A Focusing Inverse Method Theorem Prover for First-Order Linear Logic

Kaustuv Chaudhuri and Frank Pfenning*

Department of Computer Science
Carnegie Mellon University
kaustuv@cs.cmu.edu and fp@cs.cmu.edu

Abstract. We present the theory and implementation of a theorem prover for first-order intuitionistic linear logic based on the inverse method. The central proof-theoretic insights underlying the prover concern resource management and focused derivations, both of which are traditionally understood in the domain of backward reasoning systems such as logic programming. We illustrate how resource management, focusing, and other intrinsic properties of linear connectives affect the basic forward operations of rule application, contraction, and forward subsumption. We also present some preliminary experimental results obtained with our implementation.

1 Introduction

Linear logic [1] extends classical logic by internalizing *state*. This is achieved by forcing *linear assumptions* to be used exactly once during a proof. Introducing or using a linear assumption then corresponds to a change in state. This alters the fundamental character of the logic so that, for example, even the propositional fragment is undecidable. The expressive power of linear logic has been exploited to provide a logical foundation for phenomena in a number of diverse domains, such as planning, concurrency, functional programming, type systems, and logic programming.

Despite this wide range of potential applications, there has been relatively little effort devoted to theorem proving for linear logic. One class of prior work consists of fundamental proof-theoretic studies of the properties of linear logic [2–4]. Most of these are concerned with backward reasoning, that is, with analyzing the structure of proof search starting from a given goal sequent. The most important results, such as the discovery of focusing [2] and solutions to the problem of resource management [5, 6] have made their way into logic programming languages [7, 8], but not implemented theorem provers. The term *resource management* in this context refers to the problem of efficiently ensuring that the single-use semantics of linear assumptions is respected during proof search.

Another class of prior work is directly concerned with theorem proving for linear logic in the forward direction. Here, we are only familiar with Mints's theoretical study

* This work has been supported by the Office of Naval Research (ONR) under grant MURI N00014-04-1-0724 and by the National Science Foundation (NSF) under grant CCR-0306313.

of resolution for linear logic [9] and Tammet's implementation of a resolution prover for classical propositional linear logic [10].

In this paper we describe a theorem prover for first-order intuitionistic linear logic based on the inverse method. This prover can be seen as synthesizing the forward-reasoning techniques developed by Mints and Tammet with the proof-theoretic analysis of backward-reasoning by Andreoli and others by incorporating focusing into the inverse method. This allows the inverse method to proceed in much bigger steps than single inferences, generating many fewer sequents to be kept and tested for subsumption. In addition, we develop a new approach to resource management for reasoning in the forward direction and show how it extends to the first-order case. We further treat the delicate interactions of resource management with contraction and subsumption, two critical operations in a forward inference engine. Finally, we present some experimental results which demonstrate a significant speedup over Tammet's prover and help us quantify the effects of several internal optimizations.

Most closely related to the results reported in this paper is a recent submission by the first author [11] which presents focusing for the inverse method in intuitionistic *propositional* linear logic, but does not discuss many of the necessary implementation choices. We therefore concentrate here on the first-order and implementation aspects of the prover and only briefly sketch focusing. The propositional prover from [11] and the first-order prover here are separate implementations, since the propositional case (even though also undecidable) affords a number of optimizations that do not directly apply in the first-order case.

Another related development are recent implementations of theorem provers for the logic of bunched implications [12, 13]. The logic of bunched implications and linear logic have a common core, so some techniques may be transferable, something we plan to consider in future work. Méry's prover [12] uses labeled tableaux in a goal-directed manner and is therefore quite different from ours. Donnelly et al. [13] use the inverse method, but the essential difficulty in its design concerns a particular interaction between weakening and contraction that is germane to bunched implication, but foreign to linear logic. Moreover, both provers are quite preliminary at this stage and do not incorporate techniques such as subsumption or indexing.

The remainder of the paper is organized as follows. In sect. 2 we give the briefest sketch of a cut-free sequent calculus for intuitionistic linear logic and its critical subformula property. In sect. 3 we present the ground inverse method underlying our prover and show how we solve the resource management problem. In sect. 4 we show how to lift the results and operations to the case of sequents with free variables which are necessary for a first-order prover. In sections 5 and 6 we present the main points in the design of an efficient implementation and some experimental results. In sect. 7 we conclude with remarks regarding the scope of our methods and future work.

2 Backward Sequent Calculus and the Subformula Property

We use a backward cut-free sequent calculus for propositions constructed out of the linear connectives $\{\otimes, \mathbf{1}, \multimap, \&, \top, !, \forall, \exists\}$. To simplify the presentation slightly we leave out $\oplus$ and $\mathbf{0}$, though the implementation supports them and some of the experiments

in Sec. 6 use them. Propositions are written using uppercase letters A, B, C, with P standing for atomic propositions. Atomic propositions contain terms $\mathtt{t}$, which can be term variables x, parameters $\mathtt{a}$, $\mathtt{b}$, or function applications $f(\mathtt{t}_1, \ldots, \mathtt{t}_n)$. As usual, term constants are treated as nullary functions. The sequent calculus is a standard fragment of JILL [14], containing dyadic two-sided sequents of the form $\Gamma\,;\Delta \Longrightarrow C$: the zone Γ contains the unrestricted hypotheses, and Δ contains the linear hypotheses. Both Γ and Δ are unordered. For the rules of this calculus we refer the reader to [14, page 14]; the missing quantifier rules are below:

$$\frac{\Gamma\,;\Delta \Longrightarrow [\mathtt{a}/x]A}{\Gamma\,;\Delta \Longrightarrow \forall x.\, A}\ \forall R^{\mathtt{a}} \qquad \frac{\Gamma\,;\Delta, [\mathtt{t}/x]A \Longrightarrow C}{\Gamma\,;\Delta, \forall x.\, A \Longrightarrow C}\ \forall L \qquad \frac{\Gamma\,;\Delta \Longrightarrow [\mathtt{t}/x]A}{\Gamma\,;\Delta \Longrightarrow \exists x.\, A}\ \exists R \qquad \frac{\Gamma\,;\Delta, [\mathtt{a}/x]A \Longrightarrow C}{\Gamma\,;\Delta, \exists x.\, A \Longrightarrow C}\ \exists L^{\mathtt{a}}$$

where the superscript indicates the proviso that $\mathtt{a}$ may not occur in the conclusion. Also in [14] are the standard structural properties for the unrestricted hypotheses and admissibility of cut, which extend naturally to the first-order setting.

