Flexible Re-enactment of Proofs*

Matthias Fuchs
Center for Learning Systems and Applications (LSA)
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
67653 Kaiserslautern
Germany
E-mail: fuchs@informatik.uni-kl.de

January 20, 1997

Abstract

We present a method for making use of past proof experience called flexible re-enactment (FR). FR is actually a search-guiding heuristic that uses past proof experience to create a search bias. Given a proof $\mathcal{P}$ of a problem solved previously that is assumed to be similar to the current problem $\mathcal{A}$, FR *searches* for $\mathcal{P}$ and in the "neighborhood" of $\mathcal{P}$ in order to find a proof of $\mathcal{A}$.

This heuristic use of past experience has certain advantages that make FR quite profitable and give it a wide range of applicability. Experimental studies substantiate and illustrate this claim.

*This work was supported by the *Deutsche Forschungsgemeinschaft (DFG)*.

1 Introduction

Automated deduction is essentially a search problem that gives rise to potentially infinite search spaces because of general undecidability. Despite these unfavorable conditions state-of-the-art theorem provers have gained a remarkable level of performance mainly due to (problem-specific) search-guiding heuristics and advanced implementation techniques. Nevertheless theorem provers can hardly rival a mathematician when it comes to proving "challenging" theorems. The main reason for this fact is also a major shortcoming of theorem provers: Unlike humans, theorem provers lack the ability to learn. Learning, however, is a key ability in any form of human problem solving, in particular in theorem proving.

The necessity to equip theorem provers with learning capabilities has been recognized quite early. But learning for theorem proving causes much more difficulties than learning in other areas of artificial intelligence because the premise that "*small changes of the problem cause small changes of its solution*" is not fulfilled at all. As a matter of fact, in theorem proving tiny variations of a problem specification can result in significant changes of the solution (proof). This circumstance complicates matters substantially.

Learning methods based on analogous proof transformation (e.g., [3], [2], [12], [16]) basically attempt to transform a known proof (*source proof*) of a problem solved previously (*source problem*) into a proof (*target proof*) of a given problem to be solved (*target problem*). Mostly, the mainly *deterministic* transformation procedure centers on some kind of analogy mapping obtained by comparing source and target problem. The transformation may include abstraction and planning steps. Occasionally, a little search may be involved in order to patch failures of the deterministic transformation procedure (e.g., [2]).

The prevailing determinism (and restricted forms of search if any) has the advantage that analogous proof transformation can be quite fast. The downside, however, is that source and target problem have to be very similar so that the source proof can be transformed into a target proof mainly with deterministic actions. For this reason, methods based on analogous proof transformation are preferably applied to inductive theorem proving. There, the inherent proof structures (base case, hypotheses, induction step) provide a suitable platform for such methods. However, for general (deductive) theorem proving in domains without any (recognizable) structure, these methods appear to be inappropriate.

An alternative approach to learning for theorem proving is to incorporate past problem-solving experience into the search-guiding heuristic (e.g., [18], [21], [7], [10], [6]). That is, when solving a target problem, the source problem is in some way exploited by the search-guiding heuristic. The theorem prover still conducts a search, but the heuristic is biased towards a certain area of the search space depending on the source proof and on the way the source proof is utilized by the heuristic. This approach has the advantage that the demands on similarity between source and target problem do not have to be as high as for methods based on proof transformation. Naturally we incur the usual overhead of search caused by exploring areas of the search space that do not

contribute to the proof eventually found. (A "suitable" choice of the source problem—a general difficulty for all learning approaches—and a suitable method for heuristically utilizing the respective source proof can of course reduce the overhead considerably.)

In this report we investigate a method for incorporating past proof experience into a search-guiding heuristic called *flexible re-enactment* (FR). FR basically attempts to re-enact the source proof *via search*, *if* possible. Flexibility is achieved by also searching in the "neighborhood" of the source proof or by using some "standard" (non-learning) heuristic if the trace of the source proof is (temporarily) lost. Both re-enactment and flexibility are combined in a monolithic structure that allows for shifting smoothly between flexibility and re-enactment depending on the search space encountered.

FR was first introduced in [10]. In this report we present a systematic experimental evaluation of FR that illustrates its performance and range of applicability. Furthermore, we counter a frequent and unjustified criticism of FR, namely to be a costly disguise for adding the source proof (in the form of lemmas) to the axiomatization of the target problem. But first, section 2 explains some basics of our view of automated theorem proving and introduces some concepts necessary for FR. Section 3 describes FR and discusses some of its properties. Experimental studies are reported on in section 4. Finally, a discussion in section 5 concludes the report.

2 Theorem Proving

Theorem provers can attempt to accomplish a task in various ways. We focus here on theorem provers that may be referred to as saturation-based theorem provers. This type of theorem prover is very common and is employed by provers based on the resolution method (e.g., [4]) or the Knuth-Bendix completion procedure (e.g., [1]). The principle working method of such a prover is to infer facts by applying given rules of inference, starting with a given set Ax of axioms, until the *goal* λ_G (the theorem to be proved) can be shown to be a logical consequence of Ax. A proof problem $\mathcal{A}$ is hence specified by $\mathcal{A} = (Ax, \lambda_G)$.

The prover maintains two sets of facts, the set F^A of *active facts* and the set F^P of *passive* or *potential facts*. In the beginning, $F^A = \emptyset$ and $F^P = Ax$. In the *selection* or *activation step* a fact $\lambda \in F^P$ is selected, removed from F^P, and put into F^A unless there is a fact $\lambda' \in F^A$ that subsumes λ (in symbols $\lambda' \lhd \lambda$) in which case λ is simply discarded. Note that $\lambda \equiv \lambda'$ (syntactic identity modulo renaming variables) implies $\lambda' \lhd \lambda$ (and of course $\lambda \lhd \lambda'$). If λ is indeed activated (put into F^A), all (finitely many) inferences involving λ are applied exhaustively, and inferred facts are added to F^P. Facts in F^P are so to speak known to be inferable (from F^A), but are not yet considered to be actually inferred.

