Efficient Intuitionistic Theorem Proving with the Polarized Inverse Method

Sean McLaughlin and Frank Pfenning

Department of Computer Science
Carnegie Mellon University

Abstract. The inverse method is a generic proof search procedure applicable to non-classical logics satisfying cut elimination and the subformula property. In this paper we describe a general architecture and several high-level optimizations that enable its efficient implementation. Some of these rely on logic-specific properties, such as polarization and focusing, which have been shown to hold in a wide range of non-classical logics. Others, such as rule subsumption and recursive backward subsumption apply in general. We empirically evaluate our techniques on first-order intuitionistic logic with our implementation *Imogen* and demonstrate a substantial improvement over all other existing intuitionistic theorem provers on problems from the ILTP problem library.

1 Introduction

The *inverse method* [11,6] uses forward saturation, generalizing resolution to non-classical logics satisfying the subformula property and cut elimination. *Focusing* [1,10] reduces the search space in a sequent calculus by restricting the application of inference rules based on the *polarities* [9] of the connectives and atomic formulas. In this paper we describe a framework for reasoning in such logics, and exhibit a concrete implementation of a theorem prover for intuitionistic predicate logic, called *Imogen*, that is by some measure the most effective first order intuitionistic theorem prover: on the ILTP library of intuitionistic challenge problems [17], a collection similar to the well known TPTP [18] library, Imogen solves over 100 more problems than its closest competitor ileanCoP [13].[1]

This work continues a line of research on building efficient theorem provers for non-classical logics using the inverse method, following Tammet [19] (for intuitionistic and linear logic), Linprover [4,3] (for intuitionistic linear logic), and the early propositional version of Imogen [12]. We briefly highlight the principal contributions of this paper. On the logical side, *explicit polarization* of a given input formula determines basic characteristics of the search space, refining ideas from Chaudhuri et al. [5]. Our architecture provides a clean interface between the specification of basic logical inference (the *front end*) on one side and saturating proof search (the *back end*) on the other. This separation allows for generic logic-independent optimizations. Our back end maintains dynamic collections of sequents and inference rules, both of which are subject to redundancy elimination and heuristic selection. Novel redundancy elimination techniques include

[1] Imogen is available at `http://www.cs.cmu.edu/~seanmcl/research/imogen/`

inference rule subsumption and *recursive backward subsumption*. We conclude with an extensive analysis of the effects of individual optimizations.

2 Backward Polarized Sequent Calculus

The inverse method upon which our methods are based is a forward saturation strategy. However, we limit the search space of the inverse method by searching only for *focused proofs* [1] that could be found in the backwards direction. To make the relation clear, in this section we give the rules for the (ground) backward polarized sequent calculus. This ground calculus will then be lifted to a free variable calculus and proof search will proceed in the forward direction, from the initial sequents to the goal, following the inverse method recipe [6]. One novelty of our approach is the use of explicit polarization in formulas that syntactically mark the polarity of the atoms and connectives. We first describe polarized formulas, and then show the backward sequent calculus.

2.1 Polarized Formulas

A connective is *positive* if its left rule in the sequent calculus is invertible and *negative* if its right rule is invertible. As shown below, our proof search fundamentally depends on the polarity of connectives. In intuitionistic logic, the status of conjunction and truth is ambiguous in the sense that they are both positive and negative, while their status in linear logic is uniquely determined. We therefore syntactically distinguish positive and negative formulas with so-called *shift* operators [9] explicitly coercing between them. Even though we use the notation of linear logic, the behavior of the connectives is not linear.

In the following, the meta-variable P ranges over atomic formulas which have the form $p(t_1, \ldots, t_n)$ for predicates p. Note also that both in formulas and sequents, the signs are not actual syntax but mnemonic guides to the reader.

$$\text{Positive formulas } A^+ ::= P^+ \mid A^+ \otimes A^+ \mid 1 \mid A^+ \oplus A^+ \mid 0 \mid \exists x.\, A^+ \mid \downarrow A^-$$
$$\text{Negative formulas } A^- ::= P^- \mid A^- \,\&\, A^- \mid \top \mid A^+ \multimap A^- \mid \forall x.\, A^- \mid \uparrow A^+$$

The translation A^- of an (unpolarized) formula F in intuitionistic logic is nondeterministic, subject only to the constraints that the *erasure* defined below coincides with the original formula: $|A^-| = F$, and all predicates are assigned a consistent polarity.

$$\begin{array}{lll}
|A^+ \oplus B^+| = |A^+| \vee |B^+| & |0| = \bot & |1| = \top \\
|A^+ \otimes B^+| = |A^+| \wedge |B^+| & |\downarrow A^-| = |A^-| & |P^+| = P \\
|A^- \& B^-| = |A^-| \wedge |B^-| & |\top| = \top & |P^-| = P \\
|A^+ \multimap B^-| = |A^+| \supset |B^-| & |\uparrow A^+| = |A^+| & \\
|\forall x.\, A^-| = \forall x.\, |A^-| & |\exists x.\, A^+| = \exists x.\, |A^+| &
\end{array}$$

Fig. 1. Erasure of polarized formulas

For example, the formula $((p \vee r) \wedge (q \supset r)) \supset (p \supset q) \supset r$ can be interpreted as any of the following polarized formulas (among others):

$$((\downarrow p^- \oplus \downarrow r^-) \otimes \downarrow(\downarrow q^- \multimap r^-)) \multimap (\downarrow(\downarrow p^- \multimap q^-) \multimap r^-)$$
$$\downarrow\uparrow((\downarrow p^- \oplus \downarrow r^-) \otimes \downarrow(\downarrow q^- \multimap r^-)) \multimap (\downarrow\uparrow\downarrow(\downarrow p^- \multimap q^-) \multimap r^-)$$
$$\downarrow(\uparrow(p^+ \oplus r^+) \,\&\, (q^+ \multimap \uparrow r^+)) \multimap (\downarrow(p^+ \multimap \uparrow q^+) \multimap \uparrow r^+)$$

Shift operators have highest binding precedence in our presentation of the examples. As we will see from the inference rules given below, the choice of translation determines the search behavior on the resulting polarized formula. Different choices can lead to search spaces with radically different structure [5,12].