Definition 1 (subformulas). *A* decorated formula *is a tuple $\langle A, s, w\rangle$ where A is a proposition, s is a* sign *(+ or −) and w is a* weight *(h or l). The* subformula relation *$\leq$ is the smallest reflexive and transitive relation between decorated subformulas to satisfy the following conditions.*

$$\langle A, s, h\rangle \leq \langle !\,A, s, w\rangle \quad \langle A, \overline{s}, l\rangle \leq \langle A \multimap B, s, w\rangle \quad \langle B, s, l\rangle \leq \langle A \multimap B, s, w\rangle$$

$$\left.\begin{array}{ll} \langle [\mathtt{a}/x]A, +, l\rangle \leq \langle \forall x.\, A, +, w\rangle & \langle [\mathtt{a}/x]A, -, l\rangle \leq \langle \exists x.\, A, -, w\rangle \\ \langle [\mathtt{t}/x]A, +, l\rangle \leq \langle \exists x.\, A, +, w\rangle & \langle [\mathtt{t}/x]A, -, l\rangle \leq \langle \forall x.\, A, -, w\rangle \end{array}\right\} \quad \begin{array}{l} \cdots\ \mathtt{a} \text{ any parameter} \\ \ \mathtt{t} \text{ any term} \end{array}$$

$$\langle A_i, s, l\rangle \leq \langle A_1 \star A_2, s, w\rangle \qquad \ldots \star \in \{\otimes, \&\}, i \in \{1, 2\}$$

where $\overline{s}$ is the opposite of s. The notation $*$ can stand for either h or l, as necessary. Decorations and the subformula relation are lifted to (multi)sets in the obvious way.

We also need the notion of free subformula *which is precisely as above, except that we use A instead of $[\mathtt{t}/x]A$ for subformulas of positive existentials and negative universals.*

Property 2 (strong subformula property). *In any sequent $\Gamma'\,;\Delta' \Longrightarrow C'$ used in a proof of $\Gamma\,;\Delta \Longrightarrow C$:*

$$\langle \Gamma', -, h\rangle \cup \langle \Delta', -, *\rangle \cup \{\langle C', +, *\rangle\} \leq \langle \Gamma, -, h\rangle \cup \langle \Delta, -, l\rangle \cup \{\langle C, +, l\rangle\}$$

For the rest of the paper, all rules are to be understood as being restricted to decorated subformulas of the goal sequent. A right rule is only applicable if the principal formula A is a positive decorated subformula of the goal sequent (i.e., $\langle A, +, *\rangle \leq$ goal); similarly, a left rule only applies to A if $\langle A, -, *\rangle \leq$ goal. Of the judgmental rules, init is restricted to atomic subformulas that are *both* positive and negative decorated subformulas, and the copy rule is restricted to cases where the copied formula A is a heavy negative decorated subformula, i.e., $\langle A, -, h\rangle \leq$ goal.

3 Forward Sequent Calculus and Resource Management

Following the "recipe" for the inverse method outlined in [15], we first present a forward sequent calculus in a ground form (containing no term variables), and then lift it to free subformulas containing term and parameter variables and explicit unification.

As mentioned before, backward proof-search in linear logic suffers from certain kinds of non-determinism peculiar to the nature of linear hypotheses. A binary multiplicative rule like $\otimes R$ as indicated is actually deceptive as it hides the fact that the split Δ_1, Δ_2 of the linear zone is not structurally obvious. Linearity forces the division to be exact, but there are an exponential number of such divisions, which is an unreasonable branching factor for proof-search. This form of resource non-determinism is typical in backward search, but is of course entirely absent in a forward reading of multiplicative rules where the parts Δ_1 and Δ_2 are inputs. The nature of resource non-determinism in forward and backward search is therefore significantly different.

$$\frac{\Gamma\,;\Delta_1 \Longrightarrow A \quad \Gamma\,;\Delta_2 \Longrightarrow B}{\Gamma\,;\Delta_1,\Delta_2 \Longrightarrow A \otimes B}\ \otimes R$$

To distinguish forward from backward sequents, we shall use a single arrow ($\longrightarrow$), but keep the names of the rules the same. In the forward direction, the primary context management issue concerns rules where the conclusion cannot be simply assembled from the premisses. The backward $\top R$ rule has an arbitrary linear context Δ, and the unrestricted context Γ is also unknown in several rules such as init and $\top R$. For the unrestricted zone, this problem is solved in the usual (non-linear) inverse method by collecting only the needed unrestricted assumptions and remembering that they can be weakened if needed [15]. We adapt the solution to the linear zone, which may either be precisely determined (as in the case for initial sequents) or subject to weakening (as in the case for $\top R$). We therefore differentiate sequents whose linear context can be weakened and those who can not.

$$\frac{}{\Gamma\,;P \Longrightarrow P}\ \text{init} \qquad \frac{}{\Gamma\,;\Delta \Longrightarrow \top}\ \top R$$

Definition 3 (forward sequents). *A* forward sequent *is of the form* $\Gamma\,;\Delta \longrightarrow^w C$*, with* w *a Boolean (0 or 1) called the* weak-flag*. The correspondence between forward and backward sequents is governed by the following conditions:*

$$\Gamma\,;\Delta \longrightarrow^0 C \quad \textit{corresponds to} \quad \Gamma'\,;\Delta \Longrightarrow C \quad \textit{if } \Gamma \subseteq \Gamma'$$
$$\Gamma\,;\Delta \longrightarrow^1 C \quad \textit{corresponds to} \quad \Gamma'\,;\Delta' \Longrightarrow C \quad \textit{if } \Gamma \subseteq \Gamma' \textit{ and } \Delta \subseteq \Delta'$$

Sequents with $w = 1$ *are called* weakly linear *or simply* weak*, and those with* $w = 0$ *are* strongly linear *or* strong*.*

It is easy to see that weak sequents model affine logic, which is familiar from embeddings into linear logic that translate affine implications $A \to B$ as $A \multimap (B \otimes \top)$. Initial sequents are always strong, since their linear context cannot be weakened. On the other hand, $\top R$ always produces a weak sequent. The collection of inference rules for the forward calculus is in Fig. 1.

For binary rules, the unrestricted zones are simply juxtaposed. We can achieve the effect of taking their union by applying the explicit contraction rule (which is absent, but admissible in the backward calculus). The situation is not as simple for the linear zone. As shown above, in the backward direction the same linear zone is copied into both premisses of the $\&R$ rule: This rule is easily adapted to the forward direction when both premisses are strong:

$$\frac{\Gamma\,;\Delta \Longrightarrow A \quad \Gamma\,;\Delta \Longrightarrow B}{\Gamma\,;\Delta \Longrightarrow A \,\&\, B}\ \&R$$

$$\frac{\Gamma\,;\Delta \longrightarrow^0 A \quad \Gamma'\,;\Delta' \longrightarrow^0 B \quad (\Delta = \Delta')}{\Gamma,\Gamma'\,;\Delta \longrightarrow^0 A \,\&\, B}$$

judgmental rules

$$\frac{}{\cdot\,;\,P \longrightarrow^0 P}\ \text{init} \qquad \frac{\Gamma\,;\,\Delta, A \longrightarrow^w C}{\Gamma, A\,;\,\Delta \longrightarrow^w C}\ \text{copy} \qquad \frac{\Gamma, A, A\,;\,\Delta \longrightarrow^w C}{\Gamma, A\,;\,\Delta \longrightarrow^w C}\ \text{contr}$$

