# Experiments in the Heuristic Use of Past Proof Experience*

Matthias Fuchs
Fachbereich Informatik, Universität Kaiserslautern
Postfach 3049, 67653 Kaiserslautern
Germany
E-mail: fuchs@informatik.uni-kl.de

November 28, 1995

**Abstract**

Problems stemming from the study of logic calculi in connection with an inference rule called "condensed detachment" are widely acknowledged as prominent test sets for automated deduction systems and their search guiding heuristics. It is in the light of these problems that we demonstrate the power of heuristics that make use of past proof experience with numerous experiments.

We present two such heuristics. The first heuristic attempts to re-enact a proof of a proof problem found in the past in a flexible way in order to find a proof of a similar problem. The second heuristic employs "features" in connection with past proof experience to prune the search space. Both these heuristics not only allow for substantial speed-ups, but also make it possible to prove problems that were out of reach when using so-called basic heuristics. Moreover, a combination of these two heuristics can further increase performance.

We compare our results with the results the creators of OTTER obtained with this renowned theorem prover and this way substantiate our achievements.

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

# Contents

# 1 Introduction

Automated deduction is—at its lowest level—a search problem that spans huge search spaces. The general undecidability of problems connected with (automated) deduction entails an indeterminism that has to and can only be tackled with heuristics. Mostly, automated deduction is employed to prove that a given conjecture can be deduced from an also given set of axioms. In order to solve such tasks automated proving systems utilize a (fixed) set of inference rules whose applications are responsible for both indeterminism and the immense size (the "combinatorial explosion") of the search space. Despite a far superior inference rate the computer is inferior to (human) mathematicians when it comes to proving "challenging" theorems. One prominent reason for this drawback of automated proving systems is their inability to make use of past experience, which is very often quite helpful or even an indispensable key to success. Therefore, it stands to reason to upgrade automated proving systems on that score.

But exploiting past proof experience fruitfully is in general neither trivial nor does it come without hazards. The main problem is that analogy in the widest sense is hard to define, to detect and to apply in the area of automated deduction. In other branches of artificial intelligence various applications of analogy have proven to be powerful tools (see, for instance, [Ca86], [Bu89]). These research areas profit from the fact that "*small changes of the problem description (usually) cause small changes of the solution*". This is definitely not true for automated deduction (proving). Consequently, we have to be very careful about making use of past proof experience in order not to stumble into a major pitfall of this kind of reuse, namely making things considerably worse compared to proving from scratch (cp. [KN93]).

In spite of these bleak prospects of success we still think that it is worthwhile equipping an automated proving system with the option to utilize experience gained in the past.

Most researchers in this research area are attempting to devise methods that allow to construct (compute) a proof $\mathcal{P}_\mathcal{B}$ of a new proof problem $\mathcal{B}$ (the *target*) from a known proof $\mathcal{P}_\mathcal{A}$ of a previously solved problem $\mathcal{A}$ (the *source*) using some pre-defined "analogy mapping" (e.g., [Kl71], [BCP88], [KW94], [Cu95]). The principle of our method, however, consists in incorporating information obtainable from previous proofs into the omnipresent (search guiding) heuristics used by an automated deduction system (see [Fu95a], [SE90] or [SF71] for related approaches). A significant advantage of such an approach is a suitable compromise between the flexibility (generality) stemming from the "original" heuristics and the specialization coming from the incorporation of information on previous proofs. Since we do not intend to transform the source proof into the target proof through a chain of deterministic analogous transformation steps, but employ a "conventional" heuristic upgraded with information on a source proof in order to *search* for the target proof, proofs do not have to be as similar for our method to be successful as they have to be when using a "constructive" approach.

The first heuristic we are going to present attempts to re-enact *flexibly* a given proof (the *source proof*) of a proof problem found in the past in order to find a proof of a (novel) proof problem (more quickly). A *flexible* re-enactment is necessary to enlarge the class of problems that can be tackled profitably with such an approach. Flexibility is achieved by giving (moderate) preference to deductions also present in the source proof and to deductions possible on account of these. This way, we do not rigorously exclude steps that do not comply with the source proof. Consequently, necessary (moderate) deviations from the source proof can be compensated for.

The second heuristic is based on the concept of *features* which has been exploited before in different ways (e.g. [SF71], [SE90], or [Re83] for an approach tested in connection with the 'fifteen-puzzle'). Features basically represent structural properties of the objects that are manipulated (by an automated deduction system) with a (natural) number, and therefore can be considered as functions abstracting from structure. The results of abstraction, namely the feature values, are used by our second heuristic to prune the search space. The exact way how feature values are to influence the behavior of the heuristic is determined with the help of past experience. Furthermore, a combination of these two heuristics further improves performance as several experiments have shown.

For our experimentation we have chosen problems that originate from the study of logic calculi with an inference rule called *condensed detachment* (also known as "*substitution and detachment*"; see [Ta56] and [Lu70] for original work and [Pe76], [Wo90], [MW92], [Sl93] for work regarding automated deduction in this context). The reason for this choice is twofold: Firstly, there is a large number of such problems within a wide spectrum of difficulty, almost continuously ranging from (nearly) trivial to (very) challenging. This constellation is important if we want to tackle problems with methods that are based on previous proof experience. Secondly, the simplicity of the calculi in connection with condensed detachment (in terms of an unproblematic application of this inference rule) makes it easier to study the essential aspects of our methods without having to deal with complications on account of complex inference rules. But we strongly emphasize that the simplicity of the inference rule does not imply the simplicity of arising proof problems. On the contrary, the proof problems offer at least the same degree of difficulty known from other fields of automated deduction (e.g., resolution or equational reasoning), namely huge (potentially infinite) search spaces with no (obvious) hints how to proceed. Therefore, we are convinced that the methods (heuristics) proposed in this report will be profitably applicable to other deduction systems as well. (This has already been examined in part for an equational prover based on the unfailing Knuth-Bendix completion procedure, see [Fu95b]).

The experimental results sustain the viability of our approach. With an experimental program 'CoDe' we were able to achieve outstanding speed-ups and even to handle problems that were out of reach when not using past experience. A comparison with the results obtained with OTTER as reported in [MW92] underlines the significance of our results.

Section 2 introduces the study of logic calculi with condensed detachment and an automated deduction system for that purpose named 'CoDe'. Section 3 presents the basic heuristic which serves as a foundation for all other heuristics that exploit past experience. The subsequent sections 4 and 5 describe our first heuristic that makes use of past experience, namely "flexible re-enactment", and experimental results obtained with it. Sections 6 and 7 do the same with respect to our second, feature-based heuristic. Section 8 deals with a combination of these two heuristics. Finally, a discussion in section 9 brings this report to a close.

# 2 The Study of Logic Calculi With Condensed Detachment

This section introduces in subsection 2.1 an inference rule called *condensed detachment* and outlines its purpose in the study of logic calculi. Subsection 2.2 will present an automated deduction system that centers on condensed detachment.

The purpose of this section consists in presenting the study of logic calculi as a research area that can be tackled with automated deduction systems. Although we do not intend to give a detailed theoretical background (for this purpose see [Ta56] and [Lu70]), we want to emphasize that the study of logic calculi is recognized as a challenging field for automated deduction (cf. [MW92] and also [Wo90]). As a matter of fact, the arising problems are of varying difficulty, ranging from almost trivial to extremely difficult, including problems that have not yet been proven by an unassisted automated deduction system. This makes it particularly interesting to approach these problems with methods that attempt to learn from previous experiences.

## 2.1 Condensed Detachment

The inference rule 'condensed detachment' is the central part of the different logic calculi we are going to investigate. This inference rule manipulates first-order terms which we shall also call *facts*. Terms (facts) are defined as usual.

**Definition 2.1 (Term/Fact)** *Let $\mathcal{F}$ be a finite set of function symbols, $\tau : \mathcal{F} \to I\!N$ denote the arity of each $f \in \mathcal{F}$ and $\mathcal{V}$ be an enumerable set of variables ($\mathcal{V} \cap \mathcal{F} = \emptyset$). The set of terms $Term(\mathcal{F}, \mathcal{V})$ is recursively defined by*

1. *$x \in Term(\mathcal{F}, \mathcal{V})$ for all $x \in \mathcal{V}$*
2. *$f(t_1, \ldots, t_n) \in Term(\mathcal{F}, \mathcal{V})$ iff $f \in \mathcal{F}$, $\tau(f) = n$, $t_1, \ldots, t_n \in Term(\mathcal{F}, \mathcal{V})$*

We use the symbols $x$, $y$, $z$, $u$, $v$, $w$ and $x_1, x_2, \ldots$ to denote variables. Variables of a term are implicitly $\forall$-quantified.

The inference rule 'condensed detachment' is defined for a distinguished binary function symbol from $\mathcal{F}$. It employs first-order substitutions respectively most general unifiers.

**Definition 2.2 (Substitution, mgu, $\Sigma$)** *A function $\sigma : \mathcal{V} \to Term(\mathcal{F}, \mathcal{V})$ is called a* substitution, *where $\sigma(x) \not\equiv x$ only for a finite subset of $\mathcal{V}$. $\sigma$ can be extended to $Term(\mathcal{F}, \mathcal{V})$ by defining $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$ for all $f \in \mathcal{F}$.*

*Let $s, t \in Term(\mathcal{F}, \mathcal{V})$. A substitution $\sigma$ is called a* most general unifier (mgu) *of $s$ and $t$ if $\sigma(s) \equiv \sigma(t)$ and for any other substitution $\varrho$ with $\varrho(s) \equiv \varrho(t)$ there is a further substitution $\nu$ with $\nu \circ \sigma = \varrho$.*
*$\Sigma$ denotes the set of all substitutions.*

**Definition 2.3 (Condensed Detachment, Ancestor, Descendant)** *Let $i \in \mathcal{F}$ be a (distinguished) binary function symbol (i.e. $\tau(i) = 2$), $i(s,t), s' \in Term(\mathcal{F}, \mathcal{V})$ and $\sigma$ the mgu of $s$ and $s'$. Then $\sigma(t)$ is derived from $i(s,t)$ and $s'$ via* condensed detachment*. In this context, the facts $i(s,t)$ and $s'$ are called the* immediate ancestors *of the* immediate descendant *$\sigma(t)$.* Ancestors *and* descendants *can be obtained by constructing the transitive closure of the immediate ancestor respectively immediate descendant relation.*

**Example 2.1** *Let $\mathcal{F} = \{e\}$, $\tau(e) = 2$, i.e., $e$ is the distinguished binary function symbol required by condensed detachment. Let further $\lambda_1 \equiv e(\underline{e(x,y)}, e(e(z,y), e(x,z)))$ and $\lambda_2 \equiv e(u,u)$ be two facts. With the mgu $\sigma = \{x \leftarrow u, y \leftarrow u\}$ (which is an abbreviation for $\sigma(x) = u$, $\sigma(y) = u$ and $\sigma(z) = z$ for all other variables $z \in \mathcal{V}$) of $\lambda_1$'s subterm $\underline{e(x,y)}$ and $\lambda_2$ we obtain $\sigma(\lambda_1) \equiv e(\sigma(\lambda_2), e(e(z,u), e(u,z)))$ and hence can derive $e(e(z,u), e(u,z))$ respectively $e(e(x,y), e(y,x))$ after renaming ("normalizing") variables.*

*The two facts participating in the application of condensed detachment do not have to be distinct. Consider $\lambda \equiv e(\underline{e(e(e(x,y),z),x)}, e(y,z))$ and a renamed counterpart of $\lambda$, namely $\lambda' \equiv e(e(e(e(x',y'),z'),x'), e(y',z'))$. Using the mgu $\sigma = \{x \leftarrow e(z',z'), y \leftarrow z', z \leftarrow z', x' \leftarrow z', y' \leftarrow z'\}$ of $\underline{e(e(e(x,y),z),x)}$ and $\lambda'$ we can derive $e(z',z')$ (respectively $e(x,x)$).*

The purpose of this inference rule is to derive facts which are valid in *one* given model provided that the facts driving the inference are also valid in that model. (See in particular [Lu70], pp. 250–277, for a comprehensive discussion of this aspect.) The logic calculi we are going to deal with determine which model to use. A fact that is valid in the model is also referred to as a *theorem* of the respective calculus. The area of interest in the study of logic calculi with condensed detachment is to find out whether an initial (finite) set of theorems—the *axioms*—together with the inference rule 'condensed detachment' allow to deduce *all* theorems, i.e., all facts that are valid in the model. This means that we want to find out whether a given axiomatization is *complete*. This form of completeness must not be confused with the completeness of a calculus. Commonly, a set of axioms determines *all* models in which the axioms are valid. The task of the calculus respectively its inference rules then consists in deriving all consequences which are also valid in these models. "Deriving all consequences" refers to the completeness of the calculus, while "valid in these models" refers to its correctness. Here, we are *given* one model. Hence, while it still makes sense to talk about the correctness of condensed detachment ("all facts inferable with condensed detachment must be valid in the given model"), the completeness issue must be shifted from the calculus to the axiomatization.

We shall verify the completeness of an axiomatization $Ax$ by showing that an axiomatization known to be complete can be deduced from $Ax$ (with the help of condensed

```
let F^A = ∅, F^P = Ax

while F^P ≠ ∅ do
{
    select λ ∈ F^P and remove it from F^P
    if ∃λ' ∈ F^A : λ' ⊲ λ then
        discard λ
    else
    {
        apply condensed detachment employing λ
        add all deduced facts to F^P
        add λ to F^A
        if λ ⊲ ax then
            stop ('PROOF FOUND')
    }
}

stop ('PROOF FAILED')
```

Figure 1: Basic Algorithm of CoDe

detachment). This naturally necessitates that there is a "starting axiomatization" that has to be shown to be complete in some other way. But this is not of our concern here. Our concern are the deduction tasks that allow to show the completeness of an axiomatization by deducing the axioms of an axiomatization whose completeness has already been established. For this purpose we shall employ the automated deduction system 'CoDe'.

## 2.2 An Automated Deduction System For Condensed Detachment

The previous subsection presented the deduction tasks we want to solve with our automated deduction system CoDe. These consist in attempting to derive (one by one) each axiom $ax$ of a complete axiomatization $Ax'$ using condensed detachment and starting from a set of axioms $Ax$ whose completeness is to be shown. In other words, we want to deduce or *prove* each $ax \in Ax'$. In this context, such an $ax$ may be called a *goal*. The pair $\mathcal{A} = (Ax, ax)$ describes a *proof problem*.

CoDe proceeds according to the very common principle depicted in algorithmic form in figure 1. The facts in $F^A$ are called *active facts* because they can participate in the generation of new facts via condensed detachment, whereas the facts in $F^P$ cannot participate and are therefore called *passive* or *potential* facts. Selecting a fact $\lambda$

from $F^P$ to become a member of $F^A$ is called "*activating* $\lambda$". But this activation respectively selection step '`select` $\lambda \in F^P$' is an indeterminism that is inherent to problems related to deduction because of the general undecidability of such problems. This indeterminism can consequently only be resolved by heuristic means. A selection heuristic $\mathcal{H}$ associates a natural number $\mathcal{H}(\lambda) \in \mathbb{N}$ with each $\lambda \in F^P$, which is referred to as "weighting $\lambda$ with $\mathcal{H}(\lambda)$". Subsequently, that $\lambda \in F^P$ with the smallest weight $\mathcal{H}(\lambda)$ is selected. Ties are broken according to the FIFO-strategy ("*first in–first out*").

A heuristic $\mathcal{H}$ has to satisfy a criterion called *fairness* in order not to prevent a possible proof because of a "biased" selection. Informally, a heuristic is called *fair* if every $\lambda \in F^P$ (or a $\lambda' \in F^P$ that subsumes $\lambda$) is selected after finite time.

$\lambda_a \lhd \lambda_b$ denotes subsumption, i.e., $\lambda_a$ is at least as general as $\lambda_b$ and can therefore play at least as important a role in the deduction process as $\lambda_b$. Subsumption is a simple form of testing implication on a syntactic level, i.e., $\lambda_a \lhd \lambda_b$ iff $\exists \sigma \in \Sigma : \sigma(\lambda_a) \equiv \lambda_b$. The subsumption test is employed in order to find out if a fact has been inferred which is equivalent to the goal or even more general. This test is made using $\lambda \in F^A$ and not $\lambda \in F^P$ for conceptual reasons, because the elements of $F^P$ are considered as not (yet) actually inferred. They are merely known to be inferable. But in practice, it stands to reason to accelerate the selection of a $\lambda \in F^P$ subsuming the goal.

The discarding of a selected $\lambda$ which is subsumed by a $\lambda' \in F^A$ is an efficiency increasing measure. It is theoretically not necessary, but very important in practice. Note that this kind of redundancy removal is called *forward subsumption*. *Backward subsumption*, where a selected $\lambda$ subsumes a $\lambda' \in F^A$ is omitted because experiments have shown that the frequency of such a situation is far too low to justify the effort.

The execution of the `while`-loop (cp. figure 1), having replaced the indeterministic step '`select` $\lambda \in F^P$' with a selection policy employing a heuristic $\mathcal{H}$, can be taken down by recording the sequence $\mathcal{S} \equiv \lambda_1; \ldots; \lambda_n$ of selected facts. We omit those selected facts which were discarded immediately after their selection since they do not play any role in the deduction process. Therefore, there can be no $i < j$ with $\lambda_i \lhd \lambda_j$. The *(search) protocol* or *(search) sequence* $\mathcal{S}$ represents the *search* for $\lambda_n$ conducted by CoDE using the *search guiding heuristic* $\mathcal{H}$. Clearly, each $\lambda_i$ occurring in $\mathcal{S}$ is either an axiom, i.e., $\lambda_i \in Ax$, or it is an immediate descendant of some $\lambda_{j_1}$ and $\lambda_{j_2}$ also in $\mathcal{S}$, where $j_1, j_2 < i$. ($j_1$ and $j_2$ do not necessarily have to be distinct.)

If the algorithm of figure 1 terminates, then the sequence $\mathcal{S} \equiv \lambda_1; \ldots; \lambda_n$ is a *successful search* and it *contains* the proof (the deduction) of $\lambda_n$ with $\lambda_n \lhd ax$. We say 'contains', because $\mathcal{S}$ may—and in practice always will—contain $\lambda_i$ that are no ancestors of $\lambda_n$ and hence are redundant with respect to the deduction of $\lambda_n$. The *proof* $\mathcal{P}$ of $\lambda_n$ relating to $\mathcal{S}$ can be extracted[1] from $\mathcal{S}$ by omitting all those $\lambda_i$ which are *not* members of the following set

$$P = \Big\{\lambda \in \{\lambda_1, \ldots .\lambda_n\} \mid \lambda \text{ is an ancestor of } \lambda_n\Big\} \cup \Big\{\lambda_n\Big\}.$$

[1]To this end we implicitly associate a justification $\mathcal{J}_i$ with each $\lambda_i$ occurring in $\mathcal{S}$ explaining whether $\lambda_i$ is an axiom or which other facts are its immediate ancestors.

The proof $\mathcal{P} \equiv \lambda_1^+; \ldots; \lambda_m^+$ ($m \leq n$, $\lambda_m^+ \equiv \lambda_n$) is a "stripped-down" version of $\mathcal{S}$ where all $\lambda_i \notin P$ have been removed. $P$ may also be viewed as the closure of $\{\lambda_n\}$ under the ancestor relation. The members of $P$ will be referred to as *positive facts*, whereas the members of the complementary set

$$N = \{\lambda_1, \ldots, \lambda_n\} \setminus P$$

are called *negative facts* which are *irrelevant* regarding $\mathcal{P}$. Be aware that the classification of facts into positive and negative ones refers to the particular proof $\mathcal{P}$ extracted from $\mathcal{S}$. This is why it here makes sense to talk about *the* proof $\mathcal{P}$ although in general several proofs of $\lambda_n$ exist. Note that $\mathcal{S}$ and therefore $\mathcal{P}$, $P$ and $N$ depend on the proof problem $\mathcal{A}$ and the search guiding heuristic $\mathcal{H}$ in hand. But we shall not make this dependency explicit unless it is necessary to avoid confusion.

The *efficiency of the search* is reflected by the size of $N$. An "ideal" heuristic $\mathcal{H}$ produces a search $\mathcal{S}$ with $N = \emptyset$. A "bad" heuristic $\mathcal{H}'$, however, produces a search $\mathcal{S}'$ where the size of the associated set $N'$ is enormous (compared to $P'$). But note that—in general—the size of $P$ respectively the *length $m$ of the proof* $\mathcal{P} \equiv \lambda_1^+; \ldots; \lambda_m^+$ alone gives no hints as to the efficiency of the related search.

