Imogen: Focusing the Polarized Inverse Method for Intuitionistic Propositional Logic

Sean McLaughlin and Frank Pfenning

Department of Computer Science
Carnegie Mellon University

Abstract. In this paper we describe Imogen, a theorem prover for intuitionistic propositional logic using the focused inverse method. We represent fine-grained control of the search behavior by *polarizing* the input formula. In manipulating the polarity of atoms and subformulas, we can often improve the search time by several orders of magnitude. We tested our method against seven other systems on the propositional fragment of the ILTP library. We found that our prover outperforms all other provers on a substantial subset of the library.

1 Introduction

Imogen is a theorem prover for intuitionistic propositional logic (IPL) based on a focused inverse method with explicit polarities. The *inverse method* [15, 7] uses forward saturation, generalizing resolution to non-classical logics. Focusing [1, 14] reduces the search space in a sequent calculus by restricting the application of inference rules based on the *polarities* of the connectives and atomic formulas. One of the novel aspects of Imogen is that it exploits inherent flexibility in the assignment of polarities to subformulas to optimize proof search. Different assignments of polarities can yield dramatically different performance.

Raths and Otten [18] compare seven systems on the ILTP library [19], a collection of challenge problems for intuitionistic logic provers. In contrast to Imogen, all these use backward search in a contraction-free sequent calculus. This difference in basic approach is reflected in a unique performance profile. Imogen clearly outperforms the other provers on an interesting subset of the benchmark problems, with a tendency to do better on non-theorems. Some problems that appear difficult for backward search are solved almost instantaneously by Imogen, and vice versa. We therefore consider Imogen an interesting and viable alternative for intuitionistic theorem proving.

In this system description we give an overview of the basic principles underlying Imogen, its implementation, and analyze its performance compared to other provers for IPL. The theoretical foundations for Imogen are mostly contained in published papers cited in this description; we therefore do not explicitly state or prove any metatheorems. The source code for Imogen is available at `http://www.cs.cmu.edu/~seanmcl/Imogen`.

2 The Polarized Inverse Method

In this section we sketch the main principles underlying Imogen and their interaction: focusing, polarization, and the inverse method.

2.1 Focusing

Focusing is a method to restrict the space of possible proofs in a cut-free sequent calculus without affecting provability. It was originally developed for backward search in classical linear logic [1], but has been applied to other non-classical logics [11, 14] as well as forward search [5].

Focusing is based on two observations about the properties of connectives. The first is that certain connectives can always be eagerly decomposed during backward proof search without losing completeness. For example, the goal of proving $A \supset B$ can always be decomposed to proving B under additional assumption A. Such connectives are said to have *negative polarity*. As long as the top-level connective stays negative, we can continue the decomposition eagerly without considering any other possibilities. In contrast, for a formula such as $A \vee B$, we have to make a choice whether to try to prove A or B. Such connectives are said to have *positive polarity*. Surprisingly, as long as the top-level connective stays positive, we can continue the decomposition eagerly, making a choice at each step. Moreover, we can arbitrarily assign positive or negative polarity to atomic formulas and restrict the use of atoms in initial sequents.

Proofs that satisfy all three restrictions are called *focused*. Imogen restricts its forward search to *focused proofs*, in a manner explained in the next two sections, drastically reducing its search space when compared to the usual sequent calculus.

2.2 Polarized Formulas

In linear logic, the polarity of each connective is uniquely determined. This is not true for intuitionistic logic where conjunction and truth are inherently ambiguous. We therefore assign polarities to formulas in a preprocessing phase. It is convenient to represent the result as a *polarized formula* [12] where immediately nested formulas always have the same polarity, unless an explicit polarity-shifting connective $\uparrow$ or $\downarrow$ is encountered. These coercions are called *shifts*.

Implication has slightly special status, in that its left-hand side has opposite polarity from its right-hand side. This is because in the sequent calculus for intuitionistic logic, the focusing behavior of connectives on the left-hand side is the opposite of their behavior on the right-hand side. (Here the meta-variable P ranges over atomic propositions.)

$$\text{Positive formulas } A^+ ::= P^+ \mid A^+ \vee A^+ \mid \bot \mid A^+ \wedge A^+ \mid \top \mid \downarrow A^-$$

$$\text{Negative formulas } A^- ::= P^- \mid A^+ \supset A^- \mid A^- \mathbin{\bar{\wedge}} A^- \mid \bar{\top} \mid \uparrow A^+$$