multiplicative connectives

$$\frac{\Gamma\,;\,\Delta \longrightarrow^w A \quad \Gamma'\,;\,\Delta' \longrightarrow^{w'} B}{\Gamma, \Gamma'\,;\,\Delta, \Delta' \longrightarrow^{w \vee w'} A \otimes B}\ \otimes R \qquad \frac{\Gamma\,;\,\Delta, A, B \longrightarrow^w C}{\Gamma\,;\,\Delta, A \otimes B \longrightarrow^w C}\ \otimes L \qquad \frac{\Gamma\,;\,\Delta, A_i \longrightarrow^1 C \quad (A_j \notin \Delta)}{\Gamma\,;\,\Delta, A_1 \otimes A_2 \longrightarrow^1 C}\ \otimes L_i$$
$$(i, j) \in \{(1,2),(2,1)\}$$

$$\frac{}{\cdot\,;\,\cdot \longrightarrow^0 \mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma\,;\,\Delta \longrightarrow^0 C}{\Gamma\,;\,\Delta, \mathbf{1} \longrightarrow^0 C}\ \mathbf{1}L \qquad \frac{\Gamma\,;\,\Delta, A \longrightarrow^w B}{\Gamma\,;\,\Delta \longrightarrow^w A \multimap B}\ \multimap R \qquad \frac{\Gamma\,;\,\Delta \longrightarrow^1 B \quad (A \notin \Delta)}{\Gamma\,;\,\Delta \longrightarrow^1 A \multimap B}\ \multimap R'$$

$$\frac{\Gamma\,;\,\Delta, B \longrightarrow^w C \quad \Gamma'\,;\,\Delta' \longrightarrow^{w'} A \quad (w = 0 \vee B \notin \Delta')}{\Gamma, \Gamma'\,;\,\Delta, \Delta', A \multimap B \longrightarrow^{w \vee w'} C}\ \multimap L$$

additive connectives

$$\frac{\begin{array}{c}\Gamma\,;\,\Delta_1 \longrightarrow^{w_1} A \\ \Gamma'\,;\,\Delta_2 \longrightarrow^{w_2} B \\ (\Delta_1/w_1 + \Delta_2/w_2 \rightsquigarrow \Delta)\end{array}}{\Gamma, \Gamma'\,;\,\Delta \longrightarrow^{w_1 \wedge w_2} A \,\&\, B}\ \&R \qquad \frac{}{\cdot\,;\,\cdot \longrightarrow^1 \top}\ \top R \qquad \frac{\Gamma\,;\,\Delta, A_i \longrightarrow^w C}{\Gamma\,;\,\Delta, A_1 \,\&\, A_2 \longrightarrow^w C}\ \&L_i \quad i \in \{1,2\}$$

exponentials

$$\frac{\Gamma\,;\,\cdot \longrightarrow^w A}{\Gamma\,;\,\cdot \longrightarrow^0 !A}\ !R \qquad \frac{\Gamma, A\,;\,\Delta \longrightarrow^w C}{\Gamma\,;\,\Delta, !A \longrightarrow^w C}\ !L \qquad \frac{\Gamma\,;\,\Delta \longrightarrow^0 C \quad (A \notin \Gamma)}{\Gamma\,;\,\Delta, !A \longrightarrow^0 C}\ !L'$$

quantifiers

$$\frac{\Gamma\,;\,\Delta \longrightarrow^w [\mathtt{a}/x]A}{\Gamma\,;\,\Delta \longrightarrow^w \forall x.\, A}\ \forall R^{\mathtt{a}} \qquad \frac{\Gamma\,;\,\Delta, [\mathtt{t}/x]A \longrightarrow^w C}{\Gamma\,;\,\Delta, \forall x.\, A \longrightarrow^w C}\ \forall L$$

$$\frac{\Gamma\,;\,\Delta \longrightarrow^w [\mathtt{t}/x]A}{\Gamma\,;\,\Delta \longrightarrow^w \exists x.\, A}\ \exists R \qquad \frac{\Gamma\,;\,\Delta, [\mathtt{a}/x]A \longrightarrow^w C}{\Gamma\,;\,\Delta, \exists x.\, A \longrightarrow^w C}\ \exists L^{\mathtt{a}}$$

Fig. 1. forward linear sequent calculus

If one premiss is weak and the other strong, the weak zone must be a subset of the strong zone:

$$\frac{\Gamma\,;\,\Delta \longrightarrow^0 A \quad \Gamma'\,;\,\Delta' \longrightarrow^1 B \quad (\Delta' \subseteq \Delta)}{\Gamma, \Gamma'\,;\,\Delta \longrightarrow^0 A \,\&\, B}$$

If both premisses are weak, then the conclusion is also weak, but what resources are present in the conclusion? In the ground case, we can simply take the maximal multiplicity for each proposition on the two premises. To see that this is sound, simply apply weakening to add the missing copies, equalizing the linear contexts in the premisses. It is also complete because the maximum represents the least upper bound. In the free variable calculus this analysis breaks down, because the two propositions in the linear contexts in weak premisses may also be equalized by substitution. In preparation, we

therefore introduce a non-deterministic *additive contraction* judgment which is used in the &*R* rule to generate multiple valid merges of the linear contexts of the premises.

Definition 4. *A* additive contraction judgement *is of the form* $\Delta/w + \Delta'/w' \rightsquigarrow \Delta''$ *where* Δ, Δ' *and* Δ'' *are linear contexts, and* w *and* w' *are weak-flags.* Δ, Δ', w, *and* w' *are inputs, and* Δ'' *is the output. The rules are as follows:*

$$\frac{}{\cdot/0 + \cdot/0 \rightsquigarrow \cdot} \qquad \frac{}{\cdot/1 + \Delta/0 \rightsquigarrow \Delta} \qquad \frac{}{\Delta/0 + \cdot/1 \rightsquigarrow \Delta} \qquad \frac{}{\Delta/1 + \Delta'/1 \rightsquigarrow \Delta, \Delta'}$$

$$\frac{\Delta/w + \Delta'/w' \rightsquigarrow \Delta''}{\Delta, A/w + \Delta', A/w' \rightsquigarrow \Delta'', A}$$

Note that $\rightsquigarrow$ *is non-deterministic because the fourth and fifth rule overlap. Note further that* $\Delta/1 + \Delta'/0 \rightsquigarrow \Delta'$ *iff* $\Delta \subseteq \Delta'$, *and for any* Δ'' *with* $\Delta \cup \Delta' \subseteq \Delta'' \subseteq \Delta, \Delta'$, *the judgment* $\Delta/1 + \Delta'/1 \rightsquigarrow \Delta''$ *is derivable.*

The conclusion of a binary multiplicative rule is weak if either of the premisses is weak; thus, the weak-flag of the conclusion is a Boolean-or of those of the premisses. Most unary rules are oblivious to the weakening decoration, which simply survives from the premiss to the conclusion. The exception is ! *R*, for which it is unsound to have a weak conclusion; there is no derivation of $\cdot\,;\top \Longrightarrow\, !\top$, for example.

