Compact Proof Certificates for Linear Logic

Kaustuv Chaudhuri

Compact Proof Certificates For Linear Logic

Kaustuv Chaudhuri
INRIA, France
http://kaustuv.chaudhuri.info

December 3, 2012

Abstract

Linear logic is increasingly being used as a tool for communicating reasoning agents in domains such as authorization, access control, electronic voting, etc., where proof certificates represent evidence that must be verified by proof consumers as part of higher protocols. Controlling the size of these certificates is critical. We assume that the proof consumer is allowed to do some search to reconstruct details of the full proof that are omitted from the certificates. Because the decision problem for linear logic is unsolvable, the certificate must contain at least enough information to bound the search: we show how to use the sequence of contractions in the sequent proof for this bound. The remaining content of the proof, in particular the information about resource divisions, can then be omitted from the certificate. We also describe a technique for giving a variable amount of additional search hints to the proof consumer to limit its non-determinism.

1 Introduction

A *proof certificate* is a way for a *proof producer* to convey certainty about a theorem to a *proof consumer*. It is the embodiment of a compromise between the size of the certificate (which directly correlates to the difficulty of processing, transmitting, and storing the certificate) and the complexity of checking (and hence trusting) it. A fully detailed proof can be very large, but it might be checkable by a simple and trustworthy checker (the so-called *De Bruijn Criterion*). On the other extreme, the certificate might just record a "yes", and the consumer must reconstruct the entire proof. This paper explores some of the spectrum between these two extremes and provides some guidelines for producing certificates that have a *tunable* amount of detail. Of particular interest is the kind of certificate where the level of detail can be modified by intermediaries between the ultimate producer and consumer, an idea that is at the heart of the "marketplace of proofs" concept [16].

To be simple and concrete, this paper considers this question for classical linear logic, which is undecidable even in the propositional fragment [15]. The ideas readily extend to the first-order and to intuitionistic and classical logic, where many of the issues are simpler. The underlying proof system will be a *sequent calculus*. Sequent calculi are ideally suited for proof-search in many logics for at least two important reasons: first, the subformula property, which is the *sine qua non* of automation; and second, polarity and focusing, wherein the ordinary sequent rules coalesce into synthetic "macro" forms to make larger logical steps without sacrificing completeness [1]. The careful use of focusing enables a general search strategy to implement a wide variety of operational strategies directly [8, 6].

The sequent calculus is nevertheless not an ideal certificate format. It is a simple matter to build a syntax for fully detailed sequent proofs. If the proof is of an important mathematical result, then such detailed proofs can perhaps be tolerated as they are unlikely to be consumed often. Commonly, however, automatically generated proofs are used in domains where the proofs are intended to convince the consumer of some semantic property of digital artefacts, such as conformance to security policies. Probably the best example is proof-carrying code (PCC) [18]. Certificates in such domains are meta-information and generally considered to be overhead. For PCC, the standard technique for reducing the overhead is to specify the search semantics for the consumer (*e.g.*, by means of a logic programming language) and then to record the choices needed to guide the consumer in the form of oracle strings. While this may be a good engineering solution for the specific problem of PCC, oracle strings are an unsatisfactory proof certificate format in general. The strings must match the operational semantics of the consumer, for one, which limits the portability and maintainability of proofs. They are also denotationally opaque.

An alternative to the oracles approach—the one used in this paper—is to *elide* some of the details in the proof if they can be *dependably* reconstructed by the consumer. This is to say that the elision must be such that reconstructing the full sequent proof always remains at least decidable, and preferably feasible and predictable. A good example of this approach is the Dedukti system [2] where purely computational steps in a proof are elided because the consumer is equipped with a general rewrite engine. However, we want more than just the ability to omit certain classes of sub-proofs: we want the level of detail in a proof to be fully variable. Linearity also brings its own problems to such approaches: the key issue is that linear proofs *consume* resources,[1] so an omitted proof would consume an unknown amount of resources. Reconstructing the resource divisions is the well known *resource management problem* [10, 11], which is unsolvable in general as it is the source of the undecidability of linear logic.

What details are essential in a sequent proof? Every logical rule introduces some connective, so applying the rule from the conclusion to the premises *consumes* the principal formula. Of the structural rules, the only rule that strictly increases the complexity of a sequent is contraction. The contraction-free fragment of the sequent calculus is manifestly decidable, assuming the subformula relation is well-founded and no rule has infinitely many premises – reasonable assumptions for logics where automated reasoning is feasible. The proof-theoretic impact of restricting or removing the contraction rules is a long studied field (see, *e.g.*, [9]), but the following deceptively simple observation seems to be either missing or vanishingly unpopular in the literature on proof objects: *the sequence of contractions in a sequent proof is a sufficient certificate.*

One reason why this observation might be rare is that it is unclear how to isolate the contractions from the other rules in the sequent calculus. Other proof calculi, such as the *calculus of structures* [19, 7], permit inferences in any formula context and thus allow the contraction rule to permute freely. Separating the contractions from an arbitrary proof is routine in such formalisms and generally forms the basis of much of their meta-theory. From the perspective of proof search, this extra permutative freedom is actually detrimental, for the proof search trees are much more branching. But, once a proof is built, it can be freely reorganized. The record of contractions is generally much smaller than the full proof—which needs to record the contractions anyhow—because it lacks all the logical content.

Now, while the contraction rules in the sequent calculus obviously do not permute, what is important is not the ability to permute but a mechanism to *pre-compute* the contractions. This pre-computed information about just the contractions can then be used to control their application in a general search strategy, so they form a suitable proof certificate. To extract this information, we require a mechanism for uniquely indexing every subformula in a sequent.

[1] A *resource* is a hypothesis that can disappear upon use.

This is not particularly hard: we merely give a unique name to every subformula path in the sequent and perform some additional book-keeping to ensure that the names remain unique through applications of sequent rules. The contractions then manifest as subformula paths that can potentially be duplicated; this information can then be used as a *bound* on search, wherein every application of contraction "consumes" one of the copies of the replicable subformula paths.