Consequently, the search guiding heuristic $\mathcal{H}$ plays an important role. Its ability to avoid redundancies, i.e., the selection of facts that do not contribute to finding the proof eventually found, is not only crucial for efficiency in terms of computation time. Apart from that, the sets $F^A$ and above all $F^P$ are responsible for filling quickly several MB of computer memory. While the size of $F^A$ is equal to the length $n$ of the search sequence (at any state of the search), the size of the set $F^P$ has the upper bound $|Ax| + n^2 - n$ already suggesting that $F^P$ will grow much faster than $F^A$ (which it actually does). Therefore a reduction of the length of the search sequence is also welcomed regarding memory management.

Problems related to condensed detachment have been the object of intensive studies to evaluate the usefulness of heuristics and to test automated deduction systems. In [MW92] various "conventional" heuristics have been tested using OTTER. We shall also tackle problems listed in [MW92] (see appendix A) with a (conventional) basic heuristic that will be introduced in section 3. But the main topic of this report is the design and (experimental) evaluation of heuristics that make use of proofs found in the past. We shall see that such heuristics not only account for substantial speed-ups (w.r.t. conventional ones), but also make it possible to find proofs where none of the conventional heuristics succeeded (in reasonable time). Both the results produced by CoDE using its basic heuristic and those attained by OTTER (as reported in [MW92]) will serve as a point of reference.

# 3 The Basic Heuristic

Although the topic of this report is the demonstration of heuristics that make use of previously gained proof experience, this section is entirely devoted to a heuristic that does not belong to that category. There are two reasons for this. Firstly, at least for a start, a heuristic which is not based on previous proof experience is indispensable to establish a basis of proof experiences. Secondly, this heuristic serves as the foundation of those heuristics based on past experience.

Recall that a search guiding heuristic associates a weight with each fact and selects the one with the smallest weight. Since the majority of proof tasks consists in proving facts that have a comparatively small number of function symbols, our basic heuristic is designed to focus on "small" facts. (Heuristics centered on this observation are very common. See, for instance, [Hu80] or [MW92].) Instead of simply counting the number of function symbols and variables, we compute a weighted sum. To this end, function symbols are weighted with 2 and variables with 1. The function $w$ computing the weighted sum is defined as follows.

**Definition 3.1 (Weighted Sum)** *Let $\lambda$ be a fact (a term). The* weighted sum $w(\lambda)$ *is defined by*

$$w(\lambda) = \begin{cases} 1, & \text{if } \lambda \in \mathcal{V} \\ 2 + \sum_{i=1}^{n} w(t_i), & \text{if } \lambda \equiv f(t_1, \ldots, t_n),\ f \in \mathcal{F},\ \tau(f) = n \end{cases}$$

By giving a smaller weight to variables we cause a bias towards the selection of facts with more variables rather than function symbols. This makes sense because such facts tend to have more "deductive power", i.e., they are often more general and hence more useful during the deduction process.

We soon recognized that this simple heuristic $w$ had—quite understandably—apparent limitations. But we also discovered that the *level* of the proofs found was rather low. The subsequent definition defines this notion (cp. [MW92]).

**Definition 3.2 (Level of a Fact/Proof)** *Given a proof $\mathcal{P} \equiv \lambda_1; \ldots; \lambda_n$ and a fact $\lambda \in P = \{\lambda_1, \ldots, \lambda_n\}$, the* level *(or* depth*) $\delta(\lambda)$ of $\lambda$ is defined by*

$$\delta(\lambda) = \begin{cases} 0, & \text{if } \lambda \text{ is an axiom} \\ \max\{\delta(\lambda_i), \delta(\lambda_j)\} + 1, & \text{if } \lambda_i \text{ and } \lambda_j \text{ are the immediate ancestors of } \lambda \end{cases}$$

*The level (or depth) $\delta(\mathcal{P})$ of a proof $\mathcal{P}$ is defined by*

$$\delta(\mathcal{P}) = \max\{\delta(\lambda) \mid \lambda \in P\} \quad .$$

Note that $\delta(\mathcal{P}) = \delta(\lambda_n)$, if $\mathcal{P} \equiv \lambda_1; \ldots; \lambda_n$.

Having observed the prevalence of "low level proofs" the next obvious step to improve the basic heuristic was to incorporate the level of the fact $\lambda$ to be weighted, so that "deeper" facts would receive a (moderate) penalty. We chose a weighted sum of the level and the result of $w$ yielding the final basic heuristic $\varpi$. According to the considerations already outlined in subsection 2.2, we give a special treatment to a fact subsuming and hence proving the goal. This is achieved by assigning the minimal weight 0 to it which immediately leads to its activation and hence to a successful termination of the search.

Table 1: Results of the Basic Heuristic (cn01–cn06)

| Name | 0 : 1 | 1 : 3 | 1 : 2 | 2 : 3 | 1 : 1 | 4 : 3 | 3 : 2 | 2 : 1 | 3 : 1 | 4 : 1 | OTTER |
|---|---|---|---|---|---|---|---|---|---|---|---|
| cn01 | — | — | — | — | — | — | — | — | — | — | $\infty$ |
| cn02 | <1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s | 3s |
| cn03 | — | — | — | — | — | — | — | — | — | — | $\infty$ |
| cn04 | <1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s i | <1s |
| cn05 | 1.4s | 1.1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s | 14s |
| cn06 | — | — | — | — | — | — | — | 90s | 24s | 11s | 7366s |

**Definition 3.3 (Basic Heuristic)** *Let $c_\delta, c_w \in I\!N$, $\lambda$ the fact to be weighted and $\lambda_G$ the goal. The* basic heuristic *$\varpi$ weighting $\lambda$ with $\varpi(\lambda)$ is defined by*

$$\boxed{\varpi(\lambda) = \begin{cases} 0, & \textit{if } \lambda \lhd \lambda_G \\ c_\delta \cdot \delta(\lambda) + c_w \cdot w(\lambda), & \textit{otherwise} \end{cases}}$$

The design of $\varpi$ allows to examine borderline cases, namely $w$ alone ($c_\delta = 0$, $c_w \geq 1$), breadth-first search ($c_\delta \geq 1$, $c_w = 0$) or the FIFO-strategy ($c_\delta = c_w = 0$). (The exception made for facts subsuming the goal is also effective when performing breadth-first search or a search guided by $w$ only.) The additional computational effort on account of the subsumption test clearly pays off as experiments have shown.

The following subsection 3.1 presents the results we obtained by applying $\varpi$—with various configurations of $c_\delta$ and $c_w$—to the proof problems we shall also tackle with heuristics based on past proof experience. See appendix A for a complete description of these proof problems. Subsection 3.2 discusses a "by-product" we encountered when running tests with $\varpi$, namely finding *shorter* proofs *faster*.

## 3.1 Experimental Results

Tables 1–4 list the results produced by CODE using the basic heuristic $\varpi$ with various ratios $c_\delta : c_w$. The basic working method of CODE, which is written in C, was presented in subsection 2.2.

In each table, the first column lists the names of the proof problems (in compliance with appendix A). The last column displays the results of OTTER also using a basic heuristic which simply counts symbols and hence does not discriminate function symbols and variables in contrast to $\varpi$. These results are taken from [MW92]. The head of each remaining column shows the ratio '$c_\delta : c_w$' employed by CODE's basic heuristic $\varpi$. The entries of the tables list the (approximate) run times in seconds (CPU time) obtained on a SPARCstation ELC. (The run times of OTTER were obtained on a SPARCstation 1+ which is a comparable machine.) An entry '—' (CODE's columns only) denotes that no proof could be found when restricting the memory to 45 MB. An entry '$\infty$' (OTTER's

Table 2: Results of the Basic Heuristic (cn07–cn24)

| Name | 0 : 1 | 1 : 3 | 1 : 2 | 2 : 3 | 1 : 1 | 4 : 3 | 3 : 2 | 2 : 1 | 3 : 1 | 4 : 1 | OTTER |
|---|---|---|---|---|---|---|---|---|---|---|---|
| cn07 | <1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s | <1s | < 1s |
| cn08 | 9.1s | 3.1s | 2.0s | 1.8s | 1.7s | 1.5s | 1.6s | 2.2s | 2.8s | 6.2s | 60s |
| cn09 | 9.1s | 3.2s | 2.1s | 1.9s | 1.8s | 1.5s | 1.5s | 2.2s | 2.8s | 5.8s | 60s |
| cn10 | 16s | 7.4s | 6.1s | 3.6s | 3.7s | 3.1s | 3.3s | 3.5s | 6.3s | 32s | 89s |
| cn11 | 19s | 11s | 11s | 8.3s | 3.8s | 3.2s | 3.3s | 3.6s | 6.5s | 41s | 104s |
| cn12 | 19s | 11s | 11s | 8.4s | 14s | 3.7s | 3.9s | 10s | 41s | — | 105s |
| cn13 | 19s | 9.9s | 6.9s | 8.3s | 3.7s | 5.7s | 7.3s | 4.4s | 24s | — | 105s |
| cn14 | 19s | 11s | 7.1s | 8.3s | 3.8s | 5.8s | 7.5s | 4.6s | 24s | — | 109s |
| cn15 | — | — | — | — | — | — | — | — | — | — | $\infty$ |
| cn16 | 19s | 11s | 5.9s | 8.3s | 2.2s | 2.0s | 2.4s | 2.4s | 3.2s | 5.8s | 105s |
| cn17 | 18s | 9.7s | 6.0s | 8.0s | 2.2s | 2.8s | 3.8s | 4.2s | 20s | — | 104s |
| cn18 | 19s | 11s | 7.0s | 8.4s | 3.5s | 5.0s | 4.0s | 4.5s | 24s | — | 106s |
| cn19 | 84s | — | — | — | — | — | — | — | — | — | 1021s |
| cn20 | 42s | 24s | 13s | 13s | 3.3s | 5.0s | 5.1s | 7.3s | 24s | — | 260s |
| cn21 | 91s | — | — | — | — | — | — | — | — | — | 1195s |
| cn22 | — | — | — | — | — | — | — | — | — | — | $\infty$ |
| cn23 | — | — | — | — | — | — | — | — | — | — | $\infty$ |
| cn24 | — | — | — | — | — | — | — | — | — | — | $\infty$ |

column only) means that no proof was found within 4 hours. Naturally, OTTER has to control its memory usage in order to be able to cope with a limited memory (12 MB in OTTER's case). CODE also disposes of ways to control its memory usage which, just like OTTER's, can cause incompleteness. (Basically, "controlling memory usage" means discarding certain facts respectively clauses.) But we did not have CODE use them here, since the results attained without them are satisfactory. In particular, the problems for which the tables do not list any run time stay out of reach (within several hours) even when enabling the memory control features of CODE (using $\varpi$).

It must be emphasized that CODE is an experimental program whose core was developed in a couple of weeks as opposed to the renowned OTTER which has been improved over several years. CODE does not use sophisticated indexing techniques. These are crucial for efficient (forward) subsumption which is exhaustively needed in connection with condensed detachment. CODE might be faster at very early stages of the search (if at all) because of a specialized implementation of condensed detachment, which OTTER "simulates" with hyper-resolution. But efficiency increasing techniques like indexing cause OTTER to surpass CODE (in terms of inferences per second) after these early stages. Consequently, faster run times of CODE can only stem from heuristics that allow for a more efficient search in terms of selecting less facts that are irrelevant with respect to the proof eventually found.

Problem cn06 (table 1) is a "paradigm" for the benefits of taking into account the level

Table 3: Results of the Basic Heuristic (cn25–cn33)

| Name | 0 : 1 | 1 : 3 | 1 : 2 | 2 : 3 | 1 : 1 | 4 : 3 | 3 : 2 | 2 : 1 | 3 : 1 | 4 : 1 | OTTER |
|---|---|---|---|---|---|---|---|---|---|---|---|
| cn25 | — | — | — | — | — | — | 33s | 18s | 15s | 11s | $\infty$ |
| cn26 | — | 1.2s | <1s | <1s | <1s | <1s | 1.3s | 1.7s | <1s | <1s | 4s |
| cn27 | 4.7s | 1.1s | <1s | <1s | <1s | <1s | 1.3s | 1.7s | <1s | <1s | 3s |
| cn28 | — | — | — | — | — | 73s | 50s | 15s | 11s | 36s | 6038s |
| cn29 | 197s | 47s | 40s | 35s | 82s | — | — | — | — | — | 622s |
| cn30 | 154s | 23s | 11s | 5.2s | 1.7s | <1s | 1.0s | 1.3s | 3.0s | 1.6s | 161s |
| cn31 | — | — | — | — | 94s | 25s | 16s | 9.6s | 17s | — | 5611s |
| cn32 | — | 89s | 45s | 46s | 86s | — | — | — | — | — | 753s |
| cn33 | 208s | 58s | 15s | 11s | 3.4s | 1.4s | 2.8s | 2.8s | <1s | 1.0s | 239s |

Table 4: Results of the Basic Heuristic (mv55–mv62)

| Name | 0 : 1 | 1 : 3 | 1 : 2 | 2 : 3 | 1 : 1 | 4 : 3 | 3 : 2 | 2 : 1 | 3 : 1 | 4 : 1 | OTTER |
|---|---|---|---|---|---|---|---|---|---|---|---|
| mv55 | — | — | — | — | — | — | — | — | — | — | $\infty$ |
| mv56 | 2.2s | 2.2s | 2.3s | 2.6s | 1.4s | <1s | <1s | <1s | <1s | <1s | 3s |
| mv57 | — | — | 176s | 101s | 14s | 5.8s | 2.1s | 1.1s | <1s | <1s | 4475s |
| mv58 | 2.2s | 2.1s | 2.4s | 2.7s | 1.4s | <1s | <1s | <1s | <1s | <1s | 3s |
| mv59 | — | — | — | 130s | 40s | 51s | 82s | 12s | 75s | 68s | $\infty$ |
| mv60 | — | — | — | — | — | — | — | — | — | — | $\infty$ |
| mv61 | 4.5s | 4.5s | 4.6s | 5.7s | 33s | 20s | 17s | 18s | 80s | 34s | 7s |
| mv62 | — | — | — | — | — | — | — | — | — | — | $\infty$ |

of a fact. There, a proof can only be found when $c_\delta : c_w$ is at least $2 : 1$. Moreover, the higher the ratio, the faster a proof is found. A similar observation can be made regarding cn25 (table 3) and mv57 (table 4). In general, $\varpi$ profits from considering the level. The only two remarkable exceptions to the "rule" are problems cn19 and cn21 (table 2). But they reveal that computing a *weighted* sum of function symbols and variables already yields a strong heuristic.

## 3.2 Finding Shorter Proofs Faster

In [Wo90] several proof problems stemming from the equivalential calculus ([Lu70], pp. 250–277) and related calculi are discussed, which in part are also dealt with in [MW92]. Apart from the main problem of finding *some* proof (using OTTER), also the problem of finding *shorter* proofs than the ones found so far is addressed. This issue is important if the proofs obtained are to be presented to a human reader who quite naturally prefers short, less complex proofs. The search guiding heuristics, however, (in general) do not support this desire. So, in [Wo90] several methods are proposed how

Table 5: Run Time and Proof Length

| Name | 0 : 1 | 1 : 1 | 2 : 1 | 3 : 1 | 4 : 1 | 5 : 1 | 6 : 1 | 7 : 1 | 8 : 1 | 9 : 1 | 10 : 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| `ec69` | 111s<br>20 | 74s<br>20 | 66s<br>8 | 16s<br>8 | 11s<br>8 | 11s<br>8 | 6s<br>8 | 3s<br>8 | 3s<br>8 | 3s<br>8 | 3s<br>8 |
| `ec79` | 100s<br>40 | 41s<br>35 | 92s<br>46 | 51s<br>43 | 6s<br>26 | 10s<br>29 | 31s<br>26 | 73s<br>26 | 30s<br>35 | 20s<br>25 | 18s<br>25 |
| `r86` | 3s<br>31 | 1s<br>13 | 0.7s<br>8 | 0.4s<br>8 | 0.3s<br>8 | 0.3s<br>8 | 0.2s<br>8 | 0.2s<br>8 | 0.2s<br>8 | 0.1s<br>8 | 0.1s<br>8 |
| `r88` | 6s<br>13 | 2s<br>13 | 1.5s<br>7 | 0.6s<br>7 | 0.3s<br>7 | 0.3s<br>7 | 0.3s<br>7 | 0.3s<br>7 | 0.3s<br>7 | 0.3s<br>7 | 0.3s<br>7 |
| `lg89` | 31s<br>25 | 15s<br>21 | 3s<br>20 | 5s<br>7 | 2s<br>7 | 1.2s<br>7 | 1s<br>7 | 0.4s<br>7 | 0.5s<br>7 | 0.4s<br>7 | 0.5s<br>7 |
| `lg90` | 107s<br>10 | 79s<br>8 | 4s<br>8 | 1s<br>8 | 1s<br>8 | 1s<br>8 | 0.4s<br>8 | 0.3s<br>8 | 0.3s<br>8 | 0.3s<br>8 | 0.3s<br>8 |
| `lg91` | 25s<br>13 | 25s<br>13 | 9s<br>5 | 3s<br>5 | 2s<br>5 | 2s<br>5 | 0.6s<br>5 | 0.6s<br>5 | 0.6s<br>5 | 0.6s<br>5 | 0.6s<br>5 |
| `rg102` | 44s<br>32 | 13s<br>23 | 8s<br>25 | 6s<br>22 | 7s<br>22 | 8s<br>24 | 7s<br>20 | 8s<br>20 | 8s<br>22 | 13s<br>22 | 15s<br>22 |

to force OTTER into seeking (shorter) alternative proofs. But such a forced quest for shorter proofs often results in much longer run times. In particular, a level-saturation run (through the level of the shortest proof known so far)[2], which seems to offer very good chances to find shorter proofs,[3] is impractical.

The heuristic $\varpi$ incorporates the level of a fact. By increasing the ratio $c_\delta : c_w$ we gradually prefer facts with a smaller level. Experimental results summarized in table 5 show that by increasing the ratio $c_\delta : c_w$ (within reasonable boundaries) we are not only able to find (significantly) shorter proofs, but also to do so *faster*. As in subsection 3.1 the head of a row lists the name of the problem (whose description can be found in appendix A.3), the columns refer to the different ratios, while the entries display run time (in seconds) and proof length. Note that [Wo90] refers to the number of (derived) proof steps as opposed to the proof length (equal to $|P|$) we use. The connection is simple: The number of derived steps is equal to the length minus the number of axioms.

Examining table 5, we discover that except for the problems `ec79` and `rg102` all other problems exhibit a "perfectly" regular behavior. By increasing the ratio $c_\delta : c_w$ both run time and proof length continuously decrease until a certain "lower bound" is reached. As problems `ec79` and `rg102` show, the length of the proofs and the run times may show some irregular behavior. Moreover, fastest and shortest proofs are not as per-

[2]A level-saturation run through level $k$ corresponds to a breadth-first search (restricted to level $k$).

[3]A proof with level $k$ can have at most the length $2^{k+1} - 1$ and must have at least the length $k + 1$. Raising $k$ by 1 means that this upper bound will roughly double, while the lower bound is also raised by 1. This suggests that the length of a proof will (probably) increase with its level.

fectly correlated as they are for the other problems. But, as a common result, table 5 demonstrates significant improvements both w.r.t. run time and proof length compared to the case where the level is not taken into account at all (ratio 0 : 1). Please note that the ratio 10 : 1 is not a "magic" boundary. Depending on the problem at hand, increasing the ratio even further can still yield shorter proofs.

We would like to emphasize the results obtained in connection with problem `ec69` which corresponds to 'Theorem 5' in [Wo90] on pages 228 and 229. The first proof presented there has 31 derived steps. A second shorter proof has 18 derived steps, but required approximately seven times as much CPU time. By raising the ratio $c_\delta : c_w$ to 10 : 1 our heuristic produced a proof using 6 derived steps ("proof length 8 minus 2 axioms"). It succeeded in finding it roughly 37 times *faster* than it found the "longest" proof (ratio 0 : 1), which is, by the way, identical to the shorter proof found by OTTER. Appendix B displays all the different proofs found for problem '`ec69`'.

Despite these successes we nevertheless want to point out that (this way of) incorporating the level into the heuristic weight does not guarantee to find shorter proofs (faster) by increasing the ratio $c_\delta : c_w$. But it seems that some problems connected with condensed detachment and logic calculi are particularly well-suited candidates to be tackled with such a heuristic. Firstly, there *are* in many cases proofs with a low level (which almost consequently are quite short). Secondly, the facts with a high weight w.r.t. the weighted sum $w$ of (function) symbols appear at even lower levels so that the lesser "penalty" due to a lower level can compensate for their higher weight w.r.t. $w$ (cp. appendix B). This entails their earlier selection (activation) during search. If, for some proof problem, no proof with these properties exists, then the beneficial outcomes of increasing the ratio $c_\delta : c_w$ observed in connection with the examples in table 5 will (probably) not occur.

Nonetheless, the simplicity and (experimentally documented) performance of heuristic $\varpi$ lead to the conclusion that it should not be ignored when addressing the issue of finding shorter proofs.

