Designing a safe forward chaining tactic using productive proofs

Kaustuv Chaudhuri, Arunava Gantait, and Dale Miller

Inria Saclay & LIX, Institut Polytechnique Paris

Abstract. We develop a proof-theoretic framework for treating forward chaining and the associated notions of saturation within a multisorted, first-order intuitionistic logic with equality. The notions of *polarity* and *focused proofs* are central to our approach since they provide a generalization of geometric implications as *bipolar formulas*, a natural proof-theoretic interpretation of forward chaining, and the concept of *productive proofs*. We identify conditions under which forward chaining with a given set of formulas is guaranteed to terminate in a finite number of steps. The motivation for this research stems, in part, from exploring avenues to automate the Abella theorem prover, which relies on relational specifications, and where theorems in typical proof developments are essentially bipolar formulas. We illustrate the potential benefits of automating forward chaining and saturation for Abella by presenting examples that compute congruence closure and assist in other equational and relational reasoning tasks.

1 Introduction

When attempting to prove a theorem in an interactive theorem prover, the user is typically presented with a collection of *proof states* or *subgoals* where each such state consists of two key elements:

1. A *context* that consists of a collection Σ of (eigen-)variables, the collection $\mathcal{L}$ of previously loaded axioms and verified theorems, and a set Γ of local assumptions relevant to the position of the subgoal in the entire proof.[1]
2. A *goal* (which we write E) that needs to be established within the combined context of $\mathcal{L}$, Σ, and Γ.

We can write this proof state as the *sequent* $\Sigma :: \mathcal{L}, \Gamma \vdash E$. To reduce notational clutter, we will usually omit $\mathcal{L}$ as it is static and ambiently present in every sequent; we will also often omit the $\Sigma ::$ if those variables are not relevant.

In interactive provers, there are broadly two kinds of logical actions that can be performed on any proof state that correspond to the two *sides* of the sequent. The first kind analyzes the goal formula E and decomposes the proof state

[1] In formalisms based on dependent type theory, the two categories of variables and assumptions are merged; however, in logic and in proof theory they correspond to two separate and distinct connectives – universal quantification $\forall$ and implication $\supset$.

into possibly several proof states with simpler goals. The other kind of action looks at one or more facts in Γ (or $\mathcal{L}$) and then uses rules such as case-analysis, destruction, inversion, etc. to decompose the facts into simpler facts, without modifying the goal formula. In both kinds of actions, the lemmas in $\mathcal{L}$ (and possibly also in Γ) can also participate, and the user generally intends such lemmas to serve as recipes for the creation of high level inference steps. For example, suppose the user defines the binary relation $\leq$ (on natural numbers, say) and establishes the following transitivity lemma:

$$\forall x \forall y \forall z.(x \leq y \wedge y \leq z \supset x \leq z).$$

The user expects that this lemma will yield the following two derived inference rules corresponding to the two sides of the sequent:

- *Backward chaining*: In order to prove $t \leq s$, prove instead $t \leq u$ and $u \leq s$ for some interpolant term u, using the inference rule[2]

$$\frac{\Gamma \vdash t \leq u \quad \Gamma \vdash u \leq s}{\Gamma \vdash t \leq s} \ .$$

  This rule is an action of the first kind as it modifies the goal but not the proof context. Organizing proof search using backward chaining is well understood and has been used to justify the design of logic programming languages [21].
- *Forward chaining*: If the proof context contains $t \leq u$ and $u \leq s$ then produce the additional assumption $t \leq s$, which corresponds to the following rule, which only modifies the proof context but not the goal:

$$\frac{\Gamma, t \leq u, u \leq s, t \leq s \vdash E}{\Gamma, t \leq u, u \leq s \vdash E} \ .$$

Observe that in either formulation of these derived inference rules, only the relational atoms are present; indeed, none of the compound subformulas of the transitivity lemma are present.

In this paper we will use Gentzen's *sequent calculus* to analyze and transform proof states, i.e., sequents. The ordinary sequent calculus provides a foundational framework that is able to represent actions on both the proof context and the goal formula as left and right introduction rules, respectively; moreover, both of the above kinds of chaining correspond to applications of $\forall$-left, $\supset$-left, and initial rules. However, the rules of the sequent calculus operate on individual connectives at a time, so the space of sequent proofs also contains many other unprincipled proofs that do not follow the forward or backward chaining protocol, and leave partially instantiated and applied versions of lemmas in the proof contexts. The general topic of converting lemmas into more *synthetic* inference rules has been explored since at least the work of Negri [25, 26], and appears more recently in a more systematic fashion in the work on *focused proof systems*, which is the setting we will use in this paper. Focused proof systems were initially designed

[2] Like all inference rules in this paper, the reading direction should be from the conclusion to the premises.

as a particular kind of protocol for (linear) logic programming by Andreoli [3], but in a more modern presentation they define a normal form for sequent proofs that respects a *polarity* distinction. This perspective allows focused proofs to be defined for classical and intuitionistic logics as well where the polarities of connectives are not predetermined but assignable [16]. Both the forward chaining and the backward chaining interpretations of lemmas described above arise in focused proofs simply as a consequence of assigning the right polarities to the logical connectives and atomic formulas [8, 18].

In this paper we are principally concerned with the forward chaining interpretation of lemmas, which arises out of assigning a positive polarity to the conjunction and all atomic formulas. This interpretation has the obvious problem that the premise sequents can be *more* complex than the conclusions, since they may contain additional assumptions that were generated from the existing assumptions. Thus, automated proof search using this interpretation will be immediately non-terminating without further controls. Similar issues of non-termination with forward chaining have been considered in the context of bottom-up logic programming (see e.g. [24]), and termination sometimes needs to be explicitly engineered in such approaches with the use of global program transformations [5].

In this paper we will identify a class of lemmas for which termination can be achieved by direct analysis of focused proofs. In particular, we identify a notion of *productive proof rule* as a synthetic inference rule that tries to create a fact that was not already provable in a single focused phase. We can then run the forward chaining interpretation as long as the inferences stay productive. We show that this search strategy terminates on a fairly natural class of lemmas that consist of *bipolar formulas* that have an additional *safety* condition.

We also implement this strategy as a new forward chaining tactic in the Abella theorem prover, and show experimentally that it reduces much of the tedium of lemma applications. In fact, we show how to implement certain other well known operations such as transitive closure (on finite graphs) and congruence closure. We use Abella because of its proof-theoretic basis and its use of (an extension) of intuitionistic first-order logic; however, the approach of this paper can be used to define similar tactics using forward chaining in other proof assistants.

2 Focused proofs for intuitionistic logic with equality

2.1 Polarities and formulas

In a two-sided sequent presentation of linear logic, every logical connective has the property that either its right-introduction rules are invertible or its left-introduction rules are invertible (never both). This allows us to classify every connective into two *polarities*, with those with right-introduction rules being invertible called *negative* and those with left-introduction rules being invertible called *positive*. For intuitionistic and classical (non-linear) logics, this definition of polarity is not directly applicable, since there are connectives such as conjunction $\wedge$ that can be given invertible rules on both sides of the sequent arrow, as is the case with the popular *G3* and *G3i* proof systems [27]. Particular choices of rules

may give particular *polarizations* of these connectives with ambiguous polarities; for example, Gentzen's *LJ* and *LK* calculi [14] treats conjunction as uniquely negative. (Atomic formulas in every one of these logics also have ambiguous polarities, but this is only relevant in the presence of the *focusing* restriction described in the next subsection.)

One common technique in logics such as intuitionistic logic is to include both *polarizations* of connectives that can have ambiguous polarities. In our setting, the conjunction $\wedge$ and its unit $\mathtt{t}$ will both have two polarized forms, namely, $\wedge^-$, $\wedge^+$, and $\mathtt{t}^-$, $\mathtt{t}^+$. To be precise, we should maintain a distinction between *formulas* and *polarized formulas*. The latter class of formulas will not contain $\mathtt{t}$ or $\wedge$, but may include $\mathtt{t}^-$, $\mathtt{t}^+$, $\wedge^-$, and $\wedge^+$. Given a polarized formula B, we define its unpolarized form $\tilde{B}$ as that formula obtained by replacing every occurrence of $\wedge^-$ and $\wedge^+$ in B with $\wedge$, and every instance of $\mathtt{t}^-$ and $\mathtt{t}^+$ in B with $\mathtt{t}$. When this distinction is clear from the context, we may sometimes refer to polarized formulas simply as formulas.

The multisorted first-order intuitionistic logic we consider in this paper contains the following connectives.

- Equality $=$ is treated as a logical connective, and equality and all other atomic formulas are treated as positively polarized.
- The quantifiers $\forall_\tau$ and $\exists_\tau$ are first-order: i.e., the types τ range over primitive types. Of these, $\forall_\tau$ has negative polarity and $\exists_\tau$ has positive polarity.
- The propositional connectives are divided into *positive connectives* ($\wedge^+$, $\mathtt{t}^+$, $\vee$, $\mathtt{f}$) and the *negative connectives* ($\wedge^-$, $\mathtt{t}^-$, $\supset$).

