Disproving Using the Inverse Method by Iterative Refinement of Finite Approximations

Taus Brock-Nannestad and Kaustuv Chaudhuri

Inria & LIX/École polytechnique
{taus.brock-nannestad,kaustuv.chaudhuri}@inria.fr

Abstract. In first-order logic, forward search using a complete strategy such as the inverse method can get stuck deriving larger and larger consequence sets when the goal query is unprovable. This is the case even in trivial theories where backward search strategies such as tableaux methods will fail finitely. We propose a general mechanism for bounding the consequence sets by means of finite approximations of infinite types. If the inverse method also implements forward subsumption and globalization, then the search space under this approximation is finite. We therefore obtain a type-directed iterative refinement algorithm for disproving queries.
The method has been implemented for intuitionistic first-order logic, and we discuss its performance on a variety of problems.

1 Introduction

In classical first-order logic, searching for a proof or for a refutation amounts to the same thing due to the symmetry induced by an involutive negation. Classical theorem provers are therefore just as good at disproving a false conjecture as proving a true conjecture: the same search strategy applies to either case. However, for non-classical logics such as intuitionistic predicate logic, proof-search and refutation are drastically different. For proving, the search procedure simply has to explore enough of the search space to find the proof—completeness is not essential—but for refutations the procedure has to exhaustively search the *entire* space of derivations to make sure that no proof exists. Since search spaces are generally infinite for undecidable logics such as intuitionistic first-order logic, this kind of exhaustive exploration is challenging. The general technique is to use a *complete* search procedure, where the proof of this completeness is external to the logic in question, and then run the search algorithm *to failure*.

The inverse method [9] has proven to be one of the best search methods for both proof search and the above kind of refutation by failure [14], at least on the problems drawn from the ILTP benchmark suite [15]. The inverse method, like its classical cousins resolution [2] and superposition [1], has many desirable properties that make proof search efficient, particularly the proof-reuse and variable-locality that is intrinsic to forward search methods. The most powerful tool in the inverse method is *subsumption*, which discards any newly derived fact

that is simply an instance of a fact derived earlier. It is subsumption that makes the inverse method *saturating* even for infinite search spaces.

In this paper we are interested in refuting unprovable conjectures in intuitionistic first-order logic. Unfortunately, even for simple instances of such conjectures, the inverse method tends to run forever. Indeed, every run of the inverse method can have one of three possible outcomes, of which only the first two are desirable:

1. Search finds a proof of the end-sequent.
2. Search saturates with no proof of the end-sequent.
3. Search continues indefinitely, neither finding a proof nor saturating.

As an illustration of this third possibility, consider the following simple axioms, which characterize the even natural numbers:

$$\mathtt{E}(\mathtt{z}). \qquad \forall x.\, \mathtt{E}(x) \supset \mathtt{E}(\mathtt{s}(\mathtt{s}(x))).$$

In the focused version of the inverse method [4, 14], the above axioms would be transformed into the following synthetic inference rules.

$$\frac{}{\cdot \longrightarrow \mathtt{E}(\mathtt{z})} \qquad \frac{\cdot \longrightarrow \mathtt{E}(x)}{\cdot \longrightarrow \mathtt{E}(\mathtt{s}(\mathtt{s}(x)))}$$

Here, the x in the second rule signifies that this rule may match any instantiation of this variable. In contrast to this, the $\mathtt{z}$ in the first rule is a ground constant term. Now, given the (unprovable) goal of showing that 3 is even, i.e. $\cdot \longrightarrow \mathtt{E}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{z}))))$, the above rules can be combined to produce sequents of the following form:

$$\cdot \longrightarrow \mathtt{E}(\mathtt{s}(\mathtt{s}(\mathtt{z}))), \qquad \cdot \longrightarrow \mathtt{E}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{z})))))), \qquad \cdot \longrightarrow \mathtt{E}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{z}))))))), \ldots$$

and so on. At no point do we prove the desired goal, of course, but neither do we saturate. Indeed, if we were to run this example through the Imogen prover [13], which currently solves the largest fragment of the first-order ILTP problems, we would observe the looping behavior until all available memory is exhausted. Moreover, this example does not stress any of the technological aspects of the inverse method implementation such as the term-indexing, subsumption checking, or ordering heuristics; an implementation lacking any sophistication would perform no worse than the most sophisticated of implementations.

In this paper, we show (in Secs. 4 and 5) how to adapt the inverse method (sketched in Sec. 2) in such a way the core proof search procedure always terminates with one of the following outcomes:

1. Saturation without proof – in which case the conjecture is not provable.
2. Discovery of a sound proof – in which case the conjecture is provable.
3. Discovery of an *unsound* proof.

The third outcome is interesting. *A priori* it would seem that an "unsound proof" is completely useless, as it neither proves the goal nor disproves the existence of a valid proof. In fact there is useful information to be extracted from such proofs. As we shall see in Sec. 6, the exact nature of the unsoundness can be used

to automatically *refine* our conjecture in such a way that if we rerun our proof search it is now guaranteed to avoid proofs that use that particular instance of unsound reasoning. Of course, this process of refinement may need to be be repeated indefinitely, and because of undecidability, it may never terminate with a sound proof or saturate without proof. Each round of the procedure, however, is guaranteed to terminate, and for problems like the one above we do eventually find a refutation.

2 Background: Forward Search Using the Inverse Method

We begin with a quick sketch of the inverse method for first-order intuitionistic logic. A comprehensive description of the inverse method, including its history and its applicability to a variety of logics, can be found in [9]. In this work we will use a *focused* and *polarized* version of the method based on the design explained in more detail in [4, 13, 14]. Focusing and polarities are greatly beneficial for exploiting the technique outlined in this paper to the fullest, but they are not essential; moreover, they are now standard and well-documented concepts of structural proof theory [6, 10].

Our language consists of standard first-order *terms* (written $s, t, \dots$) and *formulas* (written $A, B, \dots$) that are built with the following grammar:

$$s, t, \dots ::= x \mid f(t_1, \dots, t_n)$$
$$A, B, \dots ::= p(t_1, \dots, t_n) \mid A \supset B \mid A \wedge B \mid \top \mid A \vee B \mid \bot \mid \forall x.\, A \mid \exists x.\, A$$

Here, $f, g, \dots$ ranges over *function symbols*, $p, q, \dots$ over *predicate symbols*, and $x, y, \dots$ over *variables*. We will use $P, Q, \dots$ to denote *atomic formulas*, *i.e.*, formulas of the form $p(t_1, \dots, t_n)$. We assume that function and predicate symbols are simply typed, and that all well-formed formulas are also well-typed. This in turn uniquely determines a type for all variables. For the time being, we omit these types from the depictions of formulas and terms; we will revisit them in Sec. 4. Following standard practice, we omit parentheses for nullary predicate and function symbols. Specific concrete function and predicate symbols will be written in a `monospaced` font. For a function symbol f, we write $f^n(t)$ to stand for t if $n = 0$ and for $f(f^{n-1}(t))$ if $n > 0$. For intuitionistic logic, the above collection of formula constructors has the property that no connective is definable in terms of the others. On the other hand, negation $\neg A$ is defined to be $A \supset \bot$, and equivalence $A \equiv B$ as $(A \supset B) \wedge (B \supset A)$.

