Magically Constraining the Inverse Method Using Dynamic Polarity Assignment

Kaustuv Chaudhuri

Magically Constraining the Inverse Method Using Dynamic Polarity Assignment

Kaustuv Chaudhuri

INRIA Saclay, France
kaustuv.chaudhuri@inria.fr

Abstract. Given a logic program that is terminating and mode-correct in an idealized Prolog interpreter (i.e., in a top-down logic programming engine), a bottom-up logic programming engine can be used to compute exactly the same set of answers as the top-down engine for a given mode-correct query by rewriting the program and the query using the Magic Sets Transformation (MST). In previous work, we have shown that focusing can logically characterize the standard notion of bottom-up logic programming if atomic formulas are statically given a certain polarity assignment. In an analogous manner, dynamically assigning polarities can characterize the effect of MST without needing to transform the program or the query. This gives us a new proof of the completeness of MST in purely logical terms, by using the general completeness theorem for focusing. As the dynamic assignment is done in a general logic, the essence of MST can potentially be generalized to larger fragments of logic.

1 Introduction

It is now well established that two operational "dialects" of logic programming—top-down (also known as backward chaining or goal-directed) in the style of Prolog, and bottom-up (or forward chaining or program-directed) in the style of hyperresolution—can be expressed in the uniform lexicon of polarity and focusing in the cut-free sequent calculus for a general logic such as intuitionistic logic [7]. The difference in these diametrically opposite styles of logic programming amounts to a static and global *polarity assignment* to the atomic formulas. Such a *logical characterisation* allows a general theorem proving strategy for the sequent calculus, which might be backward (goal sequent to axioms) as in tableau methods or forward (axioms to goal sequent) like in the inverse method, to implement either forward or backward chaining (or any combination) for logic programs by selecting the polarities for the atoms appropriately. Focused inverse method provers have been built for linear logic [4], intuitionistic logic [15], bunched logic [9] and several modal logics [10] in recent years.

The crucial ingredient for the characterisation is that polarities and focusing are sufficiently general that all static polarity assignments are complete [7,1]. The two assignments may be freely mixed for different atoms, which will produce hybrid strategies. The proofs are very different: in a standard example with Fibonacci numbers, one assignment admits exponentially sized derivations, while the other has only the linear proofs. Even more importantly, the search space for proofs is wildly different for different assignments. Which static assignment to pick is not always obvious and very difficult to perform automatically, as was noted in the experiments in [7,15].

In this paper, we propose to look at *dynamic polarity assignment* as a means to do better than static assignment for certain well-known classes of problems. To our knowledge, dynamic assignment of polarities has been investigated only once before [16]; however, the notion of assignment there is a means of incorporating tables into proof objects using new atomic cut rules with asymmetric assignments to the cut atoms. Our proposal, in contrast, retains the *same* inference rules as ordinary focusing, but dynamically specializes them based on polarity assignments performed at runtime; this lets us reuse the strong proof-theoretic results about focusing. Note that “dynamic polarity assignment” is not a particular algorithm but a general class of approaches for controlling search behaviour. It is useful to think of it by analogy with ordering strategies in resolution theorem proving.

In particular, we give a dynamic assignment strategy that implements the effect of the so-called *magic sets transformation* [3,18,14], which is a program transformation that constrains forward chaining to have the same set of answers as backward chaining. It is difficult to show that the transformation has this intended property. Moreover, since it is a global transformation on the program, that might even (in the general case) depend on the query, it is not modular and compositional. Our proposal reconstructs magic sets and not only avoids the transformation but also characterizes them in the common lexicon of focusing and polarities. That is, the magic sets approach is just a special case of dynamic polarity assignment, in much the same way as forward and backward chaining for Horn clauses are just special cases of static polarity assignment.

We limit our attention in this paper to the focused inverse method [4] as the particular general search strategy for the sequent calculus. Intuitively (but not precisely; see sec. 3), this method “compiles” a clause into an inference rule as follows:

$$\texttt{sum (s X) Y (s Z) :- sum X Y Z.} \quad\longrightarrow\quad \frac{\vdash \mathtt{sum}\ x\ y\ z}{\vdash \mathtt{sum}\ (\mathtt{s}\ x)\ y\ (\mathtt{s}\ z)}$$

When this inference rule is read from premise to conclusion, the interpretation is of forward chaining on the corresponding clause. Such rules can be repeatedly applied to produce an infinite number of new sequents differing only in the number of ss, which prevents *saturation* even for queries with a finite backward chaining search space. With such clauses, forward chaining cannot appeal to *negation by failure*, unlike backward chaining. We show how to use dynamic polarity assignment to instead produce a new side condition on such inference rules: the conclusion ($\mathtt{sum}\ (\mathtt{s}\ x)\ y\ (\mathtt{s}\ z)$) must be negatively polarized for the rule to be applicable. The atoms are polarized negatively by carefully selecting only those atoms that are in the *base* of the logic program.

One important feature of this re-examination of the magic sets approach is that, because it is performed in a more general setting, we can potentially generalize it to larger fragments of logic such as the uniform fragment. As it does not change the underlying proof system, it can potentially co-exist with other strategies. For example, if the dynamic assignment algorithm gets stuck, the remaining atoms can be polarized in some other fashion and the inverse method resumed without losing completeness.

The rest of this paper is organized as follows. In sec. 2 the magic sets transformation is sketched by way of example. Section 3 then summarizes the design of the focused inverse method and static polarity assignment. Section 4 introduces dynamic polarity assignment and shows how to use it to implement the magic sets restriction (sec. 4.2).

Finally, sec. 5 discusses the conclusions and scope of future work on dynamic polarity assignment.

2 Magic Sets Transformation

This section contains a quick overview of the *magic sets transformation* for logic programs. We use the "core" version presented in [14], which is less general than some other designs in the literature [3,18] but also easier to explain and reason about. The logic programs we will consider are made up of Horn clauses and satisfy a global *well-modedness* criterion.

Definition 1 (abstract syntax of Horn clauses) *A* Horn clause *is an iterated implication of atomic formulas that is implicitly universally closed over all its variables. That is, Horn clauses $(C, D, \ldots)$ satisfy the following grammar:*

$$C, D, \ldots ::= a\,\vec{t} \;\big|\; a\,\vec{t} \to C \qquad\qquad t, s, \ldots ::= x \;\big|\; f\,\vec{t}$$

where a ranges over predicate symbols, f over function symbols, and x over variables. The notation $\vec{t}$ stands for a list, possibly empty, of terms.

