7 can easily be adapted to witness that $\mathcal{R} \vdash_{\mathcal{B}^{\mathrm{REW}}} x \cdot 0 = 0$. Let us now reproduce Example 3 within $\mathcal{B}^{\mathrm{REW}}$. The use of a lemma is formalized by means of the cut rule.

EXAMPLE 8
We will construct a derivation of

$$\mathcal{R},\ x = x \cdot x \vdash_{\mathcal{B}^{\mathrm{REW}}} x + y = x + (y + (x \cdot y + y \cdot x)).$$

For this purpose we use the lemma $x+(y+z) = y+(x+z)$. That is, we use the following instance of the cut rule

$$\frac{\mathcal{R}, x = x\cdot x,\ x+(y+z) = y+(x+z) \ \vdash_{\mathcal{B}^{\text{REW}}} x+y = x+(y+(x\cdot y+y\cdot x)) \qquad \mathcal{R}, x = x\cdot x \vdash_{\mathcal{B}^{\text{REW}}} x+(y+z) = y+(x+z)}{\mathcal{R},\ x = x\cdot x \vdash_{\mathcal{B}^{\text{REW}}} x+y = x+(y+(x\cdot y+y\cdot x))}\ \text{CUT}$$

We show next only the derivation of the first premise, leaving the second as exercise. Let $\mathcal{R}' =_{def} \mathcal{R}, E_1, E_2$, where $E_1 =_{def} x = x\cdot x$ and $E_2 =_{def} x+(y+z) = y+(x+z)$. The following is a derivation of

$$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = x+(y+(x\cdot y+y\cdot x)).$$

(1)	$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = x+y$	REFL
(2)	$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = (x+y)(x+y)$	REW $(E_1, [x := x+y])$ (1)
(3)	$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = x(x+y)+y(x+y)$	REW $(R7, [x,y,z := x,y,x+y])$ (2)
(4)	$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = (xx+xy)+y(x+y)$	REW $(R6, [x,y,z := x,x,y])$ (3)
(5)	$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = (x+xy)+y(x+y)$	REW $(E_1, [\,])$ (4)
(6)	$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = (x+xy)+(yx+yy)$	REW $(R6, [x,y,z := y,x,y])$ (5)
(7)	$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = (x+xy)+(yx+y)$	REW $(E_1, [x := y])$ (6)
(8)	$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = (x+xy)+(y+yx)$	REW $(R2, [x,y := yx,y])$ (7)
(9)	$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = x+(xy+(y+yx))$	REW $(R4, [x,y,z := x,xy,y+yx])$ (8)
(10)	$\mathcal{R}' \vdash_{\mathcal{B}^{\text{REW}}} x+y = x+(y+(xy+yx))$	REW $(E_2, [x,y,z := y,xy,yx])$ (9)

As final example, we give a formal derivation of our functional programming example presented in Section 1.

EXAMPLE 9

Let Γ be the list of definitions and laws in Example 4, we prove that

$$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} cross(map\, f, map\, g)\,.\,unzip = unzip\ .\ map(cross(f,g)).$$

The details regarding specific substitution in each step are left to the reader.

(1)	$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} cross(map\, f, map\, g)\,.\,unzip =$ $cross(map\, f, map\, g)\,.\,unzip$ $r =_{def} cross(map\, f, map\, g)\,.\,unzip$	REFL
(2)	$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r = cross(map\, f, map\, g)\,.fork(map\, fst, map\, snd)$	REW $D1$
(3)	$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r = fork(map\, f\ .\ map\, fst\ ,\ map\, g\ .\ map\, snd)$	REW $Law1$
(4)	$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r = fork(map\ (f\ .fst)\ ,\ map\ (g\ .snd))$	REW $Law5$
(5)	$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r = fork(map\ (fst\ .\ cross(f,g))\ ,\ map\ (g\ .snd))$	REW $Law3$
(6)	$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r =$ $fork(map\ (fst\ .\ cross(f,g))\ ,\ map\ (snd\ .\ cross(f,g)))$	REW $Law4$
(7)	$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r =$ $fork(map\, fst\ .\ map\ (cross(f,g))\ ,\ map\, snd\ .\ map\ (cross(f,g)))$	REW $Law5$
(8)	$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r = fork(map\, fst\ ,\ map\, snd)\ .\ map\ (cross(f,g))$	REW $Law2$
(9)	$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} cross(map\, f, map\, g)\,.\,unzip = unzip\,.\,map\ (cross(f,g))$	REW $D1$

The above examples provide evidence about the adequacy of $\mathcal{B}^{\text{REW}}$, when including some convenient admissible rules, to construct backward proofs. But before pursuing this matter further, it is important to ask ourselves about the exact relation between the three systems $\mathcal{B}$, $\mathcal{B}^{\text{DEM}}$ and $\mathcal{B}^{\text{REW}}$. We do this in the next section.

6 Equivalence between equational calculi

The systems presented so far are specialized to perform equational reasoning in a structured way. Though the inference rules of $\mathcal{B}$ seem adequate, the demonstrations are neither intuitive nor easy to construct using a backward reasoning. While $\mathcal{B}^{\text{DEM}}$ simplify the proofs, it does not ease a backward proof construction because of the enforced use of the REFL and HYP rules, usually required by the DEM rule. The third system, $\mathcal{B}^{\text{REW}}$, is a more comfortable environment where the backward proof construction process is closer to the routinary mathematical arguments. Despite this, $\mathcal{B}$ holds a very important property, namely its completeness with respect to the equational fragment of first-order logic. A powerful attribute that, as we are about to show, is shared by the systems $\mathcal{B}^{\text{DEM}}$ and $\mathcal{B}^{\text{REW}}$.

The desired completeness is an easy consequence of the equivalence of our deductive systems. In order to prove the equivalence, some structural rules are needed. They are all admissible in the given systems.

LEMMA 3 (Structural rules).
The rules of weakening, contraction and exchange

$$\frac{\Gamma \vdash s = t}{\Gamma, E \vdash t = s}\text{WEAK} \qquad \frac{\Gamma, E, E \vdash s = t}{\Gamma, E \vdash t = s}\text{CONTR} \qquad \frac{\Gamma, E, E' \vdash s = t}{\Gamma, E', E \vdash t = s}\text{EXCH}$$

are admissible.