The translation A^- of an (unpolarized) formula F in IPL is nondeterministic, subject only to the constraint that the translation $|A^-| = F$.

$$
\begin{array}{lll}
|A^+ \vee B^+| = |A^+| \vee |B^+| & |\bot| = \bot & |P^+| = P \\
|A^+ \wedge B^+| = |A^+| \wedge |B^+| & |\top| = \top & |{\downarrow}A^-| = |A^-| \\
|A^- \mathbin{\overline{\wedge}} B^-| = |A^-| \wedge |B^-| & |\overline{\top}| = \top & |P^-| = P \\
|A^+ \supset B^-| = |A^+| \supset |B^-| & |{\uparrow}A^+| = |A^+| &
\end{array}
$$

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

$$
\begin{array}{l}
(({\downarrow}A^- \vee {\downarrow}C^-) \wedge {\downarrow}({\downarrow}B^- \supset C^-)) \supset ({\downarrow}({\downarrow}A^- \supset B^-) \supset C^-) \\
{\downarrow}{\uparrow}(({\downarrow}A^- \vee {\downarrow}C^-) \wedge {\downarrow}({\downarrow}B^- \supset C^-)) \supset ({\downarrow}{\uparrow}{\downarrow}({\downarrow}A^- \supset B^-) \supset C^-) \\
{\downarrow}({\uparrow}(A^+ \vee C^+) \mathbin{\overline{\wedge}} (B^+ \supset {\uparrow}C^+)) \supset ({\downarrow}(A^+ \supset {\uparrow}B^+) \supset {\uparrow}C^+)
\end{array}
$$

Shift operators have highest binding precedence in our presentation of the examples. As we will see, the choice of translation determines the search behavior on the resulting polarized formula. Different choices can lead to search spaces with radically different structure [6].

2.3 From Focused Proofs to Big-Step Inferences

A sequent of intuitionistic logic has the form $\Gamma \Longrightarrow A$, where Γ is a set or multiset of formulas. For purposes of Imogen it is convenient to always maintain Γ as a set, without duplicates. Since we can always eagerly decompose negative connectives on the right of a sequent and positive connectives on the left, the only sequents in our polarized calculus we need to consider have negative formulas on the left or positive formulas on the right, in addition to atoms which can appear with either polarity on either side. The right-hand side could also be empty if we are deriving a contradiction. We call such sequents *stable*.

$$
\begin{array}{ll}
\text{Stable Hypotheses} & \Gamma ::= \cdot \mid \Gamma, A^- \mid \Gamma, P^+ \\
\text{Stable Conclusions} & \gamma ::= A^+ \mid P^- \mid \cdot \\
\text{Stable Sequents} & \Gamma \Longrightarrow \gamma
\end{array}
$$

We exploit focusing on polarized formulas to derive big-step rules that go from stable sequents as premises to stable sequents as conclusions. Completeness of focusing tells us that these derived rules, by themselves, are sufficient to prove all valid stable sequents. Rather than formally specify this rule generation (see, for example, Andreoli [2] for the linear case), we only illustrate the process, continuing with the example above.

$$(({\downarrow}A^- \vee {\downarrow}C^-) \wedge {\downarrow}({\downarrow}B^- \supset C^-)) \supset ({\downarrow}({\downarrow}A^- \supset B^-) \supset C^-)$$

The input formula is always translated to a negative formula, which we break down to a set of stable sequents by applying invertible rules. Here we obtain the two sequents

$$
\begin{array}{l}
A, {\downarrow}B \supset C, {\downarrow}A \supset B \Longrightarrow C \\
C, {\downarrow}B \supset C, {\downarrow}A \supset B \Longrightarrow C
\end{array}
$$

We search for proofs of these two stable sequents independently. For each stable sequent, we focus on each constituent formula in turn, and decompose it until we reach all stable sequents as premises. Each possibility yields a new big-step inference rule. We continue to analyze its premises recursively in the same manner. As an example, we show the process for the first goal above.

Focusing on A yields the initial sequent $A \Longrightarrow A$. Focusing on $\downarrow B \supset C$ and $\downarrow A \supset B$ yield the big-step rules

$$\frac{\Gamma \Longrightarrow B}{\Gamma, \downarrow B \supset C \Longrightarrow C} \qquad \frac{\Gamma \Longrightarrow A}{\Gamma, \downarrow A \supset B \Longrightarrow B}$$

2.4 The Inverse Method with Big-Step Rules

The usual (small-step) inverse method applies sequent calculus rules in the forward direction so that each derived formula is a subformula of the original goal. The subformula property is already built into the generation of the rules, so all we need to do now is to apply the big-step rules to saturation in the forward direction. To start the process, each derived rule with no premises is considered as an initial sequent.