The activation step is the only inherently indeterministic step in the proof procedure just sketched. Commonly, a heuristic $\mathcal{H}$ is employed to resolve the indeterminism at the (inevitable) expense of introducing search. $\mathcal{H}$ associates a natural number $\mathcal{H}(\lambda) \in \mathbb{N}$ (a *weight*) with each $\lambda \in F^P$ which is called "weighting λ with $\mathcal{H}(\lambda)$". The fact with

the smallest weight $\mathcal{H}(\lambda)$ is next in line for activation. Ties are usually broken in compliance with the FIFO strategy.

The search-guiding heuristic $\mathcal{H}$ is pivotal for efficiency. The quality of $\mathcal{H}$ can be measured in terms of redundant search effort. The sequence $\mathcal{S} \equiv \lambda_1; \ldots; \lambda_n$ of facts activated by $\mathcal{H}$ describes the search behavior of $\mathcal{H}$. Assuming that such a *search protocol* $\mathcal{S}$ represents a successful search, the last fact λ_n of $\mathcal{S}$ concluded the proof. By tracing back the application of inference rules starting with λ_n we can identify all those facts that actually contributed to concluding the proof. These facts are called *positive facts* and are collected in the set P of positive facts. The remaining facts constitute the set N of negative facts and represent redundant search effort.

The set P obtained when solving a source problem represents past proof experience. It is utilized by FR as described in the following section in order to reduce redundant search effort.

3 Flexible Re-enactment

Similarity between two proof problems $\mathcal{A}$ and $\mathcal{B}$ can occur in many variations. One possible kind of similarity is that a considerable number of the facts that contribute to a proof of $\mathcal{A}$ are also useful for proving $\mathcal{B}$ (or vice versa). This means in our terminology that the associated sets of positive facts share many facts. FR attempts to exploit such a similarity.

Assuming that a target problem $\mathcal{A}_T$ and a source problem $\mathcal{A}_S$ are similar in the way just described, it is reasonable to concentrate on deducing facts when attempting to prove $\mathcal{A}_T$ that also played a role in finding the source proof of $\mathcal{A}_S$, namely the set P_S of positive facts. When proving $\mathcal{A}_T$, FR prefers a $\lambda \in F^P$ (by giving it a relatively small weight $\mathtt{FR}(\lambda)$) if λ "bears significant resemblance" with a fact in P_S. Such a λ is called a *focus fact*. Depending on how strong the preference of focus facts is, FR will activate focus facts as soon as they appear in F^P and give little attention to other facts while there are focus facts. (If all source axioms occur in the target axiomatization, this essentially means that FR will re-enact the source proof.)

Although we assume the proofs of $\mathcal{A}_T$ and $\mathcal{A}_S$ to share many positive facts, a few of the facts useful for proving $\mathcal{A}_S$ may lead to focus facts that do not contribute to the proof of $\mathcal{A}_T$ eventually found. Besides the redundant search effort caused by these *irrelevant* focus facts, the crucial difficulty is to find the (non-focus) facts not needed for proving $\mathcal{A}_S$, but necessary for proving $\mathcal{A}_T$. These "missing" facts have to supplement the *relevant* focus facts, i.e., the focus facts that do contribute to the target proof. It is very likely that the missing facts are descendants[1] of relevant focus facts considering that the relevant focus facts already established the core of a proof and that there are only a few steps needed to conclude the proof. It is reasonable to assume that "taking the reasoning process a few steps further in the direction given by the focus facts",

[1] When applying an inference rule to facts $\lambda_1, \ldots, \lambda_m$, thus producing a fact λ, the facts $\lambda_1, \ldots, \lambda_m$ are called the (*immediate*) *ancestors* of the (*immediate*) *descendant* λ.

i.e., that also preferring their descendants will be profitable. Immediate descendants should be preferred the most. In the sequel we give a formal definition of FR.

3.1 Details of FR

For the definition of FR the notions 'difference' and 'distance' are pivotal. First, we define the *difference diff* between two facts λ and λ' to formalize the notion 'focus fact'.

$$diff(\lambda, \lambda') = \begin{cases} 0, & \lambda \lhd \lambda' \\ 100, & \text{otherwise.} \end{cases}$$

For the time being, we content ourselves with this simple definition of difference that centers on subsumption. Note that the values 0 and 100 are somewhat arbitrary but intuitive hints of percentages, denoting "perfect similarity" and "no similarity at all", respectively. As we shall see, the restriction of *diff* to $\mathbb{N}_{100} = \{0, 1, \ldots, 100\}$ entails that all further computations will produce values from $\mathbb{N}_{100}$ which makes computations more transparent.

diff is used to find out whether a given fact λ is a focus fact. We define

$$\mathcal{D}(\lambda) = \min\Big(\{diff(\lambda, \lambda') \mid \lambda' \in P_S\}\Big) .$$

Hence, $\mathcal{D}(\lambda)$ returns the minimal difference between a given fact λ (target) and the positive facts (source). If $\mathcal{D}(\lambda) = 0$ then λ is considered to be a focus fact, which complies with the ideas from above.