PROOF. Straightforward induction on the given derivation, regarding the system in which the rule is been proved. □

We proceed by proving first the equivalence between systems $\mathcal{B}$ and $\mathcal{B}^{\text{DEM}}$.

LEMMA 4
The demodulation rule is admissible in the Birkhoff calculus $\mathcal{B}$:

$$\frac{\Gamma w \vdash_{\mathcal{B}} t = s \qquad \Gamma \vdash_{\mathcal{B}} E[x := t\sigma]}{\Gamma \vdash_{\mathcal{B}} E[x := s\sigma]}\text{DEM}$$

PROOF. This is a straightforward consequence of Lemma 1 (Leibniz rule). □

LEMMA 5
The rules for symmetry SYM, transitivity TRANS, instanceINST and congruence CONGR are admissible in the calculus $\mathcal{B}^{\text{DEM}}$.

PROOF. The proofs are straightforward. Let us show the case for the instance rule INST. Suppose that $\Gamma \vdash_{\mathcal{B}} t = s$, then we want to proof that $\Gamma \vdash_{\mathcal{B}} t\sigma = s\sigma$. By REFL, we derive $\Gamma \vdash_{\mathcal{B}} t\sigma = t\sigma$ which is the same as $\Gamma \vdash_{\mathcal{B}} (t\sigma = x)[x := t\sigma]$. By applying the DEM rule, we obtain the desired sequent $\Gamma \vdash_{\mathcal{B}} t\sigma = s\sigma$. □

PROPOSITION 2
The calculi $\mathcal{B}$ and $\mathcal{B}^{\text{DEM}}$ are equivalent.

PROOF. Immediate from Lemmas 4 and 5. □

We pursue now the equivalence between $\mathcal{B}$ and $\mathcal{B}^{\text{REW}}$, starting with a technical lemma on substitutions.

LEMMA 6
If $x \neq z$ and $x \notin Var(t)$ then $t[z := r[x := s]] = t[z := r][x := s]$.

PROOF. Straightforward induction on t. □

The Leibniz principle of replacement equals by equals is admissible in system $\mathcal{B}^{\text{REW}}$:

LEMMA 7
The following rule is admissible in $\mathcal{B}^{\text{Rew}}$

$$\frac{\Gamma \vdash_{\mathcal{B}^{\text{REW}}} t = s}{\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r[x := t] = r[x := s]}\text{LEIBNIZ}.$$

PROOF. Induction on $\Gamma \vdash t = s$. The interesting case is the one for (REW), which follows from the I.H by means of Lemma 6. □

It is important to remark that a proof of Lemma 6 by induction on the term r, like in the case of system $\mathcal{B}$ (Lemma 1), does not go through for $\mathcal{B}^{\text{REW}}$.

LEMMA 8
The rules for symmetry SYM, transitivity TRANS and instance INST are admissible in the calculus $\mathcal{B}^{\text{REW}}$.

PROOF. Transitivity is proved by induction on the first premise, symmetry is required. Instance and symmetry are proved by induction on the corresponding premise. Let us show the case for symmetry and assume that $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} t = s$. The base step is obvious, for in this case t and s are the same term. For the inductive step, we have that $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} t = s$ by the REW rule. This means that $t = s$ has the form $(r = u)[x := t_2\sigma]$ where $t_1 = t_2 \in \Gamma$ and $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} (r = u)[x := t_1\sigma]$. In particular, t is exactly $r[x := t_2\sigma]$ and s is the same as $u[x := t_2\sigma]$. Thus, we need to show that $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} u[x := t_2\sigma] = r[x := t_2\sigma]$, but this is a direct consequence of the induction hypothesis $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} u[x := t_1\sigma] = r[x := t_1\sigma]$, by REW using again $t_1 = t_2$. □

Next, we show the validity of the cut structural rule in $\mathcal{B}^{\text{REW}}$.

THEOREM 2
The cut rule

$$\frac{\Gamma,\ E \vdash_{\mathcal{B}^{\text{REW}}} t = s \qquad \Gamma \vdash_{\mathcal{B}^{\text{REW}}} E}{\Gamma \vdash_{\mathcal{B}^{\text{REW}}} t = s}\text{CUT}$$

is admissible in $\mathcal{B}^{\text{REW}}$.

PROOF. By induction on $\Gamma,\ E \vdash_{\mathcal{B}^{\text{REW}}} t = s$. The base case (reflexivity) is obvious since the conclusion is again an instance of reflexivity. Let us assume now that $\Gamma,\ E \vdash_{\mathcal{B}^{\text{REW}}} t = s$ comes from an application of the rule REW. Then, $t = s$ has the form $(r = u)[x := t_2\sigma]$ where $t_1 = t_2 \in \Gamma, E$ and $\Gamma, E \vdash_{\mathcal{B}^{\text{REW}}} (r = u)[x := t_1\sigma]$. In particular, t is exactly $r[x := t_2\sigma]$ y s is the same as $u[x := t_2\sigma]$. Thus, we need to show that $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r[x := t_2\sigma] = u[x := t_2\sigma]$. Further, the I.H. indicates that $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r[x := t_1\sigma] = u[x := t_1\sigma]$. Since $t_1 = t_2 \in \Gamma, E$, we need to analyze two cases:

1. $t_1 = t_2 \in \Gamma$. This case is immediate from the I.H., by the REW rule using again $t_1 = t_2$.
2. $t_1 = t_2$ is exactly E. In this case, the lateral hypothesis says that $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} t_1 = t_2$, which by the INST rule (Lemma 8), yields $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} t_1\sigma = t_2\sigma$. From this, LEIBNIZ rule yields

$\Gamma \vdash_{\mathcal{B}^{\text{REW}}} u[x := t_1\sigma] = u[x := t_2\sigma]$ and by transitivity with the I.H we get $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r[x := t_1\sigma] = u[x := t_2\sigma]$. Again, by LEIBNIZ, we get $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r[x := t_1\sigma] = r[x := t_2\sigma]$. Finally, from the last two sequents, using symmetry and transitivity, we arrive to $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} r[x := t_2\sigma] = u[x := t_2\sigma]$. □

As a corollary of CUT, we obtain the admissibility of the CONGR rule.

