Logical Methods in Computer Science
Volume 22, Issue 2, 2026, pp. 18:1–18:34
https://lmcs.episciences.org/

Submitted Aug. 29, 2025
Published May 20, 2026

PROVER-ADVERSARY GAMES FOR SYSTEMS OVER (NON-DETERMINISTIC) BRANCHING PROGRAMS

ANUPAM DAS AND AVGERINOS DELKOS

University of Birmingham, UK
e-mail address: a.das@bham.ac.uk, axd1010@alumni.bham.ac.uk

Abstract. We introduce Pudlák-Buss style Prover-Adversary games to characterise proof systems reasoning over deterministic branching programs (BPs) and non-deterministic branching programs (NBPs). Our starting points are the proof systems eLDT and eLNDT, for BPs and NBPs respectively, previously introduced by Buss, Das and Knop. We prove polynomial equivalences between these proof systems and the corresponding games we introduce. This crucially requires access to a form of *negation* of branching programs which, for NBPs, requires us to formalise a non-uniform version of the Immerman-Szelepcsényi theorem that $co\mathbf{NL} = \mathbf{NL}$. Thanks to the techniques developed, we further obtain a proof complexity theoretic version of Immerman-Szelepcsényi, showing that eLNDT is polynomially equivalent to systems over boundedly alternating branching programs.

1. Introduction

Proof complexity investigates the size of proofs, in terms of that of their conclusions. Originally motivated by the Cook-Levin theorem that Boolean satisfiability is **NP**-complete, finding general superpolynomial lower bounds on proof size for Boolean logic directly implies $\mathbf{P} \neq \mathbf{NP}$, what is now known as *Cook's programme*. Systems of interest in proof complexity are typically parametrised by a complexity class of interest, whose nonuniform counterpart comprises the objects of reasoning in the associated proof system. For instance Frege systems reason on formulas, the nonuniform counterpart of **ALOGTIME** [Bus87]. For **P** the corresponding system is *extended* Frege, employing 'Tseitin extension' to represent the dag structure of circuits.

Recently Buss, Das and Knop proposed a proof complexity theory of **L** and **NL** via systems, eLDT and eLNDT, reasoning about deterministic branching programs (BPs) and non-deterministic branching programs (NBPs) respectively [BDK20]. In earlier work [DD22, DD25] the current authors studied the 'positive' fragment of eLNDT, reasoning about *monotone* branching programs, and showed a polynomial simulation over positive sequents, inspired by previous developments for the sequent calculus [AGP02]. In all these works (and often in proof complexity), reasoning about families of propositional proofs can be cumbersome and notationally complex. This issue is made worse when the objects

Key words and phrases: Proof complexity, branching programs, non-determinism, logspace.

The alphabetically first author has been supported by a UKRI Future Leaders Fellowship, *Structure vs Invariants in Proofs*, project number MR/S035540/1.

DOI:10.46298/LMCS-22(2:18)2026

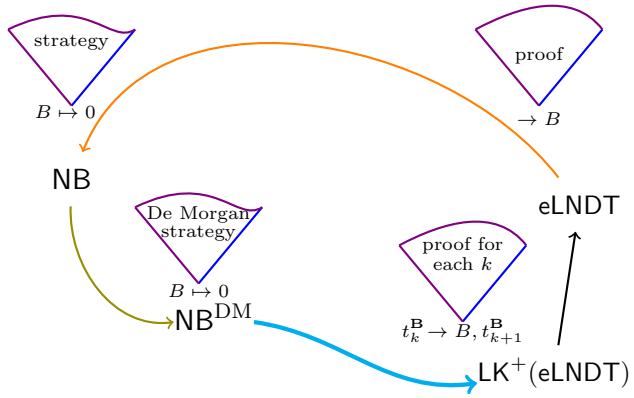


Figure 1: High-level structure of the polynomial simulations for the non-deterministic setting. The right side displays inference systems and proofs therein, while the left side displays game systems and strategies therein. The bold cyan arrow is the most technical, incorporating our formalisation of Immerman-Szelepcsényi.

of reasoning have underlying dag structures, represented using extension where indexing must be carefully controlled to maintain well-foundedness, yielding further technicalities for handling program equivalence/simulation.

A promising solution to these issues is the program of *bounded arithmetic* [Bus85, Kra95, CN10, Kra19]. Here (very) weak theories of arithmetic essentially serve as uniform counterparts of propositional proof systems. However in this work we follow a different branch of the proof complexity literature: *Prover-Adversary games*, à la Pudlák-Buss [PB94]. Roughly speaking, these are two-player games where Prover asks queries (typically the objects of reasoning) and Adversary assigns each query a Boolean value; Prover wins if they can force Adversary into a 'simple contradiction' (typically one 'easily' witnessed by polynomial-size proofs). In this way proofs are recast as *strategies*, with the logarithm of proof size proportional to the depth of a strategy. Indeed, we can see such 'game systems' as canonical 'balanced tree-like' versions of their corresponding inference systems, cf. [Kra94].

Contributions. In this work we develop games DB and NB for reasoning about BPs and NBPs, respectively, and prove their correspondence to eLDT and eLNDT, respectively. Classically, the equivalence of tree-like and dag-like versions of a system typically requires closure of the language under Boolean combinations (in particular negation). Such bootstrapping is readily formalisable in eLDT, with relatively simple constructions of Boolean combinations of (deterministic) BPs (see, e.g., [Weg00]). In fact this idea of using BP-based games for $\mathbf{L}$ proof complexity was already proposed by Cook [Coo01] (but, as far as we know, never published). However the problem becomes more complex for eLNDT, which comprises the main focus of this work. Here the idea is to compute the *negation* of an NBP as an NBP by formalising a partial non-uniform version of the Immerman-Szelepcsényi theorem that $co\mathbf{NL} = \mathbf{NL}$ [Imm88, Sze88]. A novelty of our formalisation is that we are able to simplify the inductive counting argument: we do not encode counting at the level of NBPs themselves, but rather at the level of the *proof*, relying on constructions of (positive) counting programs

from previous work [DD22, DD25]. Thus the programs we construct are *partial* negations, working relative to a fixed number of true inputs.

Let us point out a particular design choice at this point. In order to more easily reason about proof complexity via games, we simply close the queries of our games under (explicit) Boolean combinations. Of course there is no free lunch here: the aforementioned formalisation of Immerman-Szelepcsényi is duly carried out in the proof system eLNDT.

As an application of the machinery built up in this work, we develop in Section 7 a bona fide proof complexity theoretic version of the Immerman-Szelepcsényi theorem. Namely we show that eLNDT polynomially simulates a system, $\mathsf{eL}\exists\forall\mathsf{DT}$, reasoning over twice-alternating branching programs ($\exists\forall$BPs). Here we exploit the previously constructed partial negations of NBPs to reduce any $\exists\forall$BP to an equivalent NBP, relative to a fixed number of true inputs. Again, the general polynomial simulation is recovered by using a counting argument at the proof level, exhausting every possible number of true inputs.

Structure. In Section 2 we recall the systems eLDT and eLNDT from [BDK20], and in Section 3 we introduce our corresponding games DB and NB. In Section 4 we prove one direction of the correspondence, translating eLDT and eLNDT proofs to DB or NB strategies, respectively. In Section 4.3 we present the converse direction for the deterministic setting, translating DB strategies to eLDT proofs. The remainder of the paper is devoted to establishing the same for the non-deterministic setting.

In Section 5 we present our formalisation of Immerman-Szelepcsényi, witnessed by polynomial-size proofs in eLNDT. Finally in Section 6 we use this to establish a translation from NB strategies to eLNDT proofs. This final argument is quite involved, composed of several intermediate systems; we visualise its structure in Figure 1, and give some explanation in the caption. The main difficulty is the bold cyan arrow (bottom), where we rely on the Immerman-Szelepcsényi construction from Section 5.

2. Proof systems for (non-deterministic) branching programs

In this section we shall briefly recall the systems eLDT and eLNDT, from [BDK20, BDK19] and also appearing in [DD22, DD25], reasoning about deterministic and non-deterministic branching programs respectively.

Throughout this work, we make use of **propositional variables**, written p, q, etc. An **assignment** is a map from propositional variables to Booleans $\{0, 1\}$. A **Boolean function** is a map from assignments to $\{0, 1\}$.

In proof complexity, formally, a (sound) **propositional proof system** is just a polynomial-time function P from Σ^* to the set of propositional tautologies, for Σ some finite alphabet. The idea is that P checks (efficiently) that an element $\sigma \in \Sigma^*$ correctly codes a proof in the system in which case the output $P(\sigma)$ is the tautology σ proves. Otherwise P outputs the tautology 1 by convention.

A proof system P **polynomially simulates** a system Q if we can construct in polynomial-time, from a Q-proof of A, a P-proof of A. As is common in proof complexity, we will typically refrain from calculating explicit polynomial bounds throughout this work, which will always be evident from the arguments at hand. Furthermore, while we often only state the *existence* of small proofs, all of our arguments are feasibly constructive and such results comprise bona fide polynomial simulations.

A more comprehensive introduction to proof complexity can be found in the textbooks [Kra95, CN10, Kra19].

2.1. **(Non-)deterministic branching programs.** A **(non-deterministic) branching program (NBP)** is a (rooted) directed acyclic graph G such that:

- G has two distinguished **sink** nodes, 0 and 1.
- G has a unique **root** node, i.e. a unique node with in-degree 0.
- Each non-sink node of G is labelled by a propositional variable.
- Each edge of G is labelled by either 0 or 1.

We say that a NBP is **deterministic** (a **BP**) if each non-sink node has *exactly* two outgoing edges, one labelled 0 and the other labelled 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 p the run must follow an edge labelled by $\alpha(p) \in \{0,1\}$. G **accepts** α if there is a run on α reaching the sink 1. We extend α to a map from all NBPs to $\{0,1\}$ by setting $\alpha(G) = 1$ iff G accepts α. Thus each NBP **computes** a unique Boolean function $\alpha \mapsto \alpha(G)$.

Further background on (non-deterministic) branching programs can be found in, e.g., the textbook [Weg00].

2.2. **Representation via formulas with extension.** In order to syntactically represent (N)BPs, in addition to the propositional variables we fixed earlier we will also make use of a disjoint set of **extension variables**, written $e_0, e_1, \ldots$. We briefly recall the syntax for expressing and reasoning about (non-deterministic) branching programs from [BDK20, BDK19, DD22, DD25]. We refer the reader to those works for further examples and foundational details.

Extended non-deterministic decision-tree formulas, or simply **eNDT-formulas**, written A, B etc., are generated by the grammar:

$$A, B \quad ::= \quad 0 \mid 1 \mid ApB \mid A \vee B \mid e_i$$

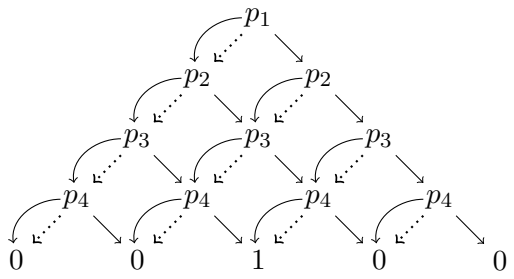
When writing formulas we employ some usual syntactic simplifications, omitting external and internal brackets when there is no ambiguity. The **size** of a formula A, written $|A|$, is the number of symbols occurring in A.

We often identify the propositional variable p with the formula $0p1$. **eDT-formulas** are eNDT-formulas not using $\vee$, and (N)DT-formulas are e(N)DT-formulas not using extension variables.

Remark 2.1 (Decision variables)**.** Note that, since extension variables are disjoint from propositional variables, Ae_iB is never a correct eNDT-formula; this will turn out to be important for our syntactic representation of (N)BPs to be faithful.

Semantically $\vee$ is interpreted as usual disjunction, while ApB is interpreted as a decision "if p then B else A". Thus formulas without extension variables compute Boolean functions as usual. Call such a formula **valid** if it returns 1 on all assignments. The following result renders systems reasoning about even extension-free formulas suitable for proof complexity analysis:

Proposition 2.2 [BDK20, BDK19]**.** *The set of valid NDT formulas (even disjunctions of DT formulas) is co***NP***-complete.**



$$
\begin{aligned}
e_{11} &\leftrightarrow e_{21}p_1(e_{21} \vee e_{22})\\
e_{21} &\leftrightarrow e_{31}p_2(e_{31} \vee e_{32})\\
e_{22} &\leftrightarrow e_{32}p_2(e_{32} \vee e_{33})\\
e_{31} &\leftrightarrow e_{41}p_3(e_{41} \vee e_{42})\\
e_{32} &\leftrightarrow e_{42}p_3(e_{42} \vee e_{43})\\
e_{33} &\leftrightarrow e_{43}p_3(e_{43} \vee e_{44})\\
e_{41} &\leftrightarrow 0p_4(0 \vee 0)\\
e_{42} &\leftrightarrow 0p_4(0 \vee 1)\\
e_{43} &\leftrightarrow 1p_4(1 \vee 0)\\
e_{44} &\leftrightarrow 0p_4(0 \vee 0)
\end{aligned}
$$

Figure 2: NBP for 2-out-of-4 threshold and representation by extension axioms. Here 0-edges are dotted, and 1-edges are solid; the multiple 0-leaves correspond to the same sink.

The interpretation of extension variables is parametrised by a set of 'extension axioms'. These axioms give meaning to extension variables by rendering them abbreviations of more complex formulas. Viewing (N)BPs as graphs, extension variables are essentially used to 'name' the nodes.

Definition 2.3 (Extension axioms)**.** A set of **extension axioms** $\mathcal{E}$ is a set of the form $\{e_i \leftrightarrow E_i\}_{i<n}$, where each formula E_i may only contain extension variables among $e_0, \ldots, e_{i-1}$.

Example 2.4 (2-out-of-4 threshold)**.** The 2-out-of-4 threshold function, returning 1 iff at least two of its four inputs are 1, is computed by the NBP on the left of Fig. 2. This NBP may be represented by the extension variable e_{11} under the extension axioms on the right of Fig. 2. Each e_{ij} represents the j^{th} node (left to right) on the i^{th} row (top to bottom) for $1 \leq i \leq 4$ and $1 \leq j \leq i$.

For well-foundedness of the extension axioms (i.e. the subscripting condition of Definition 2.3), note that we may identify each e_{ij} with $e_{4(4-i)+j}$. We shall typically leave such identifications implicit in the sequel.