Left rules with weak premisses require some attention. It is tempting to write the "weak" $\otimes L$ rules as:

$$\frac{\Gamma\,;\Delta, A \longrightarrow^1 C}{\Gamma\,;\Delta, A \otimes B \longrightarrow^1 C}\ \otimes L_1 \qquad \frac{\Gamma\,;\Delta, B \longrightarrow^1 C}{\Gamma\,;\Delta, A \otimes B \longrightarrow^1 C}\ \otimes L_2.$$

However, these rules would admit redundant inferences such as the following:

$$\frac{\Gamma\,;\Delta, A, B \longrightarrow^1 C}{\Gamma\,;\Delta, A, A \otimes B \longrightarrow^1 C}\ \otimes L_2.$$

We might as well have consumed both *A* and *B* to form the conclusion, and obtained a stronger result. The sensible strategy is: when *A* and *B* are both present, they must *both* be consumed. Otherwise, only apply the rule when one operand is present in a weak sequent. A similar observation can be made about all such rules: there is one weakness-agnostic form, and some possible refined forms to account for weak sequents.

Theorem 5 (soundness).

1. *If* $\Gamma\,;\Delta \longrightarrow^0 C$, *then* $\Gamma\,;\Delta \Longrightarrow C$.
2. *If* $\Gamma\,;\Delta \longrightarrow^1 C$, *then* $\Gamma\,;\Delta' \Longrightarrow C$ *for any* $\Delta' \supseteq \Delta$.

Proof (sketch). Structural induction on the forward derivation $\mathcal{F} :: \Gamma\,;\Delta \longrightarrow^w C$. The induction hypothesis is applicable for smaller derivations modulo α-renamings of parameters. □

For the completeness theorem we note that the forward calculus infers a possibly stronger form of the goal sequent.

Theorem 6 (completeness). *If* $\Gamma\,;\Delta \Longrightarrow C$, *then for some* $\Gamma' \subseteq \Gamma$:

1. *either* $\Gamma'\,;\Delta \longrightarrow^0 C$;
2. *or* $\Gamma'\,;\Delta' \longrightarrow^1 C$ *for some* $\Delta' \subseteq \Delta$

Proof (sketch). Structural induction on the backward derivation $\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow C$. □

4 Lifting to Free Variable Sequents

The calculus of the previous section uses only ground initial sequents, which is impossible for an implementation of the forward calculus. Continuing with the "recipe" from [15], in this section we present a lifted version of the calculus with explicit unification. We begin, as usual, by fixing a goal sequent $\Gamma_g \,;\, \Delta_g \longrightarrow^w C_g$ and considering only the free subformulas of this goal. In the presentation, the quantified propositions are silently α-renamed as necessary. In this calculus, every proposition on the left and right is accompanied by a substitution for some of its parameters or free term variables. These substitutions are built according the following grammar:

$$\begin{array}{lrll} \text{(substitutions)} & \sigma & ::= & \epsilon & \text{(identity)} \\ & & \mid & \sigma, \mathtt{a}_1/\mathtt{a}_2 & \text{(param-subst)} \\ & & \mid & \sigma, \mathtt{t}/x & \text{(term-subst)} \end{array}$$

A minor novel aspect of our formulation is that we distinguish parameters (which can be substituted only for each other) and variables (for which we can substitute arbitrary terms, including parameters). The distinction arises from the notion of subformula, since positive universal and negative existential formulas can only ever be instantiated with parameters in a cut-free backward sequent derivation. This sharpening sometimes removes unreachable initial sequents from consideration. Fortunately, the standard notion of most general unifier (written $\mathrm{mgu}(\sigma, \sigma')$) carries over in a straightforward way to this slightly more general setting. We make the customary assumption that substitutions are idempotent. We write $A[\sigma]$ (resp. $\mathtt{t}[\sigma]$) for the application of the substitution σ to the free subformula A (resp. term $\mathtt{t}$). Sequents in the free calculus contain formula/substitution pairs, written $A \cdot \sigma$. The composition of σ and ξ, written $\sigma\xi$, has the property $A[\sigma\xi] = (A[\sigma])[\xi]$. The composition of θ with every substitution in a zone Γ or Δ (now containing formula/substitution pairs) is written $\Gamma\theta$ or $\Delta\theta$.

The rules for this calculus are in Fig. 2; we use a double-headed arrow ($\longrightarrow\!\!\!\!\!\rightarrow$) to distinguish it from the ground forward calculus. The definition of additive contraction needs to be lifted to free subformulas also.

Definition 7 (lifted additive contraction). *The* lifted additive contraction judgment, *written $\Delta_1/w_1 + \Delta_2/w_2 = \langle \Delta \,;\, \xi \rangle$, takes as input the zones Δ_1 and Δ_2, together with their weak flags w_1 and w_2, and produces a contracted zone Δ and its corresponding substitution ξ. The rules for this judgment are as follows.*

$$\frac{}{\cdot/0 + \cdot/0 \rightsquigarrow \langle \cdot \,;\, \epsilon \rangle} \qquad \frac{}{\cdot/1 + \Delta/0 \rightsquigarrow \langle \Delta \,;\, \epsilon \rangle} \qquad \frac{}{\Delta/0 + \cdot/1 \rightsquigarrow \langle \Delta \,;\, \epsilon \rangle} \qquad \frac{}{\Delta/1 + \Delta'/1 \rightsquigarrow \langle \Delta, \Delta' \,;\, \epsilon \rangle}$$

$$\frac{\theta = \mathrm{mgu}(\sigma, \sigma') \quad \Delta\theta/w + \Delta'\theta/w' \rightsquigarrow \langle \Delta'' \,;\, \xi \rangle}{\Delta, A \cdot \sigma/w + \Delta', A \cdot \sigma'/w' \rightsquigarrow \langle \Delta'', A \cdot \sigma\theta\xi \,;\, \theta\xi \rangle}$$

Lemma 8 (lifted additive contraction).