LEMMA 9
The congruence rule

$$\frac{\Gamma \vdash_{\mathcal{B}^{\text{REW}}} t_1 = s_1 \quad \ldots \quad \Gamma \vdash_{\mathcal{B}^{\text{REW}}} t_n = s_n}{\Gamma \vdash_{\mathcal{B}^{\text{REW}}} f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)}\text{CONGR}$$

is admissible in $\mathcal{B}^{\text{REW}}$.

PROOF. Assume that $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} t_1 = s_1 \quad \ldots \quad \Gamma \vdash_{\mathcal{B}^{\text{REW}}} t_n = s_n$. By reflexivity and iterated applications of REW we get that

$$\Gamma, t_1 = s_1, \ldots, t_n = s_n \vdash_{\mathcal{B}^{\text{REW}}} f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n).$$

Finally, $\Gamma \vdash_{\mathcal{B}^{\text{REW}}} f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)$ is obtained by iterating the CUT rule using the assumptions. □

LEMMA 10
Demodulation is admissible in $\mathcal{B}^{\text{REW}}$.

PROOF. Straightforward with help of weakening 3 and cut 2 rules. □

We have now all ingredients to show the equivalence between $\mathcal{B}$ and $\mathcal{B}^{\text{REW}}$.

PROPOSITION 3
The calculi $\mathcal{B}$ and $\mathcal{B}^{\text{REW}}$ are equivalent.

PROOF. As a consequence of Proposition 1 and Lemmas 8, 9, any proof of $\mathcal{B}$ can be simulated in $\mathcal{B}^{\text{REW}}$. On the other direction, by Proposition 2, 1 and Lemma 10, any proof of $\mathcal{B}^{\text{REW}}$ can be simulated in $\mathcal{B}$. □

Propositions 2 and 3 ensure that the three systems presented so far are equivalent and hence the property of completeness with respect to the equational fragment of first-order logic is preserved. We continue to the next section where a system devoted to goal-oriented proofs for equational reasoning is developed.

7 $\mathcal{B}^{\text{GOAL}}$: A calculus for interactive theorem proving

Our last system $\mathcal{B}^{\text{REW}}$ is already designed with a practical use in mind, for it captures the usual equational reasoning employed in mathematics, namely the use of a particular equation available in the current information context and with an implicit instantiating process. However, this new system still has some formal restrictions that difficult its practical use, for instance, rewrite is the only available rule and only allows the rewriting using the last equation in the context. This is so, for our contexts are lists and not sets or multisets. This problem can be solved with the explicit use of

structural rules, which are inadequate for proof-search, or by defining a system that hard-wires such rules. This is the option taken here.

Considering this, we define next another system $\mathcal{B}^{\text{GOAL}}$ that includes some inference rules, which, though derivable in $\mathcal{B}^{\text{REW}}$, are common in equational reasoning. This choice is questionable from the point of view of mathematical logic, where a system with a minimum number of inference rules is preferred. However, for a practical system, it is mandatory to make available a large number of reasoning strategies that should be implemented as primitives. Moreover, to facilitate the backward reasoning, we also introduce another useful device to the notion of derivation, namely labeled hypotheses [14]. In everyday mathematical reasoning, it is common to label some hypotheses, to refer to them later on the proof, just as we did in Example 1.3. This feature can be easily added to our formalism by modifying the definition of context as follows: *a labeled context Γ is a list of labeled hypotheses of the form $\Gamma = [H_1 : E_1, \ldots, H_n : E_n]$ where $H_i \neq H_j$, if $i \neq j$ and $H_i : E_i$ indicates that the label H_i is a name or shortcut to refer to the equation E_i.* The set of labels is, of course, disjoint with the names in the current signature. In particular, in a context of the form $\Gamma, H : E, \Gamma'$, the label H does not occur in Γ, Γ'. The use of labels does not contribute much to the usual system of forward derivations, but as we will see soon, it is very useful to the backward approach.

The system $\mathcal{B}^{\text{GOAL}}$ is defined by the inference rules in Figure 4. We include the basic rules of reflexivity, symmetry and transitivity of $\mathcal{B}$: the rule of demodulation of $\mathcal{B}^{\text{DEM}}$ in two versions corresponding to both directions of equality, and the rewrite rules of $\mathcal{B}^{\text{REW}}$, also in both directions. Moreover, the rules REWIN,REWINL allow us to perform a rewriting within the context and the INSHYP rule concedes the immediate inference of an instance of a context equation. The structural rules of cut and weakening are present due to their importance in practice. Also, observe that contexts are written in their most general form, making available the immediate use of any hypothesis. Certainly, this system is equivalent to the previous ones.

PROPOSITION 4
The deductive system $\mathcal{B}^{\text{GOAL}}$ is equivalent to $\mathcal{B}^{\text{REW}}$.

PROOF. It is clear that every proof in $\mathcal{B}^{\text{REW}}$ can be replicated in $\mathcal{B}^{\text{GOAL}}$. On the other hand, it is easy to see that the rules REWIN, REWINL and INSHYP are derivable in $\mathcal{B}^{\text{REW}}$. This and previous results of Section 6 yield the desired equivalence. □

We claim that $\mathcal{B}^{\text{GOAL}}$ is more adequate for interactive theorem proving than the previous formalisms, in the sense that it has all their functionalities and allows a direct management of the contexts which in turn simplifies the construction of backwards proofs in a direct way, i.e. without any need for forward reasoning. The foundation of this proof construction process lies in three heuristics, called conclusion analysis (CA), premise analysis (PA) and lemma assertion (LA), see [29]. To search for a proof of the current goal $\Gamma \vdash E$ by means of CA, we analyze the equation E and look for some information that allows its derivation, for instance the use of transitivity. On the other hand, by using the PA heuristic, we focus on a specific equation E' belonging to the current context Γ which, according to our intuition and previously derived goals, can help us prove the original equation E. This analysis corresponds to the backward use of a rewriting rule. Finally, the (LA) heuristic allows the use of a lemma to achieve the current goal and corresponds to the backward reading of the cut rule.