Now recall that also descendants of focus facts are to be favored. The preference given to them should, however, decrease with their distance from focus facts. The *distance* $d(\lambda)$ of a given fact λ measures distance, roughly said, in terms of the number of inference steps separating λ from ancestors which are focus facts. It depends on the distance of the immediate ancestors of λ from focus facts (if λ is not an axiom) and $\mathcal{D}(\lambda)$:

$$d(\lambda) = \begin{cases} \psi\Big(q, \mathcal{D}(\lambda)\Big), & \text{if } \lambda \text{ is an axiom} \\ \psi\Big(d(\lambda_1), \mathcal{D}(\lambda)\Big), & \text{if } \lambda_1 \text{ is the (only) immediate ancestor of } \lambda \\ \psi\Big(\gamma(d(\lambda_1), d(\lambda_2)), \mathcal{D}(\lambda)\Big), & \text{if } \lambda_1 \text{ and } \lambda_2 \text{ are the immediate ancestors of } \lambda. \end{cases}$$

The first argument of ψ represents the distance of the immediate ancestors. It is simply given as the distance of the immediate ancestor if there is just one immediate ancestor. If λ is an axiom (i.e., there are no ancestors), this value is specified by a parameter $q \in \mathbb{N}_{100}$. If λ has two immediate ancestors, then γ computes this value. (Note that $d(\lambda)$ is defined if λ has zero, one, or two immediate ancestors which is sufficient for most deduction systems. If the number of immediate ancestors should be in excess of two, then a weighted average of the ancestors' distances can be employed, for instance.) We chose a parameterized γ employing a parameter $q_1 \in [0; 1]$. Depending on q_1, the

result of γ ranges between the minimum and the maximum of the distances of the immediate ancestors:

$$\gamma(x,y) = \min(x,y) + \lfloor q_1 \cdot (\max(x,y) - \min(x,y)) \rfloor \ .$$

Using $q_1 = 0$ or $q_1 = 1$, γ computes the minimum or maximum, respectively. With $q_1 = 0.5$, γ computes the (integer part of the)[2] average.

The distance of immediate ancestors (or q) and $\mathcal{D}$ are combined by ψ yielding $d(\lambda)$. ψ should—for obvious reasons—satisfy the following criteria: On the one hand, $d(\lambda)$ should be minimal (i.e., 0) if $\mathcal{D}(\lambda) = 0$, in which case λ itself is a focus fact. On the other hand, the value produced by ψ should increase (reasonably) with the values obtained from γ and $\mathcal{D}$ in order to reflect the (growing) remoteness of λ from focus facts (and, in a way, from the source proof). As a matter of fact, γ already satisfies the latter criterion. Therefore, ψ is in parts identical to γ. It also uses a parameter $q_2 \in [0;1]$.

$$\psi(x,y) = \begin{cases} 0, & y = 0 \\ \min(x,y) + \lfloor q_2 \cdot (\max(x,y) - \min(x,y)) \rfloor \ , & \text{otherwise.} \end{cases}$$

The remaining task consists in designing FR so that it offers a reasonable degree of specialization in (i.e., focus on) the source proof that is paired with an acceptable degree of flexibility, i.e., the ability to cope with a target problem with a proof that requires minor to moderate deviations from the source proof. The use of d already provides sufficient specialization by directing the search towards the source proof. Its rudimentary flexibility can be enhanced by combining it with some "standard" ("general purpose") heuristic $\mathcal{H}$.

Among several sensible alternatives we picked the following:

$$\texttt{FR}(\lambda) = (d(\lambda) + p) \cdot \mathcal{H}(\lambda) \ , \quad p \in \mathbb{N}.$$

The parameter p controls the effect of $d(\lambda)$ on the final weight $\texttt{FR}(\lambda)$. $d(\lambda)$ will be dominant if $p = 0$. In this case, if $d(\lambda) = 0$, $\texttt{FR}(\lambda)$ will also be 0 regardless of $\mathcal{H}(\lambda)$. As p grows, $\mathcal{H}$ increasingly influences the final weight, thus mitigating the inflexibility of the underlying method, namely using $d(\lambda)$ *alone* as a measure for the suitability of a fact λ. For very large p, the influence of $d(\lambda)$ becomes negligible, and FR basically degenerates into $\mathcal{H}$.

3.2 Range of Applicability

FR is most suitable in situations where each source axiom is "covered" by a target axiom, i.e., for each source axiom λ' there is a target axiom λ so that $\lambda \lhd\!\rhd \lambda'$. Then FR can conduct a search guided by the source proof in the sense that FR can re-enact the

[2] We restrict our computations to $\mathbb{N}$, because there is no gain in "high precision arithmetic" when dealing with weighting functions, but there would be a loss in efficiency w.r.t. computation time.

source proof (or a more general proof) *without search* since, for each positive fact, there will be a focus fact subsuming it. The re-enacted source proof or, more precisely, the focus facts constituting the source proof, can serve as a basis for finding the target proof by searching in the "neighborhood" of the source proof, i.e., in particular by focusing on immediate descendants of focus facts (cp. subsection 4.2, case 1). (If additionally the source goal λ_S subsumes the target goal λ_T then plain re-enactment will succeed.) Under these conditions it is also possible to add the positive facts as lemmas to the target axiomatization, because it is known that every logical consequence of the source axioms is also a logical consequence of the target axioms. Therefore, FR has often been mistaken for a costly disguise for adding lemmas. However, adding lemmas creates kind of a "flat" structure as opposed to a more hierarchical structure when using focus facts and the notion of distance. In other words, when simply adding lemmas we have an enlarged set of axioms on the one side and their descendants on the other side. FR, however, imposes a kind of ordering on descendants, essentially giving immediate descendants top priority. (See also subsection 4.2.)

But FR is also useful in case source and target axiomatizations do not agree "obviously", i.e., agreement (logical equivalence) cannot be checked with simple (syntactic) subsumption criteria. Under these conditions it is not sound to add the positive facts of the source proof to the target axiomatization because it is not known whether they are logical consequences of the target axiomatization. In subsection 4.2 we shall see that FR performs quite well when target and source axiomatizations do not agree obviously, regardless of whether target and source goal agree (i.e., $\lambda_S \lhd \lambda_T$) or not.

