A category-theoretic treatment of automated theorem proving

Maria Paola Bonacina [*]
Department of Computer Science
University of Iowa
Iowa City, IA 52242-1419, USA
bonacina@cs.uiowa.edu

Jieh Hsiang [†]
Department of Computer Science
National Taiwan University
Taipei, Taiwan
hsiang@csie.ntu.edu.tw

Abstract

In this paper we apply *category theory* to investigate the mathematical structure of theorem proving derivations.

A *theorem-proving strategy* is given by a set of *inference rules* and a *search plan*. Search plans have been usually described either informally (e.g., a criterion to select the next inference step) or procedurally (e.g., by giving a specific algorithm). Since both the completeness and efficiency of a theorem-proving strategy depend on the search plan, a formal, abstract treatment appears desirable. We propose an approach in this direction, that allows in particular for a precise definition of how the inference rules and the search plan cooperate to generate the *derivations*. Theorem-proving derivations are characterized by three essential properties: *soundness*, *relevance* and *proof reduction*. We show that they are *functoriality properties*: these results clarify which parts of the structure of a theory a theorem-proving derivation is required to preserve. We close the paper with a comparison with related work and a discussion on further extensions of our approach.

Keywords: theorem proving, search, category theory, simplification, contraction.

1 Introduction

In this paper we apply category theory to extend the framework for automated theorem proving that we developed in [1]. Our approach was originally inspired by *(Knuth-Bendix) completion-based strategies* for theorem proving (e.g., [2, 3, 4, 5, 6, 7], more references can be found in [8]). The interest in these methods is motivated mainly by their effectiveness, as demonstrated by the success of the theorem provers implementing them (e.g., [9, 10, 11]).

A *theorem-proving problem* can be specified by a pair $(S; \varphi)$, where S is a set of sentences, called the *presentation* of a theory, and φ is the theorem, called the *target*, to be proved from

[*]Supported in part by the National Science Foundation with grant CCR-94-08667 and by the GE Foundation Faculty Fellowship to the University of Iowa.

[†]Supported in part by grant NSC 84-2213-E-002-011 of the National Science Council of the Republic of China.

S. A *theorem-proving strategy* $\mathcal{C} =< I; \Sigma >$ has two components, a set of *inference rules*, or *inference system*, I, and a *search plan*, Σ. The inference rules determine what consequences can be derived from a given set of sentences. Based on their effect on the database, inference rules can be classified into *expansion inference rules* and *contraction inference rules*. Expansion inference rules are those that generate new sentences, such as *Resolution* [12], and thus "expand" the database. Contraction inference rules are those that may delete sentences, such as *Simplification* [2, 6] and *Subsumption* [13], and thus "contract" the database. Many theorem-proving strategies, completion-based strategies in particular, derive their effectiveness from the eager and systematic application of powerful contraction rules.

The search plan controls the application of the inference rules, thereby constructing a derivation:

$$(S_0; \varphi_0) \underset{\mathcal{C}}{\vdash} (S_1; \varphi_1) \underset{\mathcal{C}}{\vdash} \ldots (S_i; \varphi_i) \underset{\mathcal{C}}{\vdash} \ldots, \tag{1}$$

where at each stage the search plan selects the inference rule and the premises for the next step. Depending on the logic and the strategy, the components in the tuple of the derivation may vary. For instance, in many refutational, resolution-based strategies, the negation of the input target may be added to the input presentation and the target component is not used:

$$S_0 \underset{\mathcal{C}}{\vdash} S_1 \underset{\mathcal{C}}{\vdash} \ldots S_i \underset{\mathcal{C}}{\vdash} \ldots. \tag{2}$$

Such a derivation will succeed at stage k if S_k contains the empty clause $\Box$, denoting inconsistency and thus proving by refutation the input target. In other strategies, including some refutational ones such as linear resolution for Horn logic, derivations can be written in the form of (1). For instance, for equational logic we may have:

$$(S_0; s_0 \simeq t_0) \underset{\mathcal{C}}{\vdash} (S_1; s_1 \simeq t_1) \underset{\mathcal{C}}{\vdash} \ldots (S_i; s_i \simeq t_i) \underset{\mathcal{C}}{\vdash} \ldots, \tag{3}$$

where the presentation is a set of equations, and all variables in the target are universally quantified. A derivation in this form will succeed, if, for some stage k, s_k and t_k are identical terms. Success will be denoted by a final stage $(S_{k+1}; true)$, where the target has been reduced to the dummy target *true* and the derivation halts.

The first part of this paper treats strategies, thus inference rules and search plans, and examines theorem proving in general. Although the treatment is given for derivations of type (1), it applies also to derivations of type (2). The main contribution here is the system of mathematical definitions of the search plan and inference rules and their interplay in the generation of derivations. The search plan is the *control* that turns a set of non-deterministic inference rules into a deterministic procedure. Traditionally, the inference rules are regarded as the logical component of theorem proving, whereas the search plan is the procedural component. As a consequence, search plans are often described informally or by giving explicit algorithms. We feel that the procedural nature of search mechanisms does not eliminate the need for a rigourous definition. Indeed, search is such an essential component in theorem proving, that the problem of giving a systematic treatment of search plans and their properties is very important. We propose a definition of search plan as a pair of functions used to choose an inference rule and a tuple of sentences and we show how the inference rules and the search plan interact to generate a derivation. This result also uses a characterization of *inference rules as natural transformations*, which allows us to cover in a uniform way both expansion inference rules and contraction inference rules.

In the second part, we focus on derivations and their properties, concentrating on derivations of type (1) and (3). We associate to a theorem-proving derivation two underlying categories: the category of the *states* of a derivation, i.e. the pairs $(S_i; \varphi_i)$, and the category of the *proofs* in S_i of the theorems of S_i. First, we discuss *soundness* and *relevance*, and we show that they can be captured by the existence of functors from the category of the states of the derivation to other appropriately defined categories. Then, we explain the relationship between derivations and the underlying proofs. At each stage of a derivation, the tuple $(S_i; \varphi_i)$ represents the problem of proving the target φ_i from the presentation S_i. The key concept in our approach is to regard the problem of proving φ_i from S_i as the problem of *reducing a proof of* φ_i *in* S_i. Accordingly, a theorem-proving derivation is a *process of proof reduction* with respect to a *well-founded proof ordering*. The derivation is successful if it reaches a stage where the proof of φ_i in S_i is *empty*, i.e. φ_i is a trivially true theorem in S_i. Proof reduction is then expressed naturally as a *functoriality property*: a derivation is proof-reducing if the function that associates to a state $(S_i; \varphi_i)$ a minimal proof of φ_i in S_i is a functor from the category of the states of the derivation to the category of the ordered proofs.

We assume that the reader is familiar with the basic definitions such as those of term, substitution, equation, clause, and sentence; we refer the reader to [13] for definitions of inference rules, such as resolution and subsumption, to [8], for notations and terminology specific to completion procedures, and to [14] for the basic notions of category, functor and natural transformation.

2 Preliminary definitions

In this section we introduce terminology and notations from [15] that will be used in the following.

A first-order *signature* Θ is a pair (F, P), where F is a set of function symbols with their arities and P is a set of predicate symbols with their arities. Given two first-order signatures Θ and Θ', a *signature morphism* $h: \Theta \to \Theta'$ is given by two arity-preserving functions, mapping, respectively, the function symbols of Θ into function symbols of Θ' and the predicate symbols of Θ into predicate symbols of Θ'. Signatures and their morphisms form the category[1] $\underline{Sign}$. If there is a morphism $h: \Theta \to \Theta'$, a sentence φ on signature Θ can be translated into a sentence $h(\varphi)$ on signature Θ' by replacing each symbol in Θ with the corresponding symbol in Θ'. It is then possible to define a functor $sen: \underline{Sign} \to \underline{Set}$ from the category of signatures to the category of sets: sen associates to a signature Θ the set $sen(\Theta)$ of its sentences and to a morphism $h: \Theta \to \Theta'$ the function $sen(h): sen(\Theta) \to sen(\Theta')$ translating sentences according to h. Given signatures and sentences, the notion of *entailment* is introduced as a function $\vdash$, associating to each Θ in $\underline{Sign}$

[1]Following the notational conventions of [15], we use names with uppercase initials for sets and underlined names with uppercase initials for categories.

its entailment relation $\vdash_\Theta \subseteq \mathcal{P}(sen(\Theta)) \times sen(\Theta)$, where $\mathcal{P}$ is the power set functor $\mathcal{P}\colon \underline{Set} \to \underline{Set}$. For a set Γ of sentences and a sentence φ, both on signature Θ, $\Gamma \vdash_\Theta \varphi$ means that φ is derivable from Γ.

The category $\underline{Th}$ of *theories* has as objects the pairs (Θ, Γ), where Θ is a signature and Γ is a set of sentences on signature Θ (i.e., $\Gamma \subset sen(\Theta)$). A *morphism of theories*, $h\colon S \to S'$, for $S = (\Theta, \Gamma)$ and $S' = (\Theta', \Gamma')$, is a morphism of signatures $h\colon \Theta \to \Theta'$, with the property that $\Gamma' \vdash_{\Theta'} h(\Gamma)$. In other words, the target theory entails the source theory. The functor $sign\colon \underline{Th} \to \underline{Sign}$, defined by $sign(\Theta, \Gamma) = \Theta$, extracts the signature of a given theory. The functor $sen\colon \underline{Sign} \to \underline{Set}$ can be extended to a functor $sen\colon \underline{Th} \to \underline{Set}$ by defining $sen(S) = sen(sign(S))$. The function $\vdash$ can also be extended to theories: it associates to a theory $S = (\Theta, \Gamma)$ the relation $\vdash_S$ defined by $\Delta \vdash_S \varphi$ if and only if $\Delta \cup \Gamma \vdash_\Theta \varphi$.