With these heuristics, we might initiate the development of backward derivations in $\mathcal{B}^{\text{GOAL}}$ by a rigorous, though semi-formal, manner, like the one described in Kanger's articles [24, 25] (see also [29, Section 2.1]). Instead, we define this process precisely in the next section.

$$\frac{}{\Gamma,\ H : E,\ \Gamma' \vdash_{\mathcal{B}\textsc{Goal}} E}\ \textsc{Hyp} \qquad \frac{}{\Gamma \vdash_{\mathcal{B}\textsc{Goal}} t = t}\ \textsc{Refl} \qquad \frac{\Gamma \vdash_{\mathcal{B}\textsc{Goal}} s = t}{\Gamma \vdash_{\mathcal{B}\textsc{Goal}} t = s}\ \textsc{Sym}$$

$$\frac{\Gamma \vdash_{\mathcal{B}\textsc{Goal}} t = r \qquad \Gamma \vdash_{\mathcal{B}\textsc{Goal}} r = s}{\Gamma \vdash_{\mathcal{B}\textsc{Goal}} t = s}\ \textsc{Trans}$$

$$\frac{\Gamma \vdash_{\mathcal{B}\textsc{Goal}} t = s \qquad \Gamma \vdash_{\mathcal{B}\textsc{Goal}} E[x := t\sigma]}{\Gamma \vdash_{\mathcal{B}\textsc{Goal}} E[x := s\sigma]}\ \textsc{Dem}$$

$$\frac{\Gamma \vdash_{\mathcal{B}\textsc{Goal}} t = s \qquad \Gamma \vdash_{\mathcal{B}\textsc{Goal}} E[x := s\sigma]}{\Gamma \vdash_{\mathcal{B}\textsc{Goal}} E[x := t\sigma]}\ \textsc{DemL}$$

$$\frac{\Gamma,\ H : t = s,\ \Gamma' \vdash_{\mathcal{B}\textsc{Goal}} E[x := t\sigma]}{\Gamma,\ H : t = s,\ \Gamma' \vdash_{\mathcal{B}\textsc{Goal}} E[x := s\sigma]}\ \textsc{Rew}$$

$$\frac{\Gamma,\ H : t = s,\ \Gamma' \vdash_{\mathcal{B}\textsc{Goal}} E[x := s\sigma]}{\Gamma,\ H : t = s,\ \Gamma' \vdash_{\mathcal{B}\textsc{Goal}} E[x := t\sigma]}\ \textsc{RewL}$$

$$\frac{\Gamma, H : E[x := t\sigma], \Gamma', H' : t = s, \Gamma'' \vdash_{\mathcal{B}\textsc{Goal}} E'}{\Gamma, H : E[x := s\sigma], \Gamma', H' : t = s, \Gamma'' \vdash_{\mathcal{B}\textsc{Goal}} E'}\ \textsc{RewIn}$$

$$\frac{\Gamma, H : E[x := s\sigma], \Gamma', H' : t = s, \Gamma'' \vdash_{\mathcal{B}\textsc{Goal}} E'}{\Gamma, H : E[x := t\sigma], \Gamma', H' : t = s, \Gamma'' \vdash_{\mathcal{B}\textsc{Goal}} E'}\ \textsc{RewInL}$$

$$\frac{}{\Gamma, H : t = s, \Gamma' \vdash_{\mathcal{B}\textsc{Goal}} t\sigma = s\sigma}\ \textsc{InsHyp H} \qquad \frac{\Gamma, \Gamma' \vdash_{\mathcal{B}\textsc{Goal}} t = s}{\Gamma,\ H : E,\ \Gamma' \vdash_{\mathcal{B}\textsc{Goal}} t = s}\ \textsc{Weak}$$

$$\frac{\Gamma,\ H : E \vdash_{\mathcal{B}\textsc{Goal}} t = s \qquad \Gamma \vdash_{\mathcal{B}\textsc{Goal}} E}{\Gamma \vdash_{\mathcal{B}\textsc{Goal}} t = s}\ \textsc{Cut}$$

FIGURE 4 $\mathcal{B}^{\textsc{Goal}}$ system.

8 Tactics

The concept of tactic in theorem proving originates in the seminal work of Milner and colleagues [18]. There, a tactic is defined as a function that breaks goals into some simpler goals, called subgoals, and must be accompanied by a validation function. Indeed, tactic application corresponds to backward proof construction where the initial goal is a given conjecture or specification. In modern proof assistants, a (backward) proof corresponds to a sequence of tactics introduced by the human user in an essential interaction with the system and hence these assistants are now also named interactive theorem provers [31]. When the interaction succeeds, a usual (forward) proof is automatically constructed by the assistant from the backward proof.

However, to the best of our knowledge, there is no formal high-level [3] definition, neither of tactic nor of a proof by tactics. Perhaps because it seems a concept of mere implementation.

In this section, we provide the main contribution of this paper: the definition of a transition system $\mathcal{T}_{Eq}$ of reasoning tactics for equational logic that allows for a formal definition which captures the intuitive notion of backward proof, in the sense employed by Kanger [25]. This transition system allows us to construct $\mathcal{B}^{\text{GOAL}}$ proofs in a backward way exclusively, i.e. without resorting to any forward reasoning, thus departing from related work of e.g [13, 33, 34]. The following results extend our previous work on interactive proof-search for minimal propositional logic [29].

DEFINITION 6
A goal $\mathcal{G}$ is any judgment $\Gamma \vdash E$.[4] The set of finite sequences of goals *GSeq* is recursively defined as follows:

$$\mathcal{S} ::= \square \mid \mathcal{G} \,;\, \mathcal{S},$$

where $\square$ denotes the empty goal sequence. Moreover, if $\mathcal{S}_1, \mathcal{S}_2 \in GSeq$ then by $\mathcal{S}_1; \mathcal{S}_2$ we mean the append of $\mathcal{S}_2$ at the end of $\mathcal{S}_2$.

Let us define now our transition system of tactics.

DEFINITION 7
The transition system of tactics for equational logic is the tuple $\mathcal{T}_{Eq} = \langle GSeq, \{\square\}, \rhd \rangle$ where

- the non-empty set of states is the set of goal sequences *GSeq*;
- an initial state is any sequent [5] $\Gamma \vdash E$;
- $\square$ is the unique terminal state;
- the transition relation $\rhd \subseteq GSeq \times GSeq$ is inductively defined by the below axioms and inference rules, where a transition $\mathcal{S}_1 \rhd \mathcal{S}_2$ can be read as 'to prove the sequents in $\mathcal{S}_1$ it suffices to prove the sequents in $\mathcal{S}_2$'.