We say that a formula is *negative* if its top-level connective is negative, and is *positive* if its top-level connective is positive.

2.2 The multifocused proof system $LJF^=$

Figure 1 contains the $LJF^=$ proof system in which all formulas in all sequents are polarized. There are three kinds of sequents in this proof system:

$$\begin{array}{cl} \Gamma \Uparrow \Theta \vdash \Delta \Uparrow \Xi & \text{(unfocused sequent)} \\ \Gamma \vdash P \Downarrow & \text{(right-focused sequent)} \\ \Gamma \Downarrow \Theta \vdash P & \text{(left-focused sequent)} \end{array}$$

Here, Γ is a *set* of negatively polarized formulas or atoms, and Θ, Δ, and Ξ are multisets of polarized formulas with Ξ restricted to positive formulas. Given that we are working with intuitionistic logic, we insist that the multiset union of Δ and Ξ is a singleton multiset in unfocused sequents, i.e., one of Δ or Ξ is a singleton and the other is empty. The introduction rules introduce connectives only within the Θ and Δ zones. Whenever Θ or Δ is empty, we indicate it with an empty space; furthermore, in the case of unfocused sequents, we also elide the corresponding $\Uparrow$. Sequents of the form $\Gamma \vdash P$ (i.e., $\Gamma \Uparrow \vdash \Uparrow P$) will be called *border sequents*.

The $LJF^=$ proof system is *multifocused* in the sense that in the left-focused sequent $\Gamma \Downarrow \Theta \vdash C$ the zone of foci Θ can have multiple formulas. Multifocusing is a generalization of ordinary focusing that permits the parallel application of

Structural rules

$$\frac{\Sigma :: \Gamma \vdash P \Downarrow}{\Sigma :: \Gamma \vdash P}\ D_r \qquad \frac{\Sigma :: \Gamma \Uparrow \Theta \vdash \ \Uparrow P}{\Sigma :: \Gamma \Uparrow \Theta \vdash P \Uparrow}\ S_r \qquad \frac{\Sigma :: \Gamma \vdash N \Uparrow}{\Sigma :: \Gamma \vdash N \Downarrow}\ R_r$$

$$\frac{\Sigma :: \Gamma \Downarrow \mathcal{N} \vdash P}{\Sigma :: \Gamma \vdash P}\ D_l \qquad \frac{\Sigma :: C, \Gamma \Uparrow \Theta \vdash \Delta \Uparrow \Xi}{\Sigma :: \Gamma \Uparrow C, \Theta \vdash \Delta \Uparrow \Xi}\ S_l \qquad \frac{\Sigma :: \Gamma \Uparrow \mathcal{P} \vdash P}{\Sigma :: \Gamma \Downarrow \mathcal{P} \vdash P}\ R_l$$

Here, $\mathcal{P}$ is a multiset of positive formulas and $\mathcal{N}$ is a non-empty multiset of negative formulas such that every formula in $\mathcal{N}$ is in the set Γ.

Initial rule

$$\frac{}{\Sigma :: \Gamma, P \vdash P \Downarrow}\ I_r,\ P \text{ atomic}$$

Invertible introduction rules

$$\frac{\Sigma :: \Gamma \Uparrow \Theta \vdash B_1 \Uparrow \quad \Sigma :: \Gamma \Uparrow \Theta \vdash B_2 \Uparrow}{\Sigma :: \Gamma \Uparrow \Theta \vdash B_1 \wedge^- B_2 \Uparrow} \qquad \frac{}{\Sigma :: \Gamma \Uparrow \Theta \vdash \mathsf{t}^- \Uparrow}$$

$$\frac{x{:}\tau, \Sigma :: \Gamma \Uparrow \Theta \vdash B \Uparrow}{\Sigma :: \Gamma \Uparrow \Theta \vdash \forall x_\tau.B \Uparrow} \qquad \frac{\Sigma :: \Gamma \Uparrow B_1 \vdash B_2 \Uparrow}{\Sigma :: \Gamma \Uparrow \Theta \vdash B_1 \supset B_2 \Uparrow}$$

$$\frac{\Sigma :: \Gamma \Uparrow \Theta \vdash \Delta \Uparrow \Xi}{\Sigma :: \Gamma \Uparrow \mathsf{t}^+, \Theta \vdash \Delta \Uparrow \Xi} \qquad \frac{\Sigma :: \Gamma \Uparrow B_1, B_2, \Theta \vdash \Delta \Uparrow \Xi}{\Sigma :: \Gamma \Uparrow B_1 \wedge^+ B_2, \Theta \vdash \Delta \Uparrow \Xi}$$

$$\frac{}{\Sigma :: \Gamma \Uparrow \mathsf{f}, \Theta \vdash \Delta \Uparrow \Xi} \qquad \frac{\Sigma :: \Gamma \Uparrow B_1, \Theta \vdash \Delta \Uparrow \Xi \quad \Sigma :: \Gamma \Uparrow B_2, \Theta \vdash \Delta \Uparrow \Xi}{\Sigma :: \Gamma \Uparrow B_1 \vee B_2, \Theta \vdash \Delta \Uparrow \Xi}$$

$$\frac{x{:}\tau, \Sigma :: \Gamma \Uparrow B, \Theta \vdash \Delta \Uparrow \Xi}{\Sigma :: \Gamma \Uparrow \exists x_\tau.B, \Theta \vdash \Delta \Uparrow \Xi}$$

$$\frac{(\theta = \mathrm{mgu}(t, s)) \quad \Sigma\theta :: \Gamma\theta \Uparrow \Theta\theta \vdash \Delta\theta \Uparrow \Xi\theta}{\Sigma :: \Gamma \Uparrow s = t, \Theta \vdash \Delta \Uparrow \Xi} \qquad \frac{(t \text{ and } s \text{ not unifiable})}{\Sigma :: \Gamma \Uparrow s = t, \Theta \vdash \Delta \Uparrow \Xi}$$

Non-invertible introduction rules

$$\frac{}{\Sigma :: \Gamma \vdash t = t \Downarrow} \qquad \frac{}{\Sigma :: \Gamma \vdash \mathsf{t}^+ \Downarrow} \qquad \frac{\Sigma :: \Gamma \vdash B_i \Downarrow}{\Sigma :: \Gamma \vdash B_1 \vee B_2 \Downarrow}\ i \in \{1, 2\}$$

$$\frac{\Sigma :: \Gamma \vdash B_1 \Downarrow \quad \Sigma :: \Gamma \vdash B_2 \Downarrow}{\Sigma :: \Gamma \vdash B_1 \wedge^+ B_2 \Downarrow} \qquad \frac{(\Sigma \vdash t : \tau) \quad \Sigma :: \Gamma \vdash [t/x]B \Downarrow}{\Sigma :: \Gamma \vdash \exists x_\tau.B \Downarrow}$$

$$\frac{\Sigma :: \Gamma \vdash B_1 \Downarrow \quad \Sigma :: \Gamma \Downarrow B_2, \Theta \vdash P}{\Sigma :: \Gamma \Downarrow B_1 \supset B_2, \Theta \vdash P} \qquad \frac{\Sigma :: \Gamma \Downarrow B_i, \Theta \vdash P}{\Sigma :: \Gamma \Downarrow B_1 \wedge^- B_2, \Theta \vdash P}$$

$$\frac{(\Sigma \vdash t : \tau) \quad \Sigma :: \Gamma \Downarrow [t/x]B, \Theta \vdash P}{\Sigma :: \Gamma \Downarrow \forall x_\tau.B, \Theta \vdash P}$$

Γ ranges over a set of negatively polarized formulas or atoms; Δ, Θ range over multisets of polarized formulas; Ξ ranges over multisets of positive formulas; P denotes a positive formula; N denotes a negative formula; C denotes either a negative formula or a (positive) atom; and B denotes any polarized formula. The multiset union of Δ and Ξ must be a singleton. Parenthesized premises are to be interpreted as side conditions.

Fig. 1. The $LJF^=$ proof system: multifocused version

synthetic inference rules [6, 7]. We will exploit multifocusing in Section 4.1 when we deal with forward chaining and saturation.

Focused proofs are constructed in two phases. The *invertible phase* (a.k.a. *negative* or *asynchronous* phase) involves unfocused sequents and uses invertible rules on principal formulas drawn from Θ or Δ. The *non-invertible* phase (a.k.a. *positive* and *synchronous* phase) involves focused sequents. Transitions between these two phases happen with a collection of structural rules, where the *release* rules (R_l and R_r) transition from a focused conclusion to an unfocused premise, while the *decide* rules (D_l and D_r) transition from an unfocused conclusion to a focused premise. Note that the conclusions of both decide rules are border sequents. In the invertible phase, the two *store* rules (S_l and S_r) are used to transfer a formula from Θ and Δ to Γ and Ξ, respectively, when no further invertible rules can be applied to that formula; these store rules are internal to the invertible phase. The initial rule I_r is part of the non-invertible phase and requires a right focus on a (positive) atom.