The model-theoretic counterpart of entailment is the function $\models$ that associates to each signature Θ its *satisfaction* relation $\models_\Theta$. Similar to entailment, satisfaction can be extended to theories: $\Delta \models_S \varphi$ for $S = (\Theta, \Gamma)$, if and only if $\Delta \cup \Gamma \models_\Theta \varphi$. A logic is *sound* if entailment implies satisfaction: for all $\Theta \in \underline{Sign}$, $\Gamma \subset sen(\Theta)$ and $\varphi \in sen(\Theta)$, $\Gamma \vdash_\Theta \varphi$ implies $\Gamma \models_\Theta \varphi$. If the opposite implication also holds, the logic is said to be *sound* and *complete*. In the following we shall always assume that the logic underlying our theorem-proving problem is sound and complete.

3 Theorem proving strategies

In this section, we present the definitions of search plans, inference rules and derivations, and we introduce the category of the states of a derivation.

3.1 Inference rules and derivations

Inference rules are mechanical rules by which a strategy derives consequences from the given data. There are different ways of representing inference rules, depending on the purpose of the representation. Traditionally, inference rules are written in the form

$$f\colon \; \frac{\psi_1 \ldots \psi_n}{\psi} \tag{4}$$

where rule f derives ψ from premises $\psi_1 \ldots \psi_n$. For example, binary clausal resolution is defined as

$$\frac{L_1 \vee D, \neg L_2 \vee C}{(C \vee D)\sigma} \; L_1\sigma = L_2\sigma \tag{5}$$

where σ is the most general unifier of L_1 and L_2. One purpose of this representation is the representation of *proofs as trees*, e.g. tree-resolution proofs. By representing inference rules as in (4), the consequence, ψ, and the premises, $\psi_1 \ldots \psi_n$, are respectively the root and the leaves of a subtree of depth one. By composing several such subtrees, e.g. by attaching to the premises the subtrees of the inferences generating them, one obtains a proof tree with the proved theorem as the root and original axioms as the leaves.

Our treatment of inference rules is slightly different from the conventional one. This is because we emphasize the effect of inferences on the database. That is, we treat inference rules as rules transforming sets, as shown in the following for resolution and simplification respectively:

$$\frac{S \cup \{L_1 \vee D, \neg L_2 \vee C\}}{S \cup \{L_1 \vee D, \neg L_2 \vee C, C\sigma \vee D\sigma\}} \; L_1\sigma = L_2\sigma \qquad (6)$$

$$\frac{S \cup \{\psi[s], l \simeq r\}}{S \cup \{\psi[r\sigma], l \simeq r\}} \; s = l\sigma \quad \psi[l\sigma] \succ \psi[r\sigma] \qquad (7)$$

where σ is the matching substitution. One can observe that an expansion rule such as (6) adds a new resolvent to the database while rule (7) replaces an equation by an equivalent but simpler one.

Our treatment of inference rules makes it more natural to study theorem-proving derivations, which is our goal.

For a given theorem-proving problem $(S_0; \varphi_0)$ and strategy $\mathcal{C} =< I; \Sigma >$, we call *$I$-tree rooted at* $(S_0; \varphi_0)$ the tree such that the nodes are labelled by pairs $(S; \varphi)$, the root is labelled by $(S_0; \varphi_0)$ and a node $(S; \varphi)$ has a child $(S'; \varphi')$ if $(S'; \varphi')$ can be derived from $(S; \varphi)$ in one step by I [1]. Thus, an arc in an I-tree corresponds to a single inference step, and a derivation corresponds to a path in the tree, which we call an *I-path*. Nodes labelled by a pair of the form $(S; \mathit{true})$ are *successful nodes*. The I-tree rooted at $(S_0; \varphi_0)$ represents all the possible derivations by I from $(S_0; \varphi_0)$. Since, in general, many inference steps are applicable to a given state $(S_i; \varphi_i)$, in the corresponding I-tree, a node $(S_i; \varphi_i)$ may have many children. The search plan Σ is a mechanism that, given a node $(S_i; \varphi_i)$, selects one specific inference step and, thus, one node among its children. The selected node will be the successor $(S_{i+1}; \varphi_{i+1})$ in the derivation. The repeated application of Σ selects an I-path in the I-tree, that is the *unique* derivation computed by the strategy $\mathcal{C}$ on input $(S_0; \varphi_0)$:

$(S_0; \varphi_0) \vdash_{\mathcal{C}} (S_1; \varphi_1) \vdash_{\mathcal{C}} \ldots (S_i; \varphi_i) \vdash_{\mathcal{C}} \ldots$.

Thus, a derivation is a "linear" object, extracted from the non-linear search space (the I-tree) of all the possible inferences. Of course, a derivation may generate proofs of any sort, not necessarily linear proofs, but the derivation *itself* has a "linear" structure[2].

To summarize, we are interested in studying derivations, which are linear objects. Thus, we need a representation of inference rules and derivations that enables us to describe them. Since the conventional representation, such as (4), is geared towards representing proofs as trees, it cannot be used for this purpose.

[2]This usage of the word "linear" should not be confused with the usual notion of linear derivation in theorem proving, such as linear resolution, which generates linear proofs.

3.2 Inference rules as natural tranformations

Inference rules apply to premises and the effect of inferences is to generate new elements (expansion) and delete existing elements (contraction). Therefore, we define inference rules as *natural transformations* from tuples of sentences to pairs of sets of sentences, which are those to be generated and those to be deleted:

Definition 3.1 An inference rule f^n of arity n is a natural transformation

$$f^n\colon \mathcal{L}^n \circ sen \Rightarrow (\mathcal{P} \circ sen) \times (\mathcal{P} \circ sen),$$

where $\mathcal{L}^n\colon \underline{Set} \to \underline{Set}$ is a functor that maps a set into the set of its n-tuples.

By definition of natural transformation [14], an inference rule f^n associates to each signature $\Theta \in \underline{Sign}$ a function

$$f^n_\Theta\colon \mathcal{L}^n \circ sen(\Theta) \to (\mathcal{P} \circ sen(\Theta)) \times (\mathcal{P} \circ sen(\Theta))$$

from tuples of sentences to pairs of sets of sentences of Θ. For an input tuple of premises $\bar{x} = (\psi_1, \ldots, \psi_n)$, the first component of the output, denoted[3] by $\pi_{21}(f^n_\Theta(\bar{x}))$, is the set of sentences to be added, and the second component, $\pi_{22}(f^n_\Theta(\bar{x}))$, is the set of sentences to be deleted. The commutativity property required by the definition of natural transformation is satisfied since inferences are independent of signature morphisms. For every signature morphism $h\colon \Theta \to \Theta'$, the following diagram commutes:

$$\begin{tikzcd}
\mathcal{L}^n \circ sen(\Theta) \arrow[r, "f^n_\Theta"] \arrow[d, "\mathcal{L}^n \circ sen(h)"'] \arrow[dr, phantom, "///"] & (\mathcal{P} \circ sen)^2(\Theta) \arrow[d, "(\mathcal{P} \circ sen)^2(h)"] \\
\mathcal{L}^n \circ sen(\Theta') \arrow[r, "f^n_{\Theta'}"'] & (\mathcal{P} \circ sen)^2(\Theta')
\end{tikzcd}$$

meaning that applying first the inference rule and then translating the result according to h has the same effect as translating first the premises and then applying the inference rule. The following example shows how Definition 3.1 applies to simple steps of resolution, simplification and subsumption:

Example 3.1 $resolution(P(0), \neg P(x) \vee P(s(x))) = (\{P(s(0))\}, \emptyset)$,
$simplification(x + 0 \simeq x, P(s(x) + 0)) = (\{P(s(x))\}, \{P(s(x) + 0)\})$,
$subsumption(P(x), P(s(y))) = (\emptyset, \{P(s(y))\})$.

Then, the distinction between expansion and contraction rules can be formalized as follows:

Definition 3.2 An inference rule f^n is an *expansion inference rule* if for all signatures Θ and inputs $\bar{x} \in \mathcal{L}^n \circ sen(\Theta)$, $\pi_{22}(f^n_\Theta(\bar{x}))$ is *empty*. It is a *contraction inference rule* otherwise.

[3]We use π_{ni} to denote the projection that extracts the i-th element of an n-tuple.

We conclude by remarking that an inference rule may not apply to a given tuple of premises. For example, resolution does not apply to two parents such that no pair of their literals is unifiable. Definition 3.1 covers this case without resorting to partial functions: if the inference rule does not apply to the given input, both output sets are empty. This reflects naturally what happens in a theorem-proving derivation: if the search plan has selected an inference rule and a tuple of premises such that the inference rule does not apply to the premises, the attempted application fails, and the database remains unchanged.

Another advantage of defining inference rules as total functions is that, when evaluating the complexity (cost) of derivations, we may want to take into account the failed inference steps, such as failed unification attempts. Including failed inference steps reflects more accurately the actual costs of a derivation.

3.3 Derivations as morphisms

Having defined the inference rules, we can proceed to define the notion of *derivability* by an inference system I. First, we define formally a state of a derivation as a pair $(S;\varphi)$, where $S = (\Theta, \Gamma)$ is a theory in $\underline{Th}$ and $\varphi \in sen(S)$, and we call *States* the set of all such pairs. Then, we define derivability as a relation on *States*:

Definition 3.3 Given an inference system I, the relation $\vdash_I$ is a binary relation on *States*, such that, for all $(S;\varphi), (S';\varphi') \in States$ with $S = (\Theta, \Gamma)$ and $S' = (\Theta, \Gamma')$, $(S;\varphi) \vdash_I (S';\varphi')$ holds if and only if there exist an inference rule $f^n \in I$ and an n-tuple $\bar{x}$ in $\Gamma \cup \{\varphi\}$, such that

- either $\varphi' = \varphi$ and $\Gamma' = \Gamma \cup \pi_{21}(f^n_\Theta(\bar{x})) - \pi_{22}(f^n_\Theta(\bar{x}))$,
- or $\Gamma' = \Gamma$, φ is in $\bar{x}$, $\pi_{21}(f^n_\Theta(\bar{x})) = \{\varphi'\}$ and $\pi_{22}(f^n_\Theta(\bar{x})) = \{\varphi\}$.