Note that the clause `a :- b, ..., z` in a Prolog-like concrete syntax would be written as `z` $\to \cdots \to$ `b` $\to$ `a` in the above abstract syntax that is, the order of the clauses in the body is reversed. Many extensions of this definition of Horn clauses exist in the literature, but they are all generally equivalent to this fragment. A *logic program* is an unordered collection of Horn clauses where each predicate and function symbol has a unique arity. (We do not consider particular orderings of the clauses because we are not interested in the operational semantics of a particular logic programming language.)

Definition 2 (moding) *Every predicate symbol of arity n can be assigned a* mode, *which is a string of length n composed of the characters* `i` *and* `o`, *which are mnemonics for "input" and "output" respectively. A mode assignment to all predicates in a logic program is called a* moding. *The* inputs *of a predicate with respect to a mode are those arguments corresponding to the occurrences of* `i` *in the mode; likewise, the* outputs *are the arguments corresponding to* `o` *in the mode.*

Definition 3 (well-modedness) *All the following are with respect to a given moding:*

- *A goal query is* well-moded *iff its inputs are ground.*
- *A clause $a_1\,\vec{t_1} \to \cdots \to a_n\,\vec{t_n} \to b\,\vec{s}$ is* well-moded *iff for all $i \in 1..n$, the variables in the inputs of $a_i\,\vec{t_i}$ are contained in the union of the variables in the outputs of $a_j\,\vec{t_j}$ for $i < j \leq n$ and of the variables in the inputs of $b\,\vec{s}$.*
- *A logic program is* well-moded *iff every clause in it is well-moded.*

The definition of well-modedness for non-unit clauses intuitively states that, in a right-to-left reading of the clause, the inputs of an atomic formula must be defined by the outputs of earlier atomic formulas and the inputs of the head. Given a well-moded program and query, every derivation of an instance of the query from the program will be ground (for the proof, see [2]).

Consider the motivating example from [14]: computing the sum of the elements of a list of natural numbers. The clauses of the program are as follows in Prolog style.

```
(* mode lsum = io *)
lsum [] 0.
lsum (X :: Y) K :- lsum Y J, sum X J K.
(* mode sum = iio *)
sum 0 X X.
sum (s X) Y (s Z) :- sum X Y Z.
```

This program is well-moded because the outputs flow into the inputs from left to right in the body of the clauses. A query such as `?- lsum [1, 2, 3] X` is well-moded because the input is ground, while a query such as `?- lsum X 20` is not well-moded.

To prove a well-moded query, the *backward chaining* or *top-down logic programming* approach matches the goal with the heads of the clauses in the program, and for each successful match, replaces the goal with the matched instance of the body of the clause as new subgoals. A well-moded program is said to be *terminating* if there are no infinite backward chaining derivations for a well-moded query.

The *forward chaining* or *bottom-up logic programming* strategy starts from the unit clauses in the program, matches the body of a clause with these clauses, and adds the most general instance of the matched head as a new clause. This is iterated until (a generalisation of) the goal query is derived. This direction is not quite as obviously goal-directed as backward chaining, but it has many fundamental merits. It builds a database of computed facts that are all mutually non-interfering, and therefore requires no backtracking or global, stateful updates. Moreover, facts and therefore derivations are implicitly shared, so the loop detection issue that plagues backward chaining does not apply here.

However, forward chaining suffers from the obvious problem that it over-approximates the query, performing a lot of wasteful search. Fortunately, it is possible to constrain forward chaining for a given program and query such that the algorithm will *saturate*, *i.e.*, reach a state where no new facts can be generated, iff the query terminates in backward chaining. This is achieved by rewriting the program and the query so that the forward algorithm approximates backward search.

The common element of the approaches to constrain forward chaining is the notion of a *magic set*, which is an abstract representation of the *base* of the program [14]. We shall illustrate it here with the example above. For each predicate a, a new magic predicate a' is added that has the same arity as the input arity of the original predicate. Then, each clause of the program is transformed to depend on the magic predicate applied to the inputs of the head. That is, we obtain the following rewritten clauses:

```
lsum [] 0 :- lsum' [].
lsum (X :: Y) K :- lsum' (X :: Y), lsum Y J, sum X J K.
sum 0 X X :- sum' 0 X.
sum (s X) Y (s Z) :- sum' (s X) Y, sum X Y Z.
```

As there are no longer any unit clauses, forward chaining cannot begin without some additional input. This is provided in the form of the magic version of the goal query as a new unit clause: `lsum' [1, 2, 3]`. Finally, clauses are added for the magic predicates to propagate information about the base. For each non-unit clause, there is one propagation rule for each predicate in the body of the clause. In this example, they are:

```
lsum' Y :- lsum' (X :: Y).
sum' X J :- lsum' (X :: Y), lsum Y J.
sum' X Y :- sum' (s X) Y.
```

Forward chaining on this transformed program will compute the same instances of the query as backward chaining on the original program and query.

The correctness of this *magic sets transformation* is generally quite difficult to prove. One of the most readable proofs was provided by Mascellani *et al* [14]; that paper also contains a fully formal definition of the transformation and a number of other examples. However, all transformational approaches suffer from the same problems outlined in the introduction: they are not modular and compositional. In the rest of the paper we will give a different explanation of the magic sets transformation that does not suffer from these problems, and is moreover manifestly correct because of general proof theoretic properties of focused sequent calculi.

3 The Focused Inverse Method

In this section we review the focused inverse method for intuitionistic logic. Most of the material of this section has already appeared in in [4,7,15,8] and in references therefrom. Like other recent accounts of intuitionistic focusing [15,5], we adopt a polarized syntax for formulas. Intuitively, *positive* formulas (*i.e.*, formulas of the positive *polarity*) are those formulas whose left sequent rules are invertible and *negative* formulas are those whose right rules are invertible. Every polarized logical connective is unambiguously in one of these two classes. In order to prevent an overlap, we also *assign* the atomic formulas individually to one of the two classes. Any polarity assignment for the atoms is complete [7].

