On fairness of completion-based theorem proving strategies *

Maria Paola Bonacina **Jieh Hsiang**
Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794-4400, USA
{bonacina,hsiang}@sbcs.sunysb.edu

Abstract

Fairness is an important concept emerged in theorem proving recently, in particular in the area of completion-based theorem proving. Fairness is a required property for the *search plan* of the given strategy. Intuitively, fairness of a search plan guarantees the generation of a successful derivation if the *inference* mechanism of the strategy indicates that there is one. Thus, the completeness of the inference rules and the fairness of the search plan form the completeness of a theorem proving strategy. A search plan which exhausts the entire search space is obviously fair, albeit grossly inefficient. Therefore, the problem is to reconciliate fairness and efficiency. This problem becomes even more intricate in the presence of *contraction inference rules* - rules that remove data from the data set.

The known definitions of fairness for completion-based methods are designed to ensure the confluence of the resulting system. Thus, a search plan which is fair according to these definitions may force the prover to perform deductions completely irrelevant to prove the intended theorem. In a theorem proving strategy, on the other hand, one is usually only interested in proving a specific theorem. Therefore the notion of fairness should be defined accordingly. In this paper we present a target-oriented definition of fairness for completion, which takes into account the theorem to be proved and therefore does not require computing all the critical pairs. If the inference rules are complete and the search plan is fair with respect to our definition, then the strategy is complete. Our framework contains also notions of redundancy and contraction. We conclude by comparing our definition of fairness and the related concepts of redundancy and contraction with those in related works.

1 Introduction

A theorem proving strategy is composed of a set of inference rules and a search plan. Refutational completeness of a strategy involves both the inference rules and the search plan. First, it requires that for all unsatisfiable inputs, there exist successful derivations by the inference rules of the strategy. Second, it requires that whenever successful derivations exist, the search plan guarantees that the computed derivation is successful. We call these properties *completeness* of the inference

*Research supported in part by grants CCR-8805734, INT-8715231 and CCR-8901322, funded by the National Science Foundation. The first author has also been supported by Dottorato di ricerca in Informatica, Universitá degli Studi di Milano, Italy.

rules and *fairness* of the search plan, respectively. In this paper we concentrate our study on the fairness property. A search plan which exhaustively performs all possible steps is trivially fair, but tremendously inefficient. Therefore, the problem is to reconciliate fairness and efficiency. This problem is especially striking in case of Knuth-Bendix type completion procedures applied to theorem proving. Fairness for Knuth-Bendix completion appeared first in [10]. A more general definition of fairness in the proof ordering framework is given in [2]. A derivation by a completion procedure is fair according to [2] if all the critical pairs from *persisting* equations are eventually generated and all the generated equations are eventually reduced. This definition of fairness captures exactly the requirements that a derivation generating a canonical system must satisfy, but it is not intended for a derivation proving a specified theorem. Indeed, most theorem proving derivations do not satisfy this definition of fairness, since proving a theorem usually does not require generating all critical pairs.

In this paper we present a notion of fairness for completion procedures applied to theorem proving. This definition of fairness is part of a new approach to completion procedures, which was partially presented in [5]. All the fundamental concepts of the theory of completion are uniformly defined in terms of *proof reduction*. This approach allows us to regard theorem proving as the basic interpretation of a completion process, rather than as a side-effect of the generation of a confluent presentation of the theory. We define completeness of the inference rules and fairness of the search plan and we show that if the inference rules are complete and the search plan is fair according to our definitions, the procedure is complete.

Our approach to the problem of fairness is new in three respects, all relevant to theorem proving. First, our definition is target-oriented, that is it takes the theorem to be proved into account. A target-oriented definition of fairness is important for search plans to be both fair and efficient. For instance, we may have a problem where the target $s \simeq t$ is an equation on a signature F_1 and the input presentation E is the union of a set E_1 of equations on the signature F_1 and a set E_2 of equations on another signature F_2, disjoint from F_1. Such a problem can occur in definitions of abstract data types, where the signature F_1 contains the constructors and a set of defined symbols, whereas the signature F_2 is another set of defined symbols. According to our definition, a derivation where no inference from E_2 is performed is fair. On the other hand, fairness as defined in previous works on completion [2, 13, 4] would require to compute critical pairs from the equations in E_2 as well.

Second, we emphasize the distinction between fairness as a property of the search plan and completeness as a property of the inference rules. Most theorem proving strategies are simply presented by giving a set of inference rules and the task of designing a suitable search plan is left to the implementation phase. This is not satisfactory, since the actual performance of the prover depends heavily on the search plan. Therefore, we think that it is important to study search plans systematically and define their requirements formally. Our approach to the fairness problem is a first step in this direction. Finally, it is generally acknowledged that the application of contraction inference rules is mandatory for efficiency. However, the known definitions of fairness emphasize the application of expansion rules, whereas ours treats both expansion and contraction rules uniformly.

In the first two sections we outline our approach to completion procedures. We refer the reader

to [5] for a more detailed treatment of this framework. The fourth section contains the definition of fairness and the theorem of completeness of strategies with fair search plan. In Section 5, we show how the classical results on Knuth-Bendix completion are covered in our framework, if uniform fairness, i.e. fairness to generate confluent systems, is assumed. Finally, in Sections 6 and 7 we discuss and compare with ours the definitions of fairness and redundancy in [13] and in [4].

2 Proof orderings for theorem proving

A basic assumption of our approach is that all the elements we deal with are partially ordered. We recall that a *simplification ordering* on terms is a monotonic and stable ordering such that a term is greater than any of its subterms. A simplification ordering is well founded [6]. A *complete simplification ordering* is also total on the set of ground terms. A *proof ordering* is a monotonic, stable and well founded ordering on proofs. Given a finite set of sentences S, we denote by $Th(S)$ the *theory* of S, $Th(S) = \{\varphi | S \models \varphi\}$, and we say that S is a *presentation* of the theory $Th(S)$.