The $LJ^{=}$ proof system is defined as the unfocused version of $LJF^{=}$; that is, $LJ^{=}$ results from taking all the inference rules of $LJF^{=}$ and replacing $\Uparrow$ and $\Downarrow$ with commas and replacing all polarized formulas with their unpolarized forms. The result of doing this replacement on the store rules (S_l and S_r) and the release rules (R_l and R_r) would equate their conclusions and premises, so we drop these rules from $LJ^{=}$. Wherever an $LJF^{=}$ rule is restricted to formulas of a certain polarity, the corresponding $LJ^{=}$ rule drops the restriction. Thus, the D_l rule becomes the contraction rule in $LJ^{=}$.

Theorem 1 (Soundness and Completeness of $LJF^{=}$ wrt $LJ^{=}$). *Let B be a polarized formula defined over the variables in Σ. Then the sequent $\Sigma :: \cdot \Uparrow \vdash B \Uparrow$ is provable in $LJF^{=}$ if and only if the sequent $\Sigma :: \cdot \vdash \tilde{B}$ is provable in $LJ^{=}$.*

Proof (Sketch). The forward direction (soundness) is immediate. The converse (completeness) is more involved but follows by modifying the proof for the completeness of the singly focused version of LJF in [16] to also account for equality (following the treatment of equality in, say, [19]). This use of LJF assumes that all atomic formulas are polarized positively. When completeness holds for the singly focused proof system, it immediately holds for the multi-focused system which includes the singly focused system. □

3 Bipolar formulas and productive proofs

3.1 Purely positive and bipolar formulas

A polarized formula is *purely positive* if the only logical connectives in it are positive. Let $\mathbb{P}$ be the set of all purely positive formulas. The following theorem shows that $LJF^{=}$ proofs of purely positive formulas are particularly simple.

Theorem 2. *For any $P \in \mathbb{P}$:*

1. *If the sequent $\Sigma :: \Gamma \vdash P \Downarrow$ has an $LJF^{=}$ proof, it has a proof that is exactly one non-invertible phase composed only of right-introduction rules.*
2. *It is decidable whether or not $\Sigma :: \Gamma \vdash P \Downarrow$ is provable.*

Proof. By immediate inspection of the inference rules in Figure 1. □

The dual set $\mathbb{N}$ of purely negative formulas consists of formulas N with the grammar $N ::= \mathtt{t}^- \mid N_1 \wedge^- N_2 \mid P \supset N \mid \forall_\tau x.N$ (where $P \in \mathbb{P}$). It is easy to show that a formula in $\mathbb{N}$ is provably equivalent in $LJF^=$ to $\mathtt{t}^-$. As a result, we shall generally be able to ignore these formulas. Note that purely negative formulas can have positive subformulas, but only as antecedents of implications.

A more interesting class of formulas are the *bipolar* formulas which are negative formulas with the grammar $B ::= \mathtt{t}^- \mid B_1 \wedge^- B_2 \mid P_1 \supset P_2 \mid \forall_\tau x.B$ (where P_1 and P_2 are in $\mathbb{P}$). It is not difficult to show that every bipolar formula is logically equivalent to the conjunction ($\wedge^-$) of formulas of the form

$$\forall \bar{x}.(A_1 \wedge^+ \cdots \wedge^+ A_n \supset P), \quad (n \geq 0)$$

where $P \in \mathbb{P}$, the A_is are atoms, and the empty $\wedge^+$ is written as $\mathtt{t}^+$. Following [25], we shall call formulas of this shape *geometric implications*, and we say that the *head* of this implication is P. If B is a bipolar formula, we write $[\![B]\!]$ to denote the multiset of all such geometric implications in this conjunction. We extend this function to sets of bipolar formulas, i.e., $[\![B_1, \ldots, B_m]\!] = [\![B_1]\!] \cup \cdots \cup [\![B_m]\!]$. A derivation that starts (reading conclusion-upwards) with D_l on such a formula would have the following non-invertible phase in $LJF^=$:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, A_1\theta, \ldots, A_n\theta \vdash A_1\theta \wedge^+ \cdots \wedge^+ A_n\theta \Downarrow}\,\dagger \quad \dfrac{\Gamma, A_1\theta, \ldots, A_n\theta \Uparrow P\theta \vdash C}{\Gamma, A_1\theta, \ldots, A_n\theta \Downarrow P\theta \vdash C}}{\Gamma, A_1\theta, \ldots, A_n\theta \Downarrow A_1\theta \wedge^+ \cdots \wedge^+ A_n\theta \supset P\theta \vdash C}}{\dfrac{\Gamma, A_1\theta, \ldots, A_n\theta \Downarrow \forall \bar{x}.(A_1 \wedge^+ \cdots \wedge^+ A_n \supset P) \vdash C}{\Gamma, A_1\theta, \ldots, A_n\theta \vdash C}\,D_l}\,\ddagger$$

where the derivation marked † consists of $n-1$ applications of $\wedge^+$-right and I_r, and that marked ‡ consists of repeated applications of $\forall$-left. We say that forward chaining on $\forall \bar{x}.(A_1 \wedge^+ \cdots \wedge^+ A_n \supset P)$ succeeds if some instance of $A_1, \ldots, A_n$ are present in the proof context of the root sequent and the information present in the same instance of the head P is added to the proof context.

In the rest of this paper we will present a complete proof search strategy for formulas of the form $(B_1 \wedge^+ \cdots \wedge^+ B_n) \supset B_0$ where $B_0, \ldots, B_n$ ($n \geq 0$) are bipolar formulas. The border sequents within an $LJF^=$ proof of such a formula are of the form $B_1, \ldots, B_n, \mathcal{A} \vdash P$, where P is in $\mathbb{P}$ and $\mathcal{A}$ is a set of atomic formulas: we shall call such sequents *reduced* sequents. The $LJF^=$ system restricted to these border sequents can be simplified by requiring the goal formulas to be purely positive, and removing the now redundant R_r and S_r rules.

If we abstract the structure of $LJF^=$ proofs of reduced sequents as a tree of occurrences of decide rules, then every occurrence of a D_r rule is a terminal node and every occurrence of a D_l rule is an internal node. Additionally, a formula occurrence under left focus in the conclusion of a left-introduction rule corresponds uniquely to a formula occurrence under left focus in exactly one of the premises of that rule. As a consequence, if an instance of D_l selects $m \geq 1$ formulas for its focus, then the R_l rule at the top of that non-invertible phase also has m formulas as the left foci.

3.2 Productive proofs

An application of the R_l rule (where $\mathcal{P}$ is a multiset of purely positive formulas)

$$\frac{\Sigma :: \Gamma \Uparrow \mathcal{P} \vdash Q}{\Sigma :: \Gamma \Downarrow \mathcal{P} \vdash Q}\ R_l$$

is *unproductive* if there is a $P \in \mathcal{P}$ such that $\Sigma :: \Gamma \vdash P \Downarrow$ is provable. An instance of R_l is *productive* in case it is not unproductive. A proof is productive if all occurrences of R_l are productive. The intuition here is that if $\Sigma :: \Gamma \vdash P \Downarrow$ is provable then the information content of P is already in Γ: using R_l on $\Sigma :: \Gamma \Downarrow P \vdash E$ would then add the content of P to Γ, which is redundant.

In order to prove the completeness of productive proofs, we first consider the following cut-style rule where P and Q are restricted to be purely positive:

$$\frac{\Sigma :: \Gamma \vdash P \Downarrow \quad \Sigma :: \Gamma \Uparrow P, \Theta \vdash Q}{\Sigma :: \Gamma \Uparrow \Theta \vdash Q}\ cut\Updownarrow$$

Theorem 3 (Elimination of $cut\Updownarrow$). *If a reduced border sequent has a proof in $LJF^=$ plus $cut\Updownarrow$ then it has a proof in $LJF^=$ (without $cut\Updownarrow$).*

Proof. Say that an instance of $cut\Updownarrow$ has size n if there are n occurrences of logical connectives in the cut formula P. The size of a proof with possible occurrences of $cut\Updownarrow$ is the sum of the size of all occurrences of $cut\Updownarrow$ rules it contains.

If the size of a proof is 0 then any occurrence of $cut\Updownarrow$ in it involves an atomic cut formula. All such occurrences are of the form

$$\frac{\dfrac{}{\Sigma :: \Gamma \vdash A \Downarrow}\ I_r \quad \dfrac{\overset{\pi}{\Sigma :: \Gamma, A \Uparrow \Theta \vdash Q}}{\Sigma :: \Gamma \Uparrow A, \Theta \vdash Q}\ S_l}{\Sigma :: \Gamma \Uparrow \Theta \vdash Q}\ cut\Updownarrow.$$

Since A is a member of Γ (and since multiplicity in the set Γ does not matter), the endsequent of π is the same as $\Sigma :: \Gamma \Uparrow \Theta \vdash E$: hence, we can replace this derivation with π. This reduces the number of atomic instances of $cut\Updownarrow$ by one.

