From focussed proof systems to complexity bounds

Anupam Das

École Normale Supérieure de Lyon, France
anupam.das@ens-lyon.fr

Abstract

We conduct a complexity-theoretic study of focussed proof systems, relating the alternation of synchronous and asynchronous phases in a proof to an appropriate alternating time hierarchy, for instance the polynomial hierarchy. We propose a notion of 'over-focussing' that admits non-branching invertible rules during synchronous phases, due to the fact that deterministic computations can equally be carried out by a nondeterministic machine. As an application, we develop an over-focussed system for a fragment of intuitionistic propositional logic which we show to be already **PSPACE**-complete by a refinement of Statman's translation from true QBFs. We show that this translation has a well-behaved inverse, preserving quantifier complexity, in the form of a QBF encoding of proof search for the over-focussed system, demonstrating the usefulness of considering such systems. Consequently we are able to delineate intuitionistic tautologies according to the polynomial hierarchy and derive further proof-theoretic consequences for intuitionistic logic.

Contents

A note on the novel aspects of this work Some parts of this article do not present genuinely new material; in particular the contents of Section 2 are well-known to many proof theorists, while Section 3 essentially formalises folklore. The purpose of these sections is to set the context for the remainder of the article, in particular motivating the results of later sections.

1 Introduction

Focussed proof systems [1, 17] organise the process of bottom-up proof search into 'synchronous' and 'asynchronous' phases, distinguishing the steps where genuine choices are made from those where no information is lost. This treatment of proof search has close connections to logic programming [20] and the Curry-Howard correspondence [11]. The purpose of this article is to examine the complexity-theoretic aspects of focussed systems.

We may naturally view synchronous phases as nondeterministic computation and asynchronous phases as co-nondeterministic computation, in the proof-search-as-computation paradigm. Consequently we may use focussed systems to obtain complexity bounds for logics in terms of *alternating time*. The main advantage of this approach is the 'focussing theorem', which allows systems to significantly reduce the number of alternations between synchronous and asynchronous steps in the proof search space. As an analogy, this extends the idea that, for quantified Boolean formulae (QBF), the quantifier hierarchy tightly delineates the levels of the polynomial hierarchy; from a proof-theoretic viewpoint this is exemplified by the alternation of asynchronous and synchronous phases in Boolean Truth Trees [12].

It is not difficult to formalise the bounds obtained from a focussed system. Let us, for example, consider a logic L that is **PSPACE**-complete. We can encode a family 'provability predicates' as, say, QBFs parametrised by the *decide-depth* of the proof search space. Decide-depth can be calculated in polynomial time, as is implicitly shown in e.g. [22], and so give rises to an encoding of L into true QBFs (QCPL), with formulae of the same decide-depth mapped to ones of the same quantifier complexity. We formalise this encoding in Section 3.

However, the bounds obtained by doing this directly can be far from optimal. While there can always be improvements, there is one particular aspect of focussed systems which does not adequately preserve the analogy with nondeterministic and co-nondeterministic computation. In a focussed system invertible rules must be applied during an asynchronous phase, regardless of whether they are branching or non-branching. However, a non-branching invertible rule corresponds to deterministic computation, and so does not contribute to overall complexity if we are in a synchronous phase. For the sake of example, consider the class of formulae of some system provable by only non-branching rules, regardless of whether they are invertible or not. If the search space has depth d and lines of size w, then clearly this class is in $\mathbf{NTIME}(wd)$. However the decide-depth of focussed proofs would grow linearly in the number of alternations between invertible and non-invertible rules, and so the analogy with alternating complexity breaks down.

In fact we can go further than this and extend the focussing methodology itself: if we are in a synchronous phase in bottom-up proof search and reach a connective whose corresponding rule is invertible and non-branching, then we may apply the rule but keep a focus on one of its auxiliary formulae. This can be proved using usual rule permutation arguments. We give an example of such a system, which we call an 'over-focussed' system, for a fragment of intuitionistic propositional logic, in Section 4.

The general observation that bipoles in focussed systems sometimes commute also underlies the development of 'multi-focussed' proof systems [6]. However those systems allow *more* proofs, since each uni-focussed proof is also a multi-focussed proof. In particular this means that we cannot obtain better bounds for proof search from such systems, since their search space is strictly larger. On the other hand, an over-focussed system is more restrictive and admits fewer proofs, and so is better suited for complexity-theoretic analysis.

A motivational application In Section 4 we apply the aforementioned ideas to a concrete case study. It is well known that QCPL can be encoded into intuitionistic propositional logic (IPL), due to Statman [26], and so IPL is **PSPACE**-hard. One way to show the converse, **PSPACE**-membership, is to simply encode IPL back into QCPL. This is most naturally carried out by encoding the Kripke relational semantics of IPL as QBFs, however the composition of the two translations, QCPL $\to$ IPL $\to$ QCPL, does not preserve quantifier complexity. In fact, as far as the author knows, there is no 'natural' such translation in the literature and this seems to be related to various open problems in the area [12]. Phrased otherwise, how can we classify intuitionistic formulae according to the polynomial hierarchy? To address this question we will develop an over-focussed system for a fragment of IPL that we show to be already **PSPACE**-complete, by a refinement of Statman's translation. We further show that the natural encoding of proof search for this system as QBFs composes with the refined translation to, indeed, preserve quantifier complexity, as we wanted. This has further consequences on the proof theory of IPL, due to known analogous results for QCPL, and we discuss this further in Subsection 4.1 and Section 5.

Acknowledgments

I would like to thank David Baelde and Roy Dyckhoff for several enlightening discussions on various aspects of this work.

2 Proof systems as algorithms: an overview

In this section we will survey how proof systems can be used to obtain nondeterministic time and space bounds for logics, and even co-nondeterministic bounds, and give various examples. All the observations and results in this section are well-known and commonly used, or folklore. We write them here to set the context of this work and also, as a minor contribution, to serve as a brief introduction to proving complexity and decidability results for logics via proof systems.

Before we begin, let us make clear the logics and proofs we are considering. We will only consider logics that are **P**-hard, to avoid trivial results, and systems that are specified by some finite set of rules and axioms. Proofs are built up from substitution instances of these rules as directed acyclic graphs in the usual way. We will speak of 'lines of a proof' to refer to the nodes of these graphs, e.g. formulae, sequents etc. When it is convenient we will assume these graphs are trees, as is common in proof theory, although most of the results are independent of this choice. We use the metavariable L to vary over logics (construed as the set of their theorems) and the metavariable P for proof systems.

2.1 The classical approach: nondeterministic time bounds

In its purest form, a complete proof system is simply a nondeterministic procedure for a logic: if $\tau \in L$ and P is complete for L then we can nondeterministically 'guess' a P-proof of τ. While this only implies semi-decidability of L, the classical duality of proof theory and model theory yields the fundamental decidability result of mathematical logic:

Proposition 1. *If L has a complete proof system and the finite model property, it is decidable.*

This is, for instance, how it was first proved that full affine linear logic is decidable [15].[1]

[1] It was only recently, via automaton-theoretic methods, that its precise complexity was identified as **Tower**-complete [16].

More down-to-earth, in computational logic we typically deal with logics which are known to be decidable, and so a proof system gives a bound to its nondeterministic complexity:

Proposition 2. *If each $\tau \in L$ has a proof of size at most $s(|\tau|)$, then $L \in \mathbf{NTIME}(s)$.*

For instance, this allows us to deduce that multiplicative linear logic MLL, generated by the rules below, is in **NP**.

$$\frac{}{\vdash a, a^{\perp}} \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \vee B} \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}$$

Clearly, the sum of the number of connectives in the premisses of a rule is strictly less than in the conclusion, so any proof will have size quadratic in its conclusion. We can also infer, from the **coNP**-completeness of classical propositional logic CPL, one direction of the famous Cook-Reckhow theorem:

Corollary 3 ([8])**.** *If there is proof system for* CPL *which admits a polynomial-size proof for each tautology, then* $\mathbf{coNP} = \mathbf{NP}$.

This observation has given rise to the now substantial field of *proof complexity* [7].