2.2 Backward Polarized Sequent Calculus

The backward calculus is a refinement of Gentzen's LJ that eliminates don't-care nondeterministic choices, and manages don't-know nondeterminism by chaining such inferences in sequence. Andreoli was the first to define this *focusing* strategy and prove it complete [1], and similar proofs for other logics soon followed [8,10,20]. Therefore, polarization can be applied to optimize search in a wide variety of logics.

The polarized calculus is defined via four mutually recursive judgments. In the judgments, we separate the antecedents into positive and negative zones. We write Γ for an unordered collection of negative formulas or positive atoms. Dually, C stands for a positive formula or a negative atom.

The first two judgments concern formulas with invertible rules on the right and left. Together, the two judgments form the *inversion phase* of focusing. In the rules RA-$\forall$ and LA-$\exists$, a is a new parameter.[2] The context Δ^+ is consists entirely of positive formulas and is ordered so that inference rules can only be applied to the rightmost formula, eliminating don't-care nondeterminism.

$\boxed{\Gamma; \Delta^+ \Longrightarrow A^-; \cdot}$ (Right Inversion)

$$\frac{\Gamma; \Delta^+ \Longrightarrow \cdot; P^-}{\Gamma; \Delta^+ \Longrightarrow P^-; \cdot}\ \text{(RA-Atom)} \qquad \frac{\Gamma; \Delta^+ \Longrightarrow A_1^-; \cdot \quad \Gamma; \Delta^+ \Longrightarrow A_2^-; \cdot}{\Gamma; \Delta^+ \Longrightarrow A_1^- \& A_2^-; \cdot}\ \text{(RA-\&)}$$

$$\frac{\Gamma; \Delta^+, A_1^+ \Longrightarrow A_2^-; \cdot}{\Gamma; \Delta^+ \Longrightarrow A_1^+ \multimap A_2^-; \cdot}\ \text{(RA-}\multimap\text{)} \qquad \frac{}{\Gamma; \Delta^+ \Longrightarrow \top; \cdot}\ \text{(RA-}\top\text{)}$$

$$\frac{\Gamma; \Delta^+ \Longrightarrow A(a)^-; \cdot}{\Gamma; \Delta^+ \Longrightarrow \forall x.\, A(x)^-; \cdot}\ \text{(RA-}\forall\text{)}^a \qquad \frac{\Gamma; \Delta^+ \Longrightarrow \cdot; A^+}{\Gamma; \Delta^+ \Longrightarrow \uparrow A^+; \cdot}\ \text{(RA-}\uparrow\text{)}$$

$\boxed{\Gamma; \Delta^+ \Longrightarrow \cdot; C}$ (Left Inversion)

[2] In our calculus, parameters differ syntactically from term variables, and are thus slightly different than the *eigenvariables* found in other presentations of the inverse method. A formalization of parameters and their effect on unification can be found in [3].

$$\frac{\Gamma, P^+; \Delta^+ \Longrightarrow \cdot; C}{\Gamma; \Delta^+, P^+ \Longrightarrow \cdot; C}\ \text{(LA-Atom)} \qquad \frac{\Gamma; \Delta^+, A_1^+, A_2^+ \Longrightarrow \cdot; C}{\Gamma; \Delta^+, A_1^+ \otimes A_2^+ \Longrightarrow \cdot; C}\ \text{(LA-}\otimes\text{)}$$

$$\frac{\Gamma; \Delta^+ \Longrightarrow \cdot; C}{\Gamma; \Delta^+, 1 \Longrightarrow \cdot; C}\ \text{(LA-1)} \qquad \frac{\Gamma; \Delta^+, A_1^+ \Longrightarrow \cdot; C \quad \Gamma; \Delta^+, A_2^+ \Longrightarrow \cdot; C}{\Gamma; \Delta^+, A_1^+ \oplus A_2^+ \Longrightarrow \cdot; C}\ \text{(LA-}\oplus\text{)}$$

$$\frac{}{\Gamma; \Delta^+, 0 \Longrightarrow \cdot; C}\ \text{(LA-0)} \qquad \frac{\Gamma; \Delta^+, A(a)^+ \Longrightarrow \cdot; C}{\Gamma; \Delta^+, \exists x.\, A(x)^+ \Longrightarrow \cdot; C}\ \text{(LA-}\exists\text{)}^a$$

$$\frac{\Gamma, A^-; \Delta^+ \Longrightarrow \cdot; C}{\Gamma; \Delta^+, \downarrow A^- \Longrightarrow \cdot; C}\ \text{(LA-}\downarrow\text{)}$$

The next two judgments are concerned with non-invertible rules. These two judgments make up the *focusing phase*.

