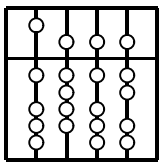
INSTITUT FÜR INFORMATIK

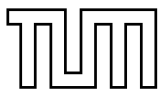


Lehrstuhl für Rechnerkommunikation und maschinelle Deduktion
Prof. Dr. E. Jessen

Example Selection for Learning in Automated Theorem Proving

Diplomarbeit
Release 3

Aufgabensteller:	Prof. E. Jessen
Betreuer:	Stephan Schulz
Bearbeiter:	Felix Brandt
Abgabedatum:	15.8.1998



TECHNISCHE UNIVERSITÄT MÜNCHEN

Ich erkläre, daß ich diese Diplomarbeit selbständig verfaßt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, der 15.8.1998

(Felix Brandt)

Abstract

The equational theorem prover DISCOUNT, based on an extended version of unfailing Knuth-Bendix completion, features advanced learning techniques for heuristic evaluation functions. We present an extension to DISCOUNT's most recent learning strategy called *term space mapping*. Experiments have shown that if too many examples are used for learning, performance of the prover decreases. Three different strategies that preselect examples from the knowledge base are evaluated using a set of TPTP examples and compared with previous learning methods and conventional heuristics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of automated theorem proving is to find a proof for a given hypothesis under the assumption of a set of axioms. Due to the semi-decidability of this problem, it is necessary to search for proofs in an infinite search space.

To accelerate the search process, improved calculi, very fast implementations [HBF96], and heuristics to guide the search process were developed. DISCOUNT (DIStributed COmpletion UsiNg Teamwork) is a proof system that originated at the University of Kaiserslautern and has been under development since 1991. Although the basic inference machine is quite slow compared with today's standards, DISCOUNT can still compete with modern provers due to very strong heuristics for search control, including a variety of unique learning heuristics. DISCOUNT's latest heuristics are based on learning methods for acquiring search control knowledge by examples of successful proof searches. The most recent learning algorithm for the equational theorem prover DISCOUNT is *learning by term space mapping*. This strategy applies data from earlier proof searches to build a tree that is then used to evaluate new equations. Experimental results show that this strategy's performance can be improved if the training examples are carefully selected.

The aim of this work is to investigate strategies for selecting training examples from a flat knowledge base. Three different selection methods were developed. The first one is based on features such as number of axioms, number of operators of a given arity in the signature of a problem, and average term size. The second one compares term space maps spanned by the axioms. A third method combines the two other strategies.

For this purpose, the learning strategies were extended to preselect examples from the knowledge base and to learn only facts generated by these examples. Numerous test runs had to be made to obtain a successful set of parameters for the new selection techniques. The resulting system was tested and evaluated by comparing it with both conventional and learning strategies without example selection.

This thesis has the following structure. After the introduction, the second chapter starts with a brief summary of the knowledge needed to comprehend the basics of equational deduction. As this work is mainly about improving learning methods, only little knowledge about automated theorem proving (ATP) is

required. Chapter 3 explains the different learning techniques developed for the DISCOUNT system. The following chapter establishes the need for knowledge restriction, offers three example selecting strategies and compares the performance of those strategies with several sets of parameters. Subsequently, chapter 5 presents detailed results of various test runs on a subset of TPTP [SS97] problems. Finally, the sixth chapter concludes this thesis with an evaluation of the improvements rendered by example selection. The appendices contain a computer-generated example proof, a description of the problem pool used for evaluation throughout this work, and implementation details concerning DISCOUNT.

Chapter 2

Equational Deduction

The basic problem in equational deduction can be expressed as follows:

> Given a set of equations E, is $s = t$ a logical consequence of E (denoted as $s =_E t$)?

To handle this problem, some theoretical background is required. We reduced the amount of theory in this chapter to the basics; for a more comprehensive introduction, see [Ave95]. We omitted proofs for theorems 1 (Correctness of U) and 2 (Completeness of U), which can be found in [Ave95] as well.

This chapter starts with fundamental definitions and notations, presents a deduction algorithm, and compares conventional evaluation functions. When learning methods are used to improve the prover's performance, the basic algorithm will not be modified; only the evaluation functions will be replaced.

2.1 Theoretical Foundations

Definition 1 (Terms)
Let F be a finite set of function symbols and V an enumerable set of variables. A term is defined recursively:

- All variables $v \in V$ are terms.
- If $t_1, \ldots, t_n$ are terms and $f \in F$ is a function symbol of arity n, then $f(t_1, \ldots, t_n)$ is a term.

The set of terms built from F and V is called $Term(F, V)$. The set of ground terms $Term(F)$ contains terms that do not contain variables. The set $var(t)$ contains the variables occurring in a term t.

According to the previous definition, terms are recursively built by combining a function symbol with a number of subterms. Subterms can be described by using *positions* in terms.

Definition 2 (Term positions and Subterms)
Positions in terms are sequences of natural numbers. λ denotes the empty sequence, $t|_p$ the *subterm* of t at the position p, and $O(t)$ the set of all positions in t ($a.b$ stands for the concatenation of two sequences a and b).

- If $t \in Term(F,V)$ is a constant or a variable, $O(t) = \{\lambda\}$ and $t|_\lambda \equiv t$.
- If $t \equiv f(t_1, \ldots, t_n)$, $O(t) = \{i.p|\ 1 \leq i \leq n,\ p \in O(t_i)\} \cup \{\lambda\}$. $t|_{i.p} \equiv t_i|p$ and $t|_\lambda \equiv t$.

Definition 3 (Equations)
A term pair $(s,t) \in (Term(F,V) \times Term(F,V))$ is called an equation. Equations are symmetrical: $(s,t) = (t,s)$. We signify an equation as $s = t$ instead of (s,t).

Apart from function symbols, terms consist of variables, which are substitutes for terms. This mapping is conveyed through a *Substitution*.

Definition 4 (Substitutions)
A substitution $\sigma : V \mapsto Term(F,V)$ maps a finite subset of variables $\{v|\ \sigma(v) \neq v\}$ to the set of terms and the rest of the variables to themselves. $\sigma = \{x_1 \leftarrow t_1, \ldots x_n \leftarrow t_n\}$ means that σ maps x_i to t_i for $1 \leq i \leq n$. σ_{id} denotes the empty substitution $\{\}$. $\sigma : V \mapsto Term(F,V)$ is extended to $\sigma : Term(F,V) \mapsto Term(F,V)$ by defining $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$. $\sigma \circ \tau$ denotes the concatenation of two substitutions σ and τ.

τ is *more general* than ρ if there is a substitution σ with $\tau \equiv \sigma \circ \rho$. This ordering is called *subsumption ordering*.

By applying substitutions, it is possible to *unify* two terms, i.e. to make them syntactically equal through a substitution. An important unifying substitution is the *most general unifier*.

Definition 5 (Most general unifier)
The most general unifier $mgu(s,t)$ is a substitution σ with $\sigma(s) \equiv \sigma(t)$, that is more general than any other substitution τ that fulfills $\tau(s) \equiv \tau(t)$.

If s and t are unifiable, a unique (apart from renaming variables) most general unifier of s and t can be computed.

One way of manipulating terms is to substitute subterms.

Definition 6 (Subterm substitutions)
$t[p \leftarrow t']$ is the term t with the subterm at position p replaced by t'.

Definition 7 (Encompassment ordering)
The encompassment ordering $\trianglerighteq$ is defined by $s \trianglerighteq t$ if and only if $\sigma(s) = t|_p$ for a substitution σ and a position p. $\triangleright$ is the strict part of $\trianglerighteq$.

Definition 8 (Noetherian orderings)
An ordering $>$ is called Noetherian (or *terminating*) if there is no infinite chain $t_1 > t_2 > t_3 > \ldots$.

Definition 9 (Reduction orderings)
A reduction ordering $>$ is a Noetherian ordering compatible with the term structure ($t_1 > t_2 \Rightarrow t[p \leftarrow t_1] > t[p \leftarrow t_2]$) and stable with respect to substitutions ($t_1 > t_2 \Rightarrow \sigma(t_1) > \sigma(t_2)$). A ground reduction ordering is a reduction ordering that is total on ground terms.

Definition 10 (Rules)
A rule is a tuple $(l, r) \in (Term(F, V) \times Term(F, V))$ with $Var(r) \subset Var(l)$ and is denoted as $l \to r$. A rule $l \to r$ is compatible with a reduction ordering $>$ if $l > r$. A set of rules R is compatible with $>$ if all rules in R are compatible with $>$.

Birkhoff [Bir35] has proven the conformity of the semantic and the operational E-equality, i.e. $s =_E t$ holds if and only if it is possible to transform s into t by applying equations from E. The application of an equation, a single operational E-equality-step, is defined as follows.

Definition 11 (Operational E-equality)
$t_1 \vdash\dashv_E t_2$ if and only if there is an equation $s = t$, a position p in t_1 and a substitution σ with $\sigma(s) \equiv t_1|p$ and $t_2 \equiv t_1[p \leftarrow \sigma(t)]$. $\vdash\dashv_E$ is symmetrical.
$=_E$ is the reflexive and transitive closure of $\vdash\dashv_E$.

Knuth-Bendix completion [KB70] builds a confluent system of rules by orienting equations according to a reduction ordering $>$, generating new equations from so-called critical pairs, and applying rules to simplify existing rules.

This calculus has been extended to handle unorientable equations [BDP89] by generating only a ground confluent system and using orientable equations for simplifications. The resulting calculus is called *unfailing Knuth-Bendix completion.*

Definition 12 (Reduction relation)
Let R be a set of rules.
$t_1 \Longrightarrow_R t_2$ if and only if there exists a rule $l \to r$, a position p in t_1 and a substitution σ with $t_1|p \equiv \sigma(l)$ and $t_2 \equiv t_1[p \leftarrow \sigma(r)]$. If R is compatible with a reduction ordering, the relation $\Longrightarrow$ is Noetherian.

Definition 13 (Confluence)
Let $\to$ be a relation and $\overset{*}{\to}$ the reflexive and transitive closure of $\to$.
A relation $\to$ is said to be *confluent* if each divergence ($u \overset{*}{\to} x$ and $u \overset{*}{\to} y$) possible in $\to$ can be rejoined ($x \overset{*}{\to} v$ and $y \overset{*}{\to} v$).

Definition 14 (Normal form)
A term that cannot be reduced by application of $\Longrightarrow_R$, is in *normal form* (with respect to a set of rules R).

Definition 15 (Orientable instances)
Let R be a set of rules.
The set $R_E = \{\sigma(s) \to \sigma(t) | \sigma(s) > \sigma(t), s = t \in E, \sigma \text{ is a substitution}\}$ is called the set of *orientable instances* for E and $>$. $R(E) = R \cup R_E$

Definition 16 (Critical pairs)
Let $l_1 = r_1$ and $l_2 = r_2$ be two equations, $>$ a reduction ordering, p a non-variable position in l_1 and $\sigma = mgu(l_1|p, l_2)$. If $\sigma(l_1) \not< \sigma(r_1)$ and $\sigma(l_2) \not< \sigma(r_2)$ then $\langle \sigma(r_1), \sigma(l_1[p \leftarrow r_2]) \rangle$ is called a *critical pair* between $l_1 = r_1$ and $l_2 = r_2$. Critical pairs between rules are defined similarly. $CP(E, R)$ is the set of all critical pairs that can be built from E and R.

2.2 Unfailing Completion

A completion-based prover tries to reduce both sides of the equation to be proven into a common normal form by using a ground confluent and terminating system of rules and equations derived from E. The following input is handed to the prover:

- A set of equations E
- A skolemized goal $s = t$
- A ground reduction ordering $>$

The current state of the proof process consists of three sets:

- E (processed, but unorientable **equations**)
- R (processed and oriented equations: **rules**)
- CP (unprocessed equations: **critical pairs**)

E and R are empty sets at the beginning of the completion process, whereas CP contains the initial axioms. The prover then chooses an equation from CP, reduces it to normal form (applying E and R), builds new critical pairs from it, and deletes redundancies from E and R using **(S1)** and **(S2)**. The resulting equation is added to R or E (depending on whether it can be oriented), and the next critical pair is then examined.
On top of this algorithm, the goal is reduced to normal form with respect to E and R. The goal is proven if both normal forms are identical or subsumed by an equation from E.

The following calculus just uses one set of equations E that contains the processed *and* unprocessed equations. The concrete algorithm in the next section will distinguish between E and CP.

(U1) (Orient an equation)
$$\frac{(E \cup \{s = t\}, R)}{(E, R \cup \{s \to t\})} \text{ if } s > t$$

(U2) (Generate an equation)
$$\frac{(E, R)}{(E \cup \{s = t\}, R)} \text{ if } s \vdash_{E \cup R} u \vdash_{E \cup R} t, s \not> u, t \not> u$$

(U3) (Simplify an equation)
$$\frac{(E \cup \{s = t\}, R)}{(E \cup \{u = t\}, R)} \text{ if } s \Longrightarrow_R u \text{ or if } s \Longrightarrow_{R_E} u \text{ using } l \to r \text{ with } s \rhd l$$

(U4a) (Delete an equation)
$$\frac{(E \cup \{s = s\}, R)}{(E, R)}$$

(U4b) (Subsume an equation)
$$\frac{(E \cup \{s = t, u = v\}, R)}{(E \cup \{s = t\}, r)} \text{ if } u|_p \equiv \sigma(s), v \equiv [p \leftarrow t], u \rhd s,$$
for a position p and a substitution σ

(S1) (Simplify the right side of a rule)
$$\frac{(E, R \cup \{s \to t\})}{(E, R \cup \{s \to u\})} \text{ if } t \Longrightarrow_{R(E)} u$$

(S2) (Simplify the left side of a rule)
$$\frac{(E, R \cup \{s \to t\})}{(E \cup \{u = t\}, R)} \text{ if } s \Longrightarrow_{R(E)} t \text{ using a rule } l \to r \text{ with } s \rhd l$$

$(E, R) \vdash_U (E', R')$ denotes that (E, R) can be transformed into (E',R') using a single inference rule from U. Applying inference rules does not change the equality described by (E, R). This is called *correctness* of the calculus.

Theorem 1 (Correctness of U)
Assume $(E, R) \vdash_U (E', R')$ and let R be compatible with a reduction ordering $>$. This has the following consequences:

- R' is compatible with $>$
- $=_{E \cup R} = =_{E' \cup R'}$

According to theorem 1 inference steps from U do not change the equality defined by E and R. In order to obtain an end system with the desired features, the sequence of inferences needs to fulfill certain conditions.

Definition 17 (U-fairness)
A sequence $(E_i, R_i)_{i \in \mathbb{N}}$ is called a *U-derivation* if $\forall i \in \mathbb{N} : \quad (E_i, R_i) \vdash_U (E_{i+1}, R_{i+1})$. The system (R^∞, R^∞) of *persistent* rules and equations is defined by

$$E^\infty = \bigcup_{i \geq 0} \bigcap_{j \geq i} E_j \qquad \text{and} \qquad R^\infty = \bigcup_{i \geq 0} \bigcap_{j \geq i} R_j \tag{2.1}$$

A U-derivation $(E_i, R_i)_{i \in \mathbb{N}}$ is *fair*, if

$$CP(E^\infty, R^\infty) \subseteq \bigcup_{i \geq 0} E_i \tag{2.2}$$

In other words, a derivation is fair, if each critical pair between persistent rules and equations is built at some time and then added to the set of equations.

Theorem 2 (Completeness of U)
Assume $(E_i, R_i)_{i \in \mathbb{N}}$ is a fair U-derivation and let $s, t \in Term(F)$ be two ground terms with $s =_E t$. Then there is an i such that the normal forms of s and t with respect to (E_i, R_i) are identical.

2.3 A Semi-Decision Algorithm for E-Equality

Using the theoretical results of the previous sections, it is possible to construct an algorithm that tries to decide on E-equality by building a fair derivation of ground confluent sets of rules and equations.

Figure 2.1 shows a simplified version of the unfailing Knuth-Bendix completion algorithm actually used in DISCOUNT. Depending on `PROOFMODE` it is able to either generate a ground confluent end system or prove a theorem. Table 2.1 lists the input, the variables and the functions used in the sketch algorithm.

Input	
`A`	A set of axioms
`>`	A ground reduction ordering
`PROOFMODE`	`TRUE` if a single goal is to be proven
	`FALSE` if a ground confluent end system is required
`(gs,gt)`	The goal to be proven (if `PROOFMODE` is `TRUE`)
Variables	
`R`	Processed rules
`E`	Processed equations
`CP`	Unprocessed equations
`s,t`	Terms of an unprocessed equation
`l,r`	Terms of a new rule
`u,v`	Terms of a processed term pair
`u',v'`	Descendants of `u` and `v`
Functions	
`NOTEMPTY(list)`	`FALSE` if `list` is empty
	`TRUE` otherwise
`FIRST(list)`	First element of `list`
`REST(list)`	`list` without first element
`NORMALFORM(t,R)`	Computes the normal form of `t` with respect to `R`
`SUBSUM(e,E)`	`TRUE` if `e` is subsumed by an equation in `E`
	or an equation equivalent to `e` exists in `E`
	`FALSE` otherwise
`INSERT_CP(list,e)`	Inserts `e` into `list` by applying a fair strategy
`CPS(e,R)`	Returns all critical pairs between `e` and `E`

Table 2.1: Completion algorithm input, variables, and functions

Importance lies in the fact that `INSERT_CP(list,e)` is required to ensure fairness. Only a finite number of pairs may be inserted in front of each equation `e`. This can be accomplished by evaluating the critical pairs using a suitable heuristic and inserting them at the appropriate position in the list. The program sketch in figure 2.2 realizes this (The symbol "`.`" is used for concatenating two lists).

`EVAL` : $Term(F,V) \times Term(F,V) \mapsto \mathbb{E}$ computes the evaluation of a critical pair based on some heuristic. Lower numbers signify more, greater numbers less "quality" of the pair. $\mathbb{E}$ is a subset of $\mathbb{R}$.

```
BEGIN
  CP := A;
  E := {};
  R := {};
  WHILE NOTEMPTY(CP)
    IF PROOFMODE = TRUE THEN
      gs := NORMALFORM(gs,R(E));
      gt := NORMALFORM(gt,R(E));
      IF gs = gt THEN END; /* Proof found */
    ENDIF
    (s,t) := FIRST(CP);
    CP := REST(CP);
    s := NORMALFORM(s,R(E));
    t := NORMALFORM(t,R(E));
    IF s≠t AND NOT SUBSUM(s=t,E) THEN
      FOREACH (u,v) ∈ E
        u' := NORMALFORM(u,R{s=t});
        v' := NORMALFORM(v,R{s=t});
        IF u≠u' OR v≠v' THEN
          E := E \ {u=v}
          CP := INSERT_CP(CP,u'=v');
        ENDIF
      ENDFOREACH
      FOREACH (u,v) ∈ R
        v := NORMALFORM(v,R(E ∪ {s=t}));
        u' := NORMALFORM(u,R{s=t});
        IF NOT u'=u THEN
          R := R \ {(u,v)}
          CP := INSERT_CP(CP,u'=v);
        ENDIF
      ENDFOREACH
      FOREACH (u,v) ∈ CPS(s=t,E ∪ R) CP := INSERT_CP(CP,u=v);
      IF s>t THEN R := R ∪ {(s,t)};
      ELSE IF s<t THEN R := R ∪ {(t,s)};
      ELSE E := E ∪ {s=t};
      ENDIF
    ENDIF
  ENDWHILE
  /* (E,R) represents the ground confluent end system */
END
```

Figure 2.1: Simplified completion algorithm

```
INSERT_CP(list,e)
  IF EVAL(e) > EVAL(FIRST(list)) THEN
    RETURN FIRST(list)+INSERT(REST(list),e))
  ELSE
    RETURN e.list
  ENDIF
END
```

Figure 2.2: INSERT_CP using an evaluation function

Since the main algorithm always selects the first critical pair from CP, the control of the completion completely depends on INSERT_CP, which is solely guided by the critical pair evaluation function EVAL. As a consequence the fairness of the completion algorithm can be attributed to the properties of EVAL.

Theorem 3 (Fairness of the completion algorithm)
The completion algorithm is U-fair, if[1]
$\forall e \in Term(F,V) \times Term(F,V): \quad \{e' |\ \texttt{EVAL}(e') \leq \texttt{EVAL}(e)\}$ is finite

Proof: As stated in definition 17 a derivation is U-fair, if every critical pair between persistent rules and equations will eventually be built. This holds for the completion algorithm, that builds all critical pairs between R and E, if no term pair remains in CP forever.

The condition ensures that not more than a finite number of critical pairs are inserted before e. This guarantees that e will finally be reached and removed as each iteration of the algorithm removes the first critical pair from CP.

If $\mathbb{E}$ is a set of integer values with a lower bound b, there is a fairness criterion that is easier to prove.

Theorem 4 (Fairness for integer evaluations)
The completion algorithm is U-fair, if[2]
$\mathbb{E} \subset \mathbb{Z} \quad \wedge \quad \exists b \in \mathbb{E}: \ \forall a \in \mathbb{E}: \ b \leq a \quad \wedge$
$\forall i \in \mathrm{N}: \quad \{e |\ e \in Term(F,V) \times Term(F,V),\ \texttt{EVAL}(e) = i\}$ is finite

Proof: Let cp $\in$ CP be a critical pair with EVAL(cp) $= j$. As EVAL maps critical pairs to integers greater than a lower bound b, there is only a finite number of evaluations that are less than or equal to j. According to the assertion only finitely many critical pairs are mapped to a single value. Thus, not more than a finite number of term pairs will ever be inserted before cp.

[1] If term pairs with equal evaluations are selected fairly (e.g. applying fifo) or the evaluation function is injective, the condition is sufficient *and* required.

[2] Please pay attention to the fact that this condition is sufficient, but *not* required.

2.4 Conventional Evaluation Functions

The selection of the next critical pair to be processed is the most crucial step in completion-based deduction. This choice is guided by an evaluation function that maps equations to real numbers representing the value of critical pairs in the proof process.