The application of f^n_Θ to $\bar{x}$ gives two sets of sentences, $\pi_{21}(f^n_\Theta(\bar{x}))$ and $\pi_{22}(f^n_\Theta(\bar{x}))$, with the meaning that the contents of $\pi_{21}(f^n_\Theta(\bar{x}))$ should replace the contents of $\pi_{22}(f^n_\Theta(\bar{x}))$. The replacement can be performed either on the presentation or on the target[4]. In the second case, the target φ is among the premises, $\pi_{22}(f^n_\Theta(\bar{x}))$ contains φ and $\pi_{21}(f^n_\Theta(\bar{x}))$ contains φ'.

The relation of derivability by an inference system I induces some structure on the set *States*, yielding the category $\underline{States_I}$:

Definition 3.4 Given an inference system I, the category $\underline{States_I}$ has as objects all the elements in *States*, and as morphisms all the *state morphisms* defined as follows: for all $(S;\varphi)$ and $(S';\varphi')$ in $\underline{States_I}$, there is a morphism $d\colon (S;\varphi) \to (S';\varphi')$ if and only if there exists a derivation $(S;\varphi) \vdash^*_I (S';\varphi')$. A morphism $d\colon (S;\varphi) \to (S';\varphi')$ is a function such that

$d(\varphi) = \varphi'$ and

for all ψ in S, $d(\psi) = \begin{cases} \psi & \text{if } \psi \in S' \\ \varphi' & \text{otherwise.} \end{cases}$

[4]In practice, an inference step may modify both the target and the presentation [1]. Such steps may be obtained as the composition of a target step and a presentation step.

A morphism of states $d: (S;\varphi) \rightarrow (S';\varphi')$ preserves the target by mapping φ into φ'. For the presentation, d is the identity function everywhere except on those sentences of S that have been deleted by contraction steps in the process of deriving S' from S. These sentences have been deleted because they are not necessary to prove the target φ'. Thus the morphism d maps them into φ', which is the "justification" of their deletion. Reflexivity and transitivity of $\vdash_I^*$ induce the identity and composition of morphisms: the identity morphism maps every state into itself by an empty derivation, and the composition of morphisms corresponds to the concatenation of inference steps.

The category $\underline{States}_I$ describes all the derivations by the inference mechanism I. If a specific theorem-proving problem $(S_0;\varphi_0)$ is given, the application of I to $(S_0;\varphi_0)$ defines the *subcategory* of $\underline{States}_I$ that contains all and only the states derivable from $(S_0;\varphi_0)$ by I, with their morphisms. We denote this subcategory as $\underline{States}_I(S_0;\varphi_0)$. The I-tree rooted at $(S_0;\varphi_0)$ is a representation of $\underline{States}_I(S_0;\varphi_0)$. If a search plan Σ is also given, the specific derivation

$$(S_0;\varphi_0) \vdash_{\mathcal{C}} (S_1;\varphi_1) \vdash_{\mathcal{C}} \ldots \vdash_{\mathcal{C}} (S_i;\varphi_i) \vdash_{\mathcal{C}} \ldots$$

computed by the strategy $\mathcal{C} =< I;\Sigma >$ is determined. This derivation is the *subcategory* $\underline{Der}_{\mathcal{C}}(S_0;\varphi_0)$ of $\underline{States}_I(S_0;\varphi_0)$, that contains the states $(S_i;\varphi_i)$, $i \geq 0$, generated during the derivation, with their morphisms. These morphisms are all the sequences of inference steps that are subsequences of the derivation. To summarize, an inference system I induces the morphisms of $\underline{States}_I$; if an initial state $(S_0;\varphi_0)$ is given, the attention is restricted to the subcategory $\underline{States}_I(S_0;\varphi_0)$; a search plan Σ determines a unique derivation $\underline{Der}_{\mathcal{C}}(S_0;\varphi_0)$. In the next section we describe how the search plan generates a derivation.

3.4 The search plan

At each stage of a derivation, the search plan determines the successor of the current state by selecting the inference rule to be applied and the premises for the application of the rule. Thus, we define formally a search plan as given by two components, $\Sigma =< \zeta, \xi >$, a function ζ to choose the inference rule and a function ξ to choose the tuple of premises.

The domains of ζ and ξ need to be defined in such a way that they contain the information used in the selection. Most of this information is usually part of the current state of the derivation. For instance, a search plan may sort all the clauses in the current set according to a certain ordering, which is then part of the search plan, and pick the smallest clause as a premise. Therefore, it is natural that ζ and ξ are functions of the current state. However, the actual state of the derivation represents only *local* information whereas a search plan may also take into account the *history* of the derivation. For instance, a derivation may reach a state $(S;\varphi)$ after 100 steps whereas another derivation may reach the same state $(S;\varphi)$ after 10,000 steps. Conceivably, a search plan may make different choices in the two cases. Since the history of a derivation is a sequence of states, including the current state, the components of a search plan will be functions of sequences of states. For the *rule-selecting function* ζ, we have

$$\zeta: States^* \rightarrow I.$$

The domain of the *premises-selecting function* ξ will be $States^* \times I$, because the selection of the premises may be affected by the knowledge of the inference rule the premises are for. The codomain of ξ will be a set of natural transformation from sets of sentences (the database) to tuples of sentences (the selected premises):

$$\xi\colon States^* \times I \rightarrow \{\alpha^n \mid \alpha^n\colon \mathcal{P} \circ sen \Rightarrow \mathcal{L}^n \circ sen, \forall n \geq 1\}.$$

By the definition of natural transformation, a natural transformation α^n provides for all signatures $\Theta \in \underline{Sign}$ a function

$$\alpha^n_\Theta\colon \mathcal{P} \circ sen(\Theta) \rightarrow \mathcal{L}^n \circ sen(\Theta)$$

that selects a tuple of sentences from a set of sentences. The following definition summarizes these elements:

Definition 3.5 A search plan Σ for a set of inference rules I is a pair of functions $\Sigma =< \zeta, \xi >$, called the *rule-selecting function* and the *premises-selecting function*, respectively, such that

1. $\zeta\colon States^* \rightarrow I$,
2. $\xi\colon States^* \times I \rightarrow \{\alpha^n \mid \alpha^n\colon \mathcal{P} \circ sen \Rightarrow \mathcal{L}^n \circ sen, \forall n \geq 1\}$ and
3. if $\zeta((S_0;\varphi_0),(S_1;\varphi_1),\ldots,(S_i;\varphi_i)) = f^n$, then $\xi((S_0;\varphi_0),(S_1;\varphi_1),\ldots,(S_i;\varphi_i),f^n) = \alpha^n$, such that $\alpha^n_\Theta(\Gamma_i \cup \{\varphi_i\}) \in (\Gamma_i \cup \{\varphi_i\})^n$, where $S_i = (\Theta, \Gamma_i)$.

Condition 3 says that if ζ selects an inference rule f^n of arity n, then ξ selects by α^n an n-tuple of premises from the database of the current state.

We can now define how the inference rules and the search plan cooperate to generate a derivation:

Definition 3.6 For all $S_0 = (\Theta, \Gamma_0) \in \underline{Th}$ and $\varphi_0 \in sen(\Theta)$, the derivation computed by a theorem-proving strategy $\mathcal{C} =< I, \Sigma >$, with inference rules I and search plan $\Sigma =< \zeta, \xi >$, on input $(S_0;\varphi_0)$ is the sequence

$(S_0;\varphi_0) \vdash_\mathcal{C} (S_1;\varphi_1) \vdash_\mathcal{C} \ldots \vdash_\mathcal{C} (S_i;\varphi_i) \vdash_\mathcal{C} \ldots$

where, for all $i \geq 0$, $S_i = (\Theta, \Gamma_i)$ and, if

- $\zeta((S_0;\varphi_0),(S_1;\varphi_1),\ldots,(S_i;\varphi_i)) = f^n$,
- $\xi((S_0;\varphi_0),(S_1;\varphi_1),\ldots,(S_i;\varphi_i),f^n) = \alpha^n$ and
- $\alpha^n_\Theta(\Gamma_i \cup \{\varphi_i\}) = \bar{x}$,

then

$$(S_{i+1};\varphi_{i+1}) = \begin{cases} (S_i;\varphi') & \text{if } \varphi_i \text{ is in } \bar{x},\ \pi_{21}(f^n_\Theta(\bar{x})) = \{\varphi'\} \text{ and} \\ & \pi_{22}(f^n_\Theta(\bar{x})) = \{\varphi_i\}, \\ (S_i \cup \pi_{21}(f^n_\Theta(\bar{x})) - \pi_{22}(f^n_\Theta(\bar{x}));\varphi_i) & \text{otherwise.} \end{cases}$$

The core of the definition is the interaction of I and $\Sigma =< \zeta, \xi >$. That is, ζ chooses the inference rule f^n, ξ chooses the tuple of premises $\bar{x}$, and the application of f^n_Θ to $\bar{x}$ determines $(S_{i+1}; \varphi_{i+1})$.

A search plan Σ is *fair*, if whenever the I-tree rooted at $(S_0; \varphi_0)$ contains successful nodes, Σ is able to reach one such node. In a refutational setting, the complementary property of the inference system is *refutational completeness*: whenever φ_0 is a theorem of S_0, the I-tree contains successful nodes. If I is refutationally complete and Σ is fair, the strategy $\mathcal{C} =< I; \Sigma >$ is *complete*. A complete strategy is a semidecision procedure for the intended logic: whenever the input target is indeed a theorem of the input presentation, the procedure is guaranteed to halt successfully.