$\boxed{\Gamma \gg A^+}$ (Right Focusing)

$$\frac{}{\Gamma, P^+ \gg P^+}\ \text{(RS-Atom)} \quad \frac{\Gamma \gg A_1^+ \quad \Gamma \gg A_2^+}{\Gamma \gg A_1^+ \otimes A_2^+}\ \text{(RS-}\otimes\text{)} \quad \frac{}{\Gamma \gg 1}\ \text{(RS-1)}$$

$$\frac{\Gamma \gg A_1^+}{\Gamma \gg A_1^+ \oplus A_2^+}\ \text{(RS-}\oplus_1\text{)} \quad \frac{\Gamma \gg A_2^+}{\Gamma \gg A_1^+ \oplus A_2^+}\ \text{(RS-}\oplus_2\text{)} \qquad \text{No rule for 0}$$

$$\frac{\Gamma \gg A(t)^+}{\Gamma \gg \exists x.\, A^+}\ \text{(RS-}\exists\text{)} \qquad \frac{\Gamma; \cdot \Longrightarrow A^-; \cdot}{\Gamma \gg \downarrow A^-}\ \text{(RS-}\downarrow\text{)}$$

$\boxed{\Gamma; A^- \ll C}$ (Left Focusing)

$$\frac{}{\Gamma; P^- \ll P^-}\ \text{(LS-Atom)} \quad \frac{\Gamma; A_1^- \ll C}{\Gamma; A_1^- \& A_2^- \ll C}\ \text{(LS-}\&_1\text{)} \quad \frac{\Gamma; A_2^- \ll C}{\Gamma; A_1^- \& A_2^- \ll C}\ \text{(LS-}\&_2\text{)}$$

$$\frac{\Gamma \gg A_1^+ \quad \Gamma; A_2^- \ll C}{\Gamma; A_1^+ \multimap A_2^- \ll C}\ \text{(LS-}\multimap\text{)} \qquad \text{No rule for } \top \qquad \frac{\Gamma; A(t)^- \ll C}{\Gamma; \forall x.\, A^- \ll C}\ \text{(LS-}\forall\text{)}$$

$$\frac{\Gamma; A^+ \Longrightarrow \cdot; C}{\Gamma; \uparrow A^+ \ll C}\ \text{(LS-}\uparrow\text{)}$$

Backward search for a proof of A^- would start with inversion from $\cdot; \cdot \Longrightarrow A^-; \cdot$ and then alternate between focusing and inversion phases. Call a focusing phase followed by an inversion phase a *block*. The boundary between blocks is of particular importance. The sequents at the boundary have the form $\Gamma; \cdot \Longrightarrow \cdot; C$, which we call *stable sequents*. There are two rules that control the phase changes and make the choices at block boundaries.

$$\frac{\Gamma \gg A^+}{\Gamma; \cdot \Longrightarrow \cdot; A^+}\ \text{(FocusR)} \qquad \frac{\Gamma, A^-; A^- \ll C}{\Gamma, A^-; \cdot \Longrightarrow \cdot; C}\ \text{(FocusL)}$$

An example of a backward derivation highlighting the block structure, is shown in Figure 2. a, b, and c are negative atoms. The elided sections are deterministic application of the above rules. Note that the nondeterminism occurs only at block boundaries and during focusing choices.

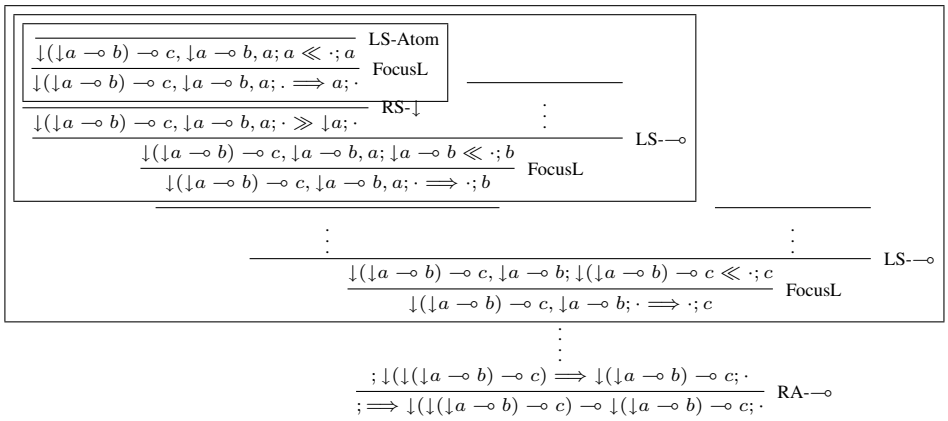


Fig. 2. Backward proof, with blocks

Theorem 1 *[10] If there exists a intuitionistic derivation of A, then for any polarization A^- of A, there exists a focused derivation of $\cdot;\cdot \Longrightarrow A^-;\cdot$.*

2.3 Synthetic Connectives and Derived Rules

We have already observed that backward proofs have the property that the proof is broken into blocks, with stable sequents at the boundary. The only rules applicable to stable sequents are the rules that select a formula on which to focus. It is the formulas occurring in stable sequents that form the primary objects of our further inquiry.

It helps to think of such formulas, abstracted over their free variables, as *synthetic connectives* [2]. Define the synthetic connectives of a formula A as all subformulas of A that could appear in stable sequents in a focused backward proof. In a change of perspective, we can consider each block of a proof as the application of a left or right rule for a synthetic connective. The rules operating on synthetic connectives are derived from the rules for its constituent formulas. We can thus consider a backward proof as a proof using only these synthetic (derived) rules. Each derived rule then corresponds to a block of the original proof.

Since we need only consider stable sequents and synthetic connectives, we can simplify notation, and ignore the (empty) positive left and negative right zones in the derived rules. Write $\Gamma;\cdot \Longrightarrow \cdot;C$ as $\Gamma \Longrightarrow C$. As a further simplification, we can give formulas a predicate label and abstract over its free variables. This labeling technique is described in detail in [6]. For the remainder, we assume this labeling has been carried out. Define an *atomic formula* as either a label or a predicate applied to a (possibly empty) list of terms. After labeling, our sequents consist entirely of atomic formulas.