2.2 Space bounds from proof search

We may also naturally extract *space* bounds from proof systems. These are naturally nondeterministic bounds, but we can infer deterministic space bounds by appealing to Savitch's theorem [25].[2]

Proposition 4. *If each $\tau \in L$ has a proof with lines of size $\leq w(|\tau|)$ and depth $\leq d(|\tau|)$, then $L \in \mathbf{SPACE}(wd)$.*

For example, Buss and Iemhoff have proved a $O(n^2)$ upper bound to the least depth of an LJ-proof of an intuitionistic propositional tautology of size n [4],[3] yielding a simple proof that IPL is in **PSPACE**. However, perhaps the most interesting way to bound proof search is by consideration of the entire 'space' of proofs, not just the shortest paths.

Definition 5 (Proof search space)**.** The *immediate substructure* relation between lines of a proof is defined as follows: $S \rightarrowtail S'$ if there is a rule instance with conclusion S' and some premiss S. The *proof search space* is the (directed) graph of $\rightarrowtail$. The proof search space of a line S is the search space restricted to elements that reach S.

Notice that the proof search space may be cyclic.

Proposition 6. *If the search space of a system has lines with size bounded by some function w, then its logic is in* $\mathbf{SPACE}(\exp(w))$.

Proof sketch. Proceed by loop-checking. There are only exponentially many lines of size $\leq w$ so a loop must be found after this many steps. The algorithm needs only store the current branch in memory, hence the $\exp(w)$ space bound. □

[2] In fact, it is not hard to see that this 'determinisation' of can often be conducted in-line in the proof search procedure.

[3] Incidentally, $O(n^2)$ is the best possible bound, as Buss and Iemhoff observed: there are propositional intuitionistic theorems which require quadratic depth to prove in LJ.

In fact, for many *analytic* systems, one can arrive at an improved bound:

Proposition 7. *If a sequent system satisfies the subformula property and admits contraction and weakening then its logic is in* **PSPACE**.

Proof sketch. Again proceed by loop-checking, but store all subformulae encountered during proof search in the current sequent, by contraction. Validity is preserved by weakening. Read bottom-up, sequents now accumulate subformulas, of which there are linearly many, and so a loop is always found in linear time. □

From this it is immediate that IPL and the modal logics K and $S4$ are in **PSPACE**, by the cut-elimination results for LJ, GK and $GS4$ respectively [23]. In fact, this is essentially how Heuerding, Seyfried and Zimmerman arrived at their proof of **PSPACE**-membership for various fragments of intuitionistic and modal logic [13].

2.3 Invertibility and co-nondeterminism

While classical proof theory is concerned with nondeterministic time and space bounds, the modern evolution of structural proof theory has allowed the extraction of more fine-grained bounds. A particularly important instance of this is the extraction of *co-nondeterministic* time bounds, via systems of invertible rules.

Since cyclic proof search spaces naturally induce algorithms of bounded space complexity, let us now focus on search spaces that are acyclic. In particular this means that we are considering systems without contraction.

Definition 8. A rule is *(weakly) invertible* if the validity of its conclusion implies the validity of each of its premisses, i.e. its inverse is admissible.

Proposition 9. *If a system of invertible rules has a search space with depth bounded by a function d and lines with size bounded by a function w, its logic is in* **coNTIME**(wd).

For example, consider the following *G3c* sytem which is sound and complete for CPL [21]:

$$\frac{}{\vdash \Gamma, a, \bar{a}} \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \vee B} \qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \wedge B} \tag{1}$$

Each rule instance has fewer connectives in each premiss than in its conclusion, and so the search space has both lines and depth bounded by the size of the conclusion. Consequently we arrive at a direct proof-theoretic proof that CPL $\in$ **coNP**.

This result also means that no terminating system of invertible rules can be optimal, in the sense of search complexity, for a logic that does not have a decision problem complete for some co-nondeterministic time class, unless, say, the exponential-time hypothesis fails [14]. Of course, this does not mean that it is not interesting for proof theorists to find terminating systems of invertible rules for logics, since this is often a tradeoff worth making. For instance, Brünnler and Lange did manage to find a system of invertible rules for LTL, which is **PSPACE**-complete [3]. Unsurprisingly, the depth of proof search there is exponential and so yields only a **coNEXP** bound, but for many practical applications and theoretical investigations this can be a useful property. A more simple example is the system for QCPL, also **PSPACE**-complete, that extends the *G3c* system in (1) above by the following invertible rules:

$$\frac{\vdash \Gamma, A(a)}{\vdash \Gamma, \forall x.A(x)} \qquad \frac{\vdash \Gamma, A(0), A(1)}{\vdash \Gamma, \exists x.A}$$

The source of complexity here lies in the $\exists$-rule, which means that the proof search space has exponential depth and lines of exponential size, yielding again a **coNEXP** procedure.

As noted by Dyckhoff in [9], it is an open problem whether, in the same vein, a terminating system of invertible rules exists for IPL. One of the motivations behind the work we present in Section 4 is to make some progress on this via more fine-grained translations between QCPL and IPL, admitting the possibility of mimicking directly the proof-theoretic behaviour of the calculus above in an appropriate system for IPL.

3 Bridging the time-space gap: the role of focussing

As we have seen, non-invertible and invertible rules play a somewhat dual role in proof systems, exemplified by the nondeterministic and co-nondeterministic bounds they induce. In recent years proof theorists have turned their efforts towards understanding the interactions between the two phenomena, giving rise to the notion of a *focussed* proof system, where such rules are separated into 'synchronous' and 'asynchronous phases [1, 17]. Such analysis naturally yields complexity bounds in *alternating time* hierarchies, which live in between a matching time and space class, thus bridging the gap between the bounds covered in the previous section.

We will now present a general high-level methodology for extracting such bounds from focussed systems, before continuing to a concrete case study in the next section.

3.1 A recap on alternating time hierarchies

In this section we will briefly and informally recap alternating time hierarchies in computational complexity. A full and formal exposition can be found in usual textbooks, e.g. [24]. We will assume basic familiarity with deterministic and nondeterministic Turing machines and their time and space complexities.

A *co-nondeterministic* algorithm for a language L is simply a nondeterministic algorithm for the complement language $\bar{L}$. A machine with an *oracle* L is a Turing machine equipped with the ability to query whether a string is in L in constant time. For a class of Turing machines C we write C^L for the same class with access to oracles from L. All notions of time and space complexity remain the same.

Definition 10. Let F be a class of functions $\mathbb{N} \to \mathbb{N}$. The F-hierarchy is defined as follows:

- $\Sigma_0^F = \Pi_0^F = \mathbf{TIME}(F)$.
- $\Sigma_{i+1}^F = \mathbf{NTIME}(F)^{\Sigma_i^F}$.
- $\Pi_{i+1}^F = \mathbf{coNTIME}(F)^{\Sigma_i^F}$.

We write $F\mathbf{H}$ to denote the union of all the classes above, for $i \in \mathbb{N}$. When $F = \text{poly}$, the class of polynomials, we simply write **PH** for poly**H** and write Σ_i^p and Π_i^p for its levels.

As usual, we can see these more intuitively from the viewpoint of alternating Turing machines with a bounded number of alternations.

Proposition 11 (Essentially from [5])**.** *We have the following characterisations:*

1. Σ_i^F *is the set of languages recognised by a predicate* $\exists \vec{x}_1 . \forall \vec{x}_2 \ldots Q \vec{x}_i . A$*, for* $A \in \mathbf{TIME}(F)$.
2. Π_i^F *is the set of languages recognised by a predicate* $\forall \vec{x}_1 . \exists \vec{x}_2 \ldots Q \vec{x}_i . A$*, for* $A \in \mathbf{TIME}(F)$.

We typically consider such expressions over a formal language that is complete for **TIME**(F), with quantifiers ranging over its elements. In the most common case, the polynomial hierarchy, we rely on the well-known correspondence between quantifier alternation in QBFs and the levels of **PH**.

