Rewriting Logic as a Metalogical Framework

David Basin[1], Manuel Clavel[2], and José Meseguer[3]

[1] Institut für Informatik, Universität Freiburg, Germany
[2] Department of Philosophy, University of Navarre, Spain
[3] Computer Science Laboratory, SRI International, USA

Abstract. A metalogical framework is a logic with an associated methodology that is used to represent other logics and to reason about their metalogical properties. We propose that logical frameworks can be good metalogical frameworks when their logics support reflective reasoning and their theories always have initial models.

We present a concrete realization of this idea in rewriting logic. Theories in rewriting logic always have initial models and this logic supports reflective reasoning. This implies that inductive reasoning is valid when proving properties about the initial models of theories in rewriting logic, and that we can use reflection to reason at the metalevel about these properties. In fact, we can uniformly reflect induction principles for proving metatheorems about rewriting logic theories and their parameterized extensions. We show that this reflective methodology provides an effective framework for different, non-trivial, kinds of formal metatheoretic reasoning; one can, for example, prove metatheorems that relate theories or establish properties of parameterized classes of theories. Finally, we report on the implementation of an inductive theorem prover in the Maude system, whose design is based on the results presented in this paper.

1 Introduction

A logical framework is a logic with an associated methodology that is employed for representing and using other logics, theories, and, more generally, formal systems. A number of logical frameworks have been proposed and to compare them and analyze their effectiveness, it is helpful to distinguish between their intended applications. In particular, we can distinguish between *logical frameworks*, where the emphasis is on reasoning *in* a logic, in the sense of simulating its derivations in the framework logic, and *metalogical frameworks*, where the emphasis is on reasoning *about* logics and even about relationships *between* logics. Metalogical frameworks are more powerful, as they include the ability to reason about a logic's entailment relation, as opposed to merely being adequate to simulate entailment.

Induction plays a central rôle in distinguishing logical frameworks from their metalogical counterparts. In a logical framework, representations of proof rules are used to construct derivations of (object logic) entailments. This approach is taken in logical frameworks like Isabelle [34] and the Edinburgh LF [22]. There, one may formalize logics and theories where induction is present *within* the theory (e.g., Peano Arithmetic), but induction is not present *over* the theories. That is, the framework does not support induction over the terms and proofs of a theory. In contrast, in a metalogical framework, it is essential to have induction over theories. Standard proof-theoretic arguments usually require induction over the formulae or derivations of the object theory. Induction is essential too for computer science applications, like reasoning about operational semantics.

1.1 Reflective Metalogical Frameworks

In this paper, we propose a new approach to metalogical frameworks motivated by the following observation. A logic's syntax and proofs can be viewed as algebras, whose carrier sets are inductively built from syntax and proof constructors. A logical framework and a metalogical framework can share these as a common basis. However, whereas for a logical framework the application of these constructors suffices to simulate derivations of the object logic, for a metalogical framework, our representation must additionally preserve the inductive nature of these algebras. That is, a formalization in the metalogic should have an *initial model* corresponding to the syntax and proofs of the formalized object logic.

Our proposal is that for some logical frameworks—namely, those that are reflective and whose theories have initial models—we can take the step from a logical framework to a metalogical framework by reflecting at the metalevel the induction principles for the formalized logics. We sum this up with the slogan "*logical frameworks with reflection and initiality are metalogical frameworks*".

After making this idea precise, we give a concrete realization of it using rewriting logic and present an example. Our example is a standard one in metareasoning: the deduction theorem for *minimal logic (of implication)*. Rewriting logic is not the only candidate for a reflective metalogical framework, but we believe it is a good one. Rewriting logic has been demonstrated to be a good logical framework [11,23,24,30,36,37] and it is balanced on a point where it is expressive enough to naturally formalize different entailment systems, but it is weak enough so that its theories always have initial models. This means that there are sound induction principles for reasoning with respect to these models. To prove metatheorems about theories in rewriting logic and their parameterized extensions, the key is to *reflect* these reasoning principles at the metalevel.

Overall, we see our contributions as both theoretical and practical. Theoretically, our work contributes to answering the question "*what is a metalogical framework?*" by proposing reflective logical frameworks, whose theories have initial models, as a possible answer. Moreover, it illuminates the interrelationship between logical and metalogical frameworks, and the rôle of reflection as a key ingredient for turning a logical framework with initial models into a metalogical one. Practically, we provide evidence that rewriting logic, combined with reflection, is an effective metalogical framework that can be used for nontrivial kinds of metatheoretic reasoning.

1.2 Related Work

Various approaches have been considered in the past to strengthen logical frameworks so that they can function as metalogical frameworks. All of these differ significantly from our proposal both in their logical basis and in the rôle of reflection in metareasoning.

One approach is to formalize theories in a framework logic supporting some notion of module, where each module is explicitly equipped with its own induction principle. For example, in [3], theories were formalized by collections of parameterized modules (Σ-types) within the Nuprl type theory (a constructive, higher-order logic), and each module included its own induction principle for reasoning about terms or proofs. This approach is powerful and can be used, for example, to relate different theories formalized in this way.

An alternative approach is to formalize theories directly using inductive definitions in a framework logic or framework theory that is strong enough to formalize the corresponding induction principles. A simple example of this is the first-order theory FS_0 of [19], which has been used by [25] to carry out experiments in formal metatheory. In FS_0, inductive definitions are terms in the framework theory, which has an induction rule for reasoning about such terms.

Another common choice is to formalize theories as inductive definitions in strong "foundational" framework logics such as higher-order logic or set-theory [21,33], or in a type theory like the calculus of constructions with inductive definitions [32]. In higher-order logic and set theory one can internally develop a theory of inductive definitions, where inductive definitions correspond to terms in the metatheory (e.g., formalized as the least fixedpoint of a monotonic function) and, from the definition, induction principles are formally derived within the framework logic. Alternatively, in the calculus of constructions, given an inductive definition, induction principles are simply added, soundly, to the metalogic. Current research in this area focuses on appropriate induction principles for logics that support higher-order abstract syntax [17,26,35].

Organization

The remainder of our paper is organized as follows. In Section 2 we present the idea of a reflective metalogical framework and abstractly formalize our requirements for such a metalogic. In Section 3 we present background material on rewriting logic, membership equational logic, and the Maude language. In Section 4 we discuss induction principles for membership equational theories and present a simple notion of parameterized membership equational theory. In Section 5 we discuss how rewriting logic can be used as a logical framework, and in Sections 6 and 7 we show how to combine initiality and reflection to use rewriting logic as a metalogical framework. In particular, we show how to reflect, in a uniform way, induction principles for reasoning, at the metalevel, about theories and their parameterized extensions. After this, we present in Section 8 an example of formal metareasoning using rewriting logic as a metalogical framework, namely the proof of the deduction theorem, and we draw conclusions in Section 9.

2 Reflective Metalogical Frameworks

In this section we begin by defining reflective logics. Based on this, we then describe properties sufficient for a reflective logical framework to function as a reflective metalogical framework.

2.1 Reflective Logics

Intuitively, a reflective logic is a logic in which important aspects of its metatheory, such as theories and entailment, can be represented and reasoned about *in* the logic. A general axiomatic notion of reflective logic was recently proposed in [7,14]. The notion is itself expressed in terms of the more general axiomatic notion of an *entailment system* [27], which captures the entailment relation of a logic. For our purposes here, an entailment system $\mathcal{E}$ consists of the following:

1. a class *Sign* of *signatures*, where each signature $\Sigma \in \mathit{Sign}$ specifies the syntax of a language;
2. a function *sen* assigning to each signature $\Sigma \in \mathit{Sign}$ a set $\mathit{sen}(\Sigma)$ of its sentences;
3. for each signature $\Sigma \in \mathit{Sign}$, an *entailment relation* $\vdash_\Sigma$, where $\vdash_\Sigma \subseteq \mathcal{P}(\mathit{sen}(\Sigma)) \times \mathit{sen}(\Sigma)$ and $\vdash_\Sigma$ satisfies the properties of reflexivity, monotonicity, and transitivity (or cut); in what follows, we omit the subscript of $\vdash_\Sigma$ when Σ is clear from the context.

A *theory* in $\mathcal{E} = (\mathit{Sign}, \mathit{sen}, \vdash)$ is then a pair $T = (\Sigma, \Gamma)$ consisting of a signature $\Sigma \in \mathit{Sign}$ and a set of sentences $\Gamma \subseteq \mathit{sen}(\Sigma)$. We can extend the entailment relation to theories in the obvious way by defining $(\Sigma, \Gamma) \vdash \varphi$ iff $\Gamma \vdash \varphi$, for $\varphi \in \mathit{sen}(\Sigma)$.