2.4.1 First In, First Out - `fifo`

The easiest fair strategy is *first in, first out* (`fifo`). Term pairs are processed in the same order in which they were generated.

Definition 18 (`fifo`)
Let $t \in Term(F,V)$ be a term and $w = 1$. w is incremented every time `fifo` is called.

$$\begin{array}{rcll} \texttt{fifo} : Term(F,V) \times Term(F,V) & \mapsto & \mathbb{N} & \\ \texttt{fifo}(s=t) & = & w & (w = w+1) \end{array} \tag{2.3}$$

This method yields very poor performance (see table 2.2).

Theorem 5
`fifo` is fair.

Proof: $\forall e \in Term(F,V) \times Term(F,V) : \quad \{e' |\ \texttt{fifo}(e') \leq \texttt{fifo}(e)\} = \emptyset$

2.4.2 Simple Term Weighting - `add_weight`

A primitive evaluation function that prefers small terms is `add_weight`.

Definition 19 (`add_weight`)
Let $t \in Term(F,V)$ be a term. Then the weight of this term is defined as

$$\texttt{weight}(t) = \begin{cases} 1 & \text{if } t \in V \\ 2 + \sum_{i=1}^{n} \texttt{weight}(t_i) & \text{if } t \equiv f(t_1, t_2, \ldots, t_n) \end{cases} \tag{2.4}$$

For the weight of the entire equation, the two equation terms are added.

$$\begin{array}{rcl} \texttt{add\_weight} : Term(F,V) \times Term(F,V) & \mapsto & \mathbb{N} \\ \texttt{add\_weight}(s=t) & = & \texttt{weight}(s) + \texttt{weight}(t) \end{array} \tag{2.5}$$

Despite its simplicity, `add_weight` performs very well (see table 2.2).

Theorem 6
`add_weight` is fair.

Proof: `add_weight` yields positive integers by definition. According to theorem 4 it remains to be shown that not more than a finite number of critical pairs are mapped onto the same integer value.

Because F is a finite set and V can be treated as a finite set (due to the subsumption in the deduction algorithm), it is obvious that `add_weight` cannot map infinitely many term pairs onto the same value.

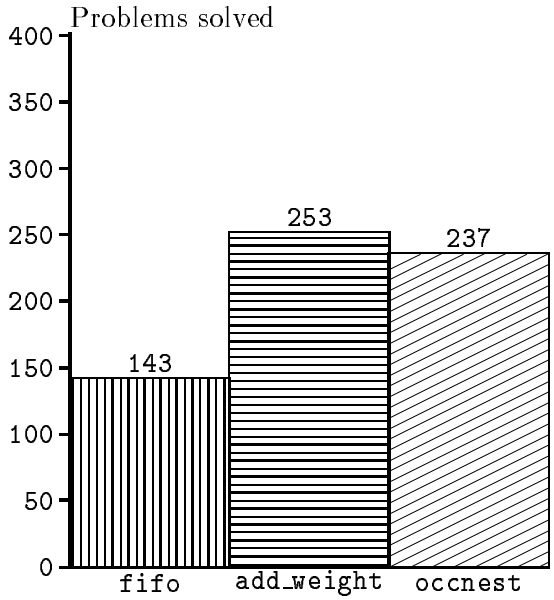


Figure 2.3: Comparison of conventional strategies

Strategy	`fifo`	`add_weight`	`occnest`
Problems solved	143	253	237
Time taken (seconds)	1040.85	2011.10	1906.22

Note: 403 problems of the TPTP library had to be proven within a time limit of 180 seconds for each problem.

Table 2.2: Performance of conventional strategies

2.4.3 Goal-Oriented Evaluation - `occnest`

A different approach for an evaluation function is `occnest` [DF94], which, contrary to `add_weight`, is goal-oriented and hand-tuned for the domain of lattice ordered groups.

Definition 20 (`occ` and `nest`)
Let $f \in F$ be a function symbol and $t \in Term(F, V)$ a term. Then the number of occurrences $occ(f, t)$ and the nesting $nest(f, t)$ are defined as follows:

$$occ(f,t) = \begin{cases} 0 & \text{if } t \in V \\ occ(f,t_1) + \cdots + occ(f,t_n) & \text{if } t \equiv g(t_1,\ldots,t_n), g \not\equiv f \\ 1 + occ(f,t_1) + \cdots + occ(f,t_n) & \text{if } t \equiv f(t_1,\ldots,t_n) \end{cases} \tag{2.6}$$

$$nest(f,t) = \begin{cases} 0 & \text{if } f \text{ is a constant} \\ hnest(f,t,0,0) & \text{otherwise} \end{cases} \tag{2.7}$$

$$hnest(f,t,cur,abs) = \begin{cases} \max(\{cur, abs\}) \\ \quad \text{if } t \text{ is a variable of a constant} \\ \max(\{hnest(f,t_i,0,max(\{cur,abs\}))|\ 1 \le i \le n\}) \\ \quad \text{if } t \equiv g(t_1,\ldots,t_n), f \not\equiv g \\ \max(\{hnest(f,t_i,cur+1,abs)|\ 1 \le i \le n\}) \\ \quad \text{if } t \equiv f(t_1,\ldots,t_n) \end{cases} \tag{2.8}$$

Definition 21 (`occ` and `nest` for equations)
occ and *nest* are extended to equations by computing the maximum of the two terms.

$$occ(f, s = t) = \max(\{occ(f,s), occ(f,t)\}) \tag{2.9}$$
$$nest(f, s = t) = \max(\{nest(f,s), nest(f,t)\}) \tag{2.10}$$

Definition 22 (`occnest`)
Let $s = t$ be a critical pair and $u = v$ the goal to be proven. $D \subseteq F$ is a subset of F containing the function symbols that are supposed to contribute to `occnest`$(s = t)$. Finally, `occnest` is defined by multiplying the weight of $s = t$ with a value describing the difference of the structural complexity of $s = t$ and the goal $u = v$. ψ is needed to ensure that the factor remains positive.

$$\texttt{occnest}(s = t) = \texttt{add\_weight}(s = t) \cdot \prod_{f \in F} m_f \tag{2.11}$$

with

$$m_f = \begin{cases} 1 \\ \quad \text{if } f \notin D \\ \psi(occ(f, s = t) - occ(f, u = v)) \cdot \psi(nest(f, s = t) - nest(f, u = v)) \\ \quad \text{otherwise} \end{cases} \tag{2.12}$$

and

$$\psi(x) = \begin{cases} 1 & \text{if } x \leq 0 \\ x+1 & \text{otherwise} \end{cases} \tag{2.13}$$

`occnest` clearly outperforms `add_weight` in the domain of groups (compare [Sch95] and [DS96a]). However, it is less successful in a general set of problems like the TPTP library (see table 2.2).

There is still no answer to the question of whether `occnest` is fair or not. Thus, according to theorem 4, the completeness of `occnest` is doubtful. Figure 2.3 shows the strong impact of heuristics on the proof success.

2.5 The TEAMWORK method

Besides the various strong heuristics, DISCOUNT's main strength lies in the distribution of proving procedures. Actually, DISCOUNT was originally intended as an experimental system for studying the TEAMWORK method.

Inspired by teams of human experts, the TEAMWORK method [AD93] implements distributed deduction, using multiple processes running on different processors. A *team*'s head is the *supervisor*, who selects *experts* to work on a specific task. This judgement is based on the conclusions drawn by the *referees* belonging to the experts. Figure 2.4 shows the basic structure of a team.

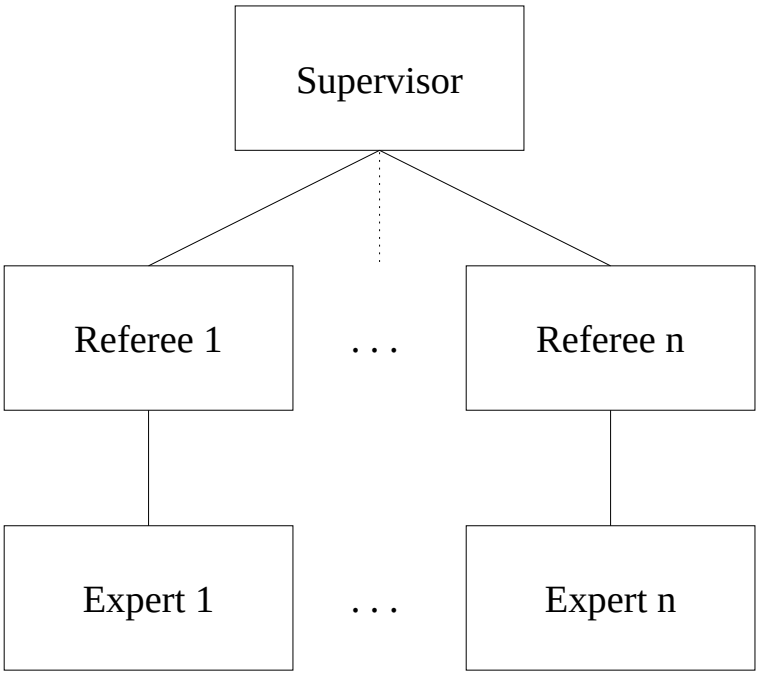


Figure 2.4: TEAMWORK team

Experts work on their own. Results are exchanged only at *team meetings*, and a new team is put together by the supervisor.

A completion based prover using TEAMWORK is complete if several fairness criteria are fulfilled (compare [AD93] for a proof). TEAMWORK is a successful extension to DISCOUNT. However, it is of no further interest in this work as it does not affect the example selection at all.

Chapter 3

Machine Learning Support

Analysis of proofs has indicated that there are regularities among proofs of identical and sometimes even different domains. Similarities with existing proofs can be taken into consideration when proving new problems, thus learning from theorems already proven. Several proofs share common lemmas, emphasizing the impression that reusing acquired proof knowledge can accelerate the proof process.

After introducing a way to extract proof knowledge and explaining the structure of the knowledge base, this chapter presents two approaches for learning from recorded knowledge. The first one is simple pattern memorization, preferring critical pairs that have the same *term patterns* as equations that have been used in successful proofs before. The second strategy is learning by term space mapping, a refined version of learning by term evaluation trees. This technique abstracts proof knowledge by building a tree structure, that is applied to evaluate critical pairs.

3.1 Extracting Proof Knowledge

In order to better understand automatically generated proofs, the *Proof Communication Language* PCL was developed ([DS94] and [DS96b]). PCL commands describe a proof protocol by using a plain notation. A unique identifier is followed by a *fact* (an equation or a rule) and an expression representing the fact's origin (the way it was created and references to utilized facts). The entire expression is called a PCL *step*.

PCL listings are prover-independent (besides DISCOUNT, PCL is used by Waldmeister [HBF96] and various other programs as well) and purely sequential[1].

A major problem with step by step listings of the proof is the overwhelming amount of data produced. Many proof protocols exceed hundred thousand steps. However, only a tiny fraction of these steps is necessary to prove the problem. A proof system leaves many dead ends in the search space.

[1]However, it is possible to describe distributed proofs (e.g., proofs by DISCOUNT using TEAMWORK) as well.

Various tools have been developed to work on PCL listings generated by a prover. Two of them are `mextract` and `lemma`.

`mextract` prunes a PCL listing down to the necessary steps (steps on the *proof path*) by tracking the listing backward, starting with the goal. This results in a listing that is by margins smaller than the original one. For example, the complete PCL file for the Lusk6 problem[2] consists of 387,273 steps, whereas the pruned listing contains only 190 steps. Most pruned proof protocols do not contain more than at most a few hundred steps. The extracted proofs allow a much easier analysis of the proofs by both humans and the prover itself (to learn from previous proofs). Facts close to the proof path (up to a certain number of edges) can be extracted too. This option is required to obtain negative facts for the learning component.

The `lemma` program marks especially important proof steps in order to make further investigations even easier and improve the proof presentation (see appendix A for a readable proof containing lemmas).

The huge difference between the lengths of original and pruned PCL listings once again emphasizes the importance of a heuristic controlling the progress in the search space.

3.2 The Knowledge Base

DISCOUNT's learning scheme is represented in figure 3.1. The PCL protocol of a successful proof is pruned by `mextract` and then handed to `kb_insert`, which computes the representative term patterns of the included facts and inserts the modified facts with accompanying data into the knowledge base. The facts collected in the knowledge base are then utilized to evaluate new critical pairs while trying to prove another problem.

If the learning component deals with positive *and* negative examples, facts close to the proof path will be extracted and annotated as negative, whereas facts that belong to the proof are marked as positive.

To provide an abstraction of the signature of a particular example without having to compute a potentially expensive (in terms of processor time) signature match, *representative term patterns* have been developed.

Definition 23 (Representative term patterns)
Let $F = \{f_{ij}\}$ with $i \in \mathbb{N}_0, j \in \mathbb{N}$ and $V = \{x_1, x_2, \ldots\}$ be two disjoint sets of symbols that comply with the following total orderings:

$$x_i >_V x_j \quad \Leftrightarrow \quad i > j \tag{3.1}$$

$$f_{ij} >_F f_{kl} \quad \Leftrightarrow \quad (i > k) \vee (i = k \wedge j > l) \tag{3.2}$$

The representative term pattern $\mathtt{pat}(t)$ for a given term t is constructed by substituting the function symbols and the variables in t with symbols from F and V. The variables are substituted with symbols from V in their order of appearance in t. The resulting term is called *variable normalized*. The jth

[2] A ring with $x^3 = x$ is Abelian [LO82].

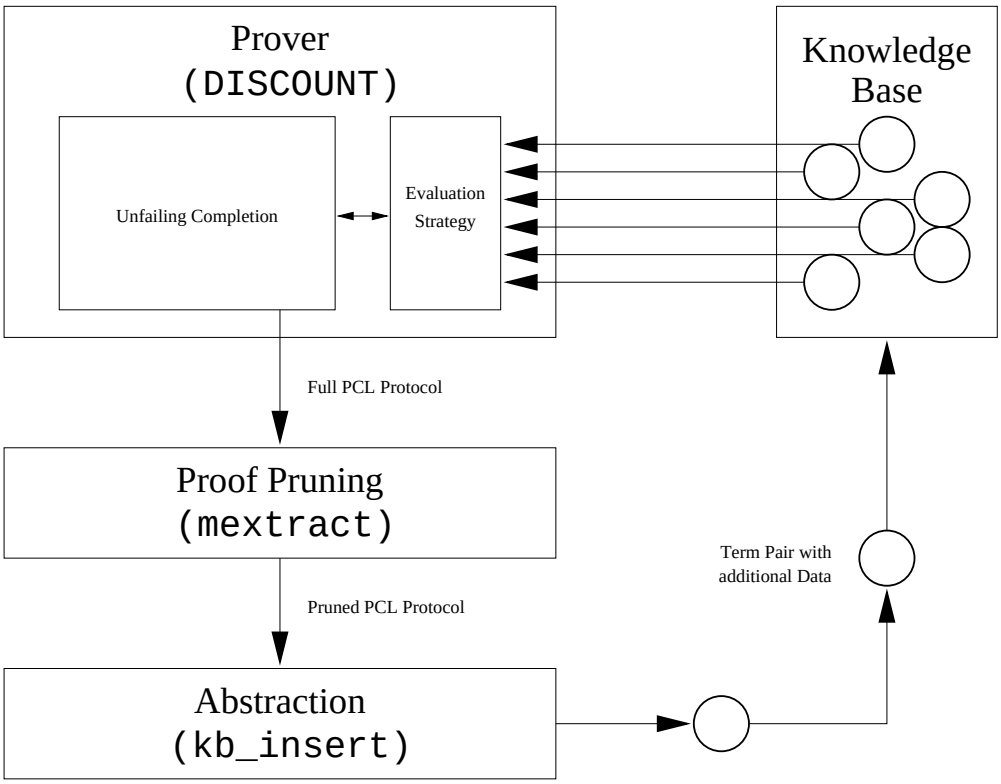


Figure 3.1: Learning in equational deduction

function symbol of arity i in t is replaced by the symbol f_{ij} from F[3]. For instance, $\mathtt{pat}(f(g(a), g(x))) = f_{21}(f_{11}(f_{01}), f_{11}(x_1))$.

In order to be able to compute representative term patterns of equations, we define a quasi-ordering on terms.

Definition 24 (The quasi-ordering $\geq_{pat}$)
Assuming $f_{ij} > x_k$ for all $f_{ij} \in F$ and all $x_k \in V$, the two orderings on F and V produce a total ordering $>$ on $F \cup V$, which can be easily generalized to a total ordering $>_{pat}$ on term patterns ($\overset{*}{>}_{pat}$ is the lexicographical extension of $>_{pat}$).

$$\begin{aligned} s >_{pat} t \quad \Leftrightarrow \quad & (s, t \in V \wedge s >_V t) \vee \\ & (s = f(s_1, s_2, \ldots, s_n) \wedge t \in V) \vee \\ & (s = f(s_1, s_2, \ldots, s_n) \wedge t = g(t_1, t_2, \ldots, t_m) \wedge f >_F g) \vee \\ & (s = f(s_1, s_2, \ldots, s_n) \wedge t = f(t_1, t_2, \ldots, t_n) \wedge \\ & (s_1, s_2, \ldots, s_n) \overset{*}{>}_{pat} (t_1, t_2, \ldots, t_n)) \end{aligned} \tag{3.3}$$

This ordering is then extended to a quasi-ordering on the whole set of terms.

$$s \geq_{pat} t \Leftrightarrow \mathtt{pat}(s) \geq_{pat} \mathtt{pat}(t) \tag{3.4}$$

[3]This is a more specialized version of the definition than the original one [Sch95]. It was first presented in [Sch98].

Definition 25 (Representative term patterns for equations)
Finally, representative term patterns for equations $\texttt{pat}(s = t)$ are obtained by orienting the equation according to $>_{pat}$ and then substituting the function symbols and variables in $s = t$ with symbols from F and V, treating the equation as a single term.

$\texttt{pat}(s = t) = \texttt{pat}(t = s)$ holds because either $s >_{pat} t$ or $t >_{pat} s$ or $\texttt{pat}(s) \equiv \texttt{pat}(t)$.

3.3 Term Pair Retrieval

The first, straight-forward learning evaluation strategy called `global_learn` simply searches the knowledge base for a term pattern identical with the one to be evaluated and obtains the annotation to compute the critical pair's weight. In the actual implementation, the complete file `cdata` (or `pdata` if only lemmas are desired) is read, and an AVL tree containing the facts is built. This tree is scanned for `pat(s=t)` each time a term pair $s = t$ has to be evaluated.

Definition 26 (`global_learn`)
In case the search for the term pattern is successful the following weight is assigned to the equation:

$$\begin{aligned}\texttt{global\_learn}(s = t) \quad = \quad & \texttt{add\_weight}(s = t) - \\ & (sc \cdot (w_{to} \cdot to + w_{pr} \cdot pr + w_{av} \cdot av + w_{gd} \cdot gd))\end{aligned} \tag{3.5}$$

to is the number of times the fact has been referenced in any proof, pr is the number of proofs that contained the fact, av is the average number of applications it had in a proof, and gd is the average distance to the goal of the fact. w_{to}, w_{pr}, w_{av} and w_{gd} are weights for the four parameters, while sc serves as a scaling factor for the whole learning component of the evaluation function.

If the search for the term pattern fails, a conventional estimation of the quality of the equation is used.

$$\texttt{global\_learn}(s = t) = \texttt{add\_weight}(s = t) + pen \tag{3.6}$$

pen is used to prevent, term pairs not contained in the knowledge base from being rated too high; learned term pairs are preferred.

Theorem 7
`global_learn` is fair.

Proof: `global_learn` yields integers by definition. There is a finite number of function symbols F and variables V in DISCOUNT. The representative generalization of a term pair contains the same number of symbols as the original term pair. Thus, only a finite set of equations shares the same representative term pattern. This means that only a finite number of equations can be evaluated using (3.5). Any finite set has a minimum.

We call this minimum a. The minimum of the backup strategy in (3.6) is pen. Concluding, the lower bound b of `global_learn` is $\min(a, pen)$.

In order to apply theorem 4, it remains to be shown that only a finite number of equations is mapped onto the same value. This is true because only finitely many equations are mapped by (3.5) at all. The backup strategy (3.6) maps only finite numbers of equations on the same value (theorem 6). Thus, this holds for `global_learn` as well.

3.4 Term Pair Abstraction

`global_learn` is only capable of evaluating critical pairs that are contained in the knowledge base. Other pairs are treated by the conventional strategy `add_weight`. It is clearly more desirable to be able to evaluate more equations, including new term pairs. This can be achieved by abstracting from the representative term patterns.
The strategy `tsm_learn` compiles knowledge derived by generalizing the term patterns into a huge, recursive tree structure called *term space map* (TSM)[4].

Definition 27 (TSMs and TSAs)
A term space map tsm is a non-empty set of *term space alternatives* (TSAs) $\{tsa_1, \ldots, tsa_n\}$. A TSA is a tuple $(index, (tsm_1, \ldots, tsm_n), info)$, whose $index$ is an index that depends on the definition of the index function (see definition 28); tsm_i is the TSM belonging to the ith argument of f; and $info$ is an annotation vector storing information about the term, which is required for term evaluation. The whole set of TSMs is called TSM; the set of TSAs is called TSA.

We use the following notations to represent TSMs[5]:

- $tsm.TSA$ is the set of TSAs belonging to tsm
- $tsm.n$ is the number of elements in $tsm.TSA$
- $tsa.n$ is the arity of the function represented by tsa
- $tsa.tsm[i]$ is the ith TSM belonging to tsa
- $tsa.i$ is the index of tsa
- $tsa.info$ is the information vector annotated to tsa

[4] TSMs are generalizations of *term evaluation trees* (TETs), introduced in [Sch95].
[5] Please note that TSM or TSA (in capital letters) stands for a set of TSMs or TSAs, whereas tsm and tsa are single elements (if not stated otherwise, TSM and TSA denote the entire set of TSMs and TSAs).