# 4 The heuristic $\varpi_{FR}$

In section 1 the principle of our way to exploit past proof experience was sketched. Instead of (deterministically) transforming a known proof $\mathcal{P}_\mathcal{A}$ of a previously solved proof problem $\mathcal{A}$ into a proof $\mathcal{P}_\mathcal{B}$ of a (new) proof problem $\mathcal{B}$, we have a search guiding heuristic use information acquired from the source proof $\mathcal{P}_\mathcal{A}$ (or even the complete search protocol $\mathcal{S}_\mathcal{A}$) in order to search more efficiently for a proof of the target $\mathcal{B}$. The advantage of this method is a suitable compromise between 'flexibility' and 'specialization'. A method for proving is called *flexible* if it is successful for a large class of (proof) problems in "acceptable" time. A method is called *specialized* if it is successful for a (very) small class of problems only, but when producing results, it does so much faster than more flexible methods. In other words, when making a flexible method more specialized, we (in general) enhance its performance with respect to a certain class of problems, but cause it to deteriorate with respect to its general applicability. We have to find a reasonable compromise between flexibility and specialization, since it does not make sense to consider the extreme cases. On the one hand, flexible methods already exist, represented by basic ("conventional") methods. On the other hand, a completely specialized method is only profitable for exactly the proof problem it has specialized in. It is hence worthless when addressing the more interesting issue of proving similar, but not identical problems.

By working information obtainable from past proof experience into the heuristic search, we do not depend on structural (syntactic) similarities to the same extent as other approaches do. Furthermore, inadequacies of the source proof concerning its applicability to the target can be handled "on the side" because of the remaining degree of flexibility without having to resort to "expensive" patching strategies. (See also section 9.)

In the sequel, we present the heuristic $\varpi_{FR}$ which complies with our principle of exploiting past proof experience. $\varpi_{FR}$ is essentially an attempt to re-enact (parts of) a given source proof.[4] This is achieved by giving smaller weights to facts derived during the attempt to solve the target problem which subsume a fact that contributed to the source proof (i.e., a positive fact). Such facts will henceforth be called *focus facts*. The inherent inflexibility of such an approach is moderated by combining it with the basic heuristic $\varpi$ (cf. section 3). Furthermore, $\varpi_{FR}$ also considers extensions of the source proof. This is accomplished by giving a reduced weight or, in other words, a lesser (weight) penalty to descendants of focus facts. Naturally, the reduction of the weight should decrease (i.e., the penalty should increase) with the distance[5] of the descendants with respect to "cornerstones", namely focus facts. Note that we do not take into account dependencies (ancestor-descendant relations) with respect to the source proof. We simply consider the set $P$ of positive facts associated with the source proof $\mathcal{P}_{\mathcal{A}}$ as a guideline by which we *may* orient ourselves (if possible).

The special treatment of focus facts and their descendants is justified considering the following: Focus facts played the role of lemmas in the source proof. Since it must be assumed that at least some of them will also be lemmas in a proof of the (similar) target problem, they themselves and their descendants should receive special attention.

Giving preference to certain facts and their descendants resembles the *set of support* (SOS) strategy used in resolution based theorem provers ([CL73]). Unlike the SOS strategy that classifies clauses as members or non-members of the SOS, we grade the credit given to a descendant which decreases with its distance from the focus of attention, namely focus facts respectively the source proof. Note that we do not rigorously exclude all other facts. We shall now explain details.

For the definition of $\varpi_{FR}$ the notions 'difference' and 'distance' *with respect to focus*

---

[4] We emphasize the expressions 'attempt' and 'parts of': It might not be possible to re-enact the source proof as a whole, because some of the positive facts may not be deducible due to a different set of axioms used for the target problem.

[5] Informally and to give the reader an intuitive idea what the notion 'distance' is supposed to denote in this context, we say that "the distance can (roughly) be measured in terms of the number of inference steps it took to generate a given descendant, starting to count when a focus fact was an immediate ancestor for the last time."

*facts* are pivotal. First, we define the *difference diff* between two facts $\lambda$ and $\lambda'$.

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

For the time being, we content ourselves with this simple definition of difference (more complicated criteria may involve, for instance, homeomorphic embedding or (full) second order matching). Note that the values 0 and 100 are somewhat arbitrary but intuitive hints of percentages, denoting 'no difference' and 'total difference', respectively. Or, expressed with opposite terms, 0 and 100 represent "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 and easier to interpret and to handle (than, for instance, computations involving unbounded values).

*diff* is used to find out whether a given fact $\lambda$ is a focus fact, i.e., if it was useful with respect to the source proof $\mathcal{P}_{\mathcal{A}}$. Let $P$ be the set of positive facts associated with $\mathcal{P}_{\mathcal{A}}$. We define

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

Hence, $\mathcal{D}(\lambda)$ returns the minimal difference between a given fact $\lambda$ (target) and the positive facts (source). If $\mathcal{D}(\lambda) = 0$ then $\lambda$ is considered as a focus fact.

The *distance* $d(\lambda)$ of a given fact $\lambda$ measures distance, roughly said, in terms of the number of inference steps separating $\lambda$ from the "nearest" ancestor which is a focus fact. It depends on the distance of the ancestors of $\lambda$ (if $\lambda$ is not an axiom) and $\mathcal{D}(\lambda)$:

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

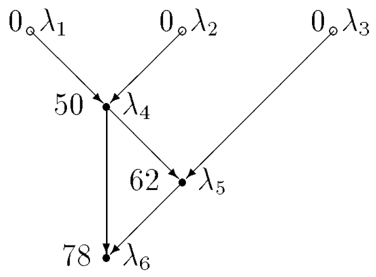
$\gamma$ computes a value that is to represent the distances of the ancestors of $\lambda$, if $\lambda$ is not an axiom. Otherwise, this value is specified by a parameter $q \in \mathbb{N}_{100}$. We chose a parameterized $\gamma$ employing a parameter $q_1 \in [0; 1]$. Depending on $q_1$, the result of $\gamma$ ranges between the minimum, the average and the maximum of the ancestors' distances.

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

Using $q_1 = 0$ or $q_1 = 1$, $\gamma$ computes the minimum or maximum, respectively. With $q_1 = 0.5$, $\gamma$ computes the (integer part of the)[6] average.

The results of $\gamma$ (respectively $q$) and $\mathcal{D}$ are combined by $\psi$ yielding $d(\lambda)$. $\psi$ 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 $\lambda$ itself is a focus fact. On the other hand, the value produced by $\psi$ should increase (reasonably) with the values obtained from $\gamma$ and $\mathcal{D}$ in order to reflect the (growing) remoteness of $\lambda$ regarding focus facts (and, in

[6]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.




The derivation dependency graph on the left displays the (immediate) ancestor relation connected with the search protocol $\mathcal{S} \equiv \lambda_1;\ \lambda_2;\ \lambda_3;\ \lambda_4;\ \lambda_5;\ \lambda_6$, assuming that $\lambda_1$ and $\lambda_2$ are the immediate ancestors of $\lambda_4$, $\lambda_3$ and $\lambda_4$ are the immediate ancestors of $\lambda_5$, and $\lambda_4$ and $\lambda_5$ are the immediate ancestors of $\lambda_6$.

Figure 2: Dependencies and distances

a way, regarding the source proof). As a matter of fact, $\gamma$ already satisfies the latter criterion. Therefore, $\psi$ is (in parts) identical to $\gamma$. It also uses a parameter $q_2 \in [0;1]$.

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

**Example 4.1** *Figure 2 depicts an exemplary ancestor-descendant relation (of an attempt to solve a target problem) in form of a* derivation-dependency graph*. Such a graph has nodes that are labeled with facts and edges pointing from immediate ancestor to immediate descendant. Next to each node we also give the distance $d(\lambda)$ w.r.t. each fact $\lambda$. We assume that $\lambda_1$, $\lambda_2$ and $\lambda_3$ are focus facts. Hence $\mathcal{D}(\lambda_i) = 0$ and therefore $d(\lambda_i) = 0$ for $i \in \{1,2,3\}$. We further assume that the remaining $\lambda_4$, $\lambda_5$ and $\lambda_6$ are not focus facts, i.e., $\mathcal{D}(\lambda_i) = 100$ for $i \in \{4,5,6\}$. The distances regarding $\lambda_4$, $\lambda_5$ and $\lambda_6$ were computed under the assumption that $q_1 = q_2 = 0.5$. The value of $q$ does not matter here. For instance, $d(\lambda_4) = \psi\Big(\gamma(d(\lambda_1), d(\lambda_2)), \mathcal{D}(\lambda_4)\Big) = \psi\Big(\gamma(0,0), 100\Big) = \psi(0,100) = 50$. The example illustrates the growing of $d(\lambda)$ as $\lambda$ becomes more and more distant with respect to (the nearest) ancestors which are focus facts.*

Before proceeding, we shall investigate some borderline cases. (The following claims can be proven by induction on the immediate ancestor relation with axioms as base case.) Naturally, when referring to "facts" we refer to (given) axioms and their descendants.

1. $q_1 = 0$ and $q_2 = 0$
   (a) $q = 0$: Then $d(\lambda) = 0$ for every fact $\lambda$.
   (b) $q = 100$: Then $d(\lambda) = 0$ if $\mathcal{D}(\lambda) = 0$ *or* $d(\lambda') = 0$ for *some* ancestor $\lambda'$ of $\lambda$. Otherwise, $d(\lambda) = 100$. This case corresponds in the main to the SOS strategy. Note that it degenerates into the preceding case if $\mathcal{D}(\lambda) = 0$ for all axioms of the target problem.
2. $q_2 = 1$: Regardless of $q_1$ or $q$ we have $d(\lambda) = \mathcal{D}(\lambda)$. This constellation can be considered as "highly specialized" in the deduction of focus facts, but unable to contribute anything beyond that.
3. $q_1 = 1$ and $q_2 = 0$

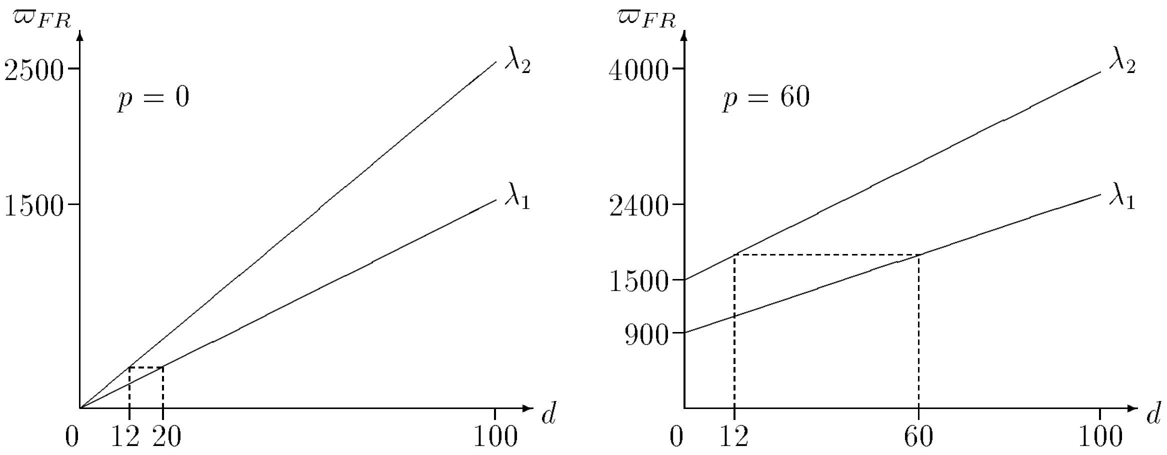



Figure 3: Behavior of $\varpi_{FR}$ depending on the distance $d$

(a) $q = 0$: Then $d(\lambda) = 0$ for every fact $\lambda$.

(b) $q = 100$: Then $d(\lambda) = \mathcal{D}(\lambda)$ if $\lambda$ is an axiom. If $\lambda$ is not an axiom, then $d(\lambda) = 0$ if all immediate ancestors $\lambda_1$ and $\lambda_2$ satisfy $d(\lambda_1) = d(\lambda_2) = 0$. Otherwise, $d(\lambda) = 100$. Note that if $\mathcal{D}(\lambda) = 0$ for all axioms of the target problem, then this case coincides with the preceding case 3a.

The remaining task consists in combining the distance $d(\lambda)$ and the value produced by the basic heuristic $\varpi$ in order to obtain a heuristic $\varpi_{FR}$ that profits from both, but also diminishes their shortcomings.

Among several sensible alternatives we picked the following:

$$\boxed{\varpi_{FR}(\lambda) = (d(\lambda) + p) \cdot \varpi(\lambda)\,, \quad p \in \mathbb{N}}$$

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

**Example 4.2** *Suppose there are two facts $\lambda_1$ and $\lambda_2$ to be weighted by $\varpi_{FR}$ whose basic weights are $\varpi(\lambda_1) = 15$ and $\varpi(\lambda_2) = 25$. Figure 3 displays the straight lines representing $\varpi_{FR}(\lambda_1)$ and $\varpi_{FR}(\lambda_2)$ depending on $d(\lambda_1)$ and $d(\lambda_2)$, respectively, for the two cases $p = 0$ and $p = 60$. Assuming $d(\lambda_2) = 12$ we compute $d(\lambda_1) = 20$ and $d(\lambda_1) = 60$ as the thresholds beyond which $\varpi_{FR}(\lambda_2)$ becomes smaller than $\varpi_{FR}(\lambda_1)$ for the cases $p = 0$ and $p = 60$, respectively (cp. figure 3). This shows that the distance $d(\lambda_1)$ of $\lambda_1$ has to be at least 20 so that $\varpi_{FR}(\lambda_1)$ can become equal or even* *greater than* $\varpi_{FR}(\lambda_2)$*, if* $d(\lambda_2) = 12$ *and* $p = 0$*, whereas this threshold is higher, namely* 60*, if* $p$ *is raised to* 60*. For* $p > 150$ *the smaller basic weight of* $\lambda_1$ *cannot be compensated for by a higher distance anymore. (To be exact, a distance* $d(\lambda_1) > 100$ *would be required, which is not possible.) This means, if* $p > 150$*, any fact* $\lambda$ *with a basic weight* $\varpi(\lambda) \leq 15$ *will always satisfy* $\varpi_{FR}(\lambda) < \varpi_{FR}(\lambda')$ *for any fact* $\lambda'$ *with* $\varpi(\lambda') \geq 25$ *regardless of* $d(\lambda)$ *and* $d(\lambda')$*.*

The results of our experiments with $\varpi_{FR}$ are documented in the subsequent section.

# 5 Experimental Results for $\varpi_{FR}$

The experiments with $\varpi_{FR}$ presented in this section are organized as follows. First of all we are going to examine the effects of various configurations of the parameters $q_1$, $q_2$ and $p$. The parameter $q$ is ignored, because the axioms of a target problem here always are focus facts which makes $q$ obsolete.

But the performance of $\varpi_{FR}$ not only depends on these parameters. To an even larger extent it depends on the set $P$ of positive facts stemming from the source proof. Given a source problem, the basic heuristic $\varpi$ provides us (in general) with a number of different proofs for different ratios $c_\delta : c_w$. We shall not examine the performance of $\varpi_{FR}$ w.r.t. all of these proofs. Since using $\varpi_{FR}$ in a non-experimental setting necessitates to pick a source proof according to some criteria concerning the source proof, we shall investigate the performance of $\varpi_{FR}$ regarding source proofs that satisfy such criteria. The criteria in question are

- the length of the source proof
- the length of the search sequence that yielded the source proof
- the level of the source proof

In this context, we consider *shortest*, *longest*, *fastest*, *slowest* proofs and proofs with a *lowest* or *highest level*. Note that "fast" and "slow" refer to the time needed to obtain a successful search and hence essentially refer to the length of the search sequence. This means that a proof is viewed as being (found) faster (slower) if the associated search sequence is shorter (longer).

Besides the parameters that directly concern $\varpi_{FR}$, there are also the parameters $c_\delta$ and $c_w$ of the basic heuristic $\varpi$ which $\varpi_{FR}$ employs. Experiments have shown that raising the ratio $c_\delta : c_w$ in order to favor facts with a lower level is (in general) not beneficial in connection with $\varpi_{FR}$. One reason for this is the "double" penalty imposed on a fact, because the distance $d$ also grows with the "relative" level (relative to an ancestor which is a focus fact). Basically, in using a ratio $c_\delta : c_w$ other than $0 : 1$ in connection with $\varpi_{FR}$, we are obviously trying to mix two strategies that do not profit from each other according to our experimental studies. We therefore keep $c_\delta : c_w$ fixed at $0 : 1$, hence not taking into account the level. This observation reveals that

Table 6: Target: `mv60`, Source: `mv59` (*fastest*, *shortest*)

| $q_1$ | $q_2$ | $p=0$ | $p=20$ | $p=40$ | $p=60$ | $p=80$ | $p=100$ | $p=120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | — | — | — |
| 0.00 | 0.50 | — | — | — | — | — | — | — |
| 0.00 | 0.75 | — | — | — | — | — | — | — |
| 0.00 | 1.00 | — | — | — | — | — | — | — |
| 0.25 | 0.25 | 35s | 87s | — | — | — | — | — |
| 0.25 | 0.50 | 69s | — | — | — | — | — | — |
| 0.25 | 0.75 | — | — | — | — | — | — | — |
| 0.50 | 0.25 | 7s | 19s | 43s | 103s | 140s | — | — |
| 0.50 | 0.50 | 19s | 41s | 68s | 113s | 121s | — | — |
| 0.50 | 0.75 | 94s | — | — | — | — | — | — |
| 0.75 | 0.25 | 5s | 10s | 17s | 37s | 53s | 76s | 97s |
| 0.75 | 0.50 | 7s | 11s | 20s | 41s | 45s | 67s | 109s |
| 0.75 | 0.75 | 32s | 71s | 98s | 141s | — | — | — |
| 1.00 | 0.25 | 4s | 7s | 11s | 17s | 41s | 47s | 53s |
| 1.00 | 0.50 | 6s | 8s | 12s | 20s | 24s | 54s | 55s |
| 1.00 | 0.75 | 26s | 33s | 61s | 99s | 115s | — | — |

(simple) algebraic combinations of heuristics do in general *not* produce a heuristic that benefits from its components in assimilating their strengths and compensating for their weaknesses. On the contrary, it is possible for such a mixture to perform worse than each of its components. More elaborate methods are called for when attempting to utilize several (different) heuristics in a profitable manner (e.g., 'TEAMWORK', cf. section 9).

For our experiments we had $q_1$ and $q_2$ range from 0 to 1 with 0.25 steps, while $p$ took on values from $\{0, 20, \ldots, 120\}$. Since $q_1$ has no influence if $q_2 = 1$ (cp. discussion of borderline cases in section 4), we only list one configuration with $q_2 = 1$. Furthermore we omit all configurations with $q_2 = 0$, because $q_2 = 0$ entails $d(\lambda) = 0$ for every fact $\lambda$ since in our experiments all axioms are focus facts (cp. borderline cases 1a and 3a), and therefore $\varpi_{FR}$ coincides with the basic heuristic $\varpi$, if $p > 0$, or with the FIFO strategy, if $p = 0$.

Consider table 6. Each row lists a different configuration of $q_1$ and $q_2$, while each column refers to a different value of $p$ displayed in the heads of the columns. The entries of the table show run times (as before in seconds, obtained on a SPARCstation ELC). The entry '—' again signifies that no proof could be found, because the memory limit (45 MB) was exceeded. Besides the target problem (`mv60`), the caption displays the source problem (`mv59`) and what kind of proof of it was used by $\varpi_{FR}$ when proving the target problem. (Recall that $\varpi_{FR}$ employs a set of positive facts when computing $\mathcal{D}$.) Here, the source proof is the one (among those found by $\varpi$, cp. table 4) which is the fastest and at the same time the shortest. (We consider two proofs as the same if their

Table 7: Overview of experiments with $\varpi_{FR}$

| Target | Source | $\varpi_{FR}$ [1] | $\varpi_{FR}$ [2] | $\varpi_{FR}$ [best] | $\varpi$ [best] | $\varpi$ [*] | OTTER |
|---|---|---|---|---|---|---|---|
| `cn19` | `cn21` | 80s | 63s | 25s | 84s | — | 423s |
| `cn21` | `cn19` | 43s | 30s | 27s | 91s | 61s | 447s |
| `cn29` | `cn28` | 12s | 28s | 4s | 35s | 19s | 257s |
| `cn29` | `cn30` | 69s | 14s | 6s | 35s | 15s | 257s |
| `cn32` | `cn31` | 14s | 15s | 10s | 45s | — | 511s |
| `cn32` | `cn33` | 25s | 16s | 9s | 45s | 28s | 511s |
| `mv60` | `mv59` | 10s | 6s | 4s | — | 20s | 2035s |
| `mv62` | `mv59` | 18s | 26s | 12s | — | 80s | 2041s |

associated sets of positive facts are equal.)