To prove the first stable sequent in our example, we begin with the initial sequent $A \Longrightarrow A$. We only have two inference rules, of which only the second applies. The application of this rule derives the new fact $A, \downarrow A \supset B \Longrightarrow B$. Once again, we have only one choice: applying the first rule to this new sequent. The application yields $A, \downarrow A \supset B, \downarrow B \supset C \Longrightarrow C$ which is our goal.

In general, forward inference may only generate a strengthened form of the goal sequent, so we need check if any derived sequents subsume the goal. $\Gamma \Longrightarrow \gamma$ *subsumes* $\Gamma' \Longrightarrow \gamma'$ if $\Gamma \subseteq \Gamma'$ and $\gamma \subseteq \gamma'$. The inference process *saturates* if any new sequent we can derive is already subsumed by a previously derived sequent. If none of these subsume the goal sequent, the goal is not provable and we explicitly fail. In this case, the saturated database may be considered a kind of countermodel for the goal sequent. If the goal sequent is found, Imogen can reconstruct a natural deduction proof term as a witness to the formula's validity.

3 Optimizations and Heuristics

A problem with focusing becomes apparent when considering formulas such as

$$A = (A_1 \vee B_1) \wedge (A_2 \vee B_2) \wedge \cdots \wedge (A_n \vee B_n)$$

Focusing on A on the right will produce 2^n inference rules. Inverting F on the left will produce a single rule with 2^n premises. To avoid exponential behavior such as this, we can change the polarities of the subformulas by adding *double shifts*, $\downarrow\uparrow$ and $\uparrow\downarrow$:

$$A' = \downarrow\uparrow(A_1 \vee B_1) \wedge \downarrow\uparrow(A_2 \vee B_2) \wedge \cdots \wedge \downarrow\uparrow(A_n \vee B_n).$$

The double shifts break the focusing and inversion phases respectively, leading to a linear number of rules and premises at the expense of an increased number of inverse

method deductions. In the extreme, if we insert double shifts before every subformula, we can emulate the inverse method for the ordinary sequent calculus. Imogen currently uses heuristics to insert double shifts for avoiding an exponential explosion.

Imogen first translates to polarized form by a simple method that inserts the fewest shifts, making the choice of conjunction accordingly. Using additional heuristics, Imogen may modify this decision, adding shifts and swapping conjunction and atom polarities to improve the search behavior. Sometimes this leads to a very different search space for problems that are syntactically very similar[1]. Roughly, we count the number of rules and premises that will result from focusing. If such numbers are very large with respect to the input formula, we insert double shifts at a subformula of the goal that is causing a part of the explosion and check the number of rules and premises on the new formula. We continue in this way until a "reasonable" number of premises and rules is reached.

We maintain the hypotheses as sets. Thus, contraction is handled automatically by the application of multi-premise rules.

Formulas which appear in a stable goal sequent will appear in *every* sequent which backward search could construct and are therefore redundant. We omit such *global* assumptions from all sequents. Another helpful optimization is *backward subsumption*. When a new sequent is derived, we remove all sequents that it subsumes from the database. These effects are quantified in section 5.

4 Inference Engine

Imogen's saturation algorithm is based on the Otter loop [16]. It maintains Otter's two distinct databases for *active* sequents[2], those sequents that have had all inference rules applied to them, and *kept* sequents that have not yet been considered for inferences. New rules are generated when a multiple premise rule is matched against an active sequent. This method of matching multi-premise rules incrementally is called *partially applied rule generation*.

The algorithm proceeds as follows. It first polarizes the input formula and runs an initial stabilization pass to determine the stable sequents to prove. The initial sequents and derived rules are then generated using focusing. As an optimization, subformulas are given unique labels to allow fast formula comparison. The final step before search is to initialize the kept sequent database with the initial sequents.

At this stage, Imogen begins the forward search. It selects a kept sequent based on some fair strategy. The sequent is matched against the first premise of all current rules. The matching process will produce new sequents that are put into the kept database, as well as new partially applied rules. The new rules are recursively matched against the active database, and the resulting sequents are put into the kept database. This process repeats until either the kept database becomes empty, in which case the search space is saturated and the formula is invalid, or until the goal sequent is subsumed by a derived sequent.