3.2 Alternating time bounds from focussed systems

From the point of view of complexity (amongst other reasons), the most important aspect of focussed systems is the 'focussing theorem', which allows us to organise the proof search space a way that significantly reduces the alternation between invertible and non-invertible rules. It says that, during bottom-up proof search, once a non-invertible rule has been applied, simplifying a certain principal formula, we may without loss of completeness continue simplifying its auxiliary formula until it no longer admits a non-invertible simplification. It is this optimisation that makes it pertinent at all to use focussed systems in order to extract alternating time bounds. Clearly such a strong property is not uniformly available for all proof systems, and so certain design features must be present. In this section we will assume some familiarity with focussing, although the next section presents a variation of a focussed system with no such prerequisite. The discussion here is, like the previous section, somewhat informal.

We will consider an arbitrary focussed system F which, for the purpose of this work, we will assume satisfies the following further properties, which are more-or-less standard:

1. F has explicit *decide* and *release* rules which, in bottom-up proof search, mark the beginning of synchronous and asynchronous phases respectively.
2. The 'contraction' rules of F (i.e. any rule which, bottom-up, increases the size of a line) are restricted to synchronous formulae (e.g. within a decide rule).
3. Non-invertible branching rules are context-splitting, i.e. the sum of the sizes of premisses of a rule instance is bound by the size of its conclusion.

We insist on 3 so that, during a synchronous phase, we can remain within a nondeterministic computation for proof search, simply splitting the work into two parallel threads, not requiring us to introduce an extraneous universal quantifier.

Encoding proof search as an alternating predicate The simplest way to extract an alternating time bound is to consider the synthetic version of a focussed system, where there is just a single synchronous rule and a single asynchronous rule. For this we need to distinguish 'synchronous' and 'asynchronous' versions of the substructure relation from Definition 5.

Recall that a structure (i.e. a line of a focussed proof) is positive if all its formulae are synchronous, i.e. only non-invertible rules apply in bottom-up proof search, for which we will use the metavariable P. Otherwise it is negative and we will use the metavariable N.

Definition 12 (Restricted substructure relations)**.** The *synchronous immediate substructure* relation $\underset{s}{\rightarrowtail}$ is the restriction of $\rightarrowtail$ to focussed lines. The *asynchronous immediate substructure* relation $\underset{a}{\rightarrowtail}$ is the restriction of $\rightarrowtail$ to unfocussed lines.

We will employ usual annotations of these arrows from rewriting theory, in particular using an overset $*$ for reflexive transitive closure, and an overset $=$ for reflexive closure.

Definition 13 (Provability predicate)**.** The Π_k *provability predicate* for a line S is defined as,

$$\forall P_1 \overset{*}{\underset{a}{\rightarrowtail}} S.\ \exists \vec{N}_2 \overset{*}{\underset{s}{\rightarrowtail}} \vec{P}_1 \ldots Q\vec{S}_k \overset{*}{\underset{x}{\rightarrowtail}} \vec{S}_{k-1}.\ \mathit{init}(S_k)$$

where:

- Q is $\forall$ or $\exists$, x is a or s, and $\vec{S}_{k-1}$ and $\vec{S}_k$ are $\vec{P}_{k-1}$ and $\vec{N}_k$ or $\vec{N}_{k-1}$ and $\vec{P}_k$, as appropriate;
- $\mathit{init}(S)$ is the predicate "S is a correct initial line of F".

The Σ_k provability predicate is defined similarly, beginning with an existential quantifier instead of a universal.

Now let us recall the usual notions of depth for a focussed system.

Definition 14 (Decide and release depth)**.** The *decide depth* of a line S, written $dd(S)$ is the maximum number of decide rule instances in a path of its search space. The *release depth*, $rd(S)$, is defined analogously.

Of course, dd and rd only ever differ by at most 1, but it is useful to have both notions rather than explicitly dealing with each situation. We can now state the correctness of the provability predicate.

Proposition 15. *We have the following:*

1. *The Π_k provability predicate accepts provable lines N such that $dd(S) + rd(S) < k$.*
2. *The Σ_k provability predicate accepts provable lines P such that $dd(S) + rd(S) \leq k$.*

Proof sketch. Follows from the completeness of the focussed system F. □

Finally we can use this to obtain bounds on the complexity of the provability predicates from bounds on the size of lines in F.

Theorem 16. *Suppose the lines of F have size bounded by a function w. Then the Π_k (or Σ_k) provability predicate can be computed in $\Pi_k^{\exp(w)}$ (or $\Sigma_k^{\exp(w)}$, respectively).*

If F is furthermore contraction-free, i.e. line size decreases bottom-up, then the Π_k (or Σ_k) provability predicate can be computed in Π_k^p (or Σ_k^p, respectively).

Proof sketch. Each expression $\forall \vec{P}_i \overset{*}{\underset{a}{\rightarrowtail}} \vec{N}_{i+1}$ is equivalent to an expression $\forall \vec{S}^1 \underset{a}{\rightarrowtail} \vec{N}_{i+1}. \forall \vec{S}^2 \overset{=}{\underset{a}{\rightarrowtail}} \vec{S}^1 \ldots \forall \vec{P}_i \overset{=}{\underset{a}{\rightarrowtail}} S^n$, where $n = w(|S|)$, each S^i_j varies over sequents of size $\leq w(|S|)$, and each tuple $\vec{S}^i$ has length w^i. This relies on the conventions 2 and 3, and we can similarly 'expand' existential quantifiers in this way. Finally every atomic predicate ($\underset{a}{\rightarrowtail}$, $\underset{s}{\rightarrowtail}$, $=$ and *init*) can easily be checked in time polyomial in $w(|S|)$.

If F is contraction-free then the proof is the same except we need only insist that the tuples $\vec{S}^i$ have length w, which we can assume to be linear by contraction-freeness. □

In fact, notice that the use of dd and rd is somewhat non-optimal, in the sense that, for example, the Π_k provability predicate accepts also those lines S which simply have *some* proof whose decide depth and release depth have sum $\leq k$. However, the advantage of using the decide and release depths of the entire search space of S is that we can efficiently calculate such values, as exemplified in [22]. Consequently, we can use the theorem above to arrive at an encoding of formulae from a logic L into a quantified language that can express the provability predicates of a focussed system for L. In particular we have the following:

Corollary 17 (Encoding as QBF)**.** *If F is contraction-free then the provability predicates for F induce a polynomial-time encoding of its logic into* QCPL.

3.3 Scope for improvement? The reason to over-focus

Until now we have more-or-less formalised well-known intuitions about the algorithmics of proof search in focussed systems from a complexity-theoretic point of view. However it is at this point that we remark upon a particular shortcoming of the framework: the synchronous-asynchronous dichotomy in focussing makes no allowance for *deterministic* computation. This corresponds to invertible non-branching rules and does not contribute to alternation complexity in proof search, whether in an asynchronous or synchronous phase.

Here is an example of the issue we are highlighting. Consider the following encoding in IPL of a SAT instance A over variables $\vec{x}$,

$$
\begin{array}{rcl}
A_0 & := & A \\
A_{i+1} & := & (A_i \supset a_i) \supset ((x_i \supset a_i) \lor (\neg x_i \supset a_i))
\end{array}
$$

where the variables a_i are fresh. (Notice that $A \in \mathsf{SAT}$ if and only if $A_{n+1} \in \mathsf{IPL}$.[4])

Let us consider a proof in the usual focussed system LJF for IPL, using the notation of [19], of A_{n+1}, built up inductively:[5]

$$
\dfrac{S_l,S_r\,\dfrac{D_r\,\dfrac{\dfrac{R_r\,\dfrac{\dfrac{S_l,S_r\,\dfrac{D_l\,\dfrac{\dfrac{R_r\,\dfrac{\dfrac{\vdots\ \mathit{IH}}{\pm x_n \Uparrow \cdot \vdash A_n \Uparrow \cdot}}{\pm x_n \vdash A_n \Downarrow} \quad \dfrac{}{\cdot \Downarrow a_n \vdash a_n}}{\pm x_n \Downarrow A_n \supset a_n \vdash a_n}}{A_n \supset a_n, \pm x_n \Uparrow \cdot \vdash \cdot \Uparrow a_n}}{A_n \supset a_n \Uparrow \pm x_n \vdash a_n \Uparrow \cdot}}{A_n \supset a_n \Uparrow \cdot \vdash (\pm x_n \supset a_n) \Uparrow \cdot}}{A_n \supset a_n \vdash (\pm x_n \supset a_n) \Downarrow}}{A_n \supset a_n \vdash (x_n \supset a_n) \lor (\neg x_n \supset a_n) \Downarrow}}{A_n \supset a_n \Uparrow \cdot \vdash \cdot \Uparrow (x_n \supset a_n) \lor (\neg x_n \supset a_n)}}{\cdot \Uparrow A_n \supset a_n \vdash (x_n \supset a_n) \lor (\neg x_n \supset a_n) \Uparrow \cdot}}{\cdot \Uparrow \cdot \vdash (A_n \supset a_n) \supset ((x_n \supset a_n) \lor (\neg x_n \supset a_n) \Uparrow \cdot}
$$

Notice that the only proofs have the format above and their decide and release depth increase linearly with the number of variables in A. By the results in the previous section this yields only a Σ^p_{2k} bound for formulae over k variables, far from the optimal bound of $\mathbf{NP} = \Sigma^p_1$.