1. *If $\Delta/0 + \Delta'/0 \rightsquigarrow \langle \Delta'' \,;\, \xi \rangle$, then $\Delta\xi = \Delta'\xi = \Delta''$.*
2. *If $\Delta/0 + \Delta'/1 \rightsquigarrow \langle \Delta'' \,;\, \xi \rangle$, then $\Delta'\xi \subseteq \Delta\xi = \Delta''$.*
3. *If $\Delta/1 + \Delta'/0 \rightsquigarrow \langle \Delta'' \,;\, \xi \rangle$, then $\Delta\xi \subseteq \Delta'\xi = \Delta''$.*
4. *If $\Delta/1 + \Delta'/1 \rightsquigarrow \langle \Delta'' \,;\, \xi \rangle$, then $\Delta\xi \subseteq \Delta''$ and $\Delta'\xi \subseteq \Delta''$.* □

judgmental rules

$$\frac{\theta = \mathrm{mgu}(P[\rho], P')}{\cdot\,;\, P\cdot\rho\theta \longrightarrow\!\!\!\!\rightarrow^{0} P'\cdot\theta}\ \text{init} \qquad \frac{\Gamma\,;\,\Delta, A\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}{\Gamma, A\cdot\sigma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}\ \text{copy}$$

$$\frac{\Gamma, A\cdot\sigma, A\cdot\sigma'\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}{\Gamma\theta, A\cdot\sigma\theta\,;\,\Delta\theta \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi\theta}\ \text{contr} \qquad \frac{\Gamma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}{\Gamma\rho\,;\,\Delta\rho \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi\rho}\ \text{ren}$$

multiplicative connectives

$$\frac{\Gamma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{w} A\cdot\sigma \quad \Gamma'\,;\,\Delta' \longrightarrow\!\!\!\!\rightarrow^{w'} B\cdot\sigma'}{\Gamma\theta, \Gamma'\theta\,;\,\Delta\theta, \Delta'\theta \longrightarrow\!\!\!\!\rightarrow^{w\vee w'} (A\otimes B)\cdot\sigma\theta}\ \otimes R$$

$$\frac{\Gamma\,;\,\Delta, A\cdot\sigma, B\cdot\sigma' \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}{\Gamma\theta\,;\,\Delta\theta, (A\otimes B)\cdot\sigma\theta \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi\theta}\ \otimes L \qquad \frac{\Gamma\,;\,\Delta, A_i\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{1} C\cdot\xi \quad (\forall\rho.\, A_j\cdot\sigma\rho \notin \Delta)}{\Gamma\,;\,\Delta, (A_1\otimes A_2)\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{1} C\cdot\xi}\ \otimes L_i$$

$$(i, j) \in \{(1,2),(2,1)\}$$

$$\frac{}{\cdot\,;\,\cdot \longrightarrow\!\!\!\!\rightarrow^{0} \mathbf{1}\cdot\epsilon}\ \mathbf{1}R \qquad \frac{\Gamma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{0} C\cdot\xi}{\Gamma\,;\,\Delta, \mathbf{1}\cdot\epsilon \longrightarrow\!\!\!\!\rightarrow^{0} C\cdot\xi}\ \mathbf{1}L$$

$$\frac{\Gamma\,;\,\Delta, A\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{w} B\cdot\sigma'}{\Gamma\theta\,;\,\Delta\theta \longrightarrow\!\!\!\!\rightarrow^{w} (A\multimap B)\cdot\sigma\theta}\ \multimap R \qquad \frac{\Gamma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{1} B\cdot\sigma \quad (\forall\rho.\, A\cdot\sigma\rho \notin \Delta)}{\Gamma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{1} (A\multimap B)\cdot\sigma}\ \multimap R'$$

$$\frac{\begin{array}{l}\Gamma\,;\,\Delta, B\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi \\ \Gamma'\,;\,\Delta' \longrightarrow\!\!\!\!\rightarrow^{w'} A\cdot\sigma' \qquad (w = 0 \vee \forall\rho.\, B\cdot\sigma\theta\rho \notin \Delta')\end{array}}{\Gamma\theta, \Gamma'\theta\,;\,\Delta\theta, \Delta'\theta, (A\multimap B)\cdot\sigma\theta \longrightarrow\!\!\!\!\rightarrow^{w\vee w'} C\cdot\xi\theta}\ \multimap L$$

additive connectives

$$\frac{\begin{array}{l}\Gamma\,;\,\Delta_1 \longrightarrow\!\!\!\!\rightarrow^{w_1} A\cdot\sigma \\ \Gamma\,;\,\Delta_2 \longrightarrow\!\!\!\!\rightarrow^{w_2} B\cdot\sigma' \qquad (\Delta_1\theta/w_1 + \Delta_2\theta/w_2 \rightsquigarrow \langle\Delta\,;\,\xi\rangle)\end{array}}{\Gamma\xi, \Gamma'\xi\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{w_1\wedge w_2} (A\,\&\,B)\cdot\sigma\xi}\ \&R \qquad \frac{}{\cdot\,;\,\cdot \longrightarrow\!\!\!\!\rightarrow^{1} \top\cdot\epsilon}\ \top R$$

$$\frac{\Gamma\,;\,\Delta, A\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}{\Gamma\,;\,\Delta, (A_1\,\&\,A_2)\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}\ \&L_i \qquad i \in \{1,2\}$$

exponentials

$$\frac{\Gamma\,;\,\cdot \longrightarrow\!\!\!\!\rightarrow^{w} A\cdot\sigma}{\Gamma\,;\,\cdot \longrightarrow\!\!\!\!\rightarrow^{0} !A\cdot\sigma}\ !R \qquad \frac{\Gamma, A\cdot\sigma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}{\Gamma\,;\,\Delta, !A\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}\ !L \qquad \frac{\Gamma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{0} C\cdot\xi \quad (\forall\rho.\, A\cdot\rho \notin \Gamma)}{\Gamma\,;\,\Delta, !A\cdot\epsilon \longrightarrow\!\!\!\!\rightarrow^{0} C\cdot\xi}\ !L'$$

quantifiers

$$\frac{\Gamma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{w} [\mathtt{a}/x]A\cdot(\sigma, \mathtt{b}/\mathtt{a})}{\Gamma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{w} \forall x.\, A\cdot\sigma}\ \forall R^{\mathtt{b}} \qquad \frac{\Gamma\,;\,\Delta, A\cdot(\sigma, \mathtt{t}/x) \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}{\Gamma\,;\,\Delta, \forall x.\, A\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}\ \forall L$$

$$\frac{\Gamma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{w} A\cdot(\sigma, \mathtt{t}/x)}{\Gamma\,;\,\Delta \longrightarrow\!\!\!\!\rightarrow^{w} \exists x.\, A\cdot\sigma}\ \exists R \qquad \frac{\Gamma\,;\,\Delta, [\mathtt{a}/x]A\cdot(\sigma, \mathtt{b}/\mathtt{a}) \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}{\Gamma\,;\,\Delta, \exists x.\, A\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}\ \exists L^{\mathtt{b}}$$

Note: $\theta = \mathrm{mgu}(\sigma, \sigma')$; ρ is a (fresh) renaming substitution; and premisses are variable-disjoint.

Fig. 2. Forward sequent calculus with free subformulas.