Definition 1 *Given an entailment system $\mathcal{E}$ and a nonempty set of theories $\mathcal{C}$ in it, a theory U is $\mathcal{C}$-*universal *if there is a function, called a* representation function*,*

$$\overline{(\_ \vdash \_)} : \bigcup_{T \in \mathcal{C}} (\{T\} \times \mathit{sen}(T)) \longrightarrow \mathit{sen}(U)\,,$$

such that for each $T \in \mathcal{C}, \varphi \in sen(T)$,

$$T \vdash \varphi \;\; \textit{iff} \;\; U \vdash \overline{T \vdash \varphi}\,. \tag{1}$$

If, in addition, $U \in \mathcal{C}$, *then the entailment system* $\mathcal{E}$ *is called* $\mathcal{C}$-reflective. *Finally, a* reflective logic *is a logic whose entailment system is* $\mathcal{C}$*-reflective for* $\mathcal{C}$*, the class of all finitely presentable theories in the logic.*

2.2 Requirements for a Reflective Metalogical Framework

We now consider what we require from a logical framework so that it can function as a metalogical framework. As indicated in Section 1.2, various approaches to formal metareasoning have been proposed in the past. Our approach is based on reflective reasoning and initiality, and here we present, abstractly our requirements for this. They are:

1. the logical framework is *weak* enough so that there are valid induction principles for reasoning about all its theories,
2. the logical framework is *expressive* enough so that it really is a viable logical framework, and
3. the logical framework is *reflective.*

Note that 1 specifies a requirement on the framework logic and can be alternatively formulated in an abstract and logic-independent way. If the framework logic is such that its theories have initial models, then an appropriate form of inductive reasoning is always valid when proving sentences with respect to the initial models of its theories. This method is very general; for example, for equational logic, induction and initiality are equivalent concepts [31].[1]

We now explain why the requirements listed above are sufficient for turning a logical framework into a metalogical framework. If requirement 2 is satisfied, then logics and their entailment relations can be represented as theories in the logical framework, and if requirement 1 is also satisfied, then these representations can preserve the inductive nature of the algebras characterizing the syntax and proofs of the logics that they represent. As a consequence, proof-theoretic arguments requiring induction over the formulae or over the derivations of a logic can be applied in the framework logic. This is enough when proving theorems about a logic. These theorems can be formalized as

[1] Since the notion of initiality is very general, the corresponding inductive reasoning principles may in each case take different forms. For example, in an equational logic allowing infinitary operations of arity smaller than a given regular cardinal α, the inductive principles will be transfinite. We are mainly interested in logical frameworks suitable for representing *finitary* logics. Therefore, we will in practice be interested in a finitary framework logic whose theories have initial models and whose induction principles are also finitary.

sentences about the initial model of the theory representing the object logic under consideration, and can be proved by induction.

However, when dealing with metatheorems we often require something more. Metatheorems may relate *different* logics in a family of logics. Consider, for example, the deduction theorem for minimal logic (of implication). This is actually a metatheorem not about a particular deduction system, but rather a metatheorem that relates different deduction systems: one in which $A \rightarrow B$ is proved and a second (which is obtained from the first by adding the axiom A) in which B is proved. In our setting, this means that sentences formalizing metatheorems should relate initial models of different theories. Here is where reflection plays a decisive rôle. Namely, if the logical framework satisfies requirement 3, then: (3a) it contains a universal theory where the metalevel of its logic can be reflected, and (3b) the universal theory is itself a theory in the logical framework. By 3b and 1 the universal theory has an initial model. The key then is to exploit 3a in order to formalize relationships between the initial models of object theories as theorems about the initial model of the universal theory, and turn, by reflection, induction principles for reasoning about the initial models of object theories into induction principles for reasoning about the initial model of the universal theory.

In the following sections, we will give a concrete instance of these ideas for the case of rewriting logic. In particular, we will show that for a certain class of rewriting logic theories the induction principles for reasoning, at the metalevel, about these theories correspond, in a simple way, to the induction principles for reasoning about the inductive properties of the theories. Moreover, by reasoning by induction in the universal theory we can inductively reason about properties satisfied by families of theories. This provides us with capabilities analogous to what is possible in metalogical frameworks based on parameterized inductive definitions.

3 Background

In this section we provide background material on rewriting logic, membership equational logic, and the Maude language. The material presented here is standard. We postpone discussion of the reflective aspects to Section 6.

3.1 Rewriting Logic

Rewriting logic [28] is a simple logic whose sentences are sequents of the form $t \longrightarrow t'$, with t and t' Ω-terms on a given signature Ω. Theories in rewriting logic are triples (Ω, E, R), with Ω a signature of operators, E a set of Ω-equations, and R a collection of (possibly conditional [28]) Ω-rewrite rules.

The inference rules of rewriting logic [28] allow the derivation of all rewrites possible in a given theory. Thus, from the logical point of view, we can think of rewriting logic as a framework logic in which formulae are formalized as elements of the initial model of an equational theory (Ω, E) and an inference system is formalized by expressing each inference rule as a (possibly conditional) rewrite rule. Rewriting is understood *modulo* the equations E. This supports a flexible and abstract kind of inference where the equations can take care of structural bookkeeping. For example, when formalizing sequent calculi, structural rules for sequents can be "internalized" by rewriting modulo appropriate equational axioms such as associativity, associativity-commutativity, and so on.

Since a rewrite theory (Ω, E, R) has an underlying equational theory (Ω, E), rewriting logic is parameterized by the choice of the equational logic. An attractive choice in terms of expressiveness is *membership equational logic* [29], a logic that has sorts, subsorts, overloading of function symbols, and is capable of expressing partiality using equational conditions. Since we can view an equational theory (Ω, E) as a rewrite theory $(\Omega, E, \emptyset)$, there is an obvious sublogic inclusion, $\mathit{MEqtl} \subseteq \mathit{RWLogic}$, from membership equational logic into rewriting logic. Both membership equational logic and rewriting logic have initial models [28,29], which provide the basis for reasoning by induction.

3.2 Membership Equational Logic

Membership equational logic is an expressive version of equational logic. A full account of the syntax and semantics of membership equational logic can be found in [6,29]. Here we define the basic notions needed in this paper.

A *signature* in membership equational logic is a triple $\Omega = (K, \Sigma, S)$ with K a set of *kinds*, Σ a K-kinded signature $\Sigma = \{\Sigma_{w,k}\}_{(w,k)\in K^* \times K}$, and $S = \{S_k\}_{k\in K}$ a pairwise disjoint K-kinded family of sets. We call S_k the set of *sorts* of kind k. The pair (K, Σ) is what is usually called a many-sorted signature of function symbols; however we call the elements of K kinds because each kind k now has a set S_k of associated *sorts*, which in the models will be interpreted as subsets of the carrier for the kind. Also, as usual, we denote by T_Σ the K-kinded algebra of ground Σ-terms, and by $T_\Sigma(X)$ the algebra of Σ-terms on the K-kinded set of variables X.

The atomic formulae of membership equational logic are either equations $t = t'$, where t and t' are Σ-terms of the same kind, or *membership assertions* of the form $t : s$, where the term t has kind k and $s \in S_k$. Sentences are Horn clauses on these atomic formulae, i.e., sentences of the form

$$\forall(x_1, \ldots, x_m).\, A_1 \wedge \ldots \wedge A_n \Rightarrow A_0 \,,$$

where each A_i is either an equation or a membership assertion, and each x_j is a K-kinded variable. For example, Figure 1 gives a set of membership equational axioms specifying minimal logic of implication, where `SentConstant`, `Formula`, and `Theorem` are sorts formalizing sentential constants, formulae,

$$\forall A.\, A\!:\!\texttt{SentConstant} \Rightarrow A\!:\!\texttt{Formula},$$
$$\forall A.\, A\!:\!\texttt{Theorem} \Rightarrow A\!:\!\texttt{Formula},$$
$$\forall (A,B).\, A\!:\!\texttt{Formula} \wedge B\!:\!\texttt{Formula} \Rightarrow A \to B\!:\!\texttt{Formula},$$
$$\forall (A,B).\, A\!:\!\texttt{Formula} \wedge B\!:\!\texttt{Formula} \Rightarrow A \to (B \to A)\!:\!\texttt{Theorem},$$
$$\forall (A,B,C).\, A\!:\!\texttt{Formula} \wedge B\!:\!\texttt{Formula} \wedge C\!:\!\texttt{Formula}$$
$$\Rightarrow (A \to B) \to ((A \to (B \to C)) \to (A \to C))\!:\!\texttt{Theorem},$$
$$\forall (A,B).\, A\!:\!\texttt{Formula} \wedge B\!:\!\texttt{Formula} \wedge (A \to B)\!:\!\texttt{Theorem} \wedge A\!:\!\texttt{Theorem}$$
$$\Rightarrow B\!:\!\texttt{Theorem}$$

Fig. 1. Membership equational axioms for minimal logic.

and theorems, respectively.[2] A theory in membership equational logic is a pair (Ω, E), where E is a finite set of sentences in membership equational logic over the signature Ω. The way in which partiality is expressed in membership equational logic is by the fact that terms always have a kind, but may not have a sort. Terms for which a sort cannot be established from the axioms E correspond to *undefined* or *error* elements.

We employ standard semantic concepts from many-sorted logic. Given a signature $\Omega = (K, \Sigma, S)$, an *Ω-algebra* is a many-kinded Σ-algebra (that is, a K-indexed-set $A = \{A_k\}_{k \in K}$ together with a collection of appropriately kinded functions interpreting the function symbols in Σ) together with an assignment to each sort $s \in S_k$ of a subset $A_s \subseteq A_k$. Hence, sorts can be thought of as unary predicates that semantically denote subsets of the appropriate kind. An algebra A and a (kind-respecting) valuation σ, assigning to variables of kind k values in A_k, satisfy an equation $t = t'$ iff $\sigma(t) = \sigma(t')$, where we overload notation by identifying σ with its unique homomorphic extension to Σ-terms. We write $A, \sigma \models t = t'$ to denote such a satisfaction. Similarly, $A, \sigma \models t\!:\!s$ holds iff $\sigma(t) \in A_s$.