Bounding the contractions in this manner makes proof search (and hence reconstruction in the consumer) decidable, but it does not necessarily make it feasible. As already mentioned, a good certificate format should allow a variable level of detail to control the non-determinism in the consumer. It turns out that, carefully done, the indexing mechanism used for the contractions can then be exploited in another important manner: the tree of names of the principal formulas in the sequent proof is also a suitable certificate format. Moreover, this tree can serve as a skeleton from which to hang the information about contractions. This is the main technical contribution of this paper.

We begin by limiting ourselves to focused proofs [1, 8, 14]. The essence of focusing is to reduce the subformula relation to one that clarifies the alternation between invertible ("don't care") and non-invertible ("don't know") choices in search. Only these points of *phase shift* need to be indexable. Of the phase shifts, the most important ones are those that *decide* to focus on a particular formula: the corresponding labels are recorded in the *decision tree* of labels. This crystallizes the common intuition that the essence of a focused proof is the sequence of decisions; the details of the choices *inside* particular phases of focusing are unimportant. In fact, the other non-deterministic choices in a proof, *viz.* the resource distribution and the disjunctive choices, can be recovered directly from this decision tree. Lastly, at any level in the decision tree we can simply replace the sub-tree with a bound on the contractions, which gives us certificates with a variable level of detail.

To summarize, our prescription for obtaining compact proof certificates is as follows: (1) start from a focused sequent proof; (2) uniquely name every subformula path in the end-sequent; (3) extract the tree of names for the principal formulas in the decision rules; and (4) replace some of the sub-trees in this tree by a suitable search bound, such as one on the contractions. We begin with an overview of the focused sequent calculus (Sect. 2), then we show how to label the sequents and tame contraction (Sect. 3), and finally we describe how to use the decision trees as determinizing hints for proof reconstruction (Sect. 4).

2 Background

We use propositional linear logic in this paper, though the technique extends straightforwardly to the first-order. It also extends to ordinary logic (intuitionistic or classical) where the problems are simpler than the linear case. Formulas (written $A, B, \dots$) have the following grammar.

$$\begin{aligned} A, B, \dots ::= {} & a \mid A \otimes B \mid \mathbf{1} \mid A \oplus B \mid \mathbf{0} \mid !A \\ & \mid \overline{a} \mid A \parr B \mid \bot \mid A \,\&\, B \mid \top \mid ?A \end{aligned}$$

Atomic formulas are written using $a, b, \dots$, and the negation of a is written as $\overline{a}$. Formulas are in negation-normal form with each vertical column in the above grammar depicting one De Morgan dual pair; we write $(A)^\perp$ for the dual of A. The linear sequent calculus, which we call LLK, is usually presented using one-sided sequents of the form $\vdash \Gamma$, where Γ is a multi-set of formulas. The rules of this system are in Fig. 1

Looking at the permutations of rules in LLK, it is easy to see that some rules can always permute because they are invertible (*i.e.*, if the conclusion of the rule is true, then so is the conjunction of its premises), while other (non-invertible) rules permute only in specific circumstances. Andreoli famously showed that these permutation properties can be exploited to define a

$$\frac{}{\vdash \overline{a}, a}\,\mathsf{ai} \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\otimes \qquad \frac{}{\vdash \mathbf{1}}\,\mathbf{1} \qquad \frac{\vdash \Gamma, A_i}{\vdash \Gamma, A_1 \oplus A_2}\oplus \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\parr \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot}\bot$$

$$\frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}\,\& \qquad \frac{}{\vdash \Gamma, \top}\top \qquad \frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,! \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\,? \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\,\mathsf{ct} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\,\mathsf{wk}$$

Figure 1: Rules of LLK. In the $\oplus$ rule, $i \in \{1, 2\}$.

restricted but complete class of sequent proofs that follow a *focusing* discipline [1]. The essence of the discipline is that the ordinary (unfocused) rules of LLK naturally coalesce into larger clumps of derived rules that abstract over the irrelevant details such as the order of application of invertible rules. Focusing was originally an operational device to control the non-determinism in proof-search in linear logic programming, but it is now seen as a general device for analyzing the structural properties of proof systems, akin to cut-elimination for the sequent calculus. Focused proof systems have been formulated for a number of other logics [8, 14] and proof systems [3, 7] besides its origin in the classical linear sequent calculus.

The standard presentation of focusing for LLK is as follows, in broad outline. The non-atomic formulas of linear logic divide evenly into those that have invertible rules and those that do not; moreover, the two sets of formulas are De Morgan duals. Following general tradition, we call the connectives with invertible logical rules *negative*, and those with non-invertible rules *positive*. The atomic formulas can be placed into either class as long as duality is maintained, but we follow tradition and classify them as positive and their negations as negative. We also adopt the device of *polarization* [12], wherein the changes between positive and negative subformulas is explicitly marked with *shift* connectives ($\downarrow$ and $\uparrow$). While the choice of a polarized syntax is usually unnecessary for focusing, we will exploit it for our certificates.

Polarized formulas have the following grammar.

$$\begin{aligned} P, Q, \ldots &::= a \mid P \otimes Q \mid \mathbf{1} \mid P \oplus Q \mid \mathbf{0} \mid !N \mid \downarrow N && \text{(positive formulas)} \\ N, M, \ldots &::= \overline{a} \mid N \parr M \mid \bot \mid N \,\&\, M \mid \top \mid ?P \mid \uparrow P && \text{(negative formulas)} \end{aligned}$$

We will continue to use $A, B, \ldots$ to refer to formulas of either polarity. The sequents in the focused variant of LLK, which we call LLKF, have one the following two forms.

$$\begin{aligned} &\vdash \Gamma; \Delta; \left[P\right] && \text{(positive sequent with } P \textit{ focused)} \\ &\vdash \Gamma; \Delta; \Omega && \text{(negative sequent with } \Omega \textit{ active)} \end{aligned}$$

The contexts Γ, Δ and Ω are all multi-sets of the following kinds of formulas.

$$\Gamma ::= \cdot \mid \Gamma, P \qquad \Delta ::= \cdot \mid \Delta, P \mid \Delta, \overline{a} \qquad \Omega ::= \cdot \mid \Omega, N$$

Although all three contexts are multi-sets, semantically the context Γ is *unrestricted* (admitting weakening and contraction) while Δ and Ω are *linear* (admitting neither weakening nor contraction).