So, FR is very versatile and covers a wide range of applicability. Nevertheless we want to point out that FR does of course not prove useful if source and target problem are not "similar enough", i.e., the sets of positive facts do not share enough facts. In other words, there are too few relevant focus facts that do not suffice to supply the core of a target proof. In addition, too many irrelevant focus facts hamper the search. (Naturally, relevant and irrelevant focus facts can only be identified at the end of a successful search.) In this case focusing on the source proof will be counterproductive. Hence it is important to find a suitable source problem. This difficulty is not addressed in this report (see [11] or [5] instead). Here, we want to show that—when given a suitable source problem—FR allows for solving target problems that pose serious difficulties or cannot be handled at all without FR.

4 Experiments

We conducted our experimental studies with a theorem prover for problems of *condensed detachment* (CD). Subsection 4.1 explains the basics of CD and motivates this choice. We want to point out, however, that we have also successfully applied FR to equational reasoning (cp. [5]). The results of our experiments with CD are summarized in section 4.2.

4.1 Condensed Detachment

CD allows for studying logic calculi with automated deduction systems. (See [22] and [13] for motivation and a detailed theoretical background.) There is only one inference rule (also denoted by CD) that manipulates first-order terms which we shall also call facts. The set of terms (facts) $Term(\mathcal{F}, \mathcal{V})$ is defined as usual, involving a finite set $\mathcal{F}$ of function symbols and an enumerable set $\mathcal{V}$ of variables.

CD (in its basic form) is defined for a distinguished binary function symbol $f \in \mathcal{F}$. CD allows us to deduce $\sigma(t)$ from two given facts $f(s,t)$ and s' if σ is the most general unifier of s and s'. A proof problem $\mathcal{A} = (Ax, \lambda_G)$ hence consists in deducing λ_G from Ax with continuous applications of CD. Subsumption is merely a matching problem, i.e., $\lambda \lhd \lambda'$ if (and only if) there is a match σ so that $\sigma(\lambda) \equiv \lambda'$.

CD fits the theorem proving framework given in section 2. Despite the simplicity of CD the arising proof problems can be very hard, sometimes even exceeding the limits of state-of-the-art provers. Therefore, CD is widely acknowledged as a testing ground for new ideas. It has been (and still is) used for this purpose quite frequently (e.g., [17], [24], [14], [19], [25], [10]). Furthermore, CD offers problems of a varying degree of difficulty, almost continuously ranging from (nearly) trivial to (very) challenging. This constellation is important if we want to employ learning techniques (like FR).

For the experiments we used the theorem prover 'CoDe' that realizes CD based on the concepts introduced in section 2. CoDe has a standard ("general purpose") heuristic $\mathcal{W}$ at its disposal that computes the weight $\mathcal{W}(\lambda)$ of a fact $\lambda \in F^P$ as $\mathcal{W}(\lambda) = c_\delta \cdot \delta(\lambda) + c_w \cdot w(\lambda)$, $c_\delta, c_w \in \mathbb{N}$. $w(\lambda)$ is equal to twice the number of function symbols occurring in λ plus the number of variables in λ. $\delta(\lambda)$ is the *level* or *depth* of λ: $\delta(\lambda) = 0$ if λ is an axiom; otherwise $\delta(\lambda)$ is the maximum of the levels of the immediate ancestors of λ plus 1. Furthermore, we set $\mathcal{W}(\lambda) = 0$ if $\lambda \lhd \lambda_G$. Consequently, facts subsuming the goal and hence concluding the proof are activated immediately. (See [8] or [9] for details.)

$\mathcal{W}$ is quite a successful heuristic in particular when taking into account the component δ reasonably (e.g., $c_\delta = 2$, $c_w = 1$). We use $\mathcal{W}$ as the standard heuristic required by FR, i.e., $\mathtt{FR}(\lambda) = (d(\lambda) + p) \cdot \mathcal{W}(\lambda)$. Here, we always set the coefficients $c_\delta = 0$ and $c_w = 1$, thus ignoring the level for reasons explained in [8]. (Basically both d and δ can be used to penalize depth which can lead to an undesirable and unprofitable "double penalty".) Based on extensive experiments (cp. [8]) the remaining parameters of FR are set as follows: $p = 20$, $q_1 = 0.75$, $q_2 = 0.25$, $q = 0$.

4.2 Experimental Results

We examined FR in the light of problems LCL040-1 through LCL072-1 (problems 1–33 in [14]) and problems LCL109-1 through LCL116-1 (problems 55–62 in [14]). These problems have the property emphasized in the preceding subsection which is important for learning, namely to offer a varying degree of difficulty that ranges from rather simple to challenging. They are a part of the public TPTP problem library ([20]) version 1.2.1.

Table 1: Case 1.

Target	Source	FR	FFU	RFF	lemmas	lemmas*	$\mathcal{W}$
058	060	23s (519)	89%	100%	—	5s (362)	25s (710)
060	058	8s (328)	95%	93%	—	12s (533)	26s (733)
071	070	4s (235)	75%	94%	—	3s (230)	—
	072	7s (366)	58%	100%	—	3s (254)	
068	067	4s (227)	72%	95%	2s (193)	2s (225)	68s (982)
	069	20s (629)	77%	100%	—	—	
114	113	3s (234)	83%	91%	12s (513)	10s (482)	—
116	113	5s (314)	78%	95%	14s (541)	10s (492)	—

(Note that the problems in the TPTP are given in CNF, a form suitable for resolution-based or tableaux-oriented theorem provers. The CNF "encoding" of problems of CD is not needed for CoDe.) We consider here mainly those problems that pose some difficulties for CoDe when using $\mathcal{W}$ to control the search. The performance of $\mathcal{W}$ with respect to the problems above (for various settings of the parameters c_δ and c_w) was thoroughly investigated in [8] (see also [9]) and compared with the performance of the renowned theorem prover Otter ([15]) as reported in [14]. This comparison showed that $\mathcal{W}$ can control the search so well that CoDe clearly outperforms Otter even though CoDe is inferior to Otter in terms of inferences per second. Therefore, it is not easy at all for FR to improve on $\mathcal{W}$.