We assume to have a complete simplification ordering $\succ$ on terms and literals [1] and a proof ordering $>_p$ on proofs. We denote proofs by capital Greek letters: $\Upsilon(S, \varphi)$ is a proof of φ from axioms in the presentation S. Our notion of proof ordering is different from the one introduced in [1]. According to the notion of proof ordering in [1], only two proofs $\Upsilon(S, \varphi)$ and $\Upsilon'(S', \varphi)$ of the same theorem in different presentations can be compared. We assume that two proofs $\Upsilon(S, \varphi)$ and $\Upsilon'(S', \varphi')$ of different theorems may also be comparable.

The minimum proof is the *empty proof*. We denote by *true* the theorem whose proof is empty and we assume that *true* is the bottom element in our ordering on terms and literals. Given a pair $(S; \varphi)$, we can select a minimal proof among all proofs of φ from S:

Definition 2.1 *Given a proof ordering $>_p$, we denote by* $\Pi(S, \varphi)$ a minimal proof *of φ from S with respect to $>_p$, i.e. a proof such that for all proofs $\Upsilon(S, \varphi)$ of φ from S, $\Upsilon(S, \varphi) \not<_p \Pi(S, \varphi)$.*

A *theorem proving problem* is a pair $(S; \varphi)$, where S is a presentation of a theory and φ the theorem to be proved, which we call the *target*. The process of proving φ from S is the process of reducing φ to *true* and $\Pi(S, \varphi)$ to the empty proof. A *theorem proving derivation* is a sequence of deductions

$$(S_0; \varphi_0) \vdash (S_1; \varphi_1) \vdash \ldots \vdash (S_i; \varphi_i) \vdash \ldots,$$

where at each step $(S_{i+1}; \varphi_{i+1})$ replaces $(S_i; \varphi_i)$ and $\Pi(S_{i+1}, \varphi_{i+1})$ replaces $\Pi(S_i, \varphi_i)$. At stage i the problem is to prove φ_i from S_i or equivalently to reduce $\Pi(S_i, \varphi_i)$. Since $S_i \neq S_{i+1}$ and $\varphi_i \neq \varphi_{i+1}$ in general, we need a proof ordering which may compare two proofs of different theorems, in order to compare $\Pi(S_i, \varphi_i)$ and $\Pi(S_{i+1}, \varphi_{i+1})$. The derivation halts successfully at stage k if φ_k is *true* and $\Pi(S_k, \varphi_k)$ is empty.

[1] A well founded, monotonic and stable ordering is sufficient.

3 Completion procedures

A theorem proving strategy is given by a set of *inference rules* I and a *search plan* Σ. The data are pairs $(S;\varphi)$, where S is the *presentation* and φ is the *target*. We distinguish four classes of inference rules, depending on whether a rule transforms the presentation or the target and whether it is an *expansion* inference rule or a *contraction* inference rule, as they are called in [8]. An expansion inference rule expands a set of sentences by deriving new sentences, whereas a contraction inference rule contracts a set of sentences by either deleting some sentences or replacing them by others. We assume that a target is a clause and therefore can be regarded as a set of literals:

- *Presentation inference rules*:
  - *Expansion inference rules*: f: $\dfrac{(S;\varphi)}{(S';\varphi)}$ where $S \subset S'$.
  - *Contraction inference rules*: f: $\dfrac{(S;\varphi)}{(S';\varphi)}$ where $S \not\subseteq S'$.
- *Target inference rules*:
  - *Expansion inference rules*: f: $\dfrac{(S;\varphi)}{(S;\varphi')}$ where $\varphi \subset \varphi'$.
  - *Contraction inference rules*: f: $\dfrac{(S;\varphi)}{(S;\varphi')}$ where $\varphi \not\subseteq \varphi'$.

We characterize a theorem proving derivation as a process of *proof reduction*. A target inference step modifies the target and therefore it affects the proof of the target. We require that the proof of the target is reduced:

Definition 3.1 *A target inference step* $(S;\varphi) \vdash (S;\varphi')$ *is* proof-reducing *if* $\Pi(S;\varphi) \geq_p \Pi(S;\varphi')$. *It is* strictly proof-reducing *if* $\Pi(S;\varphi) >_p \Pi(S;\varphi')$.

For a presentation inference step we allow more flexibility:

Definition 3.2 *Given two pairs* $(S;\varphi)$ *and* $(S';\varphi')$, *the relation* $(S;\varphi) \rhd_{p,\mathcal{T}} (S';\varphi')$ *holds if*

1. *either* $\Pi(S;\varphi) >_p \Pi(S';\varphi')$
2. *or*
   (a) $\Pi(S;\varphi) = \Pi(S';\varphi')$,
   (b) $\forall \psi \in \mathcal{T}$, $\Pi(S,\psi) \geq_p \Pi(S',\psi)$ *and*
   (c) $\exists \psi \in \mathcal{T}$ *such that* $\Pi(S,\psi) >_p \Pi(S',\psi)$.

Definition 3.3 *A presentation inference step* $(S;\varphi) \vdash (S';\varphi)$ *is* proof-reducing *on* $\mathcal{T}$ *if* $(S;\varphi) \rhd_{p,\mathcal{T}} (S';\varphi)$ *holds. It is* strictly proof-reducing *if* $\Pi(S;\varphi) >_p \Pi(S';\varphi)$.