Assume that the size of the proof is $n > 0$. We can prove by induction on the structure of cut formulas that we can lower this measure by pushing the $cut\Updownarrow$ upwards in the usual way. □

Theorem 4 (Completeness of productive proofs). *If a reduced sequent has an $LJF^=$ proof, it has a productive $LJF^=$ proof.*

Proof. Every occurrence of an unproductive R_l can be replaced with possibly several instances of $cut\Updownarrow$. In particular, consider the following unproductive R_l rule along with the D_l rule that occurs below it in the same phase.

$$\frac{\dfrac{\dfrac{\overset{\pi}{\Sigma :: \Gamma \Uparrow P_1, \ldots, P_n, \mathcal{P} \vdash Q}}{\Sigma :: \Gamma \Downarrow P_1, \ldots, P_n, \mathcal{P} \vdash Q}\ R_l}{\vdots}}{\Sigma :: \Gamma \Downarrow N_1, \ldots, N_n, \mathcal{N} \vdash Q}\ D_l \qquad \cdots \qquad \cdots$$

Also assume that we have for $1 \leq i \leq n$, π_i is a proof of $\Gamma \vdash P_i \Downarrow$ $(n \geq 1)$ and that N_i is the antecedent of P_i. We can now reorganize this proof as follows.

$$\dfrac{\dfrac{\dfrac{\pi_1 \qquad\qquad \pi}{\Gamma \vdash P_1 \Downarrow \quad \Gamma \Uparrow P_1, \ldots, P_n, \mathcal{P} \vdash Q}\, cut\Updownarrow}{\Gamma \Uparrow P_2, \ldots, P_n, \mathcal{P} \vdash Q} \qquad \begin{array}{c}\vdots\\ \pi_n\end{array}}{\dfrac{\Gamma \vdash P_n \Downarrow \qquad\qquad \Gamma \Uparrow P_n, \mathcal{P} \vdash Q}{\Gamma \Uparrow \mathcal{P} \vdash Q}\, cut\Updownarrow} \qquad \begin{array}{c}\vdots\\ \ldots\end{array} \qquad \begin{array}{c}\vdots\\ \ldots\end{array}$$

$$\dfrac{\Gamma \Downarrow \mathcal{N} \vdash Q}{\Gamma \vdash Q}\, D_l$$

Using Theorem 3, there is then a cut-free proof of the same end sequent. Since no new occurrences of R_l rules are introduced in the cut-elimination procedure, the resulting proof is productive. □

An immediate consequence of this completeness theorem is that we only need to search for productive proofs when attempting to prove a reduced sequent.

4 Forward chaining and saturation

4.1 Eliminating the non-determinism of D_l

The search for a proof of a reduced sequent in $LJF^{=}$ begins by choosing between D_l or D_r. If D_r is chosen, then the proof will terminate within the non-invertible phase above the D_r rule, ending with one of I_r, t_r^+, or $=_r$ (see Theorem 2). Otherwise, when choosing D_l we must also choose a subset of the bipolar formulas in Γ as well as choose their multiplicity in the multiset that is the left-focus zone. In this case, the entire non-invertible left-focused phase can fail if any of the selected foci cannot be used to successfully used in the forward direction (e.g., the body of a geometric implication does not hold in the current context). While there can be clever reasons for selecting certain formulas as foci—say, because the user indicates them as part of a proof script—the general problem of what formulas to pick is not simplified by the ability to pick multiple foci. Indeed, from an implementation standpoint it is better to select the left foci one at a time, which is a non-deterministic choice with finitely many possibilities, rather than to backtrack over the infinite possibilities of selecting a multiset from a set.

We now motivate a variant of the D_l rule that becomes deterministic by having it select *all* bipolar formulas in Γ with a rule similar to the following:

$$\dfrac{\Sigma :: \Gamma \Downarrow \mathcal{N} \vdash P}{\Sigma :: \Gamma \vdash P}\, D_l,$$

where $\mathcal{N}$ is the multiset of the negative formulas in Γ (all given multiplicity 1). As written, this rule has four significant problems.

First, the multiple foci in a left-focused sequent are intended to be *independent*. If there are rigid dependencies between the foci, then the non-invertible phase would immediately fail. For example, consider the sequent $r, r \supset s, s \supset q \Uparrow \vdash \Uparrow q$, for propositional constants r, s, q. Clearly, the D_l rule should not place both $r \supset s$ and $s \supset q$ in the focus that initiates the non-invertible phase since only one of these has a premise currently holding in the proof context.

Second, if $B_1 \wedge^- B_2$ is among the left foci, then the $\wedge_l^-$ rules would replace that conjunction with either B_1 or B_2. However, it may be that *both* B_1 and B_2 can be foci simultaneously: in that case, it would be better to initially chose *two* copies of $B_1 \wedge^- B_2$ in the original focus so that both conjuncts could be explored.

The third problem, similar to the second problem, concerns geometric formulas. It might be possible to find multiple instances of the antecedent of such an implication but only one of these can contribute a successful forward-chaining step. For example, the sequent $p\ a, p\ b, \forall x.p\ x \supset q\ x \Uparrow \vdash \Uparrow q\ a \wedge^+ q\ b$ has an $LJF^=$ proof with exactly one left-focused non-invertible phase that starts by using D_l with two copies of $\forall x.p\ x \supset q\ x$ in focus. Once again, the benefit of multi-focusing is lost if we can pick only one copy of each formula in the D_l rule.

Finally, the D_l rule might lead to a non-productive R_l rule (in the sense of Section 3.2).

4.2 Refined processing of geometric implications

While the problems outlined above cannot be solved for the full logic, they can be addressed successfully when we restrict our attention to reduced border sequents. All four of these problems are addressed by changing the interpretation of left-focused sequents $\Gamma \Downarrow \Theta \vdash P$. In $LJF^=$ the context of left foci Θ is treated as linear, without allowing for any weakening or contraction. The problems above seem to indicate that this restriction is too strong, since we may need to back off from a decision to focus on a formula if we discover it has an antecedent that is not satisfied by the proof context, or to duplicate a focus if we find multiple possible uses for it during the non-invertible phase. One way to achieve this would be to add contraction and weakening to the foci Θ in such sequents:

$$\frac{\Sigma :: \Gamma \Downarrow \Theta \vdash P}{\Sigma :: \Gamma \Downarrow N, \Theta \vdash P} \qquad \frac{\Sigma :: \Gamma \Downarrow N, N, \Theta \vdash P}{\Sigma :: \Gamma \Downarrow N, \Theta \vdash P}$$

Of course, adding these rules introduces non-deterministic choice points in the proof with contraction, in particular, restoring the infinite choices problem we were attempting to avoid with our naive D_l rule.

To improve our treatment of the above problems with the naive D_l, we will eventually build a new proof system called $\mathbb{FC}$ that resembles $LJF^=$ except that it is more sophisticated in its manipulations of the left multifocus zone. We justify the design of this new proof system by addressing the problems listed above in the order of easiest to hardest.

Handling non-productivity: In Section 3.2 we classified the R_l rule as productive if and only if the focus being released is on a formula that is not immediately

provable in a single phase. This can be implemented as part of the R_l rule where we simply filter out all the provable formulas with the following rule:

$$\frac{\Sigma :: \Gamma \Uparrow Q_1, \ldots, Q_n \vdash R \quad (\Sigma :: \Gamma \vdash P_i \Downarrow)_{i=1}^{m} \quad (\Sigma :: \Gamma \nvdash Q_j \Downarrow)_{j=1}^{n}}{\Sigma :: \Gamma \Downarrow P_1, \ldots, P_m, Q_1, \ldots, Q_n \vdash R} \; R_l^d.$$

Here, the notation $\nvdash$ stands for the corresponding right-focused sequent not being provable. Note that by Theorem 2, provability of sequents of the form $\Gamma \vdash P \Downarrow$ is decidable. The items in parentheses above the horizontal line are side conditions: in particular, the proof tree that is constructed using the R_l^d rule has only one premise.

Enabling the restriction to geometric implications: The second problem identified above is already solved by the restriction of Θ to geometric implication formulas, which have no occurrences of $\wedge^-$. However, this requires a further change to D_l because the formulas in Γ are bipolar, not necessarily geometric implications. However, as we have already seen, each bipolar formula B is related to a multiset of geometric formulas $[\![B]\!] = \{N_1, \ldots, N_n\}$ such that B is logically equivalent to $N_1 \wedge^- \cdots \wedge^- N_n$ (or $\mathtt{t}^-$ if $n = 0$). Therefore, we can change the D_l rule to the rule

$$\frac{\Sigma :: \Gamma \Downarrow [\![\mathcal{N}]\!] \vdash P}{\Sigma :: \Gamma \vdash P} \; [\![D_l]\!],$$

where $\mathcal{N}$ is the multiset of the negative formulas in Γ (all given multiplicity 1).