In the following we shall examine the three interesting cases regarding the axiomatizations and the goals of target and source problems $\mathcal{A}_T = (Ax_T, \lambda_T)$ and $\mathcal{A}_S = (Ax_S, \lambda_S)$:

1. The same axiomatization ($Ax_T = Ax_S$), but different goals ($\lambda_S \not\lhd \lambda_T$, $\lambda_T \not\lhd \lambda_S$);
2. Different axiomatizations (Ax_T and Ax_S may share some axioms, but it is not the case that one axiomatization is part of the other), but the same goal ($\lambda_S \lhd \lambda_T$);
3. Different axiomatizations and different goals;

(We omit the fourth case "*the same axiomatization and the same goal*" because then plain re-enactment suffices and always succeeds in a negligible period of time.)

Note that only for case 1 it is sound to add positive facts of the source proof as lemmas to Ax_T. Therefore, we shall consider this possibility only for case 1. Table 1 displays our experiments concerning case 1. The first two columns list target and source problems, respectively. (The names are abbreviated, e.g., we write 058 instead of LCL058-1.) The source problems were chosen according to some simple, automatable criteria. (This is not a relevant issue in this report. See [11] instead.) Note that we employed source proofs found by $\mathcal{W}$ whenever possible in order to avoid creating particularly similar proofs with FR. For instance, we did *not* use the proof of LCL058-1 found by FR using a proof of LCL060-1 when proving LCL060-1 with FR (and source LCL058-1).

The results of FR are given in the correspondingly labeled column. The entries display run-time (approximate CPU time in seconds obtained on a SPARCstation 10) and the number of activation steps (i.e., the length of the search protocol) in parentheses. This number gives us a rough idea of the search effort. (The length of the search protocol usually does not correlate well with the length of the proof eventually found. This is also not an issue here since we are only interested in *some* proof, not necessarily a short one.)

Column 'FFU' shows the "focus fact usage", i.e., the share (relevant) focus facts have of the target proof, given as an approximate percentage. Column 'RFF' shows the share which facts that account for relevant focus facts have of the source proof. This value hence gives us an idea of how many facts of the source proof proved useful for finding the target proof. Consider, for instance, target problem LCL071-1 and source problem LCL070-1. The target proof consists of 20 facts. 15 of the 20 facts are focus facts. Hence, FFU is $15/20 = 75\%$. These 15 relevant focus facts go back to 15 facts in the source proof. Since 16 facts constitute the source proof, RFF is $15/16 \approx 94\%$.

The values of RFF listed in table 1 indicate that it is important that RFF is rather high. Given case 1 (equal source and target axiomatization) this is understandable considering that the share $100\% - \text{RFF}$ of the source proof will definitely give rise to irrelevant focus facts that may disarrange the search. Large values of FFU indicate that a large part of the target proof goes back to re-enactment and hence could be found rather efficiently.

When adding the positive facts of the source proof as lemmas to the axiomatization of the target problem, CoDe employed $\mathcal{W}$ with $c_\delta = 2$ and $c_w = 1$ to guide the search. (This parameter setting turned out to be generally useful during the experiments reported on in [8].) The results of utilizing lemmas in this manner are listed in column 'lemmas'. The entries again show run-time and length of the search protocol. The entry '—' signifies that no proof was found in an "acceptable" period of time (one hour).

A slight modification of lemma usage is to assign the weight 0 to all axioms (including added lemmas) so that they are immediately activated. Column 'lemmas*' shows that this modification is crucial in order to make adding lemmas competitive. The improvements are understandable because we can thus coerce re-enactment and hence simulate the obviously profitable re-enactment part of FR. Nonetheless, table 1 does not reveal any significant advantage of adding lemmas compared to FR. As a matter of fact, adding lemmas can cause a failure when FR still succeeds (cp. target LCL068-1, source LCL069-1). An opposite observation (failure of FR, success of adding lemmas) has so far not been made. This leads us to conclude that the "hierarchical" search induced by the distance measure d is pivotal. (Note, however, that the component δ of $\mathcal{W}$ (switched off for FR) also allows for taking into account distance or depth, but in a much cruder way than d does. Basically, the ancestor with maximal depth determines the depth of a descendant. The depth of the other ancestor has no effect.)

The last column of table 1 lists the results of $\mathcal{W}$ as a point of reference. The generally useful parameter setting ($c_\delta = 2$ and $c_w = 1$) fails for all target problems of table 1