Note in Definition 2.3 the condition that each e_i 'abbreviates' only formulas with extension variables of smaller index. This means that the graphs described by formulas under a set of extension axioms is well-founded, i.e. a dag, inducing a natural induction principle:

Remark 2.5 ($\mathcal{E}$-induction)**.** Given a set of extension axioms $\mathcal{E} = \{e_i \leftrightarrow E_i\}_{i<n}$ we may define a strict partial order $<_{\mathcal{E}}$ on formulas over $e_0, \ldots, e_{n-1}$ by:

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

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

From here, for instance, we obtain:

Definition 2.6 (Semantics of eNDT formulas)**.** The semantics of eNDT formulas with respect to a set of extension axioms $\mathcal{E} = \{e_i \leftrightarrow E_i\}_{i<n}$, are given by $\vDash_{\mathcal{E}}$, an infix binary relation between assignments and formulas over $e_0, \ldots, e_{n-1}$ defined by $\mathcal{E}$-induction by:

Identity and cut:

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

Structural rules:

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

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

Logical rules:

$$0\text{-}l\ \frac{}{0 \to} \qquad 1\text{-}l\ \frac{\Gamma \to \Delta}{\Gamma, 1 \to \Delta} \qquad \vee\text{-}l\ \frac{\Gamma, A \to \Delta \quad \Gamma, B \to \Delta}{\Gamma, A \vee B \to \Delta}$$

$$0\text{-}r\ \frac{\Gamma \to \Delta}{\Gamma \to \Delta, 0} \qquad 1\text{-}r\ \frac{}{\to 1} \qquad \vee\text{-}r\ \frac{\Gamma \to \Delta, A, B}{\Gamma \to \Delta, A \vee B}$$

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

Figure 3: Rules for systems (e)LDT and (e)LNDT.

- $\alpha \not\models_{\mathcal{E}} 0$ and $\alpha \models_{\mathcal{E}} 1$.
- $\alpha \models_{\mathcal{E}} ApB$ if either $\alpha(p) = 0$ and $\alpha \models_{\mathcal{E}} A$, or $\alpha(p) = 1$ and $\alpha \models_{\mathcal{E}} B$.
- $\alpha \models_{\mathcal{E}} A \vee B$ if $\alpha \models_{\mathcal{E}} A$ or $\alpha \models_{\mathcal{E}} B$.
- $\alpha \models_{\mathcal{E}} e_i$ if $\alpha \models_{\mathcal{E}} E_i$.

If $\alpha \models_{\mathcal{E}} A \iff f(\alpha) = 1$ for some Boolean function f, we say that A **computes** f **over** $\mathcal{E}$. We may also say that A is **valid over** $\mathcal{E}$ if, for every assignment α we have $\alpha \models_{\mathcal{E}} A$ (equivalently, A computes the constant function 1 over $\mathcal{E}$).

Since many of our arguments are based on $\mathcal{E}$-induction, let us recall the following complexity analysis from earlier work:

Proposition 2.7 (Complexity of $\mathcal{E}$-induction, e.g. [DD25])**.** *Let A be an eDT or eNDT formula over a set of extension axioms $\mathcal{E} = \{e_i \leftrightarrow E_i\}_{i<n}$. Then $|\{B <_{\mathcal{E}} A\}| \leq |A| + \sum_{i<n} |E_i|$ and, if $B <_{\mathcal{E}} A$, then $|B| \leq \max(|A|, |E_0|, \ldots, |E_{n-1}|)$.*

2.3. **Proof systems.** We now recall the sequent systems eLDT and eLNDT reasoning over eDT and eNDT formulas, respectively, first introduced in [BDK19, BDK20].[1] Lines in these systems represent deterministic and non-deterministic branching programs respectively.

A **sequent** is an expression $\Gamma \to \Delta$, where Γ and Δ (called **cedents**) are lists of formulas ('$\to$' is just a syntactic delimiter). Such a sequent is interpreted as "if all formulas in the antecedent Γ are true then some formula of the succedent Δ is true".

[1]The qualifier L here is just a standard indication for a sequent system over the corresponding logic, e.g. LK for classical logic and LJ for intuitionistic logic, originating from Gentzen's works.

Definition 2.8 (Systems for e(N)DT formulas)**.** The system LNDT is given by the rules in Fig. 3. LDT is the restriction of LNDT not including rules for disjunction, $\vee$.

An LNDT (LDT) **derivation** of $\Gamma \longrightarrow \Delta$ from **hypotheses** $\mathcal{H} = \{\Gamma_i \longrightarrow \Delta_i\}_{i\in I}$ is defined as expected: it is a finite list of sequents, each either some $\Gamma_i \longrightarrow \Delta_i$ from $\mathcal{H}$ or following from previous ones by rules of LNDT (LDT, resp.) and ending with $\Gamma \longrightarrow \Delta$. An eLNDT (eLDT) **proof** of a sequent $\Gamma \longrightarrow \Delta$ over a set of ($\vee$-free, resp.) extension axioms $\mathcal{E}$ is an LNDT (LDT, resp.) derivation π with hypotheses $\mathcal{E}$.[2] If $\Gamma \longrightarrow \Delta$ does not have extension variables we just say that π is an eLNDT (or eLDT, resp.) proof of $\Gamma \longrightarrow \Delta$.

The **size** of a derivation π, written $|\pi|$, is the number of symbols occurring in it.

Remark 2.9 (Cedents as lists)**.** As usual, it is not important whether cedents are construed as lists, multisets or sets for studying proof complexity of eLDT and eLNDT. However working with lists accommodates more faithfully the definition of long disjunctions and conjunctions later when we work with games.

Proposition 2.10 (Soundness and completeness, [BDK20, BDK19])**.** *Suppose* Γ, Δ *are sequents of (N)DT formulas. Then, respectively,* L(N)DT *proves* $\Gamma \longrightarrow \Delta$ *iff for every assignment* α*, either* $\alpha \not\models A$ *for some* $A \in \Gamma$ *or* $\alpha \models B$ *for some* $B \in \Delta$*.*

Note that, as the validity problem for disjunctions DT formulas is *co***NP**-complete, cf. Proposition 2.2, so is the provability of DT-sequents. In fact we can establish a stronger version of the above result, relativised to a set of extension axioms, though we do not ever rely on this. Nonetheless some of our proof translations will be parametrised by sets of extension axioms.

Example 2.11. Recalling Example 2.4, let us write $\mathcal{E}$ for the set of extension axioms in Fig. 2, right. Over this set we have eLNDT proofs of, say, $e_{42} \longrightarrow e_{43}$ by,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{0 \longrightarrow}\,0\text{-}l}{0 \longrightarrow e_{43}, p_4}\,\mathsf{w} \qquad \dfrac{\dfrac{\dfrac{}{\longrightarrow 1}\,1\text{-}r}{p_4, 0 \vee 1 \longrightarrow 1, p_4}\,\mathsf{w} \qquad \dfrac{\dfrac{\dfrac{}{\longrightarrow 1}\,1\text{-}r}{p_4, 0 \vee 1 \longrightarrow 1, 0}\,\mathsf{w}}{p_4, 0 \vee 1 \longrightarrow 1 \vee 0}\,\vee\text{-}r}{p_4, 0 \vee 1 \longrightarrow 1 p_4 (1 \vee 0)}\,p_4\text{-}r}{p_4, 0 \vee 1 \longrightarrow e_{43}}\,\mathcal{E}\text{-}r}{0 p_4 (0 \vee 1) \longrightarrow e_{43}}\,p_4\text{-}l}{e_{42} \longrightarrow e_{43}}\,\mathcal{E}\text{-}l \tag{2.1}$$

where, formally, the steps $\mathcal{E}$-l and $\mathcal{E}$-r are obtained by 'cutting' against the appropriate extension hypotheses. We have also omitted several 'contraction' and 'exchange' steps, essentially treating each side of the sequent as *sets* of formulas instead of lists for convenience. Note that, while we have displayed the proof above as a tree, proofs may in general be *dags*, so that when measuring proof complexity it suffices to count only the number of symbols in *distinct* sequents. E.g. we may *identify* the two initial sequents $\longrightarrow 1$ above.

[2] $A \leftrightarrow B$ is construed as an abbreviation for the pair of sequents $A \longrightarrow B$ and $B \longrightarrow A$.

We can use this to build further proofs, e.g. of $e_{32} \longrightarrow e_{33}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(2.1)}{e_{42} \longrightarrow e_{43}}}{e_{42} \longrightarrow p_3, e_{43}}\,\mathsf{w}\text{-}r \quad \dfrac{\mathsf{id}\ \dfrac{}{p_3 \longrightarrow p_3}}{e_{42}, p_3 \longrightarrow p_3 e_{43} \vee e_{44}}\,\mathsf{w}}{e_{42} \longrightarrow p_3, e_{43} p_3 (e_{43} \vee e_{44})}\,p_3\text{-}r \quad \dfrac{\dfrac{\mathsf{id}\ \dfrac{}{p_3 \longrightarrow p_3}}{p_3, e_{42} \vee e_{43} \longrightarrow e_{43}, p_3}\,\mathsf{w} \quad \dfrac{\dfrac{\dfrac{(2.1)}{e_{42} \longrightarrow e_{43}} \quad \mathsf{id}\ \dfrac{}{e_{43} \longrightarrow e_{43}}}{e_{42} \vee e_{43} \longrightarrow e_{43}}\,\vee\text{-}l}{p_3, e_{42} \vee e_{43} \longrightarrow e_{43}, e_{44}}\,\mathsf{w}}{p_3, e_{42} \vee e_{43} \longrightarrow e_{43} \vee e_{44}}\,\vee\text{-}r}{p_3, e_{42} \vee e_{42} \longrightarrow e_{43} p_3 (e_{43} \vee e_{44})}\,p_3\text{-}r}{e_{42} p_3 (e_{42} \vee e_{43}) \longrightarrow e_{43} p_3 (e_{43} \vee e_{44})}\,p_3\text{-}l}{e_{32} \longrightarrow e_{33}}\,\mathcal{E} \tag{2.2}$$

More generally we can build eLNDT proofs of all $e_{ij} \longrightarrow e_{ik}$ for $j \leq k$.

Example 2.12. We can build a LNDT proof of $(ApB)qD \longrightarrow AqD \vee p$ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{id}\ \dfrac{}{A \longrightarrow A}}{A \longrightarrow A, q, p}\,\mathsf{w} \quad \dfrac{\mathsf{id}\ \dfrac{}{q \longrightarrow q}}{q \longrightarrow D, q, p}\,\mathsf{w}}{A \longrightarrow AqD, q, p}\,q\text{-}r \quad \dfrac{\mathsf{id}\ \dfrac{}{p \longrightarrow p}}{p, B \longrightarrow AqD, q, p}\,\mathsf{w}}{ApB \longrightarrow AqD, q, p}\,p\text{-}l \quad \dfrac{\dfrac{\mathsf{id}\ \dfrac{}{q \longrightarrow q}}{q, D \longrightarrow A, q, p}\,\mathsf{w} \quad \dfrac{\mathsf{id}\ \dfrac{}{D \longrightarrow D}}{q, D \longrightarrow D, p}\,\mathsf{w}}{q, D \longrightarrow AqD, p}\,q\text{-}r}{(ApB)qD \longrightarrow AqD, p}\,q\text{-}l}{(ApB)qD \longrightarrow AqD \vee p}\,\vee$$

This has an LDT subproof of $(ApB)qD \longrightarrow AqD, p$. We can similarly build an LDT proof of $(ApB)qD, p \longrightarrow BqD$. By combining these two as below we can obtain the 'distribution' property $(ApB)qD \longrightarrow (AqD)p(BqD)$ in LDT:

$$p\text{-}r\ \dfrac{(ApB)qD \longrightarrow AqD, p \quad (ApB)qD, p \longrightarrow BqD}{(ApB)qD \longrightarrow (AqD)p(BqD)}$$

3. Prover-Adversary games for (non-deterministic) branching programs

The goal of this section is to present games that correspond to $\mathsf{eL(N)DT}$ in the same way that the Boolean formula game of Pudlák and Buss corresponds to Frege [PB94]. The basic idea of these games is simple. Two players, **Prover** and **Adversary** alternate turns as follows: Prover asks 'queries', and Adversary assigns them Boolean values. Prover wins if the set of Adversary's answers ever include a 'simple contradiction'. Such games are duly parametrised by their notions of 'query' and 'simple contradiction'. Before presenting our games, we first need to develop some further machinery for defining these data.

3.1. 'Similar' representations of branching programs.

Since (N)BPs have underlying dag structure that we represent via extension, we must address the matter of *equivalent* representations. While we can mostly sidestep this issue for eLDT and eLNDT, with different representations being demonstrably equivalent by way of polynomial-size proofs, in the setting of games we must be more resource-conscious of the number of rounds it takes to win (which should be logarithmic). For this reason we shall 'bootstrap' our games to include native contradictory specifications, namely for a notion of *simulation* of transition systems

specialised to our syntax. In this way, we can really see our games as operating *directly* on (N)BPs rather than their representations, equating bisimilar programs.

We define a judgement $A \gtrsim_{\mathcal{E}} B$ meaning that the NBP represented by A (over $\mathcal{E}$) *simulates* the NBP represented by B (over $\mathcal{E}$), as transition systems.

Definition 3.1 (Simulation)**.** Let $\mathcal{E} = \{e_i \leftrightarrow E_i\}_{i<n}$ be a set of extension axioms. We define the judgement $A \gtrsim_{\mathcal{E}} B$, read '$A$ **$\mathcal{E}$-simulates** B', by:

$$\frac{}{A \gtrsim_{\mathcal{E}} A} \qquad \frac{A \gtrsim_{\mathcal{E}} C \quad A \gtrsim_{\mathcal{E}} D}{A \gtrsim_{\mathcal{E}} C \vee D} \qquad \frac{A \gtrsim_{\mathcal{E}} E_i}{A \gtrsim_{\mathcal{E}} e_i}$$
$$\frac{A \gtrsim_{\mathcal{E}} C \quad B \gtrsim_{\mathcal{E}} D}{ApB \gtrsim_{\mathcal{E}} CpD} \qquad \frac{A_i \gtrsim_{\mathcal{E}} B}{A_0 \vee A_1 \gtrsim_{\mathcal{E}} B} \qquad \frac{E_i \gtrsim_{\mathcal{E}} B}{e_i \gtrsim_{\mathcal{E}} B} \tag{3.1}$$

Remark 3.2. It is not difficult to see that $A \gtrsim_{\mathcal{E}} B$ if and only if the NBP represented by A (over $\mathcal{E}$) *simulates* the NBP represented by B (over $\mathcal{E}$), as labelled transition systems (see e.g. [CB08, Section 7.4] for definitions). The forwards implication is by induction on the definition of $\gtrsim_{\mathcal{E}}$, while the backwards implication follows by $\mathcal{E}$-induction on B then A.