Example 1 *In Figure 2, frame boxes surround the three blocks of the proof. The synthetic connectives are a, $\downarrow a \multimap b$ and $\downarrow(\downarrow a \multimap b) \multimap c$. There is a single derived rule for*

each synthetic connective (though this is not the case in general). We implicitly carry the principal formula of a left rule to all of its premises.

$$\frac{}{\Gamma, a \Longrightarrow a}\ \mathit{Syn}_1 \qquad \frac{\Gamma \Longrightarrow a}{\Gamma, \downarrow a \multimap b \Longrightarrow b}\ \mathit{Syn}_2 \qquad \frac{\Gamma, a \Longrightarrow b}{\Gamma, \downarrow(\downarrow a \multimap b) \multimap c \Longrightarrow c}\ \mathit{Syn}_3$$

These rules correspond to the blocks shown in Figure 2. Corresponding labeled rules for $L_1 = \downarrow a \multimap b$ and $L_2 = \downarrow(\downarrow a \multimap b) \multimap c$ are

$$\frac{}{\Gamma, a \Longrightarrow a}\ \mathit{Syn}_1 \qquad \frac{\Gamma \Longrightarrow a}{\Gamma, L_1 \Longrightarrow b}\ \mathit{Syn}_2 \qquad \frac{\Gamma, a \Longrightarrow b}{\Gamma, L_2 \Longrightarrow c}\ \mathit{Syn}_3$$

Then the blocks of the proof from Figure 2 can be compressed to the succinct

$$\begin{array}{l} \dfrac{}{L_1, L_2, a \Longrightarrow a}\ \mathit{Syn}_1 \\ \dfrac{}{L_1, L_2, a \Longrightarrow b}\ \mathit{Syn}_2 \\ \dfrac{}{L_1, L_2 \Longrightarrow c}\ \mathit{Syn}_3 \end{array}$$

3 The Polarized Inverse Method

In the previous section we developed the notion of a focused backwards proof; in this section we turn it into a forward proof search strategy.

3.1 Forward Rules

Recall the following rules from the backward (unfocused) intuitionistic sequent calculus:

$$\frac{}{\Gamma, a \Longrightarrow a}\ \text{Init} \qquad \frac{}{\Gamma, \bot \Longrightarrow A}\ \bot\text{-L} \qquad \frac{\Gamma \Longrightarrow A \quad \Gamma \Longrightarrow B}{\Gamma \Longrightarrow A \wedge B}\ \wedge\text{-R}$$

Interpreting these rules for forward search shows some difficulties. In the forward direction we want to guess neither the context Γ, nor the formula A in the $\bot$-L rule. We therefore allow the succedent to be empty, and ignore Γ in such initial sequents. The analogous forward sequents have the form

$$\frac{}{a \longrightarrow a}\ \text{Init} \qquad \frac{}{\bot \longrightarrow .}\ \bot\text{-L} \qquad \frac{\Gamma_1 \longrightarrow A \quad \Gamma_2 \longrightarrow B}{\Gamma_1 \cup \Gamma_2 \longrightarrow A \wedge B}\ \wedge\text{-R}$$

We maintain a connection with the backward calculus by saying that a forward sequent stands for all of its weakening and substitution instances. This new form of sequent requires a more complicated notion of *matching* (or *applying*) inference rules. We now define the operations necessary for proof search in the forward polarized sequent calculus *lifted* [6] to free variables. We assume the reader is familiar with the usual notions of substitutions and most general unifiers.

Definition 1 (Forward Sequents) *A* forward sequent *has the form $\Gamma \longrightarrow C$ where Γ is a set of atomic formulas and C is either the empty set or a set containing a single atomic formula. It is written $\Gamma \longrightarrow A$ in the case that $C = \{A\}$, or $\Gamma \longrightarrow \cdot$ in case $C = \emptyset$. The set Γ consists of the* antecedents*, and C is called the* succedent*. Define the functions* $\mathtt{succ}(\Gamma \longrightarrow C) = C$ *and* $\mathtt{ants}(\Gamma \longrightarrow C) = \Gamma$.

The variables of a sequent are implicitly universally quantified outside the sequent. This means every time a sequent is used, we have to rename its variables to be fresh. We will apply such renamings tacitly.

Definition 2 (Inference rules) *An* inference rule *with name id,*

$$\frac{H_1 \quad \ldots \quad H_n}{Q}\ id(\Pi)$$

has premises $H_1, \ldots, H_n$ and conclusion Q, all of which are sequents. Π is a set of parameters (the fixed parameters*) that are introduced during the inversion phase. Inference rules are schematic in their variables, which can range over formulas or terms.*

3.2 Matching

Given some known sequents, we wish to derive new sequents using the inference rules. The process of inferring new sequents from known sequents using inference rules is called *matching* (or *rule application*). In the backward direction of intuitionistic logic, matching is a simple matter, consisting of decomposing the principle connective and copying the context. The presence of fixed parameters, possibly empty succedents, and the union of contexts significantly complicate the operation in the forward direction. We therefore define matching in stages. (Recall [3] throughout that if a parameter a is in the domain of a substitution θ, then $a\theta$ is also a parameter.) First, a sequent can only match a premise of a rule if their respective succedents are unifiable.