- CA:

$$\begin{array}{lcll} \Gamma \vdash s = t & \rhd & \Gamma \vdash t = s & (\texttt{symmetry}) \\ \Gamma \vdash s = t & \rhd & \Gamma \vdash s = r \,;\, \Gamma \vdash r = t & (\texttt{transitivity r}) \end{array}$$

PA:

$$\begin{array}{lcll} \Gamma, H : s = t, \Gamma' \vdash E[s] & \rhd & \Gamma, H : s = t, \Gamma' \vdash E[t] & (\texttt{rewrite H}) \\ \Gamma, H : s = t, \Gamma' \vdash E[t] & \rhd & \Gamma, H : s = t, \Gamma' \vdash E[s] & (\texttt{rewrite} \leftarrow \texttt{H}) \\ \Gamma, H : E', \Gamma' \vdash E & \rhd & \Gamma, \Gamma' \vdash E & (\texttt{clear H}) \end{array}$$

LA:

$$\begin{array}{lcll} \Gamma \vdash E' & \rhd & \Gamma \vdash E \,;\, \Gamma, H : E \vdash E' & (\texttt{assert E}) \\ \Gamma \vdash E[t] & \rhd & \Gamma \vdash t = s \,;\, \Gamma \vdash E[s] & (\texttt{demodulation}) \\ \Gamma \vdash E[s] & \rhd & \Gamma \vdash t = s \,;\, \Gamma \vdash E[t] & (\texttt{demodulation} \leftarrow) \end{array}$$

[4] To simplify the notation in this section, we use $\vdash$ instead of $\vdash_{\mathcal{B}^{\text{GOAL}}}$.

Discarding tactics:

$$\begin{array}{rcll} \Gamma, H : E, \Gamma' \vdash E & \rhd & \square & (\texttt{assumption / exactH}) \\ \Gamma, H : s = t, \Gamma' \vdash s\sigma = t\sigma & \rhd & \square & (\texttt{apply H}) \\ \Gamma \vdash t = t & \rhd & \square & (\texttt{reflexivity}) \end{array}$$

Appending rule:

$$\frac{\mathcal{S}_1 \rhd \mathcal{S}_2}{\mathcal{S}_1; \mathcal{S} \rhd \mathcal{S}_2; \mathcal{S}}(\texttt{app})$$

A basic transition transforms a unitary goal sequence into a, perhaps empty, sequence of subgoals dictated by the backwards reading of an inference rule of $\mathcal{B}^{\text{GOAL}}$. The (app) rule determines the order in which subgoals are solved, namely from the first (most left) goal in the current list of pending subgoals.

Next, we give the promised definition of a backward proof.

DEFINITION 8

A backward proof of a judgment $\Gamma \vdash E$ is a finite sequence of states $\mathcal{S}_1, \ldots, \mathcal{S}_k$ such that

- $\mathcal{S}_1 =_{def} \Gamma \vdash E$
- For every $1 \leq i < k$, $\mathcal{S}_i \rhd \mathcal{S}_{i+1}$
- $\mathcal{S}_k =_{def} \square$

Therefore, a backward proof of a judgment $\Gamma \vdash E$ is a finite sequence of tactics that ends in the empty sequence of goals $\square$, meaning that the original goal $\Gamma \vdash E$ has no pending subgoals left to prove.

We now show backward derivations corresponding to Examples 1, 3 and 4.

EXAMPLE 10

The following is a derivation by tactics of $\mathcal{R} \vdash x \cdot 0 = 0$ where $\mathcal{R}$ is the list of ring axioms in page 7.

$$\begin{array}{l} \mathcal{R} \vdash x \cdot 0 = 0 \\ \rhd \qquad \texttt{rewrite} \leftarrow \quad (R3\,, [x := x \cdot 0]) \\ \mathcal{R} \vdash x \cdot 0 = x \cdot 0 + -(x \cdot 0) \\ \rhd \qquad \texttt{rewrite} \quad (R1^{Sym}\,, [x := 0]) \\ \mathcal{R} \vdash x \cdot 0 = x \cdot (0 + 0) + -(x \cdot 0) \\ \rhd \qquad \texttt{rewrite} \leftarrow \quad (R6^{Sym}\,, [y, z := 0, 0]) \\ \mathcal{R} \vdash x \cdot 0 = (x \cdot 0 + x \cdot 0) + -(x \cdot 0) \\ \rhd \qquad \texttt{rewrite} \quad \big(R4\,, [x, y, z := x \cdot 0, x \cdot 0, -(x \cdot 0)]\big) \\ \mathcal{R} \vdash x \cdot 0 = x \cdot 0 + (x \cdot 0 + -(x \cdot 0)) \\ \rhd \qquad \texttt{rewrite} \leftarrow \quad (R3^{Sym}\,, [x := x \cdot 0]) \\ \mathcal{R} \vdash x \cdot 0 = x \cdot 0 + 0 \\ \rhd \qquad \texttt{rewrite} \leftarrow \quad (R1^{Sym}\,, [x := x \cdot 0]) \\ \mathcal{R} \vdash x \cdot 0 = x \cdot 0 \\ \rhd \qquad \texttt{reflexivity} \\ \square \end{array}$$

EXAMPLE 11
The following is a backward derivation of

$$\mathcal{R},\ x \cdot x = x \ \vdash x + y = x + (y + (x \cdot y + y \cdot x))$$

making use of a derivation *DL* of the sequent $\mathcal{R},\ x \cdot x = x \ \vdash x + (y + z) = y + (x + z)$, that we leave to the reader. Let $\mathcal{R}' =_{def} \mathcal{R},\ H0 : x * x = x$.

