Proof complexity of positive branching programs

Das, Anupam; Delkos, Avgerinos

DOI:
10.48550/arXiv.2102.06673

License:
Creative Commons: Attribution (CC BY)

Document Version
Other version

Citation for published version (Harvard):
Das, A & Delkos, A 2021 'Proof complexity of positive branching programs' arXiv, pp. 1-31.
https://doi.org/10.48550/arXiv.2102.06673

PROOF COMPLEXITY OF POSITIVE BRANCHING PROGRAMS

ANUPAM DAS* AND AVGERINOS DELKOS*

Abstract. We investigate the proof complexity of systems based on positive branching programs, i.e. non-deterministic branching programs (NBPs) where, for any 0-transition between two nodes, there is also a 1-transition. Positive NBPs compute monotone Boolean functions, just like negation-free circuits or formulas, but constitute a positive version of (non-uniform) **NL**, rather than **P** or $\mathbf{NC}^1$, respectively.

The proof complexity of NBPs was investigated in previous work by Buss, Das and Knop, using extension variables to represent the dag-structure, over a language of (non-deterministic) decision trees, yielding the system eLNDT. Our system eLNDT^+ is obtained by restricting their systems to a positive syntax, similarly to how the 'monotone sequent calculus' MLK is obtained from the usual sequent calculus LK by restricting to negation-free formulas.

Our main result is that eLNDT^+ polynomially simulates eLNDT over positive sequents. Our proof method is inspired by a similar result for MLK by Atserias, Galesi and Pudlák, that was recently improved to a bona fide polynomial simulation via works of Jeřábek and Buss, Kabanets, Kolokolova and Koucký. Along the way we formalise several properties of counting functions within eLNDT^+ by polynomial-size proofs and, as a case study, give explicit polynomial-size poofs of the propositional pigeonhole principle.

1. Introduction

Proof complexity is the study of the size of formal proofs. This pursuit is fundamentally tied to open problems in computational complexity, in particular due the Cook-Rechow theorem [10]: $co\mathbf{NP} = \mathbf{NP}$ if and only if there is a propositional proof system (suitably defined) that has polynomial-size proofs of each propositional tautology. This has led to what is known as 'Cook's program' for separating **P** and **NP**: find superpolynomial lower bounds for stronger and stronger systems until we have found a general method (see, e.g., [7, 20]).

Systems of interest in proof complexity are typically motivated by analogous results from circuit complexity and other non-uniform models of computation. For instance bounded depth systems restrict proofs to formulas with a limit on the number of alternations between $\vee$ and $\wedge$ in its formula tree, i.e. $\mathbf{AC}^0$ concepts. Indeed, Håstad's famous lower bound techniques for $\mathbf{AC}^0$ [14] have been lifted to the setting of proof complexity, yielding lower bounds for a propositional formulation of the pigeonhole principle [4] via a refined version of the switching lemma.

Monotone proof complexity is motivated by another famous lower bound result, namely Razborov's lower bounds on the size of $\neg$-free circuits [27, 28] (and similar ones for formulas [18]). In this regard, there has been much investigation into the negation-free fragment of Gentzen's sequent calculus, called MLK [2, 3, 16,

*University of Birmingham, UK

Date: February 15, 2021.

6]. [3] showed a quasipolynomial simulation of LK by MLK on ¬-free sequents by formalising an elegant counting argument using quasipolynomial-size negation-free counting formulae. This has recently been improved to a polynomial simulation by an intricate series of results [3, 16, 6], solving a question first posed in [26]. However, note the contrast with bounded depth systems: restricting negation has different effects on computational complexity and proof complexity.

In this work we address a similar question for the setting of *branching programs*. These are (presumably) more expressive than Boolean formulas, in that they are the non-uniform counterpart of log-space ($\mathbf{L}$), as opposed to $\mathbf{NC}^1$. They have recently been given a proof theoretic treatment in [5], in particular addressing proof complexity. We work within that framework, only restricting ourselves to formulas (with extension) representing *positive* branching programs.

Positive (or 'monotone') branching programs have been considered several times in the literature, e.g. [13, 17]. They are identical to Markov's 'relay-diode bipoles' from [25]. [13, 12] give a general way of making a non-deterministic model of computation 'positive'; in particular, a non-deterministic branching program is positive if, whenever there is a 0-transition from a node u to a node v, there is also a 1-transition from u to v. As in the earlier work [5] we implement such a criterion by using disjunction to model nondeterminism. As far as we are aware, there is no other work investigating the proof complexity of systems based on positive/monotone branching programs.

1.1. Contribution.

We present a formal proof calculus eLNDT^+, reasoning with formula-based representations of positive branching programs, by restricting the calculus eLNDT from [5] appropriately. We consider the 'positive closures' of well-known polynomial-size 'ordered' BPs (OBDDs) for counting functions, and show that their characteristic properties admit polynomial-size proofs in eLNDT^+.

As a case study, we show that these properties can be used to obtain polynomial-size proofs of the propositional pigeonhole principle, by adapting an approach of [2] for MLK. Our main result is that eLNDT^+ in fact polynomially simulates eLNDT over positive sequents. For this we again use representations of positive NBPs for counting and small proofs of their characteristic properties. At a high level we adapt the approach of [3], but there are several additional technicalities specific to our setting. In particular, we require bespoke treatments of negative literals in eLNDT and (iterated) substitutions of representations of positive NBPs into other positive NBPs.

1.2. A terminological convention.

Throughout this work, we shall reserve the words 'monotone', 'monotonicity' etc. for *semantic* notions, i.e. as a property of Boolean functions. For (non-uniform) models of computation such as formulas, branching programs, circuits etc., we shall say 'positive' for the associated *syntactic* constraints, e.g. negation-freeness for the case of formulas or circuits. While many works simply say 'monotone' always, in particular [12, 13], let us note that the distinction we make is employed by several other authors too, e.g. [1, 23, 22, 11].

2. Preliminaries on proof complexity and branching programs

In this section we will recall some of the content from [5]. The reader familiar with that work can safely omit this section, though they should take note of our conventions in the definition of the system eLNDT, cf. Remark 2.10.

Throughout this work we will use a countable set of *propositional variables*, written p, q etc., and *Boolean constants* 0 and 1.

An *assignment* is just a map α from propositional variables to $\{0, 1\}$. For all intents and purposes we may assume that they have finite support, e.g. nonzero only on variables occurring in a formula or proof. We extend an assignment α to the constants in the natural way, setting $\alpha(0) = 0$ and $\alpha(1) = 1$.

A *Boolean function* is just a map from (finitely supported) assignments to $\{0, 1\}$.

2.1. **Proof complexity.** In proof complexity, a (formal) *propositional proof system* is just a polynomial-time function P from Σ^* to the set of propositional tautologies, where Σ is some finite alphabet. Intuitively, the elements $\sigma \in \Sigma^*$ code proofs in the system, while P itself is a (efficient) 'proof-checking' algorithm that verifies that σ is indeed a correctly written proof, and if so returns its conclusion, i.e. the theorem it proves. If not, it just returns 1, by convention.

The significance of this definition is due to the following result from [10]:

Theorem 2.1 (Cook-Reckhow)**.** *There is a propositional proof system with polynomial-size proofs of each tautology if and only if co*$\mathbf{NP} = \mathbf{NP}$.

In practice, this 'Cook-Reckhow definition' of a propositional proof system covers all well-studied proof systems for propositional logic, under suitable codings. We shall refrain from giving any of these codings explicitly in this work, as is standard for proof complexity. However, let us point out that the systems we consider routinely admit polynomial-time proof checking in the way described above, and so indeed constitute formal propositional proof systems.

See [19, 8, 21] for more comprehensive introductions to proof complexity.

2.2. **Non-deterministic branching programs.** A (non-deterministic) *branching program* (NBP) is a (rooted) directed acyclic graph G with two distinguished *sink* nodes, 0 and 1, such that:

- G has a unique root node, i.e. a unique node with in-degree 0.
- Each non-sink node v of G is labelled by a propositional variable.
- Each edge e of G is labelled by a constant 0 or 1.

A *run* of a NBP G on an assignment α is a maximal path beginning at the root of G consistent with α. I.e., at a node labelled by a propositional variable p the run must follow an edge labelled by $\alpha(p) \in \{0, 1\}$.

We say that G *accepts* α if there is a run on α reaching the sink 1. We may extend α to a map from all NBPs to $\{0, 1\}$ by setting $\alpha(G) = 1$ just if G accepts α. In this way, each NBP *computes* a unique Boolean function $\alpha \mapsto \alpha(G)$.

A comprehensive introduction to (variants of) branching programs and their underlying theory can be found in, e.g., [30].

Example 2.2 (OBDD for 2-out-of-4 Exact)**.** The 2-out-of-4 Exact function, which returns 1 just if precisely two of its four arguments are 1, is computed by the branching program in Fig. 1. 0-edges are indicated dotted and 1-edges are indicated solid, a convention that we adopt throughout this work. Formally, each 0-leaf corresponds to the same sink.

Note that this program is deterministic: there is exactly one 0-edge and one 1-edge outgoing from each non-sink node. It is also *ordered*: all the variables appear in the same order in each path. Thus its semantics may be verified by checking that every path leading to the 1-sink has exactly two 1-edges and vice versa.

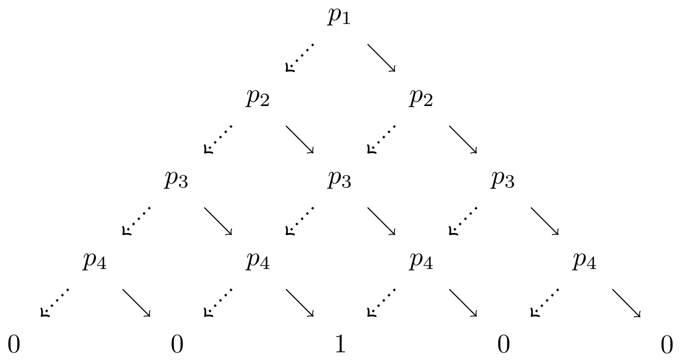

Figure 1. An OBDD computing the 2-out-of-4 Exact function. 0-edges are indicated dotted, and 1 edges are indicated solid.

2.3. **Representation of NBPs by extended formulas.** Since we will be working in formal proof systems, we shall use a natural representation of NBPs by 'formulas with extension', just like in [5]. For this, we shall make use of *extension variables* $e_0, e_1, e_2, \ldots$ in our language.

An *extended non-deterministic decision tree* formula, or *eNDT* formula, written A, B etc., is generated from propositional variables, extension variables and constants by disjunction, $\vee$, and *decisions*: for formulas A, B and p a propositional variable, (ApB) is a formula. Intuitively, (ApB) expresses "if p then B else A".

As usual, we often omit external brackets of formulas and write long disjunctions without internal brackets, under associativity. The *size* of a formula A, written $|A|$, is the number of symbols occurring in A.

Remark 2.3 (Distinguishing extension variables)**.** Note that we formally distinguish extension variables from propositional variables. This is for the same technical reasons as in [5] we must not allow extension variables to be decision variables, i.e. we forbid formulas of the form Ae_iB. If we did allow this then we would be able to express all Boolean circuits succinctly, whereas the current convention ensures that we only express NBPs.

The semantics of (non-extended) NDT formulas under an assignment will be standard. With extension variables, however, the interpretation is parametrised by a set of *extension axioms*, allowing extension variables to 'abbreviate' more complex formulas.

Definition 2.4 (Extension axioms)**.** A *set of extension axioms* $\mathcal{A}$ is a set of the form $\{e_i \leftrightarrow A_i\}_{i<n}$, where each A_i may only contain extension variables among $e_0, \ldots, e_{i-1}$.

Definition 2.5 (Semantics of eNDT formulas)**.** *Satisfaction* with respect to a set of extension axioms $\mathcal{A} = \{e_i \leftrightarrow \mathcal{A}_i\}_{i<n}$, written $\vDash_{\mathcal{A}}$, is a (infix) binary relation between assignments and formulas over $e_0, \ldots, e_{n-1}$ defined as follows:

- $\alpha \nvDash_{\mathcal{A}} 0$ and $\alpha \vDash_{\mathcal{A}} 1$.
- $\alpha \vDash_{\mathcal{A}} p$ if $\alpha(p) = 1$.
- $\alpha \vDash_{\mathcal{A}} A \vee B$ if $\alpha \vDash_{\mathcal{A}} A$ or $\alpha \vDash_{\mathcal{A}} B$.
- $\alpha \vDash_{\mathcal{A}} ApB$ if either $\alpha(p) = 0$ and $\alpha \vDash_{\mathcal{A}} A$, or $\alpha(p) = 1$ and $\alpha \vDash_{\mathcal{A}} B$.

- $\alpha \vDash_{\mathcal{A}} e_i$ if $\alpha \vDash_{\mathcal{A}} A_i$.

Example 2.6 (2-out-of-4 Exact, revisited)**.** Recall Example 2.2 and the branching program from Fig. 1. Under the semantics above, we may represent this branching program as e_{11} under the following extension axioms:

$$\begin{array}{rclcrclcrcl} & & & & & & & & e_{41} & \leftrightarrow & 0p_40 \\ e_{11} & \leftrightarrow & e_{21}p_1e_{22} & & e_{31} & \leftrightarrow & e_{41}p_3e_{42} & & e_{42} & \leftrightarrow & 0p_41 \\ e_{21} & \leftrightarrow & e_{31}p_2e_{32} & & e_{32} & \leftrightarrow & e_{42}p_3e_{43} & & e_{43} & \leftrightarrow & 1p_40 \\ e_{22} & \leftrightarrow & e_{32}p_2e_{33} & & e_{33} & \leftrightarrow & e_{43}p_3e_{44} & & e_{44} & \leftrightarrow & 0p_40 \end{array}$$

Each e_{ij} represents the j^{th} node (left to right) on the i^{th} row (top to bottom). Note that, in order to strictly comply with the subscripting condition on extension axioms, we may identify e_{ij} with e_{4i-j}.

Note that the notion $\vDash_{\mathcal{A}}$ is indeed well-defined, thanks to the subscripting conditions on sets of extension axioms: intuitively, each e_i abbreviates a formula containing only extension variables among $e_0, \dots, e_{i-1}$, and so on. More generally:

Remark 2.7 ($\mathcal{A}$-induction)**.** Given a set of extension axioms $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$ we may define a strict partial order $<_{\mathcal{A}}$ on formulas over $e_0, \dots, e_{n-1}$ by:

- $p <_{\mathcal{A}} ApB$ and $A <_{\mathcal{A}} ApB$ and $B <_{\mathcal{A}} ApB$.
- $A <_{\mathcal{A}} A \vee B$ and $B <_{\mathcal{A}} A \vee B$.
- $A_i <_{\mathcal{A}} e_i$, for each $i < n$.

Notice that $<_{\mathcal{A}}$ is indeed well-founded by the condition that each A_i must contain only extension variables among $e_0, \dots, e_{i-1}$. Thus we may carry out arguments and make definitions by induction on $<_{\mathcal{A}}$, which we shall simply refer to as '$\mathcal{A}$-induction'.