On closer inspection, observe that the only reason that so many phase alternations occur is the application of right-implication between release and decide steps, which is invertible and non-branching. Consequently, we can actually encode the proof search procedure using only existential quantifiers, in the spirit of the previous subsection, and so the dynamics of proof search 'morally' represent an $\mathbf{NP}$ procedure.

In the next section we will propose a stricter form of focussing where we explicitly admit 'deterministic' rules in synchronous phases, without releasing the focus. In this way the results of this section will be applied to avoid the apparent blowup in the complexity bound induced by the focussed proofs above.

[4] This is simply a special case of the refined Statman translation we show later on.

[5] Notice that there are just as many proofs as there are satisfying assignments.

4 Case study: a fragment of IPL

We will now consider a concrete case study, using a variation of focussing to delineate tautologies of IPL according to the polynomial hierarchy, under the results from the previous section.

Let us from now on write $\vDash_i$ for intuitionistic entailment and $\vDash_c$ for classical entailment.

4.1 PSPACE-hardness of IPL and some motivations

Statman proved the **PSPACE**-hardness of IPL in [26] by encoding QCPL, known to be **PSPACE**-complete, into it. The fundamental observation is that,

$$\begin{array}{lcl} \vDash_c \forall x.A & \iff & \vDash_i (x \vee \neg x) \supset A' \\ \vDash_c \exists x.A & \iff & \vDash_i (x \supset A') \vee (\neg x \supset A') \end{array} \tag{2}$$

where A' is obtained by the inductive hypothesis.[6] The problem with this is that the $\exists$ case includes two copies of A', and so the size of the encoding grows exponentially with the number of $\exists$ quantifiers. One way around this is to instead map QBFs to intuitionistic *circuits*, which can share the copy of A' between the two disjuncts, yielding polynomial-size circuits.[7] From here one can simply use Tseitin extension variables to encode the local conditions of the circuit as a formula, and this is what Statman does in his translation.

Definition 18 (Statman's translation)**.** For a closed prenex QBF $A = Q_n x_n . \ldots . Q_1 x_1 . A_0$, let A_i be the subformula $Q_i x_i . \ldots Q_1 x_1 . A_0$. Define $A_i^{\checkmark}$ by induction on i as follows,

$$\begin{array}{rcl} A_0^{\checkmark} & := & a_0 \equiv \neg\neg A_0 \\ A_{i+1}^{\checkmark} & := & \begin{cases} a_{i+1} \equiv ((x_{i+1} \vee \neg x_{i+1}) \supset a_i) & \text{if } Q_{i+1} \text{ is } \forall \\ a_{i+1} \equiv ((x_{i+1} \supset a_i) \vee (\neg x_{i+1} \supset a_i)) & \text{if } Q_{i+1} \text{ is } \exists \end{cases} \end{array}$$

where each a_i is fresh. Finally, define A^* as $\bigwedge_{i=0}^{n} A_0^{\checkmark} \supset a_n$.

Theorem 19 (Statman)**.** *$\vDash_c A$ if and only if $\vDash_i A^*$.*

While this indeed yields a proof of the **PSPACE**-hardness of IPL, there are many issues and questions which naturally arise.

Delineating the polynomial hierarchy in IPL There are many encodings from IPL back into QCPL, the most immediate of which arises by simply encoding the Kripke relational semantics of IPL. Other encodings can be obtained by encoding the topological semantics or game semantics [2], and of course by encoding proof search too. However, as far as we know, there is no 'natural' translation that acts as an 'inverse' of the Statman translation in the sense that quantifier complexity is preserved by the composition of the two translations. More precisely we ask the following:

Question 20. *Are there 'natural' encodings $t_1 : \mathsf{QCPL} \to \mathsf{IPL}$ and $t_2 : \mathsf{IPL} \to \mathsf{QCPL}$, such that:*

- *$\vDash_c A$ if and only if $\vDash_i t_1(A)$.*
- *$\vDash_i A$ if and only if $\vDash_c t_2(A)$.*

[6] We ignore the base case here for simplicity.

[7] In fact, it is not hard to see that these circuits have type-level 1, meaning that the class of type 1 intuitionistic circuits is already **PSPACE**-complete.

- $t_2 \circ t_1$ *preserves quantifier complexity.*

The purpose of this section is to identify a well-behaved variant of Statman's translation and a corresponding proof system with which we can resolve the above question via a suitable encoding of proof search.

Invertible rules for IPL? Dyckhoff asks in [9] whether there exists a (terminating) system of invertible rules sound and complete for IPL. This is also noted as an open problem by Buss and Iemhoff in [4]. As we have already mentioned, such a system cannot exhibit optimal proof search complexity, but it is of natural proof theoretic interest.

In fact a positive answer to Question 20 above would partially resolve this open problem: since we already have a terminating system of invertible rules for QCPL, cf. Section 1, we can just consider the translations of those rules under t_1. Invertibility and termination follows by correctness of the encoding, and the resulting system will be complete for the image of t_1.

To obtain a system complete for all of IPL we must also appeal to t_2: for each formula A of IPL we may construct a rule whose conclusion is A and whose premisses are the t_1-image of the premisses of a QCPL rule whose conclusion is $t_2(A)$. If we only consider the 'synthetic' rules, decomposing an entire block of existentials quantifiers or universal quantifiers at once, the system is terminating by the fact that $t_2 \circ t_1$ preserves quantifier complexity and the corresponding termination argument for the QCPL-calculus. The existence of such a system for an expressive fragment[8] PTPL of IPL is a direct consequence of the results in this section. While this is not an ideal formulation of a proof system, since the number of premisses for a rule is unbounded, the discovery of a 'natural' such t_2 could provide useful intuitions towards finding a *bona fide* system of terminating invertible rules for IPL.

4.2 A refined translation and the positive Tseitin fragment

One problem with resolving Question 20 above is with Statman's translation itself: it is somehow too insensitive to the structure of proofs induced by the translation. At the same time, usual proof systems do not handle well Tseitin variables, which act as abbreviations: while these are essentially deterministic pieces of information, a proof system would naturally see at least one direction of the equivalence as a nondeterministic choice.

However notice that, in (2), A' only ever occurs in positive context, and so we only need one direction of the Tseitin equivalences, of the form $A \supset a$ and not $a \supset A$. We use this to construct a refined version of the translation that allows us to have better control over the proof theory of formulae in its image.

Definition 21. We mutually define the following classes of formulae:

$$L \quad ::= \quad x \quad | \quad \bot \quad | \quad R \supset a \quad | \quad L \wedge L \quad | \quad L \vee L$$

$$R \quad ::= \quad x \quad | \quad \bot \quad | \quad L \supset R \quad | \quad R \wedge R \quad | \quad R \vee R$$

An R-formula is called a *positive Tseitin* formula. The *positive Tseitin fragment* of IPL, denoted PTPL, is the set of intuitionistically valid positive Tseitin formulae.

One particularly appealing property of positive Tseitin formulae is the following result:

Proposition 22. *An L-formula is either a disjunction or a Harrop formula.*

[8] In particular it is **PSPACE**-complete.

This means that, in bottom-up proof search, once all disjunctions on the left have been decomposed, we can always make some choice of disjunct on the right. This also gives us access to a simple contraction-free and cut-free system for PTPL, as we will see soon.

Remark 23. Note that L-formulae are not, in general, hereditary Harrop, for essentially any formulation of hereditary Harrop formulae, cf. [18].