Definition 3 (Succedent Matching) *A succeedent C* matches *a succeedent C'* with *substitution θ if either*

1. *$C = \{A\}$ and $C' = \{A'\}$ and $\theta = \mathtt{mgu}(A, A')$*
2. *$C = \cdot$ and $\theta = \emptyset$*
3. *$C' = \cdot$ and $\theta = \emptyset$*

When matching a premise of an inference rule, we need to match antecedents as well. Because a sequent stands for all of it weakening and substitution instances, not all specified antecedents in the premise of a rule actually need to be present in the sequent. There are therefore many ways the antecedents of a sequent can match the antecedents in a premise of a rule, depending on the antecedents that are *held* (that is, not matched). This consideration leads to the following:

Definition 4 (Antecedent Matching) *Sequent antecedents $\Gamma = \Gamma_1, \Gamma_2$* match *premise antecedents $\Gamma' = \Gamma'_1, \Gamma'_2$* with *$\theta$* holding *$\Gamma_2$ if $|\Gamma_1| = |\Gamma'_1|$ and θ is the pairwise mgu of Γ_1, Γ'_1.*

Definition 5 (Premise Matching) *Sequent* $\Gamma \longrightarrow C$ matches *premise* $\Gamma' \longrightarrow C'$ with θ *if* C *matches* C' *with* θ_1 *and* Γ *matches* Γ' *with* θ_2 *and* $\theta = \mathtt{mgu}(\theta_1, \theta_2)$.

Definition 6 (Rule Matching) *Sequents* $q_1, \ldots, q_n$ match *rule*

$$\frac{H_1 \quad \ldots \quad H_n}{Q}\ id(\Pi)$$

if for each $1 \leq i \leq n$, q_i *matches* H_i *with* θ_i *holding* Γ_i, $\theta = \mathtt{mgu}(\theta_1, \ldots, \theta_n)$, *and for each* $1 \leq i \leq n$, *all of the following hold:*

1. *For each element* $A \in \mathtt{ants}(q_i\theta) \setminus \mathtt{ants}(H_i\theta)$, $\mathtt{params}(A) \cap \Pi = \emptyset$
2. *For each pair of parameters* $a, b \in \Pi$, $a\theta \neq b\theta$
3. *For each variable* X *of* $H_1, \ldots, H_n$, $\mathtt{params}(X\theta) \cap \Pi = \emptyset$.
4. *For every* i *such that* $\mathtt{succ}(H_i) = \cdot$, *and every variable* X *of* $\mathtt{succ}(q_i)$, $\mathtt{params}(X\theta) \cap \Pi = \emptyset$.
5. *For every* i *such that* $\mathtt{succ}(H_i) = \cdot$, *either* $\mathtt{succ}(q_i\theta) = \cdot$ *or* $\mathtt{succ}(q_i\theta) = \mathtt{succ}(Q\theta)$.

The first four properties assure our equivalent of the eigenvariable condition: every parameter that is generated by the synthetic rule from a (derived) use of LA-$\exists$ *or RA-*$\forall$ *is new. The final property assures that the rule is consistent. If the succedent is unspecified, then it either remains unspecified in the new sequent, or is specified to exactly one sequent. If all of the above hold, then the* result *of the match is the sequent*

$$(\Gamma_1 \cup \ldots \cup \Gamma_n \cup \mathtt{ants}(Q) \longrightarrow \mathtt{succ}(Q))\theta$$

The definition of matching assures that the forward application simulates a backward rule application. Since we always combine unused premises in the same way, in the rest of the paper we omit the contexts Γ in forward inference rules.

Example 2 *If the synthetic connective is* $L_1 = \downarrow(((\exists y.\, \downarrow p(y)) \multimap \forall x.\, (p(x)\ \&\ q(x)))$ *on the right, then the backward and forward synthetic rules are*

$$\frac{\Gamma, p(a) \Longrightarrow p(b) \quad \Gamma, p(a) \Longrightarrow q(b)}{\Gamma \Longrightarrow L_1}\ \mathit{Syn}^{a,b}$$

$$\frac{p(a) \longrightarrow p(b) \quad p(a) \longrightarrow q(b)}{\longrightarrow L_1}\ \mathit{Syn}(\{a, b\})$$

3.3 Proof Search

Before we can turn our observations into a method for proof search, we need two more crucial definitions. First, the inverse method cannot in general prove a given sequent exactly, but sometimes only a stronger form of it. This is captured by the *subsumption* relation.

Definition 7 (Subsumption) *A sequent* $\Gamma_1 \longrightarrow C_1$ subsumes *a sequent* $\Gamma_2 \longrightarrow C_2$ *if there exists a substitution* θ *such that* $|\Gamma_1\theta| = |\Gamma_1|$ *(i.e.,* θ *does not contract* Γ_1*) and* $\Gamma_1\theta \subseteq \Gamma_2$ *and* $C_1\theta \subseteq C_2$. *Write* $Q \leq Q'$ *if* Q *subsumes* Q'.

Suppose $Q \leq Q'$ and we are trying to prove Q'. Since weakening is an admissible rule in the backward calculus, given a backward proof $\mathcal{D}$ of Q, we could modify $\mathcal{D}$ by weakening, yielding a proof of Q'.

The second definition comes from the following observation. It is not the case that $(p(X,Y), p(Y,X) \longrightarrow g) \leq (p(Z,Z) \longrightarrow g)$, even though $(p(Z,Z) \longrightarrow g)$ is identical to $(p(X,Y), p(Y,X) \longrightarrow g)$ under substitution $\{X \mapsto Z, Y \mapsto Z\}$. (Remember that we maintain the premises as a set.) Since a sequent stands for its substitution instances, we should be able to infer the latter sequent. This consideration motivates the definition of *contraction*:

Definition 8 (Contraction) *$\Gamma\theta, A_1\theta \longrightarrow C\theta$ is a* contraction instance *of a sequent $\Gamma, A_1, A_2 \longrightarrow C$ if θ is the most general unifier of A_1, A_2.*

Now we have the basic operations necessary to define forward search using the polarized inverse method. We begin with a negative polarized input formula A^-. We first decompose the problem into stable sequents by applying the backward rules, inverting the sequent $\cdot;\cdot \Longrightarrow A^-;\cdot$. The leaves of the backward inversion are stable sequents. Each stable sequent is solved independently. (This is why the bottom portion of Figure 2 is not contained in a block.) For each stable sequent, we determine the sequent's synthetic formulas, and generate the corresponding derived rules. We begin with a sequent database containing the *initial sequents*, those synthetic rules with no premises. We repeatedly match the synthetic rules to known sequents in the forward direction. The resulting matches, along with all of their contraction instances, are added to the database. We continue in this way until we either generate a sequent that subsumes the goal, or until the database is saturated, that is, any further inference would only add sequents subsumed by something already in the database. Due to the undecidability of the problem, if the goal is not provable, it is possible that the database will never saturate.