We can now see Definition 2.5 above of $\vDash_{\mathcal{A}}$ as just a definition by $\mathcal{A}$-induction. In this way, fixing some set of extension axioms $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$, each eNDT formula A over $e_0, \dots, e_{n-1}$ computes a unique Boolean function $f : \alpha \mapsto 1$ just if $\alpha \vDash_{\mathcal{A}} A$. In this case, we may say that *A computes f with respect to $\mathcal{A}$*.

Since many of our arguments will be based on $\mathcal{A}$-induction, let us make the following observation for complexity matters:

Observation 2.8 (Complexity of $\mathcal{A}$-induction)**.** *Let $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$ be a set of extension axioms and A contain only extension variables among $e_0, \dots, e_{n-1}$. Then $|\{B <_{\mathcal{A}} A\}| \leqslant |A| + \sum\limits_{i<n} |A_i|$ and, if $B <_{\mathcal{A}} A$, then $|B| \leqslant \max(|A|, |A_0|, \dots, |A_{n-1}|)$.*

2.4. **The system eLNDT.** We now recall the system for NBPs introduced in [5]. The language of the system eLNDT comprises of just the eNDT formulas. A *sequent* is an expression $\Gamma \rightarrow \Delta$, where Γ and Δ are multisets of eNDT formulas (' $\rightarrow$ ' is just a syntactic delimiter). Semantically, such a sequent is interpreted as a judgement "some formula of Γ is false or some formula of Δ is true".

Notice that the semantic interpretation of eNDT formulas we gave in Definition 2.5 means that ApB is logically equivalent to both $(\overline{p} \wedge A) \vee (p \wedge B)$ and $(\overline{p} \supset A) \wedge (p \supset B)$. It is this observation which naturally yields the following system for eNDT sequents from [5]:

Definition 2.9 (Systems LNDT and eLNDT)**.** The system LNDT is given by the rules in Fig. 2. An LNDT *derivation* of $\Gamma \rightarrow \Delta$ from *hypotheses* $\mathcal{H} = \{\Gamma_i \rightarrow \Delta_i\}_{i \in I}$

Initial sequents and cut:

$$\mathsf{0}\,\frac{}{0 \to} \qquad \mathsf{1}\,\frac{}{\to 1} \qquad \mathsf{id}\,\frac{}{p \to p} \qquad \mathsf{cut}\,\frac{\Gamma \to \Delta, A \quad \Gamma, A \to \Delta}{\Gamma \to \Delta}$$

Structural rules:

$$\mathsf{w}\text{-}l\,\frac{\Gamma \to \Delta}{\Gamma, A \to \Delta} \qquad \mathsf{w}\text{-}r\,\frac{\Gamma \to \Delta}{\Gamma \to \Delta, A} \qquad \mathsf{c}\text{-}l\,\frac{\Gamma, A, A \to \Delta}{\Gamma, A \to \Delta} \qquad \mathsf{c}\text{-}r\,\frac{\Gamma \to \Delta, A, A}{\Gamma \to \Delta, A}$$

Logical rules:

$$p\text{-}l\,\frac{\Gamma, A \to \Delta, p \quad \Gamma, p, B \to \Delta}{\Gamma, ApB \to \Delta} \qquad p\text{-}r\,\frac{\Gamma \to \Delta, A, p \quad \Gamma, p \to \Delta, B}{\Gamma \to \Delta, ApB}$$

$$\vee\text{-}l\,\frac{\Gamma, A \to \Delta \quad \Gamma, B \to \Delta}{\Gamma, A \vee B \to \Delta} \qquad \vee\text{-}r\,\frac{\Gamma \to \Delta, A, B}{\Gamma \to \Delta, A \vee B}$$

FIGURE 2. Rules for system (e)LNDT.

is defined as expected: it is a finite list of sequents, each either some $\Gamma_i \to \Delta_i$ from $\mathcal{H}$ or following from previous ones by rules of LNDT, ending with $\Gamma \to \Delta$.

An eLNDT *proof* is just an LNDT derivation from hypotheses that are a set of extension axioms $\mathcal{A} = \{e_i \leftrightarrow A_i(e_j)_{j<i}\}_{i<n}$; here we construe $A \leftrightarrow B$ as an abbreviation for the pair of sequents $A \to B$ and $B \to A$. We require that the conclusion of an eLNDT proof is free of extension variables.

The *size* of a proof or derivation P, written $|P|$, is just the number of symbols occurring in it.

Note that, despite the final condition that conclusions of eLNDT proofs are free of extension variables, we may sometimes consider intermediate 'proofs' with extension variables in the conclusions. In these cases we will always make explicit the underlying set of extension axioms.

Remark 2.10 (Differences from original eLNDT)**.** In order to ease the exposition, we have slightly adjusted the definition of eLNDT. The variations are minor and, in particular, the current presentation is polynomially equivalent to that of [5]. Nonetheless, let us survey these differences here:

- We admit constants 0 and 1 within the language. As mentioned in [5], this does not significantly affect proof size, since 0 can be encoded as $pp\overline{p}$ and 1 as $\overline{p}pp$, for an arbitrary propositional variable p.
- We do not have symbols for negative literals, to facilitate our later definition of 'positivity'. Note, however, that $\overline{p}$ is equivalent to the formula $1p0$ in our language.
- More generally, we admit decisions on only positive literals, not negative ones, for the same reason. Again, a formula $A\overline{p}B$ may be replaced by the equivalent one BpA.

As shown in [5], the system eLNDT is adequate for reasoning about non-deterministic decision trees.

Proposition 2.11 (Soundness and completeness, [5])**.** eLNDT *proves a sequent* $\Gamma \to \Delta$ *(without extension variables) if and only if* $\bigwedge \Gamma \supset \bigvee \Delta$ *is valid.*

One sanity check here is that the set of valid extension-free sequents is indeed *co***NP**-complete, and so comprises an adequate logic for proof complexity. This is shown explicitly in [5], but is also subsumed by the analogous statement for the 'positive' fragment of this language that we consider in the next section, namely Proposition 3.13.

3. Monotone functions and positive proofs

In this section we shall recall monotone Boolean functions and positive NBPs that compute them, and introduce a restriction of the system eLNDT that reasons only with such positive NBPs.

3.1. **Monotone Boolean functions and positive programs.** A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is usually called 'monotone' if, whenever $\mathbf{c} \in \{0,1\}^n$ is obtained from $\mathbf{b} \in \{0,1\}^n$ by flipping 0s to 1s, we have $f(\mathbf{b}) \leqslant f(\mathbf{c})$. Rephrasing this into our setting we have:

Definition 3.1 (Monotonicity)**.** Given assignments α, β, we write $\alpha \leqslant \beta$ if, for all propositional variables p, we have $\alpha(p) \leqslant \beta(p)$, i.e. if $\alpha(p) = 1$ then also $\beta(p) = 1$. A Boolean function f is *monotone* if $\alpha \leqslant \beta \implies f(\alpha) \leqslant f(\beta)$.

There are several known non-uniform 'positive' models for computing monotone functions, e.g. $\neg$-free circuits or formulas, monotone span programs [17], and, in our setting, positive NBPs:

Definition 3.2 (Positive NBPs, e.g. [13])**.** A NBP is *positive* if, for every 0-edge from a node u to a node v, there is also a 1-edge from u to v.

Fact 3.3. *A positive NBP computes a monotone Boolean function.*

Proof sketch. Suppose $\alpha \leqslant \beta$ and $\alpha(G) = 1$. Let $\mathbf{v} = (v_0, \dots, v_n)$ be an accepting run, where $v_n = 1$ and, for $i < n$, each v_i is labelled by some propositional variable p_i. We argue that $\mathbf{v}$ is also an accepting run of β. The critical case is when $\alpha(p_i) = 0$ but $\beta(p_i) = 1$; in which case the positivity condition on G ensures that there is nonetheless a 1-edge from v_i to v_{i+1}. □

3.2. **A digression on monotone complexity and closures.** NBPs are a non-uniform version of non-deterministic logspace (**NL**): each **NL** language is accepted by a polynomial-size family of NBPs and, conversely, the evaluation problem for NBPs is complete for **NL**. In particular, **NL** is precisely the class of languages accepted by, say, **L**-uniform families of NBPs.

Naturally, our positive NBPs correspond to a positive version of **NL** too, called **mNL** by Grigni and Sipser in [13, 12]. Those works present a comprehensive development of positive models of computation and their underlying theory. In particular there is a well-behaved notion of positive non-deterministic Turing machine based on a similar idea to that of positive NBPs: roughly speaking, whenever a transition is available when reading a 0, the same transition is available when reading a 1. It turns out that the class **mNL**, induced by this machine model restricted to logarithmic size work tapes, is equivalent to the class of languages recognised by **L**-uniform families of positive NBPs.

One natural construction that is available in the NBP setting (as opposed to, say, Boolean formulas or circuits) is the notion of a 'positive closure', which we shall work with later.

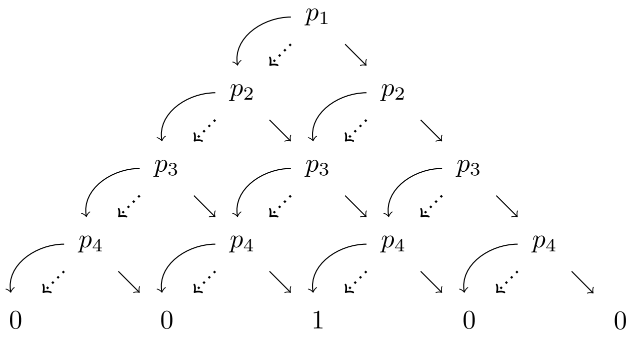

Figure 3. Positive closure of the OBDD for 2-out-of-4 Exact from Fig. 1.

Definition 3.4 (Positive closure of a NBP)**.** For a NBP G with 0-edges E_0 and 1-edges E_1, we write G^+ for the NBP with the same vertex set and 0-edges E_0 and 1-edges $E_0 \cup E_1$. I.e., G^+ is obtained from G by adding, for every 0-edge from a node u to a node v, a 1-edge from u to v (if there is not already one).

Example 3.5. The positive closure of the OBDD for 2-out-of-4 Exact from Fig. 1 is given in Fig. 3.

Note that G^+ is always a positive NBP. Thus this construction gives us a 'canonical' positive version of a NBP. In many (but not all) cases, we can precisely characterise the semantic effect of taking positive closures, thanks to the following notion:

Definition 3.6 (Monotone closure)**.** For a Boolean function f, we define its *monotone closure* $m(f)$, by $m(f)(\alpha) = 1$ just if $\exists \beta \leqslant \alpha . f(\beta) = 1$.

The point of the monotone closure of a function f is that it is the 'least' monotone function that dominates f, i.e. such that $f \leqslant m(f)$. There is also a dual notion of the 'greatest' monotone function dominated by f which is similarly related to 'positive co-NBPs', but we shall not make use of it in this work.

In certain cases, the positive closure of a NBP G computes *precisely* the monotone closure of (the function computed by) G. Call a NBP *read-once* if, on each path, each propositional variable appears at most once. We have:

Proposition 3.7 ([13])**.** *Let G be a read-once NBP computing a Boolean function f. Then G^+ computes $m(f)$.*

Proof sketch. Suppose $m(f)(\alpha) = 1$, and let $\mathbf{v}$ be an accepting run of G on some $\beta \leqslant \alpha$. Notice that $\mathbf{v}$ is also accepting for G^+ on α: at any node v_i labelled by some p on which α and β differ, i.e. $\beta(p) = 0$ and $\alpha(p) = 1$, by positivity we also have a 1-edge v_i to v_{i+1}.

Now suppose that G^+ accepts α by a run $\mathbf{v}$ of nodes labelled by $\mathbf{p}$ respectively. Define $\beta \leqslant \alpha$ by $\beta(p_i) = 1$ if there is no 0-edge from v_i to v_{i+1} in G, otherwise $\beta(p_i) = 0$. Note that β is indeed well-defined, by the read-once property, and indeed $\beta \leqslant \alpha$ by definition of G^+. □

In particular, the above result holds when G is 'ordered' (an 'OBDD'), i.e. propositional variables occur in the same relative order in each path through G. We will see an example of this with counting functions later in Section 4.

Example 3.8 (Monotone closure of Exact)**.** The monotone closure of the 2-out-of-4 Exact function is the 2-out-of-4 Threshold function, returning 1 if *at least* two of its four inputs are 1. Since the branching programs from Fig. 1 were read-once, we have by Proposition 3.7 above that the branching programs from Fig. 3 compute the 2-out-of-4 Threshold function.

Note, however, that the result does not hold for arbitrary NBPs. In fact, there is no feasible notion of 'positive closure' on NBPs that always computes the monotone closure, due to the following result:

Theorem 3.9 ([13])**.** *There are monotone functions computed by polynomial-size families of NBPs, but no polynomial-size family of positive NBPs.*

Note that this result is the analogue for the NBP model to Razborov's seminal results for the circuit model [27, 28]. This result follows by establishing a non-uniform version of '$\mathbf{mNL} \neq co\mathbf{mNL}$'; in particular there is a monotone $co\mathbf{NL}$ language (namely non-reachability in a graph) computed by no polynomial-size family of positive NBPs. The result above now follows by the Immerman-Szelepcsényi theorem that $\mathbf{NL} = co\mathbf{NL}$ [15, 29] and **NL**-completeness of NBP evaluation.

3.3. **Representations of positive branching programs.** Let us now return to our representation of NBPs by extended formulas. Recall that we implement non-determinism using disjunction, so we may duly define the corresponding notion of positivity at the level of eNDT formulas themselves:

Definition 3.10 (Positive formulas)**.** An eNDT formula is *positive* if, for each subformula of the form ApB, we have $B = A \vee C$ for some C.

A set of extension axioms $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$ is *positive* if each A_i is positive.

Positive eNDT formulas, under positive extension axioms, are just representations of positive NBPs. Notice in particular that a positive decision $Ap(A \vee B)$ is semantically equivalent to $A \vee (p \wedge B)$, which is monotone in A, p and B. Since every other symbol/connective also computes a monotone function we may thus directly obtain the analogue of Fact 3.3:

Proposition 3.11. *Suppose $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$ is a set of positive extension axioms. Each positive eNDT formula A over $e_0, \dots, e_{n-1}$ computes a monotone Boolean function with respect to $\mathcal{A}$.*

This argument proceeds by $\mathcal{A}$-induction on A and is routine. Other than the fact that all connectives are monotone, we also rely on the fact that we have no negative literals in our language.