Definition 4 (syntax) *We follow this grammar:*

$$P, Q ::= p \mid P \otimes Q \mid \mathbf{1} \mid P \oplus Q \mid \mathbf{0} \mid \exists x.\, P \mid {\downarrow}N \qquad N, M ::= n \mid N \mathbin{\&} M \mid \top \mid P \multimap N \mid \forall x.\, N \mid {\uparrow}P$$
$$p ::= \langle a\, \vec{t}, + \rangle \qquad n ::= \langle a\, \vec{t}, - \rangle \qquad P^- ::= P \mid n \qquad N^+ ::= N \mid p$$

- Formulas *$(A, B, \ldots)$ are either* positive *$(P, Q, \ldots)$ or* negative *$(N, M, \ldots)$.*
- Atomic formulas *(or* atoms*) $(p, q, n, m, \ldots)$ are also polarized. Each atom consists of an atomic predicate $(a, b, \ldots)$ applied to a (possibly empty) list of terms, and a* polarity. *We shall generally abuse notation and write $\langle a\, \vec{t}, \pm \rangle$ as $a^{\pm}\, \vec{t}$, even though it is the atom and not the predicate that carries the polarity.*
- Left passive formulas *$(N^+, M^+, \ldots)$ and* right passive formulas *$(P^-, Q^+, \ldots)$ are used to simplify the presentation of rules.*

We use connectives from polarized linear logic instead of the more usual intuitionistic connectives to make the polarities explicit. The polarity *switching* connectives $\downarrow$ and $\uparrow$ are only bureaucratic and do not change the truth value of their operands. Both $\otimes$ and & have the same truth value as the usual intuitionistic conjunction $\wedge$—that is, $A \otimes B \equiv A \mathbin{\&} B$ if we ignore polarities and omit the switching connectives $\downarrow$ and $\uparrow$—just different inference rules. In other formulations of polarized intuitionistic logic these two polarisations of conjunction are sometimes written as $\wedge^+$ or $\wedge^-$ [13], but we prefer the familiar notation from linear logic. Likewise, $\oplus$ has the same truth value as $\vee$ and $\multimap$ the same truth value as $\rightarrow$.

The inference system for this logic will be given in the form of focused sequent calculus rules [1,15]. We have the following kinds of sequents:

$$\begin{array}{llll}
\Gamma \vdash [P] & \text{right-focus on } P & \Gamma\;;\;[N] \vdash Q^- & \text{left-focus on } N \\
\Gamma\;;\;\Omega \vdash \underbrace{\begin{cases} N\;;\;\cdot \\ \cdot\;;\;Q^- \end{cases}}_{\gamma} & \text{left-active on } \Omega \text{ and right-active on } N & &
\end{array}$$

where: $\Gamma ::= \cdot \;\big|\; \Gamma, N^-$ is called the *passive context* and $\Omega ::= \cdot \;\big|\; \Omega, P$ is the *active context*. Both contexts are interpreted as multisets (admits only exchange). We adopt the usual convention of denoting multiset union with commas. It will turn out that the passive context is also a set, but this is an admissible principle and does not depend on primitive weakening and contraction rules. Note therefore that Γ_1, Γ_2 is not the same as $\Gamma_1 \cup \Gamma_2$; if the latter interpretation is needed, it will be written explicitly.

(active)

$$\dfrac{\Gamma\;;\;\Omega \vdash \cdot\;;\;n}{\Gamma\;;\;\Omega \vdash n\;;\;\cdot}\,\textsc{nr} \qquad \dfrac{\Gamma\;;\;\Omega \vdash \cdot\;;\;P}{\Gamma\;;\;\Omega \vdash {\uparrow}P\;;\;\cdot}\,{\uparrow}\textsc{r} \qquad \dfrac{\Gamma\;;\;\Omega \vdash N\;;\;\cdot \quad \Gamma\;;\;\Omega \vdash M\;;\;\cdot}{\Gamma\;;\;\Omega \vdash N \mathbin{\&} M\;;\;\cdot}\,\&\textsc{r}$$

$$\dfrac{}{\Gamma\;;\;\Omega \vdash \top\;;\;\cdot}\,\top\textsc{r} \qquad \dfrac{\Gamma\;;\;\Omega, P \vdash N\;;\;\cdot}{\Gamma\;;\;\Omega \vdash P \multimap N\;;\;\cdot}\,{\multimap}\textsc{r} \qquad \dfrac{\Gamma\;;\;\Omega \vdash N[a/x]\;;\;\cdot}{\Gamma\;;\;\Omega \vdash \forall x.\, N\;;\;\cdot}\,\forall\textsc{r}^a$$

$$\dfrac{\Gamma, p\,\vec{t}\;;\;\Omega \vdash \gamma}{\Gamma\;;\;\Omega, p\,\vec{t} \vdash \gamma}\,\textsc{pl} \qquad \dfrac{\Gamma, N\;;\;\Omega \vdash \gamma}{\Gamma\;;\;\Omega, {\downarrow}N \vdash \gamma}\,{\downarrow}\textsc{l} \qquad \dfrac{\Gamma\;;\;\Omega, P, Q \vdash \gamma}{\Gamma\;;\;\Omega, P \otimes Q \vdash \gamma}\,\otimes\textsc{l} \qquad \dfrac{\Gamma\;;\;\Omega \vdash \gamma}{\Gamma\;;\;\Omega, \mathbf{1} \vdash \gamma}\,\mathbf{1}\textsc{l}$$

$$\dfrac{\Gamma\;;\;\Omega, P \vdash \gamma \quad \Gamma\;;\;\Omega, Q \vdash \gamma}{\Gamma\;;\;\Omega, P \oplus Q \vdash \gamma}\,\oplus\textsc{l} \qquad \dfrac{}{\Gamma\;;\;\Omega, \mathbf{0} \vdash \gamma}\,\mathbf{0}\textsc{l} \qquad \dfrac{\Gamma\;;\;\Omega, N[a/x] \vdash \gamma}{\Gamma\;;\;\Omega, \exists x.\, N \vdash \gamma}\,\exists\textsc{l}^a$$

(right focus)

$$\dfrac{}{\Gamma, p \vdash [p]}\,\textsc{pr} \qquad \dfrac{\Gamma\;;\;\cdot \vdash N\;;\;\cdot}{\Gamma \vdash [{\downarrow}N]}\,{\downarrow}\textsc{r}$$

$$\dfrac{\Gamma \vdash [P] \quad \Gamma \vdash [Q]}{\Gamma \vdash [P \otimes Q]}\,\otimes\textsc{r} \qquad \dfrac{}{\Gamma \vdash [\mathbf{1}]}\,\mathbf{1}\textsc{r} \qquad \dfrac{\Gamma \vdash [P_i]}{\Gamma \vdash [P_1 \oplus P_2]}\,\oplus\textsc{r}_i \qquad \dfrac{\Gamma\;;\;[P[t/x]]}{\Gamma \vdash [\exists x.\, P]}\,\exists\textsc{r}$$

(left focus)

$$\dfrac{}{\Gamma\;;\;[n] \vdash n}\,\textsc{nl} \qquad \dfrac{\Gamma\;;\;P \vdash \cdot\;;\;Q^-}{\Gamma\;;\;[{\uparrow}P] \vdash Q^-}\,{\uparrow}\textsc{l}$$

$$\dfrac{\Gamma\;;\;[N_i] \vdash Q^-}{\Gamma\;;\;[N_1 \mathbin{\&} N_2] \vdash Q^-}\,\&\textsc{l}_i \qquad \dfrac{\Gamma \vdash [P] \quad \Gamma\;;\;[N] \vdash Q^-}{\Gamma\;;\;[P \multimap N] \vdash Q^-}\,{\multimap}\textsc{l} \qquad \dfrac{\Gamma\;;\;[N[t/x]] \vdash Q^-}{\Gamma\;;\;[\forall x.\, N] \vdash Q^-}\,\forall\textsc{l}$$