Provability of sequents will be given in terms of a forward version of Gentzen's *sequent calculus* LJ, which we call FJ. An FJ *sequent* is of the form $\Gamma \longrightarrow \gamma$ where Γ, called the *context*, is a multiset of formulas and γ, called the *conclusion*, is either $\cdot$ or a formula. The rules of FJ are depicted in Figure 1.

Definition 1 (Notational Conventions in Figure 1).

- *In the* $\vee_{\mathrm{L}}$ *rule:* $\gamma_1 \cup \gamma_2 = \begin{cases} \gamma_1 & \text{if } \gamma_2 = \cdot \\ \gamma_2 & \text{if } \gamma_1 = \cdot \\ C & \text{if } \gamma_1 = \gamma_2 = C. \end{cases}$
  The rule is inapplicable if γ_1 *and* γ_2 *are different formulas.*

$$\left[\frac{\Gamma, P \longrightarrow \gamma}{\Gamma, P' \longrightarrow \gamma}\,\mathrm{u_L} \qquad \frac{\Gamma \longrightarrow P}{\Gamma \longrightarrow P'}\,\mathrm{u_R}\right] \qquad \frac{}{P \longrightarrow P}\,\mathrm{init} \qquad \frac{\Gamma, A, A \longrightarrow \gamma}{\Gamma, A \longrightarrow \gamma}\,\mathrm{factor}$$

$$\frac{\Gamma_1 \longrightarrow A \quad \Gamma_2 \longrightarrow B}{\Gamma_1, \Gamma_2 \longrightarrow A \wedge B}\wedge_{\mathrm{R}} \qquad \frac{\Gamma, A \longrightarrow \gamma}{\Gamma, A \wedge B \longrightarrow \gamma}\wedge_{\mathrm{L1}} \qquad \frac{\Gamma, B \longrightarrow \gamma}{\Gamma, A \wedge B \longrightarrow \gamma}\wedge_{\mathrm{L2}} \qquad \frac{}{\cdot \longrightarrow \top}\top_{\mathrm{R}}$$

$$\frac{\Gamma \longrightarrow \gamma}{\Gamma\backslash\{A\} \longrightarrow A \supset B}\supset_{\mathrm{R}} \qquad \frac{\Gamma_1 \longrightarrow A \quad \Gamma_2, B \longrightarrow \gamma}{\Gamma_1, \Gamma_2, A \supset B \longrightarrow \gamma}\supset_{\mathrm{L}}$$

$$\frac{\Gamma_1, A \longrightarrow \gamma_1 \quad \Gamma_2, B \longrightarrow \gamma_2}{\Gamma_1, \Gamma_2, A \vee B \longrightarrow \gamma_1 \cup \gamma_2}\vee_{\mathrm{L}} \qquad \frac{\Gamma \longrightarrow A}{\Gamma \longrightarrow A \vee B}\vee_{\mathrm{R1}} \qquad \frac{\Gamma \longrightarrow B}{\Gamma \longrightarrow A \vee B}\vee_{\mathrm{R2}} \qquad \frac{}{\bot \longrightarrow \cdot}\bot_{\mathrm{L}}$$

$$\frac{\Gamma \longrightarrow A}{\Gamma \longrightarrow \forall x.\, A}\forall_{\mathrm{R}}\{x\} \qquad \frac{\Gamma, [t/x]A \longrightarrow \gamma}{\Gamma, \forall x.A \longrightarrow \gamma}\forall_{\mathrm{L}} \qquad \frac{\Gamma \longrightarrow [t/x]A}{\Gamma \longrightarrow \exists x.A}\exists_{\mathrm{R}} \qquad \frac{\Gamma, A \longrightarrow \gamma}{\Gamma, \exists x.\, A \longrightarrow \gamma}\exists_{\mathrm{L}}\{x\}$$

Fig. 1. FJ, a forward sequent calculus for intuitionistic first-order logic. Note the conventions in Defn. 1. The $\mathrm{u_L}$ and $\mathrm{u_R}$ rules are not part of FJ and will be explained in Sec. 5.

- *In the* $\supset_{\mathrm{R}}$ *rule, we assume that* $\Gamma \backslash \{A\} \subsetneq \Gamma$ *or* $\gamma = B$.
- *In the* $\forall_{\mathrm{R}}\{x\}$ *and* $\exists_{\mathrm{L}}\{x\}$ *rules, the variable* x *is not free in the conclusion.*

The distinguishing feature of FJ is that every element of Γ is necessary in the proof of $\Gamma \longrightarrow \gamma$, *i.e.*, this calculus actually encodes a *strict* or *relevant* logic. Full intuitionistic truth is then recovered by means of *subsumption.*

Definition 2 (Substitutions). *A* substitution θ *is a finite mapping from variables to terms such that no variable in its domain occurs among the terms in its range.*[1] *For any variable* x*, we write* $x[\theta]$ *to stand for* x *if* $x \notin dom(\theta)$*, and for* $\theta(x)$ *otherwise. Given a syntactic construct* X *(term, formula, sequent,* etc.*), we write* $X[\theta]$ *for the result of replacing every free variable* x *in* X *by* $x[\theta]$*, avoiding capture by* α*-varying* X *if needed.*

Definition 3 (Subsumption). *The sequent* $\Gamma_1 \longrightarrow \gamma_1$ subsumes $\Gamma_2 \longrightarrow \gamma_2$ *iff there is a substitution* θ *such that* $\Gamma_1[\theta] \subseteq \Gamma_2$ *and* $\gamma_1[\theta] \subseteq \gamma_2$*, where* $\subseteq$ *is interpreted as* set*-inclusion,* i.e.*,* $\Gamma_1 \subseteq \Gamma_2$ *iff for every* $A \in \Gamma_1$ *also* $A \in \Gamma_2$*. This notion is naturally generalized to sets of sequents.*

Definition 4 (Derivability). *The sequent* $\Gamma_0 \longrightarrow \gamma_0$ *is* derivable *if there is an FJ derivation of* $\Gamma \longrightarrow \gamma$ *(for some* Γ *and* γ*) that subsumes* $\Gamma_0 \longrightarrow \gamma_0$.