3.4. **A system for positive branching programs.** The semantic equivalence of $Ap(A \vee B)$ and $A \vee (p \wedge B)$ motivates the following 'positive decision' rules:

$$(1) \qquad p^+\text{-}l\,\frac{\Gamma, A \rightarrow \Delta \quad \Gamma, p, B \rightarrow \Delta}{\Gamma, Ap(A \vee B) \rightarrow \Delta} \qquad p^+\text{-}r\,\frac{\Gamma \rightarrow \Delta, A, p \quad \Gamma \rightarrow \Delta, A, B}{\Gamma \rightarrow \Delta, Ap(A \vee B)}$$

Naturally, these rules satisfy the subformula property and, moreover, are derivable in eLNDT by small proofs. As expected, none of the arguments A, p or B above 'change sides' by the rules above. This is due to the fact that positive decisions are, indeed, monotone, unlike general decisions, for which the x may change sides.

Note, however, that the two rules above are not 'dual', in the logical sense, unlike general decisions. This is because positive decisions are no longer self-dual.

Definition 3.12 (System eLNDT^+)**.** The system eLNDT^+ is defined just like eLNDT, except replacing the *p*-*l* and *p*-*r* rules by the positive ones above in Equation (1). Moreover, all extension axioms and formulas occurring in a proof (in particular cut-formulas) must be positive.

As for eLNDT, we should verify that the set of valid positive sequents (without extension variables) is actually sufficiently expressive to be meaningful for proof complexity, i.e. that they are *co***NP**-complete. While this is fairly immediate for other positive systems, such as MLK, it is not so clear here so we give a self-contained argument.

Proposition 3.13. *The set of valid positive sequents (without extension variables) is co***NP***-complete.*

Proof. By the Cook-Levin theorem [9, 24], we know that the validity problem for DNFs is *co***NP**-complete, so we will show how to encode a DNF as an equi-valid positive sequent.

First, note that we may express positive terms (i.e. conjunctions of propositional variables) as positive NDT formulas by exploiting the equivalence $0x(0 \vee B) \iff 0 \vee (x \wedge B) \iff x \wedge B$. Recursively applying this equivalence we obtain:

$$\bigwedge_{i=1}^{m} p_i \iff 0p_1(0 \vee 0p_2(0 \vee (\cdots 0p_{m-1}(0 \vee p_m)\cdots))) \tag{2}$$

Let us write $\mathrm{Conj}(p_1, \dots, p_m)$ for the positive NDT formula on the right above.

Now, fix a DNF instance A over propositional variables $p_1, \dots, p_k$ and let A' be obtained by replacing each negative literal $\overline{p}_i$ by a fresh (positive) propositional variable p'_i. Write $A' = \bigvee\limits_{i=1}^{n} \bigwedge \mathbf{p}_i$, where each $\mathbf{p}_i$ is a sequence of propositional variables (among $p_1, \dots, p_k, p'_1, \dots, p'_k$). Now we have that the following positive NDT sequent is equi-valid with A, as required:

$$p_1 \vee p'_1, \dots, p_k \vee p'_k \rightarrow \mathrm{Conj}(\mathbf{p}_1), \dots, \mathrm{Conj}(\mathbf{p}_n) \qquad \square$$

Finally, our calculus eLNDT^+ is indeed adequate for reasoning about positive sequents:

Proposition 3.14 (Soundness and completeness)**.** eLNDT^+ *proves a positive sequent* $\Gamma \rightarrow \Delta$ *(without extension variables) if and only if* $\bigwedge \Gamma \supset \bigvee \Delta$.

Proof sketch. Similarly to the argument in [BDK20] for eLNDT, we may proceed by cut-free proof search, and will not make use of any extension variables. Notice that each logical rule is *invertible*, i.e. the validity of the conclusion implies the validity of each premiss. Moreover, each premiss (of a logical rule) has fewer connectives than the conclusion. Thus, bottom-up, we may simply repeatedly apply the logical steps until we reach sequents of only atomic formulae. Such a sequent is valid if and only if there is a 0 on the LHS, a 1 on the RHS, or some propositional variable p on both sides. Each of these cases may be derived from initial sequents using the weakening rules, w-l and w-r. □

Remark 3.15 (Completeness with respect to extension axioms)**.** While it is standard to only consider extended proofs over extension-free theorems, let us point out that we also have a stronger version of completeness with respect to sets of (positive) extension axioms.

Given a set of (positive) extension axioms $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$, a (positive) formula A over $e_0, \dots, e_{n-1}$ is $\mathcal{A}$-*valid* if, for every assignment α, we have $\alpha \vDash_{\mathcal{A}} A$ (cf. Definition 2.5). The same argument as in Proposition 3.14 can now be applied to show completeness for $\mathcal{A}$-valid positive sequents by proofs using the extension axioms $\mathcal{A}$. The only difference is that, when we reach a connective-free sequent (bottom-up), extension variables may also occur, not just propositional variables and constants. In this case we must use the extension axioms to unwind the extension variables and continue the proof search algorithm. Termination of this process now follows by appealing to $\mathcal{A}$-induction (cf. Remark 2.7).

3.5. **Some basic theorems.** Let us now present some basic theorems of eLNDT^+, which will all have polynomial-size proofs. These will be useful for our later arguments and, at the same time, exemplify how we will conduct proof complexity theoretic reasoning in what follows.

Let us first point out an expected property, that we can polynomially derive a general identity rule from the atomic version included in the definition of eLNDT^+. Albeit a simple observation, it has the consequence that applying substitutions of formulas for variables in proofs has only polynomial overhead in proof size.

Proposition 3.16 (General identity)**.** *Let $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$ be a set of positive extension axioms. There are polynomial-size eLNDT^+ proofs of $A \rightarrow A$, for positive formulas A containing only extension variables among $e_0, \dots, e_{n-1}$.*

Proof. We construct a (dag-like) proof of the required sequent by $\mathcal{A}$-induction. More precisely, for each such A, we construct a polynomial-size proof containing sequents $B \rightarrow B$ for each $B \leqslant_{\mathcal{A}} A$ by $\mathcal{A}$-induction on A.

- If A is a propositional variable then we are done by the rule id.
- If $A = 0$ then we have:

$$\mathsf{w}\text{-}r\;\frac{0\;\dfrac{}{0 \rightarrow}}{0 \rightarrow 0}$$

- If $A = 1$ then we have:

$$\mathsf{w}\text{-}l\;\frac{1\;\dfrac{}{\rightarrow 1}}{1 \rightarrow 1}$$

- If $A = e_i$ for some $i < n$, then we extend the proof obtained by the inductive hypothesis as follows,

$$\mathsf{2cut}\;\frac{e_i \rightarrow A_i \qquad IH\;\overline{\overline{A_i \rightarrow A_i}} \qquad A_i \rightarrow e_i}{e_i \rightarrow e_i}$$

where the sequent marked IH is obtained by the inductive hypothesis, and the other premisses are extension axioms from $\mathcal{A}$.

- If $A = B \vee C$ then we extend the proof obtained by the inductive hypothesis as follows,

$$
{\scriptstyle \vee\text{-}l}\,\dfrac{{\scriptstyle \mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle IH}\,\dfrac{}{B \longrightarrow B}}{B \longrightarrow B, C} \qquad {\scriptstyle \mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle IH}\,\dfrac{}{C \longrightarrow C}}{C \longrightarrow B, C}}{{\scriptstyle \vee\text{-}r}\,\dfrac{B \vee C \longrightarrow B, C}{B \vee C \longrightarrow B \vee C}}
$$

where sequents marked IH are obtained by the inductive hypothesis.

- If $A = Bx(B \vee C)$ then we extend the proof obtained by the inductive hypothesis as follows:

$$
{\scriptstyle p^+\text{-}l}\,\dfrac{{\scriptstyle p^+\text{-}r}\,\dfrac{{\scriptstyle \mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle IH}\,\dfrac{}{B \longrightarrow B}}{B \longrightarrow B} \qquad {\scriptstyle \mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle IH}\,\dfrac{}{B \longrightarrow B}}{B \longrightarrow B, C}}{B \longrightarrow Bp(B \vee C)} \qquad {\scriptstyle p^+\text{-}r}\,\dfrac{{\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{p \longrightarrow p}}{p, C \longrightarrow B, p} \qquad {\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle IH}\,\dfrac{}{C \longrightarrow C}}{p, C \longrightarrow B, C}}{p, C \longrightarrow Bp(B \vee C)}}{Bp(B \vee C) \longrightarrow Bp(B \vee C)}
$$

where sequents marked IH are obtained by the inductive hypothesis.

To evaluate proof size note that, at each step of the argument above, we add a constant number of lines of polynomial size in A and $\mathcal{A}$. Thus a polynomial bound follows by Observation 2.8. □

Notice, in the final step above, that we do not formally 'duplicate' the subproof for $B \longrightarrow B$ as this, recursively applied, could cause an exponential blowup. This is why the construction by $\mathcal{A}$-induction is phrased as constructing a *single* proof that contains all 'smaller' instances of identity already, with inductive steps just extending that proof. In what follows we shall be less rigorous when constructing formal proofs in this way, simply saying that we 'construct them by $\mathcal{A}$-induction'. We shall also typically leave proof complexity analysis like the one above implicit.

For our later simulations, the following 'truth conditions' for positive decisions will prove useful:

Proposition 3.17 (Truth conditions)**.** *Let $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$ be a set of positive extension axioms and let A and B be formulas over $e_0, \dots, e_{n-1}$. There are polynomial-size* eLNDT^+ *proofs of the following sequents with respect to $\mathcal{A}$:*

1. $Ap(A \vee B) \longrightarrow A, p$
2. $Ap(A \vee B) \longrightarrow A, B$
3. $A \longrightarrow Ap(A \vee B)$
4. $p, B \longrightarrow Ap(A \vee B)$

Proof. We give the proofs explicitly:

$$
(1):\ {\scriptstyle p^+\text{-}l}\,\dfrac{{\scriptstyle \mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{A \longrightarrow A}}{A \longrightarrow A, p} \qquad {\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{p \longrightarrow p}}{p, B \longrightarrow A, p}}{Ap(A \vee B) \longrightarrow A, p} \qquad (2):\ {\scriptstyle p^+\text{-}l}\,\dfrac{{\scriptstyle \mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{A \longrightarrow A}}{A \longrightarrow A, B} \qquad {\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{B \longrightarrow B}}{p, B \longrightarrow A, B}}{Ap(A \vee B) \longrightarrow A, B}
$$

$$(3):\ p^+\text{-}r\,\dfrac{\mathsf{w}\text{-}r\,\dfrac{\mathsf{id}\,\dfrac{}{A\longrightarrow A}}{A\longrightarrow A,p} \quad \mathsf{w}\text{-}r\,\dfrac{\mathsf{id}\,\dfrac{}{A\longrightarrow A}}{A\longrightarrow A,B}}{A\longrightarrow Ap(A\vee B)} \qquad (4):\ p^+\text{-}r\,\dfrac{\mathsf{w}\text{-}l,\mathsf{w}\text{-}r\,\dfrac{\mathsf{id}\,\dfrac{}{p\longrightarrow p}}{p,B\longrightarrow A,p} \quad \mathsf{w}\text{-}l,\mathsf{w}\text{-}r\,\dfrac{\mathsf{id}\,\dfrac{}{B\longrightarrow B}}{p,B\longrightarrow A,B}}{p,B\longrightarrow Ap(A\vee B)}$$

where the steps marked id are derivable by Proposition 3.16. □

Notice that, given that we have polynomial-size proofs for general identity, Proposition 3.16, the result above also just follows immediately from semantic validity of the sequents (1)-(4) and completeness of eLNDT^+, Proposition 3.14, by simply substituting the formulas A and B for appropriate constant-size instances of (1)-(4). We gave the argument explicitly to exemplify formal proofs of the system eLNDT^+. We shall, however, make use of the aforementioned observation in the remainder of this work.

Example 3.18 (A positive 'medial')**.** Branching programs enjoy elegant symmetries. For instance, in our eNDT notation, we have validity of the following pair of sequents,

$$(AqB)p(CqD)\ \leftrightarrow\ (ApC)q(BpD)$$

corresponding to a certain permutations of nodes in NBPs.

We also have a *positive* version of the law above, namely:

$$\begin{array}{rl} & (Aq(A\vee B))p((Aq(A\vee B))\vee(Cq(C\vee D))) \\ \leftrightarrow & (Ap(A\vee C))q((Ap(A\vee C))\vee(Bp(B\vee D))) \end{array} \tag{3}$$

The validity of this equivalence can be seen by noticing that each side is equivalent to $A\vee(p\wedge C)\vee(q\wedge B)\vee(p\wedge q\wedge D)$. By completeness and substitution, we thus have polynomial-size proofs of (3).

Finally, we will make use of the following consequence of the truth conditions:

Corollary 3.19. *There are polynomial size* eLNDT^+ *derivations of*

$$\Gamma, Ap(A\vee B)\longrightarrow\Delta, A'p(A'\vee B')$$

from hypotheses $\Gamma, A\longrightarrow\Delta, A'$ *and* $\Gamma, B\longrightarrow\Delta, B'$*, over any positive extension axioms including all extension variables occurring in* A *and* B*.*

Proof. We give the derivation below:

$$p^+\text{-}l\,\dfrac{\mathsf{cut}\,\dfrac{\Gamma,A\longrightarrow\Delta,A' \quad \dfrac{\text{Proposition 3.17.(3)}}{A'\longrightarrow A'p(A'\vee B')}}{\Gamma,A\longrightarrow\Delta,A'p(A'\vee B')} \quad \mathsf{cut}\,\dfrac{\Gamma,B\longrightarrow\Delta,B' \quad \dfrac{\text{Proposition 3.17.(4)}}{p,B'\longrightarrow A'p(A'\vee B')}}{\Gamma,p,B\longrightarrow\Delta,A'p(A'\vee B')}}{\Gamma,Ap(A\vee B)\longrightarrow\Delta,A'p(A'\vee B')}$$

Note that we have omitted several structural steps, namely w-l, w-r above cut-steps, to match contexts. We will typically continue to omit these, freely using 'context splitting' and 'context sharing' behaviour, under structural rules. □

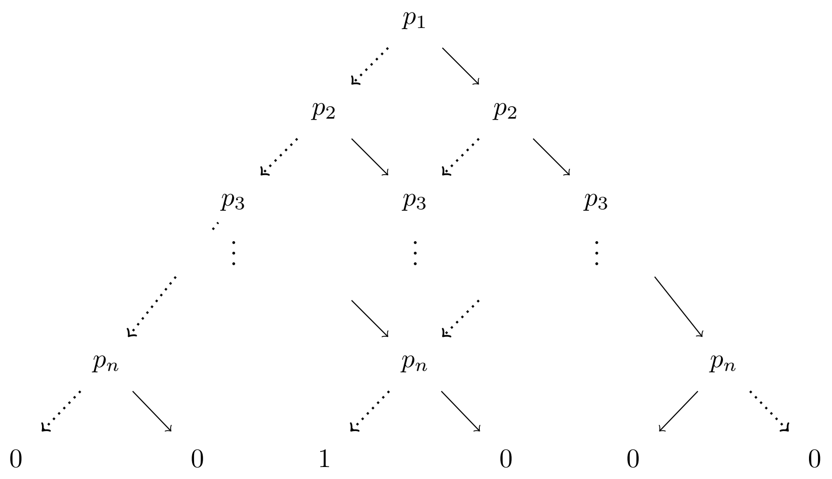

Figure 4. An 'ordered' branching program (OBDD) for $\mathrm{Ex}^n_k(p_1, \dots, p_n)$, where there are k 0s to the left of the 1, and $n - k$ to the right.