We should emphasize that fairness is a property *independent* of the properties of the inference system. Indeed, designing fair, yet non-exhaustive, search plans is one of the most difficult challenges in automated theorem proving.

4 Theorem proving derivations

In this second part of the paper, we study the properties of theorem-proving derivations as functoriality properties. To avoid repetition, we will assume in the rest of the paper that $\mathcal{C}$ is a strategy with inference system I and search plan Σ, and that a derivation $\underline{Der}_{\mathcal{C}}(S_0; \varphi_0)$ has signature $\Theta = sign(S_0)$ and states $(S_i; \varphi_i)$, where $S_i = (\Theta, \Gamma_i)$.

4.1 Soundness, relevance and monotonicity

A basic requirement for the inference system of a theorem-proving strategy is soundness. In derivations where the presentation and the target are separated, soundness is expressed in terms of two properties, *soundness* and *relevance*. In order to discuss these properties, we introduce a functor [5] $th \colon \underline{Th} \to \underline{Set}$, such that for all theories $S = (\Theta, \Gamma)$, $th(S) = \{\psi \mid \Gamma \models_\Theta \psi, \psi \in sen(\Theta)\}$. In other words, $th(S)$ is the set of theorems of S.

Definition 4.1 *[1]* An inference step $(S; \varphi) \vdash_I (S'; \varphi')$ is *sound* if $th(S') \subseteq th(S)$, *monotonic* if $th(S) \subseteq th(S')$, and *relevant* if $\varphi' \in th(S')$ if and only if $\varphi \in th(S)$.

Soundness states the requirement that inferences do not add new elements which are not true in the theory. Monotonicity is the dual property that all the theorems in the given theory are preserved. Relevance ensures that a target inference step replaces the target with a new target in such a way that proving the latter is equivalent to proving the former. A derivation is sound, monotonic and has the relevance property if all its steps are sound, monotonic and relevant respectively. Soundness and monotonicity together obviously imply relevance. For theorem-proving derivations where the target can be singled out, however, soundness and relevance are sufficient. In theorem proving, one is interested only in proving a specific target; therefore, it is not necessary to preserve the other theorems (monotonicity) as long as the intended theorem is preserved (relevance).

[5] The functoriality of $th \colon \underline{Th} \to \underline{Set}$ is an immediate consequence of the definition of theory morphism, under the assumption that the logic is sound and complete.

These properties indicate parts of the structure of a theory that theorem-proving derivations need to preserve. Indeed, they can be rephrased as *functoriality properties.* In order to extract the presentation from a given state, we define the projection function $pres\colon States \rightarrow Th$ by $pres((S;\varphi)) = S$, for all $(S;\varphi)$ in $States$. We also recall that, given a category $\underline{A}$, the opposite category, denoted by $\underline{A}^{op}$, is the category which is identical to $\underline{A}$ except that all morphisms are inverted.

Theorem 4.1 *A derivation* $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ *is sound if and only if the restriction of the projection function* $pres\colon States \rightarrow Th$ *to the subcategory* $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ *of* $\underline{States_I}$ *is a functor* $pres\colon \underline{Der_{\mathcal{C}}}(S_0;\varphi_0) \rightarrow \underline{Th}^{op}$*.*

Proof:

$\Rightarrow$) In order to prove that $pres$ is a functor, we need to show that it associates to every morphism d in $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ a morphism $pres(d)$ in $\underline{Th}^{op}$:

$$\begin{tikzcd}
(S_j;\varphi_j) \arrow[r, "d"] \arrow[d, "pres"'] & (S_{j+k};\varphi_{j+k}) \arrow[d, "pres"] \\
S_j \arrow[r, "pres(d)"'] & S_{j+k}
\end{tikzcd}$$

A morphism $d\colon (S_j;\varphi_j) \rightarrow (S_{j+k};\varphi_{j+k})$ in $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$, is a subsequence $(S_j;\varphi_j) \vdash^k_{\mathcal{C}} (S_{j+k};\varphi_{j+k})$ of the derivation. By soundness, $th(S_{j+k}) \subseteq th(S_j)$ holds. Then, $\Gamma_{j+k} \subseteq th(S_j)$ (i.e., $S_j \models_\Theta \Gamma_{j+k}$) follows. By completeness of the underlying logic, we can replace satisfaction by entailment and have $S_j \vdash_\Theta \Gamma_{j+k}$. According to the definition of theory morphism, this is equivalent to saying that there is a morphism $pd\colon S_{j+k} \rightarrow S_j$ in $\underline{Th}$. Thus, there is a morphism $pd^{op}\colon S_j \rightarrow S_{j+k}$ in $\underline{Th}^{op}$. We define $pres(d) = pd^{op}$ and $pres$ is a functor $pres\colon \underline{Der_{\mathcal{C}}}(S_0;\varphi_0) \rightarrow \underline{Th}^{op}$.

$\Leftarrow$) If $pres$ is a functor $pres\colon \underline{Der_{\mathcal{C}}}(S_0;\varphi_0) \rightarrow \underline{Th}^{op}$, then, for every morphism $d\colon (S_j;\varphi_j) \rightarrow (S_{j+k};\varphi_{j+k})$ in $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$, there is a morphism $pres(d)\colon S_j \rightarrow S_{j+k}$ in $\underline{Th}^{op}$. By definition of opposite category, $pres(d)$ is the opposite pd^{op} of some morphism $pd\colon S_{j+k} \rightarrow S_j$ in $\underline{Th}$. By definition of theory morphism, this implies that $S_j \vdash_\Theta \Gamma_{j+k}$. Then, we have $S_j \models_\Theta \Gamma_{j+k}$ by the soundness of the underlying logic and finally $th(S_{j+k}) \subseteq th(S_j)$. □

We remark that $pres$ is a functor from $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ to $\underline{Th}^{op}$ and not to $\underline{Th}$. This expresses the fact that soundness requires that for all $i \geq 0$, $th(S_{i+1}) \subseteq th(S_i)$, but not vice versa.

In order to characterize relevance also as a functoriality property, we extend the function $\models$ from Th to $States$. The function $\models\colon Th \rightarrow Set$ associates to a theory its satisfaction relation. Given a state $(S;\varphi)$ in a theorem-proving derivation, we are only interested in knowing whether the target φ is a theorem of S. Therefore, we define $\models\colon States \rightarrow Set$ in such a way that $\models((S;\varphi))$, for $S = (\Theta,\Gamma)$, tells exactly whether $\Gamma \models_\Theta \varphi$:

$$\models(((\Theta,\Gamma);\varphi)) = \begin{cases} \{(\Gamma,\varphi)\} & \text{if } \Gamma \models_\Theta \varphi, \\ \emptyset & \text{otherwise.} \end{cases}$$

Then, we can proceed with the theorem:

Theorem 4.2 *A derivation $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ has the relevance property if and only if the restriction of the function $\models\colon States \to Set$ to the subcategory $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ of $\underline{States_I}$ is a functor $\models\colon \underline{Der_{\mathcal{C}}}(S_0;\varphi_0) \to \underline{Set}$ associating to each morphism in $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ an isomorphism in $\underline{Set}$.*

Proof:

$\Rightarrow$) Let $d\colon (S_j;\varphi_j) \to (S_{j+k};\varphi_{j+k})$ be a morphism in $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$. By relevance, $\varphi_j \in th(S_j)$ if and only if $\varphi_{j+k} \in th(S_{j+k})$. Then

- either both $\varphi_j \in th(S_j)$ and $\varphi_{j+k} \in th(S_{j+k})$ are false, and
  $\models((S_j;\varphi_j)) \ = \models((S_{j+k};\varphi_{j+k})) = \emptyset$,
- or both $\varphi_j \in th(S_j)$ and $\varphi_{j+k} \in th(S_{j+k})$ are true, and
  $\models((S_j;\varphi_j)) \ = \ \{(\Gamma_j,\varphi_j)\}$ and $\models((S_{j+k};\varphi_{j+k})) \ = \ \{(\Gamma_{j+k},\varphi_{j+k})\}$.

In the first case, the two sets are equal and, therefore, trivially isomorphic. In the second case, the two sets are singletons, and the isomorphism $\models(d)$ simply associates their two elements.

$\Leftarrow$) By the stated functoriality property, if there is a morphism $d\colon (S_j;\varphi_j) \to (S_{j+k};\varphi_{j+k})$ in $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$, then the two sets $\models((S_j;\varphi_j))$ and $\models((S_{j+k};\varphi_{j+k}))$ are isomorphic. Therefore, it cannot be that one is empty, and the other one is not. By definition of $\models$, it follows that $\varphi_j \in th(S_j)$ if and only if $\varphi_{j+k} \in th(S_{j+k})$. □

4.2 Strategies as generators of decision procedures

We close this section by recalling that one of the most important families of theorem-proving methods, the Knuth-Bendix type completion procedures, is often used for generation of *decision procedures* [2]. In this kind of application, no target is given, and a derivation has the form

$$(S_0;\emptyset)\vdash_{\mathcal{C}}(S_1;\emptyset)\vdash_{\mathcal{C}}\ldots\vdash_{\mathcal{C}}(S_i;\emptyset)\vdash_{\mathcal{C}}\ldots.$$

The derivation proceeds by transforming the presentation until it obtains a *saturated* presentation S, with the property that $\varphi \in th(S)$ is decidable by applying $\mathcal{C}$ itself to $(S;\varphi)$. A presentation with this property is a *decision procedure.* If the purpose of a derivation is to generate a decision procedure, then the monotonicity property is required instead of relevance, and soundness and monotonicity together imply that $th(S_{i+1}) = th(S_i)$ for all stages i, $i \geq 0$, of a derivation. Monotonicity is necessary to ensure that if the derivation halts, the generated decision procedure is a decision procedure for the input theory. Theorem 4.1 showed that soundness implies the existence of theory morphisms. As a corollary, the addition of monotonicity implies the existence of theory isomorphisms:

Corollary 4.1 *A derivation $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ is sound and monotonic if and only if the restriction of the projection function $pres\colon States \to Th$ to the subcategory $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ of $\underline{States_I}$ is a functor* *pres*: $\underline{Der}_{\mathcal{C}}(S_0;\varphi_0) \to \underline{Th}$ *that associates to every morphism in* $\underline{Der}_{\mathcal{C}}(S_0;\varphi_0)$ *an isomorphism of theories.*

Proof: it follows from Theorem 4.1 and the definitions of soundness and monotonicity. □

We recall that, for most theories, the saturated presentation is infinite so that the derivation that tries to generate it does not halt. A more detailed treatment of these topics within our framework can be found in [1] and a general survey in [8].