The condition $(S_i;\varphi_i) \rhd_{p,\mathcal{T}} (S_{i+1};\varphi_{i+1})$ says that a step which reduces the proof of the target is proof-reducing, regardless of its effects on other theorems. However, an inference step on the presentation may not immediately decrease the proof of the target and still be necessary to decrease it eventually. Such a step is also proof-reducing, provided that it does not increase any proof and strictly decreases at least one. This notion of proof reduction applies to presentation inference steps which are either expansion steps or contraction steps which replace some sentences by others. A contraction step which deletes sentences does not modify any minimal proof. In order to characterize these steps, we introduce a notion of *redundancy*:

Definition 3.4 *A sentence φ is* redundant *in S on domain $\mathcal{T}$ if $\forall \psi \in \mathcal{T}$, $\Pi(S,\psi) = \Pi(S \cup \{\varphi\},\psi)$.*

A sentence is redundant in a presentation if adding it to the presentation does not reduce any minimal proof.

Definition 3.5 *An inference step $(S;\varphi) \vdash (S';\varphi')$ is* reducing *on $\mathcal{T}$ if either it is proof-reducing or it deletes a sentence which is redundant in S on domain $\mathcal{T}$.*

Definition 3.6 *An inference rule f is* reducing *if all the inference steps $(S;\varphi)\vdash_f (S';\varphi')$ where f is applied are reducing.*

We have finally all the elements to give the definition of a completion procedure:

Definition 3.7 *A theorem proving strategy $\mathcal{C} =< I;\Sigma >$ is a* completion procedure *on domain $\mathcal{T}$ if for all pairs $(S_0;\varphi_0)$, where S_0 is a presentation of a theory and $\varphi_0 \in \mathcal{T}$, the derivation*

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

has the following properties:

- *monotonicity: $\forall i \geq 0$, $Th(S_{i+1}) \subseteq Th(S_i)$,*
- *relevance: $\forall i \geq 0$, $\varphi_i \in \mathcal{T}$ and $\varphi_{i+1} \in Th(S_{i+1})$ if and only if $\varphi_i \in Th(S_i)$ and*
- *reduction: $\forall i \geq 0$, the step $(S_i;\varphi_i)\vdash_{\mathcal{C}} (S_{i+1};\varphi_{i+1})$ is reducing on $\mathcal{T}$.*

The *domain* $\mathcal{T}$ is the set of sentences where the inference rules of the completion procedure are reducing. For instance, for the Knuth-Bendix completion procedure $\mathcal{T}$ is the set of all equations. For the Unfailing Knuth-Bendix procedure, $\mathcal{T}$ is the set of all ground equations.

The *monotonicity* and *relevance* properties establish the soundness of the presentation and the target inference rules respectively. Monotonicity ensures that a presentation inference step does not create new elements which are not in the theory, while relevance ensures that a target inference step replaces the target by a new target in such a way that proving the latter is equivalent to proving the former.

Reduction is the key property which characterizes completion procedures. If the procedure is given a target, it tries to reduce the proof of the target until it is empty. If no target is given, the 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 goal is to transform the given presentation S_0 into a confluent one by reducing the proofs of all theorems. Since few theories have finite, confluent presentations, the application of completion to theorem proving is far more interesting in practice.

Note that our definition of completion procedures is different from the classical one [2]. Consequently, our idea of fairness is also different. In our view, a completion procedure is given a target and therefore fairness is fairness with respect to the goal of proving the given target.

4 Fairness

Given a theorem proving problem $(S_0;\varphi_0)$ and a set of inference rules I, the application of I to $(S_0;\varphi_0)$ defines a tree, the *I-tree rooted at* $(S_0;\varphi_0)$. The nodes of the tree are labeled by pairs $(S;\varphi)$. The root is labeled by the input pair $(S_0;\varphi_0)$. A node $(S;\varphi)$ has a child $(S';\varphi')$ if $(S';\varphi')$ can be derived from $(S;\varphi)$ in one step by an inference rule in I. The I-tree rooted at $(S_0;\varphi_0)$ represents all the possible derivations by the inference rules in I starting from $(S_0;\varphi_0)$.

Intuitively, a set I of inference rules is *refutationally complete* if whenever $\varphi_0\in Th(S_0)$, the I-tree rooted at $(S_0;\varphi_0)$ contains successful nodes, nodes of the form $(S;true)$. We use the term "refutational completeness" for the inference rules to differentiate it from the completeness of the theorem proving strategy. Furthermore, "refutational" emphasizes that the goal is to prove a specific theorem. The following definition is an equivalent characterization of this concept in terms of proof reduction:

Definition 4.1 *A set $I=I_p\cup I_t$ of inference rules is* refutationally complete *if whenever $\varphi\in Th(S)$ and $\Pi(S,\varphi)$ is not minimal, there exist derivations*

$(S;\varphi)\vdash_I(S_1;\varphi_1)\vdash_I\ldots\vdash_I(S';\varphi')$

such that $\Pi(S,\varphi)>_p\Pi(S',\varphi')$.

This definition says that a set of inference rules is refutationally complete if it can reduce the proof of the target whenever it is not minimal. Since a proof ordering is well founded, it follows that the I-tree rooted at $(S;\varphi)$ contains successful nodes if $\varphi\in Th(S)$. The advantage of giving the definition of completeness in terms of proof reduction is that the problem of proving completeness of I is reduced to the problem of exhibiting a suitable proof ordering [2].