$\star$	1	1		$i(x, i(y, x))$
$\star$	2	2	[1 , 1]	$i(x, i(y, i(z, y)))$
$\star$	3	3		$i(i(n(x), n(y)), i(y, x))$
$\star$	4	4		$i(i(x, y), i(i(y, z), i(x, z)))$
$\star$	5	5	[4 , 1]	$i(i(i(x, y), z), i(y, z))$
$\star$	6	6	[5 , 3]	$i(n(x), i(x, y))$
$\star$	7	7	[4 , 6]	$i(i(i(x, y), z), i(n(x), z))$
$\star$	8	8		$i(i(i(x, y), y), i(i(y, x), x))$
$\star$	9	9	[5 , 8]	$i(x, i(i(x, y), y))$
$\star$	10	10	[8 , 2]	$i(i(i(x, i(y, x)), z), z)$
$\star$	12	11	[4 , 9]	$i(i(i(i(x, y), y), z), i(x, z))$
$\star$	15	12	[4 , 3]	$i(i(i(x, y), z), i(i(n(y), n(x)), z))$
$\star$	16	13	[12 , 10]	$i(i(n(x), n(i(y, i(z, y)))), x)$
$\star$	17	14	[7 , 13]	$i(n(n(x)), x)$
$\star$	18	15	[3 , 14]	$i(x, n(n(x)))$
$\star$	19	16	[9 , 3]	$i(i(i(i(n(x), n(y)), i(y, x)), z), z)$
$\star$	21	17	[4 , 4]	$i(i(i(i(x, y), i(z, y)), u), i(i(z, x), u))$
$\star$	22	18	[11 , 17]	$i(i(x, y), i(i(z, x), i(z, y)))$
$\star$	23	19	[18 , 15]	$i(i(x, y), i(x, n(n(y))))$
$\star$	24	20	[17 , 16]	$i(i(x, i(n(y), n(z))), i(x, i(z, y)))$
	58	21	[4 , 14]	$i(i(x, y), i(n(n(x)), y))$
	148	22	[20 , 21]	$i(i(x, n(y)), i(y, n(x)))$
	233	23	[4 , 19]	$i(i(i(x, n(n(y))), z), i(i(x, y), z))$
	234	24	[23 , 22]	$i(i(x, y), i(n(y), n(x)))$

Figure 1: Proof of `LCL114-1` found by `FR` when using source `LCL113-1`.

(and also of tables 2 and 3). (Recall that the selected problems are the ones that cause difficulties for $\mathcal{W}$.) The successful runs of $\mathcal{W}$ were obtained with $c_\delta = 0$ and $c_w = 1$—the setting that also `FR` employs. These experiments show that `FR` enables CoDe to prove problems quite fast which it could not handle with $\mathcal{W}$ and one of the two mentioned parameter settings. (There are, however, two different and very problem-specific parameter settings for problems `LCL068-1` and `LCL071-1` that make it possible to prove these problems with $\mathcal{W}$. See [8].)

Figure 1 displays the proof of `LCL114-1` found by `FR` using source `LCL113-1` as a typical example of case 1. The proof listing shows (from left to right) the number of the activation step, the number of the proof step, the ancestors in brackets, given as numbers referring to previous proof steps (or a blank space for axioms), and the facts themselves. Focus facts are marked by '$\star$'. The identical axiomatizations of target and source make it possible that a large initial part of the target proof (FFU = 20/24 $\approx$ 83%) consists of focus facts. This part also constitutes the main part of the source proof (RFF = 20/22 $\approx$ 91%) and is found very efficiently with only four redundant activations. Three of the four (non-focus) facts necessary to conclude the proof can

Table 2: Case 2.

Target	Source	FR	FFU	RFF	$\mathcal{W}$
054	042	118s	31%	79%	—
058	045	4s	23%	82%	25s
045	058	< 1s	95%	51%	—
042	054	< 1s	84%	55%	—

Table 3: Case 3.

Target	Source	FR	FFU	RFF	$\mathcal{W}$
042	058	—	—	—	—
	060	126s	82%	95%	
045	060	22s	76%	41%	—
058	042	18s	20%	42%	25s
060	042	—	—	—	26s
	045	11s	18%	82%	

be considered to be "in the neighborhood" of the source proof since they have at least one ancestor that is a focus fact. The final proof step is no difficulty since a fact subsuming the goal is given the weight 0 and is therefore activated immediately (cf. subsection 4.1).

Experiments regarding cases 2 and 3 are summarized by tables 2 and 3. These tables are organized like table 1 without the columns concerning lemmas since the soundness of adding lemmas is—as mentioned earlier—not guaranteed for cases 2 and 3. As a matter of fact, for case 2 adding lemmas is completely pointless since the goal itself would be added as a lemma.

Considering tables 2 and 3 we can again observe that FR allows CoDe to solve problems rather quickly which were out of reach when using $\mathcal{W}$. The two bottom rows of table 2 indicate that FR can also cope with situations where large parts of the target proof go back to quite a small part of the source proof (i.e., large FFU, but rather small RFF). Note that—due to different axiomatizations—it is not necessarily the case that the share 100% − RFF of the source proof entails irrelevant focus facts. The results concerning problems LCL042-1, LCL058-1, and LCL060-1 in different roles as source and target problems (cp. tables 1 and 2) reveal that (quite expectedly) FR is not "symmetric". That is, a proof of a problem A found by FR when using a source problem B does not guarantee to find a proof of B when using A as source.

Figure 2 reveals that FR can succeed even if the target proof contains relatively few focus facts (FFU = 15/49 ≈ 31%). Nonetheless, the majority of facts constituting the source proof of LCL042-1 (RFF = 15/19 ≈ 79%) contribute to the proof of LCL054-1 found by FR (thus keeping the number of irrelevant focus facts small). In situations like this, the search conducted by FR is at first mainly guided by $\mathcal{W}$ as an integral part of FR. (Recall that different axiomatizations at the beginning usually only allow for re-enacting very small parts of the source proof. In figure 2 only two axioms are "re-enacted".) The "trace" of the source proof can be picked up (to a certain extent) later on when more and more focus facts appear.

Figure 2 is a perfect example for the flexibility of FR stemming from the integration of $\mathcal{W}$ that here basically prepares the ground for re-enactment. This is kind of opposite to the situation in figure 1 where $\mathcal{W}$ together with the preference of descendants account for concluding the proof while the groundwork is done by re-enactment.