Handling multiple proofs of antecedents: The first and third problems both deal with the logical rules in the non-invertible phase. In our case, since our left-foci are guaranteed to be geometric implications after a $[\![D_l]\!]$, we can replace the three left-introduction rule in the non-invertible phase $LJF^=$ (Figure 1) with the following rule of *forward chaining*:

$$\frac{\Sigma :: \Gamma \vdash P\theta_1 \Downarrow \quad \cdots \quad \Sigma :: \Gamma \vdash P\theta_n \Downarrow \quad \Sigma :: \Gamma \Downarrow Q\theta_1, \ldots, Q\theta_n, \Theta \vdash R}{\Sigma :: \Gamma \Downarrow \forall \bar{x}.(P \supset Q), \Theta \vdash R} \; fc$$

where each of the θ_i are *ground* substitutions for the variables $\bar{x}$. Note that that if the antecedent P is not provable with any substitution, then $n = 0$ and the effect of this rule is a weakening of the geometric implication. Likewise, for each instance of the antecedent P that is proved, the corresponding instance of the head Q is added to the foci, which handles the third problem outlined above.

Without any additional restriction, it is not always the case that there is a finite number of ground substitutions such that $\Sigma :: \Gamma \vdash P\theta \Downarrow$ is provable (which is a necessary requirement for the *fc* rule). We make one additional restriction on the structure of bipolar formulas: we require that every geometric implication $\forall \bar{x}.(A_1 \wedge^+ \cdots \wedge^+ A_n \supset P)$ in $[\![B]\!]$ is such that the free variables of its head P are also free in at least one of $A_1, \ldots, A_n$. This restriction is commonly used in logic and relational programming: in [17] such formulas are called *allowed clauses*. This restriction formally eliminates some bipolar formulas:

for example, $\forall x.(\mathtt{t}^+ \supset x \leq x)$. (Note that the empty $\wedge^+$ is $\mathtt{t}^+$.) However, the formula $\forall x.(nat\, x \supset x \leq x)$ does satisfy this restriction, where nat is an atomic predicate. Following this example, we can convert every bipolar formula into one that satisfies the allowed clause restriction by making typed quantification more explicit as predicates. That is, for every primitive type τ, we allow τ to be used also as a predicate of one argument and then replace quantifiers as follows: $\forall_\tau x.P\, x$ with $\forall x.(\tau x \supset P\, x)$ and $\exists_\tau x.P\, x$ with $\exists x.(\tau x \wedge^+ P\, x)$.

Let $\mathbb{FC}$ be the proof system that results from modifying the proof system in Figure 1 as follows: replace $\wedge_l^-$ and D_l with $[\![D_l]\!]$, replace $\forall_l$ and $\supset_l$ with fc, and replace R_l with R_l^d. We also remove all the existing left-introduction rules for left-focused sequents, which are now redundant given fc. It is easy to show that if a reduced sequent has an $\mathbb{FC}$ proof then it has an $LJF^=$ proof. The main advantage of using the $\mathbb{FC}$ proof system is that we need no cleverness in choosing the formulas to use with the D_l rule: instead we can select all available bipolar formulas for that rule, and then dynamically adjust the foci when searching for a productive proof.

A *forward-chaining phase* is a partial proof that has $[\![D_l]\!]$ as its last rule, has R_l^d as its uppermost rules, and only fc for its internal rules. A forward chaining phase is a *useless phase* if every occurrence of R_l^d hasborder sequents as premises because the R_l^d rules discards all the foci: for example,

$$\dfrac{\dfrac{\overset{\pi_1}{\Sigma :: \Gamma \Uparrow \vdash Q}}{\Sigma :: \Gamma \Downarrow \mathcal{P}_1 \vdash Q}\, R_l^d \quad \cdots \quad \dfrac{\overset{\pi_n}{\Sigma :: \Gamma \Uparrow \vdash Q}}{\Sigma :: \Gamma \Downarrow \mathcal{P}_n \vdash Q}\, R_l^d}{\dfrac{\Sigma :: \Gamma \Downarrow [\![\mathcal{N}]\!] \vdash Q}{\Sigma :: \Gamma \vdash Q}\, [\![D_l]\!]}\, fc* \,.$$

The end sequents of any of the proofs π_i matches that of the useless phase itself. The proof context $\langle \Sigma, \Gamma \rangle$ of a border sequent $\Sigma :: \Gamma \vdash Q$ is said to be *saturated* if the $\mathbb{FC}$ proof of this sequent must end in either a useless phase or D_r. Any attempt to use D_l for this sequent would be unproductive.

Example 1. Assume that i is a primitive type, f is a unary constructor for i, p is unary predicate for type i, and t is a Σ-term of type i.

- If Γ contains $p\, t$ and $\forall_i x.(p\, x \supset p\, (f\, x))$, then $\langle \Sigma, \Gamma \rangle$ is not saturated.
- If Γ is the two element set containing $p\, t$ and $\forall_i x.(p\, x \supset \exists y.p\, y)$, then $\langle \Sigma, \Gamma \rangle$ is saturated.
- If $\leq$ is a binary predicate on type i and Γ contains $p\, t$ and $\forall_i x.(p\, x \supset \exists_i y.(p\, y \wedge^+ x \leq y))$, then $\langle \Sigma, \Gamma \rangle$ is not saturated.

4.3 Safe for saturation

One of the reasons to identify saturated sequents is that they provide a simple decision procedure for proving $\mathbb{P}$ formulas. In particular, if $\langle \Sigma, \Gamma \rangle$ is saturated then determining if it entails a purely positive formula is immediate in $\mathbb{FC}$ (and hence in $LJF^=$). Every forward chaining phase that concludes with $\Sigma :: \Gamma \vdash Q$ will be useless by definition, so it will be provable if and only if $\Sigma :: \Gamma \vdash Q \Downarrow$ is provable, which is decidable by Theorem 2.

We say that a border sequent is *safe for saturation in n steps* (for a natural number n) if it has an $\mathbb{FC}$ phase that ends with a non-useless forward chaining phase, and each of the border sequents in the premises of that phase has a saturated context or is safe for saturation in $n-1$ steps. We say that the sequent is *safe for saturation* if there is an n such that it is safe for saturation in n steps. Any border sequent that is safe for saturation has decidable provability because one can simply compute repeatedly forward-chaining phases (of which there will be finitely many), and if any border sequent premise is saturated then its provability is decidable as argued below. Furthermore, there are no infinitely deep branches for that border sequent along non-useless forward chaining phases because the length of that branch (counting the instances of D_l) is bounded above by some finite number n.

The next theorem identifies a natural class of sequents that are safe for saturation.

Theorem 5. *Let $\langle \Sigma, \Gamma \rangle$ be a proof context and let $\mathcal{N}$ be the multiset of bipolar formulas in Γ. This context is safe for saturation if for every geometric implication in $[\![\mathcal{N}]\!]$ of the form $\forall \bar{x}.(P \supset Q)$ it is the case that Q is composed of only f, $\vee$, t^+, $\wedge^+$, equality, and atomic formulas without constructors of non-zero arity.*

Proof. Let the proof context $\langle \Sigma, \Gamma \rangle$ be divided into the negative (bipolar) formulas $\mathcal{N}$ and the positive (atomic) formulas $\mathcal{P}$. Define the *cover* of this proof context $\langle \Sigma, \mathcal{N}, \mathcal{P} \rangle^\circ$ as the set of atomic formulas of the form $p\ t_1\ \cdots\ t_n$ where p is any n-ary predicate occurring in formulas in either $\mathcal{N}$ or $\mathcal{P}$ and the terms $t_1, \ldots, t_n$ are either subterms of terms appearing in the atoms of $\mathcal{P}$ or constructors of arity 0 (i.e., constants). Note that $\langle \Sigma, \mathcal{N}, \mathcal{P} \rangle^\circ$ is a finite set. Now consider the following productive forward-chaining phase:

$$\dfrac{\dfrac{\dfrac{\ldots \quad \Sigma\theta :: \mathcal{N}\theta, \mathcal{P}\theta, \mathcal{P}' \vdash Q\theta \quad \ldots}{\Sigma :: \mathcal{N}, \mathcal{P} \Uparrow \mathcal{P}'' \vdash Q}\ \text{invertible phase}}{\Sigma :: \mathcal{N}, \mathcal{P} \Downarrow [\![\mathcal{N}]\!] \vdash Q}\ fc^*, R_l^d}{\Sigma :: \mathcal{N}, \mathcal{P} \vdash Q}\ [\![D_l]\!]$$

Here θ is a substitution (possibly the identity) resulting from possible occurrences of the $=_l$ rule in the invertible phase, and $\mathcal{P}'$ is a multiset of atomic formulas that are not present in $\mathcal{P}\theta$. Since this is a productive proof, either $\mathcal{P}'$ is not empty or the phase is useless (and can be collapsed). We make the following observations about the forward-chaining phase above.