Please note that `mv60` could not be solved by $\varpi$ (see table 4), but, as table 6 reveals, it can be solved quite fast by $\varpi_{FR}$ using the aforementioned proof of `mv59`—which *was* found by $\varpi$ in approximately 12 seconds—provided that the parameters $q_1$, $q_2$ and $p$ are chosen "appropriately". As to the different configurations of these parameters, table 6 illustrates that, given a certain $q_1$ and $q_2$, increasing $p$ *always* leads to worse performance. Furthermore, it can be observed that $q_1$ should have higher values (close to 1), whereas $q_2$ appears to guarantee better performance when set to smaller values (0.25–0.50).

Appendix C provides a detailed overview of our experimentations concerning $\varpi_{FR}$ and its performance in dependence of the parameters $p$, $q_1$, $q_2$ and the type of source proof. Unfortunately (but as expected) not all target–source combinations show such a regular behavior as displayed in table 6. As a matter of fact, the choice of $p$, $q_1$ and $q_2$ offering the best performance may vary significantly depending on the target–source combination. (E.g., the tables in subsection C.3 show that the best performance can be achieved when $q_1 = 0$, a setting which is absolutely detrimental considering subsections C.7 or C.8.) Furthermore, performance also depends on the type of source proof.

Table 7 summarizes our experiments regarding $\varpi_{FR}$. Each row refers to a target–source combination. Target and source are given in columns 1 and 2, respectively. Columns 3 and 4 show the results attained with $\varpi_{FR}$ when employing two "standard" settings of the parameters $p$, $q_1$ and $q_2$, which must be seen as an attempt to find a parameter configuration that (often) allows for satisfactory results. These standard settings are [1] $p = 20$, $q_1 = 0.75$, $q_2 = 0.25$, and [2] $p = 0$, $q_1 = 1$, $q_2 = 0.5$. In both cases the fastest source proof was used. Column 5 displays the *best* result produced by $\varpi_{FR}$ (see the respective subsections of appendix C). The best result of the basic heuristic (w.r.t. the target problem) is listed in column 6. Column 7 gives the best result obtained with $\varpi$ when adding the goal of the source problem to the axiomatization of the target problem. This is the simplest form of using past "experience" and will be discussed shortly. The last column shows the *best* result obtained with OTTER (taken from [MW92]). Note that OTTER uses a variety of heuristics besides the "basic" one we used as a point of reference in subsection 3.1, some of which rigorously eliminate facts that have certain syntactic properties, and hence are unfair.

Table 7 demonstrates that at least one of the standard settings is satisfactorily close to the best one (except for the first target–source combination in row one). Furthermore, although the best result of the basic heuristic $\varpi$ is already significantly superior to the (best) results of OTTER (except for `mv60` and `mv62`), $\varpi_{FR}$ can still improve performance by factors ranging from 3 to 9. As for `mv60` and `mv62`, $\varpi_{FR}$ not only allows to attain proofs where $\varpi$ failed ($\varpi_{FR}$ being restricted to 45 MB dynamic memory just like $\varpi$), but it also finds these proofs substantially faster than OTTER. In this context we have to address the in part excellent performance of the "simplest form of using past experience" (see column $\varpi$ [$*$]). Note in particular that `mv60` and `mv62` can be proved quite fast this way. But also note that failure (for all ratios $0:1,\ldots,4:1$) may occur where even $\varpi$ does not fail. Apart from that, $\varpi_{FR}$ performs better despite the overhead caused by its increased computational effort. Hence, better performance must be attributed to a more efficient search. Furthermore, adding the goal of the source problem to the axiomatization of the target problem is only admissible if the axiomatization of source and target are (known to be) equivalent. This is not necessary regarding $\varpi_{FR}$. We shall later see that $\varpi_{FR}$ can be successfully employed where source and target axiomatization do differ (see section 8).

Please note that all source proofs were found by $\varpi$. The choice of the source problem followed the simple and straight forward rule to pick that source problem that has the same axiomatization as the target problem and is the most difficult problem solved so far (where 'difficulty' corresponds to 'run time').

A comparison of source proofs and target proofs (found by $\varpi_{FR}$ using the respective source proofs) revealed that—as expected—the majority of the positive facts from the source proof also occur in the target proof. Most of the additional steps of the target proof have focus facts or immediate descendants of focus facts as their (immediate) ancestors. Figure 4 illustrates this aspect with an example. (Appendix B explains how to interpret proof listings.)

The next section is going to present a further heuristic that can make use of past proof experience. That heuristic and $\varpi_{FR}$ can profit from each other in two ways. First, one of these two heuristics may use a proof found by the other one (which it could not find by itself) to prove further problems (for which the other heuristic might fail). Moreover, a combination of these two heuristics further enhances performance as section 8 demonstrates.

# 6 A Heuristic Based on Features

In this section we are going to present a heuristic based on so-called features that also exploits past proof experience (cp. [SF71], [SE90], [Su90]).

| | | | | |
|---|---|---|---|---|
| $\gg$ | 1 | 1 | | $i(x,i(y,x))$ |
| $\gg$ | 2 | 2 | [ 1 , 1 ] | $i(x,i(y,i(z,y)))$ |
| $\gg$ | 3 | 3 | | $i(i(n(x),n(y)),i(y,x))$ |
| $\gg$ | 4 | 4 | | $i(i(x,y),i(i(y,z),i(x,z)))$ |
| $\gg$ | 5 | 5 | [ 4 , 1 ] | $i(i(i(x,y),z),i(y,z))$ |
| $\gg$ | 6 | 6 | [ 5 , 3 ] | $i(n(x),i(x,y))$ |
| $\gg$ | 7 | 7 | [ 4 , 6 ] | $i(i(i(x,y),z),i(n(x),z))$ |
| $\gg$ | 8 | 8 | | $i(i(i(x,y),y),i(i(y,x),x))$ |
| $\gg$ | 9 | 9 | [ 5 , 8 ] | $i(x,i(i(x,y),y))$ |
| $\gg$ | 10 | 10 | [ 8 , 2 ] | $i(i(i(x,i(y,x)),z),z)$ |
| $\gg$ | 12 | 11 | [ 4 , 9 ] | $i(i(i(i(x,y),y),z),i(x,z))$ |
| $\gg$ | 14 | 12 | [ 4 , 3 ] | $i(i(i(x,y),z),i(i(n(y),n(x)),z))$ |
| $\gg$ | 15 | 13 | [ 12 , 10 ] | $i(i(n(x),n(i(y,i(z,y)))),x)$ |
| $\gg$ | 16 | 14 | [ 7 , 13 ] | $i(n(n(x)),x)$ |
| $\gg$ | 17 | 15 | [ 3 , 14 ] | $i(x,n(n(x)))$ |
| $\gg$ | 18 | 16 | [ 9 , 3 ] | $i(i(i(i(n(x),n(y)),i(y,x)),z),z)$ |
| $\gg$ | 19 | 17 | [ 4 , 4 ] | $i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))$ |
| $\gg$ | 20 | 18 | [ 11 , 17 ] | $i(i(x,y),i(i(z,x),i(z,y)))$ |
| $\gg$ | 21 | 19 | [ 18 , 15 ] | $i(i(x,y),i(x,n(n(y))))$ |
| $\gg$ | 22 | 20 | [ 17 , 16 ] | $i(i(x,i(n(y),n(z))),i(x,i(z,y)))$ |
| $\gg$ | 23 | 21 | [ 20 , 19 ] | $i(i(n(x),y),i(n(y),x))$ |
| | 61 | 22 | [ 4 , 14 ] | $i(i(x,y),i(n(n(x)),y))$ |
| | 91 | 23 | [ 17 , 11 ] | $i(i(x,i(y,z)),i(y,i(x,z)))$ |
| | 249 | 24 | [ 18 , 21 ] | $i(i(x,i(n(y),z)),i(x,i(n(z),y)))$ |
| | 250 | 25 | [ 24 , 22 ] | $i(i(x,y),i(n(y),n(x)))$ |
| | 252 | 26 | [ 11 , 25 ] | $i(x,i(n(y),n(i(x,y))))$ |
| | 253 | 27 | [ 23 , 26 ] | $i(n(x),i(y,n(i(y,x))))$ |

Figure 4: Proof of `mv62` found by $\varpi_{FR}$ when using the proof of `mv59` found by $\varpi$ with ratio $2:1$ as source proof. Focus facts are marked by $\gg$.

## 6.1 Fundamentals

Features essentially describe syntactic (structural) properties of the facts a deduction system infers.[7] In our case we have to deal with terms and their syntactic properties. Examples for features are the number of (distinct) variables of a term or the total number of function symbols occurring in a term. Generally, we view a feature as a function mapping a term into the set of integers ($\mathbb{Z}$).

**Definition 6.1 (Feature, Feature Value)** *A function* $f : Term(\mathcal{F},\mathcal{V}) \to \mathbb{Z}$ *is called a* feature. *Given any* $\lambda \in Term(\mathcal{F},\mathcal{V})$ *and a feature* $f$, $f(\lambda)$ *is called the* feature value *of* $\lambda$ *(with respect to the feature* $f$*).*

[7] If properties have to be dealt with that are not of a pure syntactic nature (e.g., the level of a fact), then it is assumed that the necessary information is nevertheless accessible.

Hence we already know two features, namely the weighted sum $w$ (cf. definition 3.1) and the level $\delta$ (cf. definition 3.2) which are already employed by $\varpi$. This suggests that they play a "special" role which will be discussed in section 7.

The basic idea behind the use of features is to find a way to discriminate positive facts from negative ones on the basis of feature values. To this end, one feature is (mostly) not enough. Therefore, we assume to have a set of $k \geq 1$ features $f_1, \ldots, f_k$ at our disposal. When searching for a proof we then attempt to "predict" if a given fact is useful or not (i.e., if it contributes to the proof finally found or not) with the help of its *feature value vector*. A feature value vector of a fact $\lambda$ is the vector $(f_1(\lambda), \ldots, f_k(\lambda))$ whose components are the feature values of $\lambda$ with respect to the features $f_1, \ldots, f_k$. Anticipating the usefulness of a fact is accomplished by a function $\rho$ which maps a feature value vector into the set $\mathbb{N}$ for instance, and gives facts thought to be useful small values and higher values to the rest, hence acting as a weighting function. Naturally, we cannot assume that $\rho$ is always right. This means that a value computed by $\rho$ merely indicates that a given fact (represented by its feature value vector) is *probably* the more useful the smaller this value is. ($\rho$ is hence not different from any other weighting function.) It is our responsibility to design and apply $\rho$ in a way that makes it the most reliable.

There are many possibilities to design $\rho : \mathbb{Z}^k \to \mathbb{N}$. For instance, $\rho$ can be a linear polynomial in the components of the feature value vector, i.e.

$$\rho\Big((f_1(\lambda), \ldots, f_k(\lambda))\Big) = \sum_{i=1}^{k} a_i \cdot f_i(\lambda) \quad .$$

The coefficients $a_1, \ldots, a_k$ are determined based on information gained from past experience. This approach has been investigated in [SF71] w.r.t. a resolution theorem prover (and also in [Re83] applying it to solve the well-known fifteen puzzle). If we want $\rho$ to be *any* function mapping $\mathbb{Z}^k$ to $\mathbb{N}$, our choice must be a connectionist network (which has been appropriately configured). This approach has been pursued in [SE90] where the components of the feature value vector are presented to the input layer of a connectionist network. The net is trained based on past experience through back propagation.

We propose a different design of $\rho$ which is comparatively simple, but nevertheless is quite successful. It is presented in the sequel, while the next section will demonstrate its capabilities in the light of experimental results.

## 6.2 The Feature-Based Heuristic $\varpi_F$

Just like [SF71] and [SE90] we want $\rho$ to make use of proofs found in the past. (This is anyway the only reasonable way to provide the necessary information as to how feature values possibly determine which fact is a positive one or a negative one.) These two approaches discern a learning phase during which coefficients respectively weights of a neural net are determined (learned), and an application phase during which the knowledge acquired in the learning phase is applied in order to solve a (new) similar problem. Past experience (e.g., a source proof) is not considered *explicitly* during application, being available *implicitly* in the form of coefficients respectively connection weights. This distinction is not as strict regarding our approach. Although there is a kind of learning phase, we also make explicit use of a source proof during application. Similar to $\varpi_{FR}$, we employ the set $P$ of positive facts associated with the chosen source proof. But unlike $\varpi_{FR}$ which centers its computations on subsumption we concentrate here on more abstract comparisons of feature values. The following definition clarifies what we mean by 'comparison'.

**Definition 6.2 (Minimal Feature Value Difference)** *Let $f_1, \ldots, f_k$ be an arbitrary but fixed set of features, $\emptyset \neq V_i \subseteq \mathbb{Z}$ for all $1 \leq i \leq k$, and $\lambda$ a fact.*

*$V_i$ is called the* (set of) permissible feature values *(w.r.t. feature $f_i$). The* minimal feature value difference *w.r.t. feature $f_i$ is defined by*

$$\Delta_i(\lambda) = \min\Big(\{|f_i(\lambda) - v| \;\mid v \in V_i\}\Big)$$

Naturally, we shall use the set of positive facts $P$ to determine $V_1, \ldots, V_k$, which is accomplished by letting

$$V_i = \{f_i(\lambda^+) \mid \lambda^+ \in P\} \quad .$$

Under these conditions it is evident that $\Delta_i(\lambda) = 0$ for all $1 \leq i \leq k$ if $\lambda \in P$. Given $\lambda \notin P$, $\Delta_i(\lambda) \neq 0$ only if there is no $\lambda^+ \in P$ with the same feature value with respect to feature $f_i$. Hence, $\Delta_i$ allows to detect "odd" feature values, i.e., feature values that do not occur among the feature values of the positive facts associated with the given source proof. In other words, the test for $\Delta_i(\lambda) = 0$ (for at least one feature $f_i$) constitutes a necessary criterion to decide whether fact $\lambda$ does or does not contribute to the source proof at hand.

From this point of view it appears to be reasonable to add the feature values of the target goal $\lambda_G$ to the set of permissible feature values, i.e., we add $f_i(\lambda_G)$ to $V_i$ from above. This way we avoid possibly penalizing facts that share feature values with the current goal. (Note that especially features that do not encode pure syntactic properties, e.g., the level, do not always make sense in connection with a goal and should therefore be disregarded when extending the $V_i$.)

Given a fact $\lambda$, its minimal feature value differences are combined in the obvious way to yield the *feature weight* $w_F(\lambda)$:

$$w_F(\lambda) = \sum_{i=1}^{k} c_i \cdot \Delta_i(\lambda)\,, \quad c_i \in \mathbb{N} \quad .$$

It would be unwise to use $w_F$ alone as a search guiding heuristic, because we would be at the mercy of the FIFO strategy regarding all facts $\lambda$ with $w_F(\lambda) = 0$. Therefore, we combine the feature weight $w_F$ with the robust basic heuristic $\varpi$ in the following way to obtain the *feature heuristic* $\varpi_F$ (which represents our version of $\rho$). Note that we coerce the selection of a fact $\lambda$ subsuming the current goal $\lambda_G$ by assigning the minimal weight 0 to it (cp. definition 3.3).

$$\boxed{\varpi_F(\lambda) = \begin{cases} 0, & \text{if } \lambda \lhd \lambda_G \\ \varpi(\lambda) + w_F(\lambda), & \text{otherwise} \end{cases}}$$

The following subsection is devoted to determining the coefficients $c_1, \ldots, c_k$ which regulate the impact of feature value differences on the feature weight $w_F$. The method for determining these coefficients is naturally based on past experience.

## 6.3 Computing Coefficients of $\varpi_F$

Recalling subsection 2.2, past experience not only provides us with information on which facts are necessary to solve a given (source) problem $\mathcal{A}$, i.e., the set of positive facts $P$. It also supplies a set of deduced, but irrelevant facts, namely the set of negative facts $N$. Henceforth we assume that $\varpi$ (respectively $\varpi_F$ with $c_1 = \cdots = c_k = 0$) was used with a certain ratio $c_\delta : c_w$ to search for a proof $\mathcal{P}$ of $\mathcal{A}$, producing a search protocol $\mathcal{S}$ from which $P$ and $N$ derive. We shall further assume that $N \neq \emptyset$ (which is practically guaranteed for non-trivial problems).

Since $w_F(\lambda^+) = 0$ for all $\lambda^+ \in P$ regardless of the coefficients $c_i$ (because $\Delta_i(\lambda^+) = 0$ for all $1 \leq i \leq k$), increasing these coefficients will only effect negative facts $\lambda$, and only those that have $\Delta_i(\lambda) \neq 0$ for at least one $i \in \{1, \ldots, k\}$. If we choose the coefficients $c_1, \ldots, c_k$ "sufficiently large", all these negative facts $\lambda$ will eventually receive a penalty $w_F(\lambda)$ that will cause them to disappear when searching for the proof $\mathcal{P}$ using $\varpi_F(\lambda) = \varpi(\lambda) + w_F(\lambda)$. But since it is not our goal to merely speed up the search for a proof already known, we have to be careful about raising coefficients. In order to ensure a good chance for $\varpi_F$ to be successfully applicable to a similar problem $\mathcal{B}$ while at the same time reducing the number of negative facts (w.r.t. the proof of $\mathcal{B}$ finally found) $\varpi_F$ admits when searching for a proof of $\mathcal{B}$, we have to use a moderate and judicious way to set the coefficients.

Therefore, considering the source problem $\mathcal{A}$, we content ourselves with raising coefficients until "satisfactorily many" negative facts have received a "sufficiently large" penalty due to $w_F$, while at the same time the increase in the total weight due to $\varpi_F$ should be kept as low as possible. In other words, we want a certain percentage of the negative facts $\lambda \in N$ to have a weight $\varpi_F(\lambda)$ that *only just suffices* to make them disappear from the search. This way, they will not (all) be completely out of reach should a proof for a similar problem require them. In the following, we shall make these ideas more precise.

A negative fact $\lambda \in N$ will definitely disappear from the search if its weight $\varpi_F(\lambda)$ is bigger than the maximal weight $w^+_{max}$ of all positive facts, where

$$w^+_{max} = \max(\{\, \varpi(\lambda^+) \mid \lambda^+ \in P \,\}) \quad .$$

(Note again that $\varpi(\lambda^+) = \varpi_F(\lambda^+)$ for all $\lambda^+ \in P$.) We say that such a negative fact is *edged out*. Only the following set of negative facts $E \subseteq N$ can be edged out:

$$E = \{\, \lambda \in N \mid \exists 1 \leq i \leq k : \Delta_i(\lambda) \neq 0 \,\} \quad .$$

The negative facts $\lambda$ whose weight $\varpi_F(\lambda)$ is affected by increasing $c_i$ are collected in the sets

$$\mathcal{N}_i = \{\, \lambda \in N \mid \Delta_i(\lambda) \neq 0 \,\}\,, \quad 1 \leq i \leq k\,.$$

(Hence, $E = \bigcup_{i=1}^{k} \mathcal{N}_i$. Note that the $\mathcal{N}_i$ are not necessarily a partition of $E$.) Increasing $c_i$ by $1 \leq \varepsilon_i \in \mathbb{N}$ will lead—according to the experience represented by ($P$ and) $N$—to raising the weight of negative facts by at most $\varepsilon_i \cdot d_i^{max}$, where $d_i^{max}$ is the maximal feature value difference regarding feature $f_i$, i.e.,

$$d_i^{max} = \begin{cases} \max(\{\Delta_i(\lambda) \mid \lambda \in \mathcal{N}_i\}), & \mathcal{N}_i \neq \emptyset \\ 0, & \mathcal{N}_i = \emptyset \end{cases}$$

The negative facts $\mathcal{E}_i(\varepsilon_i)$ that are edged out on account of raising $c_i$ by $\varepsilon_i$ can be determined as follows. (To this end we have to make $\varpi_F$'s dependence on $c_1, \ldots, c_k$ explicit to facilitate the notation.)

$$\begin{aligned} \mathcal{E}_i(\varepsilon) = \{\, \lambda \in \mathcal{N}_i \mid \; & \varpi_F(\lambda; c_1, \ldots, c_k) \leq w^+_{max} \quad \wedge \\ & \varpi_F(\lambda; c_1, \ldots, c_{i-1}, c_i + \varepsilon, c_{i+1}, \ldots, c_k) > w^+_{max} \,\} \end{aligned}$$

Since our goal is to edge out as many negative facts as possible while at the same time keeping the increase of their weights low,

$$a_i = \begin{cases} \frac{|\mathcal{E}_i(\varepsilon_i)|}{\varepsilon_i \cdot d_i^{max}}, & \varepsilon_i \cdot d_i^{max} \geq 1 \\ -\infty, & \varepsilon_i \cdot d_i^{max} = 0 \end{cases}$$

measures the degree to which we achieved this trade off. That $c_i$ which belongs to the maximal $a_i$ is then to be raised by $\varepsilon_i$. Since it is desirable to increase coefficients in a step-by-step manner, always looking for a raise that edges out many negative facts, but increases their weights the least, $\varepsilon_i$ should be minimal, i.e., there is no $\varepsilon < \varepsilon_i$ with $\mathcal{E}_i(\varepsilon) \neq \emptyset$.