Note that this calculus is cut-free, and hence enjoys a *subformula property*: every sequent in a derivation is built out of (signed) subformulas of the end-sequent. The *inverse method* makes use of this property of the forward calculus by following the "recipe" outlined in [9]. In rough outline, ground sequents are *lifted* to sequents with free term variables, and identity of terms and formulas is replaced by unification and considering the *most general common instance*,

[1] In other words, substitutions are idempotent.

computed using most general unifiers (mgus). Here are three particular but characteristic examples of lifted rules:

$$\frac{\Gamma, A, A' \longrightarrow \gamma \qquad \theta = \mathrm{mgu}(A, A')}{(\Gamma, A \longrightarrow \gamma)[\theta]}\ \text{factor} \qquad \frac{\Gamma_1 \longrightarrow A' \qquad \Gamma_2 \longrightarrow B' \qquad \theta = \mathrm{mgu}(A' \wedge B', A \wedge B)}{(\Gamma_1, \Gamma_2 \longrightarrow A \wedge B)[\theta]}\ \wedge_{\mathrm{R}} \qquad \frac{\Gamma \longrightarrow A' \qquad \theta = \mathrm{mgu}(A', A)}{(\Gamma \longrightarrow \exists x.\, A)[\theta]}\ \exists_{\mathrm{R}}$$

The subformula property allows all the lifted rules to be further *specialized* to the signed subformulas of an *end-sequent*, which we denote by $\Gamma_0 \longrightarrow \gamma_0$ in the rest of this paper. Specifically, the principal formulas in each case (the unprimed formulas in the example rules above) are freely occurring signed subformulas of the end-sequent. In particular, the initial sequents produced by init correspond to the atomic formulas that occur both positively and negatively signed in the end-sequent. We say that these initial sequents and specialized inference rules are *based on* the end-sequent.

Search begins from an initial *set of support* (SOS) consisting of the initial sequents based on the end-sequent. Then, in each iteration of the inner loop, a sequent is *selected* from the SOS and moved into the *active* set; each specialized rule based on the end-sequent is then applied in such a way that at least one of its premises is the selected sequent and the other premises are drawn from the active set. Every conclusion of these rule applications is then tested for subsumption against all the sequents derived earlier; any new sequents that are not subsumed are inserted back into the SOS.[2] As long as the selection of sequents from the SOS is fair—every sequent is eventually selected—the search method is complete, *i.e.*, it will eventually derive a sequent that subsumes the end-sequent if it is provable. This core prover loop is essentially unchanged from the days of the Otter resolution prover, and is therefore often called the *Otter loop*. When the SOS becomes empty without the end-sequent being subsumed, we say that search has *saturated*, which in turn means that the end-sequent is *refuted*—not derivable—and hence the end-sequent does not denote a true formula of intuitionistic first-order logic.

We make two modifications to the standard Otter loop. First, we apply the factor rule eagerly on every computed sequent, storing each intermediate result (if not subsumed) in the SOS, until factor is longer applicable. Thus, we never need to consider applying factor to any selected sequent. Second, we add the following rule, which is easily seen to be admissible:

$$\frac{\Gamma, A' \longrightarrow \gamma \qquad A \in \Gamma_0 \qquad \theta = \mathrm{match}(A', A)}{(\Gamma \longrightarrow \gamma)[\theta]}\ \text{global}$$

Here, we write $\mathrm{match}(A', A)$ to stand for the most general substitution θ for which $A'[\theta] = A$. The effect of this rule is to treat every element of Γ_0 as implicitly

[2] This is sometimes called *forward subsumption* to distinguish it from the opposite operation: deleting an earlier sequent from the SOS and active sets if it is subsumed *by* a newly derived sequent, known as *back-subsumption*. While this is critical for performance, back-subsumption is not essential for this paper, so we will use "subsumption" in this paper to mean forward subsumption.

present in the context of every derived sequent. In fact, we will consider this rule as implicitly applied to the principal formula of every computed sequent, which is sometimes called *globalization* [4, 14].

3 Guaranteeing Termination – Informally

Our aim in this paper is to build a variant of the inverse method where the Otter loop for any end-sequent *terminates*, either by producing a proof or by saturating. In broad terms, our method is based on building an *over-approximation* of the set of derivable sequents from the initial sequents based on the end-sequent. Importantly, we retain *completeness* by this over-approximation, so our method can be validly used to *refute* goals. Indeed, we will sacrifice *soundness* to obtain this over-approximation.

In our particular case, the over-approximation comes in the form of *weakening* the end-sequent that the specialized rules are based on. It is immediate that if $\Gamma \longrightarrow A$ is derivable, then $\Gamma, \Gamma' \longrightarrow A$ is as well, and hence if we succeed in refuting the latter sequent, then the former sequent cannot be derivable. In backward search procedures, reasoning from end-sequent upwards to the initial sequents, applying weakening is generally bad for performance: it can only create *more* backtrack points for the prover. For forward search, however, there is no backtracking; indeed, having more assumptions in the basis can produce initial sequents that subsume (and hence filter out) sequents that may otherwise end up in the SOS. Because subsumption is used in such a key fashion, it is perhaps instructive to think of it in terms of the following intuition: a variable subsumes all instances of said variable. Thus, if x is a variable, then the sequent $\Gamma, \mathtt{E}(\mathtt{s}(x)) \longrightarrow \cdot$ subsumes, *e.g.*, $\Gamma, \Gamma', \mathtt{E}(\mathtt{s}(x)) \longrightarrow \cdot$ and $\Gamma, \mathtt{E}(\mathtt{s}(\mathtt{s}(\mathtt{z}))) \longrightarrow A$.

To see a concrete illustration of this approach, let us revisit the example from the introduction. We will modify the end-sequent by adding the assumption, P_4, that all numbers greater than four are even, so we base the initial sequents and specialized rules on:

$$\underbrace{\forall x.\, \mathtt{E}(\mathtt{s}^4(x))}_{P_4},\ \underbrace{\mathtt{E}(\mathtt{z}),\ \forall x.\, \mathtt{E}(x) \supset \mathtt{E}(\mathtt{s}^2(x))}_{\Gamma_0} \longrightarrow \mathtt{E}(\mathtt{s}^3(\mathtt{z})).$$

For the sake of simplicity, this assumption is slightly weaker than the ones we use in the rest of the paper. The full method we propose would rather add the assumption "if there exists an even number greater than four, then all numbers greater than four are even."