4. Programs for counting and their basic properties

Let us now consider some of the Boolean counting functions that appeared in our earlier examples more formally. The *Exact* functions $\mathrm{Ex}^n_k : \{0,1\}^n \to \{0,1\}$ are defined by:

$$\mathrm{Ex}^n_k(b_1, \dots, b_n) = 1 \qquad \Longleftrightarrow \qquad \sum_{i=1}^{n} b_i \ = \ k$$

I.e. $\mathrm{Ex}^n_k(b_1, \dots, b_n) = 1$ just if *exactly* k of $b_1, \dots, b_n$ are 1.

Taking the monotone closures of these functions (cf. Definition 3.6), we obtain the *Threshold* functions $\mathrm{Th}^n_k : \{0,1\}^n \to \{0,1\}$ by:

$$\mathrm{Th}^n_k(b_1, \dots, b_n) = 1 \qquad \Longleftrightarrow \qquad \sum_{i=1}^{n} b_i \ \geqslant \ k$$

I.e. $\mathrm{Th}^n_k(b_1, \dots, b_n) = 1$ just if *at least* k of $b_1, \dots, b_n$ are 1.

For consistency with the exposition so far, given a list $\mathbf{p} = p_0, \dots, p_{n-1}$ of propositional variables, we construe $\mathrm{Ex}^n_k(\mathbf{p})$ as a Boolean function from assignments to Booleans, writing, say, $\mathrm{Ex}^n_k(\mathbf{p})(\alpha)$ for its (Boolean) output. Similarly for $\mathrm{Th}^n_k(\mathbf{p})$.

4.1. OBDDs for Exact and their representations as eDTs.

It is well-known that counting functions like those above are computable by 'ordered' branching programs, or 'OBDDs' (see, e.g., [30]). These are deterministic branching programs where variables occur in the same relative order on each path. For instance we give an OBDD for $\mathrm{Ex}^n_k(p_1, \dots, p_n)$ in Figure 4. Thanks to determinism, we can formalise these programs as ($\vee$-free) eDT formulas as follows:

Definition 4.1 (eDTs for Exact)**.** For each list $\mathbf{p}$ of propositional variables, and each integer k, we introduce an extension variable $e^{\mathbf{p}}_k$ and write $\mathcal{E}$ for the set of all extension axioms of the form (i.e. for all choices of p, $\mathbf{p}$ and k),

$$\begin{array}{lcll} e^{\varepsilon}_0 & \leftrightarrow & 1 & \\ e^{\varepsilon}_k & \leftrightarrow & 0 & \text{if } k \neq 0 \\ e^{p\mathbf{p}}_k & \leftrightarrow & e^{\mathbf{p}}_k p e^{\mathbf{p}}_{k-1} & \end{array} \tag{4}$$

where we write ε for the empty list.

Note that, even though $\mathcal{E}$ is an infinite set, we may use it as the underlying set of extension axioms for proofs, with the understanding that only finitely many will actually ever be used in a particular proof. We will typically not explicitly compute this set, but such a consideration will be subsumed by our analysis of proof complexity.

While the extension variables above (and their axioms) do not strictly follow the subscripting conditions from Definition 2.4, we may understand them to be 'names' for the appropriate subscripting. It suffices to establish the well-foundedness of the extension axiom set in (4), which is clear by induction on the length of the superscript. We implicitly assume here, and for other well-founded extension axioms, that appropriately small subscripts are assigned to extension variables to satisfy the subscripting condition and to not contribute significantly to proof complexity.

Proposition 4.2. *Let* $\mathbf{p} = (p_1, \dots, p_n)$*.* $e^{\mathbf{p}}_k$ *computes* $\mathrm{Ex}^n_k(\mathbf{p})$*, with respect to* $\mathcal{E}$*.*

Proof. We show that $\alpha \vDash_{\mathcal{E}} e^{\mathbf{p}}_k \iff \mathrm{Ex}^n_k(\mathbf{p})(\alpha) = 1$ by induction on the length n of the list $\mathbf{p}$. If $n = 0$ then $\mathrm{Ex}^0_k(\,)$ attains the value 1 just if $k = 0$, so the result is immediate from the first two axioms of (4). For the inductive step we have:

$$\begin{array}{lcll} \alpha \vDash_{\mathcal{E}} e^{p\mathbf{p}}_k & \iff & \alpha \vDash_{\mathcal{E}} e^{\mathbf{p}}_k p e^{\mathbf{p}}_{k-1} & \text{by (4) and Definition 2.5} \\ & \iff & \begin{cases} \alpha \vDash_{\mathcal{E}} e^{\mathbf{p}}_k & \alpha(p) = 0 \\ \alpha \vDash_{\mathcal{E}} e^{\mathbf{p}}_{k-1} & \alpha(p) = 1 \end{cases} & \\ & \iff & \begin{cases} \mathrm{Ex}^n_k(\mathbf{p})(\alpha) = 1 & \alpha(p) = 0 \\ \mathrm{Ex}^n_{k-1}(\mathbf{p})(\alpha) = 1 & \alpha(p) = 1 \end{cases} & \text{by inductive hypothesis} \\ & \iff & \mathrm{Ex}^{n+1}_k(p, \mathbf{p})(\alpha) = 1 & \square \end{array}$$

4.2. **Programs for Threshold via positive closure.** Notice that, since the Exact programs we considered were OBDDs which are, in particular, read-once, the semantic characterisation of positive closure by monotone closure from Proposition 3.7 from applies. Looking back to Figure 4, the positive closures we are after (as NBPs) are given in Figure 5. Realising this directly as eNDT formulas and extension axioms we obtain the following:

Definition 4.3 (Positive eNDTs for Threshold)**.** For each list $\mathbf{p}$ of propositional variables, and each integer k, we introduce an extension variable $t^{\mathbf{p}}_k$ and write $\mathcal{T}$ for the set of all extension axioms of the form (i.e. for all choices of p, $\mathbf{p}$ and k):

$$\begin{array}{lcll} t^{\varepsilon}_0 & \leftrightarrow & 1 & \\ t^{\varepsilon}_k & \leftrightarrow & 0 & \text{if } k \neq 0 \\ t^{p\mathbf{p}}_k & \leftrightarrow & t^{\mathbf{p}}_k p (t^{\mathbf{p}}_k \vee t^{\mathbf{p}}_{k-1}) & \end{array} \tag{5}$$

Again, even though $\mathcal{T}$ is an infinite set, we shall typically write eLNDT^+ proofs with respect to this set of extension axioms, with the understanding that only

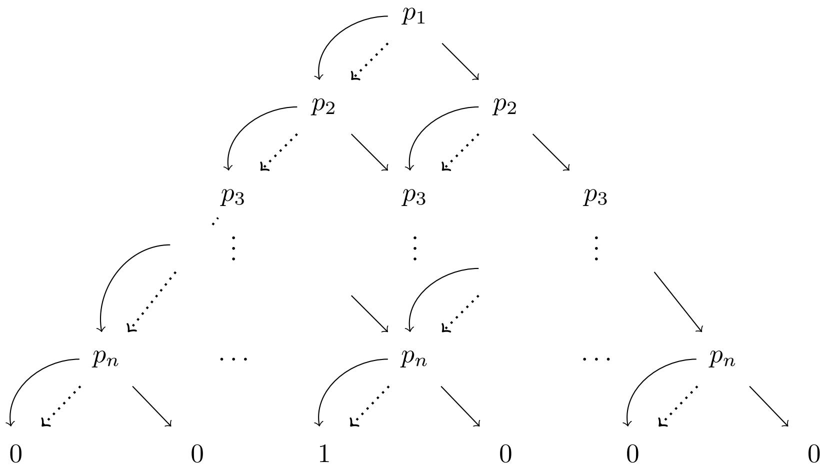

FIGURE 5. The positive closure of the OBDD for Exact from Figure 4, computing $\mathrm{Th}^n_k(p_1, \dots, p_n)$. Again, there are k 0s to the left of the 1, and $n - k$ to the right.

finitely many are ever used in any particular proof. Again, we will typically not explicitly compute this set, but such a consideration will be subsumed by our analysis of proof complexity.

Note that the extension variables $t^{\mathbf{p}}_k$ and extension axioms $\mathcal{T}$ above are just the positive closures of $e^{\mathbf{p}}_k$ and $\mathcal{E}$ earlier, within the eNDT setting. Thus, as a consequence of Proposition 3.7 we have that, for each non-negative k, $t^{\mathbf{p}}_k$ computes exactly the threshold function $\mathrm{Th}^n_k(\mathbf{p})$ with respect to $\mathcal{T}$.

Corollary 4.4. *If $k \geqslant 0$, then $t^{\mathbf{p}}_k$ computes $\mathrm{Th}^n_k(\mathbf{p})$, with respect to $\mathcal{T}$.*

Note that, for k negative, we could have alternatively set t^{ε}_k to be 1. We could have also simply set $t^{\mathbf{p}}_0$ to be 1 for arbitrary $\mathbf{p}$. Instead, we have chosen to systematically take the positive closure of the aforementioned Exact programs, to make our exposition more uniform.

4.3. **Small proofs of basic counting properties.** Our main results rely on having small proofs of characteristic properties of counting formulae, which we duly give in this section.

First we need to establish a basic monotonicity property:

Proposition 4.5 ($t^{\mathbf{p}}_k$ is decreasing in k)**.** *There are polynomial-size* eLNDT^+ *proofs of the following sequents over extension axioms $\mathcal{T}$:*

(1) $\rightarrow t^{\mathbf{p}}_0$
(2) $t^{\mathbf{p}}_{k+1} \rightarrow t^{\mathbf{p}}_k$
(3) $t^{\mathbf{p}}_k \rightarrow$ *whenever* $k > |\mathbf{p}|$

Proof. We proceed by induction on the length of $\mathbf{p}$. In the base case, when $\mathbf{p} = \varepsilon$, all three properties follow easily from $\mathcal{T}$ initial sequents, weakening and cuts.

For the inductive steps, we construct polynomial-size proofs as follows:

$$
\begin{array}{lrll}
 & & \rightarrow t_0^{\mathbf{p}} & \text{by the inductive hypothesis} \\
(1) \;: & & \rightarrow t_0^{\mathbf{p}} p(t_0^{\mathbf{p}} \vee t_{-1}^{\mathbf{p}}) & \text{by Proposition 3.17.(3)} \\
 & & \rightarrow t_0^{p\mathbf{p}} & \text{by extension axioms } \mathcal{T}
\end{array}
$$

$$
\begin{array}{lrll}
 & t_{k+1}^{p\mathbf{p}} & \rightarrow t_{k+1}^{\mathbf{p}} x(t_{k+1}^{\mathbf{p}} \vee t_k^{\mathbf{p}}) & \text{by extension axioms } \mathcal{T} \\
(2) \;: & & \rightarrow t_k^{\mathbf{p}} p(t_k^{\mathbf{p}} \vee t_{k-1}^{\mathbf{p}}) & \text{by inductive hypotheses and Corollary 3.19} \\
 & & \rightarrow t_k^{p\mathbf{p}} & \text{by extension axioms } \mathcal{T} \text{ again}
\end{array}
$$

$$
\begin{array}{lrll}
 & t_k^{p\mathbf{p}} & \rightarrow t_k^{\mathbf{p}} p(t_k^{\mathbf{p}} \vee t_{k-1}^{\mathbf{p}}) & \text{by extension axioms } \mathcal{T} \\
(3) \;: & & \rightarrow t_k^{\mathbf{p}}, t_{k-1}^{\mathbf{p}} & \text{by Proposition 3.17.(2)} \\
 & & \rightarrow t_{k-1}^{\mathbf{p}} & \text{by (2) and contraction} \\
 & & \rightarrow & \text{by inductive hypothesis} \quad \square
\end{array}
$$

The arguments above should be read by obtaining each sequent by the justification given on the right, possibly with some cuts and structural rules.

Note that Corollary 3.19 allows us to apply previously proven implications or equivalences 'deeply' within a formula. We will use such reasoning throughout this work, but shall typically omit further mentioning such uses of Corollary 3.19 to lighten the exposition.

For the complexity bound, note that only polynomially many lines occur: we only require $k+1$ instantiations of the all the sequents above, one for each choice of threshold $i \leqslant k$. This sort of complexity analysis will usually suffice for later arguments, in which case we shall suppress them unless further justification is required.

One of the key points we shall exploit in what follows is the provable *symmetry* of $t_k^{\mathbf{p}}$, in terms of the the ordering of $\mathbf{p}$. We shall establish this through a series of results, beginning by showing a form of 'case analysis' on a propositional variable occurring in a list:

Lemma 4.6 (Case analysis)**.** *There are polynomial-size* eLNDT^+ *proofs of,*

$$t_k^{\mathbf{p}q\mathbf{q}} \leftrightarrow t_k^{q\mathbf{p}\mathbf{q}}$$

over the extension axioms $\mathcal{T}$*.*

Proof. We proceed by induction on the length of $\mathbf{p}$. The base case, when $\mathbf{p}$ is empty, follows immediately by general identity, Proposition 3.16.

For the inductive step we construct polynomial-size proofs as follows:

$$
\begin{array}{lll}
 & t_k^{p\mathbf{p}q\mathbf{q}} & \\
\leftrightarrow & t_k^{\mathbf{p}q\mathbf{q}} p(t_k^{\mathbf{p}q\mathbf{q}} \vee t_{k-1}^{\mathbf{p}q\mathbf{q}}) & \text{by } \mathcal{T} \\
\leftrightarrow & t_k^{q\mathbf{p}\mathbf{q}} p(t_k^{q\mathbf{p}\mathbf{q}} \vee t_{k-1}^{q\mathbf{p}\mathbf{q}}) & \text{by } \mathit{IH} \text{ and Corollary 3.19} \\
\leftrightarrow & t_k^{\mathbf{p}\mathbf{q}} q(t_k^{\mathbf{p}\mathbf{q}} \vee t_{k-1}^{\mathbf{p}\mathbf{q}}) p(t_k^{\mathbf{p}\mathbf{q}} q(t_k^{\mathbf{p}\mathbf{q}} \vee t_{k-1}^{\mathbf{p}\mathbf{q}}) \vee t_{k-1}^{\mathbf{p}\mathbf{q}} q(t_{k-1}^{\mathbf{p}\mathbf{q}} \vee t_{k-2}^{\mathbf{p}\mathbf{q}})) & \text{by } \mathcal{T} \\
\leftrightarrow & t_k^{\mathbf{p}\mathbf{q}} p(t_k^{\mathbf{p}\mathbf{q}} \vee t_{k-1}^{\mathbf{p}\mathbf{q}}) q(t_k^{\mathbf{p}\mathbf{q}} p(t_k^{\mathbf{p}\mathbf{q}} \vee t_{k-1}^{\mathbf{p}\mathbf{q}}) \vee t_{k-1}^{\mathbf{p}\mathbf{q}} p(t_{k-1}^{\mathbf{p}\mathbf{q}} \vee t_{k-2}^{\mathbf{p}\mathbf{q}})) & \text{by Example 3.18} \\
\leftrightarrow & t_k^{p\mathbf{p}\mathbf{q}} q(t_k^{p\mathbf{p}\mathbf{q}} \vee t_{k-1}^{p\mathbf{p}\mathbf{q}}) & \text{by } \mathcal{T} \\
\leftrightarrow & t_k^{qp\mathbf{p}\mathbf{q}} & \text{by } \mathcal{T} \quad \square
\end{array}
$$

Similarly to the proof of Proposition 4.5, the above argument should be read as providing polynomial-size proofs 'in both directions', by the justifications given on the right. Note that, we restrict cedents to singletons when using ↔ in this way, to avoid ambiguity of the comma delimiter. Polynomial proof size is, again, immediate by inspection on the number of lines.