As already observed by Švejdar in [27], the double negation in the base case of Statman's translation is unnecessary for closed QBFs, although it does have some effect on the structure of proofs. From a proof-search-as-computation point of view, it might be more natural to use the Kolmogorov double-negation translation, which places $\neg\neg$ in front of every subformula, since intuitionistic proofs of the resulting formulae can directly simulate classical proofs [10]. However we do not address this issue here, being beyond the scope of this work, and we will simply include a rule for 'evaluation'. We make some comments on this in the next section.

Definition 24 (Refined translation)**.** We define the following translation from prenex QBFs to (unquantified) propositional formulae,

$$\begin{array}{rcl} \langle A_0 \rangle & := & A_0 \\ \langle \forall x.A \rangle & := & (x \vee \neg x) \supset \langle A \rangle \\ \langle \exists x.A \rangle & := & ((\langle A \rangle \supset a) \supset ((x \supset a) \vee (\neg x \supset a)) \end{array}$$

where A_0 is quantifier-free, A is in prenex normal form and a is fresh.

Theorem 25. *If A is a closed prenex QBF, then $A \in$ QCPL if and only if $\langle A \rangle \in$ PTPL.*

Before proving this we will need some intermediate results.

Lemma 26. *We have the following:*

1. $\vDash_i A(\top) \supset (x \supset A(x))$.
2. $\vDash_i A(\bot) \supset (\neg x \supset A(x))$.
3. *If* $\vDash_i x \supset A(x)$ *then* $\vDash_i A(\top)$.
4. *If* $\vDash_i \neg x \supset A(x)$ *then* $\vDash_i A(\bot)$.

Proof sketch. For 1, we show that the sequent,

$$\Gamma, x, A(\top) \vdash A(x)$$

is provable in an intuitionistic system by induction on the structure of A, whence the result follows by setting Γ empty and application of implication-right.[9] 2 is proved similarly.

For 3, we simply substitute $\top$ for x to achieve $\top \supset A(\top)$, whence $A(\top)$ follows. 4 follows similarly after substitution of $\bot$ for x.

□

Proposition 27 (Positive Tseitin extension)**.** *Let $B(x)$ be a formula where x occurs positively and let A be a formula not containing a variable a. $\vDash_i B(A)$ if and only if $\vDash_i (A \supset a) \supset B(a)$.*

Proof. The left-right direction is essentially the correctness of deep inference under positive contexts, whereas the right-left direction follows by substituting A for a. □

[9] This is most easily proved in the Vorob'ev-Hudelmaier-Dyckhoff calculus, sometimes known as *G4ip* [9].

Corollary 28. $\vDash_i \langle \exists x.A \rangle$ *if and only if* $\vDash_i (x \supset \langle A \rangle) \vee (\neg x \supset \langle A \rangle)$.

Proof. By inspection of the definition of $\langle \cdot \rangle$, Tseitin variables only ever occur positively. □

We are now ready to prove the correctness of our translation.

Proof of Theorem 25. Recall that $\neg A$ is shorthand for $A \supset \bot$, whence the fact that $\langle A \rangle$ is positive Tseitin follows by inspection. Now let us show that $A \in \mathsf{QCPL} \iff \langle A \rangle \in \mathsf{IPL}$ by induction on the structure of A. If A is quantifier-free then it must be a logical combination of constants by closedness, and so $\langle A \rangle = A \in \mathsf{IPL} \iff A \in \mathsf{QCPL}$ by conservativity of CPL over IPL for variable-free formulae.

Now, suppose A is $\forall x.B$:

$$\begin{array}{lll} \vDash_c \forall x.B(x) & \iff & \vDash_c B(\top) \text{ and } \vDash_c B(\bot) \\ & \iff & \vDash_i \langle B(\top) \rangle \text{ and } \vDash_i \langle B(\bot) \rangle \qquad \text{by inductive hypothesis} \\ & \iff & \vDash_i x \supset \langle B(x) \rangle \text{ and } \vDash_i \neg x \supset B(x) \quad \text{by Lemma 26} \\ & \iff & \vDash_i (x \vee \neg x) \supset \langle B(x) \rangle \qquad \text{by distributivity} \end{array}$$

Otherwise, A is $\exists x.B(x)$:

$$\begin{array}{lll} \vDash_c \exists x.B(x) & \iff & \vDash_c B(\top) \text{ or } \vDash_c B(\bot) \qquad \text{since } A \text{ is closed} \\ & \iff & \vDash_i \langle B(\top) \rangle \text{ or } \vDash_i \langle B(\bot) \rangle \qquad \text{by inductive hypothesis} \\ & \iff & \vDash_i x \supset \langle B(x) \rangle \text{ or } \vDash_i \neg x \supset \langle B(x) \rangle \quad \text{by Lemma 26} \\ & \iff & \vDash_i (x \supset \langle B(x) \rangle) \vee (\neg x \supset \langle B(x) \rangle) \quad \text{by the disjunction property} \\ & \iff & \vDash_i \langle \exists x.B(x) \rangle \qquad \text{by Corollary 28} \end{array}$$

□

Corollary 29. PTPL *is already* **PSPACE***-complete.*

We will now present a proof system for PTPL. Before doing so, we recall an important result from Statman's paper.

Lemma 30 (Evaluation)**.** *Let $l(x)$ be either x or $\neg x$ for each variable x, and let A be a quantifier-free formula with variables amongst $\vec{x}$. Then $l(\vec{x}) \vDash_c A$ if and only if $l(\vec{x}) \vDash_i A$.*

Proof. The right-left direction is clear so we prove the left-right direction. Notice that $l(\vec{x})$ induces a complete (classical) model determining formulae over $\vec{x}$ (including A). Suppose $l(\vec{x}) \vDash_c A$ and proceed by structural induction on A.

If A is a variable a then we must have that $l(\vec{x})$ contains a, whence $l(\vec{x}) \vDash_i A$.

If A is a disjunction $A_1 \vee A_2$ then we must have that $l(\vec{x}) \vDash_c A_i$ for some $i \in \{1, 2\}$, whence $l(\vec{x}) \vDash_i A_i$ by the inductive hypothesis and so $l(\vec{x}) \vDash_i A$.

If A is a conjunction $A_1 \wedge A_2$ then we must have that $l(\vec{x}) \vDash_c A_i$ for each $i \in \{1, 2\}$, whence $l(\vec{x}) \vDash_i A_i$ by the inductive hypothesis and so $l(\vec{x}) \vDash_i A$.

If A is an implication $A_1 \supset A_2$ then we must have that $l(\vec{x}) \vDash_c \neg A_1$ or $l(\vec{x}) \vDash_c A_2$, so $l(\vec{x}), A_1 \vDash_i$ or $l(\vec{x}) \vDash_i A_2$. In either case we have that $l(\vec{x}) \vDash_i A$, by thinning and the deduction theorem. □

Definition 31 (Proof system)**.** We define the system LPT as follows:

$$\bot\text{-}l \frac{}{\Gamma, \bot \vdash A} \qquad eval \frac{l(\vec{a}) \vDash_c A(\vec{a})}{\Gamma, l(\vec{a}) \vdash A(\vec{a})}$$

$$\vee\text{-}l\ \frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C} \qquad \supset\text{-}l\ \frac{\Gamma \vdash A}{\Gamma, A \supset a \vdash a} \qquad \wedge\text{-}l\ \frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}$$

$$\wedge\text{-}r\ \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \supset\text{-}r\ \frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B} \qquad \vee\text{-}r\ \frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 \vee A_2}\ i \in \{1, 2\}$$

By inspection of these rules, we immediately have the following:

Proposition 32. *Any LPT-derivation with a positive Tseitin conclusion contains only L-formulae on the left and R-formulae on the right.*

This result, together with Proposition 22, allows us to easily conclude the completeness of LPT with respect to PTPL.

Theorem 33. *LPT is sound and complete for* PTPL.