$\mathcal{R}' \vdash x + y = x + (y + (x \cdot y + y \cdot x))$
$\rhd$ `assert`$(H1 : x + (y + z) = y + (x + z))$
$\mathcal{R}' \vdash x + (y + z) = y + (x + z)$;
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = x + (y + (x \cdot y + y \cdot x))$
$\rhd \ldots \rhd$ *DL*
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = x + (y + (x \cdot y + y \cdot x))$
$\rhd$ `rewrite` $(H1\,, [x, y, z := y, x \cdot y, y \cdot x])$
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = x + (x \cdot y + (y + y \cdot x))$
$\rhd$ `rewrite` $\leftarrow$ $(R4\,, [x, y, z := x \cdot y, y, y \cdot x])$
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = (x + x \cdot y) + (y + y \cdot x)$
$\rhd$ `rewrite` $(R2\,, [x, y := y, y \cdot x])$
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = (x + x \cdot y) + (y \cdot x + y)$
$\rhd$ `rewrite` $\leftarrow$ $(H0\,, [x := y])$
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = (x + x \cdot y) + (y \cdot x + y \cdot y)$
$\rhd$ `rewrite` $\leftarrow$ $(R6\,, [x, y, z := y, x, y])$
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = (x + x \cdot y) + (y \cdot (x + y))$
$\rhd$ `rewrite` $\leftarrow$ $[H0\,, [\,]]$
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = (x \cdot x + x \cdot y) + (y \cdot (x + y))$
$\rhd$ `rewrite` $\leftarrow$ $(R6\,, [x, y, z := x, x, y]$
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = (x \cdot (x + y)) + (y \cdot (x + y))$
$\rhd$ `rewrite` $\leftarrow$ $(R7\,, [x, y, z := x, y, x + y]$
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = (x + y) \cdot (x + y)$
$\rhd$ `rewrite` $(H0\,, [x := x + y])$
$\mathcal{R}', H1 : x + (y + z) = y + (x + z) \vdash x + y = x + y$
$\rhd$ `reflexivity`
$\square$

EXAMPLE 12
The following is a backward derivation of

$$\Gamma \vdash cross(map\ f, map\ g)\ .\ unzip = unzip\ .\ map(cross(f, g)),$$

where Γ is the set of definitions and laws in Example 1.4. We leave to the reader to find the suitable substitution in each step.

$$
\begin{array}{l}
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = unzip\,.\,map\ (cross(f,g)) \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{rewrite}\ D1 \\
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = fork(map\, fst\,,\ map\, snd)\,.\,map\ (cross(f,g)) \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{rewrite}\ Law2 \\
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = \\
\qquad\qquad fork(map\, fst\,.\,map\ (cross(f,g))\,,\ map\, snd\,.\,map\ (cross(f,g))) \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{rewrite} \leftarrow Law5 \\
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = \\
\qquad\qquad fork(map\ (fst\,.\,cross(f,g))\,,\ map\, snd\,.\,map\ (cross(f,g))) \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{rewrite} \leftarrow Law5 \\
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = \\
\qquad\qquad\qquad fork(map\ (fst\,.\,cross(f,g))\,,\ map\ (snd\,.\,cross(f,g))) \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{rewrite}\ Law4 \\
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = fork(map\ (fst\,.\,cross(f,g))\,,\ map\ (g\,.\,snd)) \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{rewrite}\ Law3 \\
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = fork(map\ (f\,.\,fst)\,,\ map\ (g\,.\,snd)) \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{rewrite}\ Law5 \\
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = fork(map\ (f\,.\,map\, fst)\,,\ map\ (g\,.\,snd)) \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{rewrite}\ Law5 \\
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = fork(map\, f\,.\,map\, fst\,,\ map\, g\,.\,map\, snd) \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{rewrite} \leftarrow Law1 \\
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = cross(map\, f, map\, g)\,.fork(map\, fst, map\, snd) \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{rewrite} \leftarrow D1 \\
\Gamma \vdash cross(map\, f, map\, g)\,.\,unzip = cross(map\, f, map\, g)\,.\,unzip \\
\triangleright \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{reflexivity} \\
\ \square
\end{array}
$$

A natural question is to ask how to construct a usual (forward) proof from a backward proof. This is usually done internally by a proof assistant by means of proof terms or proof objects. Each tactic includes a so-called validation function which, when given theorems corresponding to each subgoal, constructs a theorem corresponding to the original goal. These validation functions are implemented and applied automatically by modern proof assistants. However, as we will show next, the validation functions are not necessary, for our concept of backward proof in the transition system $\mathcal{T}_{Eq}$ will be equivalent to the notion of forward derivability. This equivalence captures, within formal logic, the usual mathematical reasoning, where a proof of a given proposition A seldom includes an explicit derivation of A. Instead, such proof only validates another, usually simple propositions whose provability entail that of A.

8.1 Equivalence of forward and backward proofs

A backward proof has been defined as a sequence of transitions in $\mathcal{T}_{Eq}$. To being able to manipulate these sequences, we introduce the transitive closure of the transition relation.

DEFINITION 9
The transitive closure of the transition relation $\rhd$, denoted $\rhd^+$, is inductively defined as follows:

$$\frac{\mathcal{S} \rhd \mathcal{S}'}{\mathcal{S} \rhd^+ \mathcal{S}'}\text{TC1} \qquad \frac{\mathcal{S} \rhd \mathcal{S}' \qquad \mathcal{S}' \rhd^+ \mathcal{S}''}{\mathcal{S} \rhd^+ \mathcal{S}''}\text{TCM}$$

Let us observe that a backward proof of $\Gamma \vdash E$ is just a sequence of transitions that witness the fact that $\Gamma \vdash E \rhd^+ \Box$. With this concept at hand, the intuitive equivalence between forward and backward proof means that if $\Gamma \vdash E$ is provable, then it should happen that $\Gamma \vdash E \rhd^+ \Box$ and vice versa. We prove this fact after the following technical lemma and definition.

LEMMA 11
The following rule is derivable.

$$\frac{\mathcal{S}_1 \rhd^+ \mathcal{S}_2}{\mathcal{S}_1; \mathcal{S} \rhd^+ \mathcal{S}_2; \mathcal{S}}\text{TCAPP}$$

PROOF. Induction on $\mathcal{S}_1 \rhd^+ \mathcal{S}_2$. □

DEFINITION 10
We say that a goal sequence $\mathcal{S} =_{def} \mathcal{G}_1, \ldots, \mathcal{G}_k$ is derivable, if $\mathcal{G}_i$ is derivable for all $1 \leq i \leq k$.

The next lemma shows that derivability of sequences is rearward preserved by the transition relation $\rhd$.