Given a completion procedure $\mathcal{C}=<I;\Sigma>$, the I-tree rooted at $(S_0;\varphi_0)$ represents the entire search space that the procedure can potentially derive from the input $(S_0;\varphi_0)$. The search plan Σ selects a path in the I-tree: the derivation from input $(S_0;\varphi_0)$ controlled by Σ is the path selected by Σ in the I-tree rooted at $(S_0;\varphi_0)$. A pair $(S_i;\varphi_i)$ reached at stage i of the derivation is a visited node in the I-tree. Each visited node $(S_i;\varphi_i)$ may have many children, but the search plan selects only one of them to be $(S_{i+1};\varphi_{i+1})$. A search plan Σ is *fair* if whenever the I-tree rooted at $(S_0;\varphi_0)$ contains successful nodes, the derivation controlled by Σ starting at $(S_0;\varphi_0)$ is guaranteed to reach a successful node. Similar to completeness, we rephrase this concept in terms of proof reduction:

Definition 4.2 *A derivation*

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

controlled by a search plan Σ *is* fair *if and only if for all* $i \geq 0$*, if there exists a path*

$$(S_i;\varphi_i)\vdash_I\ldots\vdash_I(S';\varphi')$$

in the I*-tree rooted at* $(S_0;\varphi_0)$ *such that* $\Pi(S_i;\varphi_i) >_p \Pi(S';\varphi')$*, then there exists an* $(S_j;\varphi_j)$ *for some* $j > i$*, such that* $\Pi(S';\varphi') \geq_p \Pi(S_j;\varphi_j)$*. A search plan* Σ *is* fair *if all the derivations controlled by* Σ *are fair.*

In other words, if the inference rules can reduce the proof of the target at $(S_i;\varphi_i)$, a fair search plan guarantees that the proof of the target will be indeed reduced at a later stage $(S_j;\varphi_j)$. This definition is target-oriented because it only requires that the proof of the intended target is reduced. If the inference rules are refutationally complete and the search plan is fair, a completion procedure on domain $\mathcal{T}$ is complete, i.e. it is a *semidecision procedure* for $Th(S)\cap\mathcal{T}$ for all presentations S:

Theorem 4.1 *If a completion procedure* $\mathcal{C}$ *on domain* $\mathcal{T}$ *has refutationally complete inference rules and fair search plan, then for all derivations*

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

where $\varphi_0 \in Th(S_0)$*,* $\forall i \geq 0$*, if* $\Pi(S_i,\varphi_i)$ *is not minimal, then there exists an* (S_j,φ_j)*, for some* $j > i$*, such that* $\Pi(S_i,\varphi_i) >_p \Pi(S_j,\varphi_j)$*.*

Proof: if $\Pi(S_i,\varphi_i)$ is not minimal, then by completeness of the inference rules, there exists a path $(S_i;\varphi_i)\vdash_I\ldots\vdash_I(S';\varphi')$ such that $\Pi(S_i;\varphi_i) >_p \Pi(S';\varphi')$. By fairness of the search plan, there exists an $(S_j;\varphi_j)$, for some $j > i$, such that $\Pi(S_i;\varphi_i) >_p \Pi(S';\varphi') \geq_p \Pi(S_j;\varphi_j)$. □

Corollary 4.1 *If a completion procedure* $\mathcal{C}$ *on domain* $\mathcal{T}$ *has refutationally complete inference rules and fair search plan, then for all inputs* $(S_0;\varphi_0)$*, if* $\varphi_0 \in Th(S_0)$*, the derivation*

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

reaches a stage k*,* $k \geq 0$*, such that* φ_k *is the clause* $true$*.*

Proof: if $\varphi_0 \in Th(S_0)$, then by Theorem 4.1 and the well foundedness of $>_p$ the derivation reaches a stage k such that the proof $\Pi(S_k,\varphi_k)$ is minimal. Since the minimal proof is the empty proof, φ_k is the clause $true$. □

For instance, the Unfailing Knuth-Bendix procedure was proved in [9, 3] to be a semidecision procedure for equational theories.

5 Uniform fairness and saturated sets

Definition 4.2 of fairness is sufficient for theorem proving as shown by Theorem 4.1. If Definition 4.2 is applied to Knuth-Bendix completion, it is not sufficient to guarantee that a confluent

rewrite system is eventually generated, since it does not guarantee that all critical pairs are eventually considered. This requires a much stronger fairness property, which we call *uniform fairness*.

The first definition of uniform fairness appeared in [10], where it is required that the search plan sorts the rewrite rules in the data set by a well founded ordering, in order to ensure that no rule is indefinitely postponed. We recall this very first notion of fairness, because it states explicitly that fairness is a property of the search plan. The later definitions of fairness are given at a much higher abstraction level, which may prevent the reader from seeing that fairness is a property of the search mechanism.

Given a derivation by completion starting from a presentation S_0, the *limit* S_∞ of the derivation is the possibly infinite set $\bigcup_{j\geq 0}\bigcap_{i\geq j} S_i$ of all the *persistent* sentences, that is the sentences which are generated at some stage and never deleted afterwards [10, 2]. The advantage of dealing with the limit S_∞ is that if the derivation halts at some stage k, $S_\infty = S_k$. Therefore, properties stated in terms of the limit S_∞ apply uniformly to both halting and non halting derivations. We denote by $I_e(S)$ the set of clauses which can be generated in one step by the presentation expansion rules of the completion procedure applied to S.

Definition 5.1 (Rusinowitch 1988) [13], (Bachmair and Ganzinger 1990) [4] *A derivation*

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

by a completion procedure $\mathcal{C}$ on domain $\mathcal{T}$ is uniformly fair *on domain $\mathcal{T}$ if $\forall\varphi\in I_e(S_\infty)$ there exists an S_j such that either $\varphi\in S_j$ or φ is redundant in S_j on domain $\mathcal{T}$.*

This definition of fairness generalizes previous definitions given in [10] and [2]. It says that every clause φ that can be persistently generated by an expansion rule during the completion process is either actually generated at some step, or replaced by other clauses which yield a smaller proof of φ and make it redundant. For instance, a Knuth-Bendix derivation such that all critical pairs from persisting equations are eventually generated or subsumed or reduced to a common term is uniformly fair according to this definition.