As explained earlier, these hypotheses are used to specialize the inference rules. In the presence of specialization, focusing, and globalization, we effectively have only the following *derived* (or *synthetic*) inference rules:

$$\frac{}{\cdot \longrightarrow \mathtt{E}(\mathtt{s}^4(x))} \qquad \frac{}{\cdot \longrightarrow \mathtt{E}(\mathtt{z})} \qquad \frac{\cdot \longrightarrow \mathtt{E}(y)}{\cdot \longrightarrow \mathtt{E}(\mathtt{s}^2(y))}$$

The first two rules actually give rise to two (lifted) sequents. If the first of these sequents is applied to the third rule, we obtain $\cdot \longrightarrow \mathtt{E}(\mathtt{s}^6(x))$, which is subsumed

by the first sequent. If we try the second of the above sequents with the third rule, we obtain, successively, the sequents $\cdot \longrightarrow \mathtt{E}(\mathtt{s}^2(\mathtt{z}))$ and $\cdot \longrightarrow \mathtt{E}(\mathtt{s}^4(\mathtt{z}))$; the former fits the premise of no other rule and the latter is subsumed by the first sequent. We have then exhausted all possibilities for combining the above rules, so search terminates *without* finding a proof of $P_4, \Gamma_0 \longrightarrow \mathtt{E}(\mathtt{s}^3(\mathtt{z}))$. By completeness it now follows that this sequent was not derivable in the first place, so neither was $\Gamma_0 \longrightarrow \mathtt{E}(\mathtt{s}^3(\mathtt{z}))$.

This result may seem somewhat surprising. By adding *more* hypotheses—which one would naïvely assume just leads to more sequents being derivable—we are actually able to drastically *decrease* the size of the search space. Why did this happen, and how did we discover this particular weakening of the end-sequent? We shall explain this in the next two sections by showing how the forward search space can be guaranteed to be finite.

4 Cofinite Covers

As already mentioned in Sec. 2, we will assume that our function and predicate symbols have simple types. To every function symbol, we associate a type which we will write as $T_1 \times \cdots \times T_n \to T$, and to every predicate symbol, we will associate a type written as $T_1 \times \cdots \times T_n \to o$, where n is the arity of the function or predicate, and T_i is the type of the ith argument of the function or predicate. We use o for the "type" of formulas. For constant function symbols, we elide the arrow. In the following we will only consider terms and atoms that are well-formed *i.e.*, all terms occurring in these must obey the typing discipline. We also assume that all types are inhabited, as uninhabited types are never needed in a proof.

As an example, consider the following signature which defines types for the natural numbers, lists of numbers, and an append predicate:

$$\mathtt{z} : \mathtt{nat}. \qquad \mathtt{s} : \mathtt{nat} \to \mathtt{nat}. \qquad \mathtt{nil} : \mathtt{list}.$$
$$\mathtt{cons} : \mathtt{nat} \times \mathtt{list} \to \mathtt{list}. \qquad \mathtt{append} : \mathtt{list} \times \mathtt{list} \times \mathtt{list} \to o.$$

A benefit of this representation is that nonsensical terms such as $\mathtt{s}(\mathtt{nil})$ are not possible to construct. Note that by collapsing all types into a single type, we get a system that is essentially equivalent to ordinary untyped first-order logic. If we do have types at our disposal, however, the efficiency of our approach is greatly improved. For many untyped problems, it is possible to infer nontrivial typing information from the given formulas, e.g. using the method presented in [8].

The main construction of this section is a form of case analysis on terms, where we allow splitting a single occurrence of a variable of a given type T into all possible function symbols with codomain T. To fully describe this operation, we would therefore need to keep track of which variables occur where, and what the types of these variables are. In our case, however, all variables may be assumed to be distinct, and we may therefore use the following more parsimonious notation in our presentation of the splitting procedure:

Definition 5 (Free terms and atoms). *The free terms and free atoms are generated by the following grammar.*

$$\bar{t} ::= T \mid f\langle \bar{t}_1, \ldots, \bar{t}_n \rangle \qquad \bar{P} ::= P\langle \bar{t}_1, \ldots, \bar{t}_n \rangle$$

The formation of free terms and atoms should respect the types, thus $f\langle t_1, \ldots, t_n\rangle$ *and* $P\langle t_1, \ldots, t_n\rangle$ *are well-formed if and only if for all* $0 \leq i \leq n$, t_i *is a well-formed free term of type* T_i*. For the purposes of this definition,* T *is considered a well-formed free term of type* T*.*

Intuitively, a free atom should be interpreted as representing any instantiation of the base types present in the atom. In other words, the free atom $\texttt{append}\langle\texttt{cons}\langle\texttt{nat}, \texttt{list}\rangle, \texttt{list}, \texttt{list}\rangle$ should be seen as representing the atomic formula $\texttt{append}(\texttt{cons}(n, l_1), l_2, l_3)$ for any n of type `nat` and l_1, l_2, l_3 of type `list`.

More formally, we define the following relationship between terms, atoms and their free counterparts.

Definition 6 (Instance of free term/atom). *A term* t *is said to be an instance of a free term* $\bar{t}$ *if one of the following holds:*

1. $\bar{t} = T$*, and* t *is a term with type* T*, or*
2. $\bar{t} = f\langle \bar{t}_1, \ldots, \bar{t}_n\rangle$*,* $t = f(t_1, \ldots, t_n)$ *and for* $1 \leq i \leq n$, t_i *is an instance of* $\bar{t}_i$*.*

The atom $P(t_1, \ldots, t_n)$ *is said to be an instance of the free atom* $P\langle \bar{t}_1, \ldots, \bar{t}_n\rangle$ *if for all* $1 \leq i \leq n$, t_i *is an instance of the free term* $\bar{t}_i$*.*

The main goal will be showing that for every predicate symbol P we can find a suitable collection Γ_P of free atoms such that all but finitely many instances of P are instances of some free atom in Γ_P.

To express this more formally, we first introduce the concept of linear contexts:

Definition 7 (Linear context in free term/atom). *The following grammar defines the notion of a free term or atom with a specific chosen subterm*

$$\bar{t}[-] ::= \Box \mid f\langle \bar{t}_1, \ldots, \bar{t}_i[-], \ldots, \bar{t}_n\rangle$$
$$\bar{P}[-] ::= P\langle \bar{t}_1, \ldots, \bar{t}_i[-], \ldots, \bar{t}_n\rangle$$

With linear contexts there is a natural notion of substitution, defined by the following equations:

$$\Box[\bar{t}] = \bar{t}$$
$$f\langle \bar{t}_1, \ldots, \bar{t}_i[-], \ldots, \bar{t}_n\rangle[\bar{t}] = f\langle \bar{t}_1, \ldots, \bar{t}_i[-][\bar{t}], \ldots, \bar{t}_n\rangle$$
$$P\langle \bar{t}_1, \ldots, \bar{t}_i[-], \ldots, \bar{t}_n\rangle[\bar{t}] = P\langle \bar{t}_1, \ldots, \bar{t}_i[-][\bar{t}], \ldots, \bar{t}_n\rangle$$

Definition 8 (Free instance of function/predicate). *For any function symbol* f *with type* $T_1 \times \cdots \times T_n \to T$*, we define the free instance of* f *to be the free term* $f\langle T_1, \ldots, T_n\rangle$*. We let* φ *denote the function that maps a function symbol to its free instance. Similarly, for any predicate symbol* P *with type* $T_1 \times \cdots \times T_n \to o$ *we define its free instance to be* $\varphi(P) = P\langle T_1, \ldots, T_n\rangle$*.*

Note that any instance of a predicate P is also an instance of $\varphi(P)$.