Definition 28 (Index function)
The index function `index` is used as a partition criterion for TSMs. $f \in F$ is an arbitrary function symbol, $v \in V$ a variable and $t \in Term(F,V)$ a term.

$$\begin{array}{rcll} index : Term(F,V) & \mapsto & I & \\ index(f(t_1,\ldots,t_n)) & = & f & (I = F \cup V) \\ index(v) & = & v & \end{array} \tag{3.7}$$

This index function is currently used in the implementation. Other possible index functions are

$$\begin{array}{rcll} index(f(t_1,\ldots,t_n)) & = & n & (I = \mathbb{N})^6 \\ index(v) & = & 0 & \end{array} \tag{3.8}$$

or

$$index(t) \quad = \quad t \quad (I = Term(F,V)) \tag{3.9}$$

Definition 29 (Inserting terms into TSMs)
Let $T = (tsa_1,\ldots,tsa_m)$ be a TSM and let $t \equiv f(t_1,\ldots,t_n)$ be a ground term. $info \in INFO$ is a vector of additional data about a term, containing, e.g., the number of positive and negative proof references. The tuple $(t, info)$ is called *term with additional data.*

$\texttt{ins} : TSM \times (Term(F) \times INFO) \mapsto TSM$ maps a TSM and a ground term with additional data to a new TSM. If there is no tsa_i with the index $index(t)$, a new TSA is added to T and the subterms t_i are included recursively.

$$\begin{array}{l} \texttt{ins}(T,(t,info)) := T \cup \\ \{(index(t),(\texttt{ins}(\{\},(t_1,info)),\ldots,\texttt{ins}(\{\},(t_n,info))),info)\} \end{array} \tag{3.10}$$

In case there is a $tsa_i \equiv (index(t),(tsm_1,tsm_2,\ldots,tsm_n),info')$, the annotations are added and the subterms are added by recursion.

$$\begin{array}{l} \texttt{ins}(T,(t,info)) := \{tsa_1,\ldots,tsa_{i-1},tsa_{i+1},\ldots,tat_m\} \cup \\ \{(index(t),(\texttt{ins}(tsm_1,(t_1,info)),\ldots,\texttt{ins}(tsm_n,(t_n,info))),info + info')\} \end{array} \tag{3.11}$$

`ins` can be applied to non-ground terms by treating variables as functions of arity 0. This extends `ins` to $\texttt{ins} : TSM \times (Term(F,V) \times INFO) \mapsto TSM$.

`ins` is associative, i.e. a TSM is defined for a *set* of terms and does not depend on the order in which the terms are inserted.

Figure 3.2 shows an example TSM that was built by inserting five terms with a single integer annotation (e.g., the number of proof references).

The insertion function is defined for equations by inserting the two terms separately.

$$\texttt{ins}(T,(s=t,info)) \quad = \quad \texttt{ins}(\texttt{ins}(T,(s,info)),(t,info)) \tag{3.12}$$

Before an equation $s = t$ is inserted into a TSM, its representative term pattern $\texttt{pat}(s = t)$ is computed.

[6] In this special case, a TSM is a TET.

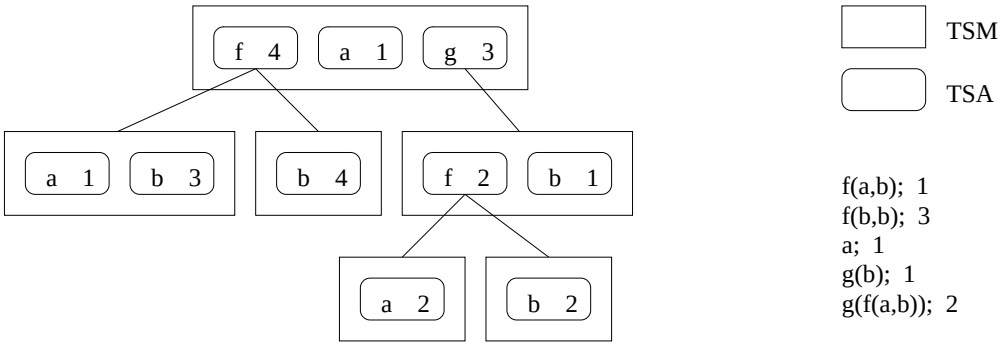


Figure 3.2: An example TSM

Definition 30 (TSM-based term pair measurement)
The weight of a term t evaluated using a TSM tsm is assigned by the function $\texttt{eval} : Term(F,V) \times TSM \mapsto \mathbb{R}^+$.

$$\text{if } \exists tsa \in tsm.TSA: \quad tsa.i = index(t)$$

$$\texttt{eval}(t, tsm) = \begin{cases} 2 \cdot e(tsa) + \sum_{i=1}^{tsa.n} \texttt{eval}(t_i, tsa.tsm[i]) & \text{if } t \equiv f(t_1, \ldots, t_n) \\ e(tsa) & \text{otherwise} \end{cases} \tag{3.13}$$

with

$$e(tsa) = (1 - (pl \cdot tsa.info.pos)) \cdot (1 + (nl \cdot tsa.info.neg)) \tag{3.14}$$

$tsa.info.pos$ is the relative quota of positive proof occurrences. It is defined as the number of positive proof references of the single TSA node tsa, divided by the accumulated number of positive proof references of all TSAs belonging to the same TSM. $tsa.info.neg$ is defined similarly for negative proof occurrences. pl and nl are corresponding weighting factors, lower than or equal to 1.

In case t cannot be mapped on tsm, the backup strategy `add_weight` is used.

$$\text{if } \nexists tsa \in tsm.TSA: \quad tsa.i = index(t)$$

$$\texttt{eval}(t, tsm) = \texttt{add\_weight}(t) \tag{3.15}$$

As usual, the weight of a term pair $\texttt{tsm\_learn} : (Term(F,V) \times Term(F,V)) \times TSM \mapsto \mathbb{R}^+$ is computed by adding the single term weights.

$$\texttt{tsm\_learn}(s = t, tsm) = \texttt{eval}(s, tsm) + \texttt{eval}(t, tsm) \tag{3.16}$$

To give an example, we compute the weight of $g(f(a,a))$ according to the TSM of figure 3.2. pl is set to 1 and the number of negative proof references is 0 for all TSA nodes.
$\texttt{eval}(g(f(a,a)), tsm) = 2(1-\frac{3}{8}) + 2(1-\frac{2}{3}) + 1 - \frac{2}{2} + 1 = \frac{5}{3} \approx 1.67$
A default term weight is assigned to unknown subterms. The weight of $g(h(a,a))$, for instance, is greater than the weight computed above.
$\texttt{eval}(g(h(a,a)), tsm) = 2(1-\frac{3}{8}) + \texttt{add\_weight}(h(a,a)) = \frac{21}{4} = 5.25$

Theorem 8
`tsm_learn` is fair.

Proof: A TSM is a finite structure and the terms in the DISCOUNT system are built from finite sets of symbols. Only a finite number of equations share the same representative term pattern. Thus, it is only possible to evaluate a finite number of terms using the TSM-based measurement defined in (3.13). The backup strategy for unknown subterms `add_weight` is fair according to theorem 6. As a result, $\forall e \in Term(F,V) \times Term(F,V)$: $\{e' |\ \texttt{tsm\_learn}(e') \leq \texttt{tsm\_learn}(e)\}$ is finite and `tsm_learn` is fair due to theorem 3.

A slight variation of the TSM-based strategy described in this section is called *ordered TSM learning*. This method builds two TSMs; tsm_l for the left and tsm_r for the right side of the oriented (according to $>_{pat}$) equations. The evaluation function evaluates each term of a term pair on the appropriate TSM.

Definition 31 (`otsm_learn`)
Let $u = v$ be the representative term pattern of $s = t$ ($(u = v) = \texttt{pat}(s = t)$).

$$\texttt{otsm\_learn}(s = t, tsm_l, tsm_r) = \texttt{eval}(u, tsm_l) + \texttt{eval}(v, tsm_r) \qquad (3.17)$$

Theorem 9
`otsm_learn` is fair.

Proof: `otsm_learn` is fair due to exactly the same reasons specified in theorem 8.

Chapter 4

Example Selection

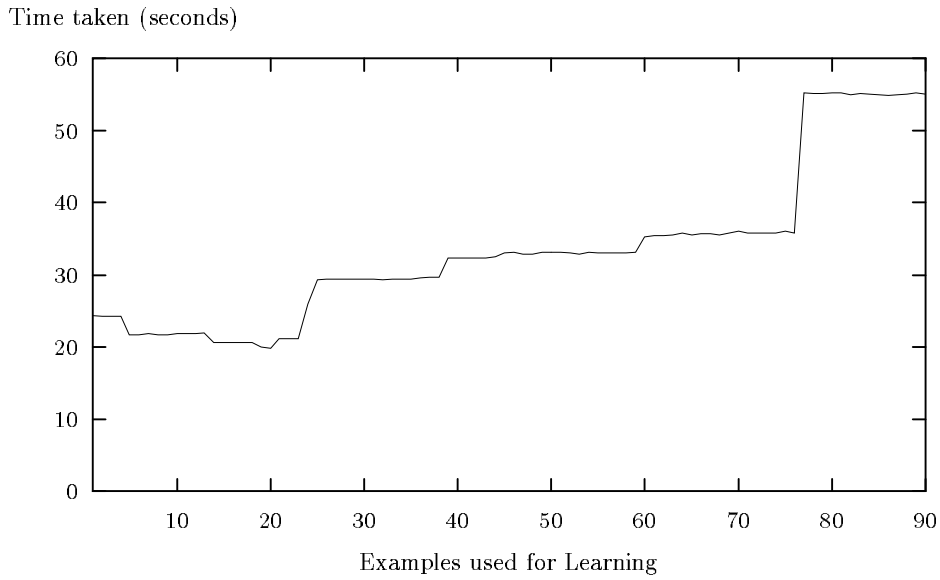
Using as much knowledge as is available can mislead the proof process. In this chapter, we develop criteria for example resemblance and present and evaluate three selection strategies based on these criteria. The first one uses simple features like number of axioms and average term depth of the axioms, whereas the second one is based on a similarity measure on TSMs (containing the axioms), developed for this purpose. A third strategy combines both measures.

4.1 The Need for Restriction

Although `tsm_learn` performs very well, the strategy suffers from misleading knowledge provided by facts from proofs that required a different solution than the problem to be solved. In contrast to `global_learn`, `tsm_learn`'s performance is not increased by every new fact inserted in the knowledge base. There is a certain degree of knowledge saturation (see figure 4.1).

To avoid knowledge overkill, we preselect fitting examples from the knowledge base (see figure 4.2). A selection strategy sorts the example proofs according to some measure of distance to the specification of the problem to be proven. Only a fraction of the proofs in the knowledge base, limited to a fixed number and a certain degree of similarity, is then used for the learning strategy.

Two parameters (`max_examples` and `max_delta`) control this process. Up to `max_examples` examples, whose distance to the current problem's specification is less than or equal to `max_delta`, are selected and inserted into the TSM.



Note: Ordered TSM learning with δ_{MIX} example selection was used to prove `BOOOO8-2`

Figure 4.1: Knowledge excess

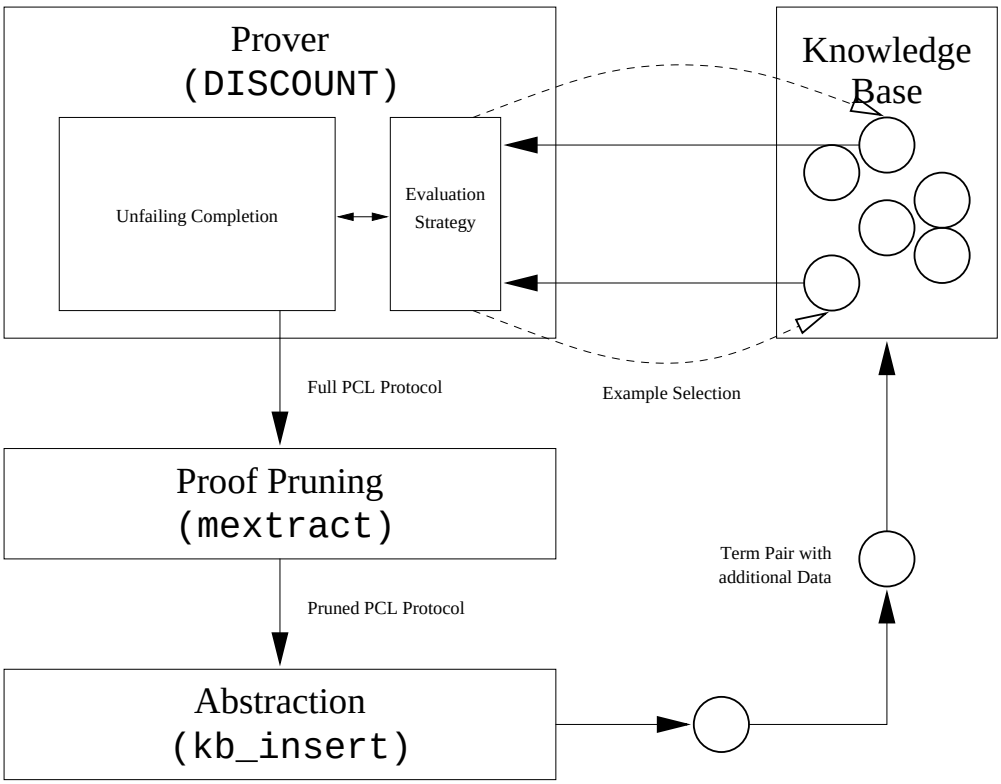


Figure 4.2: Learning with example selection

4.2 Metric Spaces

As the problem distance measures are supposed to be metrics, some theory about metric spaces is required (see [Duf95] for details about metrics and topology in general).

Definition 32 (Metrics and metric spaces)
$d : X \times X \mapsto \mathbb{R}^+$ is a metric on X if the following axioms hold. The tuple (X, d) is called a metric space.

$$\begin{array}{rcll} d(a,b) = 0 & \Leftrightarrow & a = b & \\ d(a,b) & = & d(b,a) & \text{(Symmetry)} \\ d(a,c) & \leq & d(a,b) + d(b,c) & \text{(Triangle inequality)} \end{array} \tag{4.1}$$

It can easily be shown that scaled metrics and sums of metrics are metrics too.

Theorem 10
If (X, d) is a metric space, $(X, w \cdot d)$ is a metric space as well $(w \in \mathbb{R}^+)$.

Proof:

- $w \cdot d(a,b) = 0 \Leftrightarrow d(a,b) = 0 \Leftrightarrow a = b$
- $w \cdot d(a,b) = w \cdot d(b,a) \Leftrightarrow d(a,b) = d(b,a)$
- $w \cdot d(a,c) \leq w \cdot d(a,b) + w \cdot d(b,c) \Leftrightarrow d(a,c) \leq d(a,b) + d(b,c)$

Theorem 11
If (X, d_1) and (X, d_2) are metric spaces, $(X, d_1 + d_2)$ is a metric space as well.

Proof:

- $d_1(a,b) + d_2(a,b) = 0 \Leftrightarrow d_1(a,b) = 0 \wedge d_2(a,b) = 0 \Leftrightarrow a = b$
- $d_1(a,b) + d_2(a,b) = d_1(b,a) + d_2(a,b) = d_1(b,a) + d_2(b,a)$
- $d_1(a,c) + d_2(a,c) \leq d_1(a,b) + d_2(a,b) + d_1(b,c) + d_2(b,c) \Leftrightarrow$
  $d_1(a,c) + d_2(a,c) \leq \underbrace{d_1(a,b) + d_1(b,c)}_{\geq d_1(a,c)} + \underbrace{d_2(a,b) + d_2(b,c)}_{\geq d_2(a,c)} \Leftrightarrow$
  $d_1(a,c) \leq d_1(a,b) + d_1(b,c) \wedge d_2(a,c) \leq d_2(a,b) + d_2(b,c)$

Note: This obviously also holds for metrics on different sets (X, d_1) and (Y, d_2), if the distance of two pairs $a = (x_1, y_1)$ and $b = (x_2, y_2)$ is defined as the sum of the components distances $d(a,b) = d_1(x_1, x_2) + d_2(y_1, y_2)$. The resulting metric space is $(X \times Y, d)$.

4.3 Criteria for Example Resemblance

To be able to decide whether an example proof's facts should be inserted into the TSM, criteria are needed which describe the similarity of problem specifications. We chose the following:

- Number of axioms (NA)
- Average term depth of the axioms (AD)
- Standard deviation of the term depth of the axioms (DD)
- Depth of the goal (GD)
- Number of function symbols of a given arity: arity frequencies ($\overrightarrow{AF}$)

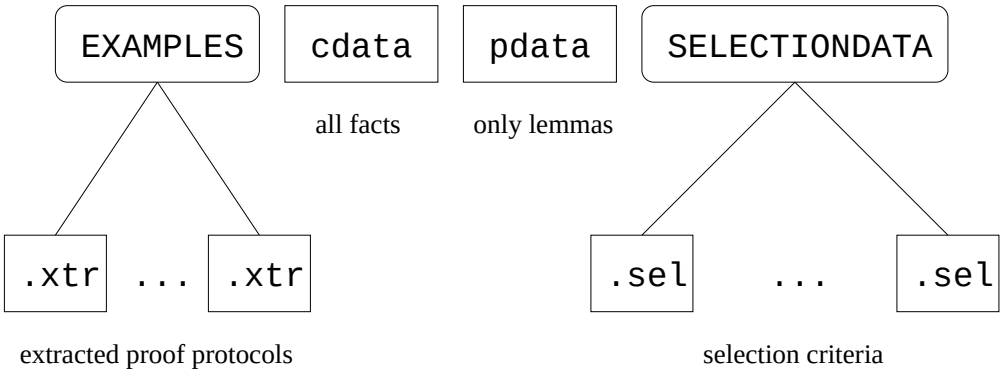


Figure 4.3: New structure of the knowledge base

These features and the representative term patterns of the axioms themselves are saved for each problem in the `SELECTIONDATA` folder of the knowledge base (see figure 4.3). The figure omits the folders `SPECDOMS` and `GOALDOMS` and the corresponding files `specdoms` and `goaldoms`. This data was used to preselect examples with identical specifications or goals for `global_learn`. As `tsm_learn` is mainly used now, these files are of no interest to us. Figure 4.4 shows an example `.sel`-file.

4.4 Test Run Setup

The subset of all unit-equality problems with universally quantified goals from version 2.1 of the TPTP problem library ([SS97]) was used to evaluate the selection strategies. All in all, there are 403 problems of which 341 are automatically provable, 57 are theorems that have not yet been proven by current ATP systems, and 5 are open problems (for more details about the used problems, see appendix B).

The knowledge base contained the positive facts of 201 proofs, all that DISCOUNT was able to solve within 180 seconds using the regular `add_weight` strategy.

```
BOO001-1  BOO002-1  BOO002-2  BOO003-2  BOO003-4  BOO004-2
BOO004-4  BOO005-2  BOO005-4  BOO006-2  BOO006-4  BOO007-2
BOO007-4  BOO009-2  BOO009-4  BOO010-2  BOO010-4  BOO011-2
BOO011-4  BOO012-2  BOO012-4  BOO013-2  BOO013-4  BOO014-2
```

```
Number_of_Axioms: 10
Average_Depth: 1.250000
Depth_Standard_Deviation: 0.829156
Goal_Depth: 2
Max_Arity: 2
Arity_Frequencies: 3 1 2

Axioms:
x1 = f2_1(x1,x1)
x1 = f2_1(f0_1(),x1)
x1 = f2_1(x1,f0_1())
f0_1() = f2_1(f1_1(x1),x1)
f0_1() = f2_1(x1,f1_1(x1))
f2_1(x1,x2) = f2_1(x2,x1)
f2_1(x1,f2_1(x2,x3)) = f2_1(f2_1(x1,x2),x3)
f2_1(x1,f2_1(x2,x3)) = f2_1(f2_1(x1,x2),x3)
f2_1(x1,f2_2(x2,x3)) = f2_2(f2_1(x1,x2),f2_1(x1,x3))
f2_1(f2_2(x1,x2),x3) = f2_2(f2_1(x1,x3),f2_1(x2,x3))
```

Figure 4.4: An example `.sel` file: `lusk3.sel` (A ring with $x^2 = x$ is Abelian [LO82])

```
BOO014-4  BOO015-2  BOO015-4  BOO016-2  BOO017-2  BOO018-4
COL003-12 COL003-13 COL003-14 COL003-15 COL003-16 COL004-3
COL042-6  COL042-7  COL042-8  COL042-9  COL058-2  COL058-3
COL059-1  COL060-2  COL060-3  COL061-2  COL061-3  COL062-2
COL062-3  COL063-2  COL063-3  COL063-4  COL063-5  COL063-6
COL064-10 COL064-11 COL064-2  COL064-3  COL064-4  COL064-5
COL064-6  COL064-7  COL064-8  COL064-9  COL066-2  COL066-3
GRP001-2  GRP001-4  GRP002-2  GRP002-3  GRP002-4  GRP010-4
GRP011-4  GRP012-4  GRP014-1  GRP022-2  GRP023-2  GRP114-1
GRP115-1  GRP116-1  GRP117-1  GRP118-1  GRP119-1  GRP120-1
GRP121-1  GRP122-1  GRP136-1  GRP137-1  GRP138-1  GRP139-1
GRP140-1  GRP141-1  GRP142-1  GRP143-1  GRP144-1  GRP145-1
GRP146-1  GRP147-1  GRP148-1  GRP149-1  GRP150-1  GRP151-1
GRP152-1  GRP153-1  GRP154-1  GRP155-1  GRP156-1  GRP157-1
GRP158-1  GRP159-1  GRP160-1  GRP161-1  GRP162-1  GRP163-1
GRP165-1  GRP165-2  GRP166-1  GRP166-2  GRP166-3  GRP166-4
GRP167-1  GRP167-2  GRP167-5  GRP168-1  GRP168-2  GRP169-1
GRP169-2  GRP171-1  GRP171-2  GRP172-1  GRP172-2  GRP173-1
GRP174-1  GRP175-2  GRP175-3  GRP176-1  GRP176-2  GRP182-1
GRP182-2  GRP182-3  GRP182-4  GRP186-3  GRP186-4  GRP188-1
GRP188-2  GRP189-1  GRP189-2  GRP190-1  GRP190-2  GRP191-1
GRP191-2  LCL110-2  LCL112-2  LCL113-2  LCL114-2  LCL115-2
LCL116-2  LCL132-1  LCL133-1  LCL134-1  LCL135-1  LCL139-1
LCL140-1  LCL141-1  LCL153-1  LCL154-1  LCL155-1  LCL156-1
```

LCL157-1	LCL158-1	LCL159-1	LCL160-1	LCL161-1	LCL164-1
LDA001-1	LDA002-1	LDA007-3	RNG007-4	RNG008-3	RNG008-4
RNG008-7	RNG009-7	RNG011-5	RNG012-6	RNG013-6	RNG014-6
RNG015-6	RNG016-6	RNG017-6	RNG018-6	RNG023-6	RNG023-7
RNG024-6	RNG024-7	ROB002-1	ROB003-1	ROB004-1	ROB010-1
ROB013-1	SYN080-1	SYN083-1			

The specified times relate to a Linux system on a Pentium PC running at 233 MHz with 64 Mb of RAM, which was used as a testing platform for all experiments.