Uniform fairness has been studied and progressively refined in [2, 13, 4] with the purpose of solving the problem of the interaction between expansion inference rules and contraction inference rules. The intuitive meaning of uniform fairness is to be fair to the inference rules, that is to apply all the inference rules to all the data. However, this is impossible, because the inference rules include both contraction rules and expansion rules: if a clause φ is deleted by a contraction step before an expansion rule f is applied to φ, the derivation is not fair to f. The problem has been then to define fairness in such a way that the application of contraction rules is fair. This problem is solved in the definition of uniform fairness by establishing that it is fair not to perform an expansion inference step if its premises are not persistent and it is fair to replace a clause φ by clauses which make it redundant. In this way it is fair to apply contraction inference rules such as simplification and subsumption. In actuality, a uniformly fair procedure will perform exhaustively all expansion steps which are not inhibited by contraction steps.

Fairness and uniform fairness are different in several basic aspects. Fairness is *target-oriented*, whereas uniform fairness is defined for a derivation without a target. Indeed Definition 5.1 is not

a definition of fairness for theorem proving. In [13, 4], Definition 5.1 is applied to refutational theorem proving, where S_0 contains the negation of the target. In this case the only persisting clause is the empty clause $\Box$ and $I_e(\bigcup_{i\geq 0}\bigcap_{j\geq i} S_j) = \{\Box\}$. Then Definition 5.1 says that the limit of the derivation is the empty clause. A notion of fairness given in terms of the limit does not represent useful information for the design of search plans because it does not say anything about how a search plan should choose the successor at any given stage of the derivation.

Our definition of fairness does not differentiate between expansion rules and contraction rules and between persisting and non-persisting clauses, because the interaction of expansion and contraction rules is no longer the issue. All inference rules are treated uniformly by considering their effect with respect to the goal of reducing the proof of the target. On the other hand, expansion rules and contraction rules are treated differently in the definition of uniform fairness. Uniform fairness emphasizes fairness with respect to the expansion inference rules, while the role of contraction inference rules is buried in the restriction to persistent clauses. The reason is that it is necessary to consider all critical pairs in order to obtain a confluent set, but it is not necessary to reduce them, although in practice completion without simplification is hopelessly inefficient.

The following example illustrates a set of conditions for an Unfailing Knuth-Bendix derivation which have been proved in [2] to be sufficient for uniform fairness. These conditions represent the most well known definition of (uniform) fairness for a completion procedure:

Example 5.1 *A derivation*

$(E_0;\emptyset)\vdash_{UKB}(E_1;\emptyset)\vdash_{UKB}\ldots\vdash_{UKB}(E_i;\emptyset)\vdash_{UKB}\ldots$

is uniformly fair if

- *for all critical pairs* $g \simeq d \in I_e(E_\infty)$, $g \simeq d \in \bigcup_{i\geq 0} E_i$ *and*
- E_∞ *is reduced.*

The first condition says that all critical pairs derivable from persisting equations are eventually generated. The second condition says that all persisting equations are eventually simplified as much as possible. As was remarked above, the application of contraction rules is allowed but not required: the first condition alone is sufficient for uniform fairness. Since at any stage of the computation it is not known which equations are going to persist and which equations are going to be simplified, the above conditions for uniform fairness prescribe in practice to apply exhaustively all the inference rules of Unfailing Knuth-Bendix completion until none applies.

The concept of uniform fairness leads to the following notion of *saturated* presentation:

Definition 5.2 (Kounalis and Rusinowitch 1988) [12], (Bachmair and Ganzinger 1990) [4] *A presentation* S *is* saturated *on the domain* $\mathcal{T}$ *of a completion procedure if and only if* $\forall\psi \in I_e(S)$, *either* $\psi \in S$ *or* ψ *is redundant in* S *on* $\mathcal{T}$.

In other words, a presentation is saturated if no non-trivial consequences can be added. In the equational case, as remarked in [12], a set of equations is saturated if no divergent critical pairs can be deduced, or equivalently, the set is *locally confluent*. As in the definition of uniform fairness,

the application of contraction inference rules is allowed but not required: contraction inference rules may still be applicable to a saturated set. A locally confluent equational presentation is not necessarily reduced.

If a derivation is uniformly fair, S_∞ is saturated. Since uniform fairness is defined in terms of redundancy and our notion of redundancy is more general than those in [13] and [4], we give a new proof of this result:

Theorem 5.1 (Kounalis and Rusinowitch 1988) [12], (Bachmair and Ganzinger 1990) [4] *If a derivation*

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

is uniformly fair on domain $\mathcal{T}$*, then* S_∞ *is* saturated *on domain* $\mathcal{T}$*.*

Proof: we show that for all $\varphi \in I_e(S_\infty)$, either $\varphi \in S_\infty$ or φ is redundant in S_∞ on $\mathcal{T}$. By uniform fairness of the derivation, there exists an S_j, for some $j \geq 0$, such that either $\varphi \in S_j$ or φ is redundant in S_j on $\mathcal{T}$:

1. if $\varphi \in S_j$, then either φ is not deleted afterwards, that is $\varphi \in S_\infty$, or φ is deleted at some stage $i > j$. There are in turn two cases: either φ is simply deleted or it is replaced by another sentence φ':

   (a) let S_i be $S \cup \{\varphi\}$ and S_{i+1} be S. By Definition 3.7 of completion, such a step deletes a redundant sentence. Then φ is redundant in S_i on $\mathcal{T}$. Since by Definition 3.7 of completion, $\forall\psi \in \mathcal{T}$, $\Pi(S_i,\psi) \geq_p \Pi(S_\infty,\psi)$, φ is also redundant in S_∞ on $\mathcal{T}$.