Proposition 3.3 (Simulation in eLNDT)**.** *Let $\mathcal{E}$ be a set of eNDT extension axioms and suppose $A \gtrsim_{\mathcal{E}} B$. There are polynomial-size eLNDT proofs of $B \longrightarrow A$ over $\mathcal{E}$. If $\mathcal{E}, A, B$ are $\vee$-free then these proofs are furthermore in eLDT.*

Proof. We proceed by induction on the definition of simulation, deriving each rule of Eq. (3.1) in eLNDT, noticing that we do not need to use $\vee$-steps when $\mathcal{E}, A, B$ are $\vee$-free. Almost every step is routine, requiring only a couple steps in eLNDT, except for the lower left rule of Eq. (3.1), which is derived by,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{C \longrightarrow A}\,IH}{C \longrightarrow A, p, p}\,\mathsf{w} \quad \dfrac{\dfrac{}{p \longrightarrow p}\,\mathsf{id}}{p, D \longrightarrow A, p}\,\mathsf{w}}{CpD \longrightarrow A, p}\,p\text{-}l \quad \dfrac{\dfrac{\dfrac{}{p \longrightarrow p}\,p}{C, p \longrightarrow B, p}\,\mathsf{w} \quad \dfrac{\dfrac{}{D \longrightarrow B}\,IH}{p, D, p \longrightarrow B}\,\mathsf{w}}{CpD, p \longrightarrow B}\,p\text{-}l}{CpD \longrightarrow ApB}\,p\text{-}r$$

where the steps marked *IH* are obtained by the inductive hypothesis. □

Note that, in the case of deterministic BPs, simulation reduces to the simpler notion of equivalence between branching programs with the same 'unfolding' as decision trees.

Definition 3.4 (Unfolding)**.** Let $\mathcal{E} = \{e_i \leftrightarrow E_i\}_{i<n}$ be a set of $\vee$-free extension axioms and A a eDT formula. We define the DT-formula $\mathrm{Unf}_{\mathcal{E}}(A)$ by $\mathcal{E}$-induction on A:

- $\mathrm{Unf}_{\mathcal{E}}(0) := 0$
- $\mathrm{Unf}_{\mathcal{E}}(1) := 1$
- $\mathrm{Unf}_{\mathcal{E}}(ApB) := \mathrm{Unf}_{\mathcal{E}}(A)p\mathrm{Unf}_{\mathcal{E}}(B)$
- $\mathrm{Unf}_{\mathcal{E}}(e_i) := \mathrm{Unf}_{\mathcal{E}}(E_i)$, for $i < n$.

In the presence of disjunction the natural extension of the notion of unfolding above is not canonical: the same NBP may be represented by different bracketings of $\vee$ under associativity and commutativity. This is why we must work with the notion of simulation above in the games we introduce later. In any case, for deterministic BPs:

Proposition 3.5. *Let $\mathcal{E}$ be a set of $\vee$-free extension axioms and A, B eDT formulas. $A \gtrsim_{\mathcal{E}} B$ if and only if $\mathrm{Unf}_{\mathcal{E}}(A) = \mathrm{Unf}_{\mathcal{E}}(B)$.*

Proof sketch. The $\Leftarrow$ direction is routine, following by $\mathcal{E}$-induction on A and B. For the $\Rightarrow$ direction we proceed by induction on the definition of simulation. Since $\mathcal{E}, A, B$ are $\vee$-free, no disjunction cases of the definition of $\gtrsim_{\mathcal{E}}$ apply. All the remaining cases follow immediately from inductive hypothesis and the definition of $\mathrm{Unf}_{\mathcal{E}}(-)$. □

3.2. **Boolean combinations of branching programs.** The next matter we must address is to some extent a design choice. To prove equivalence of our games and $\mathsf{eL(N)DT}$, recall that games are essentially tree-like versions of the inference system they correspond to, so we shall need to prove some sort of closure under Boolean combinations, cf. [Kra94]. For eLNDT, this will amount to the formalisation of a (non-uniform) version of Immerman-Szelepcsényi.

We could carry out this work either within a game system or within an inference system. However doing so within games requires us to again be resource conscious of the number of rounds, which must be logarithmic, and it is not clear to us that this can be duly carried out without further bootstrapping. Thus we simply expand the queries of our game to be as expressive as possible, closing them under Boolean combinations. This has the effect of simplifying the translation from $\mathsf{eL(N)DT}$ proofs to strategies, but rendering the converse more difficult.

Definition 3.6 (Boolean combinations)**.** Write P, Q, R etc. for **Boolean combinations** of e(N)DT formulas (Bool**(e(N)DT)-formulas**), generated by,

$$P, Q, R, \ldots \quad ::= \quad A \mid \neg Q \mid Q \vee R \mid Q \wedge R$$

where A ranges over e(N)DT formulas.

The intended semantics of this extended class of formulas are as expected, again parametrised by a set of extension axioms in order to interpret the extension variables in a e(N)DT-subformula. Note that the syntax here is 'two-tiered': Boolean connectives may not alternate with decisions and extensions (except $\vee$ in the case of Bool(eNDT)). This is reflected by the use of different metavariables (P, Q, R etc.) for Boolean combinations of branching programs. Later we consider intermediate systems that extend $\mathsf{eL(N)DT}$ by (positive) Boolean combinations, interpolating the translation from strategies to $\mathsf{eL(N)DT}$.

3.3. **Games for (non-deterministic) branching programs.** Finally we are in a position to define our games for deterministic and non-deterministic branching programs, ultimately corresponding to eLDT and eLNDT respectively.

A **(Prover-Adversary) game** is given by a set of **queries** and a set of **(simple) contradictions**, which are sets of Boolean assignments to queries. From here the mechanics of the Prover-Adversary game are defined as in [PB94]: beginning with an initial set of assignments S (a **state** or **specification**) Prover asks queries, and Adversary assigns them a Boolean value. If during a play the set of assignments accumulated (including the initial ones) includes a simple contradiction then Prover wins. A **winning strategy** σ (for Prover) from S is represented as a full binary tree in the natural way, with queries labelling non-leaf nodes, as in Figure 4 for the game we are about to define. Writing $\Gamma \mapsto b$ for $\{A \mapsto b : A \in \Gamma\}$, we call a winning strategy from state $\{\Gamma \mapsto 1, \Delta \mapsto 0\}$ a **proof** of $\Gamma \longrightarrow \Delta$, with respect to the underlying game.

We are finally ready to present our games corresponding to $\mathsf{eL(N)DT}$:

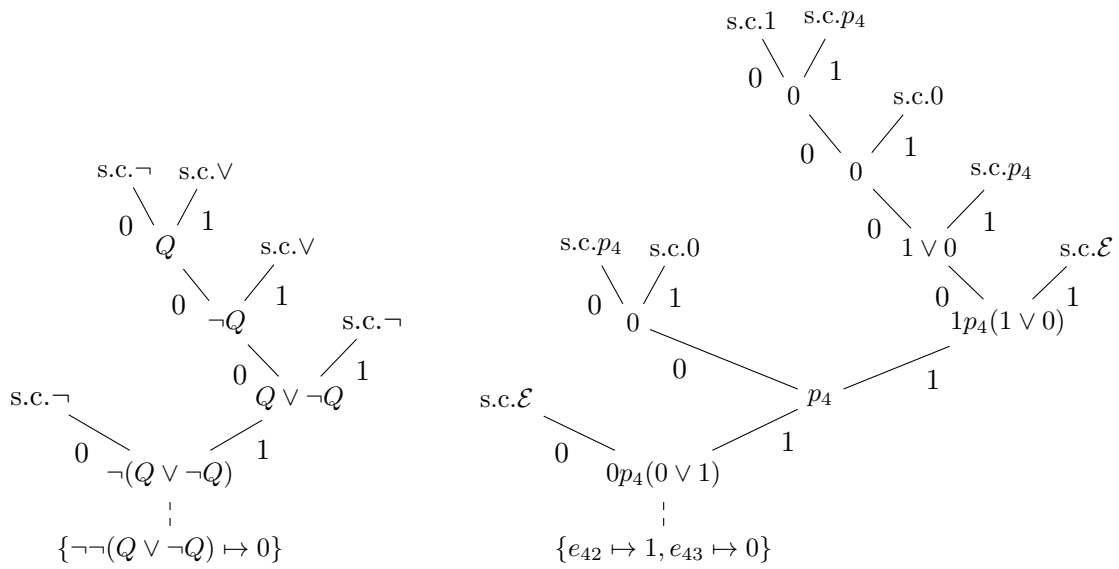


Figure 4: Two winning strategies for Prover, over $\mathcal{E}$ the set of extension axioms from Fig. 2. Initial states are indicated below the roots by dashed edges. Simple contradictions are indicated as leaves marked 's.c.' along with their type.

Definition 3.7. The **non-deterministic branching program game** NB (over $\mathcal{E} = \{e_i \leftrightarrow E_i\}_{i<n}$) is given by:

- Queries are just the Bool(eNDT)-formulas.
- Simple contradictions consist of just:
  - Boolean contradictions: $\{0 \mapsto 1\}$ and $\{1 \mapsto 0\}$.
  - Decision contradictions: $\{A_0 \mapsto b_0, A_1 \mapsto b_1, p \mapsto i, A_0 p A_1 \mapsto c : c \neq b_i\}$.
  - Boolean connective contradictions: all sets inconsistent with the truth tables for $\neg, \vee, \wedge$:
    - $\{Q \mapsto b, \neg Q \mapsto b\}$
    - $\{Q_0 \vee Q_1 \mapsto 0, Q_i \mapsto 1\}$ and $\{Q_0 \vee Q_1 \mapsto 1, Q_0 \mapsto 0, Q_1 \mapsto 1\}$
    - $\{Q_0 \wedge Q_1 \mapsto 0, Q_0 \mapsto 1, Q_1 \mapsto 1\}$ and $\{Q_0 \wedge Q_1 \mapsto 1, Q_i \mapsto 0\}$.
  - Extension contradictions: $\{e_i \mapsto b, E_i \mapsto 1 - b\}$.
  - Similarity contradictions:[3] $\{A \mapsto 0, B \mapsto 1\}$ whenever $A \gtrsim_{\mathcal{E}} B$.

The **deterministic branching game** DB (wrt $\mathcal{E}$) is defined the same way, only wrt Bool(eDT) instead of Bool(eNDT).

Example 3.8. In Fig. 4 we give two winning strategies for Prover, one from the state $\{\neg\neg(Q \vee \neg Q) \mapsto 0\}$, left, and one from $\{e_{42} \mapsto 1, e_{43} \mapsto 0\}$, right, over the set $\mathcal{E}$ of extension axioms from Fig. 2. They are NB proofs of the sequents $\longrightarrow \neg\neg(Q \vee \neg Q)$ and $e_{42} \longrightarrow e_{43}$ (over $\mathcal{E}$). The left strategy is also a DB proof, as long as Q is $\vee$-free.

We shall continue to notate (winning) strategies similarly to Fig. 4.

[3]Note that the similarity contradictions actually include the extension ones as a special case, but this serves as some useful redundancy.

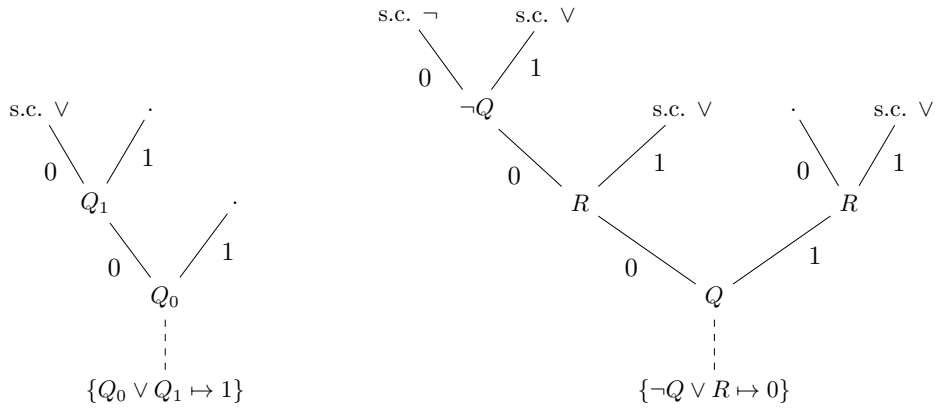


Figure 5: Examples of strategies for finding and forcing.

4. Proofs to strategies

In this section we present one direction of our main results, translating propositional proofs in eLDT or eLNDT to strategies in DB or NB, respectively:

Theorem 4.1. *If* eLNDT *(or* eLDT*) has a size N proof of a sequent $\Gamma \longrightarrow \Delta$ over $\mathcal{E}$, there is a $O(\log N)$ round winning strategy from $\{\Gamma \mapsto 1, \Delta \mapsto 0\}$ in* NB *(or* DB*, resp.) over $\mathcal{E}$.*

Also in this section we shall present a converse to this result in the deterministic setting. All definitions and results in this section apply to both DB and NB, unless otherwise stated.

4.1. **A strategic toolbox.** Before presenting a proof of Theorem 4.1 above, let us build up some machinery for building strategies. When describing strategies, we shall use some suggestive terminology:

- "**force** $Q \mapsto b$ (or win) from S in r rounds" is a (partial) strategy from S of depth $\leq r$ with each leaf either a simple contradiction or has incoming edge $Q \mapsto b$;
- "**find** $\alpha \in T$ (or win) from S in r rounds" is a (partial) strategy from S of depth $\leq r$ with each leaf either a simple contradiction or has incoming edge labelled by some assignment in T, say α.

We shall almost always omit 'or win' when using these phrases. We shall often expand out '$\alpha \in T$' according to the context, e.g. saying 'find $Q \in \Gamma$ with $Q \mapsto b$'.

Example 4.2 (Forcing and finding)**.** From $\{Q_0 \vee Q_1 \mapsto 1\}$ we can *find* $Q_i \mapsto 1$ in constantly many rounds by:

- ask Q_0; if $Q_0 \mapsto 1$ we are done; else,
- ask Q_1; if $Q_1 \mapsto 1$ we are done; else,
- we have a simple contradiction $\{Q_0 \vee Q_1 \mapsto 1, Q_0 \mapsto 0, Q_1 \mapsto 0\}$.