Theorem 4.7 (Symmetry)**.** *Let π be a permutation of $\mathbf{p}$. Then there are polynomial-size* eLNDT^+ *proofs over the extension axioms $\mathcal{T}$ of:*

$$t_k^{\mathbf{p}} \leftrightarrow t_k^{\pi(\mathbf{p})}$$

Proof. Write $\mathbf{p} = p_1 \cdots p_n$ and write $\pi(\mathbf{p}) = q_1 \cdots q_n$. We construct polynomial-size proofs by repeatedly applying Lemma 4.6 as follows:

$$\begin{array}{rcll} t_k^{\mathbf{p}} & \leftrightarrow & t_k^{q_n \mathbf{p}^n} & \text{by Lemma 4.6} \\ & \leftrightarrow & t_k^{q_{n-1} q_n \mathbf{p}^{n-1,n}} & \text{by Lemma 4.6} \\ & \vdots & & \\ & \leftrightarrow & t_k^{q_1 \cdots q_n} & \text{by Lemma 4.6} \\ & \leftrightarrow & t_k^{\pi(\mathbf{p})} & \text{by definition of } \mathbf{q} \end{array}$$

where $\mathbf{p}^{i,\dots,n}$ is just $\mathbf{p}$ with the elements $q_i, \dots, q_n$ removed, otherwise preserving relative order of the propositional variables. □

5. Case study: the pigeonhole principle

The *pigeonhole principle* is usually encoded in propositional logic by a family of $\neg$-free sequents of the following form:

$$\bigwedge_{i=1}^{n+1} \bigvee_{j=1}^{n} p_{ij} \to \bigvee_{j=1}^{n} \bigvee_{i=1}^{n} \bigvee_{i'=i+1}^{n+1} (p_{ij} \wedge p_{i'j})$$

Here it is useful to think of the propositional variables p_{ij} as expressing "pigeon i sits in hole j". In this way the left-hand side (LHS) above expresses that each pigeon $1, \dots, n+1$ sits in some hole $1, \dots, n$, and the right-hand side (RHS) expresses that there is some hole occupied by two (distinct) pigeons. Note that this encoding allows the mapping from pigeons to holes to be 'multi-functional', i.e. the LHS is allows for a pigeon to sit in multiple holes.

In the setting of eLNDT^+ we may not natively express conjunctions, so we adopt a slightly different encoding. Being a sequent, the outermost conjunctions on the LHS above can simply be replaced by commas; the subformulas $p_{ij} \wedge p_{i'j}$ may be encoded as $0p_{ij}(0 \vee p_{i'j})$.

Thus we shall work with the following encoding of the pigeonhole principle throughout this section:

Definition 5.1 (Pigeonhole principle)**.** PHP_n is the following positive sequent:

$$\left\{ \bigvee_{j=1}^{n} p_{ij} \right\}_{i=1}^{n+1} \to \bigvee_{j=1}^{n} \bigvee_{i=1}^{n} \bigvee_{i'=i+1}^{n+1} 0p_{ij}(0 \vee p_{i'j})$$

We write LPHP_n and RPHP_n for the LHS and RHS, respectively, of PHP_n.

5.1. Summary of proof structure.

The main result of this section is:

Theorem 5.2. *There are polynomial-size* eLNDT^+ *proofs of* PHP_n.

At a high level, we shall employ a traditional proof structure for proving PHP_n, specialising somewhat to our setting for certain intermediate results. Before surveying this, let us introduce some notation.

Notation 5.3. We fix $n \in \mathbb{N}$ throughout this section and write:

- $\mathbf{p}_i$ for the list $p_{i1}, \dots, p_{in}$, and just $\mathbf{p}$ for the list $\mathbf{p}_1, \dots, \mathbf{p}_{n+1}$.
- $\mathbf{p}^{\mathsf{T}}_j$ for the list $p_{1j}, \dots, p_{n+1j}$ and just $\mathbf{p}^{\mathsf{T}}$ for the list $\mathbf{p}^{\mathsf{T}}_1, \dots, \mathbf{p}^{\mathsf{T}}_n$.

The notation $\mathbf{p}^{\mathsf{T}}$ is suggestive since, construing $\mathbf{p}$ as an $(n+1) \times n$ matrix of propositional variables, $\mathbf{p}^{\mathsf{T}}$ is just the transpose $n \times (n+1)$ matrix.

Our approach towards proving PHP_n in eLNDT^+ (with small proofs) will be broken up into the three smaller steps, proving the following sequents respectively:

(1) $\mathsf{LPHP}_n \to t^{\mathbf{p}}_{n+1}$
(2) $t^{\mathbf{p}}_{n+1} \to t^{\mathbf{p}^{\mathsf{T}}}_{n+1}$
(3) $t^{\mathbf{p}^{\mathsf{T}}}_{n+1} \to \mathsf{RPHP}_n$

Notice that, since $\mathbf{p}^{\mathsf{T}}$ is just a permutation of $\mathbf{p}$, we already have small proofs of (2) from Theorem 4.7. In the next two subsections we shall focus on the other two implications, for which the following lemma will be quite useful:

Lemma 5.4 (Merging and splitting threshold arguments)**.** *There are polynomial-size* eLNDT^+ *proofs, over extension axioms* $\mathcal{T}$ *of the following sequents:*

(1) $t^{\mathbf{p}}_k, t^{\mathbf{q}}_l \to t^{\mathbf{pq}}_{k+l}$
(2) $t^{\mathbf{pq}}_{k+l} \to t^{\mathbf{p}}_{k+1}, t^{\mathbf{q}}_l$

Proof. We proceed by induction on the length of $\mathbf{p}$. In the base case, when $\mathbf{p} = \varepsilon$, we have two cases for (1):

- if $k = 0$ then $t^{\varepsilon}_0, t^{\mathbf{q}}_l \to t^{\mathbf{q}}_l$ follows by id and w-l.
- if $k \neq 0$ then we have an axiom $t^{\varepsilon}_k \to 0$ from $\mathcal{T}$, whence $t^{\varepsilon}_k, t^{\mathbf{q}}_l \to t^{\mathbf{q}}_{k+l}$ follows by 0, cut and weakenings.

For (2), we have polynomial-size proofs of $t^{\mathbf{q}}_{k+l} \to t^{\mathbf{q}}_l$ already from Proposition 4.5, whence we obtain $t^{\mathbf{q}}_{k+l} \to t^{\varepsilon}_{k+1}, t^{\mathbf{q}}_l$ by w-r.

For the inductive step we shall appeal to Corollary 3.19. First, for (1), by the inductive hypothesis we already have a polynomial-size proof of:

$$\begin{array}{rcl} t^{\mathbf{p}}_k, t^{\mathbf{q}}_l & \to & t^{\mathbf{pq}}_{k+l} \\ t^{\mathbf{p}}_{k-1}, t^{\mathbf{q}}_l & \to & t^{\mathbf{pq}}_{k+l-1} \end{array}$$

Thus, by Corollary 3.19 we can derive,

$$t^{\mathbf{p}}_k p(t^{\mathbf{p}}_k \vee t^{\mathbf{p}}_{k-1}), t^{\mathbf{q}}_l \to t^{\mathbf{pq}}_{k+l} p(t^{\mathbf{pq}}_{k+l} \vee t^{\mathbf{pq}}_{k+l-1})$$

whence the required sequent $t^{p\mathbf{p}}_k, t^{\mathbf{q}}_l \to t^{p\mathbf{pq}}_{k+l}$ follows by the extension axioms $\mathcal{T}$.

For (2), by the inductive hypothesis we have a polynomial-size proof of:

$$\begin{array}{rcl} t^{\mathbf{pq}}_{k+l} & \to & t^{\mathbf{p}}_{k+1}, t^{\mathbf{q}}_l \\ t^{\mathbf{pq}}_{k+l-1} & \to & t^{\mathbf{p}}_k, t^{\mathbf{q}}_l \end{array}$$

Thus, by Corollary 3.19, we can derive,

$$t^{\mathbf{pq}}_{k+l} p(t^{\mathbf{pq}}_{k+l} \vee t^{\mathbf{pq}}_{k+l-1}) \to t^{\mathbf{p}}_{k+1} p(t^{\mathbf{p}}_{k+1} \vee t^{\mathbf{p}}_k), t^{\mathbf{q}}_l$$

whence the required sequent $t^{p\mathbf{pq}}_{k+l} \to t^{p\mathbf{p}}_{k+1}, t^{\mathbf{q}}_l$ follows by the extension axioms $\mathcal{T}$. □

5.2. **From LPHP_n to $(n+1)$-threshold.** In this subsection we will give small proofs of the sequent (1) from Section 5.1.

Lemma 5.5. *Let* $\mathbf{q} = q_0, \dots, q_{k-1}$. *For all* $j < k$, *there are polynomial-size* eLNDT^+ *proofs over extension axioms* $\mathcal{T}$ *of:*

$$q_j \to t^{\mathbf{q}}_1$$

Proof. First we derive,

$$q_j \leftrightarrow t_1^{q_j} \tag{6}$$

as follows:

$$\begin{array}{rcll} q_j & \leftrightarrow & 0q_j(0 \vee 1) & \text{by Proposition 3.17, axioms and } \mathsf{cut} \\ & \leftrightarrow & t_1^\varepsilon q_j(t_1^\varepsilon \vee t_0^\varepsilon) & \text{by extension axioms } \mathcal{T} \text{ and Corollary 3.19} \\ & \leftrightarrow & t_1^{q_j} & \text{by extension axioms } \mathcal{T} \end{array}$$

By repeatedly applying Lemma 5.4.(1) we obtain polynomial-size proofs of,

$$t_0^{q_0}, \dots, t_0^{q_{j-1}}, t_1^{q_j}, t_0^{q_{j+1}}, \dots, t_0^{q_{k-1}} \to t_1^{\mathbf{q}}$$

However, we also have small proofs of $\to t_0^{q_j}$ by Proposition 4.5.(1), and so applying $k-1$ cuts we obtain a polynomial-size proof of:

$$t_1^{q_j} \to t_1^{\mathbf{q}} \tag{7}$$

The required sequent now follows by simply cutting Eq. (6) against Eq. (7). □

Proposition 5.6. *There are polynomial-size* eLNDT^+ *proofs of (1), i.e.,*

$$\mathsf{LPHP}_n \to t_{n+1}^{\mathbf{p}}$$

over extension axioms $\mathcal{T}$.

Proof. Let $i \in \{1, \dots, n+1\}$. By Lemma 5.5 above, we have small proofs of,

$$p_{ij} \to t_1^{\mathbf{p}_i}$$

for each $j = 1, \dots, n$. By applying $n-1$ $\vee$-l steps we derive:

$$\bigvee \mathbf{p}_i \to t_1^{\mathbf{p}_i} \tag{8}$$

Now, applying Lemma 5.4.(1) n times (and using cuts), we obtain small proofs of:

$$t_1^{\mathbf{p}_1}, \dots, t_1^{\mathbf{p}_{n+1}} \to t_{n+1}^{\mathbf{p}} \tag{9}$$

Finally, by instantiating Eq. (8) for each $i = 1, \dots, n+1$ and applying $n+1$ cut steps against Eq. (9) we derive the required sequent:

$$\bigvee \mathbf{p}_1, \dots, \bigvee \mathbf{p}_{n+1} \to t_{n+1}^{\mathbf{p}}$$ □

5.3. From $(n+1)$-threshold to RPHP_n.

Before deriving the final sequent (3) for our proof of PHP_n, we will need some lemmas.

Lemma 5.7. *Let* $\mathbf{q} = q_0, \dots, q_{k-1}$. *There are polynomial-size* eLNDT^+ *proofs of,*

$$q, t_1^{\mathbf{q}} \to \{0q(0 \vee q_i)\}_{i<k}$$

over extension axioms $\mathcal{T}$.

Proof. For each $i < k$, by Proposition 3.17.(4) we have a (constant-size) proof of,

$$q, q_i \to 0q(0 \vee q_i)$$

and so by cutting against appropriate instances of Eq. (6) we obtain:

$$q, t_1^{q_i} \to 0q(0 \vee q_i)$$

Instantiating the above for each $i < k$ and applying several w-r and $\vee$-l steps we obtain:

$$q, \bigvee_{i<k} t_1^{q_i} \to \{0q(0 \vee q_i)\}_{i<k} \tag{10}$$

Now, by repeatedly applying Lemma 5.4.(2) (under cuts) and $\vee$-r steps we obtain polynomial-size proofs of:

$$t_1^{\mathbf{q}} \longrightarrow \bigvee_{i<k} t_1^{q_i} \tag{11}$$

Finally, we conclude by cutting Eq. (11) above against Eq. (10). □

Lemma 5.8. *Let* $\mathbf{p} = p_0, \dots, p_{k-1}$*. There are polynomial-size* eLNDT^+ *proofs of,*

$$t_2^{\mathbf{p}} \longrightarrow \{0q_i(0 \vee q_{i'})\}_{i<i'<k}$$

over extension axioms $\mathcal{T}$*.*

Proof. We proceed by induction on the length k of $\mathbf{q}$. In the base case, when $\mathbf{q} = \varepsilon$, we have an axiom $t_2^{\varepsilon} \longrightarrow 0$ from $\mathcal{T}$, whence we conclude by a cut against the 0-axiom and w-r.