4.5 Feature-based Selection - δ_{FTR}

The first measure of distance computes the distance between two problem specifications by simply comparing their features.

4.5.1 The Metric δ_{FTR}

Let ma be the maximum arity of the functions. Then $AF(i)$ is defined as the number of occurrences of arity i within the function symbols and $\overrightarrow{AF}$ is a vector with the components $AF(i)$.

$$\overrightarrow{AF} = \begin{pmatrix} AF(0) \\ AF(1) \\ \cdots \\ AF(ma) \end{pmatrix} \tag{4.2}$$

Definition 33 (Feature vector)
The feature vector $\vec{f}$ is defined as follows:

$$\vec{f} = \begin{pmatrix} NA \\ AD \\ DD \\ GD \\ \overrightarrow{AF} \end{pmatrix} \tag{4.3}$$

The feature vector space is denoted as F.

Definition 34 (δ)
In order to be able to combine weighted differences of several features, we define a function δ that scales the distance of two features to the interval $[0;1]$.

$$\begin{aligned} \delta : \mathbb{R}^+ \times \mathbb{R}^+ &\mapsto [0;1] \\ \delta(a,b) &= \begin{cases} 0 & \text{if } a,b = 0 \\ \dfrac{|a-b|}{\max(a,b)} & \text{otherwise} \end{cases} \end{aligned} \tag{4.4}$$

Definition 35 (δ_{AF})
As the feature "Arity Frequencies" is a vector, the distance of two vectors $\overrightarrow{AF_1}$ and $\overrightarrow{AF_2}$ requires a special definition. δ_{AF} scales the sum of each arity distance to the interval $[0;1]$. As usual, $\pi_i(\vec{v})$ denotes the projection of the ith component of $\vec{v}$.

$$\delta_{AF}(\overrightarrow{AF_1}, \overrightarrow{AF_2}) = \frac{\sum_{i=0}^{ma} \delta(\pi_i(\overrightarrow{AF_1}), \pi_i(\overrightarrow{AF_2}))}{\max(0 \leq j \leq ma| \quad \pi_j(\overrightarrow{AF_1}) \neq 0 \vee \pi_j(\overrightarrow{AF_2}) \neq 0)} \tag{4.5}$$

Definition 36 (δ_{FTR})
Finally, the metric δ_{FTR} on the feature vector space is as follows:

$$\begin{array}{rcl} \delta_{FTR} : F \times F & \mapsto & [0;1] \\ \delta_{FTR}(\vec{f_1}, \vec{f_2}) & = & \frac{w_{NA} \cdot \delta_{NA} + w_{AD} \cdot \delta_{AD} + w_{DD} \cdot \delta_{DD} + w_{GD} \cdot \delta_{GD} + w_{AF} \cdot \delta_{AF}}{w_{NA} + w_{AD} + w_{DD} + w_{GD} + w_{AF}} \end{array} \tag{4.6}$$

We use the following abbreviations:
$\delta_{NA} = \delta(\pi_1(\vec{f_1}), \pi_1(\vec{f_2}))$, $\delta_{AD} = \delta(\pi_2(\vec{f_1}), \pi_2(\vec{f_2}))$, $\delta_{DD} = \delta(\pi_3(\vec{f_1}), \pi_3(\vec{f_2}))$, $\delta_{GD} = \delta(\pi_4(\vec{f_1}), \pi_4(\vec{f_2}))$ and $\delta_{AF} = \delta_{AF}(\pi_5(\vec{f_1}), \pi_5(\vec{f_2}))$.
w_{NA}, w_{AD}, w_{DD}, w_{GD} and w_{AF} are weighting factors for the corresponding features.

Theorem 12
δ_{FTR} is a metric on the feature vector space F.

Proof: Since δ_{FTR} is a scaled sum of δs, it suffices to show that δ is a metric on $\mathbb{R}^+$ (theorem 10 and 11). δ must fulfill the following axioms.

a) $\delta(a,b) = 0 \quad \Leftrightarrow \quad a = b$
b) $\delta(a,b) = \delta(b,a)$
c) $\delta(a,c) \leq \delta(a,b) + \delta(b,c)$

a) $\delta(a,b) = 0 \Leftrightarrow (\frac{|a-b|}{\max(a,b)} = 0 \vee a, b = 0) \Leftrightarrow |a-b| = 0 \Leftrightarrow a = b$
b) $\delta(a,b) = \frac{|a-b|}{\max(a,b)} = \frac{|b-a|}{\max(b,a)} = \delta(b,a)$
c) $\frac{|a-c|}{\max(a,c)} \leq \frac{|a-b|}{\max(a,b)} + \frac{|a-c|}{\max(a,c)}$

Case 1 $(a \geq b \geq c)$
$\frac{a-c}{a} \leq \frac{a-b}{a} + \frac{b-c}{b} \Leftrightarrow b(a-c) \leq b(a-b) + a(b-c) \Leftrightarrow$
$ab - bc \leq ab - b^2 + ab - ac \Leftrightarrow b(a-b) + c(b-a) \geq 0 \Leftrightarrow \underbrace{(a-b)}_{\geq 0}\underbrace{(b-c)}_{\geq 0} \geq 0$

Case 2 $(a \geq c \geq b)$
$\frac{a-c}{a} \leq \frac{a-b}{a} + \frac{c-b}{c} \Leftrightarrow c(a-c) \leq c(a-b) + a(c-b) \Leftrightarrow$
$ca - c^2 \leq ac - bc + ac - ab \Leftrightarrow c(c-b) + a(c-b) \geq 0 \Leftrightarrow \underbrace{(c-b)}_{\geq 0}\underbrace{(c+a)}_{\geq 0} \geq 0$

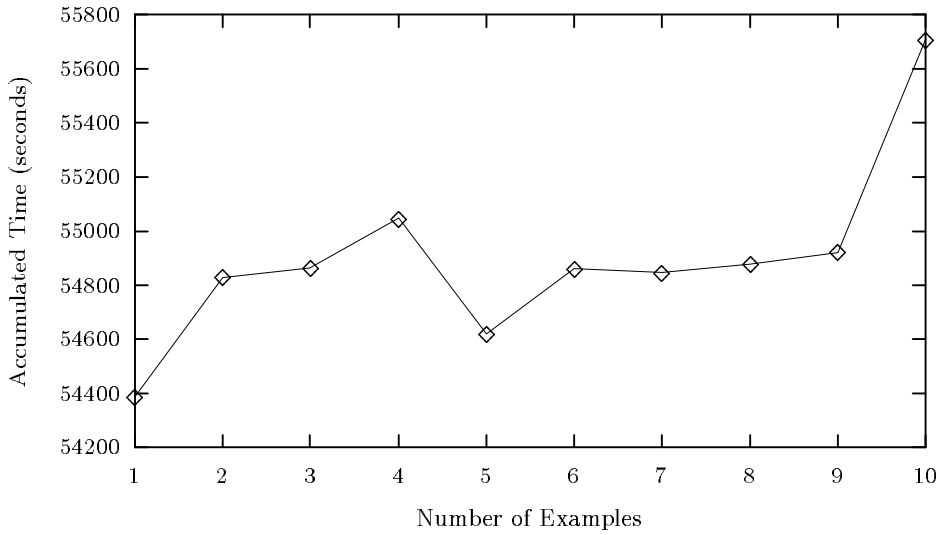
Case 3 ($b \geq a \geq c$)
$\frac{a-c}{a} \leq \frac{b-a}{b} + \frac{b-c}{b} \Leftrightarrow b(a-c) \leq a(b-a) + a(b-c) \Leftrightarrow$
$ab - bc \leq ab - a^2 + ab - ac \Leftrightarrow a(b-a) + c(b-a) \geq 0 \Leftrightarrow \underbrace{(b-a)}_{\geq 0}\underbrace{(a+c)}_{\geq 0} \geq 0$

Case 4 ($b \geq c \geq a$)
$\frac{c-a}{c} \leq \frac{b-a}{b} + \frac{b-c}{b} \Leftrightarrow b(c-a) \leq c(b-a) + c(b-c) \Leftrightarrow$
$bc - ab \leq bc - ac + bc - c^2 \Leftrightarrow c(b-c) + a(b-c) \geq 0 \Leftrightarrow \underbrace{(b-c)}_{\geq 0}\underbrace{(a+c)}_{\geq 0} \geq 0$

Case 5 ($c \geq a \geq b$)
$\frac{c-a}{c} \leq \frac{a-b}{a} + \frac{c-b}{c} \Leftrightarrow a(c-a) \leq c(a-b) + a(c-b) \Leftrightarrow$
$ac - a^2 \leq ac - bc + ac - ab \Leftrightarrow c(a-b) + a(a-b) \geq 0 \Leftrightarrow \underbrace{(a-b)}_{\geq 0}\underbrace{(a+c)}_{\geq 0} \geq 0$

Case 6 ($c \geq b \geq a$)
$\frac{c-a}{c} \leq \frac{b-a}{b} + \frac{c-b}{c} \Leftrightarrow b(c-a) \leq c(b-a) + a(c-b) \Leftrightarrow$
$bc - ab \leq bc - ac + bc - b^2 \Leftrightarrow c(b-a) - b(b-a) \geq 0 \Leftrightarrow \underbrace{(b-a)}_{\geq 0}\underbrace{(c-b)}_{\geq 0} \geq 0$

4.5.2 Parameter Test Runs



Note: *Accumulated time* is the sum of all single proving times, assuming 360 seconds for unsolved problems.

Figure 4.5: Influence of `max_examples` on δ_{FTR}

Figure 4.5 shows the performance of δ_{FTR} applying different number of examples. The strategy is very fast using only one example. A couple of problems (COL003-12 to COL003-15) that could not be proven with any other parameter set are proven very quickly (under 11 seconds per problem). However, this setting breaks down on several other demanding problems.

As a result, `max_examples`=5 is the most successful number of examples in our testing environment. It proves one example more than `max_examples`=1, even though the latter is slightly faster.

max_examples	max_delta	w_NA	w_AD	w_DD	w_GD	w_AF	Problems	Time
1	1	1	1	1	1	1	257	1828.13
2	1	1	1	1	1	1	257	2269.31
3	1	1	1	1	1	1	257	2302.81
4	1	1	1	1	1	1	256	2126.25
5	1	1	1	1	1	1	258	2418.19
6	1	1	1	1	1	1	257	2299.36
7	1	1	1	1	1	1	257	2287.26
8	1	1	1	1	1	1	257	2319.75
9	1	1	1	1	1	1	257	2362.35
10	1	1	1	1	1	1	254	2068.38
20	0.5	1	1	1	1	1	250	1796.33
1000	0.125	1	1	1	1	1	254	1859.45
10	0.5	0	1	1	1	1	254	1864.12
10	0.5	1	0	1	1	1	253	2098.07
10	0.5	1	1	0	1	1	252	1732.74
10	0.5	1	1	1	0	1	254	2182.04
10	0.5	1	1	1	1	0	252	1764.59
10	0.5	1	1	1	1	1	254	2065.38
10	0.5	1	1	1	2	3	250	1788.62
10	0.5	1	3	2	1	1	254	1910.50
10	0.5	1	4	1	1	1	254	1714.45
10	0.5	0	1	0	1	0	252	1716.62
10	0.5	0.5	4	2	1	1	254	1805.99
5	1	0.5	2	1	1	1	258	2425.20
5	1	0.5	4	2	1	1	255	2117.05
5	0.5	0	1	1	1	1	255	2075.64
5	0.5	1	1	1	1	1	258	2419.47

Table 4.1: Different parameter sets for δ_{FTR}

Various combinations of feature weights were tested, but yielded no performance gain.

4.6 TSM-based Selection - δ_{TSM}

The features described in the feature vector (4.3) represent a generalization of the actual problem specification, which is nothing but a set of equations. In

another context, structures that generalize equations have already been developed: Term Space Maps.

δ_{TSM} computes the difference of two problem specifications by building two TSMs of each set of axioms and then determining the distance of both TSMs according to a recursive measurement.

4.6.1 The Metric δ_{TSM}

To measure similarity of TSMs, we extend TSM to a metric space (TSM, δ_{TSM}). The function δ_{TSM} should be independent of the following attributes to allow fair comparison of TSMs:

- Depths of entire TSMs
- Number of TSAs per TSM
- Arity of TSAs

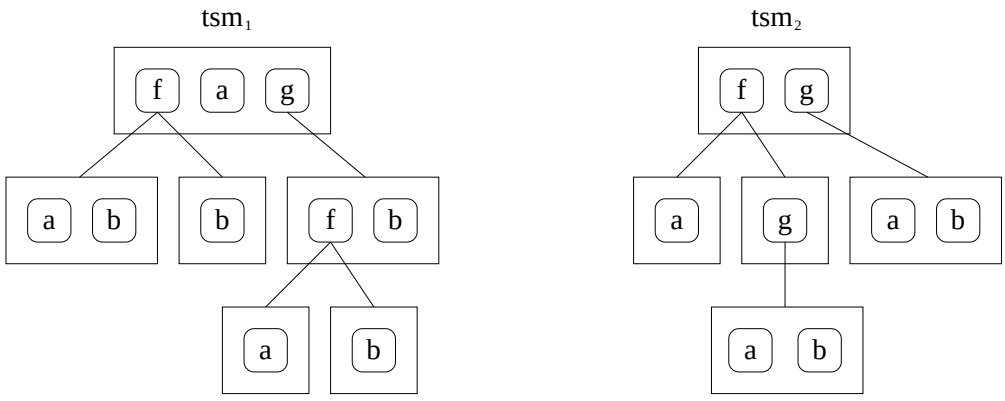
Using the TSM and TSA notations described in section 3.4, we define δ_{TSM}.

Definition 37 (δ_{TSM})
$\delta_{TSM} : TSM \times TSM \mapsto [0; 1]$

$$
\begin{aligned}
\delta_{TSM}(tsm_1, tsm_2) &= \begin{cases} \dfrac{\sum_{tsa \in tsm_1.TSA} \delta'_{TSM}(tsa, tsm_2.TSA)}{tsm_1.n} & \text{if } tsm_1.n \geq tsm_2.n \\ \dfrac{\sum_{tsa \in tsm_2.TSA} \delta'_{TSM}(tsa, tsm_1.TSA)}{tsm_2.n} & \text{otherwise} \end{cases} \\
\delta'_{TSM}(tsa, TSA) &= \begin{cases} \delta_{TSA}(tsa, tsa') & \exists tsa' \in TSA : \quad tsa.i = tsa'.i \\ 1 & \text{otherwise} \end{cases} \\
\delta_{TSA}(tsa_1, tsa_2) &= \begin{cases} \dfrac{\sum_{i=1}^{tsa_1.n} \delta_{TSM}(tsa_1.tsm[i], tsa_2.tsm[i])}{tsa_1.n} & \text{if } tsa_1.n \neq 0 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{4.7}
$$

Figure 4.6 illustrates how δ_{TSM} measures the distance between the example TSM presented in figure 3.2 and another TSM.

In order to be capable of proving some properties of the new distance measure, we define the following relation.



$$\delta_{TSM}(tsm_1, tsm_2) = \frac{\frac{\frac{0+1}{2}+1}{2} + 1 + \frac{\frac{1+0}{2}}{1}}{3} = \frac{3}{4}$$

Figure 4.6: Distance between two TSMs

Definition 38 (Mapping terms on TSMs)
We denote $t \propto tsm$, if t can be *completely mapped* on tsm.

$$\begin{aligned} \text{If } t = f(t_1, \ldots, t_n): \quad t \propto tsm \quad &\Leftrightarrow \quad (\exists tsa \in tsm.TSA: \quad index(t) = tsa.i) \wedge \\ &\quad\ (\forall i \quad 1 \leq i \leq n: \quad t_i \propto tsa.tsm[i]) \\ \text{If } t \in V: \quad t \propto tsm \quad &\Leftrightarrow \quad (\exists tsa \in tsm.TSA: \quad index(t) = tsa.i) \end{aligned} \tag{4.8}$$

Theorem 13
(TSM, δ_{TSM}) is a metric space.

Proof: δ_{TSM} has to fulfill the following axioms.

a) $\delta_{TSM}(tsm_1, tsm_2) = 0 \quad \Leftrightarrow \quad tsm_1 = tsm_2$

b) $\delta_{TSM}(tsm_1, tsm_2) = \delta_{TSM}(tsm_2, tsm_1)$

c) $\delta_{TSM}(tsm_1, tsm_3) \leq \delta_{TSM}(tsm_1, tsm_2) + \delta_{TSM}(tsm_2, tsm_3)$

a) We assume $tsm_1.n \geq tsm_2.n$.
$\delta_{TSM}(tsm_1, tsm_2) = 0 \Leftrightarrow \delta'_{TSM}$will never be 1 during the comparison $\Leftrightarrow$
$\forall t \in Term(F, V) \quad t \propto tsm_1: \quad t \propto tsm_2 \Leftrightarrow tsm_1 = tsm_2$
b) $\delta_{TSM}(tsm_1, tsm_2) = \delta_{TSM}(tsm_2, tsm_1)$ by definition.
c) We prove the triangle inequality by induction.
Induction start
Consider two TSMs tsm_1 and tsm_2 of depth 1. All TSAs in $tsm_1.TSA$ and $tsm_2.TSA$ represent either variables or functions symbols of arity 0. This implies that δ_{TSA} is 0 for all matching pairs of TSAs. δ'_{TSM} is 1 if the two TSMs do not share a given TSA, and 0 otherwise. In order to simplify the representation of δ_{TSM} for TSMs of depth 1, we introduce

the following index sets:

$$A = \bigcup_{tsa \in tsm_1.TSA} tsa.i$$
$$B = \bigcup_{tsa \in tsm_2.TSA} tsa.i$$

We now write δ_{TSM} by applying A and B.

$$\delta_{TSM}(tsm_1, tsm_2) = \frac{\max(tsm_1.n, tsm_2.n) - |A \cap B|}{\max(tsm_1.n, tsm_2.n)}$$

We henceforth assume without loss of generality that $tsm_1.n = tsm_2.n =: k$ (The set of TSAs with less elements is filled up with TSAs that have indices that are not used anywhere else). This reduces δ_{TSM} to

$$\delta_{TSM}(tsm_1, tsm_2) = \frac{k - |A \cap B|}{k}$$

We now prove the triangle inequality for three TSMs (tsm_1, tsm_2 and tsm_3) of depth 1 by assuming $tsm_1.n = tsm_2.n = tsm_3.n =: k$ and introducing a third index set $C = \bigcup_{tsa \in tsm_3.TSA} tsa.i$.

$$\begin{aligned}
\frac{k - |A \cap C|}{k} &\leq \frac{k - |A \cap B|}{k} + \frac{k - |B \cap C|}{k} \\
\Leftrightarrow k - |A \cap C| &\leq 2k - |A \cap B| - |B \cap C| \quad (*) \\
\Leftrightarrow |A \cap B| + |B \cap C| &\leq |A \cap C| + k \\
\Leftrightarrow |B \cap (A \cup C)| + |A \cap B \cap C| &\leq |A \cap C| + k \\
\Leftrightarrow |B \cap (A \cup C)| + |A \cap C| - |(A \cap C) \backslash B| &\leq |A \cap C| + k \\
\Leftrightarrow \underbrace{\underbrace{|B \cap (A \cup C)|}_{\leq |B| = k} - |(A \cap C) \backslash B|}_{\leq k - |(A \cap C) \backslash B|} &\leq k \\
\Leftarrow 0 &\leq |(A \cap C) \backslash B|
\end{aligned}$$

Induction step

We now prove that the triangle inequality holds for TSMs of depth d under the assumption that it already holds for TSMs of depth $d-1$.
Once again, we assume that $tsm_1.n = tsm_2.n = tsm_3.n =: k$ and define the following abbreviation ($1 \leq i, j \leq 3$):

$$s_{ij} = \sum_{tsa_1 \in tsm_i.TSA, tsa_2 \in tsm_j.TSA:\ tsa_1.i = tsa_2.i} \delta_{TSA}(tsa_1, tsa_2)$$

s_{13} for example is the sum of $|A \cap C|$ terms.
Using the abbreviation, the triangle inequality can be expressed as follows:

$$k - |A \cap C| + s_{13} \leq k - |A \cap B| + k - |B \cap C| + s_{12} + s_{23}$$