   (b) if $S_i = S \cup \{\varphi\}$ and $S_{i+1} = S \cup \{\varphi'\}$, then this step is proof-reducing, that is $\forall\psi \in \mathcal{T}$, $\Pi(S_i,\psi) \geq_p \Pi(S_{i+1},\psi)$ and $\exists\psi \in \mathcal{T}$ such that $\Pi(S_i,\psi) >_p \Pi(S_{i+1},\psi)$. If φ itself represents a minimal proof of φ in S_i, then there exists a minimal proof $\Pi(S_{i+1},\varphi)$ in S_{i+1} such that $\varphi \geq_p \Pi(S_{i+1},\varphi)$. Since $\varphi \notin S_{i+1}$, $\Pi(S_{i+1},\varphi)$ is not φ itself and therefore $\varphi >_p \Pi(S_{i+1},\varphi) \geq_p \Pi(S_\infty,\varphi)$. If φ does not represent a minimal proof of φ in S_i, then there exists a minimal proof $\Pi(S_i,\varphi)$ of φ in S_i such that $\Pi(S_i,\varphi) <_p \varphi$ and therefore $\varphi >_p \Pi(S_i,\varphi) \geq_p \Pi(S_{i+1},\varphi) \geq_p \Pi(S_\infty,\varphi)$. In both cases there exists a minimal proof $\Pi(S_\infty,\varphi)$ of φ in S_∞ such that $\varphi >_p \Pi(S_\infty,\varphi)$ and by monotonicity and stability of $>_p$, $P[\varphi\sigma] >_p P[\Pi(S_\infty,\varphi)\sigma]$ for all proof contexts P and substitutions σ. In other words, φ is not involved in any minimal proof in S_∞ of a theorem of $\mathcal{T}$, since any occurrence of φ in a proof can be replaced by a proof $\Pi(S_\infty,\varphi)$ smaller than φ itself. It follows that φ is redundant in S_∞ on $\mathcal{T}$.

2. If φ is redundant in S_j on $\mathcal{T}$, since $\forall\psi \in \mathcal{T}$, $\Pi(S_j,\psi) \geq_p \Pi(S_\infty,\psi)$ by Definition 3.7 of completion, φ is redundant in S_∞ on $\mathcal{T}$. □

This theorem generalizes the following classical results:

Theorem 5.2 (Knuth and Bendix 1970) [11], (Huet 1981) [10], (Bachmair, Dershowitz and Hsiang 1986) [1] *If a derivation*

$(E_0;\emptyset)\vdash_{KB}(E_1;\emptyset)\vdash_{KB}\ldots(E_i;\emptyset)\vdash_{KB}\ldots$

by the Knuth-Bendix completion procedure does not fail and is uniformly fair *on the domain* $\mathcal{T}$ *of all equations, then* E_∞ *is a* confluent *term rewriting system.*

Knuth-Bendix completion fails if an unoriented equation persists. If a derivation by Knuth-Bendix completion does not fail, all the persistent equations are oriented into rewrite rules according to a reduction ordering and therefore E_∞ is a terminating rewrite system. By Theorem 5.1, E_∞ is saturated, i.e. locally confluent. By Newman's lemma [7], a terminating rewrite system is confluent if and only if it is locally confluent. Therefore E_∞ is confluent.

Theorem 5.3 (Hsiang and Rusinowitch 1987) [9], (Bachmair, Dershowitz and Plaisted 1989) [3] *If a derivation*

$(E_0;\emptyset)\vdash_{UKB}(E_1;\emptyset)\vdash_{UKB}\ldots(E_i;\emptyset)\vdash_{UKB}\ldots$

by the Unfailing Knuth-Bendix completion procedure is uniformly fair *on the domain* $\mathcal{T}$ *of all ground equations, then* E_∞ *is a* ground confluent *set of equations.*

For this second result we recall that given a set of equations E, $s\to_E t$ if $s\leftrightarrow_E t$ and $s\succ t$ for a reduction ordering $\succ$. The Unfailing Knuth-Bendix procedure assumes that $\succ$ is a complete simplification ordering. Since a complete simplification ordering is total on ground terms, $\leftrightarrow_{E_\infty}=\to_{E_\infty}\cup\leftarrow_{E_\infty}$ holds for ground terms and E_∞ is Church-Rosser on ground terms if and only if it is ground confluent. Since a complete simplification ordering is well founded, E_∞ is terminating on ground terms. The domain $\mathcal{T}$ of Unfailing Knuth-Bendix is the set of ground equations. By Theorem 5.1, E_∞ is saturated on $\mathcal{T}$, i.e. it is locally confluent on ground terms. By Newman's lemma E_∞ is ground confluent and therefore Church-Rosser on ground terms.

The Church-Rosser property on ground terms is important because $E\models\forall\bar{x}s\simeq t$ if and only if $\hat{s}\leftrightarrow_E^*\hat{t}$. If E is Church-Rosser on ground terms, $\hat{s}\leftrightarrow_E^*\hat{t}$ if and only if $\hat{s}\to_E^*\circ\leftarrow_E^*\hat{t}$ and therefore $E\models\forall\bar{x}s\simeq t$ can be decided by well founded reduction by E.

6 Redundancy

In the previous sections we introduced a notion of redundancy. The interest in redundancy of data elements in a theorem proving derivation resides in the importance of contraction inference rules. Contraction inference rules are necessary to make theorem proving feasible. However, few contraction rules are known. The purpose of studying redundancy is to get some insight about how to design new and powerful contraction rules. Therefore we conclude with a discussion on redundancy.

A notion of redundant clauses appeared in [13] and in [4], where the term "redundant" was first used. Redundant clauses according to these works are redundant in our sense. On the other hand, there are clauses which are intuitively redundant and redundant according to our definition, but not according to the definitions in [13] and [4].

Definition 6.1 (Rusinowitch 1988) [13] *A clause* φ *is* R-redundant *in a set* S *if there exists a*

clause $\psi \in S$ such that ψ properly subsumes φ, i.e. $\varphi \succ\!\!\!\cdot\ \psi$, where $\succ\!\!\!\cdot$ is the proper subsumption ordering on clauses.

R-redundancy has been recently investigated in [14] in the context of proofs by resolution in first order logic. Very high numbers of R-redundant clauses may be generated in such derivations, resulting in waste of space to hold them and in waste of time to perform the subsumption test to detect them. Two techniques to limit the generation of R-redundant clauses are proposed in [14].