Note that an Ω-algebra is nothing but a K-kinded first-order model with function symbols Σ and an alphabet of unary predicates $\{S_k\}_{k \in K}$. Therefore, the satisfaction relation can be extended to Horn and first-order formulae ϕ over these atomic formulae in the standard way. We write $A \models \phi$ when the formula ϕ is satisfied for all valuations σ, and then say that A is a model of ϕ. Similarly, a theory (Ω, E) in membership equational logic is simply a Horn theory for the associated signature, when Ω is viewed as first-order K-kinded signature. As usual, for ϕ a first-order sentence in the language of Ω, we write $(\Omega, E) \models \phi$ when all the models of the set E of sentences are also models of ϕ.

Theories in membership equational logic have initial models [29]. This provides the basis for reasoning by induction, as is explained in detail in Section 4.1. We write $(\Omega, E) \models\!\!\!\simeq \phi$ to denote that the initial model of the membership equational logic theory (Ω, E) is also a model of ϕ. Note that

[2] Note that we write the object logic connective $\to$ in infix. We will consider this example in more detail in Section 5.

```
fmod MINIMAL is
sorts SentConstant Formula Theorem .
subsort SentConstant < Formula .
subsort Theorem < Formula .
op _→_ : Formula Formula -> Formula .
vars A B C : Formula .
mb A → (B → A) :  Theorem .
mb (A → B) → ((A →(B → C)) → (A → C)) : Theorem .
cmb B : Theorem if (A → B) : Theorem and  A : Theorem .
endfm
```

Fig. 2. The module MINIMAL.

even though we restrict the axioms E to Horn clauses, we will employ first-order formulae ϕ to formalize properties satisfied by the initial model, that is, inductive properties.

3.3 The Maude System

The Maude system [9,13] implements rewriting logic and has been designed with the explicit aims of supporting executable specification and reflective computation. Theories are specified in Maude by modules, of which there are two kinds: *functional modules* and *system modules*. Maude's functional modules are theories in membership equational logic. Equations in Maude's functional modules are assumed to be Church-Rosser and terminating; they are executed by the Maude rewrite engine according to the rewriting techniques and operational semantics developed in [6]. Maude's system modules are rewrite theories. The rules in a system module are not necessarily Church-Rosser or terminating.

The semantics of a functional (respectively system) module is *initial*, i.e., such a module denotes the *initial model* in membership equational logic (respectively rewriting logic) of the theory thus specified. The syntax for functional modules is of the form **fmod** (Ω, E) **endfm**, with (Ω, E) a membership equational theory meeting the requirements mentioned above. Figure 2 gives an example of a functional module in Maude syntax, where LaTeX symbols are used instead of ASCII characters to improve readability. Note that Maude's syntax for functional modules is syntactic sugar for introducing finite sets of membership axioms (Figure 2 is just the sugared version of Figure 1), and we will use it from now on to present membership equational theories. In particular, (possibly conditional) equations and membership axioms in Maude are Horn clauses in membership equational logic; any operation declaration `op` $f : s_1 \ldots s_n$ `->` s corresponds to the Horn clause

$$\forall(x_1, \ldots, x_n).\, x_1 : s_1 \wedge \ \ldots\ \wedge x_n : s_n \Rightarrow f(x_1, \ldots, x_n) : s\,,$$

where x_i is a variable of kind k_i and $s_i \in S_{k_i}$, for $i \in \{1, \ldots, n\}$; also, any subsort declaration `subsort` s `<` s' can be reduced to the sentence

$$\forall x.\, x : s \Rightarrow x : s'\,,$$

where x is a variable of kind k and $s, s' \in S_k$. Finally, kinds are not explicitly defined in Maude modules, but are instead inferred by the system as determined by the different connected components of the poset of sorts.

As additional syntactic sugar, we shall often write $\forall x : s.\, \phi(x)$ as shorthand for the formula $\forall x.\, x : s \Rightarrow \phi(x)$, for x a variable of kind k and $s \in S_k$. Moreover, for the formula $x : s \Rightarrow \phi(x)$, we will say that "$x$ is of sort s (in ϕ)."

4 Induction and Parameterization

In this section we introduce two concepts that play key rôles in the rest of the paper. We define an induction principle for membership equational theories and show how such theories can also be parameterized. We introduce these concepts in a simple setting that is adequate to illustrate the main ideas and carry out applications.

4.1 Induction Principles for Membership Equational Theories

Given that membership equational logic *is* a subset of equational Horn logic (indeed, they can be shown to be equivalent [29]) it follows immediately that any theory (Ω, E) has a unique (up to isomorphism) initial model [20]. The following is an induction principle for reasoning about properties of sorts, with respect to this model.

Definition 2 (Induction over sort definitions) *Let $T = (\Omega, E)$ be a theory in membership equational logic and let s be a sort in some S_k. Let $C_{[T,s]} = \{C_1, \ldots, C_n\}$ be those sentences in E that specify s, i.e., those C_i of the form*

$$\forall (x_1, \ldots, x_{p_i}).\, A_1 \wedge \ldots \wedge A_{q_i} \Rightarrow A_0\,, \tag{2}$$

where, for some t of kind k and $s \in S_k$, A_0 is $t : s$.

For τ a first-order formula with free variable x of sort s over the signature Ω, an induction principle for (Ω, E), with respect to $x : s$ and $\tau(x)$ *is the formula*

$$\psi_1 \wedge \ldots \wedge \psi_n \Rightarrow \forall x : s.\, \tau(x) \tag{3}$$

where, for $1 \leq i \leq n$ and C_i of the form (2), ψ_i is

$$\forall (x_1, \ldots, x_{p_i}).\, [A_1]_\tau \wedge \ldots \wedge [A_{q_i}]_\tau \Rightarrow [A_0]_\tau \tag{4}$$

and, for $0 \leq j \leq q_i$,

$$[A_j]_\tau = \begin{cases} \tau(u) & \textit{if } A_j = u\!:\!s, \textit{ for } u \textit{ of kind } k \\ A_j & \textit{otherwise} \end{cases}$$

For a given membership equational theory (Ω, E), the above defines an induction schema (*ind*), given by (3), in many-kinded first-order logic over the signature Ω.[3] Note that for $q_i = 0$, the nullary conjunction in the antecedent of (4) is *true* and the implication can be replaced with the succedent $\tau(t)$.

In the initial model of a membership equational theory, sorts are interpreted as the smallest sets satisfying the axioms in the theory, and equality is interpreted as the smallest congruence satisfying those axioms. Alternatively, the sets interpreting sorts can be characterized as being inductively generated in stages. This corresponds to the fixedpoint characterization of the least Herbrand model of a collection of Horn clauses [38], and the induction principle we have given formalizes induction over the stages in which the set is inductively defined [1]. By induction over the stages of the inductive definition of a sort s, which amounts to an induction over the proof that some ground term of kind k is of sort s, we can establish that reasoning in the membership equational theory (Ω, E), augmented by (*ind*), is sound.

Theorem 1 (Soundness) *Let* (Ω, E) *be a membership equational theory. If* $(\Omega, E \cup \{(ind)\}) \vdash \tau$, *then* $(\Omega, E) \models_{\simeq} \tau$.

As an example, consider the membership equational theory for minimal logic previously given in Figure 1. Definition 2 gives rise to the following induction principle over the sort `Theorem`:

$$\begin{array}{l}[\forall(A,B).\,(A\!:\!\texttt{Formula} \;\wedge\; B\!:\!\texttt{Formula} \;\Rightarrow\; \tau(A{\rightarrow}(B{\rightarrow}A))) \;\wedge \\ \forall(A,B,C).\,(A\!:\!\texttt{Formula} \;\wedge\; B\!:\!\texttt{Formula} \;\wedge\; C\!:\!\texttt{Formula} \\ \quad \Rightarrow \tau((A{\rightarrow}B){\rightarrow}(A{\rightarrow}(B{\rightarrow}C)){\rightarrow}(A{\rightarrow}C))) \;\wedge \\ \forall(A,B).\,(A\!:\!\texttt{Formula} \;\wedge\; B\!:\!\texttt{Formula} \wedge \tau(A{\rightarrow}B) \wedge \tau(A) \Rightarrow \tau(B)] \\ \qquad \Rightarrow \forall A\!:\!\texttt{Theorem}.\,\tau(A)\end{array}$$

This axiom formalizes induction over the structure of proofs in minimal logic.

Note that other induction principles are possible. In particular (*ind*) takes *all* of the sentences that specify membership in s as constituting the inductive definition of s. In some cases, a subset of the sentences (sometimes called *generators* or *constructors*) is sufficient to characterize an inductive definition. Of course additional proof obligations then arise, e.g., sufficient completeness of the chosen subset (see [6,10]).

[3] This induction schema cannot be directly formalized in the sublogic membership equational logic, since it is not, in general, a sentence in membership equational logic. However, as we will later see, inference rules for (many-kinded) first-order theories—like this induction schema—can be encoded in rewriting logic and can be used to prove properties, at the metalevel, about membership equational theories.

```
fmod MINIMAL_Ξ[𝒳] is
sorts SentConstant Formula Theorem .
subsort SentConstant < Formula .
subsort Theorem < Formula .
op 𝒳 : -> Formula .
op _→_ : Formula Formula -> Formula .
vars A B C : Formula .
mb 𝒳 : Theorem .
mb A → (B → A) : Theorem .
mb (A → B) → ((A →(B → C)) → (A → C)) : Theorem .
cmb B : Theorem if (A → B) : Theorem and A : Theorem .
endfm
```

Fig. 3. The module MINIMAL$_\Xi[\mathcal{X}]$.

4.2 Parameterized Membership Equational Theories

When carrying out metalogical reasoning, we often reason not about a fixed theory, but about a *parameterized family* of theories. There are many different ways in which a theory in membership equational logic may be parameterized. For the purposes of this paper it will be enough to consider the notion of a *parameterized extension* of a given theory T which, intuitively, describes the extensions of T by a parametric set of new axioms.