Based on the above definition, we can now present a notion of *coverage*, relating a free atom to a set of free atoms with the same instances. First, we define the set of *splitting candidates*:

Definition 9 (Splitting candidates). *The set* $\mathsf{SC}(T)$ *for a type* T *is given by*

$$\mathsf{SC}(T) = \{f \mid f : T_1 \times \cdots \times T_n \to T\}$$

Definition 10 (Coverage). *We say that $\bar{\Gamma}$ covers $\bar{P}$, if $\bar{\Gamma} \rhd \bar{P}$ holds, where this judgment is defined using the following inference rules*

$$\frac{}{\bar{P} \rhd \bar{P}}\,\text{imm} \qquad \frac{\forall f \in \mathsf{SC}(T).\quad \bar{\Gamma}_f \rhd \bar{P}[\varphi(f)]}{\biguplus_{f \in \mathsf{SC}(T)} \bar{\Gamma}_f \rhd \bar{P}[T]}\,\text{split}$$

Here, the second rule has a varying number of premises depending on the type T. The $\uplus$ signifies that the contexts in the premises are combined using multiset union to form the context in the conclusion.

Note that by construction, any instance of $\bar{P}[\varphi(f)]$ is an instance of $\bar{P}[T]$.

Definition 11 (Ground cover). *If a free atom $\bar{P}$ contains only function symbols, we say that it is a ground cover. In this case, there is exactly one ground atom P such that P is an instance of $\bar{P}$.*

As an example of the above definitions, consider a unary predicate P over $\mathtt{nat}$. In this case, we can derive

$$P\langle \mathtt{z} \rangle, P\langle \mathtt{s}\langle \mathtt{z} \rangle\rangle, P\langle \mathtt{s}\langle \mathtt{s}\langle \mathtt{nat} \rangle\rangle\rangle \rhd P\langle \mathtt{nat} \rangle,$$

and among the three covering free atoms, the first two are ground covers.

Remark 12. If a type T has only finitely many inhabitants, it is never necessary to split on this type. It is therefore possible to extend the above definition to include free atoms that contain only function symbols or finite types without changing the properties of these covers.

We may now define formally the notion of a set of free atoms that cover all but finitely many ground instances of a predicate.

Definition 13 (Cofinite cover). *We say that $\bar{\Gamma}$ is a* cofinite cover *for $\bar{P}$ if $\bar{\Gamma}, \bar{\Gamma}' \rhd \bar{P}$ where $\bar{\Gamma}'$ contains only ground covers, and $\bar{\Gamma}$ contains no ground covers. We write this as $\bar{\Gamma} \blacktriangleright \bar{P}$.*

Note that as our goal is to cover all but finitely many ground instances, we may as well discard all instances that happen to have a ground cover.

As an example, we have $P\langle \mathtt{s}\langle \mathtt{s}\langle \mathtt{nat} \rangle\rangle\rangle \blacktriangleright P\langle \mathtt{nat} \rangle$. Any ground instance of $P : \mathtt{nat} \to o$ is of the form $P(\mathtt{s}^n(\mathtt{z}))$ for some n, and all but $P(\mathtt{z})$ and $P(\mathtt{s}(\mathtt{z}))$ are instances of $P\langle \mathtt{s}\langle \mathtt{s}\langle \mathtt{nat} \rangle\rangle\rangle$.

5 Termination

We are now in a position to explain the $\mathrm{u_L}$ and $\mathrm{u_R}$ rules from Figure 1. First of all, we will assume there is a fixed finite set of free atoms $\bar{\Gamma}$. The only side-condition on the $\mathrm{u_L}/\mathrm{u_R}$ rule is now that whenever it is applied, both P and P' must be instances of the same free atom $\bar{P} \in \bar{\Gamma}$. For the $\mathrm{u_R}$ rule, this has the following effect: during forward proof search, the rule only gets applied if we manage to prove $\Gamma \longrightarrow P$ where P is an instance of $\bar{P}$. If $\Gamma \longrightarrow P$ is never derived, then

the rule never becomes active, and thus does not influence proof search. If on the other hand the rule becomes active, we can now immediately derive $\Gamma \longrightarrow P'$ for *all* instances P' of $\bar{P}$. In a sense, the free atoms act as *sentinels* that watch over an infinite set of instances. If the sentinel becomes active, it immediately subsumes any instances inside the set it is watching over. As we will show in this section, as long as these sentinels form a cofinite cover, proof search is guaranteed to terminate. Of course, extending the FJ calculus with this unfamiliar construct may seem a bit complicated, but as we shall see in Sec. 7, the complexity is in fact only skin deep, as the behavior of these rules can be implemented in terms of the usual rules of the calculus.

Definition 14 (Augmented sequents). *We say that a sequent $\Gamma \longrightarrow A$ is* augmented with $\bar{\Gamma}$ *if $\bar{\Gamma} = \bar{\Gamma}_{P_1}, \ldots, \bar{\Gamma}_{P_n}$, where $P_1, \ldots, P_n$ are all the predicate symbols occurring in Γ and A, and $\bar{\Gamma}_{P_i} \blacktriangleright \varphi(P_i)$ for all $1 \leq i \leq n$.*

Note that any sequent $\Gamma \longrightarrow A$ can be turned into an augmented sequent by letting $\bar{\Gamma}$ consist of $\varphi(P)$ for all predicate symbols P occurring in Γ and A.

Theorem 15 (Termination). *The inverse method is terminating for the end-sequent $\Gamma_0 \longrightarrow \gamma_0$ augmented with $\bar{\Gamma}_0$; that is to say, the iterated consequences of all initial sequents based on this end-sequent is a finite set.*

Proof (sketch). It is sufficient to show that only finitely many distinct collections of atoms may be derived before the set of iterated consequences of the initial sequents is saturated. Every lifted inference rule is of the form:

$$\frac{\Gamma_1 \longrightarrow A_1 \quad \cdots \quad \Gamma_n \longrightarrow A_n}{\Gamma \longrightarrow A}\,\{\Psi\}$$

for which the variables in Ψ can be instantiated with any terms. Note, however, that because $\bar{\Gamma}_0$ is a cofinite cover, all but finitely many instances of this rule will have a conclusion that is immediately subsumed, either by an instance of the $\mathrm{u_L}$ or $\mathrm{u_R}$ rule, or by a previously derived sequent. It therefore follows that each inference rule is applied only finitely many times. Since there are only finitely many subformulas of the end-sequent, this guarantees that the consequences of the initial sequents are finite. □

6 Refinement

As we have now shown, augmenting sequents with cofinite covers for each predicate symbol ensures that the inverse method with subsumption terminates on all queries. Moreover, we can trivially turn a sequent into an augmented sequent by adding free atoms of the form $\varphi(P)$ for every predicate symbol P. This is tantamount to saying that all predicates are true for all ground instances, which is almost certain to result in an unsound proof that relies on some ground instance of a free atom. The main goal of this section, then, is to show that if an unsound proof is found, then we can always refine our cofinite cover to additionally exclude the instance that lead to this unsoundness.