(decision)

$$\dfrac{\Gamma \vdash [P]}{\Gamma\;;\;\cdot \vdash \cdot\;;\;P}\,\textsc{dr} \qquad \dfrac{\Gamma, N\;;\;[N] \vdash Q^-}{\Gamma, N\;;\;\cdot \vdash \cdot\;;\;Q^-}\,\textsc{dl}$$

Fig. 1. Focused sequent calculus for polarized first-order intuitionistic logic. In the rules $\exists\textsc{l}^a$ and $\forall\textsc{r}^a$ the *parameter* a is not free in the conclusion.

$$\dfrac{\dfrac{\dfrac{}{\Gamma, N, p \vdash [p]}\,\textsc{pr}}{\Gamma, N, p \vdash [p \oplus q]} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, N, p\,;\,\cdot \vdash \cdot\,;\, l}{\Gamma, N, p\,;\,\cdot \vdash l\,;\,\cdot}}{\Gamma, N, p \vdash [{\downarrow} l]} \quad \dfrac{}{\Gamma, N, p\,;\,[n] \vdash n}\,\textsc{nl}}{\Gamma, N, p\,;\,[{\downarrow} l \multimap n] \vdash n}}{\Gamma, N, p\,;\,[m \,\&\, ({\downarrow} l \multimap n)] \vdash n}\,\&\textsc{l}_2}{\dfrac{\Gamma, N, p\,;\,[N] \vdash n}{\Gamma, N, p\,;\,\cdot \vdash \cdot\,;\, n}\,\textsc{dl}} \qquad \textit{i.e.,} \quad \dfrac{\Gamma, N, p\,;\,\cdot \vdash \cdot\,;\, l}{\Gamma, N, p\,;\,\cdot \vdash \cdot\,;\, n}$$

Fig. 2. One derived inference rule for N.

The focused sequent calculus will be presented in a stylistic variant of Andreoli's original formulation [1]. The full set of rules is in fig. 1. It has an intensional reading in terms of *phases*. At the boundaries of phases are sequents of the form $\Gamma\,;\,\cdot \vdash \cdot\,;\, Q^-$, which are known as *neutral sequents*. Proofs of neutral sequents proceed (reading from conclusion to premises) as follows:

1. *Decision*: a *focus* is selected from a neutral sequent, either from the passive context or from the right. This focused formula is moved to its corresponding focused zone using one of the rules DL or DR (D = "decision", and R/L = "right"/"left"). The left rule copies the focused formula.
2. *Focused phase*: for a left or a right focused sequent, left or right focus rules are applied to the formula under focus. These focused rules are all non-invertible in the (unfocused) sequent calculus—that is, they can fail to apply—and therefore depend on essential choices made in the proof. This is familiar from focusing for linear logic [1,7].
3. *Active phase*: once the switch rules ${\downarrow}$R and ${\uparrow}$L are applied, the sequents become active and active rules are applied. The order of the active rules is immaterial as all orderings will produce the same list of neutral sequent premises. In Andreoli's system the irrelevant non-determinism in the order of these rules was removed by treating the active context Ω as ordered; however, we do not fix any particular ordering.

The soundness of this calculus with respect to an unfocused sequent calculus, such as Gentzen's LJ, is obvious. For completeness, we refer the interested reader to a number of published proofs in the literature [7,12,17,11].

The purpose of starting with a polarized syntax and a focused calculus is that we are able to look at derived inference rules for neutral sequents as the basic unit of *steps*. For instance, one of the derived inference rules for the formula $N \triangleq p \oplus q \multimap m \,\&\, ({\downarrow} l \multimap n)$ in the passive context is given in fig. 2. The instance of PR above forces p to be in the passive context because that is the only rule that can be applied to contruct a sequent of the form $\Delta \vdash [p]$. Likewise, the NL rule forces the right hand side of the conclusion sequent to be the same as the left focused atom n. Finally, the DL rule requires N to already be present in the passive context.

As we observe, focusing *compiles* formulas such as N above, which may be clauses in a program, into (derived) inference rules. Focusing can also produce new *facts*, which are neutral sequents that have no open premises after applying a derived inference rule. An example would be the case for the derivation above where, instead of $\&\textsc{l}_2$ we were to use $\&\textsc{l}_1$. In this case we would obtain the fact $\Gamma, N, p \,;\, \cdot \vdash \cdot \,;\, m$. If the goal were of this form, we would be done.

This property of focusing can be exploited to give a purely proof-theoretic explanation for certain *dialects* of proofs. For Horn clauses, consider the case where all the atoms are negative, *i.e.*clauses are of the form $\forall \vec{x}\,.\, \downarrow m_1 \multimap \cdots \multimap \downarrow m_j \multimap n$. If clause were named N, then its derived inference rule is:

$$\frac{\Gamma, N \,;\, \cdot \vdash \cdot \,;\, m_1[\vec{t}/\vec{x}\,] \quad \cdots \quad \Gamma, N \,;\, \cdot \vdash \cdot \,;\, m_j[\vec{t}/\vec{x}\,]}{\Gamma, N \,;\, \cdot \vdash \cdot \,;\, n[\vec{t}/\vec{x}\,]}$$

Since the context is the same in all premises and the conclusion, we need only look at the right hand side. If we read the rule from conclusion to premises, then this rule implements back-chaining from an instance of the head of this Horn clause to the corresponding instances of the body of the clause, where the neutral sequents represent the current list of sub-goals. Thus, the general top-down logic programming strategy (or backward chaining) consists of performing goal-directed focused proof search on Horn clauses with negative atoms. If the atoms were all assigned positive polarity instead, then the same goal-directed focused proof search would perform a kind of bottom-up logic programming (or forward chaining). Static polarity assignment for the atoms is therefore a *logical characterisation* of forward and backward chaining strategies. Indeed, if the atoms were not uniformly given the same polarities, then the focused proofs would be a mixture of forward and backward chaining.

3.1 Forward reasoning and the inverse method

An important property of the (cut-free) sequent calculus of fig. 1 is that there is a structural cut-elimination algorithm [7]; as a consequence, the calculus enjoys the subformula property. Indeed, it is possible to state the subformula property in a very strong form that also respects the *sign* of the subformula (*i.e.*, whether it is principal on the left or the right of the sequent) and the *parametricity* of instances (*i.e.*, the subformulas of a right $\forall$ or a left $\exists$ can be restricted to generic instances). We omit a detailed definition and proof here because it is a standard result; see *e.g.* [6] for the definition.