LEMMA 12
If $\mathcal{S}_1 \rhd \mathcal{S}_2$ and the sequence $\mathcal{S}_2$ is derivable, then $\mathcal{S}_1$ is derivable.

PROOF. It is clear that the property holds for the basic transitions, since these correspond to the inference rules. The result follows now by induction on $\rhd$. □

The above property is easily lifted to the transitive closure $\rhd^+$.

LEMMA 13
If $\mathcal{S}_1 \rhd^+ \mathcal{S}_2$ and the sequence $\mathcal{S}_2$ is derivable, then $\mathcal{S}_1$ is derivable.

PROOF. Induction on $\mathcal{S}_1 \rhd^+ \mathcal{S}_2$. □

We are now ready to show the desired equivalence.

THEOREM 3 (Equivalence of forward and backward proofs).
Let $\Gamma \vdash_{\mathcal{B}\text{GOAL}} E$ be any sequent. The following conditions are equivalent:

- $\Gamma \vdash_{\mathcal{B}\text{GOAL}} E \quad \rhd^+ \Box$
- $\Gamma \vdash_{\mathcal{B}\text{GOAL}} E$ is derivable.

PROOF. The $\Rightarrow$) direction is immediate from Lemma 13. The $\Leftarrow$) direction is proved by induction on the derivability of $\Gamma \vdash_{\mathcal{B}\text{GOAL}} E$. We show the inductive step for the rule DEML: in this case we have $\Gamma \vdash_{\mathcal{B}\text{GOAL}} E[x := t\sigma]$ coming from $\Gamma \vdash_{\mathcal{B}\text{GOAL}} t = s$ and $\Gamma \vdash_{\mathcal{B}\text{GOAL}} E[x := s\sigma]$. The I.H. yields (1) $\Gamma \vdash_{\mathcal{B}\text{GOAL}} t = s \rhd^+ \Box$ and (2) $\Gamma \vdash_{\mathcal{B}\text{GOAL}} E[x := s\sigma] \rhd^+ \Box$. From these transitions, the

demodulation tactic and the TCAPP rule (Lemma 11) bring in the following goal sequence:

$$
\begin{array}{ll}
\Gamma \vdash_{\mathcal{B}^{\text{GOAL}}} E[x := t\sigma] & \\
\rhd & \texttt{demodulation } (t = s) \\
\Gamma \vdash_{\mathcal{B}^{\text{GOAL}}} t = s\,;\ \Gamma \vdash_{\mathcal{B}^{\text{GOAL}}} E[x := s\sigma] & \\
\rhd^{+} & (1) \\
\Box\,;\ \Gamma \vdash_{\mathcal{B}^{\text{GOAL}}} E[x := s\sigma] & \\
\rhd^{+} & (2) \\
\Box &
\end{array}
$$

which implies $\Gamma \vdash_{\mathcal{B}^{\text{GOAL}}} E[x := t\sigma] \rhd^{+} \Box$, as wished. The other cases are similar. □

As a consequence of the above theorem, once a backward derivation of a given judgment has been constructed, it is not necessary to produce the corresponding forward derivation.

This achieves our objective.We conclude the exposition with some final remarks.

9 Final remarks

The work here presented gives a formal notion of backward proof for first-order equational logic. The target is achieved by developing a calculus for interactive theorem proving following the LCF framework. For this purpose, we first studied the well-known Birkhoff calculus, revealing that the proofs there are quite difficult to reproduce in a backward style. Then, we developed two deductive systems following two usual notions of replacement: demodulation, coming from automated theorem proving in clausular logic; and subterm rewriting, the kind of replacement native of term rewriting systems. These calculi were explored in order to ease the proofs within a rigorous deductive system and to reflect intuition while trying a backward proof construction. By combining the features of each of these three equivalent calculi, we obtain a deductive system $\mathcal{B}^{\text{GOAL}}$ fully compatible with the practical use of logic and with the spirit of backward reasoning techniques. This system was transformed into a transition system of tactics $\mathcal{T}_{Eq}$ allowing for a direct formal definition of backward proof, one that can and is used in practice, for it corresponds to a subset of equational tactics available in the COQ proof assistant, see [8, Sections 8.6, 8.12]. The notion of backward proof is not new, for instance it is essentially what Kanger calls *quasi-deduction* in [23]. Though not really engaged in that work, this statement together with other ideas and proof sketches there shown, are outstanding for his time and must be taken up again for formal verification. We consider that the notion of backward proof here presented is a good starting point for this enterprise and also to pursue higher-order equational reasoning, like the one involving dependent types and pattern-matching or binding constructors in programming languages.

With respect to future work, we mention the need for a comparison of our work with other logics for equational reasoning, for instance rewriting logic which is the foundation of MAUDE [7], a proof assistant based on equational logic. Can the notions of tactic and backward proof here developed be adapted to obtain a better understanding of MAUDE proofs and proof objects? Another interesting topic is proof complexity in particular of those proofs obtained by cut elimination in $\mathcal{B}^{\text{REW}}$.

As a final thought, we observe that an important subject left out of this work is the role of induction in equational proofs, a vital methodology for realistic proofs. Nevertheless, we consider that a treatment of induction as an extension of pure equational logic is not worth to pursuit. For even the simpler inductive arguments related to mathematics or program verification demand a more expressive language where the manipulation of explicit quantifiers is needed.

Funding

This work was supported by Universidad Nacional Autónoma de México [PAPIME-PE102117].