5 Proof reducing inference systems

In this section, we extend our categorical treatment to the characterization of a derivation as a *process of target-oriented proof reduction*, which is the core of the approach to completion-based theorem proving of [1][6]. The concept of proof reduction rests on the principle that proofs can be compared and ordered with respect to a *proof ordering*, defined as a *well-founded, partial ordering on proofs*. Proof orderings were introduced first in [16, 17], where they were used to prove the correctness of Knuth-Bendix completion procedures. Proof orderings are usually based on well-founded orderings on terms, atoms and clauses:

Example 5.1 *Let $\succ$ be a complete simplification ordering on terms*[7]*. Equational proofs can be represented as chains [16]:*

$$s_1 \underset{l_1\simeq r_1}{\leftrightarrow} s_2 \underset{l_2\simeq r_2}{\leftrightarrow} \dots \underset{l_{n-1}\simeq r_{n-1}}{\leftrightarrow} s_n,$$

where $s_1 \leftrightarrow_{l_1\simeq r_1} s_2$ means that the equality of s_1 and s_2 is established by the equation $l_1 \simeq r_1$ because s_1 and s_2 are $c[l_1\sigma]$ and $c[r_1\sigma]$ for some context c and substitution σ. A proof step can be written in the form $s \to_{l\simeq r} t$ if $s \succ t$. A proof ordering $>_p$ can be defined by associating a measure $M(P)$ to all proofs P, and by establishing that $P >_p P'$ if and only if $M(P) >_u M(P')$ for some ordering $>_u$ on measures, with the necessary properties. For example, for ground proofs, let the measure of an equational step $s \leftrightarrow_{l\simeq r} t$ be the triple (s,l,t), if $s \succ t$. These triples can be compared by the lexicographic combination $>^e$ of the complete simplification ordering $\succ$, the strict encompassment ordering[8] $\blacktriangleright$ *and again the ordering $\succ$. For a proof $P = s \leftrightarrow^*_E t$, let $m(P)$ be the multiset of the triples of all the steps in P, and let the measure $M(P)$ be the pair $(\{s,t\}, m(P))$. Then, the ordering $>_u$ on the measures is the lexicographic combination of the multiset extension $\succ_{mul}$ of the ordering $\succ$ and the multiset extension ${>^e}_{mul}$ of $>^e$.*

5.1 Proof reduction in goal inference rules

Let $(Proofs, >_p, \varepsilon)$ denote the partially ordered set where $Proofs$ is the set of all the proofs in the theories of $\underline{Th}$, $>_p$ is a proof ordering, and ε is the *empty proof*, the bottom element of the

[6]All definitions in this section are (categorical paraphrases of) definitions which appeared in [1].

[7]A *simplification ordering* [18] is monotonic, stable and has the subterm property; hence, it is well-founded; a *complete simplification ordering* is also total on the set of ground terms [4].

[8]The *encompassment ordering* is the composition of subterm and subsumption ordering: $t \trianglerighteq s$ if $t = c[s\sigma]$; $t \blacktriangleright s$ if $t \trianglerighteq s$ and s and t are not variants.

ordered set. For example, if $\underline{Th}$ is the category of equational theories, $Proofs$ will be the set of equational proofs ordered by a proof ordering such as the one in Example 5.1. An ordered set is a category with the morphisms induced by the ordering relation: $\underline{Proofs}$ is the category whose set of objects is $Proofs$ and such that there is a morphism $h\colon P \to Q$, for $P, Q \in \underline{Proofs}$, if and only if $P <_p Q$. We interpret ε as the proof of the dummy target *true*, which denotes success in a derivation (a successful state has the form $(S; true)$). In more concrete terms, the dummy target represents a theorem whose validity has been proved. For instance, in equational logic, a theorem $s \simeq s$ is trivially true. However, it is not trivially true for a theorem prover since a procedure needs to check that the two sides of the equation are identical before stating that the theorem is true. Correspondingly, at the proof level, the proof of $s \simeq s$ is $s \leftrightarrow s$, which is not empty. Therefore, the target *true* is necessary to represent the state after the target has been recognized as being a tautology. A strategy for equational logic will contain an inference rule like *Reflexive Deletion*:

$$\frac{(E; s \simeq s)}{(E; true)}$$

which represents the process of checking that the two sides of the target are identical. This rule reduces $s \simeq s$ to *true* and the proof $s \leftrightarrow s$ to ε.

The key concept of our proof-reduction-based approach is the observation that proving φ from S is a process of reducing φ to *true* and a proof of φ in S to ε. Since the ultimate goal is to obtain the bottom element ε, we would like to focus at every stage $(S_i; \varphi_i)$ of a derivation on the smallest proof of φ_i in S_i. However, the ordering $>_p$ is partial, and in general there is no unique smallest proof but rather multiple *minimal* proofs, any two of which are not comparable in the ordering. Therefore, we need to work with sets of minimal proofs:

Definition 5.1 Given a partially ordered set of proofs $(Proofs, >_p, \varepsilon)$, the function $\Pi\colon States \to \mathcal{P}(Proofs)$ associates to every $(S; \varphi)$ in *States* the set $\Pi(S; \varphi)$ of the *minimal proofs* of φ in S, with respect to the ordering $>_p$.

If φ is not a theorem of S, $\Pi(S; \varphi)$ will be empty. Also, for all S, it is $\Pi(S; true) = \{\varepsilon\}$. Then, at every stage $(S_i; \varphi_i)$ of a derivation, the goal of proving φ_i from S_i can be restated as the goal of reducing some proof in $\Pi(S_i; \varphi_i)$. In order to achieve this goal, the inferences in a derivation need to reduce, or at least preserve, minimal proofs:

Definition 5.2 An inference step $(S; \varphi) \vdash_{\mathcal{C}} (S'; \varphi')$ is *proof-reducing on* φ if for all $P \in \Pi(S, \varphi)$, either $P \in \Pi(S', \varphi')$ or there exists a $Q \in \Pi(S', \varphi')$ such that $P >_p Q$. If the latter holds for some $P \in \Pi(S, \varphi)$, then the step is *strictly proof-reducing.*
A target inference step $(S; \varphi) \vdash_{\mathcal{C}} (S; \varphi')$ is *(strictly) proof-reducing* if it is (strictly) proof-reducing on φ.

In other words, a proof which is minimal at a certain stage of the derivation cannot be replaced by a greater proof. The first part of the definition gives the condition for a generic inference step (either on the presentation or on the target) to be proof-reducing on the target of the step.

The second part says that for a target inference step to be proof-reducing *tout court*, it must be proof-reducing on its target.

5.2 Proof reduction in presentation inference rules

A presentation inference step, on the other hand, may not immediately decrease any proof of the target of the derivation but may still be necessary to decrease it eventually. Thus, a presentation inference step may be proof-reducing simply by being proof-reducing on a theorem other than its target. In order to express this concept, we modify our condition from the given target to a larger set of theorems, whose proofs may be reduced by the inferences of the given strategy $\mathcal{C}$. We single out this set by using a subfunctor $dom_{\mathcal{C}}\colon \underline{Sign} \to \underline{Set}$ of the functor sen: for all signatures Θ, $dom_{\mathcal{C}}(\Theta) \subseteq sen(\Theta)$. If Θ is the signature of a derivation by $\mathcal{C}$, we say that $dom_{\mathcal{C}}(\Theta)$ is the *domain* of the derivation.

The set $dom_{\mathcal{C}}(\Theta)$ depends on the strategy $\mathcal{C}$. For instance, if $\mathcal{C}$ is the Knuth-Bendix completion procedure [2], it was shown in [17] that $dom_{\mathcal{C}}(\Theta) = sen(\Theta)$ because a derivation by Knuth-Bendix completion can reduce any equational proof $s \leftrightarrow^* t$ in the given rewrite system to a minimal proof $s \rightarrow^* \circ \leftarrow^* t$ (a rewrite proof) in an equivalent, confluent rewrite system. This holds under the hypothesis that the derivation is *uniformly fair* [16, 17, 1] and does not fail by generating an equation that cannot be oriented into a rewrite rule. If $\mathcal{C}$ is the Unfailing Knuth-Bendix completion procedure, it was shown in [4, 5] that $dom_{\mathcal{C}}(\Theta)$ is the subset of ground elements in $sen(\Theta)$ because a uniformly fair derivation by Unfailing Knuth-Bendix can reduce any ground equational proof $s \leftrightarrow^* t$ in the given equational presentation to a ground rewrite proof $s \rightarrow^* \circ \leftarrow^* t$ in a ground-confluent equational presentation for the same theory. Considering only ground proofs is sufficient in this case because the universally quantified variables of a theorem $\forall \bar{x} s \simeq t$ can be considered as ground. In general, from a theorem-proving point of view, the smaller the domain is, the better it is, because a smaller domain means that the strategy is focussing on the target theorem.