Proof. Soundness is clear from inspection of the rules. For completness, we give a bottom-up proof search algorithm for some PTPL tautology. First, apply the invertible rules $\bot$-l, *eval*, $\wedge$-l, $\wedge$-r, $\vee$-l and $\supset$ -r as long as possible. Now formulae must be of the form $A \supset a$ on the left and disjunctions on the right, by Proposition 32. Since the former is always a Harrop formulae, we can apply $\vee$-r. We continue to decompose the right hand side until it is atomic, say b, at which point there must be some implication $B \supset b$ on the left for which we can make an application of $\supset$ -l, by the completeness of the usual calculus LJ. Thus we can always apply some rule and this procedure must terminate since, bottom-up, lines strictly decrease in size. In the base case, when sequents are atomic, there is always an appropriate application of *eval*. □

4.3 An 'over-focussed' system for PTPL

Usually, focussed systems classify formulae as either 'negative' or 'positive'. In light of our comments in the previous section, we will be slightly more fine-grained in our classification, also distinguishing 'deterministic' formulae.

Remark 34. Notice that there is a peculiar phenomenon in LPT that the conjunction rules are both invertible. From the point of view of focussing, this is because we have actually used two different versions of conjunction, positive ($\wedge^+$) on the left and negative ($\wedge^-$) on the right. However, notice that LPT respects the polarity of a conjunction symbol in a formula (in the sense of even or odd depth to the left of implications), and so the distinction is already present at the level of formulae: L-formulae are composed by $\wedge^+$, while R-formulae are composed by $\wedge^-$. We will typically omit such annotation when it is clear from context. For similar reasons (and by the same argument), we will always assume that atoms are negative on the left of a sequent and positive on the right.

Definition 35 (Classification)**.** A *positive* formula is a $\wedge^+$ or $\vee$ formula (or a positive atom). A *negative* formula is a $\wedge^-$ or $\supset$ formula (or a negative atom). A *deterministic* formula is a $\wedge^+$ or $\supset$ formula (or an atom). We will use the following metavariables:

M	:	negative and not deterministic	$\wedge^-$
N	:	negative	$\wedge^-, \supset$
O	:	deterministic	$\wedge^+, \supset$
P	:	positive	$\wedge^+, \vee$
Q	:	positive and not deterministic	$\vee$

Co-nondeterministic phase

$$\frac{}{\Gamma \Uparrow \bot, \Delta \vdash \Uparrow P} \qquad {}_{eval\Uparrow}\frac{l(\vec{a}) \vDash_c P(\vec{a})}{\Gamma, l(\vec{a}) \Uparrow \cdot \vdash \Uparrow P(\vec{a})}$$

$$\frac{\Gamma \Uparrow A, \Delta \vdash C \Uparrow \quad \Gamma \Uparrow B, \Delta \vdash C \Uparrow}{\Gamma \Uparrow A \vee B, \Delta \vdash C \Uparrow} \qquad \frac{\Gamma \Uparrow A, B, \Delta \vdash C \Uparrow}{\Gamma \Uparrow A \wedge^{+} B, \Delta \vdash C \Uparrow} \qquad {}_{S_l}\frac{\Gamma, N \Uparrow \Delta \vdash C \Uparrow}{\Gamma \Uparrow N, \Delta \vdash C \Uparrow}$$

$$\frac{\Gamma \Uparrow \cdot \vdash A \Uparrow \quad \Gamma \Uparrow \cdot \vdash B \Uparrow}{\Gamma \Uparrow \cdot \vdash A \wedge^{-} B \Uparrow} \qquad \frac{\Gamma \Uparrow A \vdash B \Uparrow}{\Gamma \Uparrow \cdot \vdash A \supset B \Uparrow} \qquad {}_{S_r}\frac{\Gamma \Uparrow \cdot \vdash \Uparrow P}{\Gamma \Uparrow \cdot \vdash P \Uparrow}$$

Nondeterministic phase

$${}_{D_l}\frac{\Gamma;\ \Downarrow N \vdash P}{\Gamma, N \Uparrow \cdot \vdash \Uparrow P} \qquad {}_{D_r}\frac{\Gamma;\ \vdash P^{\not a} \Downarrow}{\Gamma \Uparrow \cdot \vdash \Uparrow P^{\not a}} \qquad \frac{\Gamma; \Delta \vdash A \Downarrow}{\Gamma; \Delta \Downarrow A \supset a \vdash a} \qquad \frac{\Gamma; \Delta \vdash A_i \Downarrow}{\Gamma; \Delta \vdash A_1 \vee A_2 \Downarrow}\, i \in \{1,2\}$$

where D_l and D_r apply only if the conclusion is not an instance of *eval* $\Uparrow$, and $P^{\not a}$ is not atomic.

$$\frac{\Gamma; \Delta, A_i \Downarrow A_j \vdash B}{\Gamma; \Delta \Downarrow A_1 \wedge^{+} A_2 \vdash B}\, \{i,j\} = \{1,2\} \qquad \frac{\Gamma; \Delta \Downarrow A \vdash B}{\Gamma; \Delta \vdash A \supset B \Downarrow} \qquad \frac{\Gamma; \Delta, A \vdash B \Downarrow}{\Gamma; \Delta \vdash A \supset B \Downarrow}$$

$$\frac{\Gamma; A, B, \Delta\ \mathcal{L} \vdash \mathcal{R}}{\Gamma; A \wedge^{+} B, \Delta\ \mathcal{L} \vdash \mathcal{R}} \qquad {}_{s_l^N}\frac{\Gamma, N; \Delta\ \mathcal{L} \vdash \mathcal{R}}{\Gamma; N, \Delta\ \mathcal{L} \vdash \mathcal{R}} \qquad \frac{\Gamma; \Delta, B \Downarrow A \vdash C}{\Gamma; \Delta \Downarrow A \vdash B \supset C}$$

where either $\mathcal{L}$ is $\Downarrow C$ and $\mathcal{R}$ is D or $\mathcal{L}$ is empty and $\mathcal{R}$ is $D \Downarrow$.

$${}_{s_l^Q}\frac{\Gamma; \Delta, Q \Downarrow \cdot \vdash A}{\Gamma; \Delta \Downarrow Q \vdash A} \quad {}_{s_l^a}\frac{\Gamma, a; \Delta \Downarrow \cdot \vdash A}{\Gamma; \Delta \Downarrow a \vdash A} \quad {}_{d_l}\frac{\Gamma;\ \Downarrow N \vdash P}{\Gamma, N;\ \Downarrow \cdot \vdash P} \quad {}_{d_l^a}\frac{\Gamma;\ \Downarrow N \vdash a}{\Gamma, N;\ \vdash a \Downarrow} \quad {}_{d_r}\frac{\Gamma;\ \vdash P \Downarrow}{\Gamma;\ \Downarrow \cdot \vdash P}$$

$$\frac{}{\Gamma; \Delta \Downarrow \bot \vdash R} \qquad {}_{eval\Downarrow}\frac{l(\vec{a}) \vDash_c M^a(\vec{a})}{\Gamma, l(\vec{a}); \Delta \vdash M^a(\vec{a}) \Downarrow} \qquad {}_{R_l}\frac{\Gamma \Uparrow \vec{Q} \vdash M^a \Uparrow}{\Gamma; \vec{Q} \vdash M^a \Downarrow} \qquad {}_{R_r}\frac{\Gamma \Uparrow \vec{Q} \vdash R \Uparrow}{\Gamma; \vec{Q} \Downarrow \cdot \vdash R}$$

where M^a is either M or an atom and R is either M or, as long as $\vec{Q}$ is nonempty, P. Furthermore, the rules R_l and R_r apply only if the conclusion is not an instance of *eval* $\Downarrow$.

Figure 1: Rules of the system *LPTF*.

The reason for making such distinctions is that we will admit deterministic computations in both the negative and positive phases, allowing the decomposition of positive deterministic formulae on the left and negative deterministic formulae on the right. Referring to the previous section, the corresponding such rules are invertible and non-branching. Since this is more restrictive than usual focussing, we will call this an 'over-focussed' system, and refer to the phases as 'co-nondeterministic' and 'nondeterministic' rather than 'asynchronous' and 'synchronous', respectively, which this is more compatible with our complexity-theoretic point of view.

Definition 36 (Focussed system)**.** We define the system *LPTF* by the rules in Figure 1. A *proof* of a formula A is simply a *LPTF*-proof of $\cdot \Uparrow \cdot \vdash A \Uparrow$ or $;\ \vdash A \Downarrow$.