Theorem 2 (Completeness) *If there exists a (ground) backward focused derivation of a polarized formula A, then such a derivation can be constructed using the polarized inverse method.*

Proof Analogous to the corresponding proof in [3]. □

4 An Implementation Framework

We turn now to our implementation, called *Imogen*. The implementation is designed as two distinct modules, referred to respectively as the *front end* and the *back end*. The front end deals with the specifics of a particular logic and focusing strategy. It takes a formula as input and returns the initial stable sequents, and for each sequent a complete set of synthetic inference rules and initial sequents. The back end maintains a database of known sequents, and applies the rules to the database using a *fair* strategy, generating new sequents. It stops when it finds a sequent that subsumes the goal, or when the database is saturated.

This design makes it possible to use the same back end for different logics. While Imogen now only supports two front ends, intuitionistic first order logic and an optimized front end for the propositional fragment, it would be straightforward to extend to other logics.

4.1 The Front End: Rule Generation and Matching

The front end has two distinct tasks. The first is to generate the initial rules and sequents given an input formula and a focusing strategy. This is achieved by, for each synthetic connective, evaluating the inference rules of Section 2 in the backward direction. Each block of a potential backward proof becomes an inference rule.

The second is to define the main functions outlined in the last section: subsumption, contraction, and rule matching. Subsumption can be an expensive operation, but is straightforward to implement. Contraction can be a problematic operation because if a sequent has antecedents with the same label or predicate symbol, there can be an exponential number of contraction instances. It is not uncommon for Imogen to generate tens of thousands of contraction instances of a single sequent.

To implement the function `match` of Definition 6, we use the technique of *partial rule application*. Instead of having a fixed rule set and matching all the hypotheses simultaneously, we have an expanding rule set, and match one premise at a time. The match of a rule with n premises yields a new residual rule with $n - 1$ premises.

Example 3 *Matching rule*

$$\frac{p(X,Y) \longrightarrow q(X,Y) \quad q(X,Y) \longrightarrow \cdot}{r(X,Y) \longrightarrow \cdot}$$

to sequent

$$q(Z,c), p(c,Z) \longrightarrow q(c,Z)$$

yields the new inference rule

$$\frac{q(c,Y) \longrightarrow \cdot}{q(Y,c), r(c,Y) \longrightarrow \cdot}$$

Matching the new rule against

$$q(c,d) \longrightarrow q(d,d)$$

yields the new sequent

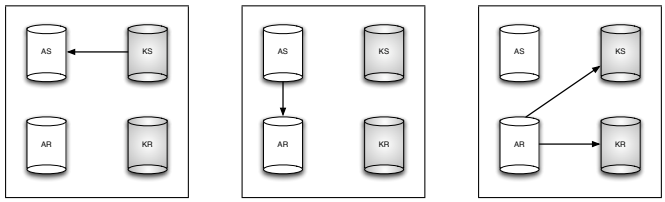
$$q(d,c), r(c,d) \longrightarrow q(d,d)$$

Similar to contraction, if both a rule and sequent have multiple instances of the same label or predicate, matching can produce an inordinate number of new rules or sequents.

4.2 The Back End: Rule Application and Subsumption

The back end takes the initial sequents and rules from the front end, along with the definitions of matching, subsumption and contraction. Then it uses a modified form of the *Otter loop* to search for proofs.

The Otter Loop. The "Otter loop" is a general strategy for automated reasoning using forward inference. In our version, there are two databases of sequents, called *kept* and *active*, and two databases of rules (because of our partial rule matching strategy) also so named. The *active sequents* (AS), consist of all the sequents that have already been matched to rules in *active rules* (AR). Symmetrically, the active rules are the rules that have been matched to the active sequents. The other databases, the *kept* rules (KR) and sequents (KS), have not yet been considered for matching. A step of the loop proceeds as follows, as shown in the figure below. Imogen chooses either a kept sequent or a kept rule according to some fair strategy. Suppose it chooses a sequent. Then we are in the situation in diagram (a). The sequent is added to the active sequents, and then is matched to all active rules (b). This matching will generate new sequents (when the matched rule has a single premise), and new rules (when the matched rule has multiple premises). The new rules and sequents are added to the respective kept databases (c). A symmetric process occurs when choosing a kept rule.



(a) (b) (c)

4.3 Subsumption

Redundancy elimination is an important part of an efficient implementation of the polarized inverse method. Imogen performs subsumption in a variety of ways. The first is *forward subsumption*: new sequents generated during the matching process that are subsumed by existing sequents are never added to the kept database. Another form of subsumption occurs when a new sequent subsumes an existing active or kept sequent. There are two forms of backward subsumption in Imogen. The first, simply called *backward subsumption* is where we delete the subsumed sequent from the database. In *recursive backward subsumption* we delete not only the subsumed sequent, but all of that sequent's descendents except those justifying the subsuming sequent. The idea being that Imogen, with the new, stronger sequent, will eventually recreate equal or stronger forms of the rules and sequents that were deleted. A final version of subsumption is called *rule subsumption*. Rule subsumption occurs when a new sequent subsumes the conclusion of an inference rule. In this case, whatever the content of the premises, the resulting conclusion would be forward subsumed, as matching only instantiates variables and adds to the antecedents. Thus, such a rule can be safely deleted.