Definition 9. *A substitution σ is a* grounding substitution *if for every term-variable $x \in \mathrm{dom}(\sigma)$, the term $x[\sigma]$ contains no term variables.*

Theorem 10 (soundness). *If $\Gamma\,;\Delta \longrightarrow\!\!\!\!\rightarrow^{w} C \cdot \xi$ and λ is a grounding substitution for Γ, Δ and $C[\xi]$, then $\Gamma[\lambda]\,;\Delta[\lambda] \longrightarrow^{w} C[\xi\lambda]$.*

Proof (sketch). Straightforward induction on the structure of $\mathcal{F} :: \Gamma\,;\Delta \longrightarrow\!\!\!\!\rightarrow^{w} C \cdot \xi$, using lem. 8 and noting that any grounding unifier must be less general than the mgu. □

Theorem 11 (completeness).
Suppose $A_1[\sigma_1], A_2[\sigma_2], \ldots\,; B_1[\tau_1], B_2[\tau_2] \ldots \longrightarrow^{w} C[\xi]$ where the A_i, B_j and C are free subformulas of the goal. Then there exist substitutions $\sigma'_1, \sigma'_2, \ldots, \tau'_1, \tau'_2, \ldots, \xi'$ and λ such that:

1. *$A_1 \cdot \sigma'_1, A_2 \cdot \sigma'_2, \ldots\,; B_1 \cdot \tau'_1, B_2 \cdot \tau'_2, \ldots \longrightarrow\!\!\!\!\rightarrow^{w} C \cdot \xi'$; and*
2. *$\sigma'_i\lambda = \sigma_i$; $\tau'_j\lambda = \tau_j$; and $\xi'\lambda = \xi$.*

Proof (sketch). Structural induction on the given ground derivation. □

5 Design of the Implementation

5.1 Representation of Sequents and Contraction

Linear hypotheses can occur more than once in the linear zone, so for each substitution we also store the *multiplicity* of that substitution; we write this as $A \cdot \sigma^n$ where n is the multiplicity of $A \cdot \sigma$. In the common case of a variable-free proposition, this not only makes the representation of sequents more efficient, but also greatly reduces the non-determinism involved in matching a hypothesis in a premiss. Contraction in the presence of multiplicities is not much different from before; the only change is that we can perform a number of contractions together as a unit.

$$\frac{k = \min(m,n) \quad \theta = \mathrm{mgu}(\sigma,\tau) \quad \Delta_1\theta, A \cdot \sigma\theta^{m-k}/w_1 + \Delta_2\theta, A \cdot \tau\theta^{n-k}/w_2 \rightsquigarrow \langle \Delta\,;\xi\rangle}{\Delta_1, A \cdot \sigma^m/w_1 + \Delta_2, A \cdot \tau^n/w_2 \rightsquigarrow \langle \Delta, A \cdot \sigma\theta\xi^k\,;\theta\xi\rangle}$$

(Note that $\Delta, A \cdot \sigma^0$ is understood as Δ.)

In the implementation we perform contractions eagerly, that is, after every rule application we calculate the possible contractions in the conclusion of the rule. This allows us to limit contractions to binary rules, and furthermore, consider only the contractions between propositions that originate in different premisses. This is complete because if two hypotheses were to be contractible in the same premiss, then we would already have generated the sequent corresponding to that contraction earlier.

The special case of contracting two weak zones, i.e., $\Delta_1/1 + \Delta_2/1$, can be greatly improved by *first* eagerly contracting propositions that have an invertible unifier. This is complete because a weak $\Delta, A \cdot \sigma, A \cdot \sigma\rho$ is subsumed by a weak $\Delta, A \cdot \sigma$.

5.2 Rule Generation

The subformula property gives us the core of the inverse method procedure. We start with all initial sequents of the form $\cdot\,;\, P \cdot \rho\theta \longrightarrow^0 P' \cdot \theta$, where P is a negative, and P' a positive free atomic subformula of the goal sequent, ρ renames them apart, and and $\theta = \mathrm{mgu}(P[\rho], P')$. Next, we name all free subformulas of the goal sequent with unique propositional labels. Then, we specialize all inference rules to these labels as principal formulas before starting the main search procedure.

5.3 Subsumption and Indexing

Our prover performes forward, but currently not backward subsumption. Subsumption has to account for linearity and the notion of weak sequent.

Definition 12 (free subsumption). *The* free subsumption relation $\preceq$ *between free forward sequents is the smallest relation satisfying:*

$$\left.\begin{array}{l} \left(\Gamma\,;\Delta \longrightarrow\!\!\!\!\rightarrow^0 C \cdot \xi\right) \preceq \left(\Gamma'\,;\Delta \longrightarrow\!\!\!\!\rightarrow^0 C \cdot \xi'\right) \\ \left(\Gamma\,;\Delta \longrightarrow\!\!\!\!\rightarrow^1 C \cdot \xi\right) \preceq \left(\Gamma'\,;\Delta' \longrightarrow\!\!\!\!\rightarrow^w C \cdot \xi'\right) \end{array}\right\} \quad \textit{for some } \theta \textit{ such that } \Gamma\theta \subseteq \Gamma',\ \Delta\theta \subseteq \Delta',\ \textit{and } \xi\theta = \xi'$$

The full subsumption check is far too expensive to perform always. Subsumption is usually implemented as a sequence of phases of increasing complexity; Tammet called them *hierarchical tests* [16]. These hierarchical tests are designed to fail as early as possible, as the overwhelming majority of subsumption queries are negative.

Definition 13 (hierarchical tests).
To check if $s = \Gamma\,;\Delta \longrightarrow\!\!\!\!\rightarrow^w C \cdot \xi$ subsumes $s' = \Gamma'\,;\Delta' \longrightarrow\!\!\!\!\rightarrow^{w'} C' \cdot \xi'$, the following tests are performed in order:

1. *if $w = 0$ and $w' = 1$ then FAIL;*
2. *if $\#\Delta > \#\Delta'$ or $\#\Gamma > \#\Gamma'$ then FAIL (where # count the number of elements);*
3. *respecting multiplicities, if a free subformula L occurs n times in Δ and m times in Δ' and $n > m$ then FAIL; similarly for Γ and Γ';*
4. *if there is no θ for which $C[\xi\theta] = C'[\xi']$, then FAIL;*
5. *if for some $A \cdot \sigma \in \Gamma$ there is no $A \cdot \sigma' \in \Gamma'$ for which $A[\sigma\theta] = A[\sigma']$ (for some θ), then FAIL; similarly for Δ and Δ';*
6. *otherwise attempt the full subsumption test $s \preceq s'$.*

Tammet gives examples of other possible tests in [16], particularly tests that consider the depth of terms and statistics such as the number of constants, but we have not so far considered them in the linear setting.

For the index we use a global forest of substitution trees [17]. Each inserted sequent is indexed into the substitution tree corresponding to the label of the principal literal, indexed by its corresponding substitution. The leaves of the substitution tree contain the sequents where the indexed formula was the principal formula. To check if a given sequent is subsumed, we look up every formula in the sequent in the index to obtain a collection of subsumption candidates, which are then tested for subsumption using the hierarchical tests above.