This (partial) strategy is visualised in Fig. 5, left.

Also, writing $Q \supset R$ as an abbreviation for $\neg Q \vee R$, we can *force* $Q \mapsto 1$ and $R \mapsto 0$ from $\{Q \supset R \mapsto 0\}$ in constantly many rounds:

- ask Q and R; if both $Q \mapsto 1$ and $R \mapsto 0$ we are done; else,
- if $R \mapsto 1$ we have a simple contradiction against $Q \supset R \mapsto 0$; else,

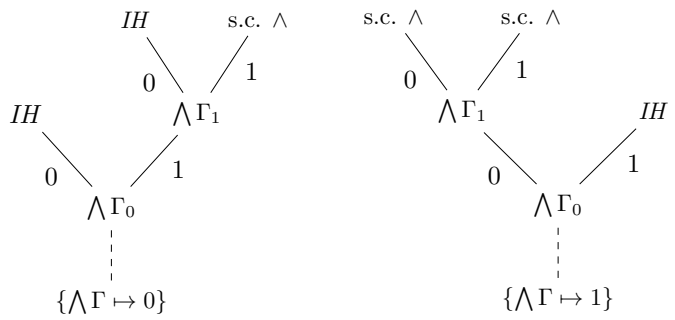


Figure 6: $O(\log n)$ round strategies for forcing and finding in long conjunctions.

- we have $Q \mapsto 0$, so ask $\neg Q$;
  - if $\neg Q \mapsto 1$ we have a simple contradiction against $Q \supset R \mapsto 0$; else,
  - if $\neg Q \mapsto 0$ we have a simple contradiction against $Q \mapsto 0$.

This (partial) strategy is visualised in Fig. 5, right.

Let us now see a more interesting, and useful, example. For a list of queries Γ we write $\bigwedge \Gamma$ for the conjunction of its members, bracketed as a (nearly) balanced binary tree. Formally, if $\Gamma = Q_1, \ldots, Q_k$ (and $k \geq 2$) then:

$$\bigwedge \Gamma := \bigwedge_{i=1}^{\lfloor k/2 \rfloor} Q_i \ \wedge \bigwedge_{i=\lfloor k/2 \rfloor + 1}^{k} Q_i$$

Similarly for long disjunctions, $\bigvee \Gamma$. We have:

Example 4.3. Let Γ be a nonempty list of queries of length n. From $\{\bigwedge \Gamma \mapsto b\}$ we can:
- $(b = 0)$ find $Q \in \Gamma$ with $Q \mapsto 0$ in $O(\log n)$ rounds.
- $(b = 1)$ force $Q \mapsto 1$ for any $Q \in \Gamma$ in $O(\log n)$ rounds.

Dually from $\{\bigvee \Gamma \mapsto b\}$ we can:
- $(b = 0)$ force $Q \mapsto 0$ for any $Q \in \Gamma$ in $O(\log n)$ rounds.
- $(b = 1)$ find $Q \in \Gamma$ with $Q \mapsto 1$ in $O(\log n)$ rounds.

To prove this we proceed by divide-and-conquer induction on n, focussing only on the first two items (the other two follow dually). The base case, when $n = 1$, is immediate. Otherwise, for $n > 1$, let Γ_0 and Γ_1 be (roughly) the first and second halves of Γ, so that $\bigwedge \Gamma = \bigwedge \Gamma_0 \wedge \bigwedge \Gamma_1$.
For the case $b = 0$:
- ask $\bigwedge \Gamma_0$ and $\bigwedge \Gamma_1$;
- if $\bigwedge \Gamma_0 \mapsto 0$ find $Q \in \Gamma_0$ with $Q \mapsto 0$, by inductive hypothesis; else,
- if $\bigwedge \Gamma_1 \mapsto 0$ find $Q \in \Gamma_1$ with $Q \mapsto 0$, by inductive hypothesis; else,
- we have a simple contradiction $\{\bigwedge \Gamma_0 \mapsto 1, \bigwedge \Gamma_1 \mapsto 1, \bigwedge \Gamma \mapsto 0\}$.

This is visualised in Fig. 6, left.
For the case $b = 1$, let $Q \in \Gamma_i$:
- ask $\bigwedge \Gamma_i$;
- if $\bigwedge \Gamma_i \mapsto 1$ force $Q \mapsto 1$ by inductive hypothesis; else,
- we have a simple contradiction $\{\bigwedge \Gamma_i \mapsto 0, \bigwedge \Gamma \mapsto 1\}$.

This is visualised in Fig. 6, right.

In the sequel we shall implicitly use similar techniques when constructing strategies.

4.2. **From proofs to strategies.** Thanks to the presence of Boolean combinations, Theorem 4.1 follows by essentially the same proof structure as Pudlák and Buss in [PB94]. The idea is also similar to the translation of Frege proofs into balanced tree-form [Kra94]: each sequent is construed as a single query by Boolean combinations, and the winning strategy searches for the first false sequent by a divide-and-conquer on conjunctions of sequents. This must contradict soundness of some rule, which is again established in logarithmically many rounds. Thus we first need the following intermediate result:

Lemma 4.4 (Local soundness)**.** *Fix an inference step as follows, where* $|\Sigma|, |\Pi|, |\Sigma_i|, |\Pi_i| \leq n$*:*

$$\mathsf{r}\,\frac{\Sigma_1 \longrightarrow \Pi_1 \quad \cdots \quad \Sigma_k \longrightarrow \Pi_k}{\Sigma \longrightarrow \Pi}$$

From $\{\bigwedge \Sigma \supset \bigvee \Pi \mapsto 0, \bigwedge \Sigma_1 \supset \bigvee \Pi_1 \mapsto 1, \ldots, \bigwedge \Sigma_k \supset \bigvee \Pi_k \mapsto 1\}$ *there is a strategy winning in* $O(\log n)$ *rounds.*

Proof. First note that by Example 4.2 we can:

($\star$) force $\bigwedge \Sigma \mapsto 1$ and force $\bigvee \Pi \mapsto 0$ in constantly many rounds.

Now, almost every case is the same as for the usual sequent calculus LK, equivalently Frege systems (see, e.g., [PB94]). The remaining cases are the decision rules of $\mathsf{eL(N)DT}$:

- If r is $_{p\text{-}l}\,\dfrac{\Gamma, A \longrightarrow \Delta, p \quad \Gamma, p, B \longrightarrow \Delta}{\Gamma, ApB \longrightarrow \Delta}$ then:
  - find $C_0 \in \Gamma, A$ with $C_0 \mapsto 0$ or $D_0 \in \Delta, p$ with $D_0 \mapsto 1$ in $O(\log n)$ rounds, similarly to Example 4.3;
  - find $C_1 \in \Gamma, p, B$ with $C_1 \mapsto 0$ or $D_1 \in \Delta$ with $D_1 \mapsto 1$ in $O(\log n)$ rounds, similarly to Example 4.3;
  - if we found some $C_i \in \Gamma$, from ($\star$) force $C_i \mapsto 1$ in $O(\log n)$ rounds for a contradiction;
  - if we found some $D_j \in \Delta$, from ($\star$) force $D_j \mapsto 0$ in $O(\log n)$ rounds for a contradiction.

  This leaves the possibilities that we found (i) $A \mapsto 0$ or $p \mapsto 1$; and, (ii) $p \mapsto 0$ or $B \mapsto 0$:
  - if $p \mapsto 1$ and $p \mapsto 0$ then we have immediately a simple contradiction;
  - in every other case, we have a decision contradiction against $ApB \mapsto 1$.
- If r is $_{p\text{-}r}\,\dfrac{\Gamma \longrightarrow \Delta, A, p \quad \Gamma, p \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, ApB}$ then:
  - find $C_0 \in \Gamma$ with $C_0 \mapsto 0$ or $D_0 \in \Delta A, p$ with $D_0 \mapsto 1$ in $O(\log n)$ rounds, similarly to Example 4.3;
  - find $C_1 \in \Gamma, p$ with $C_1 \mapsto 0$ or $D_1 \in \Delta, B$ with $D_1 \mapsto 1$ in $O(\log n)$ rounds, similarly to Example 4.3;
  - if we found some $C_i \in \Gamma$, from ($\star$) force $C_i \mapsto 1$ in $O(\log n)$ rounds for a contradiction;
  - if we found some $D_j \in \Delta$, from ($\star$) force $D_j \mapsto 0$ in $O(\log n)$ rounds for a contradiction.

  This leaves the possibilities that we found (i) $A \mapsto 1$ or $p \mapsto 1$; and, (ii) $p \mapsto 0$ or $B \mapsto 1$:
  - first, from ($\star$) force $ApB \mapsto 0$ in $O(\log n)$ rounds;
  - if $p \mapsto 1$ and $p \mapsto 0$ we immediately have a simple contradiction;
  - in every other case, we have a decision contradiction against $ApB \mapsto 0$. □

We can now finally give our translation from proofs to strategies:

Proof of Theorem 4.1. Fix a proof $\pi : (\Gamma_i \longrightarrow \Delta_i)_{i<n}$ over $\mathcal{E} = \{e_i \leftrightarrow E_i\}_{i<k}$ of size N and initial state $S = \{\Gamma_{n-1} \mapsto 1, \Delta_{n-1} \mapsto 0\}$. Let us employ the following abbreviations:

- $L_i := \bigwedge \Gamma_i$, for $i < n$;
- $R_i := \bigvee \Delta_i$, for $i < n$;
- $Q_i := L_i \supset R_i$, for $i < n$;
- $Q^i := \bigwedge_{j<i} Q_i$, for $i \leq n$.

First note that, by Examples 4.2 and 4.3, from S we can force in $O(\log N)$ rounds $L_{n-1} \mapsto 1$ and $R_{n-1} \mapsto 0$ and then also $Q_{n-1} \mapsto 0$, and so finally $Q^n \mapsto 0$. Now we construct a winning strategy from $Q^n \mapsto 0$ as follows:

- Find the least $i < n$ such that $Q_i \mapsto 0$, by a divide-and-conquer strategy: keep asking Q^j for j (roughly) halfway between the greatest l with $Q^l \mapsto 1$ (starting with $l = 0$) and least u with $Q^u \mapsto 0$ (starting with $u = n$), until $u - l = 1$. Set $i = u$ at this point, so that we have $Q_i \mapsto 0$ but $Q^i \mapsto 1$.
- If $\Gamma_i \longrightarrow \Delta_i$ is an extension axiom $e_j \longrightarrow E_j$ or $E_j \longrightarrow e_j$, then we can force a contradiction for extension in constantly many rounds.
- If $\Gamma_i \longrightarrow \Delta_i$ is the conclusion of an inference step with premisses (among) $\Gamma_j \longrightarrow \Delta_j$ and $\Gamma_k \longrightarrow \Delta_k$, then $j, k < i$ so we can force $Q_j \mapsto 1$ and $Q_k \mapsto 1$ from $Q^i \mapsto 1$ in $O(\log N)$ rounds by Example 4.3. Now from $\{Q_j \mapsto 1, Q_k \mapsto 1, Q_i \mapsto 0\}$ we can win in $O(\log N)$ rounds by Lemma 4.4. □

4.3. **Strategies to proofs: the deterministic case.** In the case of eLDT and DB, we can readily translate strategies back to proofs thanks to determinism, morally since (non-uniform) logspace is easily closed under Boolean combinations.

Theorem 4.5. *If* DB *has a size N proof of an eDT sequent $\Gamma \longrightarrow \Delta$ over $\mathcal{E}$ then* eLDT *has a* poly(N) *size proof of $\Gamma \longrightarrow \Delta$ over some $\mathcal{E}' \supseteq \mathcal{E}$.*

Corollary 4.6. eLDT *polynomially simulates* DB*, over DT sequents.*

This result serves as a warm-up before our development of the non-deterministic case in the next sections. For the proof it is convenient to work with a proof system that natively manipulates the queries of DB:

Definition 4.7 (System for Bool(eDT))**.** Write LK(eLDT) for the extension of eLDT manipulating Boolean combinations of *eDT* formulas, i.e. the queries $P, Q, \ldots$ of DB, equipped with the usual rules for negation, disjunction, and conjunction:

$$\begin{array}{ccc} \neg\text{-}l\,\dfrac{\Gamma \longrightarrow \Delta, Q}{\Gamma, \neg Q \longrightarrow \Delta} & \wedge\text{-}l\,\dfrac{\Gamma, P, Q \longrightarrow \Delta}{\Gamma, P \wedge Q \longrightarrow \Delta} & \vee\text{-}l\,\dfrac{\Gamma, P \longrightarrow \Delta \quad \Gamma, Q \longrightarrow \Delta}{\Gamma, P \vee Q \longrightarrow \Delta} \\ \neg\text{-}r\,\dfrac{\Gamma, Q \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \neg Q} & \vee\text{-}r\,\dfrac{\Gamma \longrightarrow \Delta, P, Q}{\Gamma \longrightarrow \Delta, P \vee Q} & \wedge\text{-}r\,\dfrac{\Gamma \longrightarrow \Delta, P \quad \Gamma \longrightarrow \Delta, Q}{\Gamma \longrightarrow \Delta, P \wedge Q} \end{array} \tag{4.1}$$

By the Boolean constructions of [BDK20, Section 5] (see also [BDK19]) we have:

Proposition 4.8. *If* LK(eLDT) *has a size N proof of an eDT sequent $\Gamma \longrightarrow \Delta$ over $\mathcal{E}$ then* eLDT *has a* poly(N) *size proof of $\Gamma \longrightarrow \Delta$ over some $\mathcal{E}' \supseteq \mathcal{E}$.*

The branching program construction for Boolean combinations is well-known, and the proof theoretic development of the result is similar to that of the 'positive decisions' we shall work with later in Section 5.1. Let us point out, in particular, that negation of *deterministic* branching programs is simple, by the recurrence: $\neg(ApB) \iff \neg Ap\neg B$. Naturally expressing negation is precisely the difficulty in the non-deterministic case.