We start with a few necessary lemmas.

Lemma 16 (Covers refine). *If $\bar{\Gamma} \rhd \bar{P}$ then any instance P of some $\bar{P}' \in \bar{\Gamma}$ is an instance of $\bar{P}$.*

Lemma 17 (Strictness). *If $\bar{\Gamma} \rhd \bar{P}$ and P is an instance of $\bar{P}$, then there is exactly one $\bar{P}' \in \bar{\Gamma}$ such that P is an instance of $\bar{P}'$.*

Lemma 18 (Ground coverage). *If P is a ground instance of $\bar{P}$, there exists a $\bar{\Gamma}$ such that $\bar{\Gamma} \rhd \bar{P}$, and P is an instance of a ground cover $\bar{P}' \in \bar{\Gamma}$.*

By iterating the above lemma, we get the following easy corollary.

Corollary 19. *For any set Γ of ground instances of a predicate P, there exists a $\bar{\Gamma}$ such that $\bar{\Gamma} \rhd \bar{P}$ and every $P' \in \Gamma$ has a ground cover in $\bar{\Gamma}$.* □

Lemma 20 (Refinement). *For any set Γ of ground instances of a predicate P there exists a cofinite cover $\bar{\Gamma} \blacktriangleright \bar{P}$ such that no $P' \in \Gamma$ is an instance of some $\bar{P}' \in \bar{\Gamma}$.*

Proof. From the previous corollary, it follows that we may find a $\bar{\Gamma}$ such that $\bar{\Gamma} \rhd \bar{P}$, and every $P' \in \Gamma$ has a ground cover in $\bar{\Gamma}$. Let $\bar{\Gamma}'$ be the subset of ground covers in $\bar{\Gamma}$. It is now immediate that $\bar{\Gamma} \setminus \bar{\Gamma}' \blacktriangleright \bar{P}$. □

Remark 21. Note that simply doing case splitting and discarding ground covers does not necessarily result in a minimal cofinite cover. Consider a binary predicate P over `nat` for which we wish to exclude $P(\mathtt{s}(\mathtt{z}), \mathtt{s}(\mathtt{z}))$. By the above procedure, we could get e.g. the following cofinite cover:

$$P\langle \mathtt{z}, \mathtt{s}\langle \mathtt{nat}\rangle\rangle, P\langle \mathtt{s}\langle \mathtt{nat}\rangle, \mathtt{z}\rangle, P\langle \mathtt{s}\langle \mathtt{s}\langle \mathtt{nat}\rangle\rangle, \mathtt{s}\langle \mathtt{nat}\rangle\rangle, P\langle \mathtt{s}\langle \mathtt{z}\rangle, \mathtt{s}\langle \mathtt{s}\langle \mathtt{nat}\rangle\rangle\rangle,$$

which excludes exactly the atoms $P(\mathtt{z}, \mathtt{z})$ and $P(\mathtt{s}(\mathtt{z}), \mathtt{s}(\mathtt{z}))$. Note however, that the following would also work as a cofinite cover, and additionally exclude $P(\mathtt{z}, \mathtt{s}(\mathtt{z}))$ and $P(\mathtt{s}(\mathtt{z}), \mathtt{z})$:

$$P\langle \mathtt{nat}, \mathtt{s}\langle \mathtt{s}\langle \mathtt{nat}\rangle\rangle, P\langle \mathtt{s}\langle \mathtt{s}\langle \mathtt{nat}\rangle\rangle, \mathtt{nat}\rangle.$$

This is more of an implementation detail, however, as the specifics of which cofinite cover is chosen makes no difference with regard to saturation.

Theorem 22 (Refinement of sequents). *Given a proof of $\Gamma \longrightarrow A$ augmented with $\bar{\Gamma}$ it is possible to check whether this proof is also a valid proof of $\Gamma \longrightarrow A$. If the proof is not valid, it is possible to refine the set $\bar{\Gamma}$ into $\bar{\Gamma}'$ such that the derivation is not a valid proof of the sequent $\Gamma \longrightarrow A$ augmented with $\bar{\Gamma}'$.*

Proof. From the inference rules, it follows that the only way the elements of $\bar{\Gamma}$ interact with derivations is through the $\mathrm{u_L}$ and $\mathrm{u_R}$ rules. It therefore follows that if no instances of these rules appear in the proof, then the proof is in fact a valid proof even in the absence of $\bar{\Gamma}$. If, on the other hand, some $\bar{P} \in \bar{\Gamma}$ is used in the proof, it must be in the form of a $\mathrm{u_L}$ or $\mathrm{u_R}$ rule, e.g.

$$\frac{\Gamma'' \longrightarrow P}{\Gamma'' \longrightarrow P'}\,\mathrm{u_R}$$

In this case, we may use Lemma 20 to get a new set $\bar{\Gamma}'_P$, for which P is not an instance of any $\bar{P}'' \in \bar{\Gamma}'_P$, and replace $\bar{\Gamma}_P$ with $\bar{\Gamma}'_P$, putting $\bar{\Gamma}' = \bar{\Gamma} \setminus \{\bar{\Gamma}_P\} \cup \bar{\Gamma}'_P$. This precludes P from being used as the premise of the above $\mathrm{u_R}$ rule in a proof of $\Gamma \longrightarrow A$ augmented with $\bar{\Gamma}'$. □

With these two theorems in place, we may now perform our proof search as follows. To find a proof of $\Gamma \longrightarrow A$, first augment it with a suitable $\bar{\Gamma}$. Use the inverse method to search for a proof of this sequent. If it terminates without proof, $\Gamma \longrightarrow A$ is unprovable. If it terminates with a sound proof, $\Gamma \longrightarrow A$ is provable. If it terminates with an unsound proof, refine $\bar{\Gamma}$ into $\bar{\Gamma}'$, and repeat with $\bar{\Gamma}'$ in place of $\bar{\Gamma}$.

7 Implementation

While the description of the inverse method in the previous section used free terms and free atoms, and required a means of building the full proofs, in an actual implementation we dispense with them entirely. Indeed, we need very little beyond the ordinary inverse method for first-order intuitionistic logic, and these alterations are explained below.