Case 1 ($|A \cap C| \geq |A \cap B| + |B \cap C|$)

$$\underbrace{|A \cap B| + |B \cap C|}_{\leq |A \cap C|} \leq k + |A \cap C| + s_{12} + s_{23} - s_{13}$$

$$\Leftarrow s_{13} \leq k + s_{12} + s_{23}$$

$$\Leftrightarrow \underbrace{\underbrace{\delta_{TSA}(\ldots,\ldots)}_{\leq 1} + \cdots + \underbrace{\delta_{TSA}(\ldots,\ldots)}_{\leq 1}}_{\leq k} \leq k + s_{12} + s_{23}$$

$$\Leftarrow 0 \leq s_{12} + s_{23}$$

Case 2 ($|A \cap C| \leq |A \cap B| + |B \cap C|$) (**)

$$k - |A \cap C| + s_{13} \leq 2k - |A \cap B| - |B \cap C| + s_{12} + s_{23}$$

$$\Leftarrow s_{13} \leq s_{12} + s_{23} \quad \text{because (*) is proven}$$

We now prove this inequality, starting with the induction assumption. The induction assumption states that the triangle inequality holds for any triple of sub-TSMs of tsm_1, tsm_2 or tsm_3.

$$\forall tsa_1, tsa_2, tsa_3 \in tsm_1.TSA \cup tsm_2.TSA \cup tsm_3.TSA :$$
$$\forall i,j,k \quad 1 \leq i \leq tsa_1.n, 1 \leq j \leq tsa_2.n, 1 \leq k \leq tsa_3.n :$$
$$\delta_{TSM}(tsa_1.tsm[i], tsa_3.tsm[k])$$
$$\leq \delta_{TSM}(tsa_1.tsm[i], tsa_2.tsm[j]) + \delta_{TSM}(tsa_2.tsm[j], tsa_3.tsm[k])$$

This implies that the following inequality holds as well.

$$\forall tsa_1, tsa_2, tsa_3 \in tsm_1.TSA \cup tsm_2.TSA \cup tsm_3.TSA$$
$$tsa_1.i = tsa_2.i = tsa_3.i :$$
$$\delta_{TSA}(tsa_1, tsa_3) \leq \delta_{TSA}(tsa_1, tsa_2) + \delta_{TSA}(tsa_2, tsa_3)$$

Due to (**), this directly yields

$$s_{13} \leq s_{12} + s_{23}$$

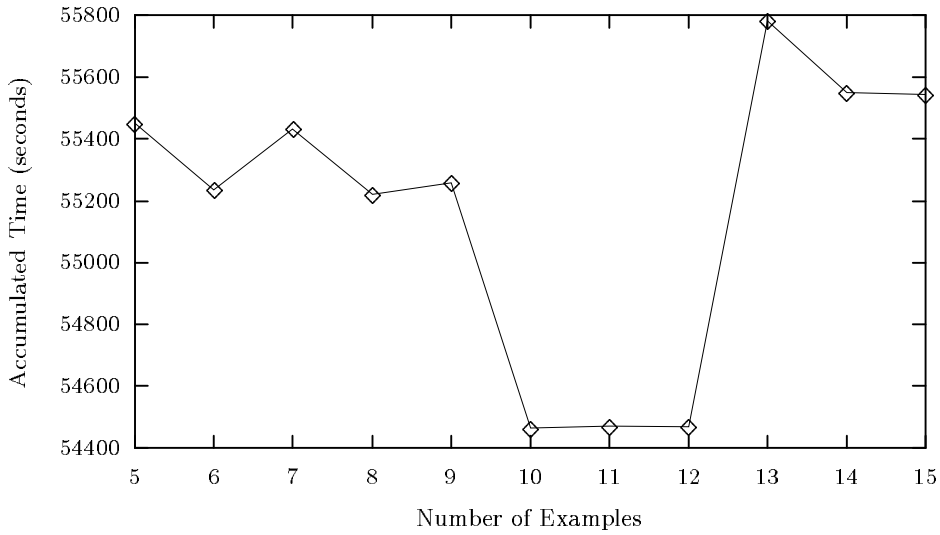
Theorem 14
$\delta_{TSM}(tsm_1, tsm_2) = 1 \Leftrightarrow \nexists t \in Term(V,F) : \quad t \propto tsm_1 \wedge t \propto tsm_2$

Proof: Assume $tsm_1.n \geq tsm_2.n$.

$$\delta_{TSM}(tsm_1, tsm_2) = 1$$
$$\Leftrightarrow \forall tsa \in tsm_1.TSA : \quad \delta'_{TSM}(tsa, tsm_2.TSA) = 1$$

This holds if either the intersection of two related sets of TSAs on a TSM path is empty (*) or, if there are two TSAs that share the same index, δ_{TSA} of those two TSAs is 1 and thus cannot be 0. This means that there are not two corresponding TSAs with equal indices and arity 0 on any TSM path (**). According to definition 1, all term structures end in a subterm of arity 0, a variable or a function symbol of arity 0.
A term t with $t \propto tsm_1$ cannot be completely mapped to tsm_2 because the mapping fails either at a certain index because of (*) or at a subterm of arity 0 because of (**).

4.6.2 Parameter Test Runs



Note: *Accumulated time* is the sum of all single proving times, assuming 360 seconds for unsolved problems.

Figure 4.7: Influence of `max_examples` on δ_{TSM}

Figure 4.7 clearly shows that the most effective number of examples is around 11. As δ_{TSM} has no weights controlling the measure of similarity, the parameter space is much smaller than δ_{FTR}'s one.

We experienced that the knowledge restriction can be easier controlled by using `max_examples` instead of `max_delta`. We mainly applied `max_delta` to filter very inappropriate examples. Please note that `max_examples` is an absolute value and should be adjusted if the knowledge base's size strongly differs from the one used in our experiments.

4.6.3 Comparison of δ_{FTR} and δ_{TSM}

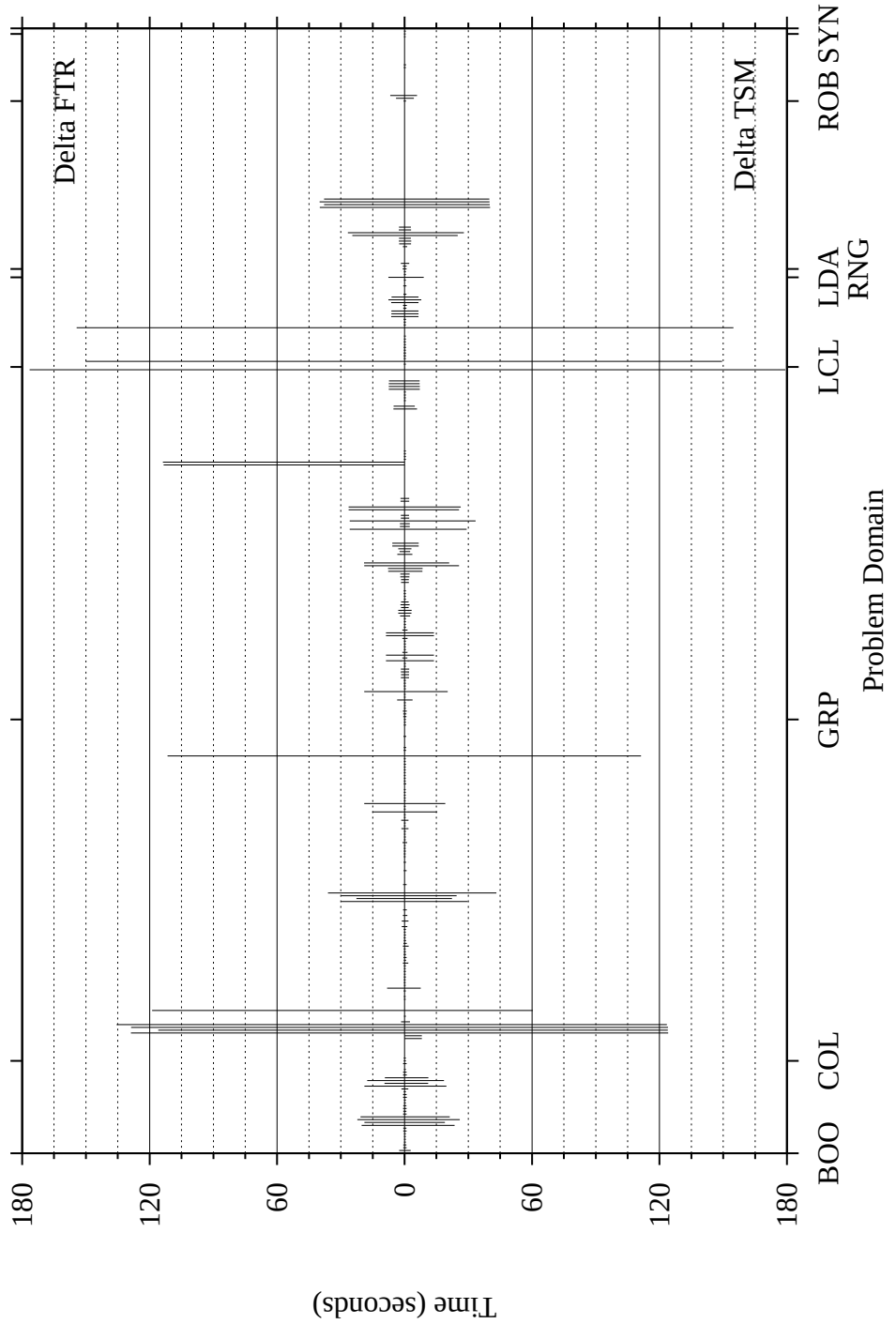
Figure 4.9 and figure 4.10 compare the distance measures δ_{TSM} and δ_{FTR}. The TSM distance between the Lusk3 specification and one problem (`SYN080-1`) is 1, which means according to theorem 14 that the TSMs spanned by the axioms do not share a single term[1]. The maximal distance to Lusk3 measured by δ_{FTR} is less than 0.8.

The figures reveal that δ_{TSM} assigns only a small set of coarse values, resulting in many equal distances. δ_{FTR} reacts more sensitively to different problem specifications. One reason for this effect is that large groups of TPTP problems share the same set of axioms. δ_{TSM} uses solely the axioms of a problem to build the TSMs; δ_{FTR}, however, takes the goal depths into consideration as well. As two TSMs, built from the goals, would be extremely small, it does not make

[1] `SYN080-1` has just one axiom: $f(x) = g(y)$.

max_examples	max_delta	Problems	Time
1	1	254	2506.61
5	0.5	256	2530.83
6	0.5	256	2316.75
7	0.5	255	2151.22
8	0.5	256	2300.68
9	0.5	256	2336.38
10	0.5	258	2263.36
11	0.5	258	2269.85
12	0.5	258	2268.37
13	0.5	253	1781.97
14	0.5	253	1550.12
15	0.5	253	1542.51
20	0.5	251	1845.18
50	0.5	253	1751.69
100	0.5	253	1844.08
1000	0.5	253	1937.93
5	0.125	256	2523.74
10	0.125	256	2271.49
20	0.125	253	1790.74
50	0.125	253	1796.10
20	0.0625	255	2216.87

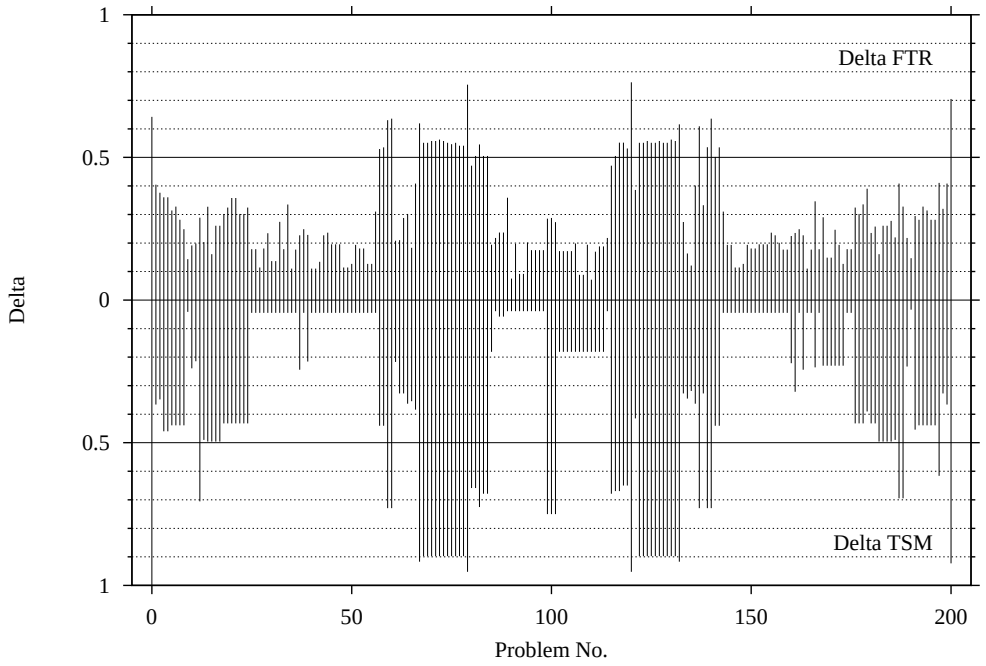
Table 4.2: Different parameter sets for δ_{TSM}



Note: δ_{FTR} ran with `max_examples` = 5 and `max_delta` = 1 and all weights set to 1, δ_{TSM} with `max_examples` = 10 and `max_delta` = 0.5. Unsolved problems have no impulse in the graph.

Figure 4.8: Comparison of δ_{FTR} and δ_{TSM}

much sense to measure their distance. A more workable method is to integrate the goal depths into δ_{TSM}. This task is accomplished in the next section by introducing δ_{MIX}.



Note: The figure shows the distance between Lusk3 and each problem in the knowledge base.

Figure 4.9: Distance measurement by δ_{FTR} and δ_{TSM}

4.7 Joined Forces - δ_{MIX}

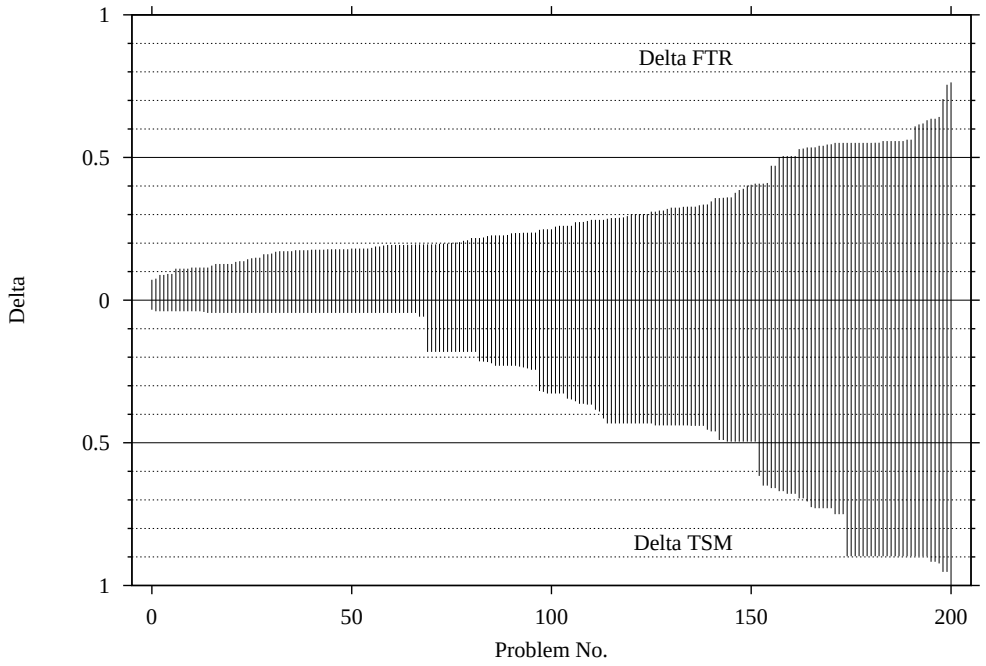
Figure 4.8 shows that, although δ_{FTR} and δ_{TSM} perform similarly, both strategies solve different examples.

In contrast to δ_{TSM}, δ_{FTR} proves `GRP181-3` and `GRP181-4`, while δ_{TSM} is solely capable of solving `COL003-15` and `COL003-16`. As a consequence, we mix both selection techniques, although they process redundant information[2]. This might fix a possible drawback of δ_{TSM}; the goal is now taken into consideration as well.

4.7.1 The Metric δ_{MIX}

We define δ_{MIX} by treating δ_{TSM} as a single feature.

[2] Attributes like number of axioms, term depths, or arity frequencies are already included in a TSM.



Note: The figure shows the distance between Lusk3 and each problem in the knowledge base, sorted by the distance.

Figure 4.10: Distance measurement by δ_{FTR} and δ_{TSM} (sorted)

Definition 39 (δ_{MIX})

$$\begin{aligned} \delta_{MIX} : (F \times TSM) \times (F \times TSM) &\mapsto [0;1] \\ \delta_{MIX}((\vec{f}_1, tsm_1), (\vec{f}_2, tsm_2)) &= \frac{s_{FTR} + w_{TSM} \cdot \delta_{TSM}(tsm_1, tsm_2)}{w_{NA} + w_{AD} + w_{DD} + w_{GD} + w_{AF} + w_{TSM}} \end{aligned} \tag{4.9}$$

$s_{FTR} = w_{NA} \cdot \delta_{NA} + w_{AD} \cdot \delta_{AD} + w_{DD} \cdot \delta_{DD} + w_{GD} \cdot \delta_{GD} + w_{AF} \cdot \delta_{AF}$ is the weighted sum of features used in 4.6.

Theorem 15
δ_{MIX} is a metric on $F \times TSM$.

Proof: The assumption is a direct implication of theorems 10, 11, 12 and 13.

4.7.2 Parameter Test Runs

Finally, the set of possible parameter combinations has reached a size, that makes it impossible to search systematically for successful settings[3]. We did not try any other feature weight combinations than the standard setting and just tested a couple of different weights for the TSM distance.

[3] A single parameter test run on our evaluation machine lasts up to eight hours.

As intended, δ_{MIX} succeeds in solving the union set of the problems proven by the previous selection strategies for a certain δ_{TSM} weighting.

max_examples	w_TSM	w_NA	w_AD	w_DD	w_GD	w_AF	Problems	Time
10	1	1	1	1	1	1	255	2322.48
10	2.5	1	1	1	1	1	260	2542.43
10	5	1	1	1	1	1	258	2263.36

Note: `max_delta` was set to 0.5 for all experiments.

Table 4.3: Different parameter sets for δ_{MIX}

4.8 Selecting Random Examples - δ_{RND}

In order to verify the usefulness of the selection strategies chosen, we define another "measure of distance" that yields random values values between 0 and 1.

Definition 40 (δ_{RND})

$$\delta_{RND} = \text{rnd}(0, 1) \tag{4.10}$$

This allows us to compare our selection methods with a strategy that selects a given number of examples purely at random.

Chapter 5

Results

The following table shows the detailed performance of each conventional strategy, regular ordered `tsm_learn` and ordered `tsm_learn` with δ_{TSM} example selection (`max_examples`=10 and `max_delta`=0.5). A dash (-) indicates that the strategy was not able to solve the problem within the 180-second time limit.

Problem	**fifo**	**add**	**occnest**	**otsm**	**otsm-δ_{TSM}**
BOO001-1	-	0.03	0.07	1	0.67
BOO002-1	-	15.34	1.69	2.24	2.03
BOO002-2	-	15.22	1.69	2.24	2.07
BOO003-2	-	1.13	0.52	0.84	0.59
BOO003-4	-	0.66	0.51	0.83	0.6
BOO004-2	-	0.53	0.21	0.82	0.58
BOO004-4	-	0.03	0.17	0.81	0.57
BOO005-2	-	0.55	1.59	0.82	0.57
BOO005-4	-	0.02	1.13	0.8	0.57
BOO006-2	-	1.15	1.06	0.84	0.6
BOO006-4	-	0.66	0.74	0.83	0.59
BOO007-2	-	161.5	63.02	30.01	20.79
BOO007-4	-	159.73	76.31	28.87	22.19
BOO008-2	-	-	33.67	33.14	24.15
BOO008-4	-	-	33.45	32.26	25.47
BOO009-2	-	1.11	2.03	0.85	0.61
BOO009-4	-	0.67	1.89	0.83	0.6
BOO010-2	-	1.11	2.02	0.85	0.61
BOO010-4	-	0.66	1.89	0.83	0.6
BOO011-2	1.84	0.02	0.02	0.81	0.57
BOO011-4	-	0.02	0.02	0.8	0.56
BOO012-2	-	0.52	0.06	0.92	0.66
BOO012-4	-	0.75	0.14	0.91	0.65
BOO013-2	-	0.64	0.5	0.84	1.89
BOO013-4	-	0.94	0.25	3.96	2.27
BOO014-2	-	26.03	-	11.52	8.18
BOO014-4	-	9.79	17.22	8.86	35.78

Problem	**fifo**	**add**	**occnest**	**otsm**	**otsm-δ_{TSM}**
BOO015-2	-	21.34	81.8	10.33	8.12
BOO015-4	-	9.6	17.33	8.86	35.52
BOO016-2	-	1.17	2.01	0.89	0.67
BOO017-2	-	1.15	2.34	0.85	0.64
BOO018-4	-	0.02	0.02	0.8	0.56
BOO019-1	-	-	-	-	-
COL001-1	-	0.39	2.84	0.94	0.89
COL001-2	0.23	0.07	0.13	0.85	0.62
COL002-1	0.17	0.04	0.11	0.82	0.58
COL002-4	0.87	-	-	-	-
COL002-5	-	-	-	-	-
COL003-1	-	-	-	-	-
COL003-12	-	10.59	-	-	-
COL003-13	-	10.61	-	-	-
COL003-14	-	10.59	-	-	2.84
COL003-15	-	10.61	-	-	2.08
COL003-16	-	10.59	-	-	2.09
COL003-17	-	-	-	-	113.28
COL003-18	-	-	-	-	112.88
COL003-19	-	-	-	-	113.04
COL003-20	-	-	-	-	112.14
COL004-1	28.43	1.25	11.4	2.91	3.04
COL004-2	-	-	-	-	-
COL004-3	0.14	0.01	0.02	0.78	0.6
COL005-1	-	-	-	-	-
COL006-1	-	41.47	-	4.53	47.59
COL006-5	-	-	-	-	-
COL006-6	-	-	-	-	-
COL006-7	-	-	-	-	-
COL007-1	0	0.01	0.02	0.82	0.54
COL008-1	0.02	0.03	0.02	0.82	0.56
COL009-1	-	-	0.32	-	-
COL010-1	0.02	0.01	0.07	0.83	0.71
COL011-1	7.07	9.38	-	8.66	50.42
COL012-1	0.01	0.01	0	0.8	0.53
COL013-1	0.01	0.02	0.02	0.82	0.55
COL014-1	0.01	0.01	0.02	0.82	0.54
COL015-1	0.01	0.04	0.04	0.85	0.59
COL016-1	0.02	0.02	0.02	0.81	0.55
COL017-1	0.02	0.04	0.03	0.85	0.66
COL018-1	0.01	0.02	0.02	0.82	0.55
COL019-1	-	0.05	2.3	0.83	0.57
COL020-1	-	0.71	-	0.98	0.83
COL021-1	0.05	0.1	0.09	0.9	0.81