We can now define proof reduction for presentation inference steps. A presentation step that reduces a proof of its target will be proof-reducing, regardless of its effects on the other theorems in the domain. Otherwise, a presentation step is proof-reducing if it does not increase any proof of a theorem in the domain and strictly decreases at least one:

Definition 5.3 A presentation inference step $(S; \varphi) \vdash_{\mathcal{C}} (S'; \varphi)$, where $sign(S) = sign(S') = \Theta$, is *proof-reducing* if

1. either it is strictly proof-reducing on φ,
2. or
   (a) $\Pi(S, \varphi) = \Pi(S', \varphi)$,
   (b) $\forall \psi \in dom_{\mathcal{C}}(\Theta)$, $(S; \psi) \vdash (S', \psi)$ is proof-reducing on ψ, and
   (c) $\exists \psi \in dom_{\mathcal{C}}(\Theta)$ such that $(S; \psi) \vdash (S', \psi)$ is strictly proof-reducing on ψ.

A derivation is proof-reducing if all its steps are proof-reducing. In the rest of this section, we show that the above proof-reduction properties are equivalent to *functoriality properties* of the function Π. In order to characterize Π as a functor, we use the following definition:

Definition 5.4 Given a partially ordered set $(U, <)$ and two subsets $A, B \subseteq U$, a function $f\colon A \rightarrow B$ is *reducing* if for all x in A,

$$f(x) = \begin{cases} x & \text{if } x \in B \text{, i.e. } x \in A \cap B, \\ y & \text{for some } y < x \text{, otherwise.} \end{cases}$$

A reducing function maps each element into itself (identity) or into a smaller element.

Definition 5.5 Let $\underline{S}$ be a category, $(U, <)$ a partially ordered set, and $\underline{\mathcal{P}(U)}$ the category (subcategory of $\underline{Set}$) whose set of objects is $\mathcal{P}(U)$. A functor $F\colon \underline{S} \rightarrow \underline{\mathcal{P}(U)}$ has *reducing images* if for all morphisms h in $\underline{S}$, $F(h)$ is a reducing function.

We can then show that Π is a functor that associates to proof-reducing steps in a derivation reducing functions on the corresponding sets of minimal proofs. First, we denote by $\underline{ProofsSet}$ the subcategory of $\underline{Set}$, whose set of objects is $\mathcal{P}(Proofs)$. (We define $\underline{ProofsSet}$ to be a subcategory of $\underline{Set}$, not a subcategory of the category of partially ordered sets, because elements in $\underline{ProofsSet}$ may not be ordered. In particular, sets of minimal un-orderable proofs, such as the images of Π, are not ordered.) Then, we start by proving this result for derivations made of target inference steps only (e.g., a derivation made only of simplification steps on the target):

Theorem 5.1 *A derivation $\underline{Der_{\mathcal{C}}}(S_0; \varphi_0)$ made only of target inference steps,*

$(S_0; \varphi_0) \vdash_{\mathcal{C}} (S_0; \varphi_1) \vdash_{\mathcal{C}} \ldots \vdash_{\mathcal{C}} (S_0; \varphi_i) \vdash_{\mathcal{C}} \ldots,$

is proof-reducing if and only if the restriction of the function $\Pi\colon States \rightarrow \mathcal{P}(Proofs)$ to the subcategory $\underline{Der_{\mathcal{C}}}(S_0; \varphi_0)$ of $\underline{States_I}$ is a functor $\Pi\colon \underline{Der_{\mathcal{C}}}(S_0; \varphi_0) \rightarrow \underline{ProofsSet}$ that has reducing images.

Proof:

$\Rightarrow$) We assume that $\underline{Der_{\mathcal{C}}}(S_0; \varphi_0)$ is proof-reducing. This implies that all sequences of inferences that are subsequences of $\underline{Der_{\mathcal{C}}}(S_0; \varphi_0)$ are proof-reducing. In order to show that Π is a functor on $\underline{Der_{\mathcal{C}}}(S_0; \varphi_0)$, we need to define $\Pi(d)$ for all morphisms $d\colon (S_0; \varphi_j) \rightarrow (S_0; \varphi_{j+k})$ $(j \geq 0,\ k > 0)$ in $\underline{Der_{\mathcal{C}}}(S_0; \varphi_0)$. Let $\Pi(d)$ be the function $\Pi(d)\colon \Pi(S_0; \varphi_j) \rightarrow \Pi(S_0; \varphi_{j+k})$ defined as follows: for all proofs P in $\Pi(S_0; \varphi_j)$,

$$\Pi(d)(P) = \begin{cases} P & \text{if } P \in \Pi(S_0; \varphi_{j+k}), \\ Q & \text{where } Q <_p P \text{, otherwise.} \end{cases}$$

The function $\Pi(d)$ is well-defined because the hypothesis that all steps in $\underline{Der_{\mathcal{C}}}(S_0; \varphi_0)$ are proof-reducing target inference steps implies that either $P \in \Pi(S_0; \varphi_{j+k})$, or there exists in $\Pi(S_0; \varphi_{j+k})$ a proof Q such that $Q <_p P$. Thus, $\Pi\colon \underline{Der_{\mathcal{C}}}(S_0; \varphi_0) \rightarrow \underline{ProofsSet}$ is a functor. Furthermore, it follows from Definition 5.4 and the definition of $\Pi(d)$ that Π is a functor with reducing images:

$$\begin{tikzcd}(S_0;\varphi_j) \arrow[r, "d"] \arrow[d, "\Pi"'] & (S_0;\varphi_{j+k}) \arrow[d, "\Pi"] \\ \Pi(S_0;\varphi_j) \arrow[r, "\Pi(d)"'] & \Pi(S_0;\varphi_{j+k})\end{tikzcd}$$

$\Leftarrow$) In order to prove that $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ is proof-reducing, we only need to consider one-step morphisms $d\colon (S_0;\varphi_j) \to (S_0;\varphi_{j+1})$. By hypothesis, $\Pi\colon \underline{Der_{\mathcal{C}}}(S_0;\varphi_0) \to \underline{ProofsSet}$ is a functor with reducing images. This means that for all $j \geq 0$, the morphism $d\colon (S_0;\varphi_j) \to (S_0;\varphi_{j+1})$ in $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ has an image function $\Pi(d)\colon \Pi(S_0;\varphi_j) \to \Pi(S_0;\varphi_{j+1})$ that is reducing. By Definition 5.4 applied to $\Pi(d)$, it follows that for all $P \in \Pi(S_0;\varphi_j)$, either $P \in \Pi(S_0;\varphi_{j+1})$ or there exists a $Q \in \Pi(S_0;\varphi_{j+1})$ such that $P >_p Q$. That is, for all $j \geq 0$, the step $(S_0;\varphi_j)\vdash_{\mathcal{C}}(S_0;\varphi_{j+1})$ is a proof-reducing target inference step according to Definition 5.2. Then, $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ is proof-reducing. □

The proof-reducing effect of presentation inference steps is considered on the domain of the derivation, not only on the target. Therefore, in order to extend Theorem 5.1 to derivations containing both target steps and presentation steps, we introduce *the subcategory induced by a presentation step*:

Definition 5.6 Let $(S;\varphi)\vdash_{\mathcal{C}}(S';\varphi)$ be a presentation inference step, where $sign(S) = sign(S') = \Theta$, and let $d\colon (S;\varphi) \to (S';\varphi)$ be the corresponding morphism in $\underline{States_I}$. The *subcategory induced by d*, denoted by $\underline{Sides_d}$, is the subcategory of $\underline{States_I}$, whose set of objects is

$\{(S;\psi) \mid \psi \in dom_{\mathcal{C}}(\Theta)\} \bigcup \{(S';\psi) \mid \psi \in dom_{\mathcal{C}}(\Theta)\}$

and whose morphisms are all the arrows:

$d_\psi\colon (S;\psi) \to (S';\psi)$ for $\psi \in dom_{\mathcal{C}}(\Theta)$.

In other words, $\underline{Sides_d}$ contains all the inference steps that are identical to d except for the choice of a different target in the domain. The purpose of $\underline{Sides_d}$ is to capture the effect of a presentation inference step d on the other theorems in the domain.

Theorem 5.2 *A derivation $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ is proof-reducing if and only if the function $\Pi\colon States \to \mathcal{P}(Proofs)$ has the following properties:*

1. *the restriction of Π to the subcategory $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ of $\underline{States_I}$ is a functor $\Pi\colon \underline{Der_{\mathcal{C}}}(S_0;\varphi_0) \to \underline{ProofsSet}$ with reducing images, and*

2. *for all one-step morphisms in $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$, $d\colon (S_i;\varphi_i) \to (S_{i+1};\varphi_{i+1})$, such that $\varphi_i = \varphi_{i+1}$, if $\Pi(d)$ is the identity function, then*

   (a) *the restriction of Π to the subcategory $\underline{Sides_d}$ induced by d is also a functor $\Pi\colon \underline{Sides_d} \to \underline{ProofsSet}$ with reducing images, and*

   (b) *there exists at least a morphism d_ψ in $\underline{Sides_d}$, for $\psi \in dom(\Theta)_{\mathcal{C}}$, such that $\Pi(d_\psi)$ is not the identity function.*

Proof:

$\Rightarrow$) We assume that $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ is proof-reducing. The proof of Part 1 of the thesis is basically the same as that in Theorem 5.1. For all morphisms $d\colon (S_j;\varphi_j) \to \Pi(S_{j+k};\varphi_{j+k})$ $(j \geq 0,\ k > 0)$ of $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$, let $\Pi(d)$ be the function $\Pi(d)\colon \Pi(S_j;\varphi_j) \to \Pi(S_{j+k};\varphi_{j+k})$ defined as follows: for all proofs P in $\Pi(S_j;\varphi_j)$,

$$\Pi(d)(P) = \begin{cases} P & \text{if } P \in \Pi(S_{j+k};\varphi_{j+k}), \\ Q & \text{where } Q <_p P, \text{ otherwise.} \end{cases}$$

By Definition 5.2 (for target inference steps) and by Conditions 1 and 2a of Definition 5.3 (for presentation inference steps), $\Pi(d)$ is well-defined, so that $\Pi\colon \underline{Der_{\mathcal{C}}}(S_0;\varphi_0) \to \underline{ProofsSet}$ is a functor and has reducing images.