With the strong subformula property, we can restrict the rules of fig. 1 to subformulas of a given fixed *goal sequent*. It then becomes possibile to apply the inference rules in a forward manner, from premises to conclusion. The inputs of such a forward reasoning strategy would be the facts that correspond to focusing on the passive formulas and operands of the switch connectives in the goal sequent, subject to the subformula restriction. That is, the initial sequents (in the rules PR and NL) correspond to atomic formulas that are both a left and a right signed subformula of the goal sequent. From these initial sequents we apply the (subformula-restricted) inference rules forward until we derive (a generalisation of) the goal sequent. In order to implement the calculus, the axiomatic rules such as $\mathbf{1}\textsc{r}$ are refined to omit the passive context; the additive rules are turned into multiplicative rules and an explicit rule of *factoring* added; and the calculus is lifted to free variables with identity replaced with unifiability, and only most general instances are considered. This core "recipe" is outlined in the *Handbook* article on the inverse method [8] and is not repeated here.

One optimisation not mentioned in [8] but implemented in many inverse method provers [4,15] is *globalisation*: the forward version of the DL rule is specialized into the following two forms:

$$\frac{\Gamma\ ;\ [N] \vdash Q^- \quad N \notin \Gamma_0}{\Gamma, N\ ;\ \cdot \vdash \cdot\ ;\ Q^-}\,\text{DLF}_1 \qquad \frac{\Gamma\ ;\ [N] \vdash Q^- \quad N \in \Gamma_0}{\Gamma\ ;\ \cdot \vdash \cdot\ ;\ Q^-}\,\text{DLF}_2$$

where Γ_0 is the passive context of the goal sequent. This context is present in every sequent in the backward proof, so there is no need to mention it explicitly in the forward direction. For logic programs, Γ_0 will contain the clauses of the program and it is not important to distinguish between two computed sequents that differ only in the used clauses of the program.

Let us revisit the static polarity assignment question in the forward direction. The forward derived rule for the Horn clause $\forall\vec{x}\,.\,{\downarrow}m_1 \multimap \cdots \multimap {\downarrow}m_j \multimap n \in \Gamma_0$, after lifting to free variables, is:

$$\frac{\Gamma_1\ ;\ \cdot \vdash \cdot\ ;\ m_1' \quad \cdots \quad \Gamma_j\ ;\ \cdot \vdash \cdot\ ;\ m_j' \quad \theta = \text{mgu}(\langle m_1, \ldots, m_j\rangle, \langle m_1', \ldots, m_j'\rangle)}{(\Gamma_1, \ldots, \Gamma_n\ ;\ \cdot \vdash \cdot\ ;\ n)[\theta]}$$

For unit clauses, which provide the initial sequents, the passive context Γ is empty (because there are no premises remaining after globalisation). Therefore, all neutral sequents computed by forward reasoning will have empty passive contexts, giving us the rule:

$$\frac{\cdot\ ;\ \cdot \vdash \cdot\ ;\ m_1' \quad \cdots \quad \cdot\ ;\ \cdot \vdash \cdot\ ;\ m_j' \quad \theta = \text{mgu}(\langle m_1, \ldots, m_j\rangle, \langle m_1', \ldots, m_j'\rangle)}{(\cdot\ ;\ \cdot \vdash \cdot\ ;\ n)[\theta]}$$

Thus, this derived inference rule implements forward chaining for this clause. This situation is dual to the backward reading of the rules of fig. 1 where a static negative assignment to the atoms implemented backward chaining. As expected, a static positive polarity assignment to the atoms implements backward chaining in the forward calculus. The technical details of operational adequacy can be found in [7].

4 Dynamic Polarity Assignment

The previous section demonstrates that we can implement forward chaining (or bottom up logic programming) using the vocabulary of focusing and polarity assignment. For the rest of this paper we shall limit or attention to forward reasoning as the global strategy, with negative polarity assignment for the atoms as our means of implementing forward chaining.

Obviously the benefit of polarity assignment is that completeness is a trivial consequence of the completeness of focusing with respect to any arbitrary, even heterogeneous, polarity assignment for the atoms. Moreover, the completeness of the inverse method merely requires that the rule application strategy be fair. This minimal requirement of fairness does not force us to assign the polarity of all atoms statically, as long as we can guarantee that every atom that is relevant to the proof is eventually assigned a polarity (and that the rest of the inverse method engine is fair). Can we do better than static assignment with dynamic assignment? This section will answer this question affirmatively.

4.1 The mechanism of dynamic polarity assignment

Let us write unpolarized atoms (*i.e.*, atoms that haven't been assigned a polarity) simply in the form $a\,\vec{t}$, and allow them to be used as both positive and negative formulas in the syntax. That is, we extend the syntax as follows:

$$\begin{aligned} P, Q, \ldots &::= a\,\vec{t} \mid p \mid P \otimes Q \mid \mathbf{1} \mid P \oplus Q \mid \mathbf{0} \mid \exists x.\, P \mid {\downarrow}N \\ N, M, \ldots &::= a\,\vec{t} \mid n \mid N \,\&\, M \mid \top \mid P \multimap N \mid \forall x.\, N \mid {\uparrow}P \end{aligned}$$

For example, A Horn clause with unpolarized atoms have the syntax $\forall \vec{x}.\, a_1\, \vec{t_1} \multimap \cdots \multimap a_j\, \vec{t_j} \multimap b\, \vec{s}$ where the $\vec{x}$ are the variables that occur in the terms $\vec{t_1}, \ldots, \vec{t_j}, \vec{s}$.

Consider a variant of the focused inverse method where we allow two kinds of sequents as premises for inference rules: neutral sequents, as before, and sequents that have a focus on an unpolarized atom which we call *proto sequents*. An inference rule with proto sequent premises will be called a *proto rule*.