$\star$ 1 $i(x, i(n(x), y))$
$\star$ 2 $i(i(x, y), i(i(y, z), i(x, z)))$
3 $i(i(n(x), x), x)$
4 [2, 1] $i(i(i(n(x), y), z), i(x, z))$
5 [4, 3] $i(x, x)$
6 [1, 5] $i(n(i(x, x)), y)$
7 [2, 3] $i(i(x, y), i(i(n(x), x), y))$
8 [2, 2] $i(i(i(i(x, y), i(z, y)), u), i(i(z, x), u))$
9 [8, 4] $i(i(x, n(y)), i(y, i(x, z)))$
10 [4, 9] $i(x, i(y, i(n(x), z)))$
11 [9, 6] $i(x, i(n(i(y, y)), z))$
12 [2,10] $i(i(i(x, i(n(y), z)), u), i(y, u))$
13 [2, 7] $i(i(i(i(n(x), x), y), z), i(i(x, y), z))$
14 [8, 8] $i(i(x, i(y, z)), i(i(u, y), i(x, i(u, z))))$
15 [2,11] $i(i(i(n(i(x, x)), y), z), i(u, z))$
16 [15, 3] $i(x, i(y, y))$
17 [2,16] $i(i(i(x, x), y), i(z, y))$
18 [17, 7] $i(x, i(i(n(y), y), y))$
19 [2,18] $i(i(i(i(n(x), x), x), y), i(z, y))$
20 [8,19] $i(i(x, i(n(y), y)), i(z, i(x, y)))$
21 [12,20] $i(x, i(y, i(z, x)))$
22 [20,21] $i(x, i(y, i(z, y)))$
$\star$23 [22,22] $i(x, i(y, x))$
24 [2,23] $i(i(i(x, y), z), i(y, z))$
25 [24, 2] $i(x, i(i(x, y), i(z, y)))$
$\vdots$

$\vdots$
26 [2,25] $i(i(i(i(x, y), i(z, y)), u), i(x, u))$
27 [26,20] $i(x, i(y, i(i(i(x, z), z))))$
28 [20,27] $i(x, i(y, i(i(y, z), z)))$
29 [28,23] $i(x, i(i(x, y), y))$
$\star$30 [14,29] $i(i(x, i(y, z)), i(y, i(x, z)))$
$\star$31 [30, 1] $i(n(x), i(x, y))$
$\star$32 [30,25] $i(i(x, y), i(x, i(z, y)))$
$\star$33 [2,31] $i(i(i(i(x, y), z), i(n(x), z))$
$\star$34 [30, 2] $i(i(x, y), i(i(z, x), i(z, y)))$
$\star$35 [33,34] $i(n(x), i(i(y, x), i(y, z)))$
$\star$36 [2,32] $i(i(i(x, i(y, z)), u), i(i(x, z), u))$
$\star$37 [34,30] $i(i(x, i(y, i(z, u))), i(x, i(z, i(y, u))))$
38 [30,35] $i(i(x, y), i(n(y), i(x, z)))$
39 [34, 3] $i(i(x, i(n(y), y)), i(x, y))$
40 [2,30] $i(i(i(i(x, i(y, z)), u), i(i(y, i(x, z)), u))$
41 [8,39] $i(i(n(x), y), i(i(y, x), x))$
42 [37,36] $i(i(i(x, i(y, z)), i(u, v)), i(u, i(i(x, z), v)))$
43 [2,38] $i(i(i(n(x), i(y, z)), u), i(i(y, x), u))$
44 [43,41] $i(i(x, y), i(i(i(x, z), y), y))$
45 [30,44] $i(i(i(x, y), z), i(i(x, z), z))$
$\star$46 [13,45] $i(i(x, y), i(i(n(x), y), y))$
$\star$47 [42,46] $i(i(n(x), i(y, z)), i(i(x, z), i(y, z)))$
$\star$48 [47,35] $i(i(x, i(y, z)), i(i(y, x), i(y, z)))$
$\star$49 [40,48] $i(i(x, i(y, z)), i(i(x, y), i(x, z)))$

Figure 2: Proof of `LCL054-1` found by `FR` using source `LCL042-1`. Activation step numbers are omitted here. (1146 activation steps were needed.)

5 Discussion

We presented a method called flexible re-enactment (`FR`) that exploits past proof experience with heuristic means. When searching for the (target) proof of a given proof problem, `FR` *attempts* to re-enact a given source proof essentially by *searching* for the facts that constitute the source proof. Flexibility is achieved by also searching in the "neighborhood" of the source proof, i.e., by also preferring descendants of the facts constituting the source proof. Experiments have demonstrated that `FR` allows for solving problems that could not be handled before.

The similarity between the source and target proof required so that `FR` or in other words a "flexible, search-based re-enactment of the source proof" can succeed is basically reflected by the number (or percentage) of facts constituting the source proof that are also useful for a target proof: the more, the better. But such a kind of similarity can only be assessed *a posteriori*. Hence, *a priori* similarity can only be estimated with "heuristic indicators" such as, for instance, the percentage of source axioms that are also present in the target axiomatization. Simple, easy to check criteria like this can already produce remarkable results (cp. [11]) which supports our belief that selecting an appropriate source problem can be automated reasonably (see also [5]). Naturally,

in order to be independent of symbol names, methods for finding a suitable renaming of symbols must be available (cf. [5]). Thus, for FR similarity is kind of a "loosely" defined notion that perforce cannot and does not play as important a role as it does, e.g., in [12]. There, similarity is a clearly defined central notion, but—probably a common trade-off—restricts applicability.

FR is not limited to the use of just one source proof. The design of FR allows us to exploit an arbitrary number of source proofs. But the usefulness of more than one source proof is questionable ("a jack of all trades, but master of none"). Redundant search effort might become a problem that outweighs possible advantages like having available more focus facts which undoubtedly increases the chance of having the "right" ones. Nonetheless, exploiting a pool of proof experience rather than one piece also has certain advantages (cp. [6] where sophisticated abstraction techniques are used).