For the inductive step, we obtain by two applications of Lemma 5.4.(2) the following sequents:

$$\begin{array}{rcl} t_2^{q\mathbf{q}} & \longrightarrow & t_1^{q}, t_2^{\mathbf{q}} \\ t_2^{q\mathbf{q}} & \longrightarrow & t_2^{q}, t_1^{\mathbf{q}} \end{array}$$

Now we already have small proofs of $t_1^q \longrightarrow q$ from Eq. (6) and of $t_2^q \longrightarrow$ from Proposition 4.5.(3), and so cutting against the respective sequents above we obtain:

$$t_2^{q\mathbf{q}} \longrightarrow q, t_2^{\mathbf{q}} \tag{12}$$

$$t_2^{q\mathbf{q}} \longrightarrow t_1^{\mathbf{q}} \tag{13}$$

Finally, we combine these sequents using cuts as follows:

$$\mathsf{cut}\,\dfrac{\mathsf{2cut}\,\dfrac{\dfrac{\text{Eq. (12)}}{t_2^{q\mathbf{q}} \longrightarrow q, t_2^{\mathbf{q}}} \quad \dfrac{\text{Eq. (13)}}{t_2^{q\mathbf{q}} \longrightarrow t_1^{\mathbf{q}}} \quad \dfrac{\text{Lemma 5.7}}{q, t_1^{\mathbf{q}} \longrightarrow \{0q(0 \vee q_i)\}_{i<k}}}{t_2^{q\mathbf{q}} \longrightarrow \{0q(0 \vee q_i)\}_{i<k}} \qquad \dfrac{\mathit{IH}}{t_2^{\mathbf{q}} \longrightarrow \{0q_i(0 \vee q_{i'})\}_{i<i'<k}}}{t_2^{q\mathbf{q}} \longrightarrow \{0q_i(0 \vee q_{i'})\}_{i<i'<k}}$$

where the proof marked *IH* is obtained from the inductive hypothesis. □

Proposition 5.9. *There are polynomial-size* eLNDT^+ *proofs over* $\mathcal{T}$ *of (3), i.e. of:*

$$t_{n+1}^{\mathbf{p}^\intercal} \longrightarrow \mathsf{RPHP}_n$$

Proof. Recall that $\mathbf{p}^\intercal = \mathbf{p}_1^\intercal, \dots, \mathbf{p}_n^\intercal$, so by $n-1$ applications of Lemma 5.4.(2) (and cuts) we have small proofs of:

$$t_{n+1}^{\mathbf{p}^\intercal} \longrightarrow t_2^{\mathbf{p}_1^\intercal}, \dots, t_2^{\mathbf{p}_n^\intercal} \tag{14}$$

Now, instantiating Lemma 5.8 with $\mathbf{q} = \mathbf{p}_j^\intercal$ we also have small proofs of,

$$t_2^{\mathbf{p}_j^\intercal} \longrightarrow \{0p_{ij}(0 \vee p_{i'j})\}_{1 \leqslant i < i' \leqslant n} \tag{15}$$

for each $j = 1, \dots, n$. Finally, we may apply n cut steps on Eq. (14) against each instance of Eq. (15) (for $j = 1, \dots, n$) and apply $\vee$-r steps to obtain the required sequent. □

5.4. **Putting it all together.**

We are now ready to assemble our proofs for PHP_n.

Proof of Theorem 5.2. We simply cut together the proofs of (1), (2) and (3) that we have so far constructed:

$$\mathsf{2cut}\ \dfrac{\dfrac{\text{Proposition 5.6}}{\mathsf{LPHP}_n \rightarrow t^{\mathbf{p}}_{n+1}} \quad \dfrac{\text{Theorem 4.7}}{t^{\mathbf{p}}_{n+1} \rightarrow t^{\mathbf{p}^\intercal}_{n+1}} \quad \dfrac{\text{Proposition 5.9}}{t^{\mathbf{p}^\intercal}_{n+1} \rightarrow \mathsf{RPHP}_n}}{\mathsf{LPHP}_n \rightarrow \mathsf{RPHP}_n} \qquad \square$$

6. Positive simulation of non-positive proofs

In the previous section we showcased the capacity of the system eLNDT^+ to formalise basic counting arguments by giving polynomial-size proofs of the pigeonhole principle. In this section we go further and give a general polynomial simulation of eLNDT, over positive sequents, by adapting a method from [3].

Theorem 6.1. eLNDT^+ *polynomially simulates* eLNDT *over positive sequents.*

While the high-level structure of the argument is similar to that of [3], we must make several specialisations to the current setting due to the peculiarities of eNDT formulas and extension axioms.

6.1. **Summary of proof structure.**

Before giving the low-level details, let us survey our approach towards proving Theorem 6.1, in particular comparing it to the analogous methodology from [3]. Our strategy is divided into three main parts, which mimic the analogous proof structure from [3].

In the first part, Section 6.2, we deal with the non-positive formulas occurring in an eLNDT proof. The intuition is similar to what is done in [3] where they first reduced all negations to the variables using De Morgan duality. In our setting formulas are no longer closed under duality but, nonetheless, we are able to devise for each formula A an appropriate 'positive normal form' A^-. A^- may contain negative literals (in particular as decision variables), but all decisions themselves are positive. We duly consider an extension eLNDT^+_- of eLNDT^+ which admits negative literals $\overline{p}$ and has two extra axioms: $p, \overline{p} \rightarrow$ and $\rightarrow p, \overline{p}$. The main result of the first part is that eLNDT^+_- polynomially simulates eLNDT over positive sequents (Corollary 6.5), by essentially replacing each formula occurrence A by A^- and locally repairing the proof (Theorem 6.3).

In the second part the aim is to 'replace' negative literals in a eLNDT^+_- proof by certain threshold formulas from Definition 4.3. The is the same idea as in [3], but in our setting we must deal with certain technicalities encountered when substituting extended formulas in eLNDT^+_- and eLNDT^+. In particular, if a literal occurs as a decision variable, then we cannot directly substitute it for an extension variable (e.g. a threshold formula $t^{\mathbf{p}}_k$), since the syntax of eLNDT (and its fragments) does not allow for this. To handle this issue appropriately, we introduce in Section 6.3 a refinement of our previous threshold extension variables and axioms, defined mutually inductively with eNDT formulas themselves, that accounts for all such substitution situations (Definition 6.6).

For the remainder of the argument, in Section 6.4 we fix a eLNDT^+_- proof P of $\Gamma \rightarrow \Delta$ over extension axioms $\mathcal{A}$ and propositional variables $\mathbf{p} = p_0, \dots, p_{m-1}$. We define, for $k \geqslant 0$, systems $\mathsf{eLNDT}^+_k(P)$ that each have polynomial-size proofs P^k of

$\Gamma \rightarrow \Delta$ (Lemma 6.12). Morally speaking, this simulation is by 'substituting' thresholds for negative literals, and the consequent new axioms required in $\mathsf{eLNDT}^+_k(P)$ are parametrised by the threshold k. We point out that $\mathsf{eLNDT}^+_k(P)$ itself is tailored to the specific set of extension axioms $\mathcal{A}$ and propositional variables $\mathbf{p}$ to facilitate the choice of threshold formulas and required extension variables/axioms.

The final part, Section 6.5, essentially stitches together proofs obtained in each $\mathsf{eLNDT}^+_k(P)$ for $0 \leqslant k \leqslant m+1$. More precisely, using basic properties of threshold formulas, we show that each $\mathsf{eLNDT}^+_k(P)$ proof of a positive sequent $\Gamma \rightarrow \Delta$ can be polynomially transformed into a eLNDT^+ proof of $t^{\mathbf{p}}_k, \Gamma \rightarrow \Delta, t^{\mathbf{p}}_{k+1}$, over appropriate extension axioms (Lemma 6.14). We conclude the argument for our main result Theorem 6.1 by simply cutting these together and appealing to Proposition 4.5.

6.2. **Positive normal form of eLNDT proofs.** We shall temporarily work with a presentation of eLNDT within eLNDT^+ by allowing *negative* literals, in order to facilitate our later translations. For this reason, let us introduce, for each propositional variable p, a distinguished variable $\overline{p}$, which we shall also refer to as 'negative literals'.

The system eLNDT^+_- is defined just like eLNDT^+ but also allows negative literals $\overline{p}$ to appear in (positive) decision steps. All syntactic positivity constraints remain. Furthermore, eLNDT^+_- has two additional initial sequents:

$$\neg\text{-}l\,\frac{}{p, \overline{p} \rightarrow} \qquad\qquad \neg\text{-}r\,\frac{}{\rightarrow p, \overline{p}}$$

The system eLNDT^+_- admits essentially a 'normal form' of eLNDT proofs:

Definition 6.2. We define a (polynomial-time) translation from an eLNDT formula A to a eLNDT^+_- formula A^- as follows:

$$\begin{aligned} 0^- &:= 0 \\ 1^- &:= 1 \\ p^- &:= p \\ \overline{p}^- &:= \overline{p} \end{aligned} \qquad\qquad \begin{aligned} e_i^- &:= e_i \\ (A \vee B)^- &:= A^- \vee B^- \\ (ApB)^- &:= 0\overline{p}(0 \vee A^-) \vee 0p(0 \vee B^-) \end{aligned}$$

For a multiset of formulas $\Gamma = A_1, \dots, A_n$ we write $\Gamma^- := A_1^-, \dots, A_n^-$. For a set of extension axioms $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$, we write $\mathcal{A}^-$ for $\{e_i \leftrightarrow A_i^-\}_{i<n}$.

Theorem 6.3. *Let P be an eLNDT proof of $\Gamma \rightarrow \Delta$ over extension axioms $\mathcal{A}$. There is a eLNDT^+_- proof P^- of $\Gamma^- \rightarrow \Delta^-$ over $\mathcal{A}^-$ of size polynomial in $|P|$.*

Notice that, since $\cdot^-$ commutes with all connectives except for decisions, to prove the above result it suffices to just derive the translation of decision steps. For this it will be useful to have another 'truth' lemma:

Lemma 6.4 (Truth for $\cdot^-$-translation)**.** *Let $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$ be a set of positive extension axioms and let A and B be formulas over $e_0, \dots, e_{n-1}$. There are polynomial-size eLNDT^+_- proofs of the following sequents over $\mathcal{A}^-$:*

(1) $(ApB)^- \rightarrow A^-, p$
(2) $(ApB)^-, p \rightarrow B^-$
(3) $A^- \rightarrow (ApB)^-, p$
(4) $p, B^- \rightarrow (ApB)^-$

Proof. We give the proofs explicitly below:

$$
{\scriptstyle \vee\text{-}l}\,\dfrac{
{\scriptstyle \overline{p}^{+}\text{-}l}\,\dfrac{
{\scriptstyle \mathsf{2w}\text{-}r}\,\dfrac{{\scriptstyle 0}\,\dfrac{}{0 \to}}{0 \to A^-, p}
\quad
{\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{A^- \to A^-}}{\overline{p}, A^- \to A^-, p}
}{0\overline{p}(0 \vee A^-) \to A^-, p}
\quad
{\scriptstyle p^{+}\text{-}l}\,\dfrac{
{\scriptstyle \mathsf{2w}\text{-}r}\,\dfrac{{\scriptstyle 0}\,\dfrac{}{0 \to}}{0 \to A^-, p}
\quad
{\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{p \to p}}{p, B^- \to A^-, p}
}{0p(0 \vee B^-) \to A^-, p}
}{(ApB)^- \to A^-, p}
$$

$$
{\scriptstyle \vee\text{-}l}\,\dfrac{
{\scriptstyle \overline{p}^{+}\text{-}l}\,\dfrac{
{\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle 0}\,\dfrac{}{0 \to}}{0, p \to B^-}
\quad
{\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \neg\text{-}l}\,\dfrac{}{\overline{p}, p \to}}{\overline{p}, A^-, p \to B^-}
}{0\overline{p}(0 \vee A^-), p \to B^-}
\quad
{\scriptstyle p^{+}\text{-}l}\,\dfrac{
{\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle 0}\,\dfrac{}{0 \to}}{0, p \to B^-}
\quad
{\scriptstyle \mathsf{2w}\text{-}l}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{B^- \to B^-}}{p, B^-, p \to B^-}
}{0p(0 \vee B^-), p \to B}
}{(ApB)^-, p \to B^-}
$$

$$
{\scriptstyle \mathsf{w}\text{-}r,\vee\text{-}r}\,\dfrac{
{\scriptstyle \overline{p}^{+}\text{-}r}\,\dfrac{
{\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \neg\text{-}r}\,\dfrac{}{\to \overline{p}, p}}{A^- \to 0\overline{p}, p}
\quad
{\scriptstyle \mathsf{2w}\text{-}r}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{A^- \to A^-}}{A^- \to 0, A^-, p}
}{A^- \to 0\overline{p}(0 \vee A^-), p}
}{A^- \to (ApB)^-, p}
\qquad
{\scriptstyle \mathsf{w}\text{-}r,\vee\text{-}r}\,\dfrac{
{\scriptstyle \overline{p}^{+}\text{-}r}\,\dfrac{
{\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{p \to p}}{p, B^- \to 0, p}
\quad
{\scriptstyle \mathsf{w}\text{-}l,\mathsf{w}\text{-}r}\,\dfrac{{\scriptstyle \mathsf{id}}\,\dfrac{}{B^- \to B^-}}{p, B^- \to 0, B^-}
}{p, B^- \to 0p(0 \vee B^-)}
}{p, B^- \to (ApB)^-}
$$

□

We can now prove our polynomial-size interpretation of eLNDT within eLNDT^{+}_{-}:

Proof of Theorem 6.3. We proceed by a straightforward induction on the length of P. The critical cases are when P ends with decision steps, which we translate as follows. A left decision step,

$$
{\scriptstyle p\text{-}l}\,\dfrac{\Gamma, A \to \Delta, p \quad \Gamma, p, B \to \Delta}{\Gamma, ApB \to \Delta}
$$

is simulated by the following derivation:

$$
{\scriptstyle \mathsf{cut}}\,\dfrac{
{\scriptstyle \mathsf{cut}}\,\dfrac{\dfrac{\text{Lemma 6.4.(1)}}{\overline{(ApB)^- \to A^-, p}} \quad \Gamma, A^- \to \Delta, p}{\Gamma, (ApB)^- \to \Delta, p}
\quad
{\scriptstyle \mathsf{cut}}\,\dfrac{\dfrac{\text{Lemma 6.4.(2)}}{\overline{(ApB)^-, p \to B^-}} \quad \Gamma, p, B^- \to \Delta}{\Gamma, (ApB)^-, p \to \Delta}
}{\Gamma, (ApB)^- \to \Delta}
$$

A right decision step,

$$
{\scriptstyle p\text{-}r}\,\dfrac{\Gamma \to \Delta, A, p \quad \Gamma, p \to \Delta, B}{\Gamma \to \Delta, ApB}
$$

is simulated by the following derivation:

$$
{\scriptstyle \mathsf{cut}}\,\dfrac{
{\scriptstyle \mathsf{cut}}\,\dfrac{\Gamma \to \Delta, A^-, p \quad \dfrac{\text{Lemma 6.4.(3)}}{\overline{A^- \to (ApB)^-, p}}}{\Gamma \to \Delta, (ApB)^-, p}
\quad
{\scriptstyle \mathsf{cut}}\,\dfrac{\Gamma, p \to \Delta, B^- \quad \dfrac{\text{Lemma 6.4.(4)}}{\overline{p, B^- \to (ApB)^-}}}{\Gamma, x \to \Delta, (ApB)^-}
}{\Gamma \to \Delta, (ApB)^-}
$$

□

Finally we note that the translation above gives rise to a bona fide polynomial simulation of eLNDT by eLNDT^{+}_{-} over positive sequents:

Corollary 6.5. eLNDT^{+}_{-} *polynomially simulates* eLNDT*, over positive sequents.*

Proof. From Theorem 6.3 above, it suffices to derive $\Gamma \rightarrow \Delta$ from $\Gamma^- \rightarrow \Delta^-$. For this we shall give short proofs of,

$$A \leftrightarrow A^- \tag{16}$$

when A is positive and free of extension variables, whence $\Gamma \rightarrow \Delta$ follows from $\Gamma^- \rightarrow \Delta^-$ by several cuts. We proceed by structural induction on A, for which the critical case is when A is a decision formula. We prove the two directions separately.