Definition 5 Environments *($\mathcal{E}, \mathcal{F}, \ldots$) are given by the following grammar:*

$$\begin{aligned} \mathcal{E}, \ldots &::= \mathcal{P} \mid \mathcal{Q} \\ \mathcal{P}, \mathcal{Q}, \ldots &::= \square \mid \mathcal{P} \otimes Q \mid P \otimes \mathcal{Q} \mid \mathcal{P} \oplus Q \mid P \oplus \mathcal{Q} \mid \exists x.\, \mathcal{P} \mid {\downarrow}\mathcal{N} \\ \mathcal{N}, \mathcal{M}, \ldots &::= \square \mid \mathcal{N} \,\&\, M \mid N \,\&\, \mathcal{M} \mid \mathcal{P} \multimap N \mid P \multimap \mathcal{N} \mid \forall x.\, \mathcal{N} \mid {\uparrow}\mathcal{P} \end{aligned}$$

We write $\mathcal{E}(A)$ for the formula formed by replacing the $\square$ in $\mathcal{E}$ with A, assuming it is syntactically valid. An environment $\mathcal{E}$ is called positive *(resp.* negative*) if $\mathcal{E}(p)$ (resp. $\mathcal{E}(n)$) is syntactically valid for any positive atom p (resp. negative atom n).*

Definition 6 (polarity assignment) *We write $A[a\,\vec{t} \leftarrow +]$ (resp. $A[a\,\vec{t} \leftarrow -]$) to stand for the positive (resp. negative) polarity assignment to the unpolarized atom $a\,\vec{t}$ in the formula A. It has the following recursive definition:*

- *If the unpolarized atom $a\,\vec{t}$ does not occur in A, then $A[a\,\vec{t} \leftarrow \pm] = A$.*
- *If $A = \mathcal{E}(a\,\vec{t})$ and $\mathcal{E}$ is positive, then*

$$\begin{aligned} A[a\,\vec{t} \leftarrow +] &= (\mathcal{E}(a^+\,\vec{t}))[a\,\vec{t} \leftarrow +] \\ A[a\,\vec{t} \leftarrow -] &= (\mathcal{E}({\downarrow}a^-\,\vec{t}))[a\,\vec{t} \leftarrow -] \end{aligned}$$

- *If $A = \mathcal{E}(a\,\vec{t})$ and $\mathcal{E}$ is negative, then*

$$\begin{aligned} A[a\,\vec{t} \leftarrow +] &= (\mathcal{E}({\uparrow}a^+\,\vec{t}))[a\,\vec{t} \leftarrow +] \\ A[a\,\vec{t} \leftarrow -] &= (\mathcal{E}(a^-\,\vec{t}))[a\,\vec{t} \leftarrow -] \end{aligned}$$

This definition is extended in the natural way to contexts, (proto) sequents, and (proto) rules.

Polarity assignment on proto rules generally has the effect of instantiating certain schematic meta-variables. For instance, consider the following proto-rule that corresponds to a left focus on the unpolarized Horn clause $C \triangleq \forall x, y.\, a\ x \multimap b\ y \multimap c\ x\ y$:

$$\frac{\Gamma, C \vdash [a\ s] \quad \Gamma, C \vdash [b\ t] \quad \Gamma, C\ ;\ [c\ s\ t] \vdash Q^-}{\Gamma, C\ ;\ \cdot \vdash \cdot\ ;\ Q^-}$$

All the premises of this rule are proto sequents. Suppose we assign a positive polarity to $a\ s$; this will change the proto rule to:

$$\frac{\Gamma, C \vdash [a^+ s] \quad \Gamma, C \vdash [b\ t] \quad \Gamma, C\ ;\ [c\ s\ t] \vdash Q^-}{\Gamma, C\ ;\ \cdot \vdash \cdot\ ;\ Q^-}$$

(where C' is $C[a\ s \leftarrow +]$). This proto rule actually corresponds to:

$$\frac{\Gamma, C', a^+ s \vdash [b\ t] \quad \Gamma, C', a^+ s\ ;\ [c\ s\ t] \vdash Q^-}{\Gamma, C', a^+ s\ ;\ \cdot \vdash \cdot\ ;\ Q^-}$$

because the only way to proceed further on the first premise is with the PR rule. This instantiates Γ with $\Gamma, a^+ s$. If we now assign a negative polarity to $c\ s\ t$, we would obtain the rule:

$$\frac{\Gamma, C'', a^+ s \vdash [b\ t]}{\Gamma, C'', a^+ s\ ;\ \cdot \vdash \cdot\ ;\ c^- s\ t}$$

(where $C'' = C'[c\ s\ t \leftarrow -]$) which instantiates Q^- to $c^- s\ t$. Finally, if we assign a negative polarity to $b\ t$, we would obtain the ordinary (non-proto) inference rule with neutral premise and conclusion:

$$\frac{\Gamma, C''', a^+ s\ ;\ \cdot \vdash \cdot\ ;\ b^- t}{\Gamma, C''', a^+ s\ ;\ \cdot \vdash \cdot\ ;\ c^- s\ t}$$

(where $C''' = C''[b\ t \leftarrow -]$).

4.2 Implementing magic sets with dynamic polarity assignment

This sub-section contains the main algorithm of this paper – a dynamic polarity assignment strategy that implements magic sets in the inverse method. The key feature of the algorithm is that it involves no global rewriting of the program clauses, so soundness is a trivial property. Completeness is obtained by showing that the algorithm together with the inverse method performs fairly on well-moded logic programs and queries.

The algorithm consists of dynamically assigning negative polarity to unpolarized atoms. Initially, all atoms in the program are unpolarized and the atom in the goal query is negatively polarized. It maintains the following lists:

- *Seeds*, which is a collection of the negatively polarized atoms;
- *Facts*, which is a list of computed facts which are ordinary neutral sequents;
- *Rules*, which is a list of partially applied, possibly proto, rules.

Whenever a fact is examined by the inner loop of the inverse method, new facts and partially applied (possibly proto) rules are generated. After the inner loop ends (*i.e.*, after all subsumption checks and indexing), the following *seeding step* is repeatedly performed until quiescence.

Definition 7 (seeding step) *For every right-focused proto-sequent in the premise of every proto rule, if the focused atom is mode correct—that is, if the input arguments of the atom are ground—then all instances of that atom for arbitrary outputs are assigned a negative polarity. These new negatively polarized atoms are added to the Seeds.*

For example, if the unpolarized atom $\mathsf{sum}\ 3\ 4\ (f(x))$ has a right focus in a proto rule and sum has mode `iio`, then all atoms of the form $\mathsf{sum}\ 3\ 4\ \_$ are assigned negative polarity. The seeding step will generate new facts or partially applied rules, which are then handled as usual by the inverse method.