[1] This effect can be seen in the erratic results of problem class SYJ206 in section 5.

[2] sometimes called the "set of support"

5 Evaluation

We evaluated our prover on the propositional fragment of the ILTP [19, version 1.1.2] library of problems for intuitionistic theorem provers. The 274 problems are divided into 12 families of difficult problems such as the pigeonhole principle, labeled SYJ201 to SYJ212. For each family, there are 20 instances of increasing size. There are also 34 miscellaneous problems. The provers that are currently evaluated are ft-C [20, version 1.23], ft-Prolog [20, version 1.23], LJT [8], PITP [3, version 3.0], PITPINV [3, version 3.0], and STRIP [13, version 1.1]. These provers represent a number of different methods of theorem proving in IPL, yet forward reasoning is conspicuously absent. Imogen solved 261 of the problems. PITPINV was the only prover to solve more. Some illustrative examples of difficult problems are shown in the following table:

Prover	**ft-Prolog**	**ft-C**	**LJT**	**PITP**	**PITPINV**	**IPTP**	**STRIP**	**Imogen**
Solved (out of 274)	188	199	175	238	262	209	205	261
SYN007+1.014	-0.01	-0.01	stack	large	large	large	alloc	-0.1
SYJ201+1.018	0.28	0.04	0.4	0.01	0.01	2.31	0.23	25.5
SYJ201+1.019	0.36	0.04	0.47	0.01	0.01	2.82	0.32	28.0
SYJ201+1.020	0.37	0.05	0.55	0.01	0.01	3.47	0.34	28.35
SYJ202+1.007	516.55	76.3	memory	0.34	0.31	13.38	268.59	64.6
SYJ202+1.008	time	time	memory	3.85	3.47	97.33	time	time
SYJ202+1.009	time	time	memory	50.25	42.68	time	time	time
SYJ202+1.010	time	time	memory	time	time	time	time	time
SYJ205+1.018	time	time	0.01	0.01	7.49	0.09	time	0.01
SYJ205+1.019	time	time	0.01	0.01	15.89	0.09	time	0.01
SYJ205+1.020	time	time	0.01	0.01	33.45	0.1	time	0.01
SYJ206+1.018	time	time	memory	1.01	0.96	9.01	8.18	56.2
SYJ206+1.019	time	time	memory	1.95	1.93	18.22	14.58	394.14
SYJ206+1.020	time	time	memory	3.92	3.89	36.35	33.24	42.7
SYJ207+1.018	time	time	time	time	-68.71	time	time	-42.6
SYJ207+1.019	time	time	time	time	-145.85	time	time	-63.6
SYJ207+1.020	time	time	time	time	-305.21	time	time	-97.25
SYJ208+1.018	time	time	memory	-0.99	-0.95	time	time	-184.14
SYJ208+1.019	time	time	memory	-1.36	-1.35	memory	mem	-314.31
SYJ208+1.020	time	time	memory	-1.76	-1.80	memory	mem	-506.02
SYJ209+1.018	time	time	time	time	-13.44	time	time	-0.01
SYJ209+1.019	time	time	time	time	-28.68	time	time	-0.01
SYJ209+1.020	time	time	time	time	-60.54	time	time	-0.02
SYJ211+1.018	time	time	time	-43.65	-31.51	time	time	-0.02
SYJ211+1.019	time	time	time	-91.75	-66.58	time	time	-0.02
SYJ211+1.020	time	time	time	-191.57	-139.67	time	time	-0.02
SYJ212+1.018	-0.01	-0.01	memory	-1.31	-1.37	time	-8.5	-0.02
SYJ212+1.019	-0.01	-0.01	memory	-2.7	-2.75	time	-17.41	-0.03
SYJ212+1.020	-0.01	-0.01	memory	-5.51	-5.51	time	-38.94	-0.04

The table uses the notation of [18]. All times are in seconds. The entry "memory" indicates that the prover process ran out of memory. A "time" entry indicates that the prover was unable to solve the problem within the ten minute time limit. A negative number indicates the time to ascertain that a formula is *not* valid. All statistics except

for those of Imogen were executed on a 3.4 GHz Xeon processor running Linux [18]. The Imogen statistics are a 2.4 GHz Intel Core 2 Duo on Mac OS X. Thus the Imogen statistics are conservative.