The definition of redundancy in [4] assumes a well founded ordering $>^d$ total on ground clauses[2]:

Definition 6.2 (Bachmair and Ganzinger 1990) [4] *A clause φ is* B-redundant *in a set S if for all ground instances $\varphi\sigma$ of φ, there are ground instances $\psi_1 \dots \psi_n$ of clauses in S such that $\{\psi_1 \dots \psi_n\} \models \varphi\sigma$ and $\forall j, 1 \leq j \leq n, \varphi\sigma >^d \psi_j$.*

It is immediate to see that if the ordering $>^d$ is the proper subsumption ordering, B-redundancy specializes to R-redundancy and therefore a clause which can be subsumed is an example of a redundant clause:

Lemma 6.1 (Bachmair and Ganzinger 1990) [4] *R-redundant clauses are B-redundant.*

In our view, the intuition behind the notion of redundancy is that a clause φ is redundant in S if adding φ to S does not decrease any minimal proof in S (Definition 3.4). In fact our definition captures the meaning of Definition 6.2:

Theorem 6.1 *If a clause φ is B-redundant in S, then it is redundant on the domain of all ground clauses.*

Proof: we assume a proof ordering $>_p$ on ground proofs such that the minimal proof of a ground clause φ in S, $\Pi(S, \varphi)$, is the smallest set $\{\psi_1 \dots \psi_n\}$ of ground instances of clauses in S such that $\{\psi_1 \dots \psi_n\} \models \varphi$, according to the multiset extension ${>^d}_{mul}$ of the ordering $>^d$. Since $>^d$ is well founded and total on ground clauses, $>_p$ is well defined. Let φ be B-redundant in S. We show that $\Pi(S \cup \{\varphi\}, \psi) = \Pi(S, \psi)$ for all ground theorems ψ. Since $S \subset S \cup \{\varphi\}$, $\Pi(S \cup \{\varphi\}, \psi) \leq_p \Pi(S, \psi)$ trivially holds and therefore we simply have to show that $\Pi(S \cup \{\varphi\}, \psi) \not<_p \Pi(S, \psi)$. The proof is done by way of contradiction: if $\Pi(S \cup \{\varphi\}, \psi) <_p \Pi(S, \psi)$, then the smallest set of ground instances of clauses in $S \cup \{\varphi\}$ which logically entails ψ has the form $S' \cup \{\varphi\sigma_1 \dots \varphi\sigma_k\}$ for some set S' of ground instances of clauses in S and some ground substitutions $\sigma_1 \dots \sigma_k$. Since φ is B-redundant in S, for all $\varphi\sigma_i$, $1 \leq i \leq k$, there are ground instances $\{\psi^i_1 \dots \psi^i_n\}$ of clauses in S such that $\{\psi^i_1 \dots \psi^i_n\} \models \varphi\sigma_i$ and $\varphi\sigma_i >^d \psi^i_j$, $\forall j, 1 \leq j \leq n$. Therefore, $S' \cup \{\psi^i_1 \dots \psi^i_n\}_{i=1}^k <^d{}_{mul} S' \cup \{\varphi\sigma_1 \dots \varphi\sigma_k\}$ and $S' \cup \{\psi^i_1 \dots \psi^i_n\}_{i=1}^k \models \psi$, that is $S' \cup \{\varphi\sigma_1 \dots \varphi\sigma_k\}$ cannot be the smallest set entailing ψ. It follows that $\Pi(S \cup \{\varphi\}, \psi) = \Pi(S, \psi)$. □

On the other hand, there are cases where trivially redundant clauses are not B-redundant, whereas they are redundant according to our definition:

[2] In [4] a notion of *deletion ordering* is defined for this purpose. Well foundedness and totality on ground clauses are the only properties of a deletion ordering which are relevant to our discussion.

Example 6.1 *If $S = \{P, \neg R, R\}$, where P and R are ground atoms, P is intuitively redundant and it is redundant according to our Definition 3.4: the minimal proof of every ground theorem is given by $\{\neg R, R\}$, since $\{\neg R, R\}$ yields the empty clause and therefore any clause. However, if $R \succ P$, P is not B-redundant.*

This example shows that a notion of redundancy based on an ordering on clauses is not ideal, since different precedences on predicate symbols may be needed in order to characterize as redundant different clauses during a computation. A notion of redundancy based on a proof ordering seems to behave more satisfactorily.

7 Discussion

Intuitively, fairness of a search strategy means that every inference step which needs to be considered will eventually be considered. In completion-based methods, this usually means resolving all potential critical pairs. In theorem proving, on the other hand, one is not interested in critical pairs which may not contribute to a proof of the target theorem. Thus, in theorem proving applications fairness does not require resolving all possibile conflicts but only those which may lead to a proof. It is therefore conceivable to design a fair search strategy which ignores the majority of possible critical pairs.

One can actually even go further: since in theorem proving we are only interested in finding *one* proof, a search strategy can be considered fair as long as it does not remove the possibility of finding any proof. Thus, a search strategy may trim the search space considerably and still be fair as long as it does not trim away all the proofs. We feel that our approach, which separates the presentation from the target, provides a better framework than others for the study of such fair search strategies.

We conclude with some discussion about contraction and redundancy. A contraction inference rule either deletes sentences or replace them by others. An alternative scheme, called *deletion*, is given in [4]. The deletion scheme differs from our contraction scheme, since it only allows to infer S' from S by deleting a sentence in S. If this deletion scheme is adopted, an inference rule which replaces a sentence by other sentences has to be schematized as the composition of an expansion rule and a deletion rule. For instance, Simplification of $p \simeq q$ into $p \simeq q[r\sigma]_u$ is described as the generation of the equation $p \simeq q[r\sigma]_u$ followed by the deletion of $p \simeq q$.