Computing the $a_i$ and raising the respective $c_i$ is iterated until all $\lambda \in E$ have been edged out or a satisfactory percentage $e \in \mathbb{N}_{100}$ of *all* negative facts has been edged out.

So far we take into account all features when computing coefficients. It makes sense to exclude a feature $f_i$ if the associated set $\mathcal{N}_i$ is rather small, i.e., by increasing $c_i$ we can only expect to get rid of a (relatively) small number of negative facts. Hence we risk to edge out negative facts that might be useful for proving similar problems, but we gain little. Therefore, we refine the procedure for determining the coefficients by enabling it to exclude those features $f_i$ whose associated sets $\mathcal{N}_i$ only account for a percentage of all negative facts that is below a given threshold $n_i \in \mathbb{N}_{100}$, i.e.,

$$100 \cdot \frac{|\mathcal{N}_i|}{|N|} < n_i \quad .$$

*input*: $P$ , $N$ , $f_1, \ldots, f_k$

compute $E$
compute $w^+_{max}$
compute $\mathcal{N}_i$ for all $1 \leq i \leq k$
determine $I$
compute $d_i^{max}$ for all $i \in I$
$m := 0$ 'number of negative facts edged out so far'
$c_1 := \cdots := c_k := 0$

while $m < |E| \quad \wedge \quad \frac{m}{|N|} \cdot 100 < e$ do
{
    compute minimal $\varepsilon_i$ for each $i \in I$
    determine $\mathcal{E}_i(\varepsilon_i)$ for each $i \in I$
    compute $a_i$ for each $i \in I$
    $a_j = \max(\{\, a_i \mid i \in I \})$
    $c_j := c_j + \varepsilon_j$
    $m := m + |\mathcal{E}_j(\varepsilon_j)|$
}

*output*: $c_1, \ldots, c_k$

Figure 5: Algorithm FC for computing coefficients of $\varpi_F$

This can be accomplished by letting $I = \{\, i \mid 100 \cdot \frac{|\mathcal{N}_i|}{|N|} \geq n_i \,\}$ and then considering only those features $f_i$ where $i \in I$ (instead of $i \in \{1, \ldots, k\}$).

Figure 5 summarizes the procedure just described in algorithmic form.

Before concluding this section and continuing with section 7, which documents our experimental results regarding $\varpi_F$, we sketch the 13 features $f_1, \ldots, f_{13}$ CoDe currently has at its disposal.

- $f_1$ is equal to the weighted sum $w$ (cp. definition 3.1).
- $f_2$ computes the number of distinct variables, e.g., $f_2(i(x, i(y, x))) = 2$.
- $f_3$ and $f_4$ compute the number of occurrences of an associated function symbol. (Note that $|\mathcal{F}| \leq 2$ for all problems considered so that two such features suffice; if $|\mathcal{F}| = 1$ then feature $f_4$ is not used.) *Example*: Suppose that the function symbol $i$ is associated with $f_3$ and $n$ with $f_4$. Then for a fact $\lambda \equiv i(x, n(n(x)))$ we obtain $f_3(\lambda) = 1$ and $f_4(\lambda) = 2$.
- $f_5$ and $f_6$ compute the maximal nesting of an associated function symbol, where the maximal nesting is the maximal number of consecutive occurrences of the

respective function symbol on the branches of the fact (term) viewed as tree. (The annotation concerning features $f_3$ and $f_4$ is correspondingly valid here.)

- $f_7$ is equal to the level $\delta$ (cp. definition 3.2).
- $f_8$ computes the difference of the weighted sum of the first and second argument of a fact (term) that has a binary function symbol at top level, i.e., $f_8(e(t_1,t_2)) = w(t_1) - w(t_2)$. If there is no binary function symbol at top level, $f_8$ returns 0.
- $f_9$ and $f_{10}$ compute the minimum respectively maximum of $w(t_1) - w(t_2)$ for *all* occurrences of a subterm $e(t_1,t_2)$ in the fact, where $e$ is some binary function symbol in $\mathcal{F}$.
- $f_{11}$ counts the number of occurrences of a term of the form $e(\xi,\xi)$, where $\xi \in \mathcal{V}$ and $e \in \mathcal{F}$.
- $f_{12}$ and $f_{13}$ return the weighted sum of the first and second argument of the fact, respectively, if there is a binary function symbol at top level; otherwise, both return the weighted sum of the whole fact (like feature $f_1$).

Naturally, the choice of features has a strong influence on what can be achieved with $\varpi_F$. The main demand on the features is to be distinctive with respect to positive and negative facts, i.e., there should not be too many negative facts for which there is no feature that allows to identify them as such. In other words, for most of the negative facts $\lambda$ there should be at least one feature $f_i$ with $f_i(\lambda) \notin V_i$. The above features suffice on that score for the problems considered here. For different problem sets and in particular in connection with different calculi extensions and modifications might be called for. (Note that an "abundance" of features is not harmful since ineffective features can be excluded by setting the respective $c_i = 0$.)

# 7 Experimental results for $\varpi_F$

In this section we present our experimental studies regarding $\varpi_F$. In order to facilitate their representation and to avoid getting lost in a multitude of possible parameter configurations, we propose the following procedure. We compute the coefficients $c_1, \ldots, c_{13}$ using algorithm FC depicted in figure 5. Consequently, we have to deal with the parameter $e$ determining the percentage of negative facts we wish to edge out and the percentages $n_i$ of negative facts that have to be in each $\mathcal{N}_i$ in order for feature $f_i$ to be taken into account (cp. section 6). By not distinguishing between the $n_i$ we reduce the number of parameters of algorithm FC to two, namely $e$ and $n = n_1 = \cdots = n_{13}$. As for the parameters $c_\delta$ and $c_w$, we shall use the same ratio when tackling the target problem as we used when proving the source problem (which provides us with the set of positive and negative facts required by algorithm FC and $\varpi_F$).

Features $f_1$ (i.e., $w$) and $f_7$ (i.e., $\delta$) are ignored by always assigning 0 to $c_1$ and $c_7$, because, in a way, both $f_1$ and $f_7$ are already taken care of by $\varpi$. But note that $\varpi$

Table 8: Using proof of `cn06` found with ratio 4 : 1 for proving `cn03`; the arrows signify that the results due to the respective parameter configuration of $e$ (rows) and $n$ (columns) are identical to the one pointed to by the arrow, because algorithm FC produced the same coefficients.

| | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
|---|---|---|---|---|---|---|---|---|
| 40 | 45% — | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ |
| 50 | 59% — | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ |
| 60 | 60% — | ⟵ | 64% 54s | ⟵ | ⟵ | ⟵ | ⟵ | 64% 46s |
| 70 | 70% 53s | 70% 48s | 71% 37s | 71% 33s | ⟵ | ⟵ | ⟵ | 70% 33s |
| 80 | 80% 21s | 80% 21s | 81% 15s | 80% 18s | ⟵ | ⟵ | ⟵ | 74% 20s |
| 90 | 89% — | 89% 48s | 88% 57s | 86% 6s | ⟵ | 85% 7s | ⟵ | ↑ |

uses so-to-speak "absolute" feature values, whereas $\varpi_F$ utilizes "relative" feature values resulting from the comparison with permissible feature values. Section 3 has motivated why $w$ ($f_1$) and $\delta$ ($f_7$) are used the way they are used by $\varpi$. Employing them again in connection with $\varpi_F$ does not seem reasonable under these circumstances. We would be imposing a "double" penalty again because of mixing two heuristics (again). On the other hand, using the absolute feature values of all remaining features is extremely questionable, since there seems to be no (apparent) reason to penalize a fact for, say, having a first *or* second argument at top level with a high weight due to $w$ *in addition* to its total weight. In other words, the feature values produced by the remaining features do—in contrast to features $f_1$ and $f_7$—not correlate to the usefulness of a fact sufficiently enough to justify the use of absolute values. Nonetheless, $w$ and $\delta$ *are* features and are therefore included into the set of features. Different approaches to utilizing features may favor their homogeneous use.

We shall demonstrate with the help of target `cn03` and source `cn06` how the parameters $e$ and $n$ influence the performance of $\varpi_F$ for all three different source proofs (and hence different sets of positive and negative facts) found by $\varpi$ for `cn06` at ratios 2 : 1, 3 : 1 and 4 : 1 (cp. table 1). Table 8 shows the results obtained when making use of the proof of `cn06` found by $\varpi$ with ratio 4 : 1, choosing $e \in \{40, 50, \ldots, 90\}$ (rows) and $n \in \{0, 10, \ldots, 70\}$ (columns). The corresponding tables for ratios 2 : 1 and 3 : 1 can be found in appendix D. The entries in table 8 again display run times in seconds obtained on a SPARCstation ELC, '—' denoting failure due to the 45 MB memory restriction. Furthermore, they display the actual percentage of negative facts edged out, since the percentage $e$ that is to be only just reached possibly cannot be reached[8] or may be significantly exceeded[9]. In particular when too many features are excluded due to increasing $n$, algorithm FC has little choice as to which coefficients to raise. The arrows indicate that the respective parameter configuration for $e$ and $n$ yields the same feature coefficients as the one pointed to by the arrow. An example should clarify

[8] $|\bigcup_{i \in I} \mathcal{N}_i| / |N|$ may be smaller than $e$.

[9] The number $m$ of negative facts edged out is increased by $|\mathcal{E}(\varepsilon_i)|$ in each iteration of algorithm FC which might cause it to "overshoot the mark".

Table 9: Overview of Experiments with $\varpi_F$

| Target | Source | 80, 10 | 80, 40 | 90, 40 | $\varpi$ [best] | $\varpi_{FR}$ [best] | OTTER |
|---|---|---|---|---|---|---|---|
| `cn03` | `cn06` | 21s | 18s | 6s | — | — | 3657s |
| `cn19` | `cn21` | — | — | ⟵ | 84s | 25s | 423s |
| `cn21` | `cn19` | 23s | 28s | ⟵ | 91s | 27s | 447s |
| `cn29` | `cn28` | 20s | 25s | 58s | 35s | 4s | 257s |
| `cn29` | `cn30` | 5s | 9s | ⟵ | 35s | 6s | 257s |
| `cn32` | `cn31` | 14s | 64s | — | 45s | 10s | 511s |
| `cn32` | `cn33` | 33s | 32s | ⟵ | 45s | 9s | 511s |
| `mv60` | `mv59` | 24s | 73s | ⟵ | — | 4s | 2035s |
| `mv62` | `mv59` | 40s | 86s | ⟵ | — | 12s | 2041s |

how to interpret table 8. Setting the parameters $e$ and $n$ to 80 and 10, respectively, (fifth row, second column) algorithm FC produces the coefficients 0, 0, 0, 0, 3, 5, 0, 2, 3, 1, 4, 0, 1 (not shown by table 8) and this results in actually edging out 80% of the negative facts (w.r.t. the search for a proof of `cn06` using $\varpi_F$ with ratio $4:1$ and $P$ from the "original" proof of `cn06` found by $\varpi$ with ratio $4:1$). When applying $\varpi_F$ with these coefficients (and $P$ obtained from the "original" proof) `cn03` could be proved within 21s. (Note that OTTER's best result regarding problem `cn03` is 3657s.) The computation time spent by algorithm FC is negligible (less than a second).

Table 9 gives an overview of our experiments with $\varpi_F$. Based on the experiments summarized by table 8 and the tables of appendix D we picked three parameter settings concerning $e$ and $n$ displayed in the heads of columns three through five. These columns show the run time obtained by $\varpi_F$ using the coefficients produced by algorithm FC (with the respective setting of $e$ and $n$) when proving the target shown in the first column by employing the source given in the second column. In case several source proofs are provided by $\varpi$ (cp. tables 1–4) we choose the fastest one. The last three columns list the *best* results attained with $\varpi$, $\varpi_{FR}$ and OTTER as a point of reference (cp. table 7).

The main finding conveyed by table 9 is the appropriateness of the parameter setting $e = 80$ and $n = 10$ (third column). Raising $e$ or $n$ causes performance deterioration for all problems considered except one, namely proving `cn03` with the help of `cn06`. The reason why proving `cn03` profits from increasing $e$ is a strong similarity between the two corresponding proofs with respect to the effective features, i.e., those features $f_i$ where $c_i \neq 0$. If this similarity is not that strong, raising $e$ too much can result in precluding facts that are necessary for a proof by assigning too big a weight penalty ($w_F$) to them. In other words, we risk to "pull the strings too much".

Section 5 and this section have demonstrated the capabilities of two different heuristics $\varpi_{FR}$ and $\varpi_F$ that exploit past proof experience. The subsequent section shows that $\varpi_{FR}$ and $\varpi_F$ can be combined very conveniently, producing a heuristic that allows to solve problems that were out of reach so far.

Table 10: Experiments with $\varpi_{FR\&F}$

| Target | Source | $\varpi$ [best] | $\varpi_{FR}$ | $\varpi_F$ | $\varpi_{FR\&F}$ | OTTER |
|---|---|---|---|---|---|---|
| `cn15` | `cn03` | — | 27s | — | 7.4s | $\infty$ |
| `cn22` | `cn15` | — | — | — | 475s | $\infty$ |
| `cn23` | `cn15` | — | — | — | — | $\infty$ |
| `cn23` | `cn22` | — | 1.1s | — | 1.4s | $\infty$ |

# 8 Combining $\varpi_{FR}$ and $\varpi_F$

Sections 4 and 6 have presented two heuristics $\varpi_{FR}$ and $\varpi_F$ that make use of past experience. In this section we shall demonstrate that a combination of $\varpi_{FR}$ and $\varpi_F$ yields an even more powerful heuristic $\varpi_{FR\&F}$ which allows to solve some of the problems that eluded both $\varpi_{FR}$ and $\varpi_F$ (and of course $\varpi$).

The basic idea behind $\varpi_{FR}$ is to multiply the weight of a fact $\lambda$ computed by the basic heuristic $\varpi$ with a multiplier determined by the distance $d$ of $\lambda$ w.r.t. the source proof. The feature-based heuristic $\varpi_F$ is designed to add a weight penalty depending on feature value differences. Both heuristics can be combined quite naturally, giving heuristic $\varpi_{FR\&F}$:

$$\boxed{\varpi_{FR\&F}(\lambda) = (d(\lambda) + p) \cdot \varpi_F(\lambda)\,, \quad p \in \mathbb{N}}$$

Table 10 summarizes our experiments with $\varpi_{FR\&F}$ regarding problems `cn15`, `cn22` and `cn23`. Note that none of these problems can be proved by OTTER with the heuristics presented in [MW92]. ([MW92] reports on OTTER solving these problems with "specialized strategies" without further information.) Note also that CODE could not solve these problems so far, and that there is a kind of interaction between $\varpi_F$ and $\varpi_{FR}$, since $\varpi_F$ allows to prove `cn03` using `cn06` (which $\varpi_{FR}$ cannot accomplish, see table 9)[10], while $\varpi_{FR}$ can prove `cn15` using the proof of `cn03` $\varpi_F$ found (where $\varpi_F$ fails, see table 10).

The first two columns show target and source problem. Columns three through five list the best results of $\varpi$, and the results of $\varpi_{FR}$ and $\varpi_F$ with parameter settings discussed below. Column four displays the results of $\varpi_{FR\&F}$ with parameter settings also discussed below, while the last column gives OTTER's (best) results. An entry '$\infty$' (OTTER's column only) signifies that no proof could be found within 4 hours. The entry '—' means that no proof could be found when restricting the number of activated facts (i.e., $|F^A|$, cp. section 2) to 1500 (CODE only).[11] We had to lift the rigid memory

[10] $\varpi_{FR\&F}$ fails too, which suggests that the similarity between `cn03` and `cn06` is restricted to feature values. The attempt to re-enact `cn06` in order to prove `cn03` is detrimental.

[11] Since CODE does not employ indexing techniques, its inference rate decreases dramatically once the search has reached this stage.

restriction this time and enable CoDE's memory control mechanism which works as follows: There is a memory *quota* of 30 MB and a memory *limit* of 45 MB. If the limit is exceeded, potential facts are deleted until the quota is reached, starting with those potential facts that have the highest weight. Note that especially for harder problems the effort spent on subsumption tests requires a huge percentage of overall CPU time, so that OTTER with its indexing techniques would clearly outperform CoDE if CoDE were using OTTER's heuristic.

Based on the experiments conducted with $\varpi_{FR}$ and $\varpi_F$ (see sections 5 and 7) the following parameter settings and source proofs were chosen. The proof $\mathcal{P}_{03}$ of `cn03` employed by $\varpi_{FR\&F}$ to prove `cn15` is the one found by $\varpi_F$ using the proof of `cn06` found by $\varpi$ with ratio 4 : 1. The coefficients $c_1, \ldots, c_{13}$ were computed by FC with $e = 80$ and $n = 10$. (This setting is apparently favorable according to the findings shown by table 9.) The same coefficients were also used by $\varpi_{FR\&F}$ for all problems of table 10. For reasons explained in section 5, the ratio $c_\delta : c_w$ is set to $0 : 1$ in connection with $\varpi_{FR\&F}$ (and $\varpi_{FR}$). However, the ratio 4 : 1 set when proving `cn06` is also used by $\varpi_F$. Furthermore, based on the experimental results regarding $\varpi_{FR}$, the parameters $p$, $q_1$ and $q_2$ were set to 20, 0.75 and 0.25, respectively. The proof of `cn15` is then employed to prove `cn22`, and its proof is used to prove `cn23`.

It is worth noting that `cn22` can be proved by $\varpi_{FR\&F}$ using the proof of `cn15`, whereas `cn23` cannot be proved this way (given the above restrictions concerning number of activated facts). On the other hand, `cn23` can be proved easily when using the proof of `cn22` both by $\varpi_{FR}$ and $\varpi_{FR\&F}$. When analyzing the proof of `cn23`, it became apparent that only one additional step is required to turn the proof of `cn22` into a proof of `cn23` which involves the goal of problem `cn22`. The reason why CoDE is not able to add this one additional step when using the source `cn15` despite the forced selection of a fact subsuming the goal is exactly this forced selection. Since the goal of `cn23` is different from the goal of `cn22`, the fact subsuming the goal of `cn22` is not activated in time when trying to prove `cn23`. In other words, if it were not for the forced selection of a fact concluding the current proof, we would not have obtained the result w.r.t. problem `cn22` in the first place.

Note that problems `cn03` and `cn15` differ in their axiomatization, but have the same goal. This is hence an example where the extension $\varpi_{FR\&F}$ of $\varpi_{FR}$ and $\varpi_{FR}$ itself are profitable even though the axiomatizations of source and target do not agree (which rules out simply adding the goal of the source to the axioms of the target, cp. section 5, table 7).

Problem `cn24` has so far resisted all our attempts. (According to our knowledge, it has not yet been proven by an automated deduction system on its own.)

# 9 Discussion

This report presented heuristics that exploit past proof experience. Approaches based on the same principle, namely incorporating information on previous experience into the search guiding heuristics (for automated deduction), are described in [SF71] and [SE90].

A different approach explicitly reuses proofs (e.g., [BCP88], [KW94]). The main idea there is to transform a given source proof into a proof of the actual target problem. The transformation is accomplished with the help of analogy mappings that represent the similarity of source and target problem. Due to the given notion of similarity, the approach is amenable to thorough theoretical examination. But also this similarity requirement as well as the restrictive principle to compute *deterministically* a proof based on a given proof curtail its practical applicability. In order to compensate for these restrictions elaborate "patching strategies" must be employed to (attempt to) save a transformation that is about to fail ([BCP88], [KW95]).

Approaches based on the heuristic use of past experience do not have to fall back on patching strategies, because they still *search* for a proof, attempting to utilize past experience to prune the search space. Therefore, moderate deviations from the source proof can be coped with due to the flexibility inherent to this kind of method. Our results sustain this claim.

A problem common to any approach for utilizing past experience is to determine *a priori* whether two problems are similar enough to be tackled profitably by using past experience. The degree of similarity may be assessed a posteriori, but cannot be decided for sure a priori. Therefore, heuristic criteria must be developed in order to achieve full automation, which includes selecting the appropriate source problem. This topic is not covered by this report, the main goal being to present experimental results regarding heuristics that make use of past experience. Future work will deal with the selection problem. Since it is next to impossible to pick always the appropriate source problem and the heuristic to use it, the TEAMWORK method ([AD93], [De95]) will play an essential role in alleviating the selection problem by allowing for the use of several heuristics (a *team*) concurrently and cooperatively. TEAMWORK is also useful for avoiding a major pitfall of using past experience, namely making things worse (cp. [KN93]). This hazard can be diminished by adding standard (basic) strategies to a team which do not rely on past experience.