In any case, from Proposition 4.8 above, we can readily establish the polynomial simulation of DB by eLDT:

Proof of Theorem 4.5. By Proposition 4.8 above, it suffices to work with $\mathsf{LK}(\mathsf{eLDT})$ instead of eLDT. We proceed by induction on the structure of a DB strategy from initial state $\{\Gamma \mapsto 1, \Delta \mapsto 0\}$, constructing an $\mathsf{LK}(\mathsf{eLDT})$ proof of $\Gamma \longrightarrow \Delta$ over $\mathcal{E}$. The construction is straightforward, similar to that of [PB94], simulating each query by a cut maintaining the invariant that queries answered 1 are on the LHS of the sequent and queries answered 0 are on the RHS of the sequent. Formally, if σ begins by querying some P, with substrategies σ_0, σ_1 for the answers $0, 1$ respectively, we construct the following $\mathsf{LK}(\mathsf{eLDT})$ proof,

$$\mathsf{cut}\ \dfrac{\overset{IH}{\triangledown}\ \Gamma \longrightarrow \Delta, P \qquad \overset{IH}{\triangledown}\ \Gamma, P \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}$$

where the subproofs marked *IH* are obtained by the inductive hypothesis, with σ_0, σ_1 construed as winning strategies from the corresponding intial states. Thus it remains to derive the base cases of simple contradictions by polynomial size proofs:[4]

- Boolean contradictions are given by the corresponding initial Boolean rule.
- Decision contradictions are given by constant length proofs using just the decision, structural and initial rules.[5] For instance $\{ApB \mapsto 1, p \mapsto 0, A \mapsto 0\}$ and $\{ApB \mapsto 0, p \mapsto 1, B \mapsto 1\}$ are respectively given by,

$$\dfrac{\mathsf{w}\dfrac{\mathsf{id}\dfrac{}{A \longrightarrow A}}{A \longrightarrow p, p, A} \quad \mathsf{w}\dfrac{\mathsf{id}\dfrac{}{p \longrightarrow p}}{p, B \longrightarrow p, A}}{ApB \longrightarrow p, A}\ p\text{-}l \qquad \text{and} \qquad \dfrac{\mathsf{w}\dfrac{\mathsf{id}\dfrac{}{p \longrightarrow p}}{p, B \longrightarrow A, p} \quad \mathsf{w}\dfrac{\mathsf{id}\dfrac{}{B \longrightarrow B}}{p, B, p \longrightarrow B}}{p, B \longrightarrow ApB}\ p\text{-}r \ .$$

- Boolean connective contradictions are the same as for LK, cf. [PB94].
- Extension contradictions are given by the corresponding extension axioms.
- Similarity contradictions are given by Proposition 3.3. □

Remark 4.9 (On tree-like proofs)**.** Note that, since strategies are trees, the proof constructed above is almost tree-like, except for the subproofs for similarity contradictions arising from Proposition 3.3. However the simulation of $\mathsf{LK}(\mathsf{eLDT})$ by eLDT, Proposition 4.8, does not preserve this tree-like structure.

[4] We omit consideration of weakenings immediately below an initial sequent in all cases.

[5] In fact, formally, constant size such proofs must exist by the previous completeness result, Proposition 2.10.

5. A non-uniform partial formalisation of Immerman-Szelepcsényi

In order to translate strategies to proofs, we need a way to simulate the *negation* of branching programs. For this, in the non-deterministic case, we need a non-uniform formalisation of the Immerman-Szelepcsényi theorem, $co\mathbf{NL} = \mathbf{NL}$. Throughout this section we focus only on eLNDT, not eLDT.

5.1. Working with positive decisions.

It will be convenient in our construction to stitch together branching programs by 'positive' decisions. This sort of decision induces what are often called 'monotone' or 'positive' (non-deterministic) branching programs (PBPs): NBPs where, for every 0-edge, there is a parallel 1-edge [GS92, Gri91], for instance as in Example 2.4 earlier. We investigated the proof complexity of PBPs in earlier work [DD22, DD25] essentially by restricting decision formulas to have the form $Ap(A \vee B)$, hence comprising a bona fide sublanguage of eNDT formulas. Here it will be convenient to 'bootstrap' our language with the capacity to make (certain) positive decisions on *compound* formulas (over a fixed set of extension variables), and even recursively so.

Definition 5.1 (Positive decisions)**.** Let $\mathcal{E} = \{e_i \leftrightarrow E_i\}_{i<n}$ be a set of extension axioms and $\mathbf{e} = \{e_i\}_{i<n}$. We introduce fresh extension variables by the closure property:

- if B is a formula over $\mathbf{e}$, and A, C are formulas, then $AB(A \vee C)$ is an extension variable.

Over this language we define $\mathcal{E}^+$ to be the smallest extension of $\mathcal{E}$ by all extension axioms of the form:

$$\begin{aligned} A0(A \vee C) &\leftrightarrow A \\ A1(A \vee C) &\leftrightarrow A \vee C \\ A(B_0 \vee B_1)(A \vee C) &\leftrightarrow (AB_0(A \vee C)) \vee (AB_1(A \vee C)) \\ A(B_0 p B_1)(A \vee C) &\leftrightarrow (AB_0(A \vee C)) p (AB_1(A \vee C)) \\ Ae_i(A \vee C) &\leftrightarrow AE_i(A \vee C) \end{aligned}$$

The notation we have used is intentionally suggestive, under the interpretation by $\mathcal{E}^+$, perhaps at the danger of ambiguity: we insist, for foundational reasons, that $AB(A \vee C)$ are *not* compound formulas, but rather bona fide extension variables. Note that the formulas A and C above may contain extension variables besides those in $\mathbf{e}$, in particular other positive decisions according to the closure property. For instance we have extension variables of the form $AB(A \vee (CB'(C \vee D)))$ and so on. However B and B' here must be over the original set of extension variables $\mathbf{e}$.

By the $\mathcal{E}$-induction principle, Remark 2.5, we may readily establish the following characteristic truth conditions:

Lemma 5.2 (Truth for positive decisions)**.** *Fix a set of extension axioms $\mathcal{E} = \{e_i \leftrightarrow E_i\}_{i<n}$. For formulas B over $\mathbf{e}$ there are polynomial size* eLNDT *proofs over $\mathcal{E}^+$ of:*

$$\begin{aligned} AB(A \vee C) &\rightarrow A, B \\ AB(A \vee C) &\rightarrow A, C \\ A &\rightarrow AB(A \vee C) \\ B, C &\rightarrow AB(A \vee C) \end{aligned}$$

Proof. The argument is a straightforward $\mathcal{E}$-induction on B. We will only present proofs for $B, C \rightarrow AB(A \vee C)$ as the other sequents follow similarly.

In the base case, when B is 0 or 1, we have:

$$\mathsf{w}\,\dfrac{0\text{-}l\,\dfrac{}{0 \to}}{0, C \to B} \qquad \mathsf{w}\,\dfrac{\mathsf{id}\,\dfrac{}{C \to C}}{1, C \to A, C}$$

If $B = D \vee E$ we have,

$$\mathcal{E}^+,\vee\text{-}r\,\dfrac{\vee\text{-}l\,\dfrac{IH\,\dfrac{}{D, C \to AD(A \vee C)} \quad IH\,\dfrac{}{E, C \to AE(A \vee C)}}{D \vee E, C \to AD(A \vee C), AE(A \vee C)}}{D \vee E, C \to A(D \vee E)(A \vee C)}$$

where the steps marked *IH* are obtained by the inductive hypothesis.

If $B = DpE$ we have the two proofs:

$$\mathcal{E}^+,p\text{-}r\,\dfrac{\mathsf{w,e}\,\dfrac{IH\,\dfrac{}{D, C \to AD(A \vee C)}}{C, D \to p, p, AD(A \vee C)} \quad \mathsf{w}\,\dfrac{\mathsf{id}\,\dfrac{}{p \to p}}{C, D, p \to p, AE(A \vee C)}}{C, D \to p, A(DpE)(A \vee C)}$$

$$\mathcal{E}^+,p\text{-}r\,\dfrac{\mathsf{w}\,\dfrac{\mathsf{id}\,\dfrac{}{p \to p}}{C, p, E \to p, AD(A \vee C)} \quad \mathsf{w,e}\,\dfrac{IH\,\dfrac{}{E, C \to AE(A \vee C)}}{C, p, E \to AE(A \vee C)}}{C, p, E \to A(DpE)(A \vee C)}$$

which we can combine to obtain, as required:

$$p\text{-}l\,\dfrac{C, D \to p, A(DpE)(A \vee C) \quad C, p, E \to A(DpE)(A \vee C)}{DpE, C \to A(DpE)(A \vee C)}$$

If B is an extension variable e_i with extension axiom $e_i \leftrightarrow E_i$, we just call the inductive hypothesis for $AE_i(A \vee C)$. □

Remark 5.3 (Positive decision rules)**.** Note that, from the Truth Conditions above, we can immediately derive the 'positive decision' rules as in [DD22, DD25], that we may use in the sequel:

$$B^+\text{-}l\,\dfrac{\Gamma, A \to \Delta \quad \Gamma, B, C \to \Delta}{\Gamma, AB(A \vee C) \to \Delta} \qquad B^+\text{-}r\,\dfrac{\Gamma \to \Delta, A, B \quad \Gamma \to \Delta, A, C}{\Gamma \to \Delta, AB(A \vee C)} \tag{5.1}$$

5.2. **Positive counting programs and their properties.** For the remainder of this section let us fix $\mathbf{B} = B_0, \ldots, B_{N-1}$ over a set of extension axioms $\mathcal{B}$. Introduce new (temporarily uninterpreted) extension variables $t_k^{\mathbf{B}^j}$ for each $\mathbf{B}^j = B_j, \ldots, B_{N-1}$, for $j \leq N$ (so $\mathbf{B}^N$ is just the empty list ε) and $k \in \mathbb{Z}$. After generating $\mathcal{B}^+$ as in Definition 5.1 above, we define $\mathcal{T}_{\mathcal{B}}^{\mathbf{B}}$ to be the smallest extension of $\mathcal{B}^+$ by all extension axioms of the form:

$$\begin{array}{ll} t_k^{\varepsilon} \leftrightarrow 1 & \text{for } k \leq 0 \\ t_k^{\varepsilon} \leftrightarrow 0 & \text{for } k > 0 \\ t_k^{\mathbf{B}^j} \leftrightarrow t_k^{\mathbf{B}^{j+1}} B_j (t_k^{\mathbf{B}^{j+1}} \vee t_{k-1}^{\mathbf{B}^{j+1}}) & \text{for } j < N \end{array}$$

It is not hard to see, by induction on $|\mathbf{B}^j|$, that $t_k^{\mathbf{B}^j}$ is true over $\mathcal{T}_{\mathcal{B}}^{\mathbf{B}}$ just when at least k of $\mathbf{B}^j$ are true. Note that we have extension variables with negative subscripts too, like t_{-1}^{ε}, which we have set to 1. It is convenient to admit these so that definitions and proofs by induction on the length of the superscript are more uniform (like the final case above).

Remark 5.4. Notice that the thresholds programs above are defined slightly differently from the ones from the earlier Example 2.4, in that the leaves left of the 1 sink in Fig. 2 are set to 1 rather than 0 by the above formulation. By monotonicity of the program, this makes no difference to the semantics, but the current formulation streamlines the exposition henceforth.

Similar 'threshold' programs were introduced in [DD22, DD25] where several basic properties were established in order to carry out counting arguments. As our formulation is slightly different, and for self-containment, we reproduce two necessary lemmata here, with reference to the analogous results from previous work.

Lemma 5.5 (Monotonicity of thresholds, cf. [DD25, Proposition 4.6])**.** *There are polynomial-size* eLNDT *proofs over* $\mathcal{T}_{\mathcal{B}}^{\mathbf{B}}$ *of:*

(1) $\longrightarrow t_k^{\mathbf{B}^j}$ *whenever* $k \leq 0$
(2) $t_k^{\mathbf{B}^j} \longrightarrow$ *whenever* $k > |\mathbf{B}^j|$
(3) $t_k^{\mathbf{B}^j} \longrightarrow t_{k-1}^{\mathbf{B}^j}$

Proof. All items are proved by induction on $|\mathbf{B}^j|$. The base cases, when $\mathbf{B}^j = \varepsilon$, are immediate from the extension axioms $\mathcal{T}_{\mathcal{B}}^{\mathbf{B}}$. We derive the inductive steps as follows,

$$(1): \quad \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\text{Lemma 5.2}\dfrac{\overset{IH}{\overline{\longrightarrow t_k^{\mathbf{B}^{j+1}}}}}{\longrightarrow t_k^{\mathbf{B}^{j+1}} B_j(t_k^{\mathbf{B}^{j+1}} \vee t_{k-1}^{\mathbf{B}^{j+1}})}}{\longrightarrow t_k^{\mathbf{B}^j}} \qquad (2): \quad \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\text{Lemma 5.2}\dfrac{\overset{IH}{\overline{t_k^{\mathbf{B}^{j+1}} \longrightarrow}} \quad \overset{IH}{\overline{t_{k-1}^{\mathbf{B}^{j+1}} \longrightarrow}}}{t_k^{\mathbf{B}^{j+1}} B_j(t_k^{\mathbf{B}^{j+1}} \vee t_{k-1}^{\mathbf{B}^{j+1}}) \longrightarrow}}{t_k^{\mathbf{B}^j} \longrightarrow}$$

$$(3): \quad \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\text{Lemma 5.2}\dfrac{\overset{IH}{\overline{t_k^{\mathbf{B}^{j+1}} \longrightarrow t_{k-1}^{\mathbf{B}^{j+1}}}} \quad \mathsf{id}\dfrac{}{t_{k-1}^{\mathbf{B}^{j+1}} \longrightarrow t_{k-1}^{\mathbf{B}^{j+1}}}}{t_k^{\mathbf{B}^{j+1}} B_j(t_k^{\mathbf{B}^{j+1}} \vee t_{k-1}^{\mathbf{B}^{j+1}}) \longrightarrow t_{k-1}^{\mathbf{B}^{j+1}} B_j(t_{k-1}^{\mathbf{B}^{j+1}} \vee t_{k-2}^{\mathbf{B}^{j+1}})}}{t_k^{\mathbf{B}^j} \longrightarrow t_{k-1}^{\mathbf{B}^j}}$$

where the steps marked *IH* are obtained by the inductive hypothesis. □

Lemma 5.6 (Truth for thresholds, cf. [DD25, Lemma 5.4])**.** *There are polynomial-size* eLNDT *proofs over* $\mathcal{T}_{\mathcal{B}}^{\mathbf{B}}$*, of:*