Problem	fifo	add	occnest	otsm	otsm-δ_{TSM}
COL022-1	0.02	0.27	0.31	1.19	0.99
COL023-1	1.48	0.13	2.84	0.87	0.74
COL024-1	0.02	0.03	0.03	0.83	0.9
COL025-1	0.02	0.02	0.03	0.83	0.56
COL026-1	0.93	1.16	4.24	1.19	2.01
COL027-1	1.94	0.24	4.84	0.9	0.83
COL028-1	1.49	0.12	2.84	0.87	0.74
COL029-1	0.02	0.02	0.02	0.8	0.54
COL030-1	0.25	0.05	0.16	0.86	0.59
COL031-1	0.02	0.02	0.01	0.82	0.55
COL032-1	0.2	0.07	0.07	0.89	0.62
COL033-1	1.97	0.57	0.55	1.93	1.3
COL034-1	-	-	-	-	-
COL035-1	8.75	1.12	1.46	2.13	0.78
COL036-1	-	-	1.43	-	-
COL037-1	-	0.9	-	1.19	0.86
COL038-1	-	-	-	-	-
COL039-1	0.04	0.28	1.75	1.31	1.11
COL041-1	-	-	-	-	-
COL042-1	-	-	-	-	-
COL042-6	40.77	42	-	88.27	7.43
COL042-7	-	42.14	-	143.46	4.78
COL042-8	41.5	41.99	-	76.1	5.69
COL042-9	-	41.91	-	-	11.16
COL043-1	-	-	-	-	-
COL043-3	-	-	-	-	-
COL044-1	1.99	0.35	5.05	0.99	1.01
COL044-6	-	-	-	-	-
COL044-7	-	-	-	-	-
COL044-8	-	-	-	-	-
COL044-9	-	-	-	-	-
COL045-1	0.48	0.14	0.72	0.95	1.09
COL046-1	-	-	-	-	-
COL047-1	-	-	-	-	-
COL048-1	0.03	0.02	0.05	0.84	0.65
COL049-1	-	-	-	-	-
COL050-1	0.02	0.02	0.02	0.83	0.63
COL051-1	0.01	0.01	0.02	0.83	0.6
COL052-1	0.03	0.03	0.03	0.79	0.59
COL053-1	0.02	0.01	0.02	0.81	0.58
COL056-1	0.02	0.01	0.01	0.79	0.61
COL057-1	-	0.89	26.46	2.42	1.31
COL058-1	0.01	0.03	0.02	0.83	0.61
COL058-2	0.06	0.03	0.03	0.84	0.64

Problem	**fifo**	**add**	**occnest**	**otsm**	**otsm-δ_{TSM}**
COL058-3	0.06	0.03	0.05	0.83	0.61
COL059-1	1.57	0.02	0.05	0.79	0.62
COL060-1	1.27	1.27	1.28	2.07	2.01
COL060-2	0.02	0.02	0.02	0.82	0.62
COL060-3	0.02	0.03	0.02	0.81	0.6
COL061-1	1.28	1.27	1.28	2.08	1.98
COL061-2	0.01	0.01	0.02	0.82	0.59
COL061-3	0.01	0.01	0.02	0.82	0.61
COL062-1	14.93	14.87	14.9	15.73	17.47
COL062-2	0.01	0.02	0.01	0.82	0.6
COL062-3	0.02	0.01	0.01	0.82	0.58
COL063-1	18.67	18.69	18.69	19.51	21.54
COL063-2	0.02	0.02	0.02	0.82	0.61
COL063-3	0.01	0.03	0.02	0.82	0.58
COL063-4	0.01	0.01	0.02	0.82	0.6
COL063-5	0.02	0.03	0.03	0.81	0.61
COL063-6	0.02	0.02	0.02	0.82	0.59
COL064-1	-	-	-	-	-
COL064-10	0.02	0.01	0.02	0.98	0.69
COL064-11	0.01	0.02	0.02	0.82	0.6
COL064-2	0.02	0.02	0.02	0.81	0.57
COL064-3	0.02	0.02	0.01	0.82	0.58
COL064-4	0.02	0.01	0.02	0.82	0.63
COL064-5	0.02	0.02	0.01	0.82	0.59
COL064-6	0.02	0.02	0.02	0.82	0.64
COL064-7	0.01	0.02	0.02	0.83	0.6
COL064-8	0.02	0.01	0.02	0.82	0.6
COL064-9	0.02	0.02	0.02	0.82	0.61
COL065-1	110.82	111.26	110.73	112.9	122.44
COL066-1	-	-	-	-	-
COL066-2	0.08	0.46	0.06	1.65	0.76
COL066-3	0.08	0.46	0.05	1.57	0.78
COL067-1	-	-	-	-	-
COL068-1	-	-	-	-	-
COL069-1	-	-	-	-	-
COL070-1	26.21	0.15	1.23	1.09	0.66
COL071-1	-	-	-	-	-
COL072-1	-	-	-	-	-
COL073-1	-	-	-	-	-
COL075-2	0.15	0.02	0.05	0.96	0.63
GRP001-2	0.1	0.02	0.02	0.85	0.57
GRP001-4	0.12	0.03	0.01	0.88	0.56
GRP002-2	28.29	0.49	1.74	1.31	1.11
GRP002-3	-	0.74	-	1.64	0.89

Problem	fifo	add	occnest	otsm	otsm-δ_{TSM}
GRP002-4	-	0.72	-	1.58	0.83
GRP010-4	0.08	0.02	0.02	0.86	0.57
GRP011-4	0.52	0.02	0.02	0.89	0.58
GRP012-4	0.08	0.02	0.02	0.89	0.57
GRP014-1	-	3.62	10.05	3.1	2.29
GRP022-2	0.01	0.02	0.02	0.84	0.58
GRP023-2	0.01	0.02	0.02	0.86	0.56
GRP114-1	-	48.91	1.12	21.31	22.04
GRP115-1	-	0.03	0.37	0.88	0.61
GRP116-1	-	0.02	0.27	0.9	0.6
GRP117-1	-	0.03	0.25	0.87	0.6
GRP118-1	-	0.03	0.22	0.86	0.6
GRP119-1	-	2.29	19.39	2.43	1.87
GRP120-1	-	2.31	13.64	2.39	1.89
GRP121-1	-	2.3	13.49	2.44	1.9
GRP122-1	-	2.66	2.13	2.54	1.92
GRP136-1	0.03	0.02	0.02	0.93	0.58
GRP137-1	0.04	0.02	0.02	0.91	0.57
GRP138-1	-	7.79	0.13	23.67	24.99
GRP139-1	0.89	0.11	0.03	2	1.35
GRP140-1	-	7.82	0.12	23.97	22.15
GRP141-1	50.97	0.11	0.18	1.96	1.45
GRP142-1	0.28	0.02	0.02	0.92	0.56
GRP143-1	0.05	0.03	0.03	0.97	0.58
GRP144-1	0.03	0.02	0.02	0.93	0.56
GRP145-1	0.18	0.03	0.03	0.97	0.58
GRP146-1	0.91	0.41	0.03	2.06	1.36
GRP147-1	-	10.86	0.34	23.43	24.93
GRP148-1	-	10.86	0.35	23.63	22.12
GRP149-1	32.01	0.41	0.18	1.94	1.46
GRP150-1	0.08	0.04	0.02	0.85	0.58
GRP151-1	0.39	0.02	0.03	0.88	0.57
GRP152-1	0.14	0.02	0.03	0.92	0.57
GRP153-1	0.03	0.03	0.02	0.91	0.57
GRP154-1	0.04	1.66	0.02	3.73	2.15
GRP155-1	0.03	2.25	0.03	3.84	2.24
GRP156-1	67.39	2.25	0.03	3.84	2.23
GRP157-1	0.04	1.06	0.02	2.65	1.41
GRP158-1	0.03	1.44	0.02	2.7	1.41
GRP159-1	0.88	1.06	0.03	2.67	1.39
GRP160-1	0.02	0.02	0.02	0.92	0.58
GRP161-1	0.02	0.01	0.02	0.92	0.57
GRP162-1	-	0.07	0.22	0.97	0.63
GRP163-1	-	0.07	0.22	0.98	0.62

Problem	fifo	add	occnest	otsm	otsm-δ_{TSM}
GRP164-1	-	-	-	-	-
GRP164-2	-	-	-	-	-
GRP165-1	0.63	1.09	0.03	2.56	1.86
GRP165-2	0.76	1.13	0.03	2.6	1.51
GRP166-1	-	1.31	18.11	2.92	2.1
GRP166-2	-	1.37	2.71	2.98	1.66
GRP166-3	0.64	11.26	0.02	5.89	4.12
GRP166-4	0.77	11.41	0.02	6.06	3.25
GRP167-1	-	49.37	2.52	22.82	6.87
GRP167-2	-	49.16	1.11	22.68	12.63
GRP167-3	-	-	-	-	-
GRP167-4	-	-	129.46	-	-
GRP167-5	-	5.39	2.5	5.03	4.45
GRP168-1	0.03	1.66	0.02	3.33	2.15
GRP168-2	0.04	2.25	0.03	3.44	2.26
GRP169-1	-	9.45	-	7.26	6.51
GRP169-2	-	9.48	-	7.23	6.55
GRP170-1	-	-	0.89	-	-
GRP170-2	-	-	0.93	-	-
GRP170-3	-	-	0.92	-	-
GRP170-4	-	-	0.89	-	-
GRP171-1	-	6.8	1.07	77.58	96.94
GRP171-2	-	1.58	1.07	3.13	1.78
GRP172-1	-	1.57	1.06	3.12	2.31
GRP172-2	-	6.81	1.07	77.02	128.39
GRP173-1	-	1.37	-	2.82	2.42
GRP174-1	-	1.35	-	2.83	2.22
GRP175-1	0.63	-	0.04	-	-
GRP175-2	0.76	57.98	0.06	43.71	35.76
GRP175-3	68.45	58.28	0.32	44.11	45.91
GRP175-4	0.88	-	0.32	-	-
GRP176-1	0.04	1.35	0.04	2.91	1.36
GRP176-2	0.03	1.35	0.04	2.92	1.75
GRP177-1	-	-	-	-	-
GRP177-2	-	-	-	-	-
GRP178-1	-	-	96.73	-	-
GRP178-2	-	-	96.77	-	-
GRP179-1	-	-	-	-	-
GRP179-2	-	-	-	-	-
GRP179-3	-	-	-	-	-
GRP180-1	-	-	-	-	-
GRP180-2	-	-	-	-	-
GRP181-1	-	-	-	-	-
GRP181-2	-	-	-	-	-

Problem	fifo	add	occnest	otsm	otsm-δ_{TSM}
GRP181-3	-	-	-	-	-
GRP181-4	-	-	-	-	-
GRP182-1	0.28	0.02	0.03	0.83	0.57
GRP182-2	0.28	0.02	0.03	0.82	0.59
GRP182-3	0.39	0.02	0.02	0.83	0.58
GRP182-4	0.39	0.03	0.02	0.83	0.59
GRP183-1	-	-	-	-	-
GRP183-2	-	-	-	-	-
GRP183-3	-	-	-	-	-
GRP183-4	-	-	-	-	-
GRP184-1	-	-	-	-	-
GRP184-2	-	-	-	-	-
GRP184-3	-	-	-	-	-
GRP184-4	-	-	-	-	-
GRP185-1	-	-	59.74	-	-
GRP185-2	-	-	60.1	-	-
GRP185-3	-	-	-	-	-
GRP185-4	-	-	-	-	-
GRP186-1	-	-	-	-	-
GRP186-2	-	-	-	-	-
GRP186-3	-	8.61	0.25	6.73	6
GRP186-4	132.81	8.6	0.04	6.7	5.14
GRP187-1	-	-	-	-	-
GRP188-1	0.15	0.03	0.02	0.82	0.58
GRP188-2	0.14	0.02	0.03	0.83	0.6
GRP189-1	0.04	0.03	0.02	0.81	0.57
GRP189-2	0.04	0.02	0.02	0.82	0.6
GRP190-1	-	12.83	12.3	17.14	12.96
GRP190-2	-	12.82	12.53	17.18	13.01
GRP191-1	-	12.82	12.53	17.33	12.93
GRP191-2	-	12.85	12.29	17.24	12.59
GRP192-1	98.75	-	0.32	-	1.13
GRP193-1	-	-	96.63	-	-
GRP193-2	-	-	96.92	-	-
LCL109-2	-	-	-	-	36.94
LCL109-6	-	-	-	-	-
LCL110-2	-	0.06	0.07	0.88	0.6
LCL111-2	-	-	148.45	-	17.77
LCL112-2	-	0.06	0.09	0.88	0.61
LCL113-2	-	0.16	0.11	0.92	0.67
LCL114-2	-	0.13	0.08	0.9	0.65
LCL115-2	-	0.07	0.07	0.91	0.63
LCL116-2	-	0.18	0.17	0.97	0.84
LCL132-1	0.38	0.03	0.03	0.81	0.57

Problem	**fifo**	**add**	**occnest**	**otsm**	**otsm-δ_{TSM}**
LCL133-1	102.06	0.07	-	0.86	0.62
LCL134-1	-	0.03	0.03	0.83	0.58
LCL135-1	-	0.03	0.03	0.81	0.57
LCL136-1	-	-	-	-	-
LCL137-1	-	-	-	-	-
LCL138-1	-	-	-	-	19
LCL139-1	-	0.05	0.14	0.89	0.61
LCL140-1	-	0.06	0.08	0.89	0.61
LCL141-1	-	0.14	0.1	0.91	0.64
LCL153-1	-	5.76	-	21.63	6.6
LCL154-1	-	5.76	-	21.63	6.62
LCL155-1	-	5.77	-	21.68	6.6
LCL156-1	-	0.24	-	1.12	0.81
LCL157-1	-	0.37	-	1.18	0.97
LCL158-1	-	5.77	-	21.62	6.61
LCL159-1	-	5.84	-	58.99	7.46
LCL160-1	-	5.75	-	21.65	6.63
LCL161-1	10.33	0.29	-	1.29	0.87
LCL162-1	-	-	-	-	-
LCL163-1	-	-	-	-	-
LCL164-1	10.65	0.29	-	1.32	0.9
LCL165-1	-	-	-	-	-
LDA001-1	-	0.03	0.02	0.79	0.57
LDA002-1	-	5.72	138.34	25.58	9.18
LDA007-3	-	0.03	0.03	0.81	0.57
RNG007-4	102.7	0.02	0.4	0.82	0.58
RNG008-3	-	0.37	21.34	1.56	1.15
RNG008-4	-	0.38	21.32	1.58	1.14
RNG008-7	-	1.65	40.58	2.48	1.89
RNG009-5	-	-	-	-	-
RNG009-7	-	133.14	-	-	158.55
RNG010-5	-	-	-	-	-
RNG010-6	-	-	-	-	-
RNG010-7	-	-	-	-	-
RNG011-5	0.05	0.92	0.02	1.42	0.84
RNG012-6	-	17.79	131.82	18.17	2.36
RNG013-6	-	17.79	1.28	18.14	2.34
RNG014-6	-	17.57	0.98	13.05	1.54
RNG015-6	-	32	0.34	23.74	2.02
RNG016-6	-	32.27	0.5	34.01	4.22
RNG017-6	-	17.86	0.61	18.09	2.38
RNG018-6	-	17.62	0.4	13.01	1.54
RNG019-6	-	-	-	-	-
RNG019-7	-	-	-	-	-

Problem	fifo	add	occnest	otsm	otsm-δ_{TSM}
RNG020-6	-	-	-	-	-
RNG020-7	-	-	-	-	-
RNG021-6	-	-	-	-	-
RNG021-7	-	-	-	-	-
RNG023-6	0.42	83.95	-	52.73	15.68
RNG023-7	0.52	81.53	-	49.35	15.52
RNG024-6	0.33	83.84	-	52.64	15.72
RNG024-7	0.4	81.53	-	49.44	15.49
RNG025-4	-	-	-	-	-
RNG025-5	-	-	-	-	-
RNG025-6	-	-	-	-	-
RNG025-7	-	-	-	-	-
RNG025-8	-	-	-	-	-
RNG025-9	-	-	-	-	-
RNG026-6	-	-	-	-	-
RNG026-7	-	-	-	-	-
RNG027-5	-	-	-	-	-
RNG027-6	-	-	-	-	-
RNG027-7	-	-	-	-	-
RNG027-8	-	-	-	-	-
RNG027-9	-	-	-	-	-
RNG028-5	-	-	-	-	-
RNG028-6	-	-	-	-	-
RNG028-7	-	-	-	-	-
RNG028-8	-	-	-	-	-
RNG028-9	-	-	-	-	-
RNG029-5	-	-	-	-	-
RNG029-6	-	-	-	-	-
RNG029-7	-	-	-	-	-
RNG030-6	-	-	-	-	-
RNG030-7	-	-	-	-	-
RNG031-6	-	-	-	-	-
RNG031-7	-	-	-	-	-
RNG032-6	-	-	-	-	-
RNG032-7	-	-	-	-	-
RNG033-6	-	-	-	-	-
RNG033-7	-	-	-	-	-
RNG033-8	-	-	-	-	-
RNG033-9	-	-	-	-	-
RNG035-7	-	-	-	-	-
RNG036-7	-	-	-	-	-
ROB001-1	-	-	-	-	-
ROB002-1	5.46	0.11	0.03	1.1	0.72
ROB003-1	-	5.5	11.67	5.03	3.89

Problem	**fifo**	**add**	**occnest**	**otsm**	**otsm-δ_{TSM}**
ROB004-1	-	5.79	11.61	5.73	4.34
ROB005-1	-	-	-	-	-
ROB006-1	-	-	-	-	-
ROB006-2	-	-	-	-	-
ROB007-1	-	-	-	-	-
ROB007-2.1	-	-	-	-	-
ROB007-2.2	-	-	-	-	-
ROB007-2.3	-	-	-	-	-
ROB008-1	-	-	-	-	-
ROB009-1	-	-	-	-	-
ROB010-1	0.38	0.1	0.1	0.91	0.67
ROB013-1	-	0.12	0.11	0.92	0.7
ROB020-1	-	-	-	-	-
ROB020-2.1	-	-	-	-	-
ROB020-2.2	-	-	-	-	-
ROB020-2.3	-	-	-	-	-
ROB022-1	-	-	-	-	44.84
ROB023-1	-	-	22.96	-	125.61
ROB024-1	-	-	-	-	-
ROB026-1	-	-	-	-	-
ROB027-1	-	-	-	-	-
SYN080-1	0.01	0.02	0.01	0.78	0.52
SYN083-1	0.02	0.01	0.02	0.81	0.55
SYN305-1	0.03	0.01	0.02	0.78	0.6

Table 5.1: Performance of evaluation strategies (in detail)

Strategy	No ex. sel.	δ_{FTR}	δ_{TSM}	δ_{MIX}	δ_{RND}
Problems solved	248	259	263	262	239
Time taken (seconds)	1936.69	1891.41	2168.11	2099.37	1764.12

Note: 403 problems of the TPTP library had to be proven within a time limit of 180 seconds for each problem. Ordered TSM learning was used with the following parameters. δ_{FTR}: `max_examples` $= 5$, `max_delta` $= 1$ and all weights set to 1; δ_{TSM}: `max_examples` $= 10$, `max_delta` $= 0.5$; δ_{MIX}: `max_examples` $= 10$, `max_delta` $= 0.5$ and all weights set to 1.

Table 5.2: Performance of example selection strategies

The overhead of the example selection mechanism has been reduced by applying a binary tree for sorting the examples. It is less than half a second in the average.