Definition 3 *Let $T = (\Omega, E)$ be a theory in membership equational logic with $\Omega = (K, \Sigma, S)$. Then, a* parameterized extension *of T (by* parameters *V, and* axioms *G) is a membership equational theory $T_G[V] = (\Omega[V], E \cup G)$, with $\Omega[V] = (K, \Sigma \cup V, S)$, where the K-kinded signatures Σ and V are mutually disjoint and V consists only of constants. We call $T_G[V]$ a* parameterized membership equational theory*.*

Let $\beta : V \longrightarrow T_\Sigma$ be a K-kinded function. Then, $T_G[\beta] = (\Omega, E \cup \beta(G))$ denotes an instance *of $T_G[V]$, where $\beta(G)$ is the homomorphic extension of β to axioms.*

The substitution β is used to generate instances of a parameterized membership equational theory. Namely, the new axioms G are instantiated so that all instances of variables in V are replaced by ground terms. For example, if $G = \{f(v) : s\}$ and $\beta(v) = q(a, b)$, then the parametric axiom $f(v) : s$ is translated as $\beta(f(v) : s) = f(q(a, b)) : s$. The result is well-kinded under the (non-parameterized) signature Ω.

Figure 3, provides an example of a parameterized module MINIMAL$_\Xi[\mathcal{X}]$, with $\mathcal{X}$ a parameter of the kind of the sort `Formula`, and Ξ a set that contains only the parametric axiom $\mathcal{X}$:`Theorem`. We will later see how this parameterized module can be used to formalize the deduction theorem.

5 Rewriting Logic as a Logical Framework

As we have already said, from the logical point of view we can think of rewriting logic as a framework logic in which an inference system can be formalized by expressing each inference rule as a (possibly conditional) rewrite rule in a rewrite theory (Ω, E, R). Note that rewriting logic is noncommittal about the structure and properties of the formulae expressed by Ω-terms. They are user-definable as an algebraic data type satisfying equational axioms, so that rewriting deduction takes place modulo such axioms. Because of this ecumenical neutrality and the simplicity of the rules of the logic, rewriting logic can be effectively applied as a logical framework. In [11,23,24,30,36,37], many examples of logic representations are given, including first-order linear logic, sequent presentations of modal and propositional logics, Horn logic with equality, the lambda calculus, and higher-order pure type systems, among others. In all such examples, the *representational distance* between the object logic and its representation in rewriting logic is virtually zero, that is, the representations are direct and reasoning with them faithfully simulates reasoning in the original logics.

In fact, there are several ways of conservatively representing a logic (with a finitary syntax and inference system) in rewriting logic. As mentioned before, a simple and direct way is to turn the inference rules into rewrite rules, which may be conditional if the inference rules have side conditions. Alternatively, we can use the underlying membership equational logic to represent theoremhood in a logic as a sort in a membership equational theory. Conditional membership axioms then directly support the representation of rules as schemas, which is typically used in presenting logics and formal systems. The module `MINIMAL`, presented previously in Figure 2, represents minimal logic in membership equational logic using this idea. A formula A is a theorem in minimal logic if and only if A is a term of sort `Theorem` in `MINIMAL`. Note that this representation preserves the inductive nature of the set of theorems and proofs in minimal logic.

Similarly, we can represent theoremhood in a parameterized family of logics as a sort in a parameterized membership equational theory.[4] As an example, consider the parameterized theory $\texttt{MINIMAL}_{\Xi}[\mathcal{X}]$ in Figure 3, with $\mathcal{X}$ a parameter of the kind of the sort `Formula`, and Ξ a set that contains only the parametric axiom $\mathcal{X}$:`Theorem`. This parameterized theory represents the family of logics that includes any extension of minimal logic with a new axiom in the following sense: a formula B is a theorem in minimal logic extended with a

[4] A sort in a parameterized membership equational theory can be used to represent theoremhood in a family of logics if and only if there is a one–to–one correspondence between logics in the family and instances of the parameterized membership equational logic theory, and this correspondence is such that theoremhood in a logic in the family can be represented as membership in this sort in the corresponding instance of the parameterized membership equational logic.

new axiom A if and only if B is a term of sort `Theorem` in $\texttt{MINIMAL}_{\Xi}[A]$, where $\texttt{MINIMAL}_{\Xi}[A]$ is the instance $\texttt{MINIMAL}_{\Xi}[\beta]$ of $\texttt{MINIMAL}_{\Xi}[\mathcal{X}]$, with $\beta(\mathcal{X}) = A$.

The ability to represent parameterized families of logics is important for using rewriting logic also as a metalogical framework, and we will give an example of this in the experimental work reported on in Section 8.

6 Reflection in Rewriting Logic and Maude

In this section we explain how rewriting logic is reflective and how the Maude system implements reflective rewriting logic deduction. We also introduce a Boolean function that reflects at the metalevel the membership relation in membership equational logic, which will be used in later sections. Finally, we explain how to combine the use of rewriting logic as a logical framework and the reflective capabilities of Maude to build a theorem prover for carrying out inductive proofs.

6.1 Reflection in Rewriting Logic

Rewriting logic is reflective [7,15,16]. There is a universal theory `UNIVERSAL`, and a representation function $\overline{(\_ \vdash \_)}$ encoding pairs consisting of a rewrite theory T and a sentence in it as sentences in `UNIVERSAL`. For any finitely presented rewrite theory T (including `UNIVERSAL` itself) and any terms t, t' in T, the representation function is defined by

$$\overline{T \vdash t \longrightarrow t'} = \langle \overline{T}, \overline{t} \rangle \longrightarrow \langle \overline{T}, \overline{t}' \rangle \,,$$

where $\overline{T}$, $\overline{t}$, $\overline{t}'$ are terms in `UNIVERSAL`. Then, the equivalence (1) in Section 2 holds for rewriting logic (as proved in [7,15,16]) and takes the form

$$T \vdash t \longrightarrow t' \text{ iff } \texttt{UNIVERSAL} \vdash \langle \overline{T}, \overline{t} \rangle \longrightarrow \langle \overline{T}, \overline{t}' \rangle \,.$$

6.2 Reflection in Maude

Maude's language design and implementation make systematic use of the fact that rewriting logic is reflective to give the user a well-defined gateway to the metatheory of rewriting logic. This entry point is the predefined module `META-LEVEL`, which is a (partial) specification in Maude of `UNIVERSAL` [8]. In the module `META-LEVEL`, a Maude term t is reified as an element $\overline{t}$ of a data type `Term` of terms, and a Maude module T, i.e., a membership equational theory, is reified as a *ground term* $\overline{T}$ in a data type `Module` of modules. See [8] for a complete definition of the module `META-LEVEL` and of the metarepresentation map for theories and terms.

The metarepresentation of a parameterized membership equational theory $T_G[V]$ is similar to that of an unparameterized theory T, except that parameters are treated in a special way. The *parametric* nature of $T_G[V]$ is expressed in its metarepresentation $\overline{T_G[V]}$ by the fact that each parameter $v \in V$ is represented by a (meta-) variable $\overline{v}$ of sort `Term`.

Note that since, in Definition 3, a parameterized theory $T_G[V]$ is technically defined as an ordinary membership equational theory (plus some extra information), one *could* metarepresent $T_G[V]$ as an ordinary theory, and then one would get a ground term of sort `Module`, instead of a term with variables. Therefore, our notation $\overline{T_G[V]}$ is potentially confusing, since it depends on whether $T_G[V]$ is metarepresented as a parameterized entity or as an unparameterized one. Rather than introducing new notation, we have chosen to solve the possible ambiguities by the context in which they occur. In particular, having introduced $T_G[V]$ as a parameterized membership equational theory, $\overline{T_G[V]}$ will always denote the metarepresentation of $T_G[V]$ as a parameterized entity. The same rule applies when metarepresenting parameterized terms in parameterized membership equational theories.

To reason about metarepresented (parameterized) theories we have defined, in an extension `META-IND` of `META-LEVEL`, a Boolean function (`_:_in_`) that reflects at the metalevel the membership relation in membership equational logic. (The theory is so named because it is in this theory that we will prove inductive metatheorems.) In particular, ($\overline{t}:\overline{s}$ `in` $\overline{T}$) checks, at the metalevel, whether the ground term t has the sort s in the functional module T. Specifically, this check of membership is based on the equivalence

$$T \vdash t : s \iff \texttt{META-IND} \vdash \overline{t}{:}\overline{s}\ \texttt{in}\ \overline{T} = \texttt{true}.$$

In [2] we give the full specification of (`_:_in_`) for a restricted subclass $\mathcal{C}$ of modules. Members of $\mathcal{C}$ are modules that correspond, basically, to membership equational theories whose axioms are Horn clauses that only involve membership assertions (no equations). The Boolean function (`_:_in_`) will play a key rôle in the rest of this paper.

In what follows, $\mathcal{C}$ will always denote the above mentioned subclass of modules. In addition, for any parameterized membership equational theory $T_G[V]$, we write $T_G[V] \in \mathcal{C}$ iff for any instance $T_G[\beta]$ of $T_G[V]$, $T_G[\beta] \in \mathcal{C}$. We also write $\overline{V}$ for the set of variables of sort `Term` that metarepresent the parameters $v \in V$ in $T_G[V]$.