(1) $B_j, t_k^{\mathbf{B}^{j+1}} \longrightarrow t_{k+1}^{\mathbf{B}^j}$
(2) $t_k^{\mathbf{B}^{j+1}} \longrightarrow t_k^{\mathbf{B}^j}$
(3) $t_k^{\mathbf{B}^j} \longrightarrow t_k^{\mathbf{B}^{j+1}}, B_j$
(4) $t_k^{\mathbf{B}^j} \longrightarrow t_{k-1}^{\mathbf{B}^{j+1}}$

Proof. Almost all cases follow by simply unfolding a threshold extension variable by $\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}$ and applying the truth conditions for positive decisions:

$$(1): \mathcal{T}^{\mathbf{B}}_{\mathcal{B}}\ \frac{\overline{B_j, t_k^{\mathbf{B}^{j+1}} \longrightarrow t_k^{\mathbf{B}^{j+1}} B_j(t_k^{\mathbf{B}^{j+1}} \vee t_k^{\mathbf{B}^{j+1}})}^{\text{Lemma 5.2}}}{B_j, t_k^{\mathbf{B}^{j+1}} \longrightarrow t_{k+1}^{\mathbf{B}^{j}}} \qquad (2): \mathcal{T}^{\mathbf{B}}_{\mathcal{B}}\ \frac{\overline{t_k^{\mathbf{B}^{j+1}} \longrightarrow t_k^{\mathbf{B}^{j+1}} B_j(t_k^{\mathbf{B}^{j+1}} \vee t_{k-1}^{\mathbf{B}^{j+1}})}^{\text{Lemma 5.2}}}{t_k^{\mathbf{B}^{j+1}} \longrightarrow t_k^{\mathbf{B}^{j}}}$$

$$(3): \mathcal{T}^{\mathbf{B}}_{\mathcal{B}}\ \frac{\overline{t_k^{\mathbf{B}^{j+1}} B_j(t_k^{\mathbf{B}^{j+1}} \vee t_{k-1}^{\mathbf{B}^{j+1}}) \longrightarrow t_k^{\mathbf{B}^{j+1}}, B_j}^{\text{Lemma 5.2}}}{t_k^{\mathbf{B}^{j}} \longrightarrow t_k^{\mathbf{B}^{j+1}}, B_j} \qquad (4): \mathcal{T}^{\mathbf{B}}_{\mathcal{B}}\ \frac{\overline{t_k^{\mathbf{B}^{j+1}} B_j(t_k^{\mathbf{B}^{j+1}} \vee t_{k-1}^{\mathbf{B}^{j+1}}) \longrightarrow t_{k-1}^{\mathbf{B}^{j+1}}}^{\text{Lemma 5.2 \& Lemma 5.5}}}{t_k^{\mathbf{B}^{j}} \longrightarrow t_{k-1}^{\mathbf{B}^{j+1}}}$$

In the final case, (4), we are also using the monotonicity property $t_k^{\mathbf{B}^{j+1}} \longrightarrow t_{k-1}^{\mathbf{B}^{j+1}}$, by Lemma 5.5.(3). □

5.3. **Decider construction.** We shall employ the counting mechanisms from Section 5.2 to define a construction that will compute a decision on eNDT formulas relative to a fixed number of true formulas, over appropriate extension axioms. For this we shall employ ideas from the Immerman-Szelepcsényi Theorem, $co\mathbf{NL} = \mathbf{NL}$, in a nonuniform setting.

Recall that, from the beginning of Section 5.2, we have fixed formulas $\mathbf{B} = B_0, \ldots, B_{N-1}$ over a set of extension axioms $\mathcal{B}$, and we have generated $\mathcal{B}^+$ and $\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}$. The main result of this section is:

Theorem 5.7 (Formalised (partial) Immerman-Szelepcsényi)**.** *For $i < N, k \in \mathbb{Z}$ and formulas A, C there are extension variables AB_i^kC and a set of extension axioms extending $\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}$ over which there are polynomial-size* eLNDT *proofs of:*

$$\begin{aligned} t_k^{\mathbf{B}}, AB_i^kC &\longrightarrow A, B_i, t_{k+1}^{\mathbf{B}} \\ t_k^{\mathbf{B}}, AB_i^kC, B_i &\longrightarrow C, t_{k+1}^{\mathbf{B}} \\ t_k^{\mathbf{B}}, B_i, C &\longrightarrow AB_i^kC, t_{k+1}^{\mathbf{B}} \\ t_k^{\mathbf{B}}, A &\longrightarrow B_i, AB_i^kC, t_{k+1}^{\mathbf{B}} \end{aligned} \tag{5.2}$$

The idea behind the sequents above is that, as long as *exactly* k of $\mathbf{B}$ are true, AB_i^kC accurately computes a decision on B_i, returning the value of A if false, and C if true. Indeed, ignoring the gray threshold formulas, the sequents above are just the truth conditions for a decision 'AB_iC'. Note here that having $t_{k+1}^{\mathbf{B}}$ on the RHS is semantically equivalent to having its negation on the LHS, and of course $t_k^{\mathbf{B}} \wedge \neg t_{k+1}^{\mathbf{B}}$ holds just when exactly k of $\mathbf{B}$ are true. The notation AB_i^kC we have used is suggestive but we emphasise that these expressions are *not* compound formulas, but rather bona fide extension variables. We will call AB_i^kC the **k-decider** of B_i (for A, C, over $\mathbf{B}$).

To give the intuition before its formal definition, the corresponding NBP that will be computed by AB_i^kC is visualised (over its corresponding extension axioms) in Fig. 7 as a NBP, in fact as a positive combination of the programs $\mathbf{B}$ we started with. Note here that each 'node' is actually a copy of an NBP B_j, indexed by a pair (c, b). c is a counter that tracks how many true B_js along the path thusfar, while b is a Boolean flag that flips from 0 to 1 if B_i is true along the path. The blue part of the diagram is all before deciding B_i, after which the program splits into two cases, indicated in red ($b = 0$) or green ($b = 1$), depending on whether the 0-direction or 1-direction, respectively, from B_i was followed. The orange 0 leaves correspond to paths where the counter does not match the number of true variables

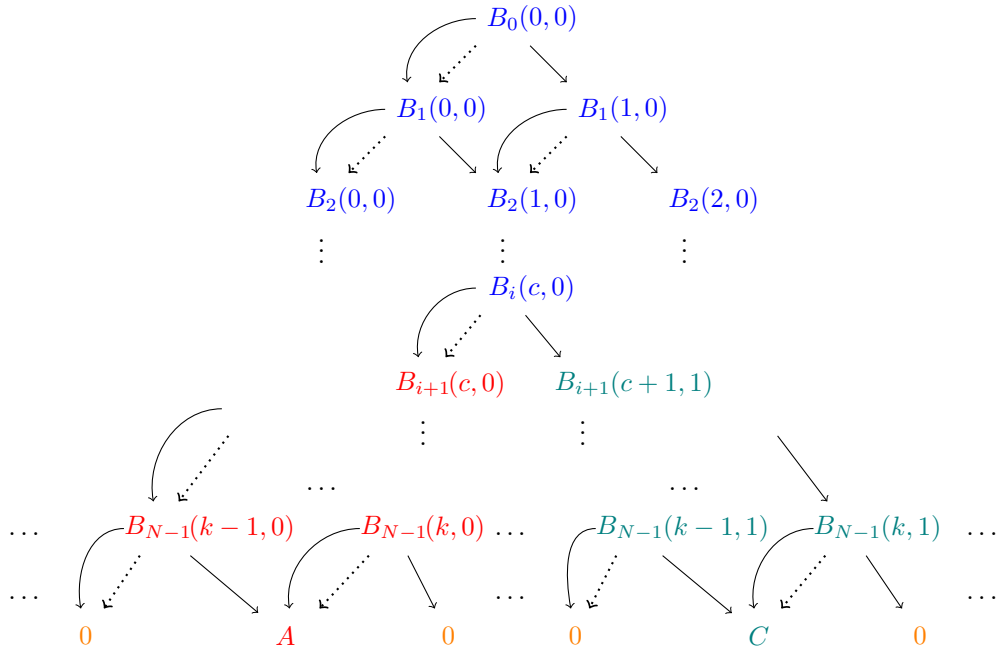


Figure 7: The k-decider AB_i^kC, over $\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}$, visualised as an NBP that is a positive combination of $\mathbf{B}$. 0-edges are dotted and 1-edges are solid.

k, and so the program returns erroneously. The extension variables used to describe this program will duly have three indices c, b, j.

Let us now define the k-decider AB_i^kC more formally and prove its relevant properties. For the remainder of this section we fix $i < N$, $k \in \mathbb{Z}$ and formulas A, C, towards proving Theorem 5.7.

Definition 5.8 (Decision programs)**.** We introduce extension variables $d_j^{c,b}$, for $j \leq N, c \leq N, b < 2$, and write $\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}$ for the smallest extension of $\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}$ by the following extension axioms, given by case analysis on $j \leq N$:

- For $j < N, j \neq i$:
$$d_j^{c,b} \leftrightarrow d_{j+1}^{c,b} B_j (d_{j+1}^{c,b} \vee d_{j+1}^{c+1,b})$$
- For $j = i$:
$$d_i^{c,0} \leftrightarrow d_{i+1}^{c,0} B_i (d_{i+1}^{c,0} \vee d_{i+1}^{c+1,1})$$
- For $j = N$:
$$\begin{aligned} d_N^{k,0} &\leftrightarrow A \\ d_N^{k,1} &\leftrightarrow C \\ d_N^{c,b} &\leftrightarrow 0 \quad \text{if } c \neq k \end{aligned}$$

From here we set $AB_i^kC := d_0^{0,0}$.

Remark 5.9 (Relation to [AGP02] and Immerman-Szelepcsényi)**.** The formulas $1B_i^k0$, by Theorem 5.7, will represent the negation of B_i in environments where exactly k of $\mathbf{B}$ are true. Formally, in the terminology of [AGP02] they are *pseudocomplements* with respect to the

'exact-k' functions. However our construction of these pseudocomplements is rather inspired by the inductive counting argument underlying the Immerman-Szelepcsényi theorem:

- Each 'node' non-deterministically checks whether some B_i is true, i.e. whether its source reaches the sink 1, similarly to the subroutine of Immerman-Szelepcsényi. The role of the counter c is similar to the local counter of the subroutine, with the computation aborting (the orange 0s of Fig. 7) if the wrong number of B_j's were guessed true.
- Instead of a main routine that inductively calculates the number of true B_j's (i.e. the number of B_j's whose sources eventually reach 1), an exact number k of true B_j's is given to us in the context of the corresponding truth conditions, cf. Eq. (5.2). The role of the main routine is rather carried out by a counting argument at the level of the proof system, essentially a (non-uniform) induction on k.

5.4. **Some intermediate results.** In order to prove Theorem 5.7, we first establish a series of intermediate lemmata, characterising evaluation at each node of the decider. Each result will rely on the previous ones, working from the leaves of the k-decider upwards, cf. Fig. 7.

All proofs in this subsection are in the system eLNDT, over extension variables $\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}$ (which also includes $\mathcal{B}^+$ and $\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}$). Recall that we fixed $i < N$, $k \in \mathbb{Z}$ and formulas A, C in the previous subsection.

Intuitively, at each node $d^{c,b}_j$, the formulas $B_0, \dots, B_{j-1}$ have already been 'evaluated', and the counter c tells us that (at most) c of them have returned true. How the remainder of the program evaluates now depends on the number l of true formulas among $\mathbf{B}^j = B_j, \dots, B_{N-1}$. If l, c are too low for the counter to eventually reach k then the k-decider will return erroneously (the orange 0 leaves in Fig. 7). This intuition is formalised by the following result:

Lemma 5.10. *If $l + c < k$ there are polynomial-size proofs of $t^{\mathbf{B}^j}_l, d^{c,b}_j \longrightarrow t^{\mathbf{B}^j}_{l+1}$, for $j \leq N$.*

Proof. By backwards induction on $j \leq N$:

- For the base case, $j = N$, we have $\mathbf{B}^N = \varepsilon$ so:
  - if $c \neq k$ we have $d^{c,b}_N \leftrightarrow 0$ by $\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}$;
  - if $c = k$ then $l < 0$ so $t^{\varepsilon}_{l+1} \leftrightarrow 1$ by $\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}$.

  In both cases we may conclude by an initial step 0-l or 1-r and weakenings.
- For the inductive step we have the following derivation, conducting a sort of case analysis on B_j:

$$\dfrac{\dfrac{\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}\dfrac{IH\dfrac{}{t^{\mathbf{B}^j}_l, d^{c,b}_{j+1} \longrightarrow t^{\mathbf{B}^j}_{l+1}}}{t^{\mathbf{B}^j}_l, d^{c,b}_j \longrightarrow t^{\mathbf{B}^j}_{l+1}, B_j} \quad \mathcal{D}^{\mathbf{B}}_{\mathcal{B}}\dfrac{IH\dfrac{}{t^{\mathbf{B}^j}_l, d^{c,b}_{j+1} \longrightarrow t^{\mathbf{B}^j}_{l+1}} \quad \mathcal{T}^{\mathbf{B}}_{\mathcal{B}}\dfrac{IH\dfrac{}{t^{\mathbf{B}^{j+1}}_{l-1}, d^{c+1,b'}_{j+1} \longrightarrow t^{\mathbf{B}^{j+1}}_{l}}}{t^{\mathbf{B}^j}_l, B_j, d^{c+1,b'}_{j+1} \longrightarrow t^{\mathbf{B}^j}_{l+1}}}{B_j, t^{\mathbf{B}^j}_l, d^{c,b}_j \longrightarrow t^{\mathbf{B}^j}_{l+1}}}{\mathsf{cut}}}{t^{\mathbf{B}^j}_l, d^{c,b}_j \longrightarrow t^{\mathbf{B}^j}_{l+1}}$$

  for appropriate b', where the steps marked *IH* are obtained by the inductive hypothesis, steps marked $\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}$ are obtained by positive truth conditions from Lemma 5.2 under the corresponding extension axioms from Definition 5.8, and steps marked $\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}$ are obtained by the truth conditions for thresholds from Lemma 5.6. □

eLNDT proofs in the remainder of this subsection will have a similar structure to the one above, and we shall duly annotate them similarly without further clarification.