Figure 2 contains the full collection of rules of LLKF. The logical rules of LLKF are applied in one of two *phases*. The positive phase consists of rules applied to positive sequents. Each such rule has the focused formula in the positive sequent as the principal formula, and if the operands of the principal connective are also positive then the focus is maintained on them in their corresponding premises. Likewise, the negative phase consists of rules applied to the active context of negative sequents. Mediating the two phases are the *decision rules* $\left[\mathsf{lf}\right]$ and $\left[\mathsf{uf}\right]$ where particular formulas are granted focus. In the case that the formula is selected from the unrestricted context Γ, it continues to be present in Γ in the premises, *i.e.*, the rule has a built in contraction. This is in fact the only form of contraction in the calculus. Instead of a structural

Positive Phase

$$\frac{}{\vdash \Gamma; \overline{a}; \left[a\right]}\left[\mathsf{fi}\right] \quad \frac{\vdash \Gamma; \Delta_1; \left[P\right] \quad \vdash \Gamma; \Delta_2; \left[Q\right]}{\vdash \Gamma; \Delta_1, \Delta_2; \left[P \otimes Q\right]}\left[\otimes\right] \quad \frac{}{\vdash \Gamma; \cdot; \left[\mathbf{1}\right]}\left[\mathbf{1}\right] \quad \frac{\vdash \Gamma; \Delta; \left[P_i\right]}{\vdash \Gamma; \Delta; \left[P_1 \oplus P_2\right]}\left[\oplus\right]$$

$$\frac{\vdash \Gamma; \cdot; N}{\vdash \Gamma; \cdot; \left[!N\right]}\left[!\right] \quad \frac{\vdash \Gamma; \Delta; N}{\vdash \Gamma; \Delta; \left[\downarrow N\right]}\left[\downarrow\right]$$

Negative Phase

$$\frac{\vdash \Gamma; \Delta, \overline{a}; \Omega}{\vdash \Gamma; \Delta; \Omega, \overline{a}}\left[\mathsf{nr}\right] \quad \frac{\vdash \Gamma; \Delta; \Omega, N, M}{\vdash \Gamma; \Delta; \Omega, N \parr M}\left[\parr\right] \quad \frac{\vdash \Gamma; \Delta; \Omega}{\vdash \Gamma; \Delta; \Omega, \bot}\left[\bot\right] \quad \frac{\vdash \Gamma; \Delta; \Omega, N \quad \vdash \Gamma; \Delta; \Omega, M}{\vdash \Gamma; \Delta; \Omega, N \,\&\, M}\left[\&\right]$$

$$\frac{}{\vdash \Gamma; \Delta; \Omega, \top}\left[\top\right] \quad \frac{\vdash \Gamma, P; \Delta; \Omega}{\vdash \Gamma; \Delta; \Omega, ?P}\left[?\right] \quad \frac{\vdash \Gamma; \Delta, P; \Omega}{\vdash \Gamma; \Delta; \Omega, \uparrow P}\left[\uparrow\right]$$

Decision

$$\frac{\vdash \Gamma; \Delta; \left[P\right]}{\vdash \Gamma; \Delta, P; \cdot}\left[\mathsf{lf}\right] \quad \frac{\vdash \Gamma, P; \Delta; \left[P\right]}{\vdash \Gamma, P; \Delta; \cdot}\left[\mathsf{uf}\right]$$

Figure 2: Rules of LLKF. In the $\left[\oplus\right]$ rule, $i \in \{1, 2\}$.

rule of weakening, rules with no premises are altered to admit any number of unrestricted side formulas.

In order to compare LLKF to LLK, we must first translate between the two syntaxes – unpolarized and polarized.

Definition 1 (Depolarization)**.** *Given a polarized formula A, let $\lfloor A \rfloor$ stand for that unpolarized formula with all occurrences of $\downarrow$ and $\uparrow$ removed from A. Extend this definition naturally to multi-sets of polarized formulas.*

Theorem 2 (Soundness and completeness of LLKF w.r.t. LLK)**.**

- *If* $\vdash \Gamma; \Delta; \left[P\right]$ *in* LLKF*, then* $\vdash ?\lfloor\Gamma\rfloor, \lfloor\Delta\rfloor, \lfloor P\rfloor$ *in* LLK*.*
- *If* $\vdash \Gamma; \Delta; \Omega$ *in* LLKF*, then* $\vdash ?\lfloor\Gamma\rfloor, \lfloor\Delta\rfloor, \lfloor\Omega\rfloor$ *in* LLK*.*
- *If* $\vdash \lfloor\Omega\rfloor$ *in* LLK*, then* $\vdash \cdot; \cdot; \Omega$ *in* LLKF*.*

Proof. There are many ways to prove this theorem. We refer the interested reader to one of the standard approaches [13, 17, 5]. □ □

3 Labelling Subformulas and Taming Contraction

As already mentioned in the introduction, there is a single rule each in LLK and in LLKF that causes the set of derivations (including open derivations) of a given sequent to be infinite: contraction (ct) in the former, and unrestricted focus ($\left[\mathsf{uf}\right]$) in the latter. The $\left[\mathsf{lf}\right]$ rule moves a positive formula into focus after which its principal connective is consumed, and the $\left[\mathsf{nr}\right]$ rule moves a negative atom into the linear context from which it can never be selected as a principal formula again. The remaining rules all consume a connective. Therefore, in order to obtain a decidable sub-logic of linear logic for which these proof systems are manifestly decision procedures, it is these rules of contraction that need to be controlled.