References

[1] J. R. Abrial. *Modeling in Event-B: System and Software Engineering*, 1st edn. Cambridge University Press, New York, NY, USA, 2010.
[2] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, New York, NY, USA, 1998.
[3] R. Bird. *Pearls of Functional Algorithm Design*, 1st edn. Cambridge University Press, New York, NY, USA, 2010.
[4] R. Bird. *Thinking Functionally with Haskell*. Cambridge University Press, 2014.
[5] G. Birkhoff. On the structure of abstract algebras. *Mathematical Proceedings of the Cambridge Philosophical Society*, **31**, 433–454, 1935.
[6] S. Burris. *Logic for Mathematics and Computer Science*. Prentice Hall, Upper Saddle River, N.J., 1998.
[7] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer and C. L. Talcott, eds. *All About Maude—A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*. Vol. **4350** of Lecture Notes in Computer Science. Springer, 2007.
[8] The Coq Development TeamThe Coq Development Team The Coq Reference Manual, Version 8.6, 2017. Available electronically at http://coq.inria.fr/doc.
[9] A. Degtyarev and A. Voronkov. Equality reasoning in sequent-based calculi. In *Handbook of Automated Reasoning*, A. Robinson and A. Voronkov, eds, pp. 611–706. North–Holland, Amsterdam, 2001.
[10] L. del Carmen González-Huesca, F. E. Miranda-Perea and P. S. Linares-Arévalo. Axiomatic and dual systems for constructive necessity, a formally verified equivalence. *Journal of Applied Non-Classical Logics*, **29**, 255–287, 2019.
[11] N. Dershowitz and D. A. Plaisted. Rewriting. In *Handbook of Automated Reasoning*, A. Robinson and A. Voronkov, eds, pp. 535–610. North–Holland, Amsterdam, 2001.
[12] P. J. Eccles. *An Introduction to Mathematical Reasoning*. Cambridge University Press, 2012.
[13] M. Ferrari and C. Fiorentini. Proof-search in natural deduction calculus for classical propositional logic. In *Automated Reasoning with Analytic Tableaux and Related Methods: 24th International Conference, TABLEAUX 2015, Wroclaw, Poland, September 21–24, 2015, Proceedings*, H. De Nivelle, ed, pp. 237–252. Springer International Publishing, 2015.
[14] D. M. Gabbay. Introduction to labelled deductive systems. In *Handbook of Philosophical Logic*, pp. 179–266. Springer, Netherlands, 2014.
[15] G. Gentzen. Untersuchungen über das logische schließen I, II. *Mathematische Zeitschrift*, **39**, 176–210, 1935.
[16] J. A. Goguen and K. Lin. Specifying, programming and verifying with equational logic. In *We Will Show Them! (2)*, pp. 1–38. College Publications, 2005.
[17] J. A. Goguen and G. Malcolm. *Algebraic Semantics of Imperative Programs*. MIT Press, Cambridge, MA, USA, 1996.
[18] M. J. C. Gordon, R. Milner and C. P. Wadsworth. *Edinburgh LCF*. Vol. **78** of Lecture Notes in Computer Science. Springer, 1979.
[19] J. Harrison. *Handbook of Practical Logic and Automated Reasoning*, 1st edn. Cambridge University Press, New York, NY, USA, 2009.

[20] G. Holmström-Hintikka, S. Lindström and R. Sliwinski, eds. *Collected Papers of Stig Kanger with Essays on his Life and Work*. Vol. **303** of Synthese Library, 1st edn. Springer Netherlands Kluwer Academic Publishers, 2001.
[21] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, New York, NY, USA, 2004.
[22] G. Hutton. *Programming in Haskell*, 2nd edn. Cambridge University Press, New York, NY, USA, 2016.
[23] S. Kanger. *Provability in Logic*. Vol. **1** of Acta Universitatis Stockholmiensis. Stockholm Studies in Philosophy. Almqvist & Wiksell, 1957.
[24] S. Kanger. A simplified proof method for elementary logic. In *Computer Programming and Formal Systems, Studies in Logic and the Foundations of Mathematics*, P. Braffort and D. Hirschberg, eds, pp. 87–94, North–Holland, 1963.
[25] S. Kanger. Equational calculi and automatic demonstration. In *Logic and Value: Essays Dedicated to Thorild Dahlquist on His Fiftieth Birthday*, T. Pauli, ed, pp. 220–226, Uppsala, 1970. Filosofiska studier utgivna av Filosofiska föreningen oeh Filosofiska institutionen vid Uppsala universitet 9, Uppsala University.
[26] E. L. Lusk and R. A. Overbeek. Reasoning about equality. *Journal of Automated Reasoning*, **1**, 209–228, 1985.
[27] Y. I. Manin. *A Course in Mathematical Logic for Mathematicians*. Vol. **53** of Graduate Texts in Mathematics. Springer, New York, 2010.
[28] E. Mendelson. *Introduction to Mathematical Logic*, 5th edn. Chapman & Hall/CRC, 2009.
[29] F. E. Miranda-Perea, S. Linares-Arévalo and A. Aliseda. How to prove it in natural deduction: A tactical approach. CoRR, abs/1507.03678, 2015.
[30] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In *Handbook of Automated Reasoning*, A. Robinson and A. Voronkov, eds, pp. 371–443. North–Holland, Amsterdam, 2001.
[31] L. C. Paulson. *Logic and Computation: Interactive Proof with Cambridge LCF*. Vol. **2** of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.
[32] D. Prawitz. *Natural Deduction: A Proof-Theoretical Study*. Dover Books on Mathematics, Dover Publications, 2006.
[33] W. Sieg and J. Byrnes. Normal natural deduction proofs (in classical logic). *Studia Logica*, **60**, 67–106, 1998.
[34] W. Sieg and S. Cittadini. Normal natural deduction proofs (in non-classical logics). In *Mechanizing Mathematical Reasoning, Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, D. Hutter and W. Stephan eds, volume 2605 of Lecture Notes in Computer Science, pp. 169–191, Springer, 2005.
[35] D. Solow. *How to Read and Do Proofs: An Introduction to Mathematical Thought Processes*, 6th edn. Wiley Global Education, 2013.
[36] V. Sperschneider and G. Antoniou. *Logic: A Foundation for Computer Science (International Computer Science Series)*, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1991.
[37] N. Vazou, J. Breitner, R. Kunkel, D. Van Horn and G. Hutton. Theorem proving for all: Equational reasoning in liquid haskell (functional pearl). In *Proceedings of the 11th ACM SIGPLAN International Symposium on Haskell, Haskell 2018*, pp. 132–144. ACM, New York, NY, USA, 2018.
[38] D. J. Velleman. *How to Prove It: A Structured Approach*. Cambridge University Press, 2006.

[39] L. T. Wos, G. A. Robinson, D. F. Carson and L. Shalla. The concept of demodulation in theorem proving. In *Automation of Reasoning 2: Classical Papers on Computational Logic 1967–1970*, pp. 66–81. Springer, Berlin Heidelberg, 1983.

Received 15 January 2018