During the extensive test runs, we were able to prove three problems that have not been solved by an ATP system before (according to the TPTP library documentation). Two of the problems were solved by other state-of-

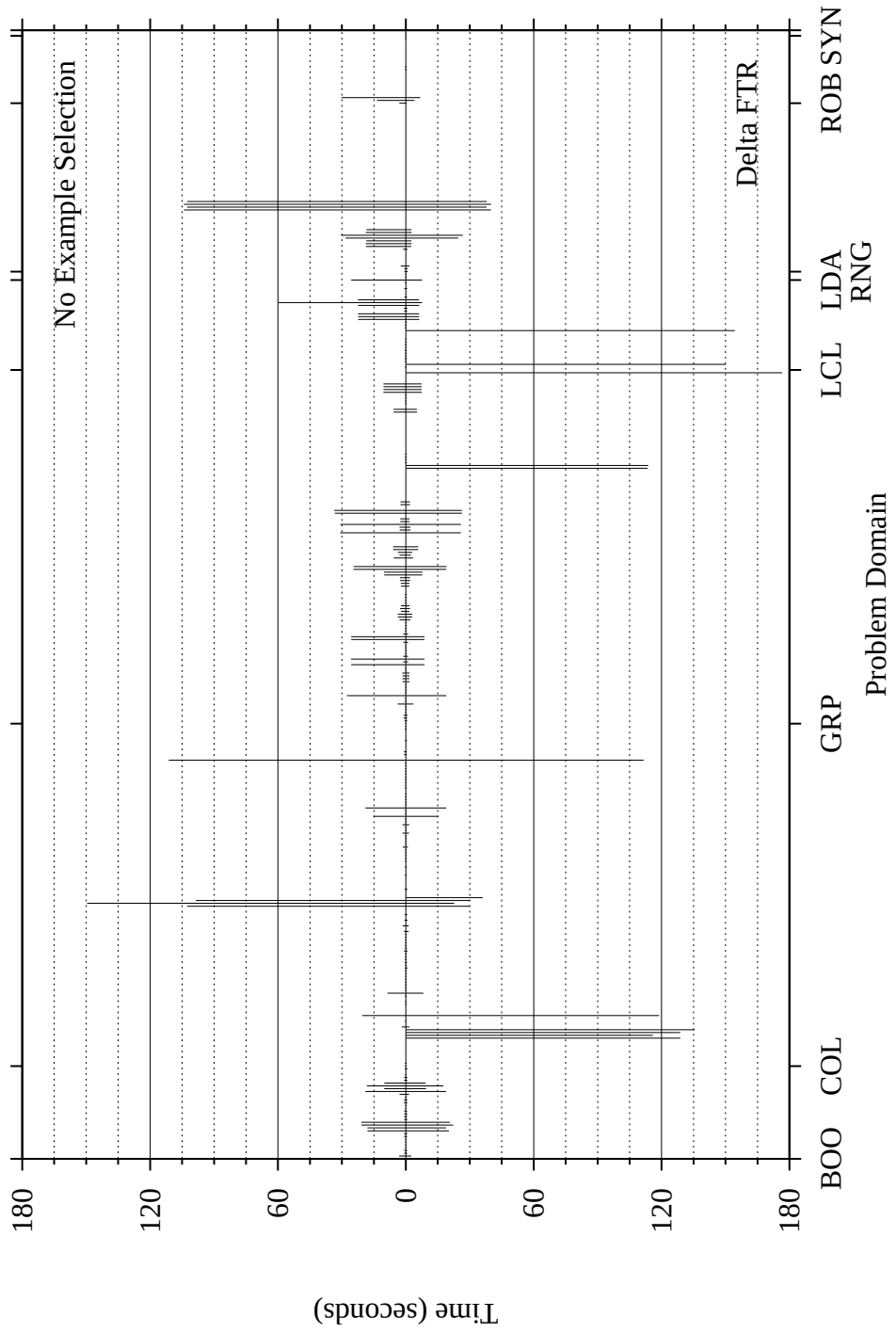
the-art provers as well, but there is no recorded proof for `COL042-9` except DISCOUNT's.

The new improved version of DISCOUNT, called *DISCOUNT-TSM 2.1*, took part in the unit equality division of the CADE-15 System Competition [Sut98] and achieved a satisfying third rank (see table 5.3). We used ordered `tsm_learn` with δ_{TSM} example selection and the standard parameters (`max_examples`=10 and `max_delta`=0.5) during the competition.

Prover	**Problems solved**	**Average time (seconds)**
Waldmeister 798	30	0.62
Otter 3.0.5	25	10.68
DISCOUNT-TSM 2.1	25	28.32
Bliksem 1.00	25	47.43
Gandalf c-1.1	25	60.96
SPASS 1.0.0a	23	33.73
E 0.1	19	31.65
SCOTT v3.1.0	19	78.57

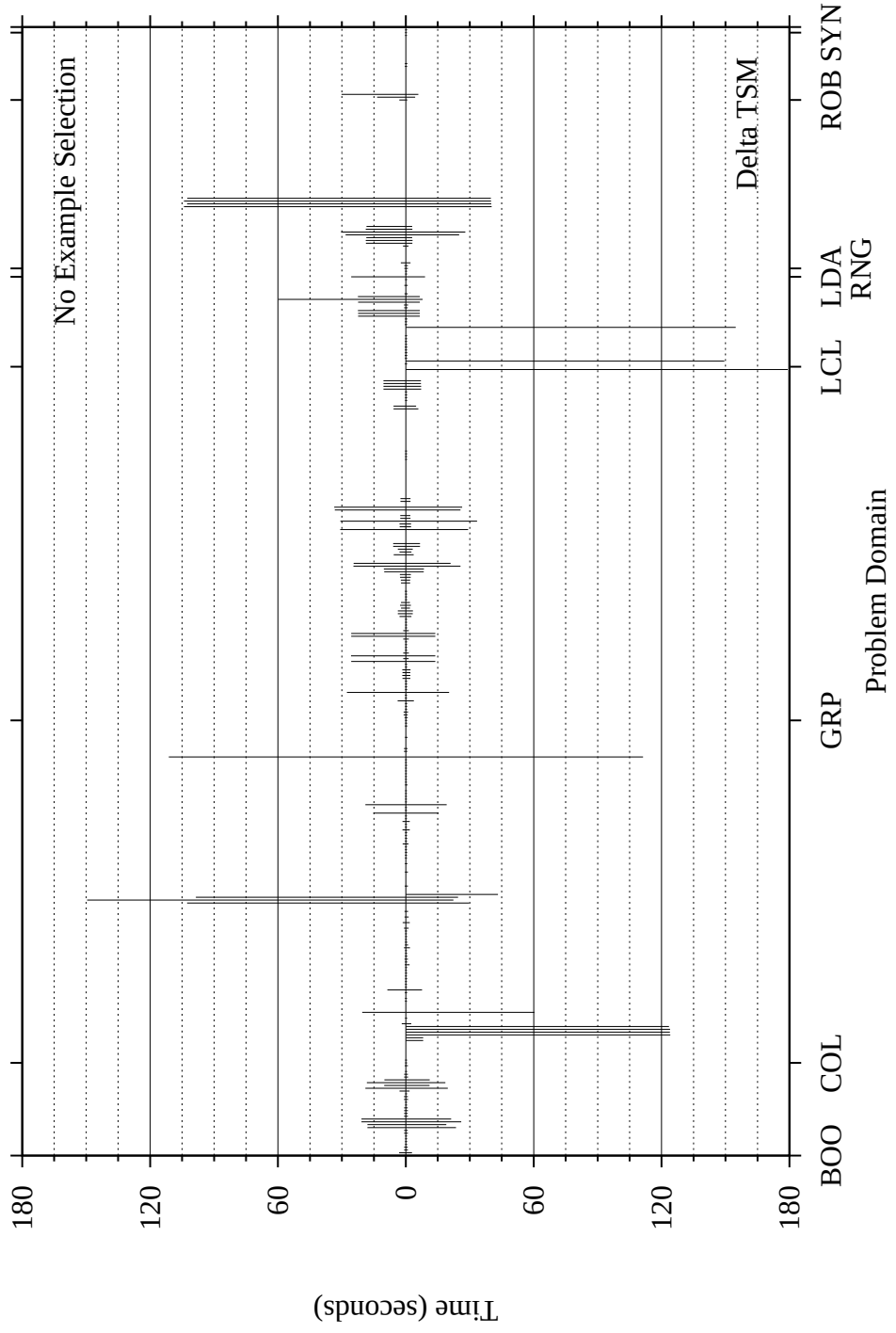
Note: 30 Problems of the TPTP library had to be solved within a time limit of 300 seconds per problem.

Table 5.3: CASC-15 Results



Note: δ_{FTR} ran with `max_examples` = 5 and `max_delta` = 1 and all weights set to 1.

Figure 5.1: δ_{FTR}'s performance gain by selecting examples



Note: δ_{TSM} ran with `max_examples` $= 10$ and `max_delta` $= 0.5$.

Figure 5.2: δ_{TSM}'s performance gain by selecting examples

Chapter 6

Conclusion

We presented metrics that measure the distance between problem specifications, thus enabling the prover to select training examples that are similar to the problem to be proven. For this purpose, we used two different approaches that both revealed advantages and disadvantages.
This resulted in a third method that combines the previous two and outperforms both of them.

The results of all three selection techniques confirmed the need for knowledge restriction of TSM-based learning. Each of them proved more problems than the regular learning strategy using all examples available.

We showed that the distance measures are appropriate to decide on similarity of problems by introducing another strategy that selects a fixed number of examples purely at random. This method performs poorly, even worse than the conventional learning strategy without example selection. Solely restricting the used knowledge does not increase performance. The crucial step is to decide which knowledge can be omitted and which is important.

When we used ordered TSM learning in the final evaluation, the TSM-based measure of similarity surprisingly yielded better results than both other strategies. One reason for this might be that we tuned all parameters for the regular TSM learning. The selection mechanism reacts sensitively to minor parameter changes. We firmly believe that parameter tuning for ordered TSM learning will result in performance gains for all three selection techniques. Unfortunately, parameter test runs take much time and the parameter space is extremely large.

The results of this work emphasize the power of learning heuristics, whose development is still at the beginning, having much room for improvements.

Finally, DISCOUNT's development might come to an end. DISCOUNT was intended as an experimental prover for the demonstration of the TEAMWORK method and has since then been modified by many different people for various reasons.

It seems to be a challenging task to couple DISCOUNT's strong learning heuristics with a faster inference engine by creating a new state-of-the-art prover.

Appendix A

An Example Proof

This appendix contains DISCOUNT's proof of the Lusk3 problem[1]. The following problem specification is handed to DISCOUNT:

```
MODE        PROOF

NAME        Lusk3

ORDERING    XKBO
            f:5 > j:4 > g:3 > 0:1 > b:1 > a:1

EQUATIONS   j (0,x)       = x                  # left identity
            j (x,0)       = x                  # right identity

            j (g (x),x)   = 0                  # left inverse
            j (x,g (x))   = 0                  # right inverse

            j (j (x,y),z) = j (x,j (y,z))      # associativity
            j (x,y)       = j(y,x)             # commutativity

            f (f (x,y),z) = f (x,f (y,z))      # associativity

            f (x,j (y,z)) = j (f (x,y),f (x,z)) # distributivity
            f (j (x,y),z) = j (f (x,z),f (y,z))

            f (x,x)       = x                  # x*x = x

CONCLUSION  f (a,b) = f (b,a)                  # theorem
```

DISCOUNT then generates a pruned PCL listing of the proof.

```
0 : tes-eqn : f(x,x) = x : initial:(0,0,T,0)
1 : tes-eqn : j(0(),x) = x : initial:(0,0,T,0)
2 : tes-eqn : j(x,0()) = x : initial:(0,0,T,0)
3 : tes-eqn : j(g(x),x) = 0() : initial:(0,0,T,2)
```

[1] A ring with $x^2 = x$ is Abelian.

```
   4 : tes-eqn : j(x,g(x)) = 0() : initial:(0,0,T,0)
   5 : tes-eqn : j(x,y) = j(y,x) : initial:(0,0,T,186)
   6 : tes-eqn : j(j(x,y),z) = j(x,j(y,z)) : initial:(0,0,T,0)
   7 : tes-eqn : f(f(x,y),z) = f(x,f(y,z)) : initial:(0,0,T,41)
   8 : tes-eqn : f(x,j(y,z)) = j(f(x,y),f(x,z)) : initial:(0,0,T,0)
   9 : tes-eqn : f(j(x,y),z) = j(f(x,z),f(y,z)) : initial:(0,0,T,0)
  10 : tes-goal : f(a(),b()) = f(b(),a()) : hypothesis:(0,0,T,0)
  11 : tes-rule : f(x,x) -> x : orient(0,u):(0,0,T,11)
  12 : tes-rule : j(0(),x) -> x : orient(1,u):(0,0,T,76)
  13 : tes-rule : j(x,0()) -> x : orient(2,u):(0,0,T,36)
  14 : tes-rule : j(g(x),x) -> 0() : orient(3,u):(0,0,F,2)
  15 : tes-eqn : 0() = g(0()) : cp(14,L,13,L):(1,1,F,26)
  16 : tes-rule : g(0()) -> 0() : orient(15,x):(1,1,F,26)
  19 : tes-rule : j(x,g(x)) -> 0() : orient(4,u):(0,0,T,67)
  40 : tes-rule : j(j(x,y),z) -> j(x,j(y,z)) : orient(6,u):(0,0,T,122)
  48 : tes-eqn : j(x,j(g(x),y)) = j(0(),y) : cp(40,L.1,19,L):(0,0,T,0)
  49 : tes-eqn : j(x,j(g(x),y)) = y : tes-red(48,R,12,L):(0,0,T,0)
  54 : tes-eqn : j(x,j(y,z)) = j(y,j(z,x)) : cp(5,L,40,L):(0,0,T,218)
  56 : tes-eqn : j(x,j(y,z)) = j(j(y,x),z) : cp(40,L.1,5,L):(1,1,F,214)
  57 : tes-eqn : j(x,j(y,z)) = j(y,j(x,z)) : tes-red(56,R,40,L):(2,1,F,214)
  60 : tes-rule : j(x,j(g(x),y)) -> y : orient(49,u):(0,0,T,81)
  66 : tes-eqn : g(g(x)) = j(x,0()) : cp(60,L.2,19,L):(0,0,T,0)
  67 : tes-eqn : g(g(x)) = x : tes-red(66,R,13,L):(0,0,T,0)
  68 : tes-eqn : x = j(y,j(z,j(g(j(y,z)),x))) : cp(60,L,40,L):(1,1,F,128)
  82 : tes-eqn : x = j(y,j(x,g(y))) : cp(60,L.2,5,L):(0,0,T,0)
  84 : tes-rule : g(g(x)) -> x : orient(67,u):(0,0,T,24)
  88 : tes-eqn : x = j(g(y),j(y,x)) : cp(60,L.2.1,84,L):(0,0,T,0)
  91 : tes-rule : j(x,j(y,g(x))) -> y : orient(82,x):(0,0,T,103)
  99 : tes-eqn : x = j(y,j(z,j(x,g(j(y,z))))) : cp(91,L,40,L):(1,1,F,99)
 100 : tes-eqn : j(x,y) = j(z,j(x,j(y,g(z)))) : cp(91,L.2,40,L):(1,1,F,131)
 101 : tes-eqn : j(x,j(j(y,g(x)),z)) = j(y,z) : cp(40,L.1,91,L):(1,1,F,158)
 102 : tes-eqn : j(x,j(y,j(g(x),z))) = j(y,z) : tes-red(101,L.2,40,L):(2,1,F,158)
 108 : tes-eqn : x = j(g(y),j(x,y)) : cp(91,L.2.2,84,L):(0,0,T,0)
 120 : tes-rule : j(g(x),j(x,y)) -> y : orient(88,x):(0,0,T,23)
 152 : tes-rule : j(g(x),j(y,x)) -> y : orient(108,x):(0,0,T,25)
 167 : tes-eqn : x = j(g(j(g(x),y)),y) : cp(152,L.2,60,L):(1,1,F,23)
 168 : tes-eqn : x = j(y,g(j(g(x),y))) : tes-red(167,R,5,L):(2,1,F,23)
 170 : tes-eqn : x = j(g(j(y,g(x))),y) : cp(152,L.2,91,L):(1,1,F,21)
 171 : tes-eqn : x = j(y,g(j(y,g(x)))) : tes-red(170,R,5,L):(2,1,F,21)
 172 : tes-eqn : g(x) = j(g(j(x,y)),y) : cp(152,L.2,120,L):(0,0,T,0)
 173 : tes-eqn : g(x) = j(y,g(j(x,y))) : tes-red(172,R,5,L):(0,0,T,0)
 176 : tes-eqn : g(x) = j(g(j(y,x)),y) : cp(152,L.2,152,L):(1,1,F,75)
 177 : tes-eqn : g(x) = j(y,g(j(y,x))) : tes-red(176,R,5,L):(2,1,F,75)
 196 : tes-rule : j(x,g(j(g(y),x))) -> y : orient(168,x):(2,1,F,23)
 247 : tes-rule : j(x,g(j(x,g(y)))) -> y : orient(171,x):(2,1,F,21)
 289 : tes-rule : j(x,g(j(y,x))) -> g(y) : orient(173,x):(0,0,T,77)
 309 : tes-eqn : g(j(x,g(y))) = j(y,g(x)) : cp(60,L.2,289,L):(1,1,F,41)
 348 : tes-rule : j(x,g(j(x,y))) -> g(y) : orient(177,x):(2,1,F,75)
 367 : tes-eqn : g(j(g(x),y)) = j(x,g(y)) : cp(348,L.2.1,60,L):(3,2,F,62)
 401 : tes-rule : f(f(x,y),z) -> f(x,f(y,z)) : orient(7,u):(0,0,F,41)
 402 : tes-eqn : f(x,f(y,f(x,y))) = f(x,y) : cp(401,L,11,L):(1,1,F,44)
 403 : tes-eqn : f(x,f(x,y)) = f(x,y) : cp(401,L.1,11,L):(1,1,F,26)
 408 : tes-rule : f(x,f(x,y)) -> f(x,y) : orient(403,u):(1,1,F,26)
 505 : tes-eqn : j(x,j(y,z)) = j(z,j(y,x)) : cp(54,L.2,5,L):(1,1,F,214)
 621 : tes-rule : g(j(x,g(y))) -> j(y,g(x)) : orient(309,u):(1,1,F,41)
 660 : tes-eqn : j(g(x),g(y)) = g(j(y,x)) : cp(621,L.1.2,84,L):(2,2,F,46)
 667 : tes-rule : g(j(g(x),y)) -> j(x,g(y)) : orient(367,u):(3,2,F,62)
 746 : tes-rule : f(x,f(y,f(x,y))) -> f(x,y) : orient(402,u):(1,1,F,44)
 919 : tes-rule : j(g(x),g(y)) -> g(j(y,x)) : orient(660,u):(2,2,F,46)
 989 : tes-rule : j(x,j(y,j(g(j(x,y)),z))) -> z : orient(68,x):(1,1,F,128)
1048 : tes-eqn : x = j(y,j(g(j(z,y)),j(g(g(z)),x))) : cp(989,L.2.2.1.1,289,L):(2,2,F,99)
1049 : tes-eqn : x = j(y,j(g(j(z,y)),j(z,x))) : tes-red(1048,R.2.2.1,84,L):(3,2,F,99)
1050 : tes-eqn : x = j(y,j(z,j(x,g(j(z,y))))) : tes-red(1049,R.2,54,L):(4,2,F,99)
1089 : tes-eqn : x = j(y,j(z,j(g(j(z,y)),x))) : cp(989,L.2.2.1.1,5,L):(2,2,F,128)
1162 : tes-rule : j(x,j(y,j(z,g(j(x,y))))) -> z : orient(99,x):(1,1,F,99)
1325 : tes-rule : j(x,j(y,j(z,g(x)))) -> j(y,z) : orient(100,x):(1,1,F,131)
```

```
1364 : tes-eqn : j(x,y) = j(j(y,z),j(x,g(z))) : cp(1325,L.2.2,348,L):(3,2,F,102)
1365 : tes-eqn : j(x,y) = j(g(z),j(j(y,z),x)) : tes-red(1364,R,54,R):(4,2,F,102)
1366 : tes-eqn : j(x,y) = j(g(z),j(y,j(z,x))) : tes-red(1365,R.2,40,L):(5,2,F,102)
1385 : tes-eqn : j(x,y) = j(g(z),j(x,j(y,z))) : cp(1325,L.2.2.2,84,L):(2,2,F,102)
1447 : tes-rule : j(x,j(y,j(g(x),z))) -> j(y,z) : orient(102,u):(2,1,F,158)
1617 : tes-rule : j(x,j(y,j(z,g(j(y,x))))) -> z : orient(1050,x):(4,2,F,99)
1828 : tes-rule : j(x,j(y,j(g(j(y,x)),z))) -> z : orient(1089,x):(2,2,F,128)
2111 : tes-rule : j(g(x),j(y,j(x,z))) -> j(z,y) : orient(1366,x):(5,2,F,102)
2366 : tes-rule : j(g(x),j(y,j(z,x))) -> j(y,z) : orient(1385,x):(2,2,F,102)
2625 : tes-rule : j(f(x,y),f(x,z)) -> f(x,j(y,z)) : orient(8,x):(0,0,T,66)
2626 : tes-eqn : f(x,j(x,y)) = j(x,f(x,y)) : cp(2625,L.1,11,L):(0,0,T,0)
2627 : tes-eqn : f(x,j(y,x)) = j(f(x,y),x) : cp(2625,L.2,11,L):(0,0,T,0)
2628 : tes-eqn : f(x,j(y,x)) = j(x,f(x,y)) : tes-red(2627,R,5,L):(0,0,T,0)
2666 : tes-rule : f(x,j(x,y)) -> j(x,f(x,y)) : orient(2626,u):(0,0,T,57)
2679 : tes-eqn : j(x,f(x,0())) = f(x,x) : cp(2666,L.2,13,L):(0,0,T,0)
2680 : tes-eqn : j(x,f(x,0())) = x : tes-red(2679,R,11,L):(0,0,T,0)
2685 : tes-eqn : j(x,f(x,j(g(x),y))) = f(x,y) : cp(2666,L.2,60,L):(1,1,F,64)
2686 : tes-eqn : j(x,f(x,j(y,g(x)))) = f(x,y) : cp(2666,L.2,91,L):(1,1,F,45)
2717 : tes-rule : j(x,f(x,0())) -> x : orient(2680,u):(0,0,T,27)
2731 : tes-eqn : f(g(x),0()) = j(x,g(x)) : cp(60,L.2,2717,L):(0,0,T,0)
2732 : tes-eqn : f(g(x),0()) = 0() : tes-red(2731,R,19,L):(0,0,T,0)
2777 : tes-rule : f(g(x),0()) -> 0() : orient(2732,u):(0,0,T,8)
2797 : tes-eqn : 0() = f(x,0()) : cp(2777,L.1,84,L):(0,0,T,0)
2800 : tes-rule : f(x,0()) -> 0() : orient(2797,x):(0,0,T,17)
2807 : tes-eqn : f(x,f(0(),y)) = f(0(),y) : cp(401,L.1,2800,L):(1,1,F,27)
2821 : tes-rule : f(x,f(0(),y)) -> f(0(),y) : orient(2807,u):(1,1,F,27)
2863 : tes-rule : f(x,j(y,x)) -> j(x,f(x,y)) : orient(2628,u):(0,0,T,56)
2886 : tes-eqn : j(g(x),f(g(x),x)) = f(g(x),0()) : cp(2863,L.2,19,L):(0,0,T,0)
2887 : tes-eqn : j(g(x),f(g(x),x)) = 0() : tes-red(2886,R,2800,L):(0,0,T,0)
2947 : tes-rule : j(g(x),f(g(x),x)) -> 0() : orient(2887,u):(0,0,T,33)
2964 : tes-eqn : f(g(x),x) = j(x,0()) : cp(60,L.2,2947,L):(0,0,T,0)
2965 : tes-eqn : f(g(x),x) = x : tes-red(2964,R,13,L):(0,0,T,0)
3047 : tes-rule : f(g(x),x) -> x : orient(2965,u):(0,0,T,12)
3051 : tes-eqn : f(g(x),f(x,y)) = f(x,y) : cp(401,L.1,3047,L):(1,1,F,29)
3063 : tes-eqn : g(x) = f(x,g(x)) : cp(3047,L.1,84,L):(1,1,F,16)
3066 : tes-rule : f(x,g(x)) -> g(x) : orient(3063,x):(1,1,F,16)
3068 : tes-eqn : f(x,f(g(x),y)) = f(g(x),y) : cp(401,L.1,3066,L):(2,2,F,29)
3093 : tes-rule : f(g(x),f(x,y)) -> f(x,y) : orient(3051,u):(1,1,F,29)
3155 : tes-rule : f(x,f(g(x),y)) -> f(g(x),y) : orient(3068,u):(2,2,F,29)
3230 : tes-rule : j(x,f(x,j(g(x),y))) -> f(x,y) : orient(2685,u):(1,1,F,64)
3364 : tes-rule : j(x,f(x,j(y,g(x)))) -> f(x,y) : orient(2686,u):(1,1,F,45)
3460 : tes-rule : j(f(x,y),f(z,y)) -> f(j(x,z),y) : orient(9,x):(0,0,T,63)
3461 : tes-eqn : f(j(x,y),x) = j(x,f(y,x)) : cp(3460,L.1,11,L):(0,0,T,0)
3462 : tes-eqn : f(j(x,y),y) = j(f(x,y),y) : cp(3460,L.2,11,L):(0,0,T,0)
3463 : tes-eqn : f(j(x,y),y) = j(y,f(x,y)) : tes-red(3462,R,5,L):(0,0,T,0)
3499 : tes-eqn : f(j(x,y),f(0(),z)) = j(f(0(),z),f(y,f(0(),z))) : cp(3460,L.1,2821,L):(2,2,F,41)
3500 : tes-eqn : f(0(),x) = j(f(0(),x),f(y,f(0(),x))) : tes-red(3499,L,2821,L):(3,2,F,41)
3501 : tes-eqn : f(0(),x) = j(f(0(),x),f(0(),x)) : tes-red(3500,R.2,2821,L):(4,2,F,41)
3502 : tes-eqn : f(0(),x) = f(0(),j(x,x)) : tes-red(3501,R,2625,L):(5,2,F,41)
3536 : tes-rule : f(0(),j(x,x)) -> f(0(),x) : orient(3502,x):(5,2,F,41)
3604 : tes-rule : f(j(x,y),x) -> j(x,f(y,x)) : orient(3461,u):(0,0,T,49)
3624 : tes-eqn : j(x,f(0(),x)) = f(x,x) : cp(3604,L.1,13,L):(0,0,T,0)
3625 : tes-eqn : j(x,f(0(),x)) = x : tes-red(3624,R,11,L):(0,0,T,0)
3626 : tes-eqn : j(x,f(g(x),x)) = f(0(),x) : cp(3604,L.1,19,L):(0,0,T,0)
3627 : tes-eqn : j(x,x) = f(0(),x) : tes-red(3626,L.2,3047,L):(0,0,T,72)
3676 : tes-rule : j(x,f(0(),x)) -> x : orient(3625,u):(0,0,T,39)
3861 : tes-eqn : x = j(x,j(x,x)) : cp(3676,L.2,3627,R):(0,0,T,0)
3935 : tes-rule : j(x,j(x,x)) -> x : orient(3861,x):(0,0,T,52)
3975 : tes-eqn : g(x) = j(j(x,x),g(x)) : cp(289,L.2.1,3935,L):(0,0,T,0)
3976 : tes-eqn : g(x) = j(g(x),j(x,x)) : tes-red(3975,R,5,L):(0,0,T,0)
3977 : tes-eqn : g(x) = j(x,j(x,g(x))) : tes-red(3976,R,54,L):(0,0,T,0)
3978 : tes-eqn : g(x) = j(x,0()) : tes-red(3977,R.2,19,L):(0,0,T,0)
3979 : tes-eqn : g(x) = x : tes-red(3978,R,13,L):(0,0,T,0)
4109 : tes-rule : g(x) -> x : orient(3979,u):(0,0,T,0)
4120 : tes-rule : j(x,x) -> 0() : tes-red(19,L.2,4109,L):(1,0,F,28)
4124 : tes-rule : j(x,j(x,y)) -> y : tes-red(60,L.2.1,4109,L):(0,0,T,0)
```

```
4125 : tes-rule : j(x,j(y,x)) -> y : tes-red(91,L.2.2,4109,L):(1,0,F,46)
4130 : tes-rule : j(x,j(y,j(z,x))) -> j(y,z) : tes-red(1325,L.2.2.2,4109,L):(2,1,F,62)
4131 : tes-rule : j(x,j(y,j(x,z))) -> j(y,z) : tes-red(1447,L.2.2.1,4109,L):(3,1,F,61)
4153 : tes-rule : j(x,x) -> 0() : orient(4120,u):(1,0,F,28)
4154 : tes-rule : f(0(),0()) -> f(0(),x) : tes-red(3536,L.2,4153,L):(6,2,F,13)
4155 : tes-eqn : 0() = f(0(),x) : tes-red(4154,L,11,L):(7,2,F,13)
4203 : tes-rule : f(0(),x) -> 0() : orient(4155,x):(7,2,F,13)
4247 : tes-rule : j(x,j(x,y)) -> y : orient(4124,u):(0,0,T,44)
4251 : tes-eqn : j(j(x,y),f(j(x,y),x)) = f(j(x,y),y) : cp(2863,L.2,4247,L):(0,0,T,0)
4252 : tes-eqn : j(j(x,y),j(x,f(y,x))) = f(j(x,y),y) : tes-red(4251,L.2,3604,L):(0,0,T,0)
4253 : tes-eqn : j(x,j(f(y,x),j(x,y))) = f(j(x,y),y) : tes-red(4252,L,54,L):(0,0,T,0)
4254 : tes-eqn : j(x,j(j(x,y),f(y,x))) = f(j(x,y),y) : tes-red(4253,L.2,5,L):(0,0,T,0)
4255 : tes-eqn : j(x,j(x,j(y,f(y,x)))) = f(j(x,y),y) : tes-red(4254,L.2,40,L):(0,0,T,0)
4256 : tes-eqn : j(x,f(x,y)) = f(j(y,x),x) : tes-red(4255,L,4247,L):(0,0,T,0)
4325 : tes-rule : j(x,j(y,x)) -> y : orient(4125,u):(1,0,F,46)
4422 : tes-rule : f(j(x,y),y) -> j(y,f(x,y)) : orient(3463,u):(0,0,T,30)
4501 : tes-rule : j(x,j(y,j(z,x))) -> j(y,z) : orient(4130,u):(2,1,F,62)
4634 : tes-rule : j(x,j(y,j(x,z))) -> j(y,z) : orient(4131,u):(3,1,F,61)
4786 : tes-eqn : j(x,f(x,y)) = j(x,f(y,x)) : tes-red(4256,R,4422,L):(0,0,T,59)
4820 : tes-eqn : j(x,f(j(x,y),x)) = j(x,j(x,f(x,y))) : cp(4786,L.2,2666,L):(0,0,T,0)
4821 : tes-eqn : j(x,j(x,f(y,x))) = j(x,j(x,f(x,y))) : tes-red(4820,L.2,3604,L):(0,0,T,0)
4822 : tes-eqn : f(x,y) = j(y,j(y,f(y,x))) : tes-red(4821,L,4247,L):(0,0,T,0)
4823 : tes-eqn : f(x,y) = f(y,x) : tes-red(4822,R,4247,L):(0,0,T,0)
4991 : tes-final : f(a(),b()) = f(b(),a()) : instance(10,4823):(0,0,T,0)
```