To find such a means of controlling contraction, we can look for inspiration at calculi with more permissive permutations in their inference rules. Generally speaking, such calculi tend to be calculi of *deep inference*, wherein there is no strong distinction between sequent and formula, and inference rules can be applied in any subformula context. The system LSF for classical linear logic in the focused calculus of structures [7] is perhaps the most closely related system to LLKF in this paper. In LSF (like in nearly every proof system in the calculus of structures), the contraction rules permute with all rules (including other contractions). Any LSF proof can therefore be divided into two phases: a bottom part consisting of the contractions, and a top part that is contraction-free. If the system is designed carefully, as LSF is, then this contraction-free sub-calculus admits only finitely many (possibly open) derivations of a given formula, and is therefore decidable.

While it is possible to adopt LSF instead of LLKF to represent proofs, this will require a complicated and not particularly enlightening detour. Instead, we will just transplant the *effect* of permuting and separating the contractions to LLKF proofs. The mechanism we will use is *labelling* the contractions: both the formulas and the inference rules will be modified to admit labels.

Definition 3. *A* label *(written* $\alpha, \beta, \dots$*) is a non-empty word formed over an infinite set of* atomic labels *(written* $\mathsf{a}, \mathsf{b}, \dots$*). Two distinct atomic labels* $\mathsf{0}$ *and* $\mathsf{1}$ *are always assumed to be present. We use* $\mathcal{L}$ *to denote the set of all labels,* Λ *to denote label multi-sets, and* $\alpha\beta$ *to denote the label formed by concatenating the labels* α *and* β*.*

The labels will be used to index particular subformulas in a derivation in a labelled version of LLKF, which we call L3KF. Intuitively, a label denotes a *path* through the subformula relation, with the formula labelled by $\alpha\beta$ being a strict subformula of that labelled by α. Not every subformula needs to be labelled – those subformulas that do not involve any polarity changes or boundary conditions can remain unlabelled. Instead, we affix labels only to the shifts, the exponentials, and the atomic formulas.

Definition 4. Action formulas *(written L) and* reaction formulas *(written R) are given by the following grammar.*

$$R ::= a^{\mathsf{a}} \mid !^{\mathsf{a}} N \mid \downarrow^{\mathsf{a}} N \qquad\qquad L ::= \overline{a}^{\mathsf{a}} \mid ?^{\mathsf{a}} P \mid \uparrow^{\mathsf{a}} P$$

Labelled polarized formulas *have the following grammar.*

$$P, Q, \dots ::= R \mid P \otimes Q \mid \mathbf{1} \mid P \oplus Q \mid \mathbf{0}$$
$$N, M, \dots ::= L \mid N \parr M \mid \bot \mid N \,\&\, M \mid \top$$

Strictly speaking, for tracking contractions we do not need to label any but the ?-formulas. We choose to label all (re)action subformulas anticipating their use in the next section. For the rest of this paper, unless indicated, we will work solely with labelled polarized formulas. The contexts of L3KF are modifications of those of LLKF to support labelled formulas.

$$\Gamma ::= \cdot \mid \Gamma, \langle \alpha{:}P \rangle \qquad \Delta ::= \cdot \mid \Delta, \langle \alpha{:}P \rangle \mid \Delta, \langle \alpha{:}\overline{a}^{\mathsf{a}} \rangle \qquad \Omega ::= \cdot \mid \Omega, \langle \alpha{:}N \rangle$$

In addition to these labelled contexts, the focused formula in positive sequents will also be labelled, written as $[\alpha{:}P]$. From any L3KF sequent we can index particular subformulas using the labels.

Definition 5 (indexing)**.** *Given an* L3KF *sequent* σ *and a label* α*, we write* $\sigma(\alpha)$ *for an arbitrary (re)action subformula of* σ *such that:*

Positive Phase

$$\frac{}{\vdash \Gamma; \langle \alpha{:}\overline{a}^{\mathsf{a}} \rangle; [\beta{:}a^{\mathsf{b}}]}\,[\mathsf{fi}] \qquad \frac{\vdash \Gamma_1; \Delta_1; [\alpha{:}P] \quad \vdash \Gamma_2; \Delta_2; [\alpha{:}Q]}{\vdash \Gamma_1, \Gamma_2; \Delta_1, \Delta_2; [\alpha{:}P \otimes Q]}\,[\otimes] \qquad \frac{}{\vdash \Gamma; \cdot; [\alpha{:}\mathbf{1}]}\,[\mathbf{1}]$$

$$\frac{\vdash \Gamma; \Delta; [\alpha{:}P_i]}{\vdash \Gamma; \Delta; [\alpha{:}P_1 \oplus P_2]}\,[\oplus] \qquad \frac{\vdash \Gamma; \cdot; \langle \alpha\mathsf{a}{:}N \rangle}{\vdash \Gamma; \cdot; [\alpha{:}!^{\mathsf{a}}N]}\,[!] \qquad \frac{\vdash \Gamma; \Delta; \langle \alpha\mathsf{a}{:}N \rangle}{\vdash \Gamma; \Delta; [\alpha{:}{\downarrow}^{\mathsf{a}}N]}\,[\downarrow]$$

Negative Phase

$$\frac{\vdash \Gamma; \Delta, \langle \alpha{:}\overline{a}^{\mathsf{a}} \rangle; \Omega}{\vdash \Gamma; \Delta; \Omega, \langle \alpha{:}\overline{a}^{\mathsf{a}} \rangle}\,[\mathsf{nr}] \qquad \frac{\vdash \Gamma; \Delta; \Omega, \langle \alpha{:}N \rangle, \langle \alpha{:}M \rangle}{\vdash \Gamma; \Delta; \Omega, \langle \alpha{:}N \parr M \rangle}\,[\parr] \qquad \frac{\vdash \Gamma; \Delta; \Omega}{\vdash \Gamma; \Delta; \Omega, \langle \alpha{:}\bot \rangle}\,[\bot]$$

$$\frac{\vdash \Gamma; \Delta; \Omega, \langle \alpha{:}N \rangle \quad \vdash \Gamma; \Delta; \Omega, \langle \alpha{:}M \rangle}{\vdash \Gamma; \Delta; \Omega, \langle \alpha{:}N \,\&\, M \rangle}\,[\&] \qquad \frac{}{\vdash \Gamma; \Delta; \Omega, \langle \alpha{:}\top \rangle}\,[\top]$$

$$\frac{\vdash \Gamma, \langle \alpha\mathsf{a}{:}P \rangle; \Delta; \Omega}{\vdash \Gamma; \Delta; \Omega, \langle \alpha{:}?^{\mathsf{a}}P \rangle}\,[?] \qquad \frac{\vdash \Gamma; \Delta, \langle \alpha\mathsf{a}{:}P \rangle; \Omega}{\vdash \Gamma; \Delta; \Omega, \langle \alpha{:}{\uparrow}^{\mathsf{a}}P \rangle}\,[\uparrow]$$