In closing, we want to point out that despite a considerable number of potential parameter settings for the heuristics presented here, we succeeded in isolating a small number of them that perform satisfactorily well. Moreover, the capabilities of the heuristics have been demonstrated with numerous proof problems. The results we obtained with our experimental program CODE show significant improvements even in comparison with the renowned theorem prover OTTER which is clearly superior to CODE in terms of inference rate.

# A Proof Problems

In this appendix we give a complete description of all proof problems dealt with in this report. All of these problems stem from [MW92]. We use the same abbreviations to label axioms respectively theorems. The name of a problem is composed of the abbreviation of the calculus it belongs to and of the continuous numbering used in [MW92]. For instance, problem # 7 which is a problem of the CN calculus in [MW92] is named '`cn07`', while problem # 69, which is the first problem of the EC calculus, is given the name '`ec69`'. The problems themselves are presented in the obvious way, namely axiomatization and goal to prove. For a brief overview of the historical background and origin of the problems see [MW92] and in particular for the EC calculus [Wo90].

## A.1 The Implication/Negation Two-Valued Sentential Calculus (CN)

Each of the following facts is a theorem of the CN calculus.

| | | | |
|---|---|---|---|
| (CN-1) | $i(i(x,y),i(i(y,z),i(x,z)))$ | (CN-30) | $i(i(x,i(x,y)),i(x,y))$ |
| (CN-2) | $i(i(n(x),x),x)$ | (CN-35) | $i(i(x,i(y,z)),i(i(x,y),i(x,z)))$ |
| (CN-3) | $i(x,i(n(x),y))$ | (CN-37) | $i(i(i(x,y),z),i(n(x),z))$ |
| (CN-16) | $i(x,x)$ | (CN-39) | $i(n(n(x)),x)$ |
| (CN-18) | $i(x,i(y,x))$ | (CN-40) | $i(x,n(n(x)))$ |
| (CN-19) | $i(i(i(x,y),z),i(y,z))$ | (CN-46) | $i(i(x,y),i(n(y),n(x)))$ |
| (CN-20) | $i(x,i(i(x,y),y))$ | (CN-49) | $i(i(n(x),n(y)),i(y,x))$ |
| (CN-21) | $i(i(x,i(y,z)),i(y,i(x,z)))$ | (CN-54) | $i(i(x,y),i(i(n(x),y),y))$ |
| (CN-22) | $i(i(x,y),i(i(z,x),i(z,y)))$ | (CN-59) | $i(i(n(x),z),i(i(y,z),i(i(x,y),z)))$ |
| (CN-24) | $i(i(i(x,y),x),x)$ | | |

| | |
|---|---|
| (CN-60) | $i(i(x,i(n(y),z)),i(x,i(i(u,z),i(i(y,u),z))))$ |
| (CN-CAM) | $i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))$ |

The proof problems are:

| Name | Axiomatization | Goal |
|---|---|---|
| `cn01` | CN-18, CN-35, CN-39, CN-40, CN-46 | CN-21 |
| `cn02` | CN-18, CN-21, CN-22, CN-54 | CN-30 |
| `cn03` | CN-18, CN-21, CN-22, CN-54 | CN-35 |
| `cn04` | CN-18, CN-21, CN-22, CN-54 | CN-39 |
| `cn05` | CN-18, CN-21, CN-22, CN-54 | CN-40 |
| `cn06` | CN-18, CN-21, CN-22, CN-54 | CN-46 |

| Name | Axiomatization | Goal |
|---|---|---|
| cn07 | CN-1, CN-2, CN-3 | CN-16 |
| cn08 | CN-1, CN-2, CN-3 | CN-18 |
| cn09 | CN-1, CN-2, CN-3 | CN-19 |
| cn10 | CN-1, CN-2, CN-3 | CN-20 |
| cn11 | CN-1, CN-2, CN-3 | CN-21 |
| cn12 | CN-1, CN-2, CN-3 | CN-22 |
| cn13 | CN-1, CN-2, CN-3 | CN-24 |
| cn14 | CN-1, CN-2, CN-3 | CN-30 |
| cn15 | CN-1, CN-2, CN-3 | CN-35 |
| cn16 | CN-1, CN-2, CN-3 | CN-37 |
| cn17 | CN-1, CN-2, CN-3 | CN-39 |
| cn18 | CN-1, CN-2, CN-3 | CN-40 |
| cn19 | CN-1, CN-2, CN-3 | CN-46 |
| cn20 | CN-1, CN-2, CN-3 | CN-49 |
| cn21 | CN-1, CN-2, CN-3 | CN-54 |
| cn22 | CN-1, CN-2, CN-3 | CN-59 |
| cn23 | CN-1, CN-2, CN-3 | CN-60 |
| cn24 | CN-1, CN-2, CN-3 | CN-CAM |

| Name | Axiomatization | Goal |
|---|---|---|
| cn25 | CN-18, CN-35, CN-49 | CN-1 |
| cn26 | CN-18, CN-35, CN-49 | CN-2 |
| cn27 | CN-18, CN-35, CN-49 | CN-3 |
| cn28 | CN-19, CN-37, CN-59 | CN-1 |
| cn29 | CN-19, CN-37, CN-59 | CN-2 |
| cn30 | CN-19, CN-37, CN-59 | CN-3 |
| cn31 | CN-19, CN-37, CN-60 | CN-1 |
| cn32 | CN-19, CN-37, CN-60 | CN-2 |
| cn33 | CN-19, CN-37, CN-60 | CN-3 |

## A.2 The Many-Valued Sentential Calculus (MV)

Each of the following facts is a theorem of the MV calculus.

| | | | |
|---|---|---|---|
| (MV-1) | $i(x, i(y, x))$ | (MV-25) | $i(i(x, y), i(i(z, x), i(z, y)))$ |
| (MV-2) | $i(i(x, y), i(i(y, z), i(x, z)))$ | (MV-29) | $i(x, n(n(x)))$ |
| (MV-3) | $i(i(i(x, y), y), i(i(y, x), x))$ | (MV-33) | $i(i(n(x), y), i(n(y), x))$ |
| (MV-4) | $i(i(i(x, y), i(y, x)), i(y, x))$ | (MV-36) | $i(i(x, y), i(n(y), n(x)))$ |
| (MV-5) | $i(i(n(x), n(y)), i(y, x))$ | (MV-39) | $i(n(i(x, y)), n(y))$ |
| (MV-24) | $i(n(n(x)), x)$ | (MV-50) | $i(n(x), i(y, n(i(y, x))))$ |

The proof problems are:

| Name | Axiomatization | Goal |
|---|---|---|
| mv55 | MV-1, MV-2, MV-3, MV-5 | MV-4 |
| mv56 | MV-1, MV-2, MV-3, MV-5 | MV-24 |
| mv57 | MV-1, MV-2, MV-3, MV-5 | MV-25 |
| mv58 | MV-1, MV-2, MV-3, MV-5 | MV-29 |
| mv59 | MV-1, MV-2, MV-3, MV-5 | MV-33 |
| mv60 | MV-1, MV-2, MV-3, MV-5 | MV-36 |
| mv61 | MV-1, MV-2, MV-3, MV-5 | MV-39 |
| mv62 | MV-1, MV-2, MV-3, MV-5 | MV-50 |

## A.3 Problems considered when trying to find shorter proofs

### A.3.1 Theorems of the Equivalential Calculus (EC)

(EC-1) $e(e(e(x,y),e(z,x)),e(y,z))$
(EC-4) $e(e(x,y),e(y,x))$
(EC-5) $e(e(e(x,y),z),e(x,e(y,z)))$

(YRO) $e(e(x,y),e(z,e(e(z,y),x)))$
(YRM) $e(e(x,y),e(z,e(e(y,z),x)))$

### A.3.2 Theorems of the R Calculus (R)

(WO) $e(e(x,e(y,z)),e(z,e(y,x)))$
(YQM) $e(e(x,y),e(e(z,y),e(z,x)))$

(QYF) $e(e(e(x,y),e(x,z)),e(z,y))$
(XGJ) $e(x,e(e(y,e(z,x)),e(y,z)))$

### A.3.3 Theorems of the Left Group Calculus (LG)

(LG-1) $e(e(e(x,e(e(y,y),x)),z),z)$
(LG-2) $e(e(e(e(e(x,y),e(x,z)),e(y,z)),u),u)$
(LG-3) $e(e(e(e(e(e(x,y),e(x,z)),u),e(e(y,z),u)),v),v)$
(LG-4) $e(e(e(e(x,y),z),u),e(e(e(x,v),z),e(e(y,v),u)))$

### A.3.4 Theorems of the RG Calculus (RG)

(LG-1′) $e(x,e(x,e(e(y,z),e(e(y,u),e(z,u)))))$
(LG-2′) $e(x,e(x,e(e(y,e(z,z)),y)))$

### A.3.5 The Proof Problems

| Name | Axiomatization | Goal |
|---|---|---|
| `ec69` | EC-4, EC-5 | EC-1 |
| `ec79` | YRO | YRM |
| `r86` | WO | YQM |
| `r88` | QYF | XGJ |
| `lg89` | LG-2, LG-3, LG-4 | LG-1 |
| `lg90` | LG-2, LG-3 | LG-4 |
| `lg91` | LG-3 | LG-4 |
| `rg102` | LG-2′ | LG-1′ |

# B Proofs of 'ec69'

We list here the proofs of problem 'ec69' referred to by subsection 3.2. A proof $\mathcal{P} \equiv \lambda_1^+; \ldots; \lambda_m^+$ is displayed as a sequence of $m$ lines. Line $i$ has the following format:

$$a_i \quad n_i \quad \mathcal{J}_i \quad \lambda_i^+$$

where $n_i = i$ is simply the ordinal of line $i$ allowing for easier access. $\mathcal{J}_i$ shows the justification for deriving $\lambda_i^+$: If $\lambda_i^+$ is an axiom, then $\mathcal{J}_i$ contains no information ("blank"). Otherwise, $\mathcal{J}_i \equiv [n_1, n_2]$, where $n_1$ and $n_2$ refer to the immediate ancestors $\lambda_{n_1}^+$ and $\lambda_{n_2}^+$ of $\lambda_i^+$ which can be found in lines $n_1$ and $n_2$, respectively. In the latter case, $\lambda_{n_1}^+$ hosts (an instance of) $\lambda_{n_2}^+$ as a subterm as required by the inference rule of condensed detachment (cp. definition 2.3).

In order to explain $a_i$ we have to recall that a proof $\mathcal{P} \equiv \lambda_1^+; \ldots; \lambda_m^+$ is obtained from a search sequence $\mathcal{S} \equiv \lambda_1; \ldots; \lambda_n$ by omitting those $\lambda_i$ from $\mathcal{S}$ which are not an element of the set $P$ defined in subsection 2.2. Hence we can define an injective function $\pi : \{1, \ldots, m\} \to \{1, \ldots, n\}$, where $\pi(i) = a_i$ iff $\lambda_i^+ \equiv \lambda_{a_i}$. Hence $\pi$ connects the $i^{th}$ fact of a proof with its original position in the associated search sequence. The information contained by $a_i$ is useful if we are interested in assessing the efficiency of the search, since the difference $a_{i+1} - a_i - 1$ tells us how many "unnecessary" activation steps were performed between the activations of the "necessary" facts $\lambda_i^+$ and $\lambda_{i+1}^+$.

## B.1 Proof # 1 (Ratio $0:1$; Level: 8)

| | | | |
|---|---|---|---|
| 1 | 1 | | $e(e(y,x),e(x,y))$ |
| 2 | 2 | | $e(e(e(z,y),x),e(z,e(y,x)))$ |
| 3 | 3 | [ 2 , 1 ] | $e(y,e(x,e(x,y)))$ |
| 4 | 4 | [ 1 , 3 ] | $e(e(y,e(y,x)),x)$ |
| 6 | 5 | [ 2 , 4 ] | $e(y,e(e(y,x),x))$ |
| 17 | 6 | [ 2 , 2 ] | $e(e(z,y),e(x,e(z,e(y,x))))$ |
| 18 | 7 | [ 1 , 2 ] | $e(e(z,e(y,x)),e(e(z,y),x))$ |
| 119 | 8 | [ 6 , 5 ] | $e(z,e(y,e(e(e(y,x),x),z)))$ |
| 122 | 9 | [ 1 , 6 ] | $e(e(z,e(y,e(x,z))),e(y,x))$ |
| 123 | 10 | [ 6 , 1 ] | $e(z,e(e(y,x),e(e(x,y),z)))$ |
| 133 | 11 | [ 7 , 2 ] | $e(e(e(e(z,y),x),z),e(y,x))$ |
| 189 | 12 | [ 7 , 8 ] | $e(e(z,y),e(e(e(y,x),x),z))$ |
| 202 | 13 | [ 7 , 10 ] | $e(e(z,e(y,x)),e(e(x,y),z))$ |
| 214 | 14 | [ 11 , 9 ] | $e(z,e(y,e(x,e(e(y,x),z))))$ |
| 347 | 15 | [ 1 , 12 ] | $e(e(e(e(z,y),y),x),e(x,z))$ |
| 391 | 16 | [ 13 , 9 ] | $e(e(z,y),e(x,e(y,e(z,x))))$ |
| 424 | 17 | [ 7 , 14 ] | $e(e(z,y),e(x,e(e(y,x),z)))$ |
| 518 | 18 | [ 7 , 16 ] | $e(e(e(z,y),x),e(y,e(z,x)))$ |
| 543 | 19 | [ 15 , 17 ] | $e(e(z,e(e(y,z),e(x,y))),x)$ |
| 544 | 20 | [ 18 , 19 ] | $e(e(e(z,y),e(x,z)),e(y,x))$ |

## B.2 Proof # 2 (Ratio 1 : 1; Level: 7)

| | | | |
|---|---|---|---|
| 1 | 1 | | $e(e(y,x),e(x,y))$ |
| 2 | 2 | | $e(e(e(z,y),x),e(z,e(y,x)))$ |
| 17 | 3 | [ 2 , 2 ] | $e(e(z,y),e(x,e(z,e(y,x))))$ |
| 18 | 4 | [ 1 , 2 ] | $e(e(z,e(y,x)),e(e(z,y),x))$ |
| 23 | 5 | [ 1 , 3 ] | $e(e(z,e(y,e(x,z))),e(y,x))$ |
| 24 | 6 | [ 3 , 1 ] | $e(z,e(e(y,x),e(e(x,y),z)))$ |
| 28 | 7 | [ 4 , 2 ] | $e(e(e(e(z,y),x),z),e(y,x))$ |
| 39 | 8 | [ 4 , 5 ] | $e(e(e(z,e(y,e(x,z))),y),x)$ |
| 42 | 9 | [ 4 , 6 ] | $e(e(z,e(y,x)),e(e(x,y),z))$ |
| 82 | 10 | [ 1 , 8 ] | $e(z,e(e(y,e(x,e(z,y))),x))$ |
| 85 | 11 | [ 9 , 9 ] | $e(e(z,e(y,x)),e(z,e(x,y)))$ |
| 92 | 12 | [ 9 , 7 ] | $e(e(z,y),e(e(e(x,y),z),x))$ |
| 96 | 13 | [ 9 , 5 ] | $e(e(z,y),e(x,e(y,e(z,x))))$ |
| 230 | 14 | [ 5 , 10 ] | $e(e(z,e(e(y,x),e(x,z))),y)$ |
| 268 | 15 | [ 11 , 4 ] | $e(e(z,e(y,x)),e(x,e(z,y)))$ |
| 287 | 16 | [ 11 , 12 ] | $e(e(z,y),e(x,e(e(x,y),z)))$ |
| 298 | 17 | [ 4 , 13 ] | $e(e(e(z,y),x),e(y,e(z,x)))$ |
| 520 | 18 | [ 15 , 16 ] | $e(e(e(z,y),x),e(e(x,y),z))$ |
| 546 | 19 | [ 17 , 14 ] | $e(e(e(z,y),e(y,x)),e(x,z))$ |
| 547 | 20 | [ 18 , 19 ] | $e(e(e(z,y),e(x,z)),e(y,x))$ |

## B.3 Proof # 3 (Ratio 2 : 1; Level: 4)

| | | | |
|---|---|---|---|
| 1 | 1 | | $e(e(y,x),e(x,y))$ |
| 2 | 2 | | $e(e(e(z,y),x),e(z,e(y,x)))$ |
| 7 | 3 | [ 2 , 2 ] | $e(e(z,y),e(x,e(z,e(y,x))))$ |
| 9 | 4 | [ 1 , 2 ] | $e(e(z,e(y,x)),e(e(z,y),x))$ |
| 149 | 5 | [ 3 , 3 ] | $e(u,e(e(z,y),e(e(x,e(z,e(y,x))),u)))$ |
| 156 | 6 | [ 3 , 4 ] | $e(u,e(e(z,e(y,x)),e(e(e(z,y),x),u)))$ |
| 547 | 7 | [ 1 , 5 ] | $e(e(e(u,z),e(e(y,e(u,e(z,y))),x)),x)$ |
| 548 | 8 | [ 7 , 6 ] | $e(e(e(z,y),e(x,z)),e(y,x))$ |

## B.4 Proof # 4 (Ratio 3 : 1; Level: 4)

| | | | |
|---|---|---|---|
| 1 | 1 | | $e(e(y,x),e(x,y))$ |
| 2 | 2 | | $e(e(e(z,y),x),e(z,e(y,x)))$ |
| 6 | 3 | [ 2 , 2 ] | $e(e(z,y),e(x,e(z,e(y,x))))$ |
| 7 | 4 | [ 1 , 2 ] | $e(e(z,e(y,x)),e(e(z,y),x))$ |
| 52 | 5 | [ 3 , 3 ] | $e(u,e(e(z,y),e(e(x,e(z,e(y,x))),u)))$ |
| 56 | 6 | [ 3 , 4 ] | $e(u,e(e(z,e(y,x)),e(e(e(z,y),x),u)))$ |
| 295 | 7 | [ 1 , 5 ] | $e(e(e(u,z),e(e(y,e(u,e(z,y))),x)),x)$ |
| 296 | 8 | [ 7 , 6 ] | $e(e(e(z,y),e(x,z)),e(y,x))$ |

## B.5 Proof # 5 (Ratios $4:1$, $5:1$; Level: 4)

| | | | |
|---|---|---|---|
| 1 | 1 | | $e(e(y,x),e(x,y))$ |
| 2 | 2 | | $e(e(e(z,y),x),e(z,e(y,x)))$ |
| 6 | 3 | [ 2 , 2 ] | $e(e(z,y),e(x,e(z,e(y,x))))$ |
| 7 | 4 | [ 1 , 2 ] | $e(e(z,e(y,x)),e(e(z,y),x))$ |
| 52 | 5 | [ 3 , 3 ] | $e(u,e(e(z,y),e(e(x,e(z,e(y,x))),u)))$ |
| 56 | 6 | [ 3 , 4 ] | $e(u,e(e(z,e(y,x)),e(e(e(z,y),x),u)))$ |
| 240 | 7 | [ 1 , 5 ] | $e(e(e(u,z),e(e(y,e(u,e(z,y))),x)),x)$ |
| 241 | 8 | [ 7 , 6 ] | $e(e(e(z,y),e(x,z)),e(y,x))$ |

## B.6 Proof # 6 (Ratio $6:1$; Level: 4)

| | | | |
|---|---|---|---|
| 1 | 1 | | $e(e(y,x),e(x,y))$ |
| 2 | 2 | | $e(e(e(z,y),x),e(z,e(y,x)))$ |
| 4 | 3 | [ 2 , 2 ] | $e(e(z,y),e(x,e(z,e(y,x))))$ |
| 5 | 4 | [ 1 , 2 ] | $e(e(z,e(y,x)),e(e(z,y),x))$ |
| 25 | 5 | [ 3 , 3 ] | $e(u,e(e(z,y),e(e(x,e(z,e(y,x))),u)))$ |
| 29 | 6 | [ 3 , 4 ] | $e(u,e(e(z,e(y,x)),e(e(e(z,y),x),u)))$ |
| 169 | 7 | [ 1 , 5 ] | $e(e(e(u,z),e(e(y,e(u,e(z,y))),x)),x)$ |
| 170 | 8 | [ 7 , 6 ] | $e(e(e(z,y),e(x,z)),e(y,x))$ |

## B.7 Proof # 7 (Ratios $7:1$,...,$10:1$; Level: 4)

| | | | |
|---|---|---|---|
| 1 | 1 | | $e(e(y,x),e(x,y))$ |
| 2 | 2 | | $e(e(e(z,y),x),e(z,e(y,x)))$ |
| 4 | 3 | [ 2 , 2 ] | $e(e(z,y),e(x,e(z,e(y,x))))$ |
| 5 | 4 | [ 1 , 2 ] | $e(e(z,e(y,x)),e(e(z,y),x))$ |
| 25 | 5 | [ 3 , 3 ] | $e(u,e(e(z,y),e(e(x,e(z,e(y,x))),u)))$ |
| 29 | 6 | [ 3 , 4 ] | $e(u,e(e(z,e(y,x)),e(e(e(z,y),x),u)))$ |
| 115 | 7 | [ 1 , 5 ] | $e(e(e(u,z),e(e(y,e(u,e(z,y))),x)),x)$ |
| 116 | 8 | [ 7 , 6 ] | $e(e(e(z,y),e(x,z)),e(y,x))$ |

## Notes

Please note that proof # 1 and proof # 2 do not differ in their length, but they do differ in the level.