5.4 Focusing and Lazy Rule Application

Efficient indexing and subsumption algorithms, though important, are not as critical to the design of the prover as the use of derived big-step rules. The inference rules of Fig.2 take tiny steps and thereby produce too many sequents. In our implementation we use a version of focusing [18, 2] tailored for forward reasoning to construct derived inference rules with many premisses. The essential insight of focusing is that every proof can be converted to one that alternates between two phases–*active* and *focused*. Thinking in the backward direction, during the active phase we break down all connectives whose left or right rules are invertible. This phase has no essential non-determinism. This leads to a so-called *neutral sequent* where we have to choose a formula to focus on, which is then successively decomposed by chaining together non-invertible rules on this particular focus formula. It turns out that in the forward direction we only need to keep neutral sequents if we construct big-step forward rules by analyzing those *frontier propositions* which can occur in neutral sequents and which are also subformulas of the goal. Essentially we simulate a backward focusing phase followed by a backward active phase by inferences in the forward direction. This construction is detailed in [11] for the propositional fragment and can easily be extended to the first-order setting.

We implement a derived rule as a curried function from sequents (premisses) to the conclusion of the rule. Each application of a rule to a sequent first tests if the sequent can match the corresponding premiss of the rule; if the match is successful, then the application produces a new partially instantiated rule, or if there are no remaining premisses then it produces a new sequent. The order of arguments of this curried function fixes a particular ordering of the premisses of the rule; the search procedure is set up so that any ordering guarantee completeness.

We use a lazy variant of the OTTER loop [19] as the main loop of the search procedure. We maintain two global sets of derived sequents:

- the *active* set containing sequents to be considered as premisses of rules; and
- the *inactive* set (sometimes referred to as the *set of support*) that contains all facts that have not yet been transferred to the active set.

This inner loop of the search procedure repeats the following lazy activation step until either the goal sequent is subsumed (in which case the search is successful), or no further rules are applicable to the sequents in the active set and the inactive set is exhausted (in which case the search saturates).

Definition 14 (lazy activation). *To activate the sequent s, i.e., to transfer it from the inactive to the active set, the following steps are performed:*

1. *After renaming, s is inserted into the active set.*
2. *All available rules are applied to s. If these applications produce new rules, R, then the following two steps are performed in a loop until there are no additions to R.*
   (a) *For every sequent s' in the active set, every rule in R is applied to s', and*
   (b) *any new rules generated are added to R.*
3. *The collection of rules R is added to the set of rules.*
4. *All sequents generated during the above applications are tested for subsumption, and the un-subsumed sequents and all their associated contractions are added to the inactive set.*

A sequent is added to the inactive set if it is not globally subsumed by some other sequent derived earlier. In fact, if it is subsumed, then none of its contracts need to be computed. We use the following heuristic for the order of insertion of the contracts of a given sequent: if s is the result of a sequence of contractions from s', then s is considered for insertion in the inactive set before s'.

The initial inactive set and rules are produced uniformly by focusing on the frontier literals of the goal sequent. The collection of (partially applied) rules grows at run-time; this is different from usual implementations of the OTTER loop where the rules are fixed before-hand. On the other hand, rule application is much simpler when each rule is treated as a single-premiss rule (producing sequents or other rules, possibly nothing). Furthermore, the same rule is never applied more than once to any sequent, because a previously derived rule is applied only to newly activated sequents (which were not in the active set before). Thus, the lazy activation strategy implicitly memoizes earlier matches.

5.5 Globalization

The final unrestricted zone Γ_g is shared in all branches in a proof of $\Gamma_g\,;\Delta_g \Longrightarrow C_g$. One thus thinks of Γ_g as part of the ambient state of the prover, instead of representing it explicitly as part of the current goal. Hence, there is never any need to explicitly record Γ_g or portions of it in the sequents themselves. This gives us the following global and local versions of the copy rule:

$$\frac{\Gamma\,;\Delta, A\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi \quad (\exists\rho.\, A\cdot\sigma\rho \in \Gamma_g)}{\Gamma\,;\Delta \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}\ \text{delete} \qquad \frac{\Gamma\,;\Delta, A\cdot\sigma \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi \quad (\forall\rho.\, A\cdot\sigma\rho \notin \Gamma_g)}{\Gamma, A\cdot\sigma\,;\Delta \longrightarrow\!\!\!\!\rightarrow^{w} C\cdot\xi}\ \text{copy}$$

6 Some Experimental Results

For our experiments we compared a few internal versions of the prover and two provers in the Gandalf family. For comparison purposes, we implemented a purely propositional version of our prover, which performs some additional optimizations that are not possible in the first-order case. The main differences are: contraction in the propositional case always produces exactly one sequent, as opposed to (potentially) exponentially many sequents in the first-order case; furthermore, subsumption is simply a matter of comparing the multiplicities, which is linear instead of quadratic. These properties greatly simplifies rule generation and application.

The internal versions of the prover are named **L** followed by a selection of **P** (propositional), **F** (big-step rules using focusing) and **G** (globalization) as suffixes. The default prover is named **L** (first-order, small-step rules, no globalization); **LPF**, for example, is the purely propositional prover with focused big-step rules, but no globalization. In total there are six internal versions. (We do not currently have a propositional prover that incorporates globalization.) These provers are written in Standard ML and are available from the first author's website[1]

[1] `http://www.cs.cmu.edu/~kaustuv/`

For our experiments the provers were compiled using MLTon version 20041119 with the default optimization flags. All time measurements are wall-clock times measured on an unloaded computer with a 2.80GHz Pentium 4 processor with a 512KB L1 cache and 1GB of main memory. Time measurements of less than 0.01 seconds should be taken as unreliable.

6.1 Purely Propositional Problems

For external provers we are aware of only Tammet's Gandalf "nonclassical" distribution (version 0.2), compiled using a packaged version of the Hobbit Scheme compiler. This prover is limited to the propositional fragment of classical linear logic, but comes in two flavors: resolution (**Gr**) and tableaux (**Gt**). Neither version incorporates focusing or globalization, and we did not attempt to bound the search for either prover. Our examples are, therefore, restricted to the propositional fragment. They include: simple theorems in linear logic (`basic`), blocks-world problems for a fixed set of blocks (`bw-prop`), a change machine encoding (`coins`), several affine logic problems encoded in linear logic (`affine1` and `affine2`), and a few hard examples of quantified Boolean formulas compiled to linear logic (`qbf1`, `qbf2` and `qbf3`, in order of increasing complexity) that implement the algorithm of [20].