For Part 2 of the thesis, let $d\colon (S_i;\varphi_i) \to (S_{i+1};\varphi_{i+1})$ be a one-step morphism of $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$, such that $\varphi_i = \varphi_{i+1}$ and $\Pi(d)$ is identity. Since $\varphi_i = \varphi_{i+1}$ and $\Pi(d)$ is identity, d corresponds to an inference step on the presentation that does not reduce any proof of the target. Thus, the step associated with d, $(S_i;\varphi_i) \vdash_{\mathcal{C}} (S_{i+1};\varphi_{i+1})$, must be proof-reducing by Condition 2 of Definition 5.3. Condition 2b implies that $\Pi(d_\psi)$ defined as above is well-defined also for all morphisms d_ψ in $\underline{Sides_d}$. Then, Π is a functor $\Pi\colon \underline{Sides_d} \to \underline{ProofsSet}$ with reducing images:

$$\begin{tikzcd}
(S_i;\psi) \arrow[r, "d_\psi"] \arrow[d, "\Pi"'] & (S_{i+1};\psi) \arrow[d, "\Pi"] \\
\Pi(S_i;\psi) \arrow[r, "\Pi(d_\psi)"'] & \Pi(S_{i+1};\psi)
\end{tikzcd}$$

Finally, by Condition 2c of Definition 5.3, there exists a $\psi \in dom_{\mathcal{C}}(\Theta)$ such that $(S_i;\psi) \vdash_{\mathcal{C}} (S_{i+1};\psi)$ is strictly proof-reducing on ψ: it follows that for such ψ, $\Pi(d_\psi)$ is not the identity.

$\Leftarrow$) In order to prove that $\underline{Der_{\mathcal{C}}}(S_0;\varphi_0)$ is proof-reducing, we need to show that all its steps are proof-reducing. For target inference steps, the proof is the same as that in Theorem 5.1. Presentation inference steps correspond to morphisms $d\colon (S_j;\varphi_j) \to (S_{j+1};\varphi_{j+1})$, where $\varphi_{j+1} = \varphi_j$. By Part 1 of the hypothesis, $\Pi\colon \underline{Der_{\mathcal{C}}}(S_0;\varphi_0) \to \underline{ProofsSet}$ is a functor with reducing images. This means that for all $j \geq 0$, the morphism $d\colon (S_j;\varphi_j) \to (S_{j+1};\varphi_j)$ has an image function $\Pi(d)\colon \Pi(S_j;\varphi_j) \to \Pi(S_{j+1};\varphi_j)$ that is reducing. By Definition 5.4 applied to $\Pi(d)$, it follows that for all $P \in \Pi(S_j;\varphi_j)$, either $P \in \Pi(S_{j+1};\varphi_j)$ or there exists a $Q \in \Pi(S_{j+1};\varphi_j)$ such that $P >_p Q$. If the latter condition is true for at least one proof $P \in \Pi(S_j;\varphi_j)$, then the step $(S_j;\varphi_j) \vdash_{\mathcal{C}} (S_{j+1};\varphi_j)$ is strictly proof-reducing on the target and, thus, proof-reducing according to Condition 1 of Definition 5.3. Otherwise, it means that $\Pi(d)$ is the identity function, which implies Condition 2a of Definition 5.3. Then, by Part 2a of the hypothesis, $\Pi\colon \underline{Sides_d} \to \underline{ProofsSet}$ is a functor with reducing images. That is, for all morphisms $d_\psi\colon (S_j;\psi) \to (S_{j+1};\psi)$ in $\underline{Sides_d}$ there exists an image function $\Pi(d_\psi)\colon \Pi(S_j;\psi) \to \Pi(S_{j+1};\psi)$ that is reducing. By Definition 5.4 applied to $\Pi(d_\psi)$, we have that for all $P \in \Pi(S_j;\psi)$, either $P \in \Pi(S_{j+1};\psi)$, or there exists a $Q \in \Pi(S_{j+1};\psi)$ such that $P >_p Q$. This is equivalent to $(S_j;\varphi_j) \vdash_{\mathcal{C}} (S_{j+1};\varphi_j)$ being proof-reducing on ψ. Since by Definition 5.6 there exists a morphism d_ψ in $\underline{Sides_d}$ for all $\psi \in dom_{\mathcal{C}}(\Theta)$, it follows that the step $(S_j;\varphi_j) \vdash_{\mathcal{C}} (S_{j+1};\varphi_j)$ is proof-reducing on all $\psi \in dom_{\mathcal{C}}(\Theta)$, so that Condition 2b of

Definition 5.3 is satisfied. Furthermore, by Part 2b of the hypothesis, $\Pi(d_\psi)$ is not the identity function for at least one morphism d_ψ in $\underline{Sides}_d$. Let $d_{\psi'}$ be one such morphism. Then, there exists a proof $P \in \Pi(S_j;\psi')$ such that $\Pi(d_{\psi'})(P) \neq P$. By Definition 5.4 applied to $\Pi(d_{\psi'})$, $\Pi(d_{\psi'})(P) = Q$ such that $P >_p Q$. Thus, there exists a proof $Q \in \Pi(S_{j+1};\psi')$ such that $P >_p Q$. In other words, the step $(S_j;\varphi_j) \vdash_{\mathcal{C}} (S_{j+1};\varphi_j)$ is strictly proof-reducing on ψ', so that Condition 2c of Definition 5.3 is also fulfilled. All inference steps in $\underline{Der}_{\mathcal{C}}(S_0;\varphi_0)$ are proof-reducing, and therefore $\underline{Der}_{\mathcal{C}}(S_0;\varphi_0)$ is proof-reducing. □

5.3 Redundancy

The notion of proof reduction developed so far applies to inference steps that are either expansion steps or contraction steps that replace some sentences with others. There is another type of contraction steps, such as *tautology deletion* or *subsumption*, that simply delete elements in the database. In order to characterize these steps, we introduced a notion of *redundancy* [1], to say that these contraction inferences delete redundant sentences. Informally, a sentence is *redundant* in a presentation on a specific target if deleting it from the presentation does not affect any minimal proof of the target. If this holds on the entire domain, the sentence is said to be redundant on the domain:

Definition 5.7 A sentence φ is *redundant* in S on ψ if $\Pi(S,\psi) = \Pi(S \cup \{\varphi\},\psi)$; it is redundant in S on $dom_{\mathcal{C}}(\Theta)$, where $\Theta = sign(S)$, if it is redundant on ψ for all $\psi \in dom_{\mathcal{C}}(\Theta)$.

Redundancy can be captured in categorical terms by using the subcategory $\underline{Sides}_d$:

Corollary 5.1 *Let $d\colon (S \cup \{\varphi\};\psi) \to (S;\psi)$, be the morphism corresponding to a contraction step on the presentation, deleting φ, in a proof-reducing derivation $\underline{Der}_{\mathcal{C}}(S_0;\varphi_0)$. Then, φ is redundant in S on ψ if and only if $\Pi(d)$ is the identity function. It is redundant in S on $dom_{\mathcal{C}}(\Theta)$ if and only if the restriction of Π to $\underline{Sides}_d$ is a functor $\Pi\colon \underline{Sides}_d \to \underline{ProofsSet}$ that associates to every morphism in $\underline{Sides}_d$ the identity function.*

Proof: assuming the definition of $\Pi(d)$ given in the proofs of Theorem 5.1 and 5.2, the result follows from Definition 5.6, Definition 5.7, and the proofs of the two theorems. □

Since deletion of redundant sentences makes the database smaller, every type of inference step induces some form of *reduction*: proof-reduction for expansion rules and proof-reduction or reduction of the database for contraction rules. If we call *reducing* an inference step that either is proof-reducing or deletes a redundant sentence, we can define a *completion procedure* as a strategy whose derivations are sound, have the relevance property, and are reducing [1]. This idea of theorem proving as a process of reduction is fundamental in the theory of completion procedures, especially but not exclusively (e.g., [17, 7]) in our approach. We refer to [1] for a full treatment, which includes the characterizations of refutational completeness, fairness, and completeness of a strategy in terms of proof reduction.

6 Discussion

We have applied category theory to formalize some fundamental ideas in theorem proving. Category theory has turned out to be a valuable tool in analyzing concepts and in refining their definitions to a finer level of detail.

In the first part of the paper, we defined formally the components of a theorem-proving strategy and the associated derivations. Theorem-proving strategies are traditionally described by giving a set of non-deterministic inference rules and by indicating the requirements, e.g. some fairness property, that the search plan which schedules the application of the rules should satisfy. The determination of how the search plan actually works is usually left to the implementation of the strategy. While there are standard ways of defining inference rules, there is no standard, formal definition of search plans. It follows that there is a substantial gap between the specification of a strategy as a set of inference rules and the task of implementing it in a theorem prover. Also, we feel that a significant amount of knowledge about search in theorem proving remains hidden in the code of the implementations, partly because of the lack of formal tools needed to discuss search in theorem proving.