Continuing our informal intuition, if $l + c = k$ then the k-decider will return correctly (i.e. it will correctly decide B_i, returning A if false and C if true). If $j > i$ then $d_j^{c,b}$ has already evaluated B_i and the latter's truth value is stored as b. Thus such nodes will return A or C depending on the value of b. This intuition is formalised by the following result:

Lemma 5.11. *If $l + c = k$ then, for $i < j \leq N$, there are polynomial-size proofs of:*

(1) $t_l^{\mathbf{B}^j}, d_j^{c,0} \longrightarrow A, t_{l+1}^{\mathbf{B}^j}$
(2) $t_l^{\mathbf{B}^j}, A \longrightarrow d_j^{c,0}, t_{l+1}^{\mathbf{B}^j}$
(3) $t_l^{\mathbf{B}^j}, d_j^{c,1} \longrightarrow C, t_{l+1}^{\mathbf{B}^j}$
(4) $t_l^{\mathbf{B}^j}, C \longrightarrow d_j^{c,1}, t_{l+1}^{\mathbf{B}^j}$

Proof. We prove only Items 1 and 2, with Items 3 and 4 respectively proved similarly. We proceed by backwards induction on j.

For the base case, $j = N$, note that $\mathbf{B}^N = \varepsilon$. We have:

(1) By the $\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}$ axioms, we have either $d_N^{c,0} \leftrightarrow 0$ or $d_N^{c,0} \leftrightarrow A$, so we can conclude by id or 0-l and weakenings.
(2)
- If $l < 0$ then we have $t_{l+1}^{\varepsilon} \leftrightarrow 1$ from $\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}$, and so we conclude by 1-r and weakenings.
- If $l > 0$ then we have $t_l^{\varepsilon} \leftrightarrow 0$ from $\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}$, and so we conclude by 0-l and weakenings.
- If $l = 0$ then $k = c$ by assumption, so from $\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}$ we have $d_N^{k,0} \leftrightarrow A$, whence we conclude by id and weakening steps.

For the inductive steps we have the following derivations:

$$(1): \quad \dfrac{\dfrac{\dfrac{IH\ \dfrac{}{t_l^{\mathbf{B}^{j+1}}, d_{j+1}^{c,0} \longrightarrow A, t_{l+1}^{\mathbf{B}^{j+1}}}}{\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}\ \dfrac{t_l^{\mathbf{B}^j}, d_{j+1}^{c,0} \longrightarrow A, t_{l+1}^{\mathbf{B}^j}, B_j}{\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}\ \dfrac{t_l^{\mathbf{B}^j}, d_j^{c,0} \longrightarrow A, t_{l+1}^{\mathbf{B}^j}, B_j}{}}}}{\mathsf{cut}\ \dfrac{}{t_l^{\mathbf{B}^j}, d_j^{c,0} \longrightarrow A, t_{l+1}^{\mathbf{B}^j}}} \quad \dfrac{\dfrac{\dfrac{\text{Lemma 5.10}}{t_{l-1}^{\mathbf{B}^{j+1}}, d_{j+1}^{c,0} \longrightarrow A, t_l^{\mathbf{B}^{j+1}}}}{\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}\ \dfrac{B_j, t_l^{\mathbf{B}^j}, d_{j+1}^{c,0} \longrightarrow A, t_{l+1}^{\mathbf{B}^j}}{}}}{\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}\ \dfrac{}{B_j, t_l^{\mathbf{B}^j}, d_j^{c,0} \longrightarrow A, t_{l+1}^{\mathbf{B}^j}}} \quad \dfrac{\dfrac{IH\ \dfrac{}{t_{l-1}^{\mathbf{B}^{j+1}}, d_{j+1}^{c+1,0} \longrightarrow A, t_l^{\mathbf{B}^{j+1}}}}{\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}\ \dfrac{}{B_j, t_l^{\mathbf{B}^j}, d_{j+1}^{c+1,0} \longrightarrow A, t_{l+1}^{\mathbf{B}^j}}}}{\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}\ \dfrac{}{B_j, t_l^{\mathbf{B}^j}, d_j^{c,0} \longrightarrow A, t_{l+1}^{\mathbf{B}^j}}}$$

$$(2): \quad \dfrac{\dfrac{\dfrac{IH\ \dfrac{}{t_l^{\mathbf{B}^{j+1}}, A \longrightarrow d_{j+1}^{c,0}, t_{l+1}^{\mathbf{B}^{j+1}}}}{\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}\ \dfrac{t_l^{\mathbf{B}^j}, A \longrightarrow d_{j+1}^{c,0}, t_{l+1}^{\mathbf{B}^j}, B_j}{\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}\ \dfrac{t_l^{\mathbf{B}^j}, A \longrightarrow d_j^{c,0}, t_{l+1}^{\mathbf{B}^j}, B_j}{}}}}{\mathsf{cut}\ \dfrac{}{t_l^{\mathbf{B}^j}, A \longrightarrow d_j^{c,0}, t_{l+1}^{\mathbf{B}^j}}} \quad \dfrac{\dfrac{IH\ \dfrac{}{t_{l-1}^{\mathbf{B}^{j+1}}, A \longrightarrow d_{j+1}^{c+1,0}, t_l^{\mathbf{B}^{j+1}}}}{\mathcal{T}^{\mathbf{B}}_{\mathcal{B}}\ \dfrac{}{B_j, t_l^{\mathbf{B}^j}, A \longrightarrow d_{j+1}^{c+1,0}, t_{l+1}^{\mathbf{B}^j}}}}{\mathcal{D}^{\mathbf{B}}_{\mathcal{B}}\ \dfrac{}{B_j, t_l^{\mathbf{B}^j}, A \longrightarrow d_j^{c,0}, t_{l+1}^{\mathbf{B}^j}}}$$

□

Finally we consider the case when $l + c = k$ and $j \leq i$. The intuition here is that B_i has not yet been evaluated, so the k-decider behaves like a bona fide decision on B_i, returning A if false and C if true. This intuition is formalised by the following result:

Lemma 5.12. *If $l + c = k$ then, for $j \leq i$, there are polynomial-size proofs of:*

(1) $t_l^{\mathbf{B}^j}, d_j^{c,0} \longrightarrow A, B_i, t_{l+1}^{\mathbf{B}^j}$
(2) $t_l^{\mathbf{B}^j}, A \longrightarrow B_i, d_j^{c,0}, t_{l+1}^{\mathbf{B}^j}$
(3) $t_l^{\mathbf{B}^j}, d_j^{c,0}, B_i \longrightarrow C, t_{l+1}^{\mathbf{B}^j}$

(4) $t_l^{\mathbf{B}^j}, B_i, C \longrightarrow d_j^{c,0}, t_{l+1}^{\mathbf{B}^j}$

Proof. We proceed by backwards induction on $j \leq i$.

For the base case, $j = i$, Items 1 and 2 are proved similarly:

$$(1): \quad \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\dfrac{}{t_l^{\mathbf{B}^{i+1}}, d_{i+1}^{c,0} \longrightarrow A, t_{l+1}^{\mathbf{B}^{i+1}}}}{\mathcal{D}_{\mathcal{B}}^{\mathbf{B}} \dfrac{t_l^{\mathbf{B}^i}, d_{i+1}^{c,0} \longrightarrow A, B_i, t_{l+1}^{\mathbf{B}^i}}{t_l^{\mathbf{B}^i}, d_i^{c,0} \longrightarrow A, B_i, t_{l+1}^{\mathbf{B}^i}}} \qquad (2): \quad \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\dfrac{}{t_l^{\mathbf{B}^{i+1}}, A \longrightarrow d_{i+1}^{c,0}, t_{l+1}^{\mathbf{B}^{i+1}}}}{\mathcal{D}_{\mathcal{B}}^{\mathbf{B}} \dfrac{t_l^{\mathbf{B}^i}, A \longrightarrow B_i, d_{i+1}^{c,0}, t_{l+1}^{\mathbf{B}^i}}{t_l^{\mathbf{B}^i}, A \longrightarrow B_i, d_i^{c,0}, t_{l+1}^{\mathbf{B}^i}}}$$

Items 3 and 4 are proved dually, accounting for the choice when B_i is true:

$$(3): \quad \dfrac{\mathsf{w}\dfrac{\mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\mathcal{D}_{\mathcal{B}}^{\mathbf{B}} \dfrac{}{t_{l-1}^{\mathbf{B}^{i+1}}, d_{i+1}^{c,0} \longrightarrow t_l^{\mathbf{B}^{i+1}}}{t_{l-1}^{\mathbf{B}^{i+1}}, d_{i+1}^{c,0} \longrightarrow C, t_l^{\mathbf{B}^{i+1}}}}{t_l^{\mathbf{B}^i}, d_{i+1}^{c,0}, B_i \longrightarrow C, t_{l+1}^{\mathbf{B}^i}} \quad \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\dfrac{}{t_{l-1}^{\mathbf{B}^{i+1}}, d_{i+1}^{c+1,1}, B_i \longrightarrow C, t_l^{\mathbf{B}^{i+1}}}}{t_l^{\mathbf{B}^i}, d_{i+1}^{c+1,1}, B_i \longrightarrow C, t_{l+1}^{\mathbf{B}^i}}}{t_l^{\mathbf{B}^i}, d_i^{c,0}, B_i \longrightarrow C, t_{l+1}^{\mathbf{B}^i}}$$

$$(4): \quad \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\mathcal{D}_{\mathcal{B}}^{\mathbf{B}} \dfrac{}{t_{l-1}^{\mathbf{B}^{i+1}}, C \longrightarrow d_{i+1}^{c+1,1}, t_l^{\mathbf{B}^{i+1}}}}{t_l^{\mathbf{B}^i}, B_i, C \longrightarrow d_i^{c,0}, t_{l+1}^{\mathbf{B}^i}}$$

For the inductive step we have the following derivations:

$$(1): \quad \dfrac{\mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\mathcal{D}_{\mathcal{B}}^{\mathbf{B}} \dfrac{IH\dfrac{}{t_l^{\mathbf{B}^{j+1}}, d_{j+1}^{c,0} \longrightarrow A, B_i, t_{l+1}^{\mathbf{B}^{j+1}}}}{t_l^{\mathbf{B}^j}, d_{j+1}^{c,0} \longrightarrow A, B_i, t_{l+1}^{\mathbf{B}^j}, B_j}}{t_l^{\mathbf{B}^j}, d_j^{c,0} \longrightarrow A, B_i, t_{l+1}^{\mathbf{B}^j}, B_j} \quad \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\mathcal{D}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\mathsf{w}\dfrac{\mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{}{t_{l-1}^{\mathbf{B}^{j+1}}, d_{j+1}^{c,0} \longrightarrow t_l^{\mathbf{B}^{j+1}}}}{t_{l-1}^{\mathbf{B}^{j+1}}, d_{j+1}^{c,0} \longrightarrow A, B_i, t_l^{\mathbf{B}^{j+1}}}}{B_j, t_l^{\mathbf{B}^j}, d_{j+1}^{c,0} \longrightarrow A, B_i, t_{l+1}^{\mathbf{B}^j}} \quad \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{IH\dfrac{}{t_{l-1}^{\mathbf{B}^{j+1}}, d_{j+1}^{c+1,0} \longrightarrow A, B_i, t_l^{\mathbf{B}^{j+1}}}}{B_j, t_l^{\mathbf{B}^j}, d_{j+1}^{c+1,0} \longrightarrow A, B_i, t_{l+1}^{\mathbf{B}^j}}}{B_j, t_l^{\mathbf{B}^j}, d_j^{c,0} \longrightarrow A, B_i, t_{l+1}^{\mathbf{B}^j}}}{t_l^{\mathbf{B}^j}, d_j^{c,0} \longrightarrow A, B_i, t_{l+1}^{\mathbf{B}^j}}$$

$$(2): \quad \dfrac{\mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{\mathcal{D}_{\mathcal{B}}^{\mathbf{B}} \dfrac{IH\dfrac{}{t_l^{\mathbf{B}^{j+1}}, A \longrightarrow B_i, d_{j+1}^{c,0}, t_{l+1}^{\mathbf{B}^{j+1}}}}{t_l^{\mathbf{B}^j}, A \longrightarrow B_i, d_{j+1}^{c,0}, t_{l+1}^{\mathbf{B}^j}, B_j}}{t_l^{\mathbf{B}^j}, A \longrightarrow B_i, d_j^{c,0}, t_{l+1}^{\mathbf{B}^j}, B_j} \quad \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \dfrac{IH\dfrac{}{t_l^{\mathbf{B}^j}, A \longrightarrow B_i, d_{j+1}^{c+1,0}, t_{l+1}^{\mathbf{B}^j}}}{\mathcal{D}_{\mathcal{B}}^{\mathbf{B}} \dfrac{B_j, t_{l-1}^{\mathbf{B}^{j+1}}, A \longrightarrow B_i, d_{j+1}^{c+1,0}, t_l^{\mathbf{B}^{j+1}}}{B_j, t_l^{\mathbf{B}^j}, A \longrightarrow B_i, d_j^{c,0}, t_{l+1}^{\mathbf{B}^j}}}}{t_l^{\mathbf{B}^j}, A \longrightarrow B_i, d_j^{c,0}, t_{l+1}^{\mathbf{B}^j}}$$

The derivations for Items 3 and 4 are similar. □

5.5. **Putting it all together.** From here the main result of this section is an immediate consequence of our final lemma:

Proof of Theorem 5.7. As $AB_i^k C := d_0^{0,0}$, the result is now just a special case of Lemma 5.12, setting $c = 0$, $l = k$ and $j = 0$. □

6. Strategies to proofs: the non-deterministic case

Our final main result is the converse of Theorem 4.1, in the non-deterministic case:

Theorem 6.1. *If* NB *has a size N proof of an eNDT sequent $\Gamma \longrightarrow \Delta$ over $\mathcal{E}$ then* eLNDT *has a* poly(N) *size proof of $\Gamma \longrightarrow \Delta$ over some $\mathcal{E}' \supseteq \mathcal{E}$.*

Corollary 6.2. eLNDT *polynomially simulates* NB*, over NDT sequents.*

This argument is much more complicated than the deterministic case, Theorem 4.5, and requires the formalisation of Immerman-Szelepcsényi from the previous section. The argument follows by composing a series of intermediate polynomial simulations.

6.1. De Morgan normal form of strategies.

Before translating strategies to proofs, it will be useful to work with strategies in 'De Morgan' form, where each query only has negations in front of eLNDT formulas, with no $\wedge$ in its scope. Let us write $\mathsf{NB}^{\mathrm{DM}}$ for the restriction of NB to this syntax of queries. We can define for each NB-query Q an 'equivalent' $\mathsf{NB}^{\mathrm{DM}}$-query Q^{DM} in the natural way, by pushing negations down to eNDT formulas.