Decision

$$\frac{\vdash \Gamma; \Delta; [\alpha{:}P]}{\vdash \Gamma; \Delta, \langle \alpha{:}P \rangle; \cdot}\,[\mathsf{lf}] \qquad \frac{\vdash \Gamma, \langle \alpha\mathsf{0}{:}P \rangle; \Delta; [\alpha\mathsf{1}{:}P]}{\vdash \Gamma, \langle \alpha{:}P \rangle; \Delta; \cdot}\,[\mathsf{uf}]$$

Figure 3: Rules of L3KF. In the $[\oplus]$ rule, $i \in \{1, 2\}$.

- α *is of the form* $\beta\gamma$ *(with* γ *possibly empty);*
- β *labels some contextual element of* σ*, i.e., the contexts of* σ *contain an element of the form* $\langle \beta{:}A \rangle$*; and*
- γ *indicates that subformula of* A *reached by the trail of atomic labels in* γ*.*

For example, if the sequent σ contains the labelled element $\langle \alpha{:}a^{\mathsf{b}} \oplus {\downarrow}^{\mathsf{c}}(N \parr \overline{c}^{\mathsf{d}}) \rangle$ in a context, then $\sigma(\alpha\mathsf{b}) = a^{\mathsf{b}}$, $\sigma(\alpha\mathsf{c}) = N \parr c^{\mathsf{d}}$, and $\sigma(\alpha\mathsf{cd}) = \overline{c}^{\mathsf{d}}$. Indexing can be non-deterministic if there are duplicates of labels in an L3KF sequent. We will generally only work with sequents where there are no duplicates, which we call standard sequents.

Definition 6. *We say that an* L3KF *sequent* σ *is* standard *if for every label* α*, there is at most one labelled formula* A *such that* $\sigma(\alpha) = A$*.*

We assume that all L3KF sequents in the the rest of the paper are standard. We shall design the rules of L3KF in such a way that if the end-sequent is standard, then in every (possibly open) L3KF derivation of that sequent all intermediate sequents are also standard.

The full set of rules of L3KF is shown in Fig. 3. For the rules involving shifts and exponentials, the atomic label on the principal formula is appended to the relevant label in the sequent. The other rules preserve the labels from conclusion to premises. The binary rules $[\otimes]$ and $[\&]$ are the only rules that cause a complete duplication of the labels in the side formulas; however, if the conclusion is standard, then each premise of these two rules is individually also standard. For the $[\parr]$ rule, although the label for the principal formula is duplicated, if the conclusion is standard then each operand of the $\parr$ will have a disjoint set of atomic labels in its respective subformulas and hence the premise is also standard.

The only rule that differs from the pattern is the $[\mathsf{uf}]$ rule that appends a new atomic label $\mathsf{0}$ or $\mathsf{1}$ to the end of the contextual label of the principal formula. Repeated applications of this rule on the same principal formula will keep one version with a sequence of $\mathsf{0}$s appended in the unrestricted context, while the remaining copies will be focused on and decomposed. As we intend to control this rule, we will impose a global restriction on the number of $\mathsf{0}$s that can be appended to a given label: we call this a *contraction bound*.

Definition 7. *A* contraction bound $\mathcal{B}$ *is a total function from* $\mathcal{L}$ *to* $\mathbf{N}$ *that maps all but a finite set of labels to* 0*s with the additional property that for every* $\alpha \in \mathcal{L}$*, if* $\mathcal{B}(\mathsf{\alpha}) = n > 0$ *then* $\mathcal{B}(\mathsf{\alpha 0}) = n - 1$*.*

Definition 8. *Given a contraction bound* $\mathcal{B}$*, the system* L3KF($\mathcal{B}$) *is a proof system consisting of the inference rules of* L3KF *(Fig. 3) such that all the instances of the* $[\mathsf{uf}]$ *rule with principal formula* $\langle \mathsf{\alpha}{:}P \rangle$ *have the property that* $\mathcal{B}(\alpha) > 0$*.*

Remark 9. *Any* L3KF *sequent has only finitely many (possibly open)* L3KF($\mathcal{B}$) *derivations, as the contraction bounds get stricter with more* $\mathsf{0}$*s.* □

Obviously, therefore, L3KF($\mathcal{B}$) for an arbitrary $\mathcal{B}$ is not complete with respect to LLKF (nor to LLK) because propositional linear logic is undecidable [15]. Yet, for any (possibly open) LLKF derivation we can indeed construct a $\mathcal{B}$ such that the corresponding labelled form of the LLKF end-sequent is provable in L3KF($\mathcal{B}$). To make this formal, we compare the LLKF and L3KF systems modulo labelling.

Definition 10. *Given an* L3KF *sequent* σ*, we write* $\mathrm{unl}\,(\sigma)$ *for that* LLKF *sequent that: replaces all instances of* $\langle \mathsf{\alpha}{:}A \rangle$ *in* σ *by* A*, then erases all labels from the (re)action subformulas of* σ*.*

Theorem 11 (soundness and completeness)**.** *1. For any* σ *derivable in* L3KF($\mathcal{B}$)*, the sequent* $\mathrm{unl}\,(\sigma)$ *is derivable in* LLKF*.*

2. For any standard σ *for which* $\mathrm{unl}\,(\sigma)$ *is derivable in* LLKF*, there is a contraction bound* $\mathcal{B}$ *such that* σ *is derivable in* L3KF($\mathcal{B}$)*.*

Sketch. Each case follows by a simple induction.

1. If $\mathrm{unl}\,(-)$ is applied to every sequent in every inference rule of L3KF($\mathcal{B}$), then the result is an inference rule of LLKF.