We define a strategy $\mathcal{C}$ as a set of inference rules I, the inference system, and a search plan Σ, using "search plan" for what is otherwise referred to as the "control" or the "control strategy." A derivation is the computation produced by a strategy on a given input. A (sequential) computer program is generally expected to be functional and deterministic, in the sense that whenever it is executed with a certain input it gives the same result (functional) and the computation goes through the same stages (deterministic). A theorem prover is no exception to this rule. Therefore, we seek to incorporate these properties in the description of a strategy since a strategy can be regarded as a high-level specification of a theorem prover.

For the inference rules, we propose a characterization of inference rules as *natural transformations*. The functional nature of natural transformations captures the fact that once the premises are given, the result of the application of an inference rule is uniquely determined. Our characterization applies symmetrically to expansion and contraction inference rules, and covers also the case where the applicability conditions of the inference rule fail, as happens in any real execution of a theorem prover. Having defined inference rules, we can define derivability by an inference system I. Derivability induces the structure, i.e. the morphisms, of the category $\underline{States}_I$, that represents the space of all the possible derivations by I.

For the search plan Σ, we present a formal definition in terms of two choice function, the rule-selecting function ζ and the premises-selecting function ξ. With these elements, we decompose the notion of inference step, usually considered an atomic one, into finer steps. First, ζ is applied to choose an inference rule f^n; second, ξ is applied to choose a tuple of premises $\bar{x}$; third, f^n_Θ is applied to $\bar{x}$ to generate two sets $\pi_{21}(f^n_\Theta(\bar{x}))$ and $\pi_{22}(f^n_\Theta(\bar{x}))$; finally $\pi_{21}(f^n_\Theta(\bar{x}))$ is replaced by $\pi_{22}(f^n_\Theta(\bar{x}))$ in the database. Given the search plan Σ, the subcategory $\underline{Der}_{\mathcal{C}}(S_0;\varphi_0)$ of $\underline{States}_I$, representing the derivation generated by the strategy $\mathcal{C} =< I;\Sigma >$ on input $(S_0;\varphi_0)$ can be determined uniquely.

We remark that we distinguish between "derivations" and "proofs". A derivation is the sequence of all the inference steps performed by the procedure whereas a proof is the result of a successful derivation. For instance, given a successful refutational derivation, the proof is made of the empty clause, all its ancestors generated during the derivation, and the inference steps connecting them. Thus, the proof is usually much smaller in terms of steps than is the derivation itself.

In the second part of this paper, we studied the properties of derivations. We showed that *soundness*, *relevance* and *monotonicity* correspond to functors from the derivation to the category of theories. The bulk of the second part was dedicated to the characterization of a derivation as a process of reduction, either *proof reduction* or deletion of *redundant* data. We introduced the categories $\underline{Proofs}$, with the morphisms induced by a proof ordering $>_p$, and $\underline{ProofsSet}$. Proof reduction and elimination of redundancy consist then in the existence of a functor Π from the derivation to $\underline{ProofsSet}$. One difficulty in obtaining this result is in handling those inferences steps that neither reduce a proof of the given target nor delete a redundant sentence, but reduce a proof of some other theorem in the domain of the derivation. For the purpose of treating these inferences, we define the *subcategory* $\underline{Sides}_d$ *induced by a presentation inference step d*. This subcategory allows us to consider the proof-reducing effect of an inference step on theorems other than the given target. In summary, the categories $\underline{States}_I$ and $\underline{Der}_{\mathcal{C}}(S_0; \varphi_0)$ represent the level of the inferences, the categories $\underline{Proofs}$ and $\underline{ProofsSet}$ represent the underlying level of proofs, and suitable functors connect the two levels.

We conclude this discussion with a comparison with related work and directions for future research.

Our framework for completion-based theorem proving and our ideas on proof reduction and redundancy are closely related to the research of other authors. Since a comparison of approaches to completion procedures is beyond the scope of this paper, we refer to [1] for comparisons in this area.

As we mentioned previously, not much work has been done towards a formal treatment of search in theorem proving. An inspirational paper was [19], in which a search plan was defined as a function on the power set of the nodes of a search graph. What we found difficult with this interpretation was capturing the behaviour of contraction, which was not considered in that paper. Our purpose in defining the search plan was to define formally the operations of a theorem prover. Since a theorem prover works on a database of formulae, not on an explicit representation of the underlying search graph, we decided to define the search plan as a pair of selecting functions which choose the inference rule and the premises. In our approach the search graph will then be transformed according to how the search plan guides the derivation. This and other parts of our work on search are currently under development.

More recently, another approach to the study of search in theorem proving appeared in [20]. The goal of this approach is to analyze and compare the efficiency of some well-known theorem-proving strategies in terms of the size of the generated search space. Thus, the emphasis is on defining measures of the size of the search space for given strategies, rather than on the search plans themselves. The strategies are compared assuming specific search plans, e.g. breadth-first search or depth-first search with iterative deepening, and a general definition of search plan is not proposed.

We are indebted to [15] for the notions reported in Section 2 and more importantly for the idea of applying category theory to formalize concepts in theorem proving. The system of definitions presented in [15] is a general framework for categorical logic, and, as such, is not concerned with concrete issues in theorem proving, such as contraction inference rules and search plans, that are of foremost importance to us. We believe that our approach is compatible with the foundations proposed in [15], but since it uses a fairly limited number of notions from [15], it is basically independent and could fit into other general approaches to logic.

Acknowledgements

A large portion of this work was done when both authors were with the Department of Computer Science of the State University of New York at Stony Brook, and was supported in part by grant CCR-8901322, funded by the National Science Foundation, and, for the first author, also by a scholarship of Università degli Studi di Milano, Italy.

The authors also gratefully acknowledge discussions and e-mail exchanges with José Meseguer.

References

[1] Bonacina, M.P. and Hsiang, J., "Towards a foundation of completion procedures as semidecision procedures", *Theoretical Computer Science*, Vol. 146, 1995, pp. 199–242.

[2] Knuth, D.E. and Bendix, P.B., "Simple word problems in universal algebras", in Leech, J. (ed.), *Proc. of the Conference on Computational Problems in Abstract Algebras*, 1967, Pergamon Press, Oxford, 1970, pp. 263–298.

[3] Hsiang, J., "Refutational theorem proving using term rewriting systems", *Artificial Intelligence*, Vol. 25, 1985, pp. 255–300.

[4] Hsiang, J. and Rusinowitch, M., "On word problems in equational theories", in Ottman, Th. (ed.), *Proc. of the Fourteenth International Conference on Automata, Languages and Programming*, 1987, Springer Verlag, Lecture Notes in Computer Science 267, pp. 54–71.

[5] Bachmair, L., Dershowitz, N. and Plaisted, D.A., "Completion without failure", in Aït-Kaci, H. and Nivat, M. (eds.), *Resolution of Equations in Algebraic Structures*, Vol. II: Rewriting Techniques, Academic Press, New York, 1989, pp. 1–30.

[6] Rusinowitch, M., "Theorem proving with resolution and superposition", *Journal of Symbolic Computation*, Vol. 11, No. 1 & 2, 1991, pp. 21–50.

[7] Bachmair, L. and Ganzinger, H., "On restrictions of ordered paramodulation with simplification", in Stickel, M.E. (ed.), *Proc. of the Tenth International Conference on Automated Deduction*, 1990, Springer Verlag, Lecture Notes in Artificial Intelligence 449, pp. 427–441.

[8] Dershowitz, N. and Jouannaud, J.-P., "Rewrite systems", in van Leeuwen, J. (ed.), *Handbook of Theoretical Computer Science*, Vol. B, Elsevier Science Publishers B.V., Amsterdam, 1989, pp. 243–320.

[9] McCune, W.W., "OTTER 3.0 reference manual and guide", Technical Report, ANL-94/6, Argonne National Laboratory, 1994.

[10] Kapur, D. and Zhang, H., "An overview of rewrite rule laboratory (RRL)", in Dershowitz, N. (ed.), *Proc. of the Third International Conference on Rewriting Techniques and Applications*, 1989, Springer Verlag, Lecture Notes in Computer Science 355, pp. 559–563.

[11] Anantharaman, S. and Hsiang, J., "Automated proofs of the Moufang identities in alternative rings", *Journal of Automated Reasoning*, Vol. 6, No. 1, 1990, pp. 76–109.

[12] Robinson, J.A., "A machine oriented logic based on the resolution principle", *Journal of the ACM*, Vol. 12, No. 1, 1965, pp. 23–41.

[13] Chang, C.L. and Lee, R.C., *Symbolic logic and mechanical theorem proving*, Academic Press, New York, 1973.

[14] MacLane, S., *Categories for the working mathematician*, Springer Verlag, New York, 1971.

[15] Meseguer, J., "General logics", in Ebbinghaus, H.-D. et al. (eds.), *Proc. of Logic Colloquium '87*, Elsevier Science Publishers B.V., Amsterdam, 1989.

[16] Bachmair, L., Dershowitz, N. and Hsiang, J., "Orderings for equational proofs", in *Proc. of the First IEEE Symposium on Logic in Computer Science*, 1986, pp. 346–357.

[17] Bachmair, L. and Dershowitz, N., "Equational inference, canonical proofs and proof orderings", *Journal of the ACM*, Vol. 41, No. 2, 1994, pp. 236–276.

[18] Dershowitz, N., "Orderings for term-rewriting systems", *Theoretical Computer Science*, Vol. 17, No. 3, 1982, pp. 279–301.

[19] Kowalski, R., "Search strategies for theorem proving", in Meltzer, B. and Michie, D. (eds.), *Machine Intelligence 5*, Edinburgh University Press, Edinburgh, 1969, pp. 181–201.

[20] Plaisted, D.A., "The search efficiency of theorem proving strategies", in Bundy, A. (ed.), *Proc. of the Twelfth International Conference on Automated Deduction*, 1994, Springer Verlag, Lecture Notes in Artificial Intelligence 814, pp. 57–71.