4.3 Example

Let us revisit the example of sec. 2. Let Π_0 be the collection of unpolarized Horn clauses representing the program, *i.e.*:

$$\begin{aligned}
\Pi_0 = {} & \mathtt{lsum}\,[\,]\ 0, && (C_1)\\
& \forall x, y, j, k.\ \mathtt{lsum}\ y\ j \multimap \mathsf{sum}\ x\ j\ k \multimap \mathtt{lsum}\ (x :: y)\ k, && (C_2)\\
& \forall x.\ \mathsf{sum}\ 0\ x\ x, && (C_3)\\
& \forall x, y, z.\ \mathsf{sum}\ x\ y\ z \multimap \mathsf{sum}\ (\mathsf{s}\ x)\ y\ (\mathsf{s}\ z) && (C_4)
\end{aligned}$$

As before, let the modes be `io` for `lsum` and `iio` for sum. The above program is terminating and mode-correct for this moding. Consider the query $\mathtt{lsum}\ [1, 2, 3]\ X$, i.e., we are proving the goal sequent:

$$\underbrace{\Pi_0, \forall x.\ \mathtt{lsum}\ [1, 2, 3]\ x \multimap \mathsf{g}}_{\Gamma_0}\ ;\ \cdot \vdash \cdot\ ;\ \mathsf{g}$$

Since there are no switched subformulas, the only available rules will be for clauses in Γ_0 and the goal g. Using the subformula restriction and globalisation, we would then obtain the following derived proto rules:

$$\frac{\Gamma\ ;\ [\mathtt{lsum}\,[\,]\ 0] \vdash Q^-}{\Gamma\ ;\ \cdot \vdash \cdot\ ;\ Q^-}(C_1) \quad \frac{\Gamma_1\ ;\ [\mathtt{lsum}\ (x :: y)\ k] \vdash Q^- \quad \Gamma_2 \vdash [\mathtt{lsum}\ y\ j] \quad \Gamma_3 \vdash [\mathsf{sum}\ x\ j\ k]}{\Gamma_1, \Gamma_2, \Gamma_3\ ;\ \cdot \vdash \cdot\ ;\ Q^-}(C_2)$$

$$\frac{\Gamma\ ;\ [\mathsf{sum}\ 0\ x\ x] \vdash Q^-}{\Gamma\ ;\ \cdot \vdash \cdot\ ;\ Q^-}(C_3) \quad \frac{\Gamma_1\ ;\ [\mathsf{sum}\ (\mathsf{s}\ x)\ y\ (\mathsf{s}\ z)] \vdash Q^- \quad \Gamma_2 \vdash [\mathsf{sum}\ x\ y\ z]}{\Gamma_1, \Gamma_2\ ;\ \cdot \vdash \cdot\ ;\ Q^-}(C_4)$$

$$\frac{\Gamma_1\ ;\ [\mathsf{g}] \vdash Q^- \quad \Gamma_2 \vdash [\mathtt{lsum}\ [1, 2, 3]\ x]}{\Gamma_1, \Gamma_2\ ;\ \cdot \vdash \cdot\ ;\ Q^-}(\mathsf{g})$$

There are no initial sequents, so we perform some seeding steps. The initial polarity assignment is negative for the goal g; this produces the following instance of (g):

$$\frac{\Gamma_2 \vdash [\mathtt{lsum}\ [1, 2, 3]\ x]}{\Gamma\ ;\ \cdot \vdash \cdot\ ;\ \mathsf{g}^-}(\mathsf{g}')$$

Now we have a right focus on a well-moded unpolarized atom, *viz.* $\mathtt{lsum}\ [1,2,3]\ x$, so we add $\mathtt{lsum}^-\ [1,2,3]\ \_$ to the *Seeds*. This produces two instances of the proto rule (C_2) depending on the two ways in which the seed can match the proto premises.

$$\frac{\Gamma_1 \vdash [\mathtt{lsum}\ [2,3]\ j] \quad \Gamma_2 \vdash [\mathtt{sum}\ 1\ j\ k]}{\Gamma_1, \Gamma_2\ ;\ \cdot \vdash \cdot\ ;\ \mathtt{lsum}^-\ [1,2,3]\ k}\ (C_{21})$$

$$\frac{\Gamma_1\ ;\ [\mathtt{lsum}\ (x :: [1,2,3])\ k] \vdash Q^- \quad \Gamma_2\ ;\ \cdot \vdash \cdot\ ;\ \mathtt{lsum}^-\ [1,2,3]\ j \quad \Gamma_3 \vdash [\mathtt{sum}\ x\ j\ k]}{\Gamma_1, \Gamma_2, \Gamma_3\ ;\ \cdot \vdash \cdot\ ;\ Q^-}\ (C_{22})$$

The first premise in (C_{21}) is well-moded and will produce further seeds. However, (C_{22}) produces no seeds as there are no proto premises with a right focus on a well-moded unpolarized atom. Continuing the seeding steps for (C_{21}) we produce the following new useful proto rules:

$$\frac{\Gamma_1 \vdash [\mathtt{lsum}\ [2]\ j] \quad \Gamma_2 \vdash [\mathtt{sum}\ 2\ j\ k]}{\Gamma_1, \Gamma_2\ ;\ \cdot \vdash \cdot\ ;\ \mathtt{lsum}^-\ [2,3]\ k}\ (C_{211}) \qquad \frac{\Gamma_1 \vdash [\mathtt{lsum}\ []\ j] \quad \Gamma_2 \vdash [\mathtt{sum}\ 3\ j\ k]}{\Gamma_1, \Gamma_2\ ;\ \cdot \vdash \cdot\ ;\ \mathtt{lsum}^-\ [3]\ k}\ (C_{2111})$$

The rule (C_{2111}) produces a seed $\mathtt{lsum}^-\ []\ \_$ that matches the premise of (C_1) to produce our first fact: $\cdot\ ;\ \cdot \vdash \cdot\ ;\ \mathtt{lsum}^-\ []\ 0$. This can now be applied in the premise of (C_{2111}) by the inverse method loop to produce the following partially applied instance:

$$\frac{\Gamma \vdash [\mathtt{sum}\ 3\ 0\ k]}{\Gamma\ ;\ \cdot \vdash \cdot\ ;\ \mathtt{lsum}^-\ [3]\ k}\ (C_5)$$

This finally gives us our first seed for $\mathtt{sum}$, *viz.* $\mathtt{sum}^-\ 3\ 0\ \_$. This seed will, in turn produce seeds $\mathtt{sum}^-\ 2\ 0\ \_$, $\mathtt{sum}^-\ 1\ 0\ \_$, and $\mathtt{sum}^-\ 0\ 0\ \_$ from instances of the rule (C_4). The last of these seeds will instantiate (C_3) to give our second fact, $\cdot\ ;\ \cdot \vdash \cdot\ ;\ \mathtt{sum}^-\ 0\ 0\ 0$. The inverse method will then be able to use this rule to partially apply the instances of the (C_3) rule to produce, eventually, $\cdot\ ;\ \cdot \vdash \cdot\ ;\ \mathtt{sum}^-\ 3\ 0\ 3$, which can be matched to (the instance of) (C_5) to give our second derived fact about $\mathtt{lsum}$, *viz.* $\cdot\ ;\ \cdot \vdash \cdot\ ;\ \mathtt{lsum}^-\ [3]\ 3$. These steps repeat twice more until we eventually derive $\cdot\ ;\ \cdot \vdash \cdot\ ;\ \mathtt{lsum}^-[1,2,3]\ 6$, which finally lets us derive the goal sequent using (the instance of) (g).