It is worth taking a moment to explain the format of lines and the dynamics of proof search

in this system. There are three kinds of sequent in this system:

$$\Gamma \Uparrow \Delta \vdash \Lambda_1 \Uparrow \Lambda_2 \tag{3}$$
$$\Gamma; \Delta \Downarrow A \vdash B \tag{4}$$
$$\Gamma; \Delta \vdash A \Downarrow \tag{5}$$

where Γ and Δ are multisets of formulae and $|\Lambda_1 \sqcup \Lambda_2| = 1$. (3) is called an *unfocussed* sequent, whereas (4) and (5) are *focussed*, left and right respectively. In both the latter cases the 'focus' of the sequent is A, whereas Δ consists of the 'residues'.

Bottom-up proof search operates as follows. First, a co-nondeterministic phase is conducted exactly analogous to an asynchronous phase in, say, *LJF*, until we reach a positive sequent. Then we enter a non-deterministic phase after 'deciding' on a focus. This is similar to a synchronous phase in *LJF*, except that if we reach a deterministic formula we must keep decomposing one of its auxiliary formulae. Other auxiliary formulae are temporarily stored as residues and are reduced until they become positive, when they are stored for real, or negative and non-deterministic. The phase is complete once only some invertible branching rules apply to the focus and the remaining residues, when we release and enter again the co-nondeterministic phase. If there is no such rule, i.e. there are no remaining residues and the focus is atomic, then we must 'redecide' a formula without releasing and remain in the nondeterministic phase.

Theorem 37. *LPTF is sound and complete for* PTPL.

Proof sketch. By usual rule permutation arguments. The general idea is that we can permute non-branching bipoles upwards in a proof until an auxiliary formula becomes principal again. The prototypical proof manipulation has the following format,

$$\dfrac{\exists\dfrac{\forall\dfrac{\exists\dfrac{A_0^0, A_1, B_0}{A_0, A_1, B_0} \quad \exists\dfrac{A_0^1, A_1, B_1}{A_0, A_1, B_1}}{A_0, A_1, B}}{\det\dfrac{A_0, A_1, B'}{A, B'}}}{\exists\dfrac{A, B'}{A', B'}} \quad \mapsto \quad \dfrac{\forall\dfrac{\exists\dfrac{\det\dfrac{\exists\dfrac{A_0^0, A_1, B_0}{A_0, A_1, B_0}}{A, B_0}}{A', B_0} \quad \exists\dfrac{\det\dfrac{\exists\dfrac{A_0^1, A_1, B_1}{A_0, A_1, B_1}}{A, B_1}}{A', B_1}}{A', B}}{\exists\dfrac{A', B}{A', B'}}$$

where non-invertible rules are marked $\exists$, invertible branching rules are marked $\forall$ and invertible non-branching rules are marked det. What corresponds to usual bipoles of a focussed proof appear in the same colour connected component. □

Remark 38. After a deterministic step during a nondeterministic phase we cannot, in general, remain focussed on *both* auxiliary formulae, in the sense of multifocussing, since there may be a strict order of dependency between the auxiliary formulae. This is exemplified by the *LJF*-derivation in Figure 2, again using the notation of [19]. Here the marked $\wedge^+$ rule has two auxiliary formulae, of which $a \vee b$ must be decomposed first since $(a \supset d) \wedge^- (b \supset d)$ is decomposed differently depending on which branch is taken, coloured in red in the derivation. In an over-focussed system, such a proof would therefore have to remain focussed on $a \vee b$ in the premiss of the marked $\wedge^+$ step.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a \vdash a \Downarrow} \quad \dfrac{}{a \Downarrow d \vdash d}}{a \Downarrow a \supset d \vdash d}}{a \Downarrow (a \supset d) \wedge^{-} (b \supset d) \vdash d}\, D_l}{a, (a \supset d) \wedge^{-} (b \supset d) \Uparrow \cdot \vdash \Uparrow d}\, 2S_l}{\Uparrow a, (a \supset d) \wedge^{-} (b \supset d) \vdash \Uparrow d} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{b \vdash b \Downarrow} \quad \dfrac{}{b \Downarrow d \vdash d}}{b \Downarrow b \supset d \vdash d}}{b \Downarrow (a \supset d) \wedge^{-} (b \supset d) \vdash d}\, D_l}{b, (a \supset d) \wedge^{-} (b \supset d) \Uparrow \cdot \vdash \Uparrow d}\, 2S_l}{\Uparrow b, (a \supset d) \wedge^{-} (b \supset d) \vdash \Uparrow d}}{\dfrac{\dfrac{\Uparrow (a \vee b), (a \supset d) \wedge^{-} (b \supset d) \vdash \Uparrow d}{\Uparrow (a \vee b) \wedge^{+} ((a \supset d) \wedge^{-} (b \supset d)) \vdash \Uparrow d}\, \wedge^{+}}{\Downarrow (a \vee b) \wedge^{+} ((a \supset d) \wedge^{-} (b \supset d)) \vdash d}\, R_l}{\Downarrow ((a \vee b) \wedge^{+} ((a \supset d) \wedge^{-} (b \supset d))) \wedge^{-} c \vdash d}$$

Figure 2: An example of dependency between auxiliary formulae of a deterministic step in LJF.

4.4 Compatibility of proof search with the refined translation

We will now pick up from the complexity analysis of Section 3, specialising to our over-focussed sytem $LPTF$. Our notions of decide depth (dd) and release depth (rd) remain the same, counting only D_l or D_r and R_l or R_r steps respectively, and can still be calculated in polynomial time from a formula by a 'worst case' analysis, similar to that in [22]. Most importantly, the appropriate versions of Proposition 15 and Theorem 16 remain valid for $LPTF$. Consequently, we obtain from $LPTF$ an encoding of PTPL back into QCPL.

Definition 39. For a positive Tseitin formula A, let $\lceil A \rceil$ denote the sequent obtained from A after maximally applying the deterministic rules $\wedge$-l and $\supset$ -r, bottom-up. Now we define,

$$[A] \quad := \quad \begin{cases} \text{the } \Sigma_{dd(A)+rd(A)} \text{ provability predicate for } \lceil A \rceil & \text{if } \lceil A \rceil \text{ is positive} \\ \text{the } \Pi_{1+dd(A)+rd(A)} \text{ provability predicate for } \lceil A \rceil & \text{otherwise} \end{cases}$$

with respect to the system $LPTF$.

In light of Theorem 16 and Corollary 17, we have the following:

Proposition 40. $[\cdot]$ *induces an encoding from* PTPL *to* QCPL*, with quantifier complexity in the image matching that of the corresponding provability predicate.*

By examining the dynamics of proof search, let us explicitly calculate the decide and release depths of formulae in the image of our refined translation.

Lemma 41. *We have the following:*

1. $dd(\vec{a} \Uparrow \cdot \vdash \langle \forall x.A \rangle \Uparrow) = dd(\vec{a}, x \Uparrow \cdot \vdash \langle A \rangle \Uparrow)$.
2. $dd(\vec{a} \Uparrow \cdot \vdash \langle \exists x.A \rangle \Uparrow) = 1 + dd(\vec{a};\ \vdash \langle \exists x.A \rangle \Downarrow)$.
3. $dd(\vec{a};\ \vdash \langle \exists x.A \rangle \Downarrow) = dd(\vec{a}, x;\ \vdash \langle A \rangle \Downarrow)$.
4. $dd(\vec{a};\ \vdash \langle \forall x.A \rangle \Downarrow) = dd(\vec{a} \Uparrow \cdot \vdash \langle \forall x.A \rangle \Uparrow)$.

and:

5. $rd(\vec{a} \Uparrow \cdot \vdash \langle \forall x.A \rangle \Uparrow) = rd(\vec{a}, x \Uparrow \cdot \vdash \langle A \rangle \Uparrow)$.