6.3 Building an Inductive Theorem Prover

In Section 4.1 we have presented induction principles for reasoning about first-order formulae over sorts defined in functional modules, i.e., membership equational theories. Here we explain how to combine the use of rewriting logic as a logical framework and the reflective capabilities of Maude to build a theorem prover for carrying out inductive proofs. The paper [10] provides further details on building theorem proving tools in Maude.

To build an inductive theorem prover, we use rewriting logic to specify its inference system (as explained in Section 5) and reflection to define strategies that control rule application. Strategies are needed here since inference rules will be specified as rewrite rules that are not necessarily Church-Rosser or terminating. Hence, it is important to have some way of controlling the application of these rewrite rules in order to drive rewriting in some desired direction. Maude users can control rewriting by specifying, at the metalevel, their own rewriting strategies. (See [7,9] for more details on defining strategies in Maude.)

Our inductive theorem prover—the `ITP` tool[5]—has a reflective design. The functional module T, about which we want to prove inductive theorems, is at the object level. An inference system $\mathcal{I}$ for inductive proofs uses T as data and therefore is specified as a system module `ITP-RULES` at the metalevel. In particular, `ITP-RULES` encodes syntax and proof rules for first-order logic as well as the induction over sort definitions introduced in Definition 2. Finally, different *proof tactics* to guide the application of the rewrite rules specifying the inference rules in $\mathcal{I}$ are strategies, which are defined at the meta-metalevel in a module `ITP-TACTICS`.

Operationally, to use the `ITP` tool, the user submits as an initial goal the pair formed by (the metarepresentation) of a functional module and (the representation of) the first-order sentence over its signature that is to be proved, and then this goal is successively transformed by rewriting—using the inference rules as rewrite rules—into different sets of subgoals, until (in the case of a successful proof) no subgoals are left. The application of the inference rules as rewrite rules is controlled by the user using strategies.

Finally, note that building the theorem prover using different levels of reflection results in a modular design with a clean separation between the logical and the control components. For example, we can simply extend the tool by specifying additional inference rules in the module `ITP-RULES` without changing the strategy language defined in the module `ITP-TACTICS` and vice versa.

7 Rewriting Logic as a Metalogical Framework

We now show how the induction principles introduced in Section 4.1 for reasoning in membership equational theories can be uniformly reflected for reasoning, at the metalevel, about membership equational theories and their parameterized extensions. The results presented in this section provide the basis for using rewriting logic as a metalogical framework.

[5] `http://sophia.unav.es/~clavel` contains the most recent version of this tool.

7.1 Inductive Theorems versus Inductive Metatheorems

The induction principle presented in Section 4.1 is well-suited for proving properties of ground terms of sort s in a given membership equational theory (Ω, E), with $\Omega = (K, \Sigma, S)$, and s a sort in some S_k, *when these properties are expressible as first-order formulae over the signature* Ω; when this is the case, a property P holds if the first-order formula that expresses P holds in the initial model of (Ω, E). Of course, there are many interesting properties satisfied by ground terms of sort s that cannot be expressed as first-order formulae over the signature Ω, despite the fact that they are typically proved by induction over the definition of the sort s, e.g., properties that *relate different membership equational theories*. Many of these properties can be naturally expressed, at the metalevel, as first-order formulae over the signature of `META-IND`. Consider, for example, the following property: let $T = (\Omega, E) \in \mathcal{C}$ and $T' = (\Omega', E') \in \mathcal{C}$ be membership equational theories; then the property

> **if** *t is a ground term of sort s in T,*
> **then** *t is also a ground term of sort s' in T'*

is not expressible as a first-order formula over either T or T'. Notice that, using the Boolean function $(\_:\_\mathtt{in}\_)$, we can express this as a first-order formula at the metalevel, namely,

$$\forall x : \mathtt{Term}.\, (x : \overline{s} \;\; \mathtt{in} \;\; \overline{T} = \mathtt{true} \Rightarrow x : \overline{s}' \;\; \mathtt{in} \;\; \overline{T'} = \mathtt{true})\,. \tag{5}$$

The situation is similar when proving properties of ground terms of sort s for all instances $T_G[\beta]$ of a parameterized theory $T_G[V] \in \mathcal{C}$. Consider the following generalization of the above statement. Let $T_G[V] \in \mathcal{C}$ be a parameterized extension of $T = (\Omega, E)$, with $V = \{v_1, \ldots, v_n\}$, and k_i the kind of the parameter v_i. Then we might formalize that for all instances $T_G[\beta]$ of $T_G[V]$

> **if** *t is a ground term of sort s in $T_G[\beta]$,*
> **then** *t is a ground term of sort s' in T'.*

Again, we cannot express this property as a first-order formula over the signature of any particular instance of $T_G[V]$. Notice, however, that, using the Boolean function $(\_:\_\mathtt{in}\_)$, we can formalize this as

$$\begin{aligned}\forall(\overline{V} : \mathtt{Term}, x : \mathtt{Term}).\, ((\overline{v_1} : \overline{s}_1 \;\; \mathtt{in} \;\; \overline{T} = \mathtt{true} \wedge \ldots \wedge \overline{v_n} : \overline{s}_n \;\; \mathtt{in} \;\; \overline{T} = \mathtt{true}) \\ \Rightarrow \;\; (x : \overline{s} \;\; \mathtt{in} \;\; \overline{T_G[V]} = \mathtt{true} \;\Rightarrow\; x : \overline{s}' \;\; \mathtt{in} \;\; \overline{T'} = \mathtt{true}))\,,\end{aligned} \tag{6}$$

where, for each parameter $v_i \in V$, s_i is a sort in S_{k_i}.

We claim that instances of both (5) and (6) can be proved by induction in a way that mirrors their expected inductive proofs (we will show this for a particular example in Section 8). To see this, the crucial observation—allowing us to mirror inductive reasoning over a sort s in a theory T (or in a parameterized extension $T_G[V]$) by inductive reasoning over `META-IND`—is the following. Let s be a sort in T defined by a set of Horn clauses $\{C_1, \ldots, C_n\}$. By the definition of the Boolean function (`_:_in_`), the set of ground terms u of sort `Term` such that

$$u{:}\overline{s}\ \texttt{in}\ \overline{T} = \texttt{true} \tag{7}$$

is precisely the set of terms of the form $u = \overline{t}$, for t a ground term of sort s in T, and can be defined inductively by a set of Horn clauses $\{\overline{C}_1, \ldots, \overline{C}_n\}$ that reflect, at the metalevel, the set of Horn clauses $\{C_1, \ldots, C_n\}$. The idea is that we can then use $\{\overline{C}_1, \ldots, \overline{C}_n\}$ to derive an induction rule $(\overline{ind})$ to prove metaproperties about the ground terms of sort s in T, in exactly the same way as we obtained the induction rule (ind) from $\{C_1, \ldots, C_n\}$ in Definition 2. Since each $\overline{C}_i$ mirrors at the metalevel the corresponding C_i, inductive metareasoning with $(\overline{ind})$ also mirrors inductive reasoning with (ind). Notice that, when dealing with parameterized extensions, the resulting induction rule $(\overline{ind})$ will have to be universally quantified over the variables representing the parameters.

7.2 Metalevel Inference Rules for Parameterized Theories

We now formalize the above intuitions and introduce a new inference rule for proving a broad class of metatheorems about parameterized membership equational theories. These metatheorems correspond to inductive properties of the initial model of the module `META-IND` of the general form

$$\begin{aligned}\forall(\overline{V} : \texttt{Term}, x : \texttt{Term}).\,((\overline{v}_1{:}\overline{s}_1\ \texttt{in}\ \overline{T} = \texttt{true} \wedge \ldots \wedge \overline{v}_m{:}\overline{s}_m\ \texttt{in}\ \overline{T} = \texttt{true})\\ \Rightarrow\ ((x{:}\overline{s}\ \texttt{in}\ \overline{T_G[V]} = \texttt{true} \wedge \Phi) \Rightarrow \phi))\,,\end{aligned}$$

where ϕ and Φ range over first-order formulae over the signature of the module `META-IND`. The formulae (5) and (6) above are instances of this general form. In the case of (5), the nonparameterized theory T constitutes a trivial parameterized extension $T_\emptyset[\emptyset]$ of itself.

The soundness of the inference rule that we introduce is based on the fact that, for any instance $T_G[\beta]$ of $T_G[V]$, the set of terms of sort `Term` that metarepresent terms of sort s in $T_G[\beta]$ is inductively defined. In essence, the new inference rule reflects at the metalevel the induction principle defined in Definition 2 for reasoning over the terms of sort s in any instance of $T_G[V]$.

First, we define for any parameterized extension $T_G[V] = (\Omega[V], E \cup G) \in \mathcal{C}$, with $\Omega[V] = (K, \Sigma \cup V, S)$, and any sort s in some S_k, a set of clauses $\overline{C}_{[T_G[V],s]}$, that mirrors, at the metalevel, the set of Horn clauses $C_{[T_G[V],s]}$ that inductively define the terms of sort s in $T_G[V]$. (Recall that parameters in V are metarepresented as variables of sort `Term`.) Then, we define an inference rule for proving certain metatheorems about $T_G[V]$.

Definition 4 *Let $T_G[V] = (\Omega[V], E \cup G) \in \mathcal{C}$ be a parameterized extension, with $\Omega[V] = (K, \Sigma \cup V, S)$, and let s_0 be a sort in some S_k, such that $C_{[T_G[V],s_0]} = \{C_1, \dots, C_n\}$ is the set of sentences of the form*

$$\forall(x_1, \dots, x_{p_i}).\, t_1 : s_1 \wedge \dots \wedge t_{q_i} : s_{q_i} \Rightarrow t_0 : s_0$$

that specify s_0 in $T_G[V]$.