4.4 Correctness

Crucially, no further inferences are possible in the example of the previous section. There will never be any facts generated about $\mathtt{lsum}^-\ [5,1,2,3,4]\ x$, for instance, because there is never a seed of that form. Thus, as long as there is a well-founded measure on the seeds that is strictly decreasing for every new seed, this implementation of the inverse method will saturate.

Lemma 8 (seeding lemma) *All atoms occurring to the right of sequents in the Facts list are instances of atoms in the Seeds.*

Proof. Since the only polarity assignment is to assign an unpolarized atom the negative polarity, the only effect it has on proto inference rules is to finish left-focused proto sequent premises with NL, and turn right-focused proto sequent premises into neutral sequent premises. Finishing the left-focused premises has the side effect of instantiating the right hand side with the newly negatively polarized atom. If there are no neutral premises as a result of this assignment, then the newly generated fact satisfies the required criterion. Otherwise, when the conclusion is eventually generated by applying the rule in the inverse method, the right hand side will be an instance of the negatively polarized atom. □

The main result of this paper is a simple corollary.

Corollary 9 (saturation) *Given a well-moded logic program that terminates on all well-moded queries—i.e., all derivations of a well-moded query are finite—the inverse method with the dynamic polarity assignment algorithm of sec. 4.1 saturates for all well-moded queries.*

Proof (Sketch). Instead of giving a fully formal proof, which is doable in the style of [14], we give only the intuition for the proof. Note that if the logic program is terminating for all well-moded queries, then there is a bounded measure | | that is strictly decreasing from head to body of all clauses in the program. We use this measure to build a measure on the *Seeds* collection as follows:

- For each atom in *Seeds*, pick the element with the smallest | |-measure.
- For each atom not in *Seeds*, pick greatest lower bound of the | |-measure.
- Pick a strict but arbitrary ordering of all the predicate symbols and arrange the measures selected in the previous two steps in a tuple according to this ordering. This tuple will be the measure of *Seeds*.

It is easy to see that this measure on *Seeds* has a lower bound according to the lexicographic ordering. Therefore, all we need to show is that this measure is decreasing on *Seeds* for every seeding step and then we can use lem. 8 to guarantee saturation. But this is easily shown because the | |-measure decreases when going from the conclusion to the premises of every derived inference rule for the clauses of the logic program (see the example in sec. 4.3). □

The completeness of the dynamic polarity assignment algorithm follows from the completeness of focusing with (arbitrary) polarity assignment, the completeness of the inverse method given a fair strategy, and the observation that *Seeds* contains a superset of all predicates that can appear as subgoals in a top-down search of the given logic program.

5 Conclusion

We have shown how to implement the magic sets constraint on a focused forward search strategy by dynamically assigning polarities to unpolarized atoms. As one immediate consequence, our forward engine can respond with the same answer set as traditional back-chaining for well-moded and terminating programs. The notion of dynamic polarity assignment is novel to this work and the last word on it is far from written. The obvious next step is to see how it generalizes to fragments larger than Horn theories. More fundamentally, while fairness in the inverse method gives a general external criterion for completeness, an internal criterion for judging when a given dynamic polarity assignment strategy will be complete is currently missing.

Acknowledgement: We thank Brigitte Pientka for several useful discussions about magic sets and polarity assignment.

References

1. J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *J. of Logic and Computation*, 2(3):297–347, 1992.
2. K. R. Apt and E. Marchiori. Reasoning about prolog programs from modes through types to assertions. *Formal Aspects of Computing*, 6(A):743–764, 1994.
3. C. Beeri and R. Ramakrishnan. On the power of magic. *Journal of Logic Programming*, 10(1/2/3&4):255–299, 1991.
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. Classical and intuitionistic subexponential logics are equally expressive. In A. Dawar and H. Veith, editors, *CSL 2010: Computer Science Logic*, volume 6247 of *LNCS*, pages 185–199, Brno, Czech Republic, Aug. 2010. Springer.
6. K. Chaudhuri and F. Pfenning. A focusing inverse method theorem prover for first-order linear logic. In *Proceedings of the 20th Conference on Automated Deduction (CADE)*, volume 3632 of *LNCS*, pages 69–83, Tallinn, Estonia, July 2005.
7. 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.
8. A. Degtyarev and A. Voronkov. The inverse method. In *Handbook of Automated Reasoning*, pages 179–272. Elsevier and MIT Press, 2001.
9. K. Donnelly, T. Gibson, N. Krishnaswami, S. Magill, and S. Park. The inverse method for the logic of bunched implications. In *Proceedings of the 11th International Conference on Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 3452 of *LNCS*, pages 466–480, Montevideo, Uruguay, Mar. 2004.
10. S. Heilala and B. Pientka. Bidirectional decision procedures for the intuitionistic propositional modal logic IS4. In F. Pfenning, editor, *Proceedings of the 21st International Conference on Automated Deduction (CADE)*, volume 4603 of *LNAI*, pages 116–131, Bremen, Germany, July 2007. Springer.
11. J. M. Howe. *Proof Search Issues in Some Non-Classical Logics*. PhD thesis, University of St Andrews, Dec. 1998. Available as University of St Andrews Research Report CS/99/1.
12. C. Liang and D. Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, 410(46):4747–4768, 2009.
13. C. Liang and D. Miller. A unified sequent calculus for focused proofs. In *LICS: 24th Symp. on Logic in Computer Science*, pages 355–364, 2009.
14. P. Mascellani and D. Pedreschi. The declarative side of magic. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, pages 83–108, London, UK, 2002. Springer-Verlag.
15. 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.
16. D. Miller and V. Nigam. Incorporating tables into proofs. In J. Duparc and T. A. Henzinger, editors, *CSL 2007: Computer Science Logic*, volume 4646 of *Lecture Notes in Computer Science*, pages 466–480. Springer, 2007.
17. D. Miller and A. Saurin. From proofs to focused proofs: a modular proof of focalization in linear logic. In J. Duparc and T. A. Henzinger, editors, *CSL 2007: Computer Science Logic*, volume 4646 of *Lecture Notes in Computer Science*, pages 405–419. Springer, 2007.

18. J. D. Ullman. *Principles of Database and Knowledge-base Systems, Volume II: The New Techniques*. Principles of Computer Science. Computer Science Press, 1989.