2. Begin with an empty bound and read the LLKF derivation from conclusion upwards. Whenever $[\mathsf{uf}]$ is applied in the LLKF derivation, for the corresponding label $\mathsf{\alpha}$ in the L3KF derivation we set $\mathcal{B}(\mathsf{\alpha}) = 1$, then increment the bounds for every prefix of $\mathsf{\alpha}$ formed by removing $\mathsf{0}$s from the end. We then perform this instance of $[\mathsf{uf}]$ in the L3KF($\mathcal{B}$) derivation. For all the other rules, the contraction bound remains untouched and the L3KF rule is the obvious one from Fig. 3. □

□

The above completeness theorem is much more general than needed. As we intend the contraction bounds to be used in proof certificates for LLKF derivations, we may give ourselves the freedom to choose a labelling for the end-sequent. In fact, we may choose a most parsimonious simple labelling.

Definition 12. *An* L3KF *sequent is said to be* simply labelled *if there is at most a single occurrence of every atomic label in the sequent. In other words, no two contextual elements share a non-empty label prefix, and every (re)action subformula has a unique atomic label.*

It is easy to see that a simply labelled sequent is standard. In an implementation, if the LLKF sequents have some canonical universal representation, then this simply labelled form is predictable and so the labelled form of the LLKF end-sequent need not be recorded in the proof certificate. Nevertheless, to be general, we will mention the simply labelled forms in the certificates.

Definition 13. *A* contraction certificate *for an* LLKF *sequent σ is a pair $(\tau, \mathcal{B})$ where: (1) τ is a simply labelled* L3KF *sequent with* $\mathrm{unl}(\tau) = \sigma$*; and (2) $\mathcal{B}$ is a contraction bound.*

Contraction certificates obviously exist even for unprovable LLKF sequents. Completeness (Thm. 11.2) guarantees that any provable LLKF sequent will have a corresponding contraction bound $\mathcal{B}$ for which the simply labelled form is provable in L3KF($\mathcal{B}$). To consume—check—a contraction certificate is equivalent to constructing this L3KF($\mathcal{B}$) proof knowing just the end-sequent and the contraction bound. Now, for any contraction bound $\mathcal{B}$, the system L3KF($\mathcal{B}$) is manifestly a decision procedure. After all, L3KF has only finitely many open derivations (Remark 9). Thus, in order to consume a contraction certificate $(\tau, \mathcal{B})$, it is sufficient to enumerate all L3KF($\mathcal{B}$) derivations of τ, succeeding if any one of them is an L3KF($\mathcal{B}$) proof. The LLKF proof of $\mathrm{unl}(\tau)$ can be reconstructed from this L3KF($\mathcal{B}$) proof by applying the procedure outlined in the proof of Thm. 11.1.

To represent the contraction certificate, we require no more space than the product of the number of labels in the end-sequent and the number of uses of the contraction rule. This will always be smaller than the full proof (which needs to record the contractions anyhow) because it omits all the logical content of the proof. However, it is an easy exercise to construct a series of problems where the number of required uses of contraction grows exponentially, so in the worst case the contraction certificate will not necessarily improve over the full proof by more than a polynomial factor. In practical uses of proof certificates, however, the contractions will only be expected to be used for "facts" from the ambient unrestricted context (in other words, the axioms in the theory and the lemmas), which is not so pathological. Indeed, in the very expressive multi-set rewriting fragment of linear logic, the only uses of contraction will be for the (encoding of the) rewrite rules, and there will be exactly as many contractions as steps in the trace. The contraction bounds in the corresponding contraction certificates will be considerably smaller than the full proofs; indeed, the space requirement for the bound will be linear in the length of the trace.

4 Determinizing Hints

We can potentially declare success at this point, but it is worth noting that consuming a contraction certificate by enumerating all proofs up to a bound may not be very practical. If the LLKF proof is of a purely MALL formula, then there are no occurrences of $\left[\mathsf{uf}\right]$ at all, and hence the contraction bound will be empty. Since the proof certificate records none of the logical rules, the reconstruction of the L3KF($\mathcal{B}$) proof is then at least as computationally expensive as searching for the MALL proof, which is a PSPACE-complete problem [15]. In this section, we will add more information to the proof certificates to make reconstruction more deterministic in exchange for an increase in the size of the certificates. This additional detail in the certificate will be a tunable parameter: with enough detail, the reconstruction should be completely deterministic, but a certificate without any detail should still remain consumable.

To motivate the additions, let us first consider the kinds of information that are recorded in a fully detailed proof. For linear logic, we have the following general non-deterministic choices when searching for an LLKF (or an L3KF) proof.

- Choices between multiple rules for the same principal formula, caused by the $[\oplus]$ rule, also known as *disjunctive non-determinism*.
- Choices involving splitting the linear context in the $[\otimes]$ rule, also known as *multiplicative non-determinism*.
- Choices of foci in the $[\mathsf{lf}]$ and $[\mathsf{uf}]$ rules, or the *decision non-determinism*.

(In the first-order case, constructing the existential witnesses is also non-deterministic, which is very similar to the disjunctive case.) None of the other choices matter for focused search. In particular, the order of application of the rules in the negative phase is immaterial. Every unfinished premise of a negative LLKF (or L3KF($\mathcal{B}$)) sequent will be *neutral* (*i.e.*, of the form $\vdash \Gamma; \Delta; \cdot$). Regardless of the order of application of the negative rules there will always be the same multi-set of neutral premises of a negative sequent.

In order to determinize proof reconstruction, the certificate will have to record these non-deterministic choices. For disjunctive choices, one of the operands of a $\oplus$ formula disappears from the sequent. Recall that in a standard L3KF sequent, every (re)action subformula is indexed by a unique label. Therefore, for the operand of the $\oplus$ rule that disappears, so will all the indexes associated to the subformulas of that lost operand. Since the positive phase must (eventually) finish[2] by one of $[\mathsf{fi}]$, $[!]$ or $[\downarrow]$, it follows that exactly one of the topmost atomic labels in the focused $\oplus$ formula will eventually be mentioned in the derivation above, so the choice made in the $\oplus$ rule can be deduced from the indexes in the sequents higher in the proof.