First, note that we have polynomial-size proofs of the following sequents:

$$\begin{array}{llll}
(17) & Ap(A \vee B) \rightarrow A, p & & \text{by Proposition 3.17.(1)} \\
(18) & Ap(A \vee B) \rightarrow A \vee B & & \text{by Proposition 3.17.(2) and } \vee\text{-}r \\
(19) & A \rightarrow (Ap(A \vee B))^-, p & & \text{by Lemma 6.4.(3) and } IH \\
(20) & p, A \vee B \rightarrow (Ap(A \vee B))^- & & \text{by Lemma 6.4.(4) and } IH
\end{array}$$

We arrange these into a proof of the left-right direction as follows:

$$\scriptstyle p\text{-cut}\ \dfrac{\scriptstyle A\text{-cut}\ \dfrac{\text{Eq. (17)} \qquad \text{Eq. (19)}}{Ap(A \vee B) \rightarrow (Ap(A \vee B))^-, p} \qquad \scriptstyle A \vee B\text{-cut}\ \dfrac{\text{Eq. (18)} \qquad \text{Eq. (20)}}{Ap(A \vee B), p \rightarrow (Ap(A \vee B))^-}}{Ap(A \vee B) \rightarrow (Ap(A \vee B))^-}$$

Next, note that we have small proofs of the following sequents:

$$\begin{array}{llll}
(21) & A \rightarrow Ap(A \vee B) & & \text{by Proposition 3.17.(3)} \\
(22) & p, B \rightarrow Ap(A \vee B) & & \text{by Proposition 3.17.(4)} \\
(23) & (Ap(A \vee B))^- \rightarrow A, p & & \text{by Lemma 6.4.(1) and } IH \\
(24) & (Ap(A \vee B))^- \rightarrow A \vee B & & \text{by Lemma 6.4.(2) and } IH
\end{array}$$

We arrange these into a proof of the left-right direction as follows:

$$\scriptstyle p\text{-cut}\ \dfrac{\scriptstyle A\text{-cut}\ \dfrac{\text{Eq. (23)} \qquad \text{Eq. (21)}}{Ap(A \vee B)^- \rightarrow Ap(A \vee B), p} \qquad \scriptstyle A \vee B\text{-cut}\ \dfrac{\text{Eq. (24)} \qquad \text{Eq. (21)}}{Ap(A \vee B)^-, p \rightarrow Ap(A \vee B)}}{Ap(A \vee B)^- \rightarrow Ap(A \vee B)} \qquad \square$$

6.3. Generalised counting formulas.

The argument of [3] relies heavily on substitution of formulas for variables in proofs of LK. Being based on usual Boolean formulae, this is entirely unproblematic in that setting, whereas in our setting we deal with extension variables that represent NBPs via extension axioms, and so handling substitutions is much more subtle and notationally heavy.

We avoid giving a uniform treatment of this, instead specialising to counting formulas, but we must nonetheless carefully give an appropriate mutually recursive formulation of formulas and extension variables.

Definition 6.6 (Threshold decisions)**.** We introduce extension variables $[At^{\mathbf{p}}_k(A \vee B)]$ for each list $\mathbf{p}$ of propositional variables, integer k, and formulas A, B.

We extend $\mathcal{T}$ to include all extension axioms of the following form, with $\mathbf{p}, k, A, B$ ranging as just described:

$$\begin{array}{rcll}
[At^{\varepsilon}_0(A \vee B)] & \leftrightarrow & A \vee B & \\
[At^{\varepsilon}_k(A \vee B)] & \leftrightarrow & A & k \neq 0 \\
[At^{p\mathbf{p}}_k(A \vee B)] & \leftrightarrow & [At^{\mathbf{p}}_k(A \vee B)]p([At^{\mathbf{p}}_k(A \vee B)] \vee [At^{\mathbf{p}}_{k-1}(A \vee B)]) &
\end{array} \tag{25}$$

Note that, despite the presentation, $[At^{\mathbf{p}}_k(A \vee B)]$ is, formally speaking, a single extension variable, not a decision on the extension variable $t^{\mathbf{p}}_k$ which, recall, we do not permit. This is why we use the square brackets to distinguish it from other formulas, though we shall justify this notation shortly.

One should view the extension variables above and the notion of an eLNDT^+ formula as being *mutually* defined, so as to avoid foundational issues. For instance, we allow an extension variable $[Ct^{\mathbf{p}}_k(C \vee D)]$, where C or D may themselves contain extension variables of the form $[At^{\mathbf{q}}_k(A \vee B)]$. By building up formulas and extension variables by mutual induction we ensure that such constructions are well-founded. Let us briefly make this formal in the following remark:

Remark 6.7 (Well-foundedness of $\mathcal{T}$)**.** We may define the extension variables $[At^{\mathbf{p}}_k(A \vee B)]$ and eLNDT^+ formulas 'in stages' as follows:

- Write Φ_0 for the set of all eLNDT^+ formulas over some base set of extension variables $E_0 = \{e_{00}, e_{01}, \dots\}$.
- Write T_n for the set of all extension variables $t^{\mathbf{p}}_k$ and of the form $[At^{\mathbf{p}}_k(A \vee B)]$ with $A, B \in \Phi_n$.
- Write Φ_{n+1} for the set of all formulas built from propositional variables, disjunctions, positive decisions and extension variables from T_n, E_n and a fresh set of new extension variables $E_{n+1} = \{e_{(n+1)0}, e_{(n+1)1}, \dots\}$.

Within each T_n we (partially well-)order extension variables by the length of the superscript $\mathbf{p}$, and within each E_n we (well-)order the e_{ni}s by the subscript i. Finally, we set $E_n < T_n < E_{n+1}$ (i.e. if $e \in E_n$, $t \in T_n$ and $e' \in E_{n+1}$ then $e < t < e'$). In this way the extension axioms from Eq. (25) (and Eq. (5)) indeed satisfy the required well-foundedness criterion. We may also admit further extension axioms allowing elements of E_n to abbreviate formulas in Φ_n (satisfying the indexing condition internal to E_n), preserving well-foundedness. We shall indeed do this later.

Note, however, that the required order type on indices of these extension variables, a priori, exceeds the ordinal ω. This causes no issue for us since, in any finite proof, we will only use finitely many extension variables, and so may construe each index as a (relatively small) natural number while preserving the aforementioned order. We shall gloss over this issue in what follows.

As previously mentioned, our notation $[At^{\mathbf{p}}_k(A \vee B)]$ is designed to be suggestive, justified by the following counterpart of the truth conditions from Proposition 3.17:

Proposition 6.8 (Truth conditions for threshold decisions)**.** *There are polynomial size* eLNDT^+ *proofs over* $\mathcal{T}$ *of:*

(1) $[At^{\mathbf{p}}_k(A \vee B)] \rightarrow A, t^{\mathbf{p}}_k$
(2) $[At^{\mathbf{p}}_k(A \vee B)] \rightarrow A, B$
(3) $A \rightarrow [At^{\mathbf{p}}_k(A \vee B)]$
(4) $t^{\mathbf{p}}_k, B \rightarrow [At^{\mathbf{p}}_k(A \vee B)]$.

Proof. We proceed by induction on the length of $\mathbf{p}$. For the base case when $\mathbf{p} = \varepsilon$, we have the following proofs:

$$\mathsf{w}\text{-}l,\mathsf{w}\text{-}r\ \dfrac{\mathcal{T},\mathsf{cut}\ \dfrac{1\ \dfrac{}{\longrightarrow 1}}{\longrightarrow t_0^\varepsilon}}{[At_0^\varepsilon(A\vee B)] \longrightarrow A, t_0^\varepsilon} \qquad \mathsf{id},\vee,\mathsf{cut}\ \dfrac{\mathcal{T}\ \dfrac{}{[At_0^\varepsilon(A\vee B)] \longrightarrow A \vee B}}{[At_0^\varepsilon(A\vee B)] \longrightarrow A, B}$$

$$\mathcal{T},\mathsf{cut}\ \dfrac{\mathsf{w}\text{-}r,\vee\text{-}r\ \dfrac{\mathsf{id}\ \dfrac{}{A \longrightarrow A}}{A \longrightarrow A \vee B}}{A \longrightarrow [At_0^\varepsilon(A\vee B)]} \qquad \mathcal{T},\mathsf{cut}\ \dfrac{\mathsf{w}\text{-}l,\mathsf{w}\text{-}r,\vee\text{-}r\ \dfrac{\mathsf{id}\ \dfrac{}{B \longrightarrow B}}{t_0^\varepsilon, B \longrightarrow A \vee B}}{t_0^\varepsilon, B \longrightarrow [At_0^\varepsilon(A\vee B)]}$$

For the inductive step for (1) we derive the following sequents:

$$\begin{array}{rcll}
[At_k^{\mathbf{p}}(A\vee B)] & \longrightarrow & A, t_k^{\mathbf{p}} & \text{by } \mathit{IH} \\
[At_{k-1}^{\mathbf{p}}(A\vee B)] & \longrightarrow & A, t_{k-1}^{\mathbf{p}} & \text{by } \mathit{IH} \\
[At_k^{\mathbf{p}}(A\vee B)]p([At_k^{\mathbf{p}}(A\vee B)] \vee [At_{k-1}^{\mathbf{p}}(A\vee B)]) & \longrightarrow & A, t_k^{\mathbf{p}}p(t_k^{\mathbf{p}} \vee t_{k-1}^{\mathbf{p}}) & \text{by Corollary 3.19} \\
[At_k^{p\mathbf{p}}(A\vee B)] & \longrightarrow & A, t_k^{p\mathbf{p}} & \text{by } \mathcal{T} \text{ and } 2\mathsf{cut}
\end{array}$$

For the inductive step for (2) we derive the following sequents:

$$\begin{array}{rcll}
[At_k^{\mathbf{p}}(A\vee B)] & \longrightarrow & A, B & \text{by } \mathit{IH} \\
p, [At_{k-1}^{\mathbf{p}}(A\vee B)] & \longrightarrow & A, B & \text{by } \mathit{IH} \text{ and } \mathsf{w}\text{-}l \\
[At_k^{\mathbf{p}}(A\vee B)]p([At_k^{\mathbf{p}}(A\vee B)] \vee [At_{k-1}^{\mathbf{p}}(A\vee B)]) & \longrightarrow & A, B & \text{by } p^+\text{-}l \\
[At_k^{p\mathbf{p}}(A\vee B)] & \longrightarrow & A, B & \text{by } \mathcal{T} \text{ and } \mathsf{cut}
\end{array}$$

For the inductive step for (3) we derive the following sequents:

$$\begin{array}{rcll}
A & \longrightarrow & [At_k^{\mathbf{p}}(A\vee B)], p & \text{by } \mathit{IH} \text{ and } \mathsf{w}\text{-}r \\
A & \longrightarrow & [At_k^{\mathbf{p}}(A\vee B)], [At_{k-1}^{\mathbf{p}}(A\vee B)] & \text{by } \mathit{IH} \text{ and } \mathsf{w}\text{-}r \\
A & \longrightarrow & [At_k^{\mathbf{p}}(A\vee B)]p([At_k^{\mathbf{p}}(A\vee B)] \vee [At_{k-1}^{\mathbf{p}}(A\vee B)]) & \text{by } p^+\text{-}r \\
A & \longrightarrow & [At_k^{p\mathbf{p}}(A\vee B)] & \text{by } \mathcal{T} \text{ and } \mathsf{cut}
\end{array}$$

For the inductive step for (4) we derive the following sequents:

$$\begin{array}{rcll}
t_k^{\mathbf{p}}, B & \longrightarrow & [At_k^{\mathbf{p}}(A\vee B)] & \text{by } \mathit{IH} \\
t_{k-1}^{\mathbf{p}}, B & \longrightarrow & [At_{k-1}^{\mathbf{p}}(A\vee B)] & \text{by } \mathit{IH} \\
t_k^{\mathbf{p}}p(t_k^{\mathbf{p}} \vee t_{k-1}^{\mathbf{p}}), B & \longrightarrow & [At_k^{\mathbf{p}}(A\vee B)]p([At_k^{\mathbf{p}}(A\vee B)] \vee [At_{k-1}^{\mathbf{p}}(A\vee B)]) & \text{by Corollary 3.19} \\
t_k^{p\mathbf{p}}, B & \longrightarrow & [At_k^{p\mathbf{p}}(A\vee B)] & \text{by } \mathcal{T} \text{ and } 2\mathsf{cut} \quad \square
\end{array}$$

6.4. **'Substituting' thresholds for negative literals.** For the remainder of this section, let us work with a fixed eLNDT^+_- proof P, over extension axioms $\mathcal{A} = \{e_i \leftrightarrow A_i(e_0, \dots, e_{i-1})\}_{i<n}$, of a positive sequent $\Gamma \longrightarrow \Delta$ containing propositional variables among $\mathbf{p} = p_0, \dots, p_{m-1}$ and extension variables among $\mathbf{e} = e_0, \dots, e_{n-1}$.

Recall that, since we are eventually trying to give a polynomial simulation of eLNDT by eLNDT^+ over positive sequents, our consideration of eLNDT^+_- here suffices, by Corollary 6.5. We shall also work with the extension axioms $\mathcal{T}$ from the previous subsection, and will soon explain its interaction with $\mathcal{A}$ from P.

Throughout this section, we shall write $\mathbf{p}_i$ for $p_0, \dots, p_{i-1}, p_{i+1}, \dots, p_{m-1}$, i.e. $\mathbf{p}$ with the variable p_i removed. We shall define yet another intermediary system $\mathsf{eLNDT}^+_k(P)$, or rather a family of such systems, one for each $k \geqslant 0$. Before that, we need to introduce the following translation of formulas.

Definition 6.9 ('Substituting' thresholds)**.** We define a (polynomial-time) translation from an eLNDT^+_- formula A (over $\mathbf{p}$, $\overline{\mathbf{p}}$ and $\mathbf{e}$) to a eLNDT^+ formula A^k (over $\mathbf{p}$, some extension variables $\mathbf{e}^k$ and extension variables from $\mathcal{T}$) as follows:

- $0^k := 0$
- $1^k := 1$
- $p_i^k := p_i$
- $\overline{p}_i^k := t_k^{\mathbf{p}_i}$
- e_i^k is a fresh extension variable.
- $(A \vee B)^k := A^k \vee B^k$
- $(Ap_i(A \vee B))^k := A^k p_i(A^k \vee B^k)$
- $(A\overline{p}_i(A \vee B))^k := \left[{\scriptstyle A^k}t_k^{\mathbf{p}[0/p_i]}{\scriptstyle (A^k \vee B^k)}\right]$

We also define $\mathcal{A}^k := \{e_i^k \leftrightarrow A_i^k(e_0^k, \dots, e_{i-1}^k)\}_{i<n}$, and if $\Gamma = B_1, \dots, B_l$ we write Γ^k for $B_1^k, \dots, B_l^k$.