The remaining proofs only differ in the efficiency of the search, which is illustrated by the numbers representing the activation steps.

# C Experimental Evaluation of $\varpi_{FR}$

We list here our experimental results concerning the performance of $\varpi_{FR}$ in dependence of the parameters $p$, $q_1$, $q_2$ and the different types of source proofs obtained with the basic heuristic $\varpi$ (cf. subsection 3.1). Each subsection C.1–C.8 deals with a different target–source combination given in its heading. The heading also shows the *best* (fastest) result produced by $\varpi$ when employed to prove the target problem. This information stems from the respective table (2, 3 or 4) in subsection 3.1 and is intended to serve as a point of reference to rate the results listed in the tables of the respective subsection C.1–C.8. The different tables of each subsection refer to the different types of source proofs which are displayed by the respective caption. The caption also shows (in brackets) the ratio $c_\delta : c_w$ used by $\varpi$ to find the respective source proof (cp. tables 2–4 in subsection 3.1). The following example illustrates the information made available by each of the subsequent subsections.

Consider, for instance, subsection C.3. Subsection C.3 deals with the performance of $\varpi_{FR}$ with respect to the target problem `cn29` and the source problem `cn28`. The best (fastest) result produced by $\varpi$ is displayed in the heading, namely a proof found within 35 seconds. The (body of the) first table lists the results obtained with $\varpi_{FR}$ when using a proof of `cn28` which is the fastest and at the same time the longest and the one with the lowest level. This information is provided by the caption which also shows that this particular proof was found (by $\varpi$) when employing the ratio $3:1$ for $c_\delta : c_w$ (cp. table 3 in subsection 3.1).

Please note that in case a caption enumerates several types of source proofs this does not mean that all these source proofs are distinct, yet yielding the same results displayed in the body of the table. It means that these proofs are identical (in the sense that the associated sets of positive sets are equal).

*Remark*: There is no caption if only one (type of) source proof was found by $\varpi$ (see C.1 and C.2).

## C.1 Target: `cn19`, Source: `cn21`; [$\varpi$: 84 seconds]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | 158s | 158s | 64s |
| 0.00 | 0.50 | — | — | — | — | 150s | 151s | 156s |
| 0.00 | 0.75 | — | 151s | 151s | 141s | 54s | 54s | 54s |
| 0.00 | 1.00 | 28s | 28s | 28s | 28s | 28s | 29s | 29s |
| 0.25 | 0.25 | — | — | 82s | 70s | 71s | 59s | 55s |
| 0.25 | 0.50 | — | — | 82s | 71s | 72s | 61s | 60s |
| 0.25 | 0.75 | 155s | 125s | 59s | 59s | 51s | 51s | 45s |
| 0.50 | 0.25 | — | 87s | 86s | 59s | 44s | 43s | 30s |
| 0.50 | 0.50 | 115s | 91s | 99s | 70s | 52s | 45s | 45s |
| 0.50 | 0.75 | 72s | 53s | 44s | 33s | 27s | 27s | 27s |
| 0.75 | 0.25 | — | 80s | 77s | 63s | 52s | 40s | 41s |
| 0.75 | 0.50 | 92s | 80s | 62s | 54s | 55s | 30s | 30s |
| 0.75 | 0.75 | 45s | 32s | 33s | 33s | 27s | 27s | 27s |
| 1.00 | 0.25 | 82s | 58s | 98s | 83s | 68s | 55s | 54s |
| 1.00 | 0.50 | 63s | 90s | 91s | 72s | 67s | 56s | 42s |
| 1.00 | 0.75 | 52s | 41s | 42s | 30s | 30s | 30s | 25s |

## C.2 Target: `cn21`, Source: `cn19`; [$\varpi$: 91 seconds]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | 42s | 43s | 47s | 52s | 53s | 57s | 63s |
| 0.00 | 0.50 | 42s | 43s | 46s | 51s | 51s | 52s | 52s |
| 0.00 | 0.75 | 47s | 50s | 51s | 53s | 52s | 52s | 52s |
| 0.00 | 1.00 | 62s | 61s | 62s | 62s | 62s | 62s | 62s |
| 0.25 | 0.25 | 62s | 40s | 44s | 41s | 42s | 45s | 45s |
| 0.25 | 0.50 | 39s | 42s | 44s | 41s | 41s | 41s | 42s |
| 0.25 | 0.75 | 55s | 51s | 52s | 52s | 51s | 52s | 51s |
| 0.50 | 0.25 | 33s | 61s | 59s | 46s | 47s | 48s | 50s |
| 0.50 | 0.50 | 61s | 55s | 62s | 48s | 49s | 49s | 49s |
| 0.50 | 0.75 | 64s | 49s | 50s | 50s | 51s | 44s | 44s |
| 0.75 | 0.25 | 28s | 43s | 59s | 58s | 60s | 45s | 46s |
| 0.75 | 0.50 | 34s | 50s | 65s | 63s | 65s | 47s | 49s |
| 0.75 | 0.75 | 80s | 70s | 79s | 52s | 52s | 52s | 52s |
| 1.00 | 0.25 | 27s | 35s | 50s | 69s | 59s | 59s | 59s |
| 1.00 | 0.50 | 30s | 36s | 55s | 72s | 73s | 67s | 74s |
| 1.00 | 0.75 | 85s | 92s | 80s | 84s | 55s | 55s | 55s |

## C.3 Target: `cn29`, Source: `cn28`; [$\varpi$: 35 seconds]

*fastest, longest, lowest level* [ 3 : 1 ]

| $q_1$ | $q_2$ | $p=0$ | $p=20$ | $p=40$ | $p=60$ | $p=80$ | $p=100$ | $p=120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | 5s | 5s | 5s | 11s | 23s | 19s | 31s |
| 0.00 | 0.50 | 5s | 5s | 5s | 5s | 5s | 8s | 12s |
| 0.00 | 0.75 | 5s | 5s | 5s | 6s | 5s | 5s | 5s |
| 0.00 | 1.00 | 7s | 7s | 7s | 7s | 7s | 7s | 7s |
| 0.25 | 0.25 | 7s | 6s | 6s | 7s | 13s | 20s | 19s |
| 0.25 | 0.50 | 6s | 6s | 7s | 4s | 4s | 4s | 8s |
| 0.25 | 0.75 | 8s | 5s | 5s | 5s | 5s | 5s | 5s |
| 0.50 | 0.25 | 10s | 7s | 7s | 7s | 9s | 13s | 21s |
| 0.50 | 0.50 | 8s | 7s | 7s | 7s | 7s | 8s | 9s |
| 0.50 | 0.75 | 8s | 9s | 9s | 9s | 10s | 5s | 5s |
| 0.75 | 0.25 | 15s | 12s | 8s | 7s | 8s | 9s | 12s |
| 0.75 | 0.50 | 14s | 10s | 10s | 8s | 9s | 8s | 9s |
| 0.75 | 0.75 | 13s | 11s | 12s | 11s | 11s | 11s | 11s |
| 1.00 | 0.25 | 13s | 36s | 17s | 12s | 12s | 8s | 11s |
| 1.00 | 0.50 | 28s | 21s | 15s | 15s | 14s | 11s | 10s |
| 1.00 | 0.75 | 13s | 16s | 14s | 14s | 12s | 12s | 12s |

*slowest, shortest, highest level* [ 4 : 3 ]

| $q_1$ | $q_2$ | $p=0$ | $p=20$ | $p=40$ | $p=60$ | $p=80$ | $p=100$ | $p=120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | 5s | 5s | 6s | 13s | 16s | 17s | 30s |
| 0.00 | 0.50 | 5s | 5s | 5s | 6s | 6s | 6s | 13s |
| 0.00 | 0.75 | 5s | 6s | 6s | 6s | 6s | 6s | 6s |
| 0.00 | 1.00 | 7s | 7s | 7s | 7s | 7s | 7s | 11s |
| 0.25 | 0.25 | 8s | 6s | 5s | 6s | 16s | 17s | 17s |
| 0.25 | 0.50 | 6s | 5s | 5s | 5s | 6s | 6s | 12s |
| 0.25 | 0.75 | 5s | 5s | 5s | 5s | 5s | 5s | 6s |
| 0.50 | 0.25 | 13s | 11s | 8s | 6s | 11s | 17s | 26s |
| 0.50 | 0.50 | 12s | 8s | 6s | 6s | 5s | 5s | 6s |
| 0.50 | 0.75 | 7s | 6s | 6s | 6s | 6s | 7s | 7s |
| 0.75 | 0.25 | 12s | 8s | 11s | 7s | 6s | 12s | 17s |
| 0.75 | 0.50 | 10s | 8s | 16s | 16s | 9s | 8s | 8s |
| 0.75 | 0.75 | 8s | 8s | 8s | 6s | 7s | 6s | 6s |
| 1.00 | 0.25 | 9s | 6s | 8s | 12s | 9s | 9s | 11s |
| 1.00 | 0.50 | 8s | 6s | 9s | 10s | 9s | 6s | 6s |
| 1.00 | 0.75 | 8s | 8s | 6s | 6s | 5s | 5s | 5s |

## C.4 Target: `cn29`, Source: `cn30`; [$\varpi$: 35 seconds]

*fastest, longest, highest level* [ 4 : 3 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | 80s | 80s | 70s |
| 0.00 | 0.50 | — | — | — | — | — | 77s | 77s |
| 0.00 | 0.75 | — | — | 77s | 77s | 73s | 80s | 82s |
| 0.00 | 1.00 | 95s | 95s | 95s | 95s | 95s | 97s | 96s |
| 0.25 | 0.25 | — | — | — | 72s | 62s | 62s | 36s |
| 0.25 | 0.50 | — | — | 126s | 74s | 68s | 62s | 62s |
| 0.25 | 0.75 | 127s | 74s | 72s | 72s | 71s | 80s | 79s |
| 0.50 | 0.25 | 28s | 82s | 56s | 32s | 30s | 26s | 24s |
| 0.50 | 0.50 | — | 72s | 43s | 33s | 28s | 28s | 27s |
| 0.50 | 0.75 | 62s | 30s | 30s | 29s | 29s | 33s | 67s |
| 0.75 | 0.25 | 19s | 69s | 38s | 28s | 23s | 23s | 16s |
| 0.75 | 0.50 | 43s | 33s | 25s | 24s | 21s | 15s | 15s |
| 0.75 | 0.75 | 17s | 15s | 12s | 15s | 27s | 28s | 32s |
| 1.00 | 0.25 | 48s | 24s | 80s | 45s | 41s | 33s | 23s |
| 1.00 | 0.50 | 14s | 50s | 25s | 25s | 19s | 18s | 18s |
| 1.00 | 0.75 | 20s | 15s | 12s | 12s | 13s | 17s | 17s |

*shortest, lowest level* [ 4 : 1 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | 99s | 116s | 99s |
| 0.00 | 0.50 | — | — | — | — | — | 83s | 92s |
| 0.00 | 0.75 | — | 106s | 92s | 93s | 34s | 34s | 34s |
| 0.00 | 1.00 | 9s | 9s | 9s | 9s | 9s | 9s | 9s |
| 0.25 | 0.25 | — | — | 111s | 68s | 54s | 45s | 53s |
| 0.25 | 0.50 | — | — | 100s | 83s | 70s | 72s | 73s |
| 0.25 | 0.75 | 109s | 96s | 66s | 38s | 38s | 36s | 37s |
| 0.50 | 0.25 | 13s | 45s | 32s | 22s | 20s | 20s | 28s |
| 0.50 | 0.50 | 74s | 50s | 34s | 28s | 23s | 23s | 26s |
| 0.50 | 0.75 | 62s | 37s | 36s | 26s | 21s | 16s | 15s |
| 0.75 | 0.25 | 11s | 25s | 22s | 13s | 12s | 10s | 11s |
| 0.75 | 0.50 | 29s | 26s | 50s | 21s | 18s | 15s | 16s |
| 0.75 | 0.75 | 26s | 18s | 17s | 18s | 17s | 17s | 15s |
| 1.00 | 0.25 | 26s | 13s | 18s | 17s | 13s | 15s | 12s |
| 1.00 | 0.50 | 9s | 23s | 40s | 29s | 25s | 10s | 10s |
| 1.00 | 0.75 | 17s | 15s | 7s | 15s | 14s | 15s | 15s |

*slowest* [ 0 : 1 ]

| $q_1$ | $q_2$ | $p=0$ | $p=20$ | $p=40$ | $p=60$ | $p=80$ | $p=100$ | $p=120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | 6s | 7s | 7s | 8s | 10s | 14s | 21s |
| 0.00 | 0.50 | 6s | 7s | 7s | 8s | 8s | 8s | 8s |
| 0.00 | 0.75 | 7s | 8s | 8s | 8s | 8s | 8s | 8s |
| 0.00 | 1.00 | 11s | 11s | 11s | 11s | 11s | 11s | 11s |
| 0.25 | 0.25 | 10s | 9s | 9s | 9s | 9s | 12s | 14s |
| 0.25 | 0.50 | 9s | 8s | 9s | 9s | 8s | 8s | 8s |
| 0.25 | 0.75 | 9s | 8s | 8s | 8s | 8s | 8s | 8s |
| 0.50 | 0.25 | 14s | 12s | 13s | 10s | 11s | 11s | 12s |
| 0.50 | 0.50 | 12s | 10s | 11s | 11s | 10s | 10s | 10s |
| 0.50 | 0.75 | 11s | 10s | 10s | 11s | 11s | 11s | 9s |
| 0.75 | 0.25 | 13s | 32s | 21s | 20s | 16s | 15s | 15s |
| 0.75 | 0.50 | 20s | 17s | 16s | 16s | 15s | 15s | 12s |
| 0.75 | 0.75 | 12s | 12s | 13s | 13s | 12s | 12s | 12s |
| 1.00 | 0.25 | 22s | 25s | 92s | 50s | 48s | 34s | 24s |
| 1.00 | 0.50 | 15s | 56s | 27s | 27s | 20s | 19s | 20s |
| 1.00 | 0.75 | 21s | 16s | 15s | 15s | 15s | 15s | 15s |

## C.5 Target: cn32, Source: cn31; [$\varpi$: 45 seconds]

*slowest, shortest, highest level* [ 1 : 1 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | 13s | 13s | 14s | 15s | 24s | 24s | 35s |
| 0.00 | 0.50 | 13s | 13s | 14s | 14s | 14s | 14s | 16s |
| 0.00 | 0.75 | 14s | 14s | 14s | 15s | 16s | 16s | 16s |
| 0.00 | 1.00 | 19s | 18s | 19s | 19s | 19s | 19s | 19s |
| 0.25 | 0.25 | 16s | 17s | 15s | 15s | 19s | 21s | 20s |
| 0.25 | 0.50 | 17s | 15s | 15s | 15s | 14s | 15s | 17s |
| 0.25 | 0.75 | 15s | 14s | 16s | 16s | 16s | 16s | 16s |
| 0.50 | 0.25 | 29s | 20s | 22s | 23s | 21s | 28s | 22s |
| 0.50 | 0.50 | 22s | 17s | 19s | 20s | 21s | 17s | 17s |
| 0.50 | 0.75 | 19s | 17s | 17s | 17s | 17s | 18s | 17s |
| 0.75 | 0.25 | 40s | 17s | 11s | 22s | 23s | 29s | 31s |
| 0.75 | 0.50 | 20s | 14s | 18s | 20s | 26s | 26s | 26s |
| 0.75 | 0.75 | 21s | 23s | 22s | 23s | 18s | 17s | 17s |
| 1.00 | 0.25 | 62s | 21s | 15s | 11s | 14s | 19s | 24s |
| 1.00 | 0.50 | 18s | 17s | 10s | 12s | 20s | 27s | 29s |
| 1.00 | 0.75 | 12s | 21s | 24s | 25s | 25s | 25s | 25s |

*fastest, longest, lowest level* [ 2 : 1 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | 42s | 44s | 42s | 78s | 25s | 32s | 34s |
| 0.00 | 0.50 | 43s | 44s | 44s | 77s | 81s | 88s | 88s |
| 0.00 | 0.75 | 84s | 85s | 87s | 86s | 18s | 18s | 18s |
| 0.00 | 1.00 | 19s | 19s | 19s | 19s | 19s | 19s | 19s |
| 0.25 | 0.25 | 26s | 44s | 50s | 83s | 74s | 30s | 20s |
| 0.25 | 0.50 | 63s | 49s | 82s | 88s | 95s | 73s | 24s |
| 0.25 | 0.75 | 86s | 94s | 47s | 19s | 19s | 18s | 18s |
| 0.50 | 0.25 | 56s | 22s | 21s | 48s | 27s | 30s | 28s |
| 0.50 | 0.50 | 41s | 26s | 26s | 36s | 28s | 24s | 19s |
| 0.50 | 0.75 | 55s | 23s | 19s | 18s | 17s | 17s | 17s |
| 0.75 | 0.25 | 30s | 14s | 9s | 16s | 22s | 30s | 34s |
| 0.75 | 0.50 | 16s | 10s | 10s | 22s | 26s | 27s | 27s |
| 0.75 | 0.75 | 24s | 22s | 22s | 23s | 17s | 18s | 17s |
| 1.00 | 0.25 | 46s | 18s | 12s | 9s | 12s | 16s | 22s |
| 1.00 | 0.50 | 15s | 13s | 8s | 9s | 18s | 27s | 27s |
| 1.00 | 0.75 | 10s | 21s | 25s | 25s | 26s | 25s | 26s |

## C.6 Target: `cn32`, Source: `cn33`; [$\varpi$: 45 seconds]

*slowest, highest level* [ 0 : 1 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | 13s | 12s | 14s | 17s | 22s | 30s |
| 0.00 | 0.50 | 63s | 13s | 11s | 12s | 12s | 12s | 14s |
| 0.00 | 0.75 | 12s | 12s | 13s | 13s | 13s | 13s | 13s |
| 0.00 | 1.00 | 17s | 18s | 17s | 17s | 18s | 17s | 17s |
| 0.25 | 0.25 | — | 14s | 14s | 15s | 16s | 19s | 23s |
| 0.25 | 0.50 | 16s | 14s | 14s | 14s | 13s | 14s | 14s |
| 0.25 | 0.75 | 14s | 13s | 13s | 13s | 13s | 13s | 13s |
| 0.50 | 0.25 | 62s | 19s | 20s | 20s | 17s | 19s | 22s |
| 0.50 | 0.50 | 20s | 16s | 16s | 16s | 15s | 15s | 17s |
| 0.50 | 0.75 | 16s | 15s | 16s | 17s | 18s | 18s | 15s |
| 0.75 | 0.25 | 61s | 52s | 28s | 26s | 22s | 26s | 25s |
| 0.75 | 0.50 | 27s | 21s | 21s | 20s | 23s | 23s | 22s |
| 0.75 | 0.75 | 19s | 19s | 19s | 19s | 17s | 18s | 18s |
| 1.00 | 0.25 | — | 111s | 121s | 65s | 61s | 43s | 33s |
| 1.00 | 0.50 | 58s | 71s | 36s | 36s | 25s | 25s | 25s |
| 1.00 | 0.75 | 26s | 20s | 22s | 22s | 22s | 22s | 22s |

*fastest* [ 3 : 1 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | 111s | 132s | 108s |
| 0.00 | 0.50 | — | — | — | — | — | 100s | 110s |
| 0.00 | 0.75 | — | 113s | 108s | 111s | 47s | 47s | 47s |
| 0.00 | 1.00 | 16s | 16s | 16s | 16s | 16s | 16s | 16s |
| 0.25 | 0.25 | — | — | 109s | 53s | 45s | 55s | 58s |
| 0.25 | 0.50 | — | — | 44s | 63s | 57s | 39s | 77s |
| 0.25 | 0.75 | 109s | 104s | 72s | 49s | 49s | 47s | 47s |
| 0.50 | 0.25 | — | 39s | 33s | 26s | 18s | 22s | 29s |
| 0.50 | 0.50 | 68s | 51s | 33s | 28s | 19s | 16s | 16s |
| 0.50 | 0.75 | 56s | 43s | 50s | 22s | 14s | 16s | 17s |
| 0.75 | 0.25 | 45s | 25s | 15s | 13s | 13s | 24s | 37s |
| 0.75 | 0.50 | 30s | 22s | 48s | 20s | 22s | 24s | 25s |
| 0.75 | 0.75 | 23s | 24s | 19s | 18s | 18s | 17s | 17s |
| 1.00 | 0.25 | 45s | 20s | 13s | 12s | 12s | 12s | 17s |
| 1.00 | 0.50 | 16s | 17s | 30s | 30s | 26s | 15s | 30s |
| 1.00 | 0.75 | 9s | 22s | 24s | 23s | 18s | 18s | 18s |