For the multiplicative choices, we can use the input-output interpretation of the linear context [10, 11]. Briefly, the entire linear context is sent to the left premise of an instance of $[\otimes]$; this premise consumes as much of the context as it needs and sends the rest to the right premise, which in turn sends its unconsumed portion "down" the proof. The proof of the end-sequent is accepted if it is able to consume the entire linear context. The input-output interpretation thus determinizes the multiplicative non-determinism, *i.e.*, backtracking over different ways of splitting the context is unnecessary. It is nevertheless not complete: it forces a sequence between different multiplicative—and semantically concurrent—branches of the proof, and so an adversarial problem can be constructed for which committing to, say, the left premise before the right will lead to infinitely deep proofs. This incompleteness is not an issue for us, however, as the contraction bound makes all derivations finite. No matter which multiplicative branch is scheduled first, the search procedure on that branch must terminate within the bound. Because this technique is well known and standard, we omit a more detailed and formal description in this paper.

This leaves only the focusing decisions. An obvious way to record these is to just extract the tree of decision rules—which we will call the *decision tree* (to be formalized presently)—from the L3KF($\mathcal{B}$) proof. Because every sequent in the derivation is standard, the contextual label of the principal formula in the decision rule is unambiguous. Hence, the decision tree can be straightforwardly built using these labels for the internal nodes. Still, this representation is not wholly satisfactory: the decision tree is, in the worst case, a constant fraction of the size of the entire L3KF($\mathcal{B}$) proof.[3] To save space, proof certificates must be allowed to omit portions of the full tree.

[2] Reading, as usual, in the direction of conclusion to premises.

[3] In practice, of course, this fraction will tend to be small because focusing already eliminates much of the noise in LLK proofs.

Now, the complete decision tree already contains a record of all the decision rules required to build a proof, and hence an additional bound such as one on contractions is redundant. But, if we omit portions of the tree, it does become important to record the contraction bounds so that reconstruction remains decidable. A single contraction bound for the entire proof can certainly be recorded in the certificate anyway, but we can avoid the redundancy with the (recorded portion of the) decision tree by a simple trick. For every unrecorded suffix of the decision tree, we compute locally the contraction bound of the corresponding sub-proof and make it the leaf of the tree. In other words, the certificate would contain a prefix of the decision tree, with contraction bounds at the leaves that correspond to omitted sub-proofs.

It should be intuitively obvious that reconstruction using this representation is decidable, as every sub-proof is built either deterministically from the record of focusing steps in the tree or by bounded search using the contraction bounds. It is also clear that proof reconstruction will get increasingly deterministic as more of the decision tree is recorded. The level of detail in the certificate thus becomes a tunable parameter that can be tailored to particular needs or even negotiated between the producer and the consumer.

Let us now crystallize these intuitions with formal definitions.

Definition 14. *A* decision tree $\mathcal{D}$ *is a tree where each node: (1) contains a pair* $\langle \alpha, \Lambda \rangle$ *where* α *is a label and* Λ *is a set of atomic labels, and has a finite number (possibly zero) of children; or (2) contains a contraction bound (Defn. 7) and no children. A decision tree is* full *if it contains no nodes with contraction bounds.*

The pairs $\langle \alpha, \Lambda \rangle$ are interpreted as follows: α is the contextual label of the principal formula of a corresponding decision rule ($\left[\mathsf{lf}\right]$ or $\left[\mathsf{uf}\right]$), and Λ represents the labels of all the reaction formulas—*i.e.*, the labels of the principal formulas of the $\left[\mathsf{fi}\right]$, $\left[!\right]$ and $\left[\downarrow\right]$ rules—at the boundaries of the positive phase that immediately follows (reading from conclusion upwards) the decision rule. Any disjunctive choices made in the positive phase is fully determined by this second component, as it will only contain the labels corresponding to disjuncts that are selected in the $\left[\oplus\right]$ rules. Technically, this set of labels merely needs to be large enough to disambiguate all the disjunctive choices made in the positive phase.

Definition 15. *A* certificate *for an* LLKF *sequent* σ *is of the form* $\langle \tau, \mathcal{D} \rangle$ *where* τ *is a neutral simply labelled* L3KF *sequent with* $\mathrm{unl}(\tau) = \sigma$ *and* $\mathcal{D}$ *is a decision tree. A* full certificate *is a certificate with a full decision tree.*

To consume (*i.e.*, check) a certificate, we execute the following algorithm.

Definition 16 (checking proof certificates)**.** *The following algorithm decides if a given proof certificate* $\langle \tau, \mathcal{D} \rangle$ *is valid or not. We proceed by induction on the structure of* $\mathcal{D}$*.*

1. *If the root node of* $\mathcal{D}$ *contains a contraction bound* $\mathcal{B}$*, then we enumerate all* L3KF($\mathcal{B}$) *derivation of* τ*, succeeding if any of them is a proof and failing otherwise.*

2. *If the root node of* $\mathcal{D}$ *contains* $\langle \alpha, \Lambda \rangle$ *and has children* $\mathcal{D}_1, \ldots, \mathcal{D}_n$*, then we find* $\langle \alpha{:}P \rangle$ *in* τ *(failing if it doesn't exist) and perform the corresponding decision rule (*$\left[\mathsf{lf}\right]$ *or* $\left[\mathsf{uf}\right]$*). In the subsequent positive phase, for all disjunctive choices we select that disjunct whose immediate reactive subformulas have atomic labels found amongst* Λ *(failing if both disjuncts meet this criterion or if neither does). If this phase results in* n *neutral sequents* $\tau_1, \ldots, \tau_n$*, we then check each certificate* $\langle \tau_i, \mathcal{D}_i \rangle$ *for* $i \in 1..n$*.*

In order for this checking procedure to avoid unnecessary work, the sub-trees in case 2 must line up precisely with the sub-derivations. This requires that the premises of an inference rule be

produced in a predictable order, which in turn requires determinizing the order of application of the rules in the negative phase. This is easily done by treating the active context Ω as a list instead of as a multi-set, with the principal formula then always at the head of the list. This is precisely Andreoli's original proposal for the negative (or asynchronous) phase in focusing [1]. Note that this is purely a matter of performance. The checking algorithm can backtrack over all the ways to match up sub-trees to neutral premises (there are only a finite number of permutations). The available indexes in the neutral premises and in the sub-trees can also give hints as to the right match-up.