6 Conclusion

The most closely related system to Imogen is Linprover [4], which is an inverse method prover for intuitionistic linear logic exploiting focusing, but not polarization. We do not explicitly compare our results to Linprover, which incurs additional overhead due to the necessary maintenance of linearity constraints. We are also aware of two provers for first-order intuitionistic logic based on the inverse method, Gandalf [21] and Sandstorm [9], both of which partially exploit focusing. We do not compare Imogen to these either, since they incur substantial overhead due to unification, contraction, and more complex subsumption.

Imogen could be improved in a number of ways. Selecting sequents from the kept database for rule application could be improved by a more intelligent ordering of formulas. Better heuristics for assigning polarities to subformulas, especially atoms, seem to offer the biggest source of performance gains. Experimenting with double shifts and atom polarities by hand greatly increased performance, but as yet we have no sophisticated methods for determining more optimal assignments of polarities.

We implemented Imogen with the eventual goal to generalize the system to first-order intuitionistic logic and logical frameworks and were somewhat surprised that even a relatively straightforward implementation in a high-level language is not only competitive with previous provers based on backward search, but clearly better on a significant portion of accepted benchmark problems. We plan to begin experiments using the polarized inverse method for LF [10] and M2, the metalogic of Twelf [17].

One of Imogen's strengths is its ability to do redundancy elimination. The databases can grow large, making deriving further inferences slower. Yet when a strong sequent is derived, it is not uncommon for half or more of the database to be subsumed and eliminated with backward subsumption, thus allowing Imogen to continue making deductions at a much higher rate. We believe that this will be important in solving difficult problems in more complex logics.

Our experience with Imogen, in addition to the evidence provided by the provers cited above, adds strength to our thesis that the polarized inverse method works well on non-classical logics of different kinds.

Acknowledgments

We have been generously supported in this work by NSF grant CCR-0325808. We would like to thank Kaustuv Chaudhuri for helpful discussions regarding Imogen. We also thank the anonymous reviewers for helpful suggestions and correcting some minor errors.

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. A. Avellone, G. Fiorino, and U. Moscato. A new $o(n\ log_n)$-space decision procedure for propositional intuitionistic logic. In *Kurt Goedel Society Collegium Logicum*, volume VIII, pages 17–33, 2004.
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 and F. Pfenning. Focusing the inverse method for linear logic. In L. Ong, editor, *Computer Science Logic*, pages 200–215. Springer, 2005.
6. K. Chaudhuri, F. Pfenning, and G. Price. A logical characterization of forward and backward chaining in the inverse method. In U. Furbach and N. Shankar, editors, *International Joint Conference on Automated Reasoning*, pages 97–111. Springer, 2006.
7. 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.
8. R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 57(3):795–807, 1992.
9. D. Garg, T. Murphy, G. Price, J. Reed, and N. Zeilberger. Team red: The sandstorm theorem prover. `http://www.cs.cmu.edu/~tom7/papers/`.
10. R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.
11. J. M. Howe. *Proof Search Issues in Some Non-Classical Logics*. PhD thesis, University of St. Andrews, Scotland, 1998.
12. F. Lamarche. Games semantics for full propositional linear logic. In *Proceedings of the 10th Annual Symposium on Logic in Computer Science (LICS'95)*, pages 464–473, San Diego, California, June 1995. IEEE Computer Society.
13. D. Larchey-Wendling, D. Méry, and D. Galmiche. STRIP: Structural sharing for efficient proof-search. In R. Goré, A. Leitsch, and T. Nipkow, editors, *International Joint Conference on Automated Reasoning*, pages 696–700. Springer, 2001.
14. 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.
15. S. Y. Maslov. An inverse method for establishing deducibility in classical predicate calculus. *Doklady Akademii nauk SSSR*, 159:17–20, 1964.
16. W. W. McCune. OTTER 3.0 reference manual and guide. Technical Report ANL-94/6, Argonne National Laboratory/IL, USA, 1994.
17. F. Pfenning and C. Schürmann. System description: Twelf : A meta-logical framework for deductive systems. In H. Ganzinger, editor, *Conference on Automated Deduction*, pages 202–206. Springer-Verlag, 1999.
18. T. Raths and J. Otten. The ILTP Library. `http://www.cs.uni-potsdam.de/ti/iltp/`.
19. T. Raths, J. Otten, and C. Kreitz. The ILTP problem library for intuitionistic logic. *J. Autom. Reasoning*, 38(1-3):261–271, 2007.
20. D. Sahlin, T. Franzén, and S. Haridi. An intuitionistic predicate logic theorem prover. *Journal of Logic and Computation*, 2(5):619–656, Oct. 1992.
21. 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.