In what follows, we shall work with the set of extension axioms $\mathcal{T} \cup \mathcal{A}^k$, so let us take a moment to justify that this set of extension axioms is indeed well-founded.

Remark 6.10 (Well-foundedness, again)**.** Following on from Remark 6.7, well-foundedness of $\mathcal{T} \cup \mathcal{A}^k$ follows from a suitable indexing of the extension variables therein. To this end we assign 'stages' to each formula A^k by $\mathcal{A}$-induction on A, using the notation of Remark 6.7:

- $p_i^k, 0^k, 1^k \in \Phi_0$.
- $\overline{p}_i^k \in T_0$.
- $e_i^k \in E_m \subseteq \Phi_m$ if $A_i^k(e_0^k, \dots, e_{i-1}^k) \in \Phi_m$, with index i (i.e., e_i^k is e_{mi}).
- $(A \vee B)^k \in \Phi_m$ if $A^k, B^k \in \Phi_m$.
- $(Ap_i(A \vee B))^k \in \Phi_m$ if $A^k, B^k \in \Phi_m$.
- $(A\overline{p}_i(A \vee B))^k \in T_m \subseteq \Phi_{m+1}$ if $A^k, B^k \in \Phi_m$.

Note in particular that stages can grow even for formulas free of e_i^k since we may have nested decisions on negated variables $\overline{p}_i$.

Once again, in terms of proof complexity, we will gloss over this subtlety and simply count the number of propositional and extension variable occurrences in a proof, assuming that each variable can be equipped with a 'small' index.

We are now ready to define our intermediary systems.

Definition 6.11. The system $\mathsf{eLNDT}^+_k(P)$ is defined just like eLNDT^+, but includes additional initial sequents,

$$t\text{-}l\,\frac{}{p_i, t_k^{\mathbf{p}_i} \to} \qquad\qquad t\text{-}r\,\frac{}{\to p_i, t_k^{\mathbf{p}_i}} \tag{26}$$

and may only use the extension axioms $\mathcal{T} \cup \mathcal{A}^k$.

Lemma 6.12. *There is an* $\mathsf{eLNDT}^+_k(P)$ *proof of* $\Gamma \to \Delta$ *of size polynomial in* $|P|$.

Proof. We construct the required proof P^k by replacing every formula occurrence A in P by A^k. Note that all structural steps, identities, cuts remain correct. An extension axiom for e_i from $\mathcal{A}$ is just translated to the corresponding extension axiom for e_i^k from $\mathcal{A}^k$, and the initial sequents $\neg$-l and $\neg$-r from eLNDT^+_- are translated to the two new initial sequents t-l and t-r, respectively, from Eq. (26) above. It remains to simulate the logical steps.

The simulation of $\vee$ steps is immediate, since the $\cdot^k$-translation commutes with $\vee$. Similarly for positive decisions on p_i. A left positive decision step on $\overline{p}_i$,

$$\overline{p}_i^+\text{-}l\ \dfrac{\Gamma, A \longrightarrow \Delta \quad \Gamma, \overline{p}_i, B \longrightarrow \Delta}{\Gamma, A\overline{p}_i(A \vee B) \longrightarrow \Delta}$$

is translated to the following derivation:

$$\mathsf{cut}\ \dfrac{\mathsf{cut}\ \dfrac{\dfrac{\text{Proposition 6.8.(1)}}{\overline{(A\overline{p}_i(A \vee B))^k \longrightarrow A^k, \overline{p}_i^k}} \quad \mathsf{cut}\ \dfrac{\dfrac{\text{Proposition 6.8.(2)}}{\overline{(A\overline{p}_i(A \vee B))^k \longrightarrow A^k, B^k}} \quad \Gamma^k, \overline{p}_i^k, B^k \longrightarrow \Delta}{\Gamma^k, (A\overline{p}_i(A \vee B))^k, \overline{p}_i^k \longrightarrow \Delta^k, A^k}}{\Gamma^k, (A\overline{p}_i(A \vee B))^k \longrightarrow \Delta^k, A^k} \qquad \Gamma^k, A^k \longrightarrow \Delta^k}{\Gamma^k, (A\overline{p}_i(A \vee B))^k \longrightarrow \Delta^k}$$

A right positive decision rule on $\overline{p}_i$,

$$\overline{p}_i^+\text{-}r\ \dfrac{\Gamma \longrightarrow \Delta, A, \overline{p}_i \quad \Gamma \longrightarrow \Delta, A, B}{\Gamma \longrightarrow \Delta, A\overline{p}_i(A \vee B)}$$

is translated to the following derivation:

$$\mathsf{cut}\ \dfrac{\mathsf{cut}\ \dfrac{\Gamma^k \longrightarrow \Delta^k, A^k, \overline{p}_i^k \quad \mathsf{cut}\ \dfrac{\Gamma^k \longrightarrow \Delta^k, A^k, B^k \quad \dfrac{\text{Proposition 6.8.(4)}}{\overline{\overline{p}_i^k, B^k \longrightarrow A\overline{p}_i(A \vee B)^k}}}{\Gamma^k, \overline{p}_i^k \longrightarrow \Delta^k, A^k, A\overline{p}_i(A \vee B)^k}}{\Gamma^k \longrightarrow \Delta^k, A^k, A\overline{p}_i(A \vee B)^k} \qquad \dfrac{\text{Proposition 6.8.(3)}}{\overline{A^k \longrightarrow A\overline{p}_i(A \vee B)^k}}}{\Gamma^k \longrightarrow \Delta^k, A\overline{p}_i(A \vee B)^k}$$

□

6.5. **Putting it all together.** We are now ready to assemble the proof our main simulation result, Theorem 6.1. Recall that we are still working with the fixed eLNDT^+_- proof P of a positive sequent $\Gamma \longrightarrow \Delta$ from Section 6.4, over extension axioms $\mathcal{A} = \{e_i \leftrightarrow A_i\}_{i<n}$ and propositional variables $\mathbf{p} = p_0, \dots, p_{m-1}$. We continue to write $\mathbf{p}_i$ for $p_0, \dots, p_{i-1}, p_{i+1}, \dots, p_{m-1}$, i.e. $\mathbf{p}$ with p_i removed.

Proposition 6.13. *For $k \geqslant 0$, there are polynomial size eLNDT^+ proofs of,*

$$p_i, t_k^{\mathbf{p}_i} \longrightarrow t_{k+1}^{\mathbf{p}} \tag{27}$$

$$t_k^{\mathbf{p}} \longrightarrow p_i, t_k^{\mathbf{p}_i} \tag{28}$$

over extension axioms $\mathcal{T}$.

Proof. We derive Eq. (27) as follows:

$$\begin{array}{rcll} t_1^{p_i}, t_{k+1}^{\mathbf{p}_i} & \longrightarrow & t_{k+1}^{p_i\mathbf{p}_i} & \text{by Lemma 5.4.(1)} \\ p_i, t_k^{\mathbf{p}_i} & \longrightarrow & t_{k+1}^{p_i\mathbf{p}_i} & \text{by Eq. (6) and } \mathsf{cut} \\ & \longrightarrow & t_{k+1}^{\mathbf{p}} & \text{by Lemma 4.6 and } \mathsf{cut} \end{array}$$

We derive Eq. (28) as follows:

$$\begin{array}{rcll} t_k^{\mathbf{p}} & \longrightarrow & t_k^{p_i\mathbf{p}_i} & \text{by Lemma 4.6} \\ & \longrightarrow & t_1^{p_i}, t_k^{\mathbf{p}_i} & \text{by Lemma 5.4.(2) and } \mathsf{cut} \\ & \longrightarrow & p_i, t_k^{\mathbf{p}_i} & \text{by Eq. (6) and } \mathsf{cut} \quad \square \end{array}$$

Lemma 6.14. *For $k \geqslant 0$, there are polynomial size eLNDT^+ proofs of,*

$$t_k^{\mathbf{p}}, \Gamma \longrightarrow \Delta, t_{k+1}^{\mathbf{p}}$$

over extension axioms $\mathcal{T} \cup \mathcal{A}^k$.

Proof. By Lemma 6.12, we already have a polynomial-size proof $\mathsf{eLNDT}^+_k(P)$ proof P^k of $\Gamma \to \Delta$. By definition of $\mathsf{eLNDT}^+_k(P)$, we construe P^k as an eLNDT^+ derivation of $\Gamma \to \Delta$ over extension axioms $\mathcal{T} \cup \mathcal{A}^k$ from hypotheses:

$$p_i, t^{\mathbf{P}_i}_k \to \tag{29}$$

$$\to p_i, t^{\mathbf{P}_i}_k \tag{30}$$

We obtain the required proof by adding $t^{\mathbf{P}}_k$ to the LHS of each sequent and $t^{\mathbf{P}}_{k+1}$ to the RHS of each sequent in P^k. Each local inference step remains correct, except that some weakenings may be required to repair initial steps. Finally we replace occurrences of the hypotheses Eq. (29) and Eq. (30) above by the proofs of Eq. (27) and Eq. (28) respectively from Proposition 6.13. □

We are now ready to prove our main result, that eLNDT^+ polynomially simulates eLNDT over positive sequents:

Proof of Theorem 6.1 . By Corollary 6.5, without loss of generality let P be a eLNDT^+_- proof of a positive sequent $\Gamma \to \Delta$ over extension axioms $\mathcal{A}$. By Lemma 6.14 we construct, for each $k \leqslant m+1$, polynomial-size proofs of $t^{\mathbf{P}}_k, \Gamma \to \Delta, t^{\mathbf{P}}_{k+1}$, over $\mathcal{T} \cup \mathcal{A}^k$, and we simply 'cut' them all together as follows:

$$
{\scriptstyle (m+2)\mathsf{cut}}\ \dfrac{\dfrac{\text{Proposition 4.5.(1)}}{\to t^{\mathbf{P}}_0} \quad \dfrac{\text{Lemma 6.14}}{t^{\mathbf{P}}_0, \Gamma \to \Delta, t^{\mathbf{P}}_1} \quad \cdots \quad \dfrac{\text{Lemma 6.14}}{t^{\mathbf{P}}_m, \Gamma \to \Delta, t^{\mathbf{P}}_{m+1}} \quad \dfrac{\text{Proposition 4.5.(3)}}{t^{\mathbf{P}}_{m+1} \to}}{\Gamma \to \Delta}
$$

The resulting proof is an eLNDT^+ proof of the required sequent, over extension axioms $\mathcal{T} \cup \mathcal{A}^0 \cup \mathcal{A}^1 \cup \cdots \cup \mathcal{A}^{m+1}$. Note that this set of extension axioms is indeed well-founded, since each $\mathcal{A}^k$ is only used in distinct subproofs. □

References

[1] Miklos Ajtai and Yuri Gurevich. Monotone versus positive. *J. ACM*, 34(4):1004–1015, October 1987.

[2] Albert Atserias, Nicola Galesi, and Ricard Gavaldà. Monotone proofs of the pigeon hole principle. *Math. Log. Q.*, 47(4):461–474, 2001.

[3] Albert Atserias, Nicola Galesi, and Pavel Pudlák. Monotone simulations of non-monotone proofs+. *J. Comput. Syst. Sci.*, 65(4):626–638, 2002.

[4] Paul Beame, Russell Impagliazzo, Jan Krajícek, Toniann Pitassi, Pavel Pudlák, and Alan R. Woods. Exponential lower bounds for the pigeonhole principle. In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 200–220. ACM, 1992.

[5] Sam Buss, Anupam Das, and Alexander Knop. Proof complexity of systems of (non-deterministic) decision trees and branching programs. In *CSL*, volume 152 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[6] Sam Buss, Valentine Kabanets, Antonina Kolokolova, and Michal Koucký. Expander construction in VNC1. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 31:1–31:26, 2017.

[7] Samuel R. Buss. Towards NP-P via proof complexity and search. *Ann. Pure Appl. Log.*, 163(7):906–917, 2012.

[8] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity.* Cambridge University Press, 2010.

[9] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.

[10] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979.

[11] Anupam Das and Isabel Oitavem. A Recursion-Theoretic Characterisation of the Positive Polynomial-Time Functions. In Dan Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, volume 119 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[12] Michelangelo Grigni. *Structure in monotone complexity.* PhD thesis, Citeseer, 1991.

[13] Michelangelo Grigni and Michael Sipser. Monotone complexity, 1990.

[14] Johan Håstad, Ingo Wegener, Norbert Wurm, and Sang-Zin Yi. Optimal depth, very small size circuits for symmetric functions in ac^0. *Inf. Comput.*, 108(2):200–211, 1994.

[15] N. Immerman. Nondeterministic space is closed under complementation. In *[1988] Proceedings. Structure in Complexity Theory Third Annual Conference*, pages 112–115, 1988.

[16] Emil Jerábek. A sorting network in bounded arithmetic. *Ann. Pure Appl. Logic*, 162(4):341–355, 2011.

[17] M. Karchmer and A. Wigderson. On span programs. In *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pages 102–111, 1993.

[18] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 539–550, New York, NY, USA, 1988. Association for Computing Machinery.

[19] Jan Krajícek. *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of mathematics and its applications.* Cambridge University Press, 1995.

[20] Jan Krajícek. The cook-reckhow definition. *CoRR*, abs/1909.03691, 2019.

[21] Jan Krajíček. *Proof Complexity.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2019.

[22] Clemens Lautemann, Thomas Schwentick, and Iain Stewart. Positive versions of polynomial time. *Inf. Comput.*, 147:145–170, 01 1998.

[23] Clemens Lautemann, Thomas Schwentick, and Iain A. Stewart. On positive p. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, CCC '96, page 162, USA, 1996. IEEE Computer Society.

[24] L. Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9:115–116, 1973.

[25] A. A. Markov. Minimal relay-diode bipoles for monotonic symmetric functions. *Problemy Kibernetiki*, 8:117–121, 1962.

[26] Pavel Pudlák and Samuel R. Buss. How to lie without being (easily) convicted and the length of proofs in propositional calculus. In *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, pages 151–162, 1994.

[27] Alexander A Razborov. Lower bounds for the monotone complexity of some boolean functions. In *Soviet Math. Dokl.*, volume 31, pages 354–357, 1985.

[28] Alexander A Razborov. Lower bounds on monotone complexity of the logical permanent. *Mathematical Notes of the Academy of Sciences of the USSR*, 37(6):485–493, 1985.

[29] Róbert Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Inf.*, 26(3):279–284, 1988.

[30] Ingo Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM, 2000.