We define $\overline{C}_{[T_G[V],s_0]} = \{\overline{C}_1, \dots, \overline{C}_n\}$, where, for $1 \le i \le n$, $\overline{C}_i$ is

$$\begin{aligned}&\forall(x_1, \dots, x_{p_i}).\\ &\quad \hat{t}_1 : \overline{s}_1 \ \mathtt{in}\ \overline{T_G[V]} = \mathtt{true} \wedge \dots \wedge \hat{t}_{q_i} : \overline{s}_{q_i} \ \mathtt{in}\ \overline{T_G[V]} = \mathtt{true}\\ &\qquad \Rightarrow \hat{t}_0 : \overline{s}_0 \ \mathtt{in}\ \overline{T_G[V]} = \mathtt{true}\,,\end{aligned}$$

where $\{x_1, \dots, x_{p_i}\}$ are variables of the kind of the sort `Term`*, and, for $0 \le i \le q_i$, $\hat{t}_i$ is the metarepresentation of the term t_i, except that any variable x in t_i is not metarepresented but, instead, it is replaced by a (meta-) variable x of the kind of the sort* `Term`*. Note that, in general, some clauses in $\overline{C}_{[T_G[V],s_0]}$ may contain free variables.*

Definition 5 *Let $T_G[V] = (\Omega[V], E \cup G) \in \mathcal{C}$ be a parameterized extension, with $\Omega[V] = (K, \Sigma \cup V, S)$ and $V = \{v_1, \dots, v_m\}$. Let s_0 be a sort in some S_k and let $C_{[T_G[V],s_0]} = \{C_1, \dots, C_n\}$ be those sentences of the form*

$$\forall(x_1, \dots, x_{p_i}).\, t_1 : s_1 \wedge \dots \wedge t_{q_i} : s_{q_i} \Rightarrow t_0 : s_0, \tag{8}$$

that specify s_0 in $T_G[V]$. Finally, let τ be a first-order formula, with free variable x of the kind of the sort `Term`*, of the form*

$$\begin{aligned}\forall \overline{V} : \mathtt{Term}.\, ((\overline{v_1} : \overline{z}_1 \ \mathtt{in}\ \overline{T} = \mathtt{true} \ \wedge\ \dots\ \wedge\ \overline{v_m} : \overline{z}_m \ \mathtt{in}\ \overline{T} = \mathtt{true})&\\ \Rightarrow ((x : \overline{s}_0 \ \mathtt{in}\ \overline{T_G[V]} = \mathtt{true} \wedge \Phi) \Rightarrow \phi))\,,&\end{aligned}$$

where, for each parameter $v_i \in V$, $z_i \in S_{k_i}$, for k_i the kind of v_i.

An inductive inference rule for `META-IND`, with respect to x and $\tau(x)$ *is the formula*

$$\begin{aligned}&(\forall \overline{V} : \mathtt{Term}.\, (\overline{v_1} : \overline{z}_1 \ \mathtt{in}\ \overline{T} = \mathtt{true} \ \wedge\ \dots\ \wedge\ \overline{v_m} : \overline{z}_m \ \mathtt{in}\ \overline{T} = \mathtt{true})\\ &\qquad \Rightarrow [\overline{C}_1]_\tau \wedge \dots \wedge [\overline{C}_n]_\tau)\\ &\quad \Rightarrow \forall x : \mathtt{Term}.\, \tau(x)\,,\end{aligned} \tag{9}$$

where, for each C_i of the form (8)*, $\overline{C}_i$ is defined as in* Definition 4*, and $[\overline{C}_i]_\tau$ is the formula*

$$\begin{aligned}&\forall(x_1, \dots, x_{p_i}).\\ &\quad [\hat{t}_1 : \overline{s}_1 \ \mathtt{in}\ \overline{T_G[V]} = \mathtt{true}]_\tau \wedge \dots \wedge [\hat{t}_{q_i} : \overline{s}_{q_i} \ \mathtt{in}\ \overline{T_G[V]} = \mathtt{true}]_\tau\\ &\qquad \Rightarrow\ [\hat{t}_0 : \overline{s}_0 \ \mathtt{in}\ \overline{T_G[V]} = \mathtt{true}]_\tau\end{aligned}$$

where, for $0 \le j \le q_i$,

$$[\hat{t}_j : \overline{s}_j \ \mathtt{in}\ \overline{T_G[V]} = \mathtt{true}]_\tau = \begin{cases}(\Phi \Rightarrow \phi)(\hat{t}_j) & \text{if } s_j = s_0\\ \hat{t}_j : \overline{s}_j \ \mathtt{in}\ \overline{T_G[V]} = \mathtt{true} & \text{otherwise.}\end{cases}$$

The soundness of the inference rule (9) is proved in [2].

7.3 Building a Inductive Metatheorem Prover

In Section 6.3 we indicated how it is possible to use reflection in Maude to design modular, extensible, theorem proving tools. In particular, we explained the reflective design of the `ITP` tool and how we implemented the inference rule (3) for induction over sort definitions. For carrying out formal metatheory, we have also extended the `ITP` tool with the inductive inference rule (9). It is this extended version of the tool that we have used in the experimental work reported on in the next section.

8 An Example

In this section we give an example that illustrates how rewriting logic can be used as a reflective metalogical framework. Our example is a standard one in metareasoning, namely, the deduction theorem.

8.1 The Deduction Theorem for Minimal Logic

We present here the deduction theorem for minimal logic of implication. This theorem is interesting for several reasons. To begin with, it is a central metatheorem that holds for Hilbert systems for many logics and justifies proof under temporary assumption in the manner of a natural deduction system. Moreover, although relatively simple, it illustrates some subtle aspects of formal metareasoning. For example, it is actually a metatheorem not about a particular deduction system, but rather a metatheorem that relates different deduction systems: one in which $A \to B$ is proved, and a second (which is obtained from the first by adding the axiom A) in which B is proved. Indeed, since A is an arbitrary formula, the standard statement of the deduction theorem is actually a statement about the relationship between a *family* of pairs of deduction systems.

For A and B formulae, we write $\vdash_{\mathcal{M}} A$ to denote that A is a theorem in minimal logic, and $\vdash_{\mathcal{M}[A]} B$ to denote that if minimal logic is extended with the additional axiom A, then B belongs to the resulting set of theorems. The deduction theorem then states that, for any formulae A and B in minimal logic,

$$\textbf{if } \vdash_{\mathcal{M}[A]} B \quad \textbf{then} \quad \vdash_{\mathcal{M}} A {\to} B\,. \tag{10}$$

This metatheorem is proven by induction on the structure of derivations in minimal logic extended with the axiom A.

Formalization. Consider now the representation of minimal logic in rewriting logic provided by `MINIMAL` in Figure 2, and its parameterized extension $\texttt{MINIMAL}_{\Xi}[\mathcal{X}]$ introduced in Figure 3. Recall that `MINIMAL` represents minimal logic in the sense that a formula A is a theorem in minimal logic if and only if A is a term of sort `Theorem`. We can rephrase the deduction theorem as follows: for any formulae A and B, if B is a term of sort `Theorem` in $\texttt{MINIMAL}_{\Xi}[A]$, then $A \rightarrow B$ is a term of sort `Theorem` in `MINIMAL`.

Notice that this theorem states an implication between the truth of two membership assertions over two different membership equational theories (in fact, a whole family of such pairs, since A is a parameter). Hence, to formalize the deduction theorem, we must move up a level, to `META-IND`. We claim that the following formula formalizes the deduction theorem as a metatheorem about the initial model of `META-IND`:

$$\begin{aligned}
&\forall (\mathcal{X}, B) : \texttt{Term}. \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (11)\\
&\quad ((\mathcal{X} : \overline{\texttt{Formula}} \texttt{ in } \overline{\texttt{MINIMAL}} = \texttt{true} \wedge B : \overline{\texttt{Formula}} \texttt{ in } \overline{\texttt{MINIMAL}} = \texttt{true})\\
&\quad \Rightarrow (B : \overline{\texttt{Theorem}} \texttt{ in } \overline{\texttt{MINIMAL}_{\Xi}[\mathcal{X}]} = \texttt{true}\\
&\qquad \Rightarrow (\overline{\mathcal{X} \rightarrow B}) : \overline{\texttt{Theorem}} \texttt{ in } \overline{\texttt{MINIMAL}} = \texttt{true}))\,,
\end{aligned}$$

where in the term denoted by $\overline{\mathcal{X} \rightarrow B}$ the (meta-) variables $\mathcal{X}$ and B of sort `Term` are not metarepresented as if they were object level variables, but are instead preserved as (meta-) variables. From now on, we will follow the same convention for terms of this kind, i.e., terms that include elements of sort `Term` in `META-IND`.

When performing metareasoning we must reason about terms being well-sorted with respect to particular theories, in this case, the membership equational theory `MINIMAL`. For this reason, we have explicitly assumed in our formalization of the deduction theorem the well-typedness of $\mathcal{X}$ and B. Of course, standard textbook proofs also require this, but such well-formedness details are usually glossed over as trivial. The correctness of the formalization then follows from the definition of the Boolean function `(_:_in_)` and the fact that `MINIMAL` is a conservative representation of minimal logic.

Note, incidentally, that the requirement that B is a formula in minimal logic is actually superfluous and can be dropped (we will do so for the proof below). It is provable (again by induction) that any theorem in any extension of minimal logic with a new axiom is also a formula in minimal logic.

Proof of the Deduction Theorem We show here how we prove (11). Note that our proof mirrors the standard proof of the deduction theorem.