Theorem 17 (soundness of checking)**.** *If the certificate* $\langle \tau, \mathcal{D} \rangle$ *is accepted by the algorithm of Defn. 16, then* $\mathrm{unl}(\tau)$ *is a provable* LLKF *sequent.*

Proof. Immediate from Thm. 11 (1). □ □

Theorem 18 (completeness of certification)**.** *If* σ *is provable in* LLKF*, then there is a valid proof certificate* $\langle \tau, \mathcal{D} \rangle$ *for which* $\mathrm{unl}(\tau) = \sigma$*.*

Sketch. By Thm. 11 (2), there is an L3KF derivation of a τ for some τ with $\mathrm{unl}(\tau) = \sigma$. The full decision tree from this L3KF derivation gives a suitable certificate. □ □

Observe that checking a full certificate involves no non-deterministic choices at all. There is, in fact, an order of determinacy among proof certificates, stated below as a theorem. Its proof is omitted here because it requires a fairly unilluminating sequence of technical lemmas.

Proposition 19 (determinacy)**.** *Given two certificates* $\xi_1 = \langle \tau, \mathcal{D}_1 \rangle$ *and* $\xi_2 = \langle \tau, \mathcal{D}_2 \rangle$*, say that* ξ_1 *is* more deterministic than ξ_2 *if* $\mathcal{D}_2$ *with all contraction bound nodes removed is a prefix of* $\mathcal{D}_1$ *likewise. Then, checking* ξ_1 *involves fewer non-deterministic choices than checking* ξ_2*.* □

5 Concluding Remarks

We have given a way of systematically building proof certificates with a variable level of detail from focused sequent proofs. The main technical device is labelling of particular subformulas, which is both used to extract pre-computed information about contractions and to obtain a skeletal form of the proof as a *decision tree*. We have intentionally limited ourselves to bounded contraction as the mechanism for eliding detail from the proof certificate. There are obviously other—even simpler—means of eliding detail: for instance, instead of bounding contractions, we can just bound the overall depth of the sub-proof. These alternative mechanisms are readily compatible with the proposed design.

For intuitionistic or classical logic, the situation is generally simpler because of the absence of resource non-determinism. (Their propositional fragments are generally decidable anyway.) For the first-order case, the proof certificates would additionally have to depend on first-order unification in the consumer; alternatively, the decision tree nodes would have to record the existential witnesses.

As rightly pointed out by the anonymous referees, the claim in this paper of building compact proof certificates will ultimately have to be validated empirically. To this end, we are in the process of adapting the LI family of automated proof-producing linear logic provers [4] to function as "proof-elaborators" that will convert a certificate into a full proof, in essence implementing the algorithm of Defn. 16.

Acknowledgement: we thank Nicolas Guenot and Lutz Straßburger for many useful discussions on the nature of contraction, and the anonymous referees for their insightful comments.

References

[1] J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *J. of Logic and Computation*, 2(3):297–347, 1992.

[2] M. Boespflug. *Conception d'un noyau de vérification de preuves pour le λΠ-calcul modulo*. PhD thesis, Ecole Polytechnique, 2011.

[3] T. Brock-Nannestad and C. Schürmann. Focused natural deduction. In C. Fermüller and A. Voronkov, editors, *LPAR 17*, volume 6397 of *LNCS*, pages 157–171, Yogyakarta, Indonesia, 2010. Springer.

[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. Focusing strategies in the sequent calculus of synthetic connectives. In I. Cervesato, H. Veith, and A. Voronkov, editors, *LPAR: International Conference on Logic, Programming, Artificial Intelligence and Reasoning*, volume 5330 of *LNCS*, pages 467–481. Springer, Nov. 2008.

[6] K. Chaudhuri. Magically constraining the inverse method using dynamic polarity assignment. In C. Fermüller and A. Voronkov, editors, *Proc. 17th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 6397 of *LNCS*, pages 202–216, Yogyakarta, Indonesia, Oct. 2010. Springer.

[7] K. Chaudhuri, N. Guenot, and L. Straßburger. The Focused Calculus of Structures. In *Computer Science Logic: 20th Annual Conference of the EACSL*, Leibniz International Proceedings in Informatics (LIPIcs), pages 159–173. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Sept. 2011.

[8] 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.

[9] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *J. of Symbolic Logic*, 57(3):795–807, Sept. 1992.

[10] J. Hodas and D. Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994.

[11] J. Hodas, K. Watkins, N. Tamura, and K.-S. Kang. Efficient implementation of a linear logic programming language. In J. Jaffar, editor, *Proceedings of the 1998 Joint International Conference and Symposium on Logic Programming*, pages 145–159, 1998.

[12] O. Laurent. *Etude de la polarisation en logique*. PhD thesis, Université Aix-Marseille II, Mar. 2002.

[13] O. Laurent. A proof of the focalization property of linear logic. Unpublished note, May 2004.

[14] C. Liang and D. Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, 410(46):4747–4768, 2009.

[15] P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. *Annals Pure Applied Logic*, 56:239–311, 1992.

[16] D. Miller. A proposal for broad spectrum proof certificates. In J.-P. Jouannaud and Z. Shao, editors, *CPP: First International Conference on Certified Programs and Proofs*, volume 7086 of *LNCS*, pages 54–69, 2011.

[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 *LNCS*, pages 405–419. Springer, 2007.

[18] G. C. Necula. Proof-carrying code. In *Conference Record of the 24th Symposium on Principles of Programming Languages 97*, pages 106–119, Paris, France, 1997. ACM Press.

[19] L. Straßburger. *Linear Logic and Noncommutativity in the Calculus of Structures*. PhD thesis, Technische Universität Dresden, 2003.