1. $\langle \Sigma\theta, \mathcal{N}\theta, \mathcal{P}\theta \cup \mathcal{P}' \rangle^\circ \subseteq \langle \Sigma, \mathcal{N}, \mathcal{P} \rangle^\circ$. This follows simply from the fact that since (i) there are no existential quantifiers in the head of bipolar formulas, no new eigenvariables are introduced into the proof context and (ii) since the head of geometric clauses do not contain constructors of non-zero arity, any substitution instance of such head will be composed of subterms on the $\mathcal{P}$ component of the context.
2. Both $\mathcal{P} \subseteq \langle \Sigma, \mathcal{N}, \mathcal{P} \rangle^\circ$ and $\mathcal{P}\theta \cup \mathcal{P}' \subseteq \langle \Sigma, \mathcal{N}, \mathcal{P} \rangle^\circ$.

3. If the forward chaining phase is not useless, then $\mathcal{P}'$ is non-empty and the number of elements in $\mathcal{P}\theta \cup \mathcal{P}'$ is strictly greater than the number of elements in $\mathcal{P}$ (ignoring multiplicity).

As a result of these observations, the number of not useless forward-chaining phases must be limited by the size of the set $\langle \Sigma, \mathcal{N}, \mathcal{P} \rangle^\circ$. □

Theorem 5 provides a useful criterion for determining when one can expect the saturation process to terminate. Undoubtedly, many other criteria can also be developed.

4.4 Implementing a forward chaining tactic (in Abella)

We have implemented a forward chaining tactic based on $\mathbb{FC}$ in the Abella [4] interactive theorem prover.[3] Abella is based on a variant of the $\mathcal{G}$ logic [11], which is an extension of first-order logic with inductive and co-inductive fixed points, the ∇ quantifier [22] and nominal constants for reasoning about binding structures in λ-terms, and a generalization of the usual $\alpha\beta\eta$ equational theory of λ-terms to also incorporate *equivariance* of nominal constants.

The implementation of the $[\![D_l]\!]$ rule of $\mathbb{FC}$ has to be adapted for the implementation because Abella is intended as a general purpose system where we cannot *a priori* limit the shape of the elements of Γ: users are allowed to write any theorems they want, even those that are not bipolar formulas. We implement a variant of the system where the user picks the specific multiset of lemmas to decide on, and the geometric implication set $[\![B]\!]$ of every lemma B is computed on the fly. The implementation also does not currently detect whether the lemmas the user indicates for saturation are safe for saturation. Instead, the saturation mechanism is run with an implicit finite *phase height* bound, which bounds the number of instances of D_l needed on any branch of the derivation.

Another key difference with $\mathbb{FC}$ is that the implementation does not keep the context Γ as a set. This is mainly for performance reasons, since transforming a multiset into a set is a worst-case quadratic operation that should therefore not be done too often.

5 Related work

Marin et al. [18] showed how focused proof systems can be used to account for a range of synthetic inference rules that appeared in the literature. In particular, the work of Negri in [25, 26] was accounted for by polarizing atomic formulas positively (as done in this paper). Synthetic inference rules developed by Viganò [28] are different and can be accounted for by polarizing atoms negative. Negatively polarized atomic formulas can also be used to describe the two phases of uniform proofs (goal-reduction and backchaining) in the setting of logic programming [20].

Although this paper mostly deals with only bipolar formulas and geometric implications, such formulas appear in many situations. For example, Horn clauses are clearly bipolar formulas, as are many axioms describing algebraic structures.

[3] See `https://abella-prover.org/fc` for a page dedicated to this implementation, with installation instruction and browsable examples.

If one examines published proof developments in, say, Abella one finds that a large majority of theorems proved in these settings are bipolar. For example, every theorem in the development of a library for multisets (https://github.com/meta-logic/abella-reasoning/tree/master/lib) is not only a bipolar formula but also a geometric implication. Even in more meta-theoretic papers, such as [1, 2], the collection of theorems is either entirely or almost all bipolar formulas. There are also results that describe how it is possible to convert non-bipolar formulas into geometric implications if one is willing to add new predicate constant [9].

Saturation has been used as a computational device not only in the setting of Datalog [24] but also in the setting of *logical algorithms* [13, 12].

6 Conclusion

This paper introduces a proof-theoretic framework for defining forward chaining and saturation within a multisorted, first-order intuitionistic logic with equality. Key to this framework are the notions of polarity and focused proofs, which provide a principled interpretation of forward chaining and *productive proofs*. We also define the notion of *safe for saturation* under which forward chaining is guaranteed to terminate and exhibit a natural class of formulas that are shown to be safe in this sense. This theoretical analysis motivates and supports the implementation of a novel forward chaining tactic within the Abella theorem prover, which has been shown to reduce manual proof effort by automating lemma applications and facilitating tasks like congruence closure and various equational and relational reasoning.

References

1. Accattoli, B.: Proof pearl: Abella formalization of lambda calculus cube property. In: Hawblitzel, C., Miller, D. (eds.) Second International Conference on Certified Programs and Proofs. LNCS, vol. 7679, pp. 173–187. Springer (2012)
2. Accattoli, B., Blanc, H., Coen, C.S.: Formalizing Functions as Processes. In: Naumowicz, A., Thiemann, R. (eds.) 14th International Conference on Interactive Theorem Proving (ITP 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 268, pp. 5:1–5:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2023). https://doi.org/10.4230/LIPIcs.ITP.2023.5, https://drops.dagstuhl.de/opus/volltexte/2023/18380
3. Andreoli, J.M.: Logic programming with focusing proofs in linear logic. J. of Logic and Computation **2**(3), 297–347 (1992). https://doi.org/10.1093/logcom/2.3.297
4. Baelde, D., Chaudhuri, K., Gacek, A., Miller, D., Nadathur, G., Tiu, A., Wang, Y.: Abella: A system for reasoning about relational specifications. J. of Formalized Reasoning **7**(2), 1–89 (2014). https://doi.org/10.6092/issn.1972-5787/4650
5. Bancilhon, F., Maier, D., Sagiv, Y., Ullman, J.D.: Magic sets and other strange ways to implement logic programs. In: Silberschatz, A. (ed.) Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. pp. 1–15. ACM (1986). https://doi.org/10.1145/6012.15399, https://doi.org/10.1145/6012.15399
6. Chaudhuri, K., Hetzl, S., Miller, D.: A multi-focused proof system isomorphic to expansion proofs. J. of Logic and Computation **26**(2), 577–603 (2016). https://doi.org/10.1093/logcom/exu030

7. Chaudhuri, K., Miller, D., Saurin, A.: Canonical sequent proofs via multi-focusing. In: Ausiello, G., Karhumäki, J., Mauri, G., Ong, L. (eds.) Fifth International Conference on Theoretical Computer Science. IFIP, vol. 273, pp. 383–396. Springer (Sep 2008). https://doi.org/10.1007/978-0-387-09680-3˙26
8. Chaudhuri, K., Pfenning, F., Price, G.: A logical characterization of forward and backward chaining in the inverse method. J. of Automated Reasoning **40**(2-3), 133–177 (2008). https://doi.org/10.1007/s10817-007-9091-0
9. Dyckhoff, R., Negri, S.: Geometrisation of first-order logic. The Bulletin of Symbolic Logic **21**(02), 123–163 (2015)
10. Flanagan, C., Sabry, A., Duba, B.F., Felleisen, M.: The essence of compiling with continuations. ACM SIGPLAN Notices **28**(6), 237–247 (1993). https://doi.org/10.1145/155090.155113
11. Gacek, A., Miller, D., Nadathur, G.: Nominal abstraction. Information and Computation **209**(1), 48–73 (2011). https://doi.org/10.1016/j.ic.2010.09.004
12. Ganzinger, H., McAllester, D.: Logical algorithms. In: Proc. International Conference on Logic Programming (ICLP) 2002. LNCS, vol. 2401, pp. 209–223. Springer (2002)
13. Ganzinger, H., McAllester, D.A.: A new meta-complexity theorem for bottom-up logic programs. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) First International Joint Conference Automated Reasoning (IJCAR). LNCS, vol. 2083, pp. 514–528. Springer (2001)
14. Gentzen, G.: Investigations into logical deduction. In: Szabo, M.E. (ed.) The Collected Papers of Gerhard Gentzen, pp. 68–131. North-Holland, Amsterdam (1935). https://doi.org/10.1007/BF01201353, translation of articles that appeared in 1934-35. Collected papers appeared in 1969.
15. Gérard, U., Miller, D.: Separating functional computation from relations. In: Goranko, V., Dam, M. (eds.) 26th EACSL Annual Conference on Computer Science Logic (CSL 2017). LIPIcs, vol. 82, pp. 23:1–23:17 (2017). https://doi.org/10.4230/LIPIcs.CSL.2017.23
16. Liang, C., Miller, D.: Focusing and polarization in linear, intuitionistic, and classical logics. Theoretical Computer Science **410**(46), 4747–4768 (2009). https://doi.org/10.1016/j.tcs.2009.07.041, Abstract Interpretation and Logic Programming: A Special Issue in Honor of Professor Giorgio Levi
17. Lloyd, J., Topor, R.: A basis for deductive database systems ii. Journal of Logic Programming **3**(1), 55–67 (1986). https://doi.org/10.1016/0743-1066(86)90004-X
18. Marin, S., Miller, D., Pimentel, E., Volpe, M.: From axioms to synthetic inference rules via focusing. Annals of Pure and Applied Logic **173**(5), 1–32 (2022). https://doi.org/10.1016/j.apal.2022.103091
19. McDowell, R., Miller, D.: Cut-elimination for a logic with definitions and induction. Theoretical Computer Science **232**, 91–119 (2000). https://doi.org/10.1016/S0304-3975(99)00171-1
20. Miller, D.: A survey of the proof-theoretic foundations of logic programming. Theory and Practice of Logic Programming **22**(6), 859–904 (Oct 2022). https://doi.org/10.1017/S1471068421000533, published online November 2021
21. Miller, D., Nadathur, G., Pfenning, F., Scedrov, A.: Uniform proofs as a foundation for logic programming. Annals of Pure and Applied Logic **51**(1–2), 125–157 (1991). https://doi.org/10.1016/0168-0072(91)90068-W
22. Miller, D., Tiu, A.: A proof theory for generic judgments. ACM Trans. on Computational Logic **6**(4), 749–783 (Oct 2005). https://doi.org/10.1145/1094622.1094628
23. Miller, D., Wu, J.H.: A positive perspective on term representations. In: Klin, B., Pimentel, E. (eds.) 31st EACSL Annual Conference on Computer Science Logic