6. $rd(\vec{a} \Uparrow \cdot \vdash \langle \exists x.A \rangle \Uparrow) = rd(\vec{a};\ \vdash \langle \exists x.A \rangle \Downarrow)$.
7. $rd(\vec{a};\ \vdash \langle \exists x.A \rangle \Downarrow) = rd(\vec{a}, x;\ \vdash \langle A \rangle \Downarrow)$.
8. $rd(\vec{a};\ \vdash \langle \forall x.A \rangle \Downarrow) = 1 + rd(\vec{a} \Uparrow \cdot \vdash \langle \forall x.A \rangle \Uparrow)$.

Finally, we can use these calculations to show that our encoding of proof search for *LPTF* in QCPL is well-calibrated with our refined Statman translation from QCPL to PTPL, in the sense of Subsection 4.1.

Let us write Π_i^q (or Σ_i^q) to denote the class of prenex QBFs with i alternations of quantifiers beginning with a universal (or existential, respectively).

Theorem 42. *If $A \in \Pi_i^q$ (or Σ_i^q) is valid then $\langle A \rangle$ is accepted by the Π_i (or Σ_i, respectively) provability predicate. In particular, $\langle A \rangle \in \Pi_i^p$ (or Σ_i^p, respectively).*

Proof sketch. Applying Lemma 41 above inductively, we have that $dd(\cdot \Uparrow \cdot \vdash \langle A \rangle \Uparrow) = \lfloor \frac{i}{2} \rfloor$ and that $rd(\cdot \Uparrow \cdot \vdash \langle A \rangle \Uparrow) = \lfloor \frac{i}{2} \rfloor - 1$. The result then follows by Proposition 40 and Theorem 16. The case for Σ_i^q is similar, on consideration of the sequent $\ ;\ \vdash \langle A \rangle \Downarrow$. □

Thus we conclude our partial resolution to Question 20:

Corollary 43. *Construing $[\cdot]$ as an encoding* PTPL $\to$ QCPL *under Proposition 40, we have that $[\cdot] \circ \langle \cdot \rangle$: QCPL $\to$ PTPL $\to$ QCPL preserves quantifier complexity.*

5 Conclusions, further work and remarks

In this work we have surveyed various methods for obtaining complexity bounds from proof systems, highlighting in particular how variants of focussed systems can be used to extract alternating time bounds for fragments of a logic. With this in mind, we have motivated the notion of 'over-focussing', in order to more faithfully represent the stages of nondeterministic and co-nondeterministic computation in proof search.

We considered as a case study an adequately expressive fragment PTPL of intuitionistic propositional logic and gave a sound and complete over-focussed system for it. We showed that the natural bounds extracted from this system are tight with respect to a refinement of Statman's translation from QCPL, which has various consequences for the proof theory of IPL, cf. Subsection 4.1.

We conclude this paper with various remarks, further results, and ideas for further work.

5.1 On the evaluation rule

We used a nonstandard version of the identity rule that allowed us to deterministically evaluate a formula in the presence of a complete assignment. This was to avoid any extraneous alternation complexity generated by a proof of such a sequent, but we point out that this deterministic computation can be implemented by explicit invertible non-branching rules, inspired by the Vorob'ev-Hudelmaier-Dyckhoff rules for left-implication [9], as follows:

$$\frac{\Gamma, a, b \supset c \vdash D}{\Gamma, a, (a \wedge b) \supset c \vdash D} \qquad \frac{\Gamma, A \supset a, B \supset b, (a \wedge b) \supset c \vdash D}{\Gamma, (A \wedge B) \supset c \vdash D} \qquad \frac{\Gamma, A \supset c, B \supset c \vdash D}{\Gamma, (A \vee B) \supset c \vdash D}$$

We do not develop this further as it is beyond the scope of this work, but it should be routine to give an over-focussed system for a version of *LPT* with these rules instead of *eval*.

5.2 Lazy polarisation

The fact that deterministic rules do not contribute to the overall alternation complexity of proof search is a special case of the fact that any predicate written as a Σ_i^F or Π_i^F prenex with $\Sigma_1^F \cap \Pi_1^F$ matrix remains in Σ_i^F or Π_i^F respectively. In terms of proof theory this is realised by connectives that can be polarised either positively or negatively, for example atoms and $\wedge$ in this work. This suggests that, in general, bounds on proof search could be improved by admitting special rules that polarise connectives during proof search, rather than doing so in advance. This would be particularly useful if it is more efficient for a subformula to be decomposed during an asynchronous phase in one branch and a synchronous phase in another branch, although we cannot think of an explicit example where such a phenomenon occurs.

5.3 Towards a system of invertible rules for IPL

As we already mentioned, our quantifier-complexity preserving pair of encodings QCPL $\to$ PTPL $\to$ QCPL allows us to infer the existence of a terminating 'macro-system' of invertible rules for PTPL, partially addressing a question of Dyckhoff [9]. It would be interesting to see if analysis of this macro-system could yield the necessary insights to construct a *bona fide* terminating system of invertible rules for IPL. However for this it seems likely that we would need to have a richer encoding from all of IPL, not just PTPL, to QCPL, that similarly behaves well with respect to the (refined) Statman translation. In future work we intend to investigate this possibility via an over-focussed version of the contraction-free system *G4ip* for IPL [9].

We repeat the point that just the existence of an encoding of $t_1 : \mathsf{QCPL} \to \mathsf{IPL}$, e.g. Statman's translation or the refined version in this work, already yields a terminating system of invertible rules for the image of t_1, by direct translation of such a system for QCPL, cf. Section 1.

References

[1] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. Log. Comput.*, 2(3):297–347, 1992.

[2] Nick Bezhanishvili and Dick de Jongh. *Intuitionistic logic*.

[3] Kai Brünnler and Martin Lange. Cut-free sequent systems for temporal logic. *The Journal of Logic and Algebraic Programming*, 76(2):216–225, 2008.

[4] Samuel R Buss and Rosalie Iemhoff. The depth of intuitionistic cut free proofs, 2003.

[5] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, January 1981.

[6] Kaustuv Chaudhuri, Stefan Hetzl, and Dale Miller. A multi-focused proof system isomorphic to expansion proofs. *Journal of Logic and Computation*, page exu030, 2014.

[7] Stephen Cook and Phuong Nguyen. Foundations of proof complexity: Bounded arithmetic and propositional translations. 2006.

[8] Stephen A Cook and Robert A Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(01):36–50, 1979.

[9] Roy Dyckhoff. Intuitionistic decision procedures since gentzen. In *Advances in Proof Theory*, pages 245–267. Springer, 2016.

[10] Gilda Ferreira and Paulo Oliva. On various negative translations. *arXiv preprint arXiv:1101.5442*, 2011.

[11] Stéphane Graham-Lengrand. Polarities & focussing: a journey from realisability to automated reasoning. *CoRR*, abs/1412.6781, 2014.

[12] Alexander Hertel and Alasdair Urquhart. Proof complexity of intuitionistic propositional logic. 2006.

[13] Alain Heuerding, Michael Seyfried, and Heinrich Zimmermann. Efficient loop-check for backward proof search in some non-classical propositional logics. In *International Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 210–225. Springer, 1996.

[14] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-sat. In *Computational Complexity, 1999. Proceedings. Fourteenth Annual IEEE Conference on*, pages 237–240. IEEE, 1999.

[15] Alexey P. Kopylov. Decidability of linear affine logic. *Information and Computation*, 164(1):173–198, 2001.

[16] Ranko Lazić and Sylvain Schmitz. Nonelementary complexities for branching vass, mell, and extensions. *ACM Transactions on Computational Logic (TOCL)*, 16(3):20, 2015.

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

[18] Dale Miller. Logic programming based on higher-order hereditary harrop formulas. 1988.

[19] Dale Miller. Focused LJ. *Encyclopaedia of Proof Systems*, 2015.

[20] Dale Miller and Gopalan Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, 2012.

[21] Sara Negri and Jan von Plato. *Structural proof theory*. Cambridge University Press, 2001.

[22] Vivek Nigam. Investigating the use of lemmas.

[23] Hiroakira Ono et al. Proof-theoretic methods in nonclassical logican introduction.

[24] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[25] Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970.

[26] Richard Statman. Intuitionistic propositional logic is polynomial-space complete. *Theor. Comput. Sci.*, 9:67–72, 1979.

[27] Vitezslav Svejdar. On the polynomial-space completeness of intuitionistic propositional logic. *Arch. Math. Log.*, 42(7):711–716, 2003.