Finally, FR can of course also be employed for *proof checking* and *proof completion*. Proof checking essentially corresponds to plain re-enactment. Proof completion means that the given positive facts represent the "skeleton" of a source proof with a few intermediate steps missing. Under these conditions the positive facts will give rise only to relevant focus facts. Thus, FR can search for the missing intermediate steps *without* being confused and possibly misguided by irrelevant focus facts which makes things easier than in general applications of FR.

In [23] the *hints strategy* (HS) has been investigated as to its usefulness for proof checking, proof completion, and also for proof finding. Hints basically are the counterparts of the positive facts of a source proof. HS and FR have commonalities, but also have different features and design concepts. In order to search for (or complete) a proof with certain properties, HS also allows for *avoiding* (instead of focusing on) hints—a feature that was not an objective when designing FR. But when it comes to searching for a (target) proof without such constraints, FR subsumes HS in particular because of its ability to deal sensibly with non-focus facts that are descendants of focus facts, thus exploiting a source proof beyond mere re-enactment—a concept not present in HS.

References

[1] **Bachmair, L.; Dershowitz, N.; Plaisted, D.:** *Completion without Failure*, Coll. on the Resolution of Equations in Algebraic Structures, Austin, TX, USA (1987), Academic Press, 1989.

[2] **Brock, B.; Cooper, S.; Pierce, W.:** *Analogical Reasoning and Proof Discovery*, Proc. CADE-9, Argonne, IL, USA, 1988, LNCS 310, pp. 454–468.

[3] **Bundy, A.:** *The Use of Explicit Plans to Guide Inductive Proofs*, Proc. CADE-9, Argonne, IL, USA, 1988, LNCS 310, pp. 111–120.

[4] **Chang, C.L.; Lee, R.C.:** *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973.

[5] **Denzinger, J.; Fuchs, Matt.; Fuchs, Marc:** *High Performance ATP Systems by Combining Several AI Methods*, SEKI Report SR-96-09, University of Kaiserslautern, 1996, `http://www.uni-kl.de/AG-AvenhausMadlener/fuchs.html`.

[6] **Denzinger, J.; Schulz, S.:** *Learning Domain Knowledge to Improve Theorem Proving*, Proc. CADE-13, New Brunswick, NJ, USA, 1996, LNAI 1104, pp. 62–76.

[7] **Fuchs, Matt.:** *Learning Proof Heuristics by Adapting Parameters*, Proc. 12^{th} ICML, Tahoe City, CA, USA, 1995, pp. 235–243.

[8] **Fuchs, Matt.:** *Experiments in the Heuristic Use of Past Proof Experience*, SEKI Report SR-95-10, University of Kaiserslautern, 1996, obtainable via WWW at `http://www.uni-kl.de/AG-AvenhausMadlener/fuchs.html`.

[9] **Fuchs, Matt.:** *Powerful Search Heuristics Based on Weighted Symbols, Level, and Features*, Proc. FLAIRS-96, Key West, FL, USA, 1996, pp. 449–453.

[10] **Fuchs, Matt.:** *Experiments in the Heuristic Use of Past Proof Experience*, Proc. CADE-13, New Brunswick, NJ, USA, 1996, LNAI 1104, pp. 523–537.

[11] **Fuchs, Matt.:** *Towards Full Automation of Deduction: A Case Study*, SEKI Report SR-96-07, University of Kaiserslautern, 1996, obtainable via WWW at `http://www.uni-kl.de/AG-AvenhausMadlener/fuchs.html`.

[12] **Kolbe, T.; Walther, C.:** *Reusing Proofs*, Proc. 11^{th} ECAI '94, Amsterdam, HOL, 1994, pp. 80–84.

[13] **Łukasiewicz, J.:** *Selected Works*, L. Borkowski (ed.), North-Holland, 1970.

[14] **McCune, W.; Wos, L.:** *Experiments in Automated Deduction with Condensed Detachment*, Proc. CADE-11, Saratoga Springs, NY, USA, 1992, LNAI 607, pp. 209–223.

[15] **McCune, W.:** *OTTER 3.0 reference manual and guide*, Techn. report ANL-94/6, Argonne Natl. Laboratory, 1994.

[16] **Melis, E.:** *A Model of Analogy-driven Proof-plan Construction*, Proc. 14^{th} IJCAI, Montreal, CAN, AAAI Press, 1995, pp. 182–189.

[17] **Peterson, G.J.:** *An Automatic Theorem Prover for Substitution and Detachment Systems*, Notre Dame Journal of Formal Logic, Vol. 19, Number 1, January 1976, pp. 119–122.

[18] **Slagle, J.R.; Farrell, C.D.:** *Experiments in Automatic Learning of a Multipurpose Heuristic Program*, Comm. of the ACM, Vol. 14, No. 2, 1971, pp. 91–99.

[19] **Slaney, J.:** *SCOTT: A Model-guided Theorem Prover*, Proc. IJCAI '93, Chambery, FRA, 1993, pp. 109–114.

[20] **Sutcliffe, G.; Suttner, C.; Yemenis, T.:** *The TPTP Problem Library*, Proc. CADE-12, Nancy, FRA, 1994, LNAI 814, pp. 252–266.

[21] **Suttner, C.; Ertel, W.:** *Automatic Acquisition of Search-guiding Heuristics*, Proc. CADE-10, Kaiserslautern, FRG, 1990, LNAI 449, pp. 470-484.

[22] **Tarski, A.:** *Logic, Semantics, Metamathematics*, Oxford University Press, 1956.

[23] **Veroff, R.:**, *Using Hints to Increase the Effectiveness of an Automated Reasoning Program: Case Studies*, JAR **16:**223–239, 1996.

[24] **Wos, L.:** *Meeting the Challenge of Fifty Years of Logic*, JAR **6:**213–232, 1990.

[25] **Wos, L.:** *Searching for Circles of Pure Proofs*, JAR **15:**279–315, 1995.