Theorem 3 *If there exists a derivation of A, then there exists a derivation that respects forward, backward, and rule subsumption.*

Proof For forward and backward subsumption, the proof is analogous to the one given by Degtyarev and Voronkov [6]. Since each sequent that could be generated by a subsumed rule would itself be subsumed, their argument extends easily to our framework. □

For recursive backward subsumption, while the soundness is clear, it is not as easy to see that our strategy is still complete.

Theorem 4 *Recursive backward subsumption is nondeterministically complete, that is, if the database saturates without subsuming the goal, then the goal can not be derived.*

Proof For every recursively deleted sequent we either retain a stronger sequent, or we retain the possibility to recreate a stronger sequent. For the database to be saturated, we must have subsumed or recreated all the deleted sequents. □

This is enough to obtain the correctness of our prover: by soundness (and, in addition, through an independently verifiable natural deduction proof object) we have a proof when the goal is subsumed. If the databased is saturated without subsuming the goal, there cannot be a proof. We conjecture that recursive backward subsumption is also complete in the stronger sense that if there is a proof we could in principle always find it (since rule and sequent selection are fair), but we do not at present have a rigorous proof.

Besides fairness, the proofs of completeness under the various forms of redundancy elimination rely mainly on the following property of the (derived) rules used in the forward direction: If the premise of a rule is subsumed, either the conclusion is already subsumed or we can reapply the rule and obtain a new conclusion which subsumes the old one.

4.4 Other Features

The back end implements a few other notable features. Since in a backward proof, the antecedents of the goal sequent will occur in every sequent in the proof after the initial stabilization phase. We can globalize these antecedents [5], which reduces the space required to store sequents and avoids unnecessary operations on them. Imogen implements a variety of term indexing algorithms [7], including path and substitution tree indexing to quickly retrieve elements of the databases. Experimental results show that in our case path indexing is more efficient than substitution tree indexing. The back end also maintains a descendent graph of the rules and sequents. This graph is used by the front end to reconstruct a natural deduction proof term of a proved formula. This gives external evidence that the proof found by Imogen is indeed a proof.

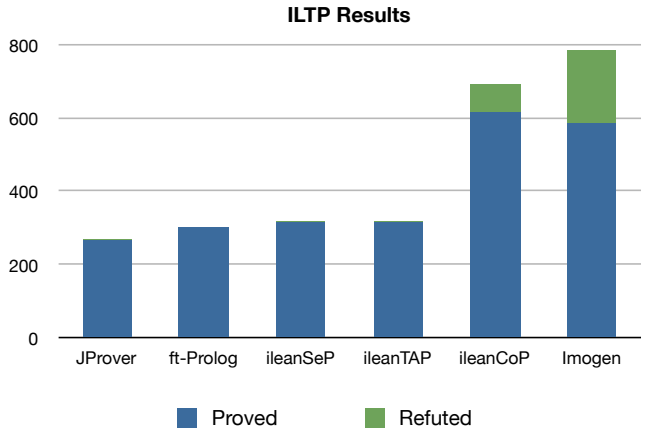
5 Performance Evaluation

We now give some performance statistics and internal comparisons of the effects of different optimizations. All of the Imogen statistics from this paper are from a 2.4 Ghz Intel Macintosh, Darwin 9.6.0, with 2Gb of memory. Imogen is written in Standard ML, and is compiled with MLton.

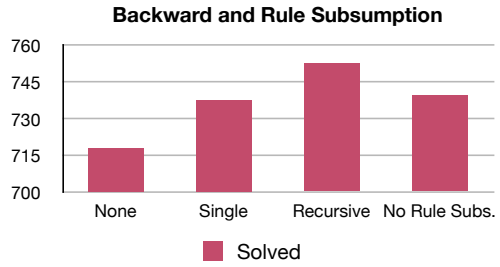
ILTP. We evaluated Imogen on ILTP, the Intuitionistic Logic Theorem Proving library [17]. The statistics from the ILTP website [16] are shown below. Currently the library gives detailed results for 6 intuitionistic theorem provers on 2550 problems,

with a time limit of 10 minutes. The other provers from ILTP use various optimizations of backward search. The non-Imogen statistics are run on a Xeon 3.4 GHz Linux, Mandrake 10.2. The amount of memory is not given on the website.

Besides solving the most total problems, Imogen does much better than other provers at disproving non-theorems. This is a similar result to Imogen's intuitionistic propositional theorem prover described in McLaughlin and Pfenning [12].

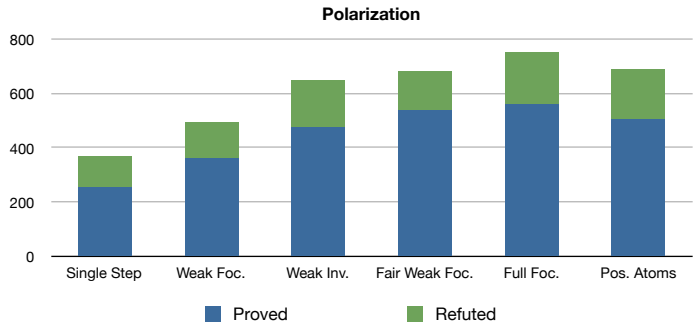


Subsumption. The following table shows the performance of Imogen with different settings for subsumption. Because of a lack of computer resources, we only ran the internal experiments on the a subset of the ILTP library (about 800 problems), and only for 10 seconds each. The first three columns are for backward subsumption settings. The last column does no rule subsumption.