In order to obtain more readable proofs, the `proof` program was developed. It is able to convert a PCL proof to a proof appropriate to human understanding.

Terms that are to be replaced are underlined, while inserted terms are set in bold face.

Consider the following set of axioms:

Axiom 1: f(x, x) = x
Axiom 2: j(0, x) = x
Axiom 3: j(x, 0) = x
Axiom 4: j(g(x), x) = 0
Axiom 5: j(x, g(x)) = 0
Axiom 6: j(x, y) = j(y, x)
Axiom 7: j(j(x, y), z) = j(x, j(y, z))
Axiom 8: f(f(x, y), z) = f(x, f(y, z))
Axiom 9: f(x, j(y, z)) = j(f(x, y), f(x, z))
Axiom 10: f(j(x, y), z) = j(f(x, z), f(y, z))

This theorem holds true:

Theorem 1: f(a, b) = f(b, a)

Proof:

Lemma 1: $j(u, j(g(u), z)) = z$

$$\begin{aligned} \underline{j(u, j(g(u), z))} &= \mathbf{j(\underline{j(u, g(u))}, z)} \\ &= \underline{j(\mathbf{0}, z)} \\ &= \mathbf{z} \end{aligned}$$

Lemma 2: $g(g(u)) = u$

$$\begin{aligned} \underline{g(g(u))} &= \mathbf{j(u, \underline{j(g(u), g(g(u)))})} \\ &= \underline{j(u, \mathbf{0})} \\ &= \mathbf{u} \end{aligned}$$

Lemma 3: $j(g(z), f(g(z), z)) = 0$

$$\begin{aligned} \underline{j(g(z), f(g(z), z))} &= \mathbf{j(f(g(z), z), \underline{g(z)})} \\ &= \underline{j(f(g(z), z), \mathbf{f(g(z), g(z))})} \\ &= \mathbf{f(g(z), \underline{j(z, g(z))})} \\ &= f(\underline{g(z)}, \mathbf{0}) \\ &= \underline{f(\mathbf{g(g(g(z)))}, 0)} \\ &= \mathbf{j(g(g(z)), \underline{j(g(g(g(z)))}, f(g(g(g(z))), 0)))} \\ &= j(g(g(z)), \underline{j(\mathbf{f(g(g(g(z))), g(g(g(z))))}, f(g(g(g(z))), 0))}) \\ &= j(g(g(z)), \mathbf{f(g(g(g(z))), \underline{j(g(g(g(z))), 0)})}) \\ &= j(g(g(z)), \underline{f(g(g(g(z))), \mathbf{g(g(g(z)))})}) \\ &= \underline{j(g(g(z)), \mathbf{g(g(g(z)))})} \\ &= \mathbf{0} \end{aligned}$$

Lemma 4: $f(j(v, y), v) = j(v, f(y, v))$

$$\begin{aligned} \underline{f(j(v, y), v)} &= \mathbf{j(\underline{f(v, v)}, f(y, v))} \\ &= j(\mathbf{v}, f(y, v)) \end{aligned}$$

Lemma 5: g(y) = j(j(y, y), g(y))

g(y) = **j(g(j(y, j(y, y))), j(g(g(j(y, j(y, y)))), g(y)))**
= j(g(j(y, j(y, y))), **j(g(y), g(g(j(y, j(y, y))))))**
= j(g(j(y, j(y, y))), j(g(y), **j(y, j(y, y))**))
= j(g(j(y, j(y, y))), j(g(y), j(**g(g(y))**, j(y, y))))
= j(g(j(y, j(y, y))), **j(y, y)**)
= **j(j(y, y), g(j(y, j(y, y))))**
= j(j(y, y), g(j(y, j(y, **j(y, 0)**))))
= j(j(y, y), g(j(y, j(y, j(y, **j(g(y), f(g(y), y))**)))))
= j(j(y, y), g(j(y, j(y, **f(g(y), y)**))))
= j(j(y, y), g(j(y, **f(j(y, g(y)), y)**)))
= j(j(y, y), g(j(y, f(**0**, y))))
= j(j(y, y), g(**f(j(y, 0), y)**))
= j(j(y, y), g(f(**y**, y)))
= j(j(y, y), g(**y**))

Lemma 6: j(u, j(u, z)) = z

j(u, j(u, z)) = j(u, j(**j(u, 0)**, z))
= j(u, j(j(u, **j(u, g(u))**), z))
= j(u, j(**j(j(u, u), g(u))**, z))
= j(u, j(**g(u)**, z))
= **z**

Lemma 7: j(z, f(z, w)) = j(z, f(w, z))

j(z, f(z, w)) = **j(w, j(w, j(z, f(z, w))))**
= j(w, **j(j(w, z), f(z, w))**)
= j(w, **j(f(z, w), j(w, z))**)
= **j(j(w, f(z, w)), j(w, z))**
= **j(j(w, z), j(w, f(z, w)))**
= j(j(w, z), **f(j(w, z), w)**)
= **j(f(j(w, z), w), j(w, z))**
= j(f(j(w, z), w), **f(j(w, z), j(w, z)))**
= **f(j(w, z), j(w, j(w, z)))**
= f(j(w, z), **z**)
= **j(f(w, z), f(z, z))**
= j(f(w, z), **z**)
= **j(z, f(w, z))**

Theorem 1: f(a, b) = f(b, a)

f(a, b) = **j(b, j(b, f(a, b)))**
= j(b, **f(j(b, a), b)**)
= **j(b, f(b, j(b, a)))**
= j(b, **j(f(b, b), f(b, a))**)
= j(b, j(**b**, f(b, a)))
= **f(b, a)**

Appendix B

Problems Used

The subset of unit-equality problems of the TPTP library [SS97] we used throughout this work contains 403 problems from eight domains.

BOO Boolean Algebra
: A Boolean algebra is a set with two binary operations that are idempotent, commutative, associative and mutually distributive and an unary operation of complementation. Additionally, there are universal bounds 0 and 1. A Boolean algebra can be described by three equations: commutativity, associativity, and the so-called *Huntington* equation.

COL Combinatory logic
: Combinatory logic is a system satisfying two combinators and reflexivity, symmetry, transitivity, and two equality substitution axioms for a function that exists implicitly for applying one combinator to another.

GRP Groups
: A group is a set A and a binary, associative operation $+ : A \times A \mapsto A$ with an identity element $i \in A| \quad \forall a \in A : \quad i + a = a$ and each element $x \in G$ has its inverse $y \in G| \quad x + y = i$.

LCL Logic Calculi
: A logic calculus consists of axioms and inference rules that can be used to prove theorems.

LDA Left distributive Algebra
: Left distributive algebras are algebras with a binary operation $\circ$ generated by a single element 1 using the left distributive law $a \circ (b \circ c) = a \circ b \circ (a \circ c)$.

RNG Rings
: A ring is an Abelian (commutative) group with another binary, associative, and distributive operation $\circ : A \times A \mapsto A$ with an identity element $j \in A| \quad \forall a \in A : \quad j \circ a = a$.

ROB Robbins Algebra
: A Robbins algebra is defined exactly like a Boolean algebra but the Huntington equation is replaced by the *Robbins* equation. It was actually

shown by an ATP system (EQP), that every Robbins algebra is a Boolean algebra [McC97].

SYN Syntactic
Syntactic problems are problems without an obvious semantic interpretation.

Appendix C

Changes to DISCOUNT

C.1 New Command-Line Options

As δ_{MIX} subsumes δ_{FTR} and δ_{TSM}, the final implementation just features δ_{MIX}. Setting w_{TSM} to 0 results in δ_{FTR}; setting all feature weights to 0 yields δ_{TSM}.

`-max_examples <arg>` Maximum number of examples used for building the TSM (Default: 10)

`-max_delta <arg>` Only examples with $\delta_{MIX} \leq$ `<arg>` will be used for learning (Default: 0.5)

`-w_TSM <arg>` Weight of δ_{TSM} (Default: 1)

`-w_NA <arg>` Feature-weight: Number of Axioms (Default: 0)

`-w_AD <arg>` Feature-weight: Average term Depth (Default: 0)

`-w_DD <arg>` Feature-weight: Depth standard Deviation (Default: 0)

`-w_GD <arg>` Feature-weight: Goal Depth (Default: 0)

`-w_AF <arg>` Feature-weight: Arity Frequency (Default: 0)

`-N` **or** `-no_selection` Disable example selection (Default: false)

C.2 Knowledge Base Maintenance

The new knowledge base is in *version 1.2* format and contains an additional flag for example selection (see figure C.1).

```
Version = "1.2 (Mar 2 1998)"

PreserveArity = TRUE
PosAndNeg     = FALSE
HasSpecDoms   = FALSE
HasGoalDoms   = FALSE
HasSelData    = TRUE
```

Figure C.1: An example `kb_variables` file

`kb_create` now also creates a folder called `SELECTIONDATA`, and `kb_insert` computes the features and inserts them and the axioms of the given proof into `SELECTIONDATA`.

Both programs use the new option "`-E`" to ignore example selection.

Bibliography

[AA90] **Anantharaman, D.; Andrianarievelo, N.**: *Heuristical criteria in refutational theorem proving*, Proc. DISCO'90, LNCS 429:184–193, 1990.

[Ave95] **Avenhaus, J.**: *Reduktionssysteme*, Springer, 1995.

[AD93] **Avenhaus, J.; Denzinger, J.**: *Distributing equational theorem proving*, Proc. of the RTA-93, Montreal, LNCS 690:62–76, Springer, 1993.

[BDP89] **Bachmair, L.; Derschowitz, N.; Plaisted, D.A.**: *Completion without Failure*, Coll. on the Resolution of Equations in Algebraic Structures, Austin, Academic Press, 1989.

[Bir35] **Birkhoff, G.**: *On the Structure of Abstract Algebras*, Proc. Cambridge Philos. Society, 31:433–454, 1935.

[DF94] **Denzinger, J.; Fuchs, M.**: *Goal Orientated Equational Theorem Proving Using Teamwork*, Proc. 8th KI-94, Saarbrücken, LNAI 861:343–354, Springer, 1994.

[Duf95] **Dufner, G.**: *Topologie*, Skript zur Vorlesung im SS95, Department of Mathematics, Technical University Munich, 1995.

[DFGS98] **Denzinger, J.; Fuchs, M.; Goller, C.; Schulz, S.**: *Learning from Previous Proof Experience*, Journal of Symbolic Computation, to appear, 1998.

[DKS97] **Denzinger, J.; Kronenburg, M.; Schulz, S.**: *DISCOUNT: A Distributed and Learning Equational Prover*, Journal of Automated Reasoning, 18(2):189–198, 1997.

[DS94] **Denzinger, J.; Schulz, S.**: *Analysis and Representation of Equational Proofs Generated by a Distributed Completion Based Proof System*, SEKI-Report SR-94-05, Kaiserslautern, 1994.

[DS95] **Denzinger, J.; Schulz, S.**: *Automatic Acquisition of Search Control Knowledge from Multiple Proof Attempts*, Journal of Information and Computation, to appear, 1998.

[DS96a] **Denzinger, J.; Schulz, S.**: *Learning Domain Knowledge to Improve Theorem Proving*, Proc. of CADE-13, New Brunswick, LNAI 1104:62–76, Springer, 1996.

[DS96b] **Denzinger, J.; Schulz, S.**: *Recording and Analysing Knowledge-Based Distributed Deduction Processes*, Journal of Symbolic Computation, 21:523–541, 1996.

[Gue97] **Guess, J.C.**: *Professional English*, Oldenbourg Verlag, 1997.

[HBF96] **Hillenbrand, T.; Buch, A.; Fettig, R.**: *On Gaining Efficiency in Completion-Based Theorem Proving*, Proc. of the 7th RTA, LNCS 1103:432–435, Springer, 1996.

[Hue80] **Huet, G.**: *Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems*, Journal of ACM 27, 4:798–821, 1980.

[Kar74] **Karzel, H.**: *Lineare Algebra und Analytische Geometrie*, Skript zur Vorlesung im WS73/74 und SS74, Department of Mathematics, Technical University Munich, 1992.

[KB70] **Knuth, D.E.; Bendix, P.B.**: *Simple Word Problems in Universal Algebras*, Computational Problems in Abstract Algebra, ed.: J. Leech, pp. 263–297, Pergamon Press, 1970.

[LO82] **Lusk, E.L.; Overbeck, R.A.**: *A Short Problem Set for Testing Systems that Include Equality Reasoning*, Argonne National Laboratory, Illinois, 1982.

[LW92] **Lusk, E.L.; Wos, L.**: *Benchmark problems in which equality plays the major role*, Proc. CADE-11, LNAI 607:781–785, Saratoga Springs, Springer, 1992.

[Mar94] **Markel, M.**: *Writing in the Technical Fields*, New York, IEEE Press, 1994.

[McC97] **McCune, W.** : *Solution of the Robbins Problem*, Journal of Automated Reasoning, 19(3):263–276, 1997.

[Sch93] **Schulz, S.**: *Analyse und Transformation von Gleichheitsbeweisen*, Projektarbeit, University of Kaiserslautern, 1993.

[Sch95] **Schulz, S.**: *Explanation Based Learning for Distributed Equational Deduction*, Diploma Thesis, University of Kaiserslautern, 1995.

[Sch98] **Schulz, S.**: *Term Space Mapping for DISCOUNT*, Proc. of the CADE-15 Workshop on *Using AI methods in Deduction*, Lindau, 1998.

[Sut98] **Sutcliffe, G.**: *CADE-15 ATP System Competition*, http://www.cs.jcu.edu.au/ tptp/CASC-15/, 1998.

[SS97] **Suttner, C.B.; Sutcliffe, G.**: *The TPTP Problem Library v2.1.0*, Technical Report AR-97-04, Technical University Munich, 1997.

[Win84] **Winston, P.H.**: *Artificial Intelligence*, Addison-Wesley, 1984.

[Wos96] **Wos, L.**: *The Automation of Reasoning*, Academic Press, 1996.