Recall that when defining free atoms we suggested that one should consider the types T occurring in a free atom $\bar{P}$ as representing all instances of that type. This suggests that one may interpret T as a universally quantified variable. More formally, we have the following definition:

Definition 23. *The judgment $\Psi \vdash \bar{t} \mapsto t$ represents a mapping from free terms to terms in a context Ψ of typed eigenvariables. It is defined by the following inference rules*

$$\frac{}{x\colon T \vdash T \mapsto x} \qquad \frac{\Psi_1 \vdash \bar{t}_1 \mapsto t_1 \quad \cdots \quad \Psi_n \vdash \bar{t}_n \mapsto t_n}{\Psi_1, \ldots, \Psi_n \vdash \gamma\langle \bar{t}_1, \ldots, \bar{t}_n \rangle \mapsto \gamma(t_1, \ldots, t_n)}$$

where γ is either a function or predicate symbol. We furthermore require that all variables occurring in Ψ are distinct.

Note that up to renaming of the variables in Ψ, the derivation of the judgment $\Psi \vdash \bar{t} \mapsto t$ is unique for any given $\bar{t}$, hence the above defines a function from free atoms to atoms in a context of eigenvariables.

We may now define the interpretation of free atoms as follows

Definition 24. *Let $\alpha_{\mathrm{L}}, \alpha_{\mathrm{R}}$ be functions defined by the following equations:*

$$\alpha_{\mathrm{L}}(\bar{P}) = ((\forall \Psi.P) \supset \bot) \supset ((\exists \Psi.P) \supset \bot) \qquad \text{where } \Psi \vdash \bar{P} \mapsto P$$
$$\alpha_{\mathrm{R}}(\bar{P}) = (\exists \Psi.P) \supset (\forall \Psi.P) \qquad \text{where } \Psi \vdash \bar{P} \mapsto P$$

We define the function α on covers as $\alpha(\bar{\Gamma}) = \alpha_{\mathrm{L}}(\bar{\Gamma}), \alpha_{\mathrm{R}}(\bar{\Gamma})$. Note that this maps covers to contexts in FJ.

Theorem 25 (Equivalence). *The following methods are equivalent, in the sense that they are both terminating for the same choice of Γ and A:*

– *the polarized and focused inverse method for the end-sequent* $\Gamma \longrightarrow A$ *augmented with* $\bar{\Gamma}$ *and using the* $\mathrm{u_L}$ *and* $\mathrm{u_R}$ *rules; and*
– *the polarized and focused inverse method for the end-sequent* $\Gamma, \alpha(\bar{\Gamma}) \longrightarrow A$ without *using the* $\mathrm{u_L}$ *and* $\mathrm{u_R}$ *rules.*

Proof. It is sufficient to show that for every free atom $\bar{P} \in \bar{\Gamma}$, the synthetic inference rule that arises from focusing on $\alpha(\bar{P})$ corresponds exactly to an instance of the $\mathrm{u_L}$ or $\mathrm{u_R}$ rules. As we assume all atoms have negative polarity, the focusing phase and synthetic inference rules have the following forms for the $\mathrm{u_R}$ rule:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \longrightarrow \psi(P)}\,\exists_{\mathrm{R}}}{\dfrac{\vdots}{\Gamma \longrightarrow \exists \Psi.P}} \qquad \dfrac{\dfrac{}{\psi'(P) \longrightarrow \psi'(P)}\,\mathrm{init}}{\dfrac{\vdots}{\forall \Psi.P \longrightarrow \psi'(P)}\,\forall_{\mathrm{L}}}}{\Gamma, (\exists \Psi.P) \supset (\forall \Psi.P) \longrightarrow \psi'(P)}\,\supset_{\mathrm{L}} \qquad \dfrac{\dfrac{\vdots}{\Gamma \longrightarrow \psi(P)}}{\Gamma, (\exists \Psi.P) \supset (\forall \Psi.P) \longrightarrow \psi'(P)}\,\{\Psi\}$$

where ψ and ψ' are substitutions that instantiate all variables in Ψ. This exactly matches the $\mathrm{u_R}$ rule. The case for the $\mathrm{u_L}$ rule is similar. □

Despite this rather pleasing equivalence, there is a major implementation hurdle: if the $\alpha(\bar{\Gamma})$ assumptions were in the end-sequent, then they would be *globalized*, meaning that they would be deleted in the specialized left rules for these hypotheses. This is always sound because the assumptions in the end-sequent may be assumed to be implicilty present in every forward sequent, so the deletion is just a variant of the factor rule. However, this would mean that the only way to access the corresponding instances of $\mathrm{u_L}$ and $\mathrm{u_R}$, which is necessary for the refinement procedure, would be to keep the full derivations around during search. This is an insurmountable cost in the forward direction because of the memory pressure caused by conjunctive non-determinism. Indeed, it even induces a significant time overhead for the search loop as every forward inference requires copying the full premise. (Recall that every lifted forward sequent stands for all its instances, and it may well be that incompatible instances of the same sequent are needed in different parts of a derivation.)

Our approach to this is *not* to globalize the $\alpha(\bar{\Gamma})$ assumption, but to instead keep them around in the constructed sequents. It is then immediately obvious when a sequent is unsound: since forward sequents have only relevant hypotheses, we merely have to see that a $\alpha(\bar{P})$ (for some P) occurs among them. However, this induces two new kinks: (1) different selections of unsound assumptions may counteract subsumption, even though the corresponding derivations with $\mathrm{u_L}$ and $\mathrm{u_R}$ would be fine, and (2) if the instantiation terms were kept in addition to the unsound assumptions, then they may violate the eigenvariable check for the $\forall_{\mathrm{R}}$ and $\exists_{\mathrm{L}}$ rules. To solve this, we use a more relaxed subsumption relation.

Definition 26 (Relaxed Subsumption). *The sequent* $\Gamma_0, \alpha(\bar{\Gamma}'_0) \longrightarrow \gamma_0$ subsumes $\Gamma_1, \alpha(\bar{\Gamma}'_1) \longrightarrow \gamma_1$ *iff* $\Gamma_0 \longrightarrow \gamma_0$ *subsumes* $\Gamma_1 \longrightarrow \gamma_1$.

ILTP Status	#Problems	Refuted <1s	Refuted <60s	Timeout
Open	93	20	0	73
Non-theorem	56	48	0	8

Table 1. Results of testing Mætning on non-propositional, non-equality SYN problems from the ILTP.

It is clear that this relation coincides with ordinary subsumption on sound sequents, *i.e.*, sequents that contain no instances of $\alpha(\bar{P})$. It is also equivalent to the globalized version, which would always have deleted the unsound hypotheses, so by Theorems 25 and 15 it retains its termination properties.

As mentioned above, keeping entire proofs around is quite costly. On the other hand, we would like the prover to be able to construct an actual sequent calculus proof if it manages to derive a sequent that subsumes the goal. To facilitate this, we instead store a *proof skeleton* for each derived sequent. This skeleton keeps track of which rules appear in the corresponding proof tree, and in what order, but not what the principal formulas were.