(CSL 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 252, pp. 3:1–3:21. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2023). https://doi.org/10.4230/LIPIcs.CSL.2023.3
24. Naughton, J.F., Ramakrishnan, R.: Bottom-up evaluation of logic programs. In: Lassez, J., Plotkin, G.D. (eds.) Computational Logic - Essays in Honor of Alan Robinson. pp. 640–700. The MIT Press (1991)
25. Negri, S.: Contraction-free sequent calculi for geometric theories with an application to Barr's theorem. Archive for Mathematical Logic **42**, 389–401 (2003). https://doi.org/10.1007/s001530100124
26. Negri, S., von Plato, J.: Cut elimination in the presence of axioms. Bulletin of Symbolic Logic **4**(4), 418–435 (1998). https://doi.org/10.2307/420956
27. Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory. Cambridge University Press, 2nd edn. (2000)
28. Viganò, L.: Labelled Non-Classical Logics. Kluwer Academic Publishers (2000)

A Examples of forward chaining

A.1 Simple closure properties

Consider a simple finite graph structure $\langle N, E\rangle$ where N is a set of nodes and $E \subseteq N \times N$. To encode this, let the primitive type g denote nodes, and let Σ assign to each member of N the type g. Also, let adj and $path$ be binary predicates over type g. The set $\mathcal{A} = \{node\ x \mid x \in N\} \cup \{adj\ x\ y \mid \langle x, y\rangle \in E\}$ encodes the adjacency information of the given graph. Now consider the following geometric implications.

$$\forall_g x.\forall_g y.(adj\ x\ y \supset path\ x\ y)$$
$$\forall_g x.(node\ x \supset path\ x\ x)$$
$$\forall_g x.\forall_g y.(path\ x\ y \supset path\ y\ x)$$
$$\forall_g x.\forall_g y.\forall_g z.((path\ x\ y \wedge^{+} path\ y\ z) \supset path\ x\ z)$$

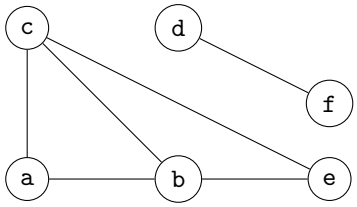
By Theorem 5, the proof context that contains Σ, $\mathcal{A}$, and any subset of these four formulas is safe for saturation. Proof search in $\mathbb{FC}$ can decide whether two nodes in N are in the same connected component or not.

In Abella: In Abella the above signature and lemmas can be added to the context using `Kind`, `Type`, and `Theorem` declarations as follows:

```
Kind g               type.
Type node   g -> prop.
Type adj    g -> g -> prop.
Type path   g -> g -> prop.

Theorem refl : forall n, node n -> path n n.
Theorem trans : forall x y z, adj x y -> path y z -> path x z.
```

Suppose we want to encode the following graph and determine all the nodes reachable from `a`.



In Abella we would do this by declaring the structure as lemmas.

```
Type a, b, c, d, e, f   g. % declare the nodes as new constants
Theorem nodes : node a /\ node b /\ node c /\ node d /\ node e /\ node f.
Theorem adjs : adj a b /\ adj a c /\ adj b c /\ ··· /\ adj d f.
```

Say we want to prove that there is a path from, say, `a` to `e`, which in ordinary Abella looks like the following script:

```
Theorem example1 : path a e.
apply nodes. apply adjs.
apply refl to H5.
apply trans to H10 H13.
apply trans to H7 H14.
search.
```

This proof is tedious to write and difficult to maintain since minor changes in an unrelated part of the graph would alter the numbering of hypotheses. The new `fchain` command offers a much shorter and more flexible proof script.

```
Theorem example1_fchain : path a e.
fchain 6 nodes adjs refl trans.
search.
```

This use of `fchain` generates 28 consequences, with one of them being the desired goal. As explained in Section 4.4, the implementation of the `fchain` tactic uses a depth bound instead of checking that the declared lemmas are safe for saturation. With this depth bound, Abella automatically computes the entire `path` relation for this graph.

Of course, `fchain` can make an even greater difference when the graph structure grows. This use of the `fchain` command is invoked with a depth bound of 6 and a selection of four theorems to use for forward chaining (all of which are safe for saturation). In general, any larger number than 6 can be used since the implementation stops forward chaining when it detects saturation, i.e., when all future forward chaining phases would be useless. The list of lemmas used for forward chaining could be larger as well without a loss of completeness.

A.2 Reasoning in group theory

If one attempts to reason about algebraic structures (such as semigroups, monoids, and groups) abstractly, one needs to encode the operators-as-function routinely used in algebra instead via operators-as-relations. One also needs to accept various axioms about those relations. For example, the following proof script illustrates the introduction of some simple aspects of group theory via four predicates: `group` is the predicate that denotes the carrier of the set, `times` denotes the group operation, `unit` denotes the unit, and `inv` denotes the inverse operation. Some of the associate axioms are also introduced into the proof context.

```
Kind group    type.
Type group    group -> prop.
Type times    group -> group -> group -> prop.
Type unit     group -> prop.
Type inv      group -> group -> prop.

Theorem times-types :
 forall x y z, times x y z -> (group x /\ group y /\ group z).
Theorem times-total :
 forall x y, group x -> group y -> exists u, group u /\ times x y u.
Theorem times-determ : forall x y u v, times x y u -> times x y v -> u = v.
Theorem times-assoc : forall x y xy u xyu yu, times x y xy ->
                      times xy u xyu -> times y u yu -> times x yu xyu.
Theorem unit-total : exists u, group u /\ unit u.
Theorem unit-rule :
 forall u x, unit u -> group x -> times x u x /\ times u x x.
Theorem inv-total :
 forall x, group x -> exists i, group i /\ inv x i.
Theorem inv-rule :
 forall u x y, unit u -> inv x y -> times x y u /\ times y x u.
```

Note that while the theorem `unit-total` is a purely positive formula it is logically equivalent to the bipole

```
Theorem unit-total : true -> exists u, group u /\ unit u.
```

Also note that all these theorems are safe for saturation except for `times-total`.

Given this situation, it is possible to prove the following three example theorems using saturation with all but `times-total`: this lemma must be applied via explicit applications.

```
Theorem times-assoc-left : forall x y u yu xyu xy, times y u yu ->
                 times x yu xyu -> times x y xy -> times xy u xyu.
intros.
fchain times-types.
apply times-total to _ _ with x = xy, y = u.
fchain 2 times-assoc times-determ.
search.

Theorem cancel-left : forall x a b y, times x a y -> times x b y -> a = b.
intros.
fchain 2 unit-total times-types.
apply inv-total to H3.
fchain inv-rule.
apply times-total to _ _ with x = i, y = y.
apply times-total to _ _ with x = u, y = a.
apply times-total to _ _ with x = u, y = b.
fchain 2 times-determ unit-rule times-assoc.
search.

Theorem inv-of-times : forall x y xy i xi yi j, times x y xy ->
           inv xy i -> inv x xi -> inv y yi -> times yi xi j -> i = j.
intros.
fchain 3 unit-total times-types times-types inv-rule.
apply times-total to _ _ with x = xy, y = j.
apply times-total to _ _ with x = y, y = j.
apply times-total to _ _ with x = xy, y = yi.
fchain 1 times-assoc-left times-assoc times-assoc-left
           unit-rule times-determ cancel-left.
fchain 3 times-determ cancel-left.
search.
```

The uses of `times-total` are used to introduce into the proof context explicit term constructions. For example, the proof that $(xy)^{-1} = y^{-1}x^{-1}$ is a simple argument once one first invents the term $(xy)(y^{-1}x^{-1})$, which is done using `times-total`. Observe that the final proof scripts are largely free of any mention of hypotheses created during the proof. In particular, nearly every application of a lemma is automated by `fchain`. This makes the proof script quite resilient to changes in the specification, which might otherwise have changed the number or order of hypotheses and would have therefore required manually updating existing proof scripts.