	Gr	**Gt**	**LP**	**LPF**	**L**	**LF**
`basic`	0.06 s	0.08 s	0.024 s	0.018 s	0.058 s	0.037 s
`bw-prop`	×	×	×	0.001 s	×	0.007 s
`coins`	0.63 s	×	3.196 s	0.001 s	8.452 s	0.001 s
`affine1`	×[2]	0.01 s	0.003 s	0.001 s	1.645 s	3.934 s
`affine2`	×[2]	×	≈ 12 m	1.205 s	≈ 34 m	4.992 s
`qbf1`	2.40 s	×	0.013 s	0.001 s	0.038 s	0.002 s
`qbf2`	×	×	0.037 s	0.001 s	0.512 s	0.060 s
`qbf3`	×	×	0.147 s	0.003 s	2.121 s	0.820 s

× means no proof found within time limit

It is evident that focusing greatly speeds up both the propositional and first-order cases. For the propositional case, the speedup from the focusing prover to the non-focusing one is between 1.33 (`basic`) and 597.5 (`affine2`); for the first-order case, the speedups range from 1.57 (`basic`) to 408.7 (`affine2`). Except for `bw-prop`, these examples were all within the realm of possibility for the small-step provers, though some of them like `affine2` severely strain the provers. In the `affine1` case the focusing prover **LF** appears to take longer than the small-step prover **L**; this is because this example contains an unprovable proposition for which the inverse method procedure fails to saturate. The test is run for 1000 iterations of the lazy OTTER loop. The focusing prover gets much further than the small-step prover in 1000 iterations, and the delay is due entirely to the fact that the sequents it works with, after even a few dozen iterations, are far more complex than the small-step prover generates in 1000 iterations.

[2] **Gr** appears to saturate incorrectly in these cases (fails to prove a true proposition), so we have left out the running time.

Comparing to Gandalf, the small-step prover **L** is generally competitive with **Gr**: it is slower on some examples (`coins`), but succeeds on a wider range of problems. **Gt** was uniformly the slowest of the lot, taking a long time on even simple problems.

6.2 First-Order Problems

Our first-order problems include the following: a first-order blocks world planning example (`bw-fo`), Dijkstra's urn game (`urn`), simple intuitionistic first-order propositions encoded as linear propositions, using either Girard's translation (`int-gir`), or a focusing-aware translation (`int-foc`) outlined in [11].

	L	**LG**	**LF**	**LFG**
`bw-fo`	×	×	0.460 s	0.036 s
`urn`	×	×	0.413 s	0.261 s
`int-gir`	×	×	1.414 s	1.410 s
`int-foc`	≈11m	≈10 m	0.051 s	0.058 s

× means no proof found within time limit (20 minutes)

Again, it is fairly obvious that focusing is the dramatic winner, making some problems tractable, and being several orders of magnitude faster for the rest. Adding globalization also seems to have a significant effect here for the examples that are not constructed in an ad-hoc fashion (`bw-fo` and `urn`).

7 Conclusion

We have presented a theorem prover for first-order intuitionistic linear logic based on the inverse method which is already practical for a range of examples and significantly improves on prior, more restricted provers. The design is based on general principles that apply to both classical linear logic (which is simpler because it admits a one-sided sequent formulation with more symmetries) and affine logic (via weak sequents). Both of these can also be treated by uniform translations to intuitionistic linear logic [14], as can (ordinary) intuitionistic logic [11]. A generalization from a first-order logic to a type theory such as CLF [21, 22] would seem to require mostly a proper treatment of linear higher-order unification constraints, but otherwise be relatively straightforward.

Our prover also leaves room for further high-level and low-level optimizations. In particular, we plan to investigate how to limit the multiplicities of linear hypotheses, either a priori or as a complete heuristic. Some manual experiments seem to indicate that this could have a significant impact on a certain class of problems. We also plan to make our prover certifying; currently only the propositional version generates independently verifiable proof objects in a type theory.

References

1. Girard, J.Y.: Linear logic. Theoretical Computer Science **50** (1987) 1–102

2. Andreoli, J.M.: Logic programming with focusing proofs in linear logic. Journal of Logic and Computation **2** (1992) 297–347
3. Galmiche, D., Perrier, G.: Foundations of proof search strategies design in linear logic. In: Symposium on Logical Foundations of Computer Science, St. Petersburg, Russia, Springer-Verlag LNCS 813 (1994) 101–113
4. Galmiche, D.: Connection methods in linear logic and proof nets constructions. Theoretical Computer Science **232** (2000) 213–272
5. Harland, J., Pym, D.J.: Resource-distribution via boolean constraints. In McCune, W., ed.: Proceedings of CADE-14, Springer-Verlag LNAI 1249 (1997) 222–236
6. Cervesato, I., Hodas, J.S., Pfenning, F.: Efficient resource management for linear logic proof search. Theoretical Computer Science **232** (2000) 133–163
7. Pym, D.J., Harland, J.A.: The uniform proof-theoretic foundation of linear logic programming. Journal of Logic and Computation **4** (1994) 175–207
8. Hodas, J.S., Miller, D.: Logic programming in a fragment of intuitionistic linear logic. Information and Computation **110** (1994) 327–365
9. Mints, G.: Resolution calculus for the first order linear logic. Journal of Logic, Language and Information **2** (1993) 59–83
10. Tammet, T.: Proof strategies in linear logic. Journal of Automated Reasoning **12** (1994) 273–304
11. Chaudhuri, K.: Focusing the inverse method for linear logic. Technical Report CMU-CS-05-106, Carnegie Mellon University (2005) submitted for publication.
12. Méry, D.: Preuves et Sémantiques dans des Logiques de Ressources. PhD thesis, Université Henri Poincaré, Nancy, France (2004)
13. Donnelly, K., Gibson, T., Krishnaswami, N., Magill, S., Park, S.: The inverse method for the logic of bunched implications. In F.Baader, A.Voronkov, eds.: Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, Montevideo, Uruguay, Springer LNCS 3452 (2005) 466–480
14. Chang, B.Y.E., Chaudhuri, K., Pfenning, F.: A judgmental analysis of linear logic. Technical Report CMU-CS-03-131R, Carnegie Mellon University (2003)
15. Degtyarev, A., Voronkov, A.: The inverse method. In Robinson, J.A., Voronkov, A., eds.: Handbook of Automated Reasoning. MIT Press (2001) 179–272
16. Tammet, T.: Towards efficient subsumption. In: Proceedings of CADE-15. (1998) 427–441
17. Graf, P.: Term Indexing. Springer LNAI 1053 (1996)
18. Andreoli, J.M.: Focussing and proof construction. Annals of Pure and Applied Logic **107** (2001) 131–163
19. Tammet, T.: A resolution theorem prover for intuitionistic logic. In McRobbie, M., Slaney, J., eds.: Proceedings of CADE-13, New Brunswick, New Jersey, Springer-Verlag LNCS 1104 (1996) 2–16
20. Lincoln, P., Mitchell, J.C., Scedrov, A., Shankar, N.: Decision problems for propositional linear logic. Annals of Pure and Applied Logic **56** (1992) 239–311
21. Watkins, K., Cervesato, I., Pfenning, F., Walker, D.: A concurrent logical framework I: Judgments and properties. Technical Report CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University (2002) Revised May 2003.
22. Cervesato, I., Pfenning, F., Walker, D., Watkins, K.: A concurrent logical framework II: Examples and applications. Technical Report CMU-CS-02-102, Department of Computer Science, Carnegie Mellon University (2002) Revised May 2003.