To prove (11) in `META-IND` we apply the reflected version of the induction principle for the sort `Theorem` in the parameterized extension $\texttt{MINIMAL}_{\Xi}[\mathcal{X}]$, that is, the corresponding instance of the inference rule (9). This reduces proving (11) to proving the formula given in Figure 4. Notice that the four conjuncts correspond to the cases involved in proving the deduction theorem by induction over the proof that B is a theorem in minimal logic extended with the axiom A. The first formalizes the case when B is $\mathcal{X}$. The next two conjuncts formalize the cases where B is either an instance of the K or S axiom schemata. The final conjunct formalizes the case of B being proved by an instance of *modus ponens*.

$$
\begin{array}{l}
\forall \mathcal{X} : \mathtt{Term}.[\\
\quad \mathcal{X}{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}} = \mathtt{true}\\
\quad \Rightarrow\\
\quad [(\overline{\mathcal{X}{\to}\mathcal{X}}){:}\overline{\mathtt{Theorem}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}} = \mathtt{true}\\
\quad \wedge\\
\quad (\forall (A,B).\,(A{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}_{\Xi}[\mathcal{X}]} = \mathtt{true}\ \wedge\\
\qquad\qquad\quad B{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}_{\Xi}[\mathcal{X}]} = \mathtt{true})\\
\qquad \Rightarrow\ (\overline{\mathcal{X}{\to}(A{\to}(B{\to}A))}){:}\overline{\mathtt{Theorem}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}} = \mathtt{true})\\
\quad \wedge\\
\quad (\forall (A,B,C).\,(A{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}_{\Xi}[\mathcal{X}]} = \mathtt{true}\ \wedge\\
\qquad\qquad\qquad B{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}_{\Xi}[\mathcal{X}]} = \mathtt{true}\ \wedge\\
\qquad\qquad\qquad C{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}_{\Xi}[\mathcal{X}]} = \mathtt{true})\\
\qquad \Rightarrow (\overline{\mathcal{X}{\to}((A{\to}B){\to}((A{\to}(B{\to}C)){\to}(A{\to}C)))}){:}\overline{\mathtt{Theorem}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}}\\
\qquad\quad = \mathtt{true})\\
\quad \wedge\\
\quad (\forall (A,B).\,(A{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}_{\Xi}[\mathcal{X}]} = \mathtt{true}\ \wedge\\
\qquad\qquad\quad B{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}_{\Xi}[\mathcal{X}]} = \mathtt{true}\ \wedge\\
\qquad\qquad\quad \overline{\mathcal{X}{\to}(A{\to}B)}{:}\overline{\mathtt{Theorem}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}} = \mathtt{true}\ \wedge\\
\qquad\qquad\quad \overline{\mathcal{X}{\to}A}{:}\overline{\mathtt{Theorem}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}} = \mathtt{true})\\
\qquad \Rightarrow (\overline{\mathcal{X}{\to}B}){:}\overline{\mathtt{Theorem}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}} = \mathtt{true})]]
\end{array}
$$

Fig. 4. Goal resulting after induction

By the theorem of constants for membership equational logic [29], we can reduce proving this formula to proving the four conjuncts that result from replacing the variable $\mathcal{X}$ by a new constant symbol x of sort `Term`, under the assumption that

$$x{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}} = \mathtt{true}. \tag{12}$$

The proof of each of the resulting conjuncts mirrors the proof of the corresponding case in the standard inductive proof of the deduction theorem. In what follows, $\overline{\mathtt{MINIMAL}_{\Xi}[x]}$ denotes the term of sort `Module` that results from $\overline{\mathtt{MINIMAL}_{\Xi}[\mathcal{X}]}$ by replacing the free variable $\mathcal{X}$ of sort `Term` by the new constant symbol x. Consider, for example, how we prove the third conjunct:

$$
\begin{array}{l}
\forall (A,B,C).\\
\quad A{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}_{\Xi}[x]} = \mathtt{true}\ \wedge\\
\quad B{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}_{\Xi}[x]} = \mathtt{true}\ \wedge\\
\quad C{:}\overline{\mathtt{Formula}}\ \mathtt{in}\ \overline{\mathtt{MINIMAL}_{\Xi}[x]} = \mathtt{true}\\
\quad \Rightarrow (\overline{x{\to}((A{\to}B){\to}((A{\to}(B{\to}C)){\to}(A{\to}C)))}){:}\overline{\mathtt{Theorem}}\\
\qquad \mathtt{in}\ \overline{\mathtt{MINIMAL}} = \mathtt{true}).
\end{array} \tag{13}
$$

Using the theorem of constants again, we can reduce proving (13) to proving

$$\overline{x \rightarrow ((a \rightarrow b) \rightarrow ((a \rightarrow (b \rightarrow c)) \rightarrow (a \rightarrow c)))} : \overline{\texttt{Theorem}} \\ \qquad \texttt{in } \overline{\texttt{MINIMAL}} = \texttt{true}\,, \tag{14}$$

under the assumptions that a, b, c are new constants of sort `Term` such that

$$\begin{aligned} &a : \overline{\texttt{Formula}} \texttt{ in } \overline{\texttt{MINIMAL}_{\Xi}[x]} = \texttt{true} \wedge \\ &b : \overline{\texttt{Formula}} \texttt{ in } \overline{\texttt{MINIMAL}_{\Xi}[x]} = \texttt{true} \wedge \\ &c : \overline{\texttt{Formula}} \texttt{ in } \overline{\texttt{MINIMAL}_{\Xi}[x]} = \texttt{true}\,. \end{aligned} \tag{15}$$

Note that, from the assumptions (12) and (15), by using the fact (which must be proven separately) that any theorem in any extension of minimal logic with a new axiom is a well-formed formula in minimal logic, we can derive the formula

$$\begin{aligned} &a : \overline{\texttt{Formula}} \texttt{ in } \overline{\texttt{MINIMAL}} = \texttt{true} \wedge \\ &b : \overline{\texttt{Formula}} \texttt{ in } \overline{\texttt{MINIMAL}} = \texttt{true} \wedge \\ &c : \overline{\texttt{Formula}} \texttt{ in } \overline{\texttt{MINIMAL}} = \texttt{true} \wedge \\ &x : \overline{\texttt{Formula}} \texttt{ in } \overline{\texttt{MINIMAL}} = \texttt{true}\,. \end{aligned} \tag{16}$$

Finally, we prove (14) using the equations in `META-IND` and (16). This proof mirrors the proof that, for any formulae X, A, B, and C,

$$X \rightarrow ((A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow B))) \tag{17}$$

is a theorem in minimal logic; in particular, this proof mirrors proving (17) by *modus ponens*, using the following instance of the S axiom

$$(A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C))$$

and the following instance of the K axiom

$$[(A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C))] \\ \rightarrow (X \rightarrow [(A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C))])\,.$$

8.2 Other Examples and Experience

We have used rewriting logic as a reflective metalogical framework to carry out a number of other proofs in formal metatheory based on more sophisticated versions of the deduction theorem for minimal logic. In particular, we have proved results similar to those of Basin and Matthews [4,5], who have shown how metatheorems that are parameterized by their scope of application can be proved using a theory of parameterized inductive definitions as a metatheory. For example, they present a generalized version of the deduction theorem that can be applied to all extensions of the language and axioms of minimal logic. From their theorem it follows that the deduction theorem holds for the minimal logic of implication and for any propositional extension of it, but not necessarily for extensions to modal logics (which would require adding new rules, as opposed to new axioms). Although rewriting logic is based on a rather different foundation than those considered in [5], our representation of the object logic is quite similar and—abstracting away from the details involved in moving between levels of representation—the basic structure of the proofs is also similar.

One promising area to apply our results is program transformation and metaprogramming. From a reflective declarative point of view, programs that transform other programs are first-order functions acting on terms that metarepresent theories, and the properties that they satisfy are metatheorems, as they are understood in this paper. This reflective declarative methodology has been used in [12] to specify polytypic programs like *map* and *cata* in Maude. Accordingly, polytypic programs are specified as metalevel functions that add to a module the equations defining the desired object function by structural induction over the sort definitions. Properties of polytypic programs, like the functoriality of *map*, are then metatheorems that can be proved, as it is showed in [12], using the corresponding induction rule (Definition 5).

Here we would also like to comment on our experience in proving these theorems and on the issue of managing proofs that combine reasoning at different levels. To the working logician or computer scientist, reflective metalogical frameworks may seem complicated and not particularly user-friendly since there is quite a bit of encoding involved in stating a metatheorem and in carrying out its proof. In particular, reasoning can involve three or more levels (object, meta, meta-meta, ...).[6]

In our case, we have been able to avoid many of the practical problems of working with a reflective hierarchy by exploiting the reflective capabilities of Maude to build tools and suitable interfaces that hide levels of reflection. As part of our work, we have built an interface—fully specified in Maude—to interact with the ITP inductive theorem prover described in Sections 6.3 and 7.3. As already explained, ITP automatically extracts from a theory the induction principles for reasoning over its sorts (Definition 2), and (in its metaprover extension) the induction rules that correspond to reflecting those induction principles at the metalevel when the task at hand is to prove a metatheorem (Definition 5). Proving an inductive theorem then amounts to computing a strategy at the meta-metalevel, or at the meta-meta-metalevel if the theorem is, as in the case of (11), a metatheorem about the initial model of `META-IND`. Fortunately, the interface we use hides all these levels of

[6] Although note that reasoning about a logic encoded as an inductive definition in a logical framework like Isabelle also involves multiple levels, e.g., the framework's metalogic, the theory of inductive definitions, and the object logic. Moreover, there is often an additional language for writing tactics.

encoding from the user. Hence the user can actually abstract away many of the metarepresentation details and focus on the essential structure of proofs of theorems.