This approach has the substantial drawback that it requires to consider more general inference rules than those actually used in the set of inference rules of a given strategy. For the simplification of $p \simeq q$ into $p \simeq q[r\sigma]_u$, the equation $p \simeq q[r\sigma]_u$ may not be a critical pair derivable by a superposition step. This is the case if $p \succ q$, that is the right hand side of a rewrite rule is simplified. A general paramodulation inference rule, which is not featured by the Unfailing Knuth-Bendix procedure, is then required in order to simulate simplification. We prefer our expansion/contraction schemes, since they allow us to classify directly every concrete inference rule of a given strategy as either an expansion rule or a contraction rule.

In Section 3 we have used redundancy to characterize those contraction rules which delete sentences. It is immediate to show that redundancy plays a role also for contraction rules which

replace a sentence by others. If a contraction inference step $(S \cup \{\psi\}; \varphi) \vdash (S \cup \{\psi'\}; \varphi)$ is proof-reducing by Condition 2 in Definition 3.2, the deleted sentence ψ is clearly redundant in $S \cup \{\psi'\}$. However, this is not necessarily true for a contraction step which strictly reduces the proof of the target. Such a step is reducing because it reduces the proof of the target, without any provision for the proofs of the other theorems. Therefore, according to our definition of proof-reduction, a contraction inference rule may also delete non redundant sentences, provided it is sound and it strictly reduces the proof of the target.

This is a further difference between our approach and the one in [4]. In [4] redundancy is used to define the deletion scheme itself: a deletion inference rule is a rule which deletes redundant sentences. We prefer not to relate so tightly the notions of contraction and redundancy, as long as the assumed notion of redundancy is not target-oriented. According to the three definitions of redundancy we have considered, a sentence is redundant if it is useless with respect to all the theorems in the domain. Therefore, these three definitions are not target-oriented. Our definition of proof reduction instead allows in principle very strong contraction rules which may replace even non redundant sentences if they do not help in proving a specific target.

Our whole framework is target-oriented tough, and therefore it naturally leads to a target-oriented definition of redundancy:

Definition 7.1 *A sentence φ is* redundant for ψ *in S, if* $\Pi(S, \psi) = \Pi(S \cup \{\varphi\}, \psi)$.

Clearly, a clause which is redundant on a domain $\mathcal{T}$ is redundant for all the targets in $\mathcal{T}$.

A target-oriented definition of redundancy applies for instance to the example we gave in the introduction, where the target $s \simeq t$ is an equation on a signature F_1 and the input presentation is the union of a set E_1 of equations on the signature F_1 and a set E_2 of equations on another signature F_2, disjoint from F_1. If we consider the problem in terms of the search process, it is fair to indefinitely postpone the equations in E_2. If we consider it in terms of the inference process, all the equations in E_2 are redundant for $s \simeq t$ and therefore can be eliminated. This duality of inference and search is not surprising, since most issues in theorem proving can be described both as properties of the inference mechanism and as properties of the search plan. We feel that thinking in terms of search is less categorical and therefore may give more flexibility. Further work is necessary to turn an abstract study of fairness, contraction and redundancy into concrete search plans and inference rules.

References

[1] L.Bachmair, N.Dershowitz, J.Hsiang, Orderings for Equational Proofs, in *Proceedings of the First Annual IEEE Symposium on Logic in Computer Science*, 346–357, Cambridge, MA, June 1986.

[2] L.Bachmair, Proofs Methods for Equational Theories, Ph.D. thesis, Department of Computer Science, University of Illinois, Urbana, IL.,1987.

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

[4] L.Bachmair, H.Ganzinger, Completion of First-Order Clauses with Equality by Strict Superposition, to appear in M.Okada, S.Kaplan (eds.), *Proceedings of the Second International Workshop on Conditional and Typed Rewriting Systems*, Montréal, Canada, June 1990.

[5] M.P.Bonacina, J.Hsiang, Completion procedures as Semidecision procedures, to appear in M.Okada and S.Kaplan (eds.), *Proceedings of the Second International Workshop on Conditional and Typed Term Rewriting Systems*, Montréal, Canada, June 1990.

[6] N.Dershowitz, Orderings for term-rewriting systems, *Theoretical Computer Science*, Vol. 17, 279–301, 1982.

[7] N.Dershowitz, J.-P.Jouannaud, Rewrite Systems, Chapter 15, Volume B, *Handbook of Theoretical Computer Science*, North-Holland, 1989.

[8] N.Dershowitz, A Maximal-Literal Unit Strategy for Horn Clauses, to appear in M.Okada, S.Kaplan (eds.), *Proceedings of the Second International Workshop on Conditional and Typed Rewriting Systems*, Montréal, Canada, June 1990.

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

[10] G.Huet, A Complete Proof of Correctness of the Knuth-Bendix Completion Algorithm, *Journal of Computer and System Sciences*, Vol. 23, 11–21, 1981.

[11] D.E.Knuth, P.Bendix, Simple Word Problems in Universal Algebras, in J.Leech (ed.), *Proceedings of the Conference on Computational Problems in Abstract Algebras*, Oxford, England, 1967, Pergamon Press, Oxford, 263–298, 1970.

[12] E.Kounalis, M.Rusinowitch, On Word Problems in Horn Theories, in E.Lusk, R.Overbeek (eds.), *Proceedings of the Ninth International Conference on Automated Deduction*, 527–537, Argonne, Illinois, May 1988, Springer Verlag, Lecture Notes in Computer Science 310, 1988.

[13] M.Rusinowitch, Theorem-proving with resolution and superposition: an extension of Knuth and Bendix procedure as a complete set of inference rules, Thèse d'Etat, Université de Nancy, 1987.

[14] R.Socher-Ambrosius, How to Avoid the Derivation of Redundant Clauses in Reasoning Systems, to appear in *Journal of Automated Reasoning*, 1990.