Polarization. One benefit of the polarized inverse method is that it is simple to simulate different focusing strategies using appropriate placement of double shifts. For instance, if we wish to measure the effect of the inversion phase without the focusing phase, or vice versa, we can strategically insert double shifts $\downarrow\uparrow$ or $\uparrow\downarrow$ at the locations where focusing (or inversion) would take place. The double shifts will break the current phase and generate a block boundary. The following table gives the performance of some of these strategies, again using 10 second timeouts. *Single Step* simulates the unfocused inverse method. *Weak Focusing* makes all focusing phases complete, but breaks the inversion phases into single steps. *Weak Inversion* makes the inversion phase complete, but

breaks the focusing phase into single steps. *Fair Weak Focusing* is like weak focusing but allows the initial stabilization phase to run unchecked. In all of these experiments, we assigned negative polarity to all atoms. *Positive Atoms* makes all atoms positive, but otherwise does no unnecessary shifting.



6 Conclusion

In this paper we presented a basis for forward reasoning using the polarized inverse method, and demonstrated its practical effectiveness in the case of intuitionistic logic. In related work, Pientka et. al. [15] describe an experimental implementation of a focused inverse method for LF. Chaudhuri [3] describes a focused inverse method prover for linear logic. Earlier work is by Tammet [19] who describes an implementation of a forward intuitionistic theorem prover. We did not compare Imogen to his system because it is not part of ILTP. According to the ILTP website [16], the existing implementation is unsound.

Our work is by no means complete. While the current implementation is flexible with polarities, we have not investigated more than rather trivial heuristics for selecting the polarity of atoms, conjunctions, and inserting shifts. It is known, for instance [5], that using positive atoms simulates backward chaining in the inverse method. In our experiments however, we find that Imogen performs poorly on some problems that backchaining solves quickly. Using positive atoms, we can solve at least 50 more ILTP problems in less than a minute, but knowing when to choose positive atoms is still an open question.

One new optimization would be to consider a subordination relation on propositions [14]. This would prune the search space by deleting sequents of the form $\Gamma, p \longrightarrow q$ if no proof of q could depend on a proof of p as determined by the subordination relation. We are currently extending Imogen to other logics, including first order logic with constraints, and LF.

Acknowledgments. This work has been partially supported by the Air Force Research Laboratory grant FA87500720028 *Accountable Information Flow via Explicit Formal Proof* and the National Science Foundation grant NSF-0716469 *Manifest Security*. The authors would like to thank Kaustuv Chaudhuri for helpful insight into his experience with focusing the inverse method in linear logic, Jens Otten for his help with the ILTP library, and Geoff Sutcliffe for help with the TPTP library.

References

1. J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.
2. J.-M. Andreoli. Focussing and proof construction. *Annals of Pure and Applied Logic*, 107(1–3):131–163, 2001.
3. K. Chaudhuri. *The Focused Inverse Method for Linear Logic*. PhD thesis, Carnegie Mellon University, Dec. 2006. Technical report CMU-CS-06-162.
4. K. Chaudhuri and F. Pfenning. Focusing the inverse method for linear logic. In L. Ong, editor, *Computer Science Logic*, pages 200–215. Springer, 2005.
5. K. Chaudhuri, F. Pfenning, and G. Price. A logical characterization of forward and backward chaining in the inverse method. *Journal of Automated Reasoning*, 40(2–3):133–177, 2008.
6. A. Degtyarev and A. Voronkov. The inverse method. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 4, pages 179–272. Elsevier Science, 2001.
7. P. Graf. *Term Indexing*, volume 1053 of *Lecture Notes in Computer Science*. Springer, 1996.
8. J. M. Howe. *Proof Search Issues in Some Non-Classical Logics*. PhD thesis, University of St. Andrews, Scotland, 1998.
9. F. Lamarche. Games semantics for full propositional linear logic. In *Logic in Computer Science*, pages 464–473, 1995.
10. C. Liang and D. Miller. Focusing and polarization in intuitionistic logic. In J. Duparc and T. A. Henzinger, editors, *Computer Science Logic*, pages 451–465. Springer, 2007.
11. S. Y. Maslov. An inverse method for establishing deducibility in classical predicate calculus. *Doklady Akademii nauk SSSR*, 159:17–20, 1964.
12. S. McLaughlin and F. Pfenning. Imogen: Focusing the polarized inverse method for intuitionistic propositional logic. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR*, pages 174–181. Springer, 2008.
13. J. Otten. Clausal connection-based theorem proving in intuitionistic first-order logic. In B. Beckert, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX*, Lecture Notes in Computer Science, pages 245–261. Springer, 2005.
14. F. Pfenning and C. Schürmann. System description: Twelf — A meta-logical framework for deductive systems. In H. Ganzinger, editor, *16th Conference on Automated Deduction (CADE)*, number 1632 in LNAI, pages 202–206. Springer, 1999.
15. B. Pientka, D. X. Li, and F. Pompigne. Focusing the inverse method for LF: a preliminary report. In *International Workshop on Logical Frameworks and Meta-languages:Theory and Practice (LFMTP'07)*, 2007.
16. T. Raths and J. Otten. The ILTP Library. `http://www.iltp.de`.
17. T. Raths, J. Otten, and C. Kreitz. The ILTP problem library for intuitionistic logic. *J. Autom. Reasoning*, 38(1-3):261–271, 2007.
18. C. Suttner and G. Sutcliffe. The TPTP problem library: TPTP v2.0.0. Technical Report AR-97-01, Institut für Informatik, TU München, 1997.
19. T. Tammet. A resolution theorem prover for intuitionistic logic. In M. A. McRobbie and J. K. Slaney, editors, *Conference on Automated Deduction*, pages 2–16. Springer, 1996.
20. N. Zeilberger. Focusing and higher-order abstract syntax. In G. C. Necula and P. Wadler, editors, *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008*, pages 359–369. ACM, 2008.