To ensure soundness, we reconstruct the full proof based on the proof skeleton by using it to direct a simple backtracking search. This is done using an OCaml implementation of the *Foundational Proof Certificates* [7] approach.

We have implemented a polarized and focused inverse method prover with support the relaxed subsumption (and hence free atoms) described above. This prover, called Mætning, is available from the following URL.

`https://github.com/chaudhuri/maetning`

From the ILTP [15], we tested the prover on every problem marked "Open" or "Non-theorem" from the SYN category which was not propositional, and did not contain any equality. A timeout of 1 minute was used. The results can be seen in Table 1. It should be noted that the vast majority of these are already refutable without the addition of cofinite covers, hence an inverse method prover such as Imogen should succeed in refuting them as well. On the other hand, even simple examples such as the one in the introduction fail to be refutable by Imogen. In short, anything Imogen refutes is also refuted by Mætning, but the converse is not true. The results may be found at the aforementioned URL.

8 Related work

Our approach is similar in many respects to that of Lynch *et al.* [11, 3]. In [3], the authors present a combination of a superposition-based system and SMT solver that uses so-called "speculative inferences" to keep the search space finite, at a possible loss of soundness. One major difference is that our approach can be straightforwardly applied to *nonclassical* logics that can be implemented in the inverse method. Moreover, if our method terminates with an unsound proof, this unsoundness can be used to automatically and intelligently refine the proof search to ensure that the same proof isn't discovered again.

Another similar approach is McCune's MACE [12]. Given a set of first-order formulas, MACE attempts to find a model satisfying these formulas by an exhaustive search of all ground instances for a given domain size. If no model is found, the domain size is increased, yielding an iterative deepening algorithm. However, the deepening process is "blind" in the sense that a failure at a certain depth is not itself informative.

The closest related work to the present paper is on *dynamic polarity assignment* [5] in the inverse method. The main goal of that paper is similar – to use the inverse method to perform proof search in such a way that forward reasoning is guaranteed to terminate. There are, however, substantial differences: the input in [5] is a collection of Horn clauses that is assumed to be both mode-correct and terminating on all well-moded queries. In constrast, our method does not require anything apart from the presence of forward subsumption. On the other hand, any solution found through dynamic polarity assignment is guaranteed to be sound, whereas with our approach, a separate check of soundness is required; furthermore, that procedure runs exactly once, whereas we may need to iterate with refinements.

9 Conclusion and future work

In this paper, we have shown how having *more* hypotheses available can turn a proof search procedure that doesn't necessarily terminate into one that is *guaranteed* to terminate. Furthermore, we have shown that this guarantee can be achieved without any changes to the core theorem proving procedure. Although our primary focus in this paper was intuitionistic first-order logic, we expect that it should be possible to generalize the results of this paper to many other logics with first-order quantification. The method we have presented is *simple*, relying on the use of forward subsumption to cull the search space down to a finite subset of derivable sequents.

Currently, the implementation does not provide a certificate witnessing the nonprovability of a given goal. For refutations, one could in principle simply output the final, saturated database of sequents. Checking the validity of this refutation would then consist of running a simplified version of the Otter loop, checking that in no case is it possible to derive a sequent that is not immediately subsumed. Such a certificate could be quite big, however, and it might therefore be useful to investigate whether the database can be used to construct other witnesses of non-provability, for instance Kripke countermodels.

Ultimately, the power of this approach stems from the use of subsumption in the inverse method. It would be interesting to see if a similar approach works for the more traditional top-down proof search as well.

Acknowledgements We thank Sean McLaughlin for useful discussions on the inverse method.The first author would also like to thank Cody Roux for inspiring him to revisit this line of research. This work has been partially funded by the ERC Advanced Grant *ProofCert*.

References

1. L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. of Logic and Computation*, 3(4), 1994.
2. L. Bachmair and H. Ganzinger. Resolution theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. Elsevier Science, New York, 2001.
3. M. P. Bonacina, C. Lynch, and L. M. de Moura. On deciding satisfiability by theorem proving with speculative inferences. *J. of Automated Reasoning*, 47(2):161–189, 2011.
4. K. Chaudhuri. *The Focused Inverse Method for Linear Logic*. PhD thesis, Carnegie Mellon University, Dec. 2006. Technical report CMU-CS-06-162.
5. K. Chaudhuri. Magically constraining the inverse method using dynamic polarity assignment. In C. Fermüller and A. Voronkov, editors, *Proc. 17th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 6397 of *Lecture Notes in Computer Science*, pages 202–216, Yogyakarta, Indonesia, Oct. 2010. Springer.
6. K. Chaudhuri, F. Pfenning, and G. Price. A logical characterization of forward and backward chaining in the inverse method. *J. of Automated Reasoning*, 40(2-3):133–177, Mar. 2008.
7. Z. Chihani, D. Miller, and F. Renaud. Foundational proof certificates in first-order logic. In M. P. Bonacina, editor, *CADE 24: Conference on Automated Deduction 2013*, number 7898 in Lecture Notes in Artificial Intelligence, pages 162–177, 2013.
8. K. Claessen and N. Sorensson. New techniques that improve MACE-style finite model finding. In P. Baumgartner and C. Fermueller, editors, *Proceedings of the CADE-19 Workshop: Model Computation - Principles, Algorithms, Applications*, Miami, USA, 2003.
9. A. Degtyarev and A. Voronkov. The inverse method. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 179–272. Elsevier and MIT Press, 2001.
10. C. Liang and D. Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, 410(46):4747–4768, 2009.
11. C. Lynch. Unsound theorem proving. In J. Marcinkowski and A. Tarlecki, editors, *Procedings of the 13th Annual EACS Conference on Computer Science Logic*, volume 3210 of *Lecture Notes in Computer Science*, pages 473–487. Springer, 2004.
12. W. McCune. Mace4 reference manual and guide. Technical Report cs.SC/0310055, 2003.
13. S. McLaughlin and F. Pfenning. Imogen: Focusing the polarized focused inverse method for intuitionistic propositional logic. In I. Cervesato, H. Veith, and A. Voronkov, editors, *15th International Conference on Logic, Programming, Artificial Intelligence and Reasoning (LPAR)*, volume 5330 of *Lecture Notes in Computer Science*, pages 174–181, Nov. 2008.
14. S. McLaughlin and F. Pfenning. Efficient intuitionistic theorem proving with the polarized inverse method. In R. A. Schmidt, editor, *Proceedings of the 22nd International Conference on Automated Deduction*, volume 5663 of *Lecture Notes in Computer Science*, pages 230–244. Springer, 2009.
15. T. Raths, J. Otten, and C. Kreitz. The ILTP problem library for intuitionistic logic. *Journal of Automated Reasoning*, 38(1):261–271, 2007.