*longest* [ 3 : 2 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | 115s | 117s | 98s |
| 0.00 | 0.50 | — | — | — | — | — | 113s | 116s |
| 0.00 | 0.75 | — | — | 109s | 109s | 99s | 109s | 114s |
| 0.00 | 1.00 | 119s | 118s | 119s | 119s | 121s | 126s | 128s |
| 0.25 | 0.25 | — | — | — | 110s | 95s | 57s | 56s |
| 0.25 | 0.50 | — | — | — | 105s | 98s | 88s | 86s |
| 0.25 | 0.75 | — | 108s | 99s | 99s | 99s | 109s | 107s |
| 0.50 | 0.25 | — | — | 78s | 47s | 47s | 40s | 42s |
| 0.50 | 0.50 | — | 97s | 60s | 44s | 42s | 42s | 41s |
| 0.50 | 0.75 | 74s | 45s | 45s | 43s | 41s | 46s | 86s |
| 0.75 | 0.25 | — | 107s | 49s | 37s | 30s | 34s | 29s |
| 0.75 | 0.50 | 73s | 42s | 32s | 31s | 26s | 23s | 22s |
| 0.75 | 0.75 | 21s | 23s | 19s | 24s | 39s | 40s | 44s |
| 1.00 | 0.25 | — | — | 104s | 58s | 53s | 41s | 31s |
| 1.00 | 0.50 | 54s | 63s | 32s | 32s | 24s | 23s | 23s |
| 1.00 | 0.75 | 25s | 19s | 22s | 21s | 21s | 27s | 27s |

*shortest, lowest level* [ 4 : 1 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | 101s | 110s | 95s |
| 0.00 | 0.50 | — | — | — | — | — | 110s | 117s |
| 0.00 | 0.75 | — | 96s | 115s | 113s | 44s | 44s | 44s |
| 0.00 | 1.00 | 16s | 16s | 15s | 16s | 15s | 16s | 16s |
| 0.25 | 0.25 | — | — | — | — | 73s | 64s | 97s |
| 0.25 | 0.50 | — | — | — | 62s | 50s | 53s | 31s |
| 0.25 | 0.75 | 70s | 113s | 96s | 46s | 46s | 46s | 45s |
| 0.50 | 0.25 | — | — | — | 173s | 69s | 40s | 36s |
| 0.50 | 0.50 | — | — | 160s | 75s | 35s | 31s | 24s |
| 0.50 | 0.75 | 33s | 26s | 14s | 14s | 15s | 15s | 17s |
| 0.75 | 0.25 | 113s | 62s | 45s | 56s | 38s | 33s | 36s |
| 0.75 | 0.50 | 60s | 78s | 27s | 24s | 26s | 23s | 17s |
| 0.75 | 0.75 | 14s | 17s | 16s | 16s | 16s | 16s | 16s |
| 1.00 | 0.25 | — | 57s | 72s | 59s | 22s | 38s | 47s |
| 1.00 | 0.50 | 20s | 31s | 21s | 22s | 29s | 28s | 27s |
| 1.00 | 0.75 | 20s | 20s | 16s | 16s | 16s | 16s | 16s |

## C.7 Target: `mv60`, Source: `mv59`; [$\varpi$ failed]

*longest, highest level* [ 1 : 1 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | — | — | — |
| 0.00 | 0.50 | — | — | — | — | — | — | — |
| 0.00 | 0.75 | — | — | — | — | — | — | — |
| 0.00 | 1.00 | — | — | — | — | — | — | — |
| 0.25 | 0.25 | 38s | 95s | — | — | — | — | — |
| 0.25 | 0.50 | 73s | — | — | — | — | — | — |
| 0.25 | 0.75 | — | — | — | — | — | — | — |
| 0.50 | 0.25 | 8s | 22s | 47s | 113s | 151s | — | — |
| 0.50 | 0.50 | 22s | 44s | 73s | 119s | 126s | — | — |
| 0.50 | 0.75 | 98s | — | — | — | — | — | — |
| 0.75 | 0.25 | 6s | 12s | 21s | 42s | 57s | 83s | 105s |
| 0.75 | 0.50 | 8s | 13s | 23s | 46s | 48s | 73s | 116s |
| 0.75 | 0.75 | 34s | 75s | 101s | 141s | — | — | — |
| 1.00 | 0.25 | 5s | 9s | 13s | 21s | 48s | 53s | 57s |
| 1.00 | 0.50 | 7s | 9s | 15s | 24s | 28s | 59s | 59s |
| 1.00 | 0.75 | 28s | 34s | 65s | 104s | 116s | — | — |

*fastest, shortest* [ 2 : 1 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | — | — | — |
| 0.00 | 0.50 | — | — | — | — | — | — | — |
| 0.00 | 0.75 | — | — | — | — | — | — | — |
| 0.00 | 1.00 | — | — | — | — | — | — | — |
| 0.25 | 0.25 | 35s | 87s | — | — | — | — | — |
| 0.25 | 0.50 | 69s | — | — | — | — | — | — |
| 0.25 | 0.75 | — | — | — | — | — | — | — |
| 0.50 | 0.25 | 7s | 19s | 43s | 103s | 140s | — | — |
| 0.50 | 0.50 | 19s | 41s | 68s | 113s | 121s | — | — |
| 0.50 | 0.75 | 94s | — | — | — | — | — | — |
| 0.75 | 0.25 | 5s | 10s | 17s | 37s | 53s | 76s | 97s |
| 0.75 | 0.50 | 7s | 11s | 20s | 41s | 45s | 67s | 109s |
| 0.75 | 0.75 | 32s | 71s | 98s | 141s | — | — | — |
| 1.00 | 0.25 | 4s | 7s | 11s | 17s | 41s | 47s | 53s |
| 1.00 | 0.50 | 6s | 8s | 12s | 20s | 24s | 54s | 55s |
| 1.00 | 0.75 | 26s | 33s | 61s | 99s | 115s | — | — |

*slowest* [ 2 : 3 ]

| $q_1$ | $q_2$ | $p=0$ | $p=20$ | $p=40$ | $p=60$ | $p=80$ | $p=100$ | $p=120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | — | — | — |
| 0.00 | 0.50 | — | — | — | — | — | — | — |
| 0.00 | 0.75 | — | — | — | — | — | — | — |
| 0.00 | 1.00 | — | — | — | — | — | — | — |
| 0.25 | 0.25 | 38s | 93s | — | — | — | — | — |
| 0.25 | 0.50 | 72s | — | — | — | — | — | — |
| 0.25 | 0.75 | — | — | — | — | — | — | — |
| 0.50 | 0.25 | 8s | 21s | 48s | 110s | 153s | — | — |
| 0.50 | 0.50 | 21s | 44s | 74s | 120s | 126s | — | — |
| 0.50 | 0.75 | 97s | — | — | — | — | — | — |
| 0.75 | 0.25 | 6s | 11s | 21s | 43s | 55s | 87s | 112s |
| 0.75 | 0.50 | 8s | 12s | 22s | 46s | 48s | 76s | 118s |
| 0.75 | 0.75 | 33s | 77s | 100s | 142s | — | — | — |
| 1.00 | 0.25 | 5s | 8s | 12s | 20s | 47s | 52s | 56s |
| 1.00 | 0.50 | 6s | 8s | 14s | 23s | 27s | 60s | 59s |
| 1.00 | 0.75 | 28s | 33s | 64s | 101s | 115s | — | — |

*lowest level* [ 3 : 1 ]

| $q_1$ | $q_2$ | $p=0$ | $p=20$ | $p=40$ | $p=60$ | $p=80$ | $p=100$ | $p=120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | — | — | — |
| 0.00 | 0.50 | — | — | — | — | — | — | — |
| 0.00 | 0.75 | — | — | — | — | — | — | — |
| 0.00 | 1.00 | — | — | — | — | — | — | — |
| 0.25 | 0.25 | 36s | — | — | — | — | — | — |
| 0.25 | 0.50 | 80s | — | — | — | — | — | — |
| 0.25 | 0.75 | — | — | — | — | — | — | — |
| 0.50 | 0.25 | 7s | 19s | 45s | 110s | 134s | — | — |
| 0.50 | 0.50 | 19s | 42s | 67s | 123s | 126s | — | — |
| 0.50 | 0.75 | 105s | — | — | — | — | — | — |
| 0.75 | 0.25 | 7s | 10s | 18s | 36s | 44s | 67s | 88s |
| 0.75 | 0.50 | 7s | 11s | 21s | 41s | 45s | 66s | 119s |
| 0.75 | 0.75 | 33s | 74s | 109s | 139s | — | — | — |
| 1.00 | 0.25 | 12s | 7s | 12s | 16s | 32s | 35s | 42s |
| 1.00 | 0.50 | 8s | 8s | 13s | 20s | 23s | 53s | 51s |
| 1.00 | 0.75 | 25s | 34s | 64s | 102s | 119s | 159s | 159s |

## C.8 Target: `mv62`, Source: `mv59`; [$\varpi$ failed]

*longest, highest level* [ 1 : 1 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | — | — | — |
| 0.00 | 0.50 | — | — | — | — | — | — | — |
| 0.00 | 0.75 | — | — | — | — | — | — | — |
| 0.00 | 1.00 | — | — | — | — | — | — | — |
| 0.25 | 0.25 | — | 96s | — | — | — | — | — |
| 0.25 | 0.50 | — | — | — | — | — | — | — |
| 0.25 | 0.75 | — | — | — | — | — | — | — |
| 0.50 | 0.25 | 57s | 23s | 48s | 114s | 152s | — | — |
| 0.50 | 0.50 | 42s | 44s | 73s | 120s | 126s | — | — |
| 0.50 | 0.75 | 98s | — | — | — | — | — | — |
| 0.75 | 0.25 | 55s | 21s | 21s | 44s | 58s | 84s | 106s |
| 0.75 | 0.50 | 19s | 14s | 23s | 46s | 49s | 74s | 118s |
| 0.75 | 0.75 | 34s | 76s | 101s | 141s | — | — | — |
| 1.00 | 0.25 | 89s | 49s | 24s | 23s | 49s | 54s | 58s |
| 1.00 | 0.50 | 30s | 16s | 16s | 24s | 28s | 60s | 60s |
| 1.00 | 0.75 | 29s | 35s | 66s | 106s | 117s | — | — |

*fastest, shortest* [ 2 : 1 ]

| $q_1$ | $q_2$ | $p = 0$ | $p = 20$ | $p = 40$ | $p = 60$ | $p = 80$ | $p = 100$ | $p = 120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | — | — | — |
| 0.00 | 0.50 | — | — | — | — | — | — | — |
| 0.00 | 0.75 | — | — | — | — | — | — | — |
| 0.00 | 1.00 | — | — | — | — | — | — | — |
| 0.25 | 0.25 | — | 87s | — | — | — | — | — |
| 0.25 | 0.50 | — | — | — | — | — | — | — |
| 0.25 | 0.75 | — | — | — | — | — | — | — |
| 0.50 | 0.25 | 54s | 20s | 44s | 105s | 145s | — | — |
| 0.50 | 0.50 | 38s | 41s | 69s | 115s | 123s | — | — |
| 0.50 | 0.75 | 95s | — | — | — | — | — | — |
| 0.75 | 0.25 | 48s | 18s | 18s | 38s | 53s | 77s | 98s |
| 0.75 | 0.50 | 16s | 12s | 21s | 41s | 45s | 67s | 110s |
| 0.75 | 0.75 | 33s | 71s | 99s | 144s | — | — | — |
| 1.00 | 0.25 | 76s | 43s | 20s | 19s | 42s | 48s | 54s |
| 1.00 | 0.50 | 26s | 14s | 14s | 21s | 25s | 55s | 56s |
| 1.00 | 0.75 | 27s | 33s | 61s | 100s | 117s | — | — |

*slowest* [ 2 : 3 ]

| $q_1$ | $q_2$ | $p=0$ | $p=20$ | $p=40$ | $p=60$ | $p=80$ | $p=100$ | $p=120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | — | — | — |
| 0.00 | 0.50 | — | — | — | — | — | — | — |
| 0.00 | 0.75 | — | — | — | — | — | — | — |
| 0.00 | 1.00 | — | — | — | — | — | — | — |
| 0.25 | 0.25 | — | 94s | — | — | — | — | — |
| 0.25 | 0.50 | — | — | — | — | — | — | — |
| 0.25 | 0.75 | — | — | — | — | — | — | — |
| 0.50 | 0.25 | 57s | 22s | 48s | 112s | 153s | — | — |
| 0.50 | 0.50 | 41s | 44s | 74s | 121s | 126s | — | — |
| 0.50 | 0.75 | 98s | — | — | — | — | — | — |
| 0.75 | 0.25 | 54s | 21s | 21s | 43s | 56s | 87s | 111s |
| 0.75 | 0.50 | 19s | 13s | 23s | 47s | 49s | 76s | 117s |
| 0.75 | 0.75 | 33s | 78s | 101s | 141s | — | — | — |
| 1.00 | 0.25 | 88s | 49s | 23s | 22s | 48s | 52s | 57s |
| 1.00 | 0.50 | 29s | 16s | 15s | 24s | 28s | 60s | 59s |
| 1.00 | 0.75 | 28s | 33s | 65s | 102s | 116s | — | — |

*lowest level* [ 3 : 1 ]

| $q_1$ | $q_2$ | $p=0$ | $p=20$ | $p=40$ | $p=60$ | $p=80$ | $p=100$ | $p=120$ |
|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.25 | — | — | — | — | — | — | — |
| 0.00 | 0.50 | — | — | — | — | — | — | — |
| 0.00 | 0.75 | — | — | — | — | — | — | — |
| 0.00 | 1.00 | — | — | — | — | — | — | — |
| 0.25 | 0.25 | — | — | — | — | — | — | — |
| 0.25 | 0.50 | — | — | — | — | — | — | — |
| 0.25 | 0.75 | — | — | — | — | — | — | — |
| 0.50 | 0.25 | — | 90s | 45s | 113s | 137s | 138s | 137s |
| 0.50 | 0.50 | — | 48s | 67s | 126s | 130s | — | 150s |
| 0.50 | 0.75 | 105s | — | — | — | — | — | — |
| 0.75 | 0.25 | — | 70s | 35s | 37s | 46s | 68s | 90s |
| 0.75 | 0.50 | 52s | 34s | 21s | 43s | 47s | 67s | 120s |
| 0.75 | 0.75 | 34s | 73s | 110s | 144s | — | — | — |
| 1.00 | 0.25 | — | — | 92s | 38s | 34s | 36s | 43s |
| 1.00 | 0.50 | 66s | 24s | 21s | 20s | 24s | 55s | 51s |
| 1.00 | 0.75 | 25s | 34s | 63s | 104s | 121s | — | — |

# D Experiments with $\varpi_F$

Using proof of `cn06` found with ratio 2 : 1 for proving `cn03`

| | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
|---|---|---|---|---|---|---|---|---|
| 40 | 40% — | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | 56% 78s |
| 50 | 58% 81s | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ↑ |
| 60 | 68% 32s | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | 69% 46s |
| 70 | 70% 32s | ⟵ | 70% 32s | 75% 24s | ⟵ | ⟵ | ⟵ | 73% 40s |
| 80 | 82% 18s | ⟵ | 80% 22s | 81% 17s | 80% 20s | ⟵ | 82% 17s | 80% 21s |
| 90 | 90% 52s | 90% 17s | 90% 24s | 90% 18s | 89% 6s | ⟵ | 88% 6s | 86% 5s |

Using proof of `cn06` found with ratio 3 : 1 for proving `cn03`

| | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
|---|---|---|---|---|---|---|---|---|
| 40 | 55% – | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | 51% — |
| 50 | ↑ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ↑ |
| 60 | 64% 40s | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | ⟵ | 64% 56s |
| 70 | 72% 37s | ⟵ | 70% 27s | ⟵ | ⟵ | ⟵ | ⟵ | 70% 31s |
| 80 | 80% 23s | 80% 24s | 80% 17s | 80% 14s | 81% 12s | 80% 14s | ⟵ | 77% 17s |
| 90 | 90% 14s | 90% 6s | 90% 12s | 90% 10s | 87% 6s | 87% 7s | ⟵ | ↑ |

# References

[AD93] **Avenhaus, J.; Denzinger, J.**: *Distributing equational theorem proving*, Proc. $5^{th}$ RTA, Montreal, CAN, 1993, LNCS 690, pp. 62–76

[BCP88] **Brock, B.; Cooper, S.; Pierce, W.**: *Analogical reasoning and proof discovery*, Proc. CADE 9, Argonne, IL, USA, 1988, LNCS 310, pp. 454–468

[Bu89] **Burstein, M.H.**: *Analogy vs. CBR: The purpose of mapping*, in: K.J. Hammond (ed.), Proceedings: Second case-based reasoning workshop (DARPA), Morgan Kaufmann, San Mateo, CA, USA, 1989, pp. 133–136

[Ca86] **Carbonell, J.**: *Derivational analogy: a theory of reconstructive problem solving and expertise acquisition*, in: R.S. Michalski et al. (eds.), Machine Intelligence; an AI approach, Vol. 2, 1986, pp. 371–392

[CL73] **Chang, C.L.; Lee, R.C.**: *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973

[Cu95] **Curien, R.**: *Outils pour la preuve par analogie*, thèse de doctorat, Faculté des Sciences, Université Henri Poincaré – Nancy I, 1995 (french language)

[De95] **Denzinger, J.**: *Knowledge-Based Distributed Search Using Teamwork*, Proc. $1^{st}$ ICMAS, San Francisco, CA, USA, 1995, pp. 81–88

[Fu95a] **Fuchs, M.**: *Learning proof heuristics by adapting parameters*, In Armand Prieditis & Stuart Russell, eds., *Machine Learning: Proceedings of the Twelfth International Conference*, Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 235–243

[Fu95b] **Fuchs, M.**: *Exploiting past proof experience*, Techn. Report LSA-95-08E, University of Kaiserslautern, 1995

[Hu80] **Huet, G.**: *Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems*, Journal of the ACM 27, No. 4, 1980, pp. 798–821

[Kl71] **Kling, R.E.**: *A Paradigm for Reasoning by Analogy*, Artificial Intelligence 2, 1971, pp. 147–178

[KN93] **Koehler, J.; Nebel, B.**: *Plan modification versus plan generation*, Proc. IJCAI '93, Chambery, FRA, 1993, pp. 1436–1444

[KW94] **Kolbe, T.; Walther, C.**: *Reusing proofs*, Proc. $11^{th}$ ECAI '94, Amsterdam, HOL, 1994, pp. 80–84

[KW95] **Kolbe, T.; Walther, C.**: *Patching Proofs for Reuse*, Proc. $8^{th}$ ECML '95, Heraklion, Crete/Greece, 1995

[Lu70] **Lukasiewicz, J.**: *Selected Works*, edited by L. Borkowski, North-Holland, 1970

[MW92] **McCune, W.; Wos, L.:** *Experiments in Automated Deduction with Condensed Detachment*, Proc. CADE 11, Saratoga Springs, NY, USA, 1992, LNAI 607, pp. 209–223

[Pe76] **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

[Re83] **Rendell, L.:** *A new basis for state-space learning systems and a successful implementation*, Artificial Intelligence 20, 1983, pp. 369–392

[SE90] **Suttner, C.; Ertel, W.:** *Automatic acquisition of search guiding heuristics*, Proc. CADE 10, Kaiserslautern, FRG, 1990, LNAI 449, pp. 470–484

[SF71] **Slagle, J.R.; Farrell, C.D.:** *Experiments in automatic learning for a multipurpose heuristic program*, Communications of the ACM, Vol. 14, Nr. 2, 1971, pp. 91–99

[Sl93] **Slaney, J.:** *SCOTT: A Model-Guided Theorem Prover*, Proc. IJCAI '93, Chambery, FRA, 1993, pp. 109–114

[Su90] **Suttner, C.:** *Representing heuristic-relevant information for an automated theorem prover*, Proc. $6^{th}$ IMYCS: Aspects and Prospects of Theoretical Computer Science, LNAI 464, 1990

[Ta56] **Tarski, A.:** *Logic, Semantics, Metamathematics*, Oxford University Press, 1956

[Wo90] **Wos, L.:** *Meeting the Challenge of Fifty Years of Logic*, JAR 6, 1990, pp. 213-232