B Application: Relational administrative normal form

B.1 Replacing constructors with relations

Closure of binary relations on terms based on congruences is a common and useful operation. For example, if we have a binary relation r on terms that we wish to close under the application of the unary constructor f, we would add the assumption $\forall_i x.\forall_i y.(r\ x\ y \supset r\ (f\ x)\ (f\ y))$ to the proof context.[4] This geometric implication is not safe for saturation. While forward chaining with this kind of

[4] In the rest of this section, we assume that we assume there is exactly one primitive type, say, i and assume that "term" and "term of type i" mean the same.

formula is certainly possible and useful in many situations, we can use another approach to the construction of terms that can benefit from our analysis of forward chaining and saturation.

The *administrative normal form* (ANF) used within some compiler to represent term structures [10] can be used to give an alternative representation of terms using first-order quantified formulas that do not contain constructors [15, 23]. By way of example, consider the atomic formula $(2 * (5 + 2)) < 8 + 7$ involving the less-than predicate and terms denoting natural numbers. The ANF form of this expression is the following kind of inverted syntax expression

$$\textbf{name } x = 5 + 2 \textbf{ in name } y = 2 * x \textbf{ in name } z = 8 + 7 \textbf{ in } y < z.$$

Here, constructors such as $+$ and $*$ are applied to constants and variables but not to more complex term structures. We can replace the function symbols $+$ and $*$ by introducing the predicates *plus* and *times* so that $\mathit{plus}\ x\ y\ z$ denotes $x + y = z$ and $\mathit{times}\ x\ y\ z$ denotes $x * y = z$. In order for these predicates to capture their intended functional meaning, they must satisfy the following *totality* formulas

$$\forall x \forall y.(\mathit{nat}\ x \wedge \mathit{nat}\ y) \supset \exists z.\mathit{nat}\ z \wedge \mathit{plus}\ x\ y\ z$$
$$\forall x \forall y.(\mathit{nat}\ x \wedge \mathit{nat}\ y) \supset \exists z.\mathit{nat}\ z \wedge \mathit{times}\ x\ y\ z$$

as well as the following *determinacy* formulas

$$\forall x \forall y \forall u \forall v.(\mathit{plus}\ x\ y\ u \wedge \mathit{plus}\ x\ y\ v) \supset u = v$$
$$\forall x \forall y \forall u \forall v.(\mathit{times}\ x\ y\ u \wedge \mathit{times}\ x\ y\ v) \supset u = v$$

Note that these four formulas can be polarized as bipolar formulas and that when they both hold, we have the following ambiguity of quantifications:

$$\forall x \forall y.[(\forall z.\mathit{plus}\ x\ y\ z \supset Q\ z) \equiv (\exists z.\mathit{plus}\ x\ y\ z \wedge Q\ z)]$$

(also holds if *plus* is replaced by *times*). We can rewrite the ANF expression above as the purely positive formula and the bipolar formula

$$\exists x(\mathit{plus}\ 5\ 2\ x \wedge^{+} \exists y(\mathit{times}\ 2\ x\ y \wedge^{+} \exists z(\mathit{plus}\ 8\ 7\ z \wedge^{+} y < z)))$$
$$\forall x(\mathit{plus}\ 5\ 2\ x \supset \forall y(\mathit{times}\ 2\ x\ y \supset \forall z(\mathit{plus}\ 8\ 7\ z \supset y < z))).$$

Given the ambiguity mentioned above, these expressions are logically equivalent. While totality and determinacy are only two of many other properties that are true of addition and multiplication (e.g., commutativity, associativity, distributivity, etc), we single out these properties for the sake of the following examples.

Using the ANF representation of term structures, we can represent terms using relations in the following sense. Every m-ary constructor f can be encoded as using a $m+1$-ary predicate P_f so that the atomic formula $p_f\ (f\ t_1 \cdots t_m)\ t_1 \ldots t_m$ holds for all terms $t_1, \ldots, t_m$. Clearly, both determinacy and totality properties hold for such predicates representing constructors in this fashion.

B.2 Congruence closure

Mixing the ANF representation of terms as quantified formulas with saturation provides a simple (and naive) implementation of congruence closure. We illustrate how this is possible with the following example.

Example 2. Let a, b, c be constructors of arity 0, and let g and f be constructors of arity 1 and 2, respectively. We wish to prove the entailment

$$ga = b, fab = c, fbb = gc \vdash g(fa(ga)) = fbb.$$

This entailment can be converted to ANF and written as the following formula.

$$\begin{gathered}
\forall\ x_a.p_a\ x_a \supset \forall\ x_b.p_b\ x_b \supset \forall\ x_c.p_c\ x_c \supset \\
\forall\ x_{ga}.p_g\ x_{ga}\ x_a \supset \forall\ x_{gc}.p_g\ x_{gc}\ x_c \supset \\
\forall\ x_{fab}.p_f\ x_{fab}\ x_a\ x_b \supset \forall\ x_{fbb}.p_f\ x_{fbb}\ x_b\ x_b \supset \forall\ x_{faga}.p_f\ x_{faga}\ x_a\ x_{ga} \supset \\
\forall\ x_{gfaga}.p_g\ x_{gfaga}\ x_{faga} \supset \\
x_{ga} = x_b \supset x_{fab} = x_c \supset x_{fbb} = x_{gc} \supset x_{gfaga} = x_{fpbb}.
\end{gathered}$$

Attempting a proof of this formula starts with the invertible phase, which reduces this to an attempt to prove the sequent $\Sigma :: \Gamma \vdash x_{gfaga} = x_{gc}$, where Σ is $\{x_a, x_b, x_c, x_{gc}, x_{faga}, x_{gfaga}\}$ and Γ is $\{p_a\ x_a,\ p_b\ x_b,\ p_c\ x_c,\ p_g\ x_b\ x_a,\ p_g\ x_{gc}\ x_c,\ p_f\ x_c\ x_a\ x_b,\ p_f\ x_{gc}\ x_b\ x_b,\ p_f\ x_{faga}\ x_a\ x_b,\ p_g\ x_{gfaga}\ x_{faga}\}$ (Recall that the invertible phase solves the equality $x_{ga} = x_b,\ x_{fab} = x_c,\ x_{fbb} = x_{gc}$ by, say, instantiating the second variable with the first.) The formula

$$\forall x \forall y \forall u \forall v.(p_f\ u\ x\ y \wedge^+ p_f\ v\ x\ y) \supset u = v$$

states the determinacy of the constructor f. Forward chaining using this formula will force x_c to replace x_{faga}. Forward chaining using determinacy for g forces x_{gfaga} to be X_{gc}. At this point, the goal equality is now trivial.

The theorem named `example` below encodes that statement and proof of the entailment $ga = b, fab = c, fbb = gc \vdash g(fa(ga)) = fbb$. First, we declare five constructors, and five predicates used to encode those constructors, then five lemmas declaring those predicates to support the determinancy of using constructors.

```
Kind i       type.
Type a,b,c   i.
Type g       i -> i.
Type f       i -> i -> i.

Type aa, bb, cc    i -> prop.
Type gg            i -> i -> prop.
Type ff       i -> i -> i -> prop.

Theorem a-determ : forall A B, aa A -> aa B -> A = B.
Theorem b-determ : forall A B, bb A -> bb B -> A = B.
Theorem c-determ : forall A B, cc A -> cc B -> A = B.
Theorem f-determ : forall A B X Y, ff A X Y -> ff B X Y -> A = B.
Theorem g-determ : forall A B X, gg A X -> gg B X -> A = B.
```

The following theorem encodes the equality entailment above using the structure of ANF. The proofs then use forward reasoning with only these determinancy lemmas.

```
Theorem example :
  forall Xa, aa Xa -> forall Xb,  bb Xb -> forall Xc,  cc Xc ->
  forall Xga, gg Xga Xa -> forall Xgc,  gg Xgc Xc ->
  forall Xfab, ff Xfab Xa Xb ->
  forall Xfbb, ff Xfbb Xb Xb ->
  forall Xfaga, ff Xfaga Xa Xga ->
  forall Xgfaga, gg Xgfaga Xfaga ->
     Xga = Xb -> Xfab = Xc -> Xfbb = Xgc -> Xgfaga = Xfbb.
intros. case H10. case H11. case H12. % Remove equations
fchain 2 f-determ g-determ. search.
```

Without using `fchain`, this last line needs to be more specific.

```
apply f-determ to H8 H6.
apply g-determ to H9 H5. search.
```

Of course, with larger examples, this difference is more striking.