9 Conclusion

We have presented, both abstractly and concretely, a new approach to metatheoretic reasoning based on using reflective logical frameworks whose theories have initial models. Initial experiments demonstrate that the machinery for reflective deduction in membership equational logic provides a rich foundation for formalizing and proving metatheorems. Our experiments show, for example, that one can prove metatheorems similar to those provable in logical frameworks based on parameterized inductive definitions, and that one has considerable flexibility in moving between theories and proving theorems that relate theories or establish properties of parameterized classes of theories. In essence, we can do this because the requirements that such metatheorems pose on the metatheory—namely, that one can build families of sets using parameterized inductive definitions and that one can reason about their elements by induction—are realizable in membership equational logic using reflection.

There are a number of directions for further work. One concerns generalizing our notion of a parameterized theory. Currently we can reason at the metalevel about families of theories that are parameterized by sets of new constants and new axioms, which may make use of the new constants. For proving other metatheorems it would be useful to develop a more general theory representation calculus where one could reason at the metalevel about families of theories that are parameterized by arbitrary sets of new sorts, operators, and axioms. In particular, this would allow us to prove metatheorems involving the more general *parameterized modules* of Full Maude [9,18].

Also, our example illustrates how it is possible to carry out proofs similar to those possible in stronger framework logics. However, it would be interesting to have a more formal comparison of the relative strengths of membership equational logic with reflection versus stronger metalogics like higher-order logic or set theory. Finally, related to this is the question of how easy it is to reflect induction principles other than structural induction, e.g., induction over an arbitrary, user-definable well-founded order.

Acknowledgments

This research was supported by DARPA through Rome Laboratories Contract F30602-C-0312, by DARPA and NASA through Contract NAS2-98073, by Office of Naval Research Contract N00014-99-C-0198, and by National Science Foundation Grant CCR-9900334. The authors also thank Narciso Martí-Oliet for his careful reading of a draft of this paper and his detailed suggestions for improving the exposition.

References

1. P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 739–782. North-Holland, Amsterdam, 1977.
2. D. Basin, M. Clavel, and J. Meseguer. Rewriting logic as a metalogical framework. Technical report, September 2000, `http://maude.csl.sri.com`.
3. D. Basin and R. Constable. Metalogical frameworks. In G. Huet and G. Plotkin, editors, *Logical Environments*, pages 1–29. Cambridge University Press, 1993.
4. D. Basin and S. Matthews. Scoped metatheorems. In *International Workshop on Rewriting Logic and its Applications*, volume 15, pages 1–12. Electronic Notes in Theoretical Computer Science (ENTCS), September 1998.
5. D. Basin and S. Matthews. Structuring metatheory on inductive definitions. *Information and Computation*, 2000. To appear.
6. A. Bouhoula, J.-P. Jouannaud, and J. Meseguer. Specification and proof in membership equational logic. *Theoretical Computer Science*, 236:35–132, 2000.
7. M. Clavel. *Reflection in General Logics and in Rewriting Logic with Applications to the Maude Language.* PhD thesis, University of Navarre, 1998.
8. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, and J. Meseguer. Metalevel computation in Maude. In C. Kirchner and H. Kirchner, editors, *Second International Workshop on Rewriting Logic and its Applications*, volume 15 of *Electronic Notes in Theoretical Computer Science*, pages 3–23, Pont-à-Mousson, France, September 1998. Elsevier.
9. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. Quesada. Maude: Specification and programming in rewriting logic. SRI International, January 1999, `http://maude.csl.sri.com`.
10. M. Clavel, F. Durán, S. Eker, and J. Meseguer. Building equational proving tools by reflection in rewriting logic. In *Proceedings of the CafeOBJ Symposium '98, Numazu, Japan.* CafeOBJ Project, April 1998.
11. M. Clavel, F. Durán, S. Eker, J. Meseguer, and M.-O. Stehr. Maude as a formal meta-tool. In J. Wing and J. Woodcock, editors, *FM'99 — Formal Methods*, volume 1709 of *Lecture Notes in Computer Science*, pages 1684–1703. Springer-Verlag, 1999.
12. M. Clavel, F. Durán, and N. Martí-Oliet. Polytypic programming in Maude. To appear in Proc. WRLA 2000, ENTCS, Elsevier, 2000.
13. M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. In J. Meseguer, editor, *First International Workshop on Rewriting Logic and its Applications*, volume 4 of *Electronic Notes in Theoretical Computer Science*, pages 65–89, Asilomar (California), September 1996. Elsevier.
14. M. Clavel and J. Meseguer. Axiomatizing reflective logics and languages. In G. Kiczales, editor, *Proceedings of Reflection'96*, pages 263–288, San Francisco (California), April 1996. Xerox PARC.
15. M. Clavel and J. Meseguer. Reflection and strategies in rewriting logic. In J. Meseguer, editor, *First International Workshop on Rewriting Logic and its Applications*, volume 4 of *Electronic Notes in Theoretical Computer Science*, pages 125–147, Asilomar (California), September 1996. Elsevier.
16. M. Clavel and J. Meseguer. Reflection in condition rewriting logic. Manuscript. Submitted for publication, 2000.
17. J. Despeyroux, F. Pfenning, and C. Schürmann. Primitive recursion for higher-order abstract syntax. In *Proceedings of the 3rd International Conference on*

Typed Lambda Calculi and Applications (TLCA'97), volume 1210 of *Lecture Notes in Computer Science*, Nancy, France, April 1997. Springer-Verlag.
18. F. Durán. *A Reflective Module Algebra with Applications to the Maude Language*. PhD thesis, University of Málaga, 1999.
19. S. Feferman. Finitary inductively presented logics. In *Logic Colloquium '88*. North-Holland, 1988.
20. J. Goguen and J. Meseguer. Models and equality for logical programming. In H. Ehrig, G. Levi, R. Kowalski, and U. Montanari, editors, *Proceedings TAPSOFT'87*, volume 250 of *Lecture Notes in Computer Science*, pages 1–22. Springer-Verlag, 1987.
21. M. Gordon and T. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.
22. R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *J. ACM*, 40(1):143–184, January 1993.
23. N. Martí-Oliet and J. Meseguer. Rewriting logic as a logical and semantic framework. Technical Report SRI-CSL-93-05, SRI International, Computer Science Laboratory, August 1993. To appear in D. Gabbay, ed., *Handbook of Philosophical Logic*, Kluwer Academic Publishers.
24. N. Martí-Oliet and J. Meseguer. General logics and logical frameworks. In D. Gabbay, editor, *What is a Logical System?*, pages 355–392. Oxford University Press, 1994.
25. S. Matthews, A. Smaill, and D. Basin. Experience with FS_0 as a framework theory. In G. Huet and G. Plotkin, editors, *Logical Environments*, pages 61–82. Cambridge University Press, 1993.
26. R. McDowell and D. Miller. A logic for reasoning with higher-order abstract syntax. In *Twelfth Annual IEEE Symposium on Logic in Computer Science*, June 1997.
27. J. Meseguer. General logics. In H.-D. Ebbinghaus et al., editor, *Logic Colloquium'87*, pages 275–329. North-Holland, 1989.
28. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.
29. J. Meseguer. Membership algebra as a semantic framework for equational specification. In F. Parisi-Presicce, editor, *Proceedings of WADT'97*, volume 1376 of *Lecture Notes in Computer Science*, pages 18–61. Springer-Verlag, 1998.
30. J. Meseguer. Research directions in rewriting logic. In U. Berger and H. Schwichtenberg, editors, *Computational Logic, NATO Advanced Study Institute, Marktoberdorf, Germany, July 29 - August 6, 1997*. Springer-Verlag, 1998.
31. J. Meseguer and J. A. Goguen. Initiality, induction and computability. In M. Nivat and J. C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 459–541. Cambridge University Press, 1985.
32. C. Paulin-Mohring. Inductive Definitions in the System Coq — Rules and Properties. In M. Bezem and J.-F. Groote, editors, *Proceedings of the conference Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, 1993.
33. L. C. Paulson. A fixedpoint approach to implementing (co)inductive definitions. In *Proceedings of the 12th International Conference on Automated Deduction (CADE-12)*, volume 814 of *Lecture Notes in Artificial Intelligence*, Nancy, France, June 1994. Springer-Verlag.

34. L. C. Paulson. *Isabelle : a generic theorem prover; with contributions by Tobias Nipkow*, volume 828 of *Lecture Notes in Computer Science*. Springer, Berlin, 1994.
35. C. Schürmann and F. Pfenning. Automated theorem proving in a simple meta-logic for LF. In C. Kirchner and H. Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction (CADE-15)*, volume 1421 of *Lecture Notes in Computer Science*, pages 286–300, Lindau, Germany, July 1998. Springer-Verlag.
36. M.-O. Stehr and J. Meseguer. Pure type systems in rewriting logic. In *Proc. of LFM'99: Workshop on Logical Frameworks and Meta-languages*, 1999. `http://www.cs.bell-labs.com/~felty/LFM99/`.
37. M.-O. Stehr, P. Naumov, and J. Meseguer. A proof-theoretic approach to the HOL-Nuprl connection with applications to proof translation. Manuscript, SRI International, `http://www.csl.sri.com/~stehr/fi_eng.html`, February 2000.
38. M. van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *J. ACM*, 23:733–42, 1976.