Definition 6.3 (DM-translation)**.** For each NB-query Q define a $\mathsf{NB}^{\mathrm{DM}}$-query Q^{DM} by,

$$\begin{array}{rl} B^{\mathrm{DM}} & := B \\ (Q \wedge R)^{\mathrm{DM}} & := Q^{\mathrm{DM}} \wedge R^{\mathrm{DM}} \\ (Q \vee R)^{\mathrm{DM}} & := Q^{\mathrm{DM}} \vee R^{\mathrm{DM}} \end{array} \qquad \begin{array}{rl} (\neg B)^{\mathrm{DM}} & := \neg B \\ (\neg\neg Q)^{\mathrm{DM}} & := Q^{\mathrm{DM}} \\ (\neg(Q \vee R))^{\mathrm{DM}} & := (\neg Q)^{\mathrm{DM}} \wedge (\neg R)^{\mathrm{DM}} \\ (\neg(Q \wedge R))^{\mathrm{DM}} & := (\neg Q)^{\mathrm{DM}} \vee (\neg R)^{\mathrm{DM}} \end{array}$$

where B is an eNDT formula.

We will need to lift this translation to strategies too for which, we require some intermediate results.

Let us write $\mathrm{dp}(Q)$ for the **depth** of Q, i.e. the maximum length of a path from the root of Q to a (maximal) eNDT subformula. Our first intermediate result constructs a winning strategy from situations when the assignment is inconsistent with some substitution of queries.

Lemma 6.4 (Leibniz property)**.** *Let $P, P', Q[X]$ be* NB *(or* $\mathsf{NB}^{\mathrm{DM}}$*) queries, where X is a distinguished single subquery occurrence in $Q[X]$. From $\{P \mapsto b, P' \mapsto b, Q[P] \mapsto c, Q[P'] \mapsto 1-c\}$ there is a* NB *(resp.,* $\mathsf{NB}^{\mathrm{DM}}$*) winning strategy with $O(\log(\mathrm{dp}(Q[X])))$ rounds.*

Proof. Let π be the (unique) path in the of $Q[X]$ from the root to X. We proceed by divide-and-conquer induction on the length $|\pi|$ of π.

- If $|\pi| = 0$ then $Q[X] = X$, and $\{P \mapsto b, P' \mapsto b, P \mapsto c, P' \mapsto 1-c\}$ always contains a simple contradiction.
- If $|\pi| = 1$ then X is an immediate subquery of Q:
  - if $Q[X] = \neg X$ then $\{P \mapsto b, P' \mapsto b, \neg P \mapsto c, \neg P' \mapsto 1-c\}$ always contains a simple contradiction;
  - if $Q[X] = X \circ T$ for some $\circ \in \{\vee, \wedge\}$ and query T, then ask T. From here note that $\{P \mapsto b, P' \mapsto b, P \circ T \mapsto c, P' \circ T \mapsto 1-c, T \mapsto d\}$ always contains a simple contradiction. Similarly if $Q[X] = T \circ X$.
- For the inductive step, write $Q[X] = Q'[R[X]]$ where $R[X]$ is the subquery of $Q[X]$ rooted (roughly) halfway along π. Ask $R[P]$ and $R[P']$, with responses d and d' respectively:
  - if $d \neq d'$ then apply the inductive hypothesis to $P, P', R[X]$;

– if $d=d'$ then apply the inductive hypothesis to $R[P],R[P'],Q'[X]$. □

For a tree T write $|T|$ for its number of leaves. Recall 'Spira's lemma', that each binary tree has a subtree of roughly half the number of leaves:

Lemma 6.5 (Spira's lemma, [Spi71])**.** *For any binary tree T there is a subtree T' such that $\frac{1}{3}|T|\leq|T'|<\frac{2}{3}|T|$.*

We can use the above fact to drive a divide-and-conquer strategy for De Morgan duality. Let us write $\#Q$ for the number of (maximal) eNDT subformula occurrences in Q, i.e. writing $Q=\varphi(\mathbf{B})$ where $\varphi(\mathbf{x})$ is a Boolean formula and $\mathbf{B}$ are maximal eNDT subformulas, $\#Q$ is just the number of leaves of the formula tree of $\varphi(\mathbf{x})$.

Lemma 6.6 (Duality)**.** *Given a NB-query Q there is a winning strategy in $\mathsf{NB}^{\mathrm{DM}}$ from $\{Q^{\mathrm{DM}}\mapsto b,(\neg Q)^{\mathrm{DM}}\mapsto b\}$ with $O(\log(\#Q)$ rounds.*

Proof. We proceed by divide-and-conquer induction on $\#Q$:

- If Q is just an eNDT-formula or its negation, then the statement is immediate.
- For the inductive step, write $Q=P[R]$ where R is a subquery of Q of roughly half the size, more precisely s.t. $\frac{1}{3}\#Q\leq\#R\leq\frac{2}{3}\#Q$, by Lemma 6.5. Ask R^{DM} and $(\neg R)^{\mathrm{DM}}$, with responses c and c' respectively:
  - if $c=c'$ then apply the inductive hypothesis to R;
  - if $c\neq c'$ then ask $(P[c])^{\mathrm{DM}}$ and $(\neg P[c])^{\mathrm{DM}}$, with responses d and d' respectively:
    - if $d\neq b$ then apply Lemma 6.4 to $R^{\mathrm{DM}},c,(P[X])^{\mathrm{DM}}$;[6]
    - if $d'\neq b$ then apply Lemma 6.4 to $(\neg R)^{\mathrm{DM}},c',(\neg P[X])^{\mathrm{DM}}$;
    - otherwise $d=d'=b$ so apply the inductive hypothesis to $P[c]$. □

Finally we can obtain the translation of NB strategies to De Morgan form. For a NB strategy σ, write $\#\sigma:=\sum_{Q\in\sigma}\#Q$, where $Q\in\sigma$ means that Q varies over nodes labelled by the query Q in σ.

Proposition 6.7. *Given a NB strategy σ over $\mathcal{E}$ from an eNDT sequent winning in d rounds there is a $\mathsf{NB}^{\mathrm{DM}}$ strategy over $\mathcal{E}$ for the same sequent winning in $O(d+\log(\#\sigma))$ rounds.*

Proof. From σ construct a $\mathsf{NB}^{\mathrm{DM}}$ 'pre-strategy' σ^{DM} for the same sequent by replacing each query Q by Q^{DM} (σ^{DM} is visualised in Fig. 8). The only paths of σ^{DM} that no longer end after a simple contradiction nonetheless contain a subset of queries of form $\{Q^{\mathrm{DM}}\mapsto b,(\neg Q)^{\mathrm{DM}}\mapsto b\}$. So we may extend σ^{DM} into a bona fide $\mathsf{NB}^{\mathrm{DM}}$ winning strategy by appealing to Lemma 6.6. □

6.2. **From De Morgan strategies to $\mathsf{LK}^+(\mathsf{eLNDT})$.** We now use our formalisation of Immerman-Szelepcsényi in order to translate De Morgan strategies to proofs without negation. Similarly to the deterministic case, cf. Section 4.3, it will be useful to first translate to a version of eLNDT with formulas closed under certain Boolean combinations, this time only positively so:

Definition 6.8 (System for positive Boolean combinations of eNDT formulas)**.** Write $\mathsf{LK}^+(\mathsf{eLNDT})$ for the extension of eLNDT that allows $\{\vee,\wedge\}$-combinations of eNDT formulas and the corresponding rules for these connectives from Eq. (4.1).

[6] Note that $Q^{\mathrm{DM}}=(P[R])^{\mathrm{DM}}=P^{\mathrm{DM}}[R^{\mathrm{DM}}]$.

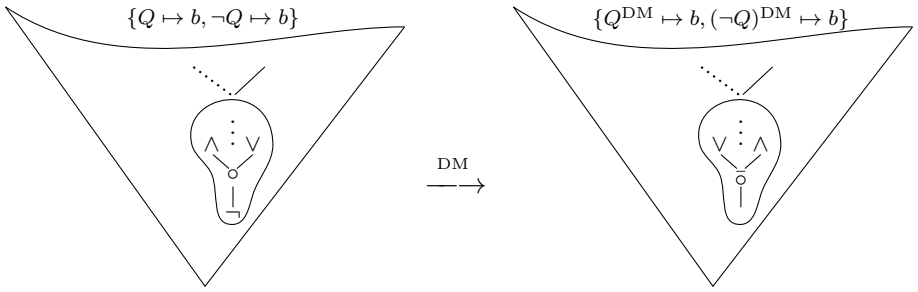


Figure 8: Visualisation of a NB strategy (left) and its $\cdot^{\mathrm{DM}}$ translation (right). 'Leaves' of the form $\{Q^{\mathrm{DM}} \mapsto b, (\neg Q)^{\mathrm{DM}} \mapsto b\}$ are not simple contradictions of $\mathsf{NB}^{\mathrm{DM}}$, but are rather justified by the $O(\log(\#Q))$-depth strategies from Lemma 6.6.

The main result of this subsection is:

Proposition 6.9. *Given a* $\mathsf{NB}^{\mathrm{DM}}$ *proof* σ *over* $\mathcal{B}$ *of an eNDT sequent* $\Gamma \longrightarrow \Delta$ *there is a* $\mathsf{LK}^+(\mathsf{eLNDT})$ *proof of* $\Gamma \longrightarrow \Delta$ *over some* $\mathcal{B}' \supseteq \mathcal{B}$ *of size polynomial in* $|\sigma|$.

Towards a proof of this result, let us fix for the remainder of this subsection $\sigma, \mathcal{B}, \Gamma, \Delta$ as declared in the statement above, and let $\mathbf{B} = B_0, \ldots, B_{N-1}$ enumerate all the eNDT formulas occurring in σ. By Theorem 5.7 let $\mathcal{D}_i^k \supseteq \mathcal{T}_{\mathcal{B}}^{\mathbf{B}} \supseteq \mathcal{B}^+ \supseteq \mathcal{B}$ be such that we have polynomial-size eLNDT proofs of the sequents in Eq. (5.2) over $\mathcal{D}_i^k$, for $A = 1$ and $C = 0$. Using those sequents, we can simulate negated eNDT formulas of σ in $\mathsf{LK}^+(\mathsf{eLNDT})$ relative to a fixed number of true formulas. Formally, writing $\mathcal{D}^k := \mathcal{D}_0^k \cup \cdots \cup \mathcal{D}_{N-1}^k$, we have:

Lemma 6.10. *For each* $k \in \mathbb{Z}$ *there are* $\mathsf{LK}^+(\mathsf{eLNDT})$ *proofs over* $\mathcal{D}^k$ *of* $t_k^{\mathbf{B}}, \Gamma \longrightarrow \Delta, t_{k+1}^{\mathbf{B}}$ *of size polynomial in* $|\sigma|$.

Proof. For each query Q of σ write Q^k for the result of replacing each eNDT subquery $\neg B_i$ by the respective decider $1B_i^k0$. We proceed inductively on the structure of σ, similarly to the deterministic case, cf. Proposition 4.8, instead conducting cuts on Q^k for each query Q, always maintaining the invariant that if $Q \mapsto 0$ then Q^k is on the RHS of a sequent, and if $Q \mapsto 1$ then Q^k is on the LHS. We also always keep $t_k^{\mathbf{B}}$ on the LHS and $t_{k+1}^{\mathbf{B}}$ on the RHS. Formally, if σ begins by querying some Q, with substrategies σ_0, σ_1 for the answers $0,1$ respectively, we construct the following $\mathsf{LK}^+(\mathsf{eLNDT})$ proof:

$$\mathsf{cut}\ \frac{\overset{IH}{\triangledown}\ t_k^{\mathbf{B}}, \Gamma \longrightarrow \Delta, Q^k, t_{k+1}^{\mathbf{B}} \qquad \overset{IH}{\triangledown}\ t_k^{\mathbf{B}}, \Gamma, Q^k \longrightarrow \Delta, t_{k+1}^{\mathbf{B}}}{t_k^{\mathbf{B}}, \Gamma \longrightarrow \Delta, t_{k+1}^{\mathbf{B}}}$$

where the subproofs marked *IH* are obtained by the inductive hypohesis, with σ_0, σ_1 construed as winning strategies from the corresponding initial states. Thus it remains to derive the base cases of simple contradictions by polynomial size proofs:

- $\neg$ contradictions of σ have form either of:
  - $\{B_i \mapsto 0, \neg B_i \mapsto 0\}$: these are translated to polynomial size proofs over $\mathcal{D}_i^k$ of $t_k^{\mathbf{B}}, \Gamma \longrightarrow B_i$, $1B_i^k0, \Delta, t_{k+1}^{\mathbf{B}}$, by Theorem 5.7.

– $\{B_i \mapsto 1, \neg B_i \mapsto 1\}$: these are translated to polynomial size proofs over $\mathcal{D}^k_i$ of $t^{\mathbf{B}}_k, \Gamma, B_i, 1B^k_i 0 \longrightarrow \Delta, t^{\mathbf{B}}_{k+1}$, again by Theorem 5.7.

- all other simple contradictions, namely the $\{\vee, \wedge\}$ contradictions, decision contradictions, extension contradictions and similarity contradictions, are handled in exactly the same way as for the deterministic case, in the proof of Proposition 4.8, just weakening the extraneous threshold extension variables. □

Now we obtain Proposition 6.9 by composing proofs from Lemma 6.10, for $k = 0, \ldots, N$, and appealing to the monotonicity properties of thresholds from Lemma 5.5 earlier:

Proof of Proposition 6.9. Set $\mathcal{B}' := \mathcal{D}^0 \cup \cdots \cup \mathcal{D}^N$ and construct the $\mathsf{LK}^+(\mathsf{eLNDT})$ proof:

$$\mathsf{c}\,\dfrac{(N+2)\cdot\mathsf{cut}\,\dfrac{\dfrac{\text{Lemma 5.5}}{\longrightarrow t^{\mathbf{B}}_0} \quad \dfrac{\text{Lemma 6.10}}{t^{\mathbf{B}}_0, \Gamma \longrightarrow \Delta, t^{\mathbf{B}}_1} \quad \cdots \quad \dfrac{\text{Lemma 6.10}}{t^{\mathbf{B}}_N, \Gamma \longrightarrow \Delta, t^{\mathbf{B}}_{N+1}} \quad \dfrac{\text{Lemma 5.5}}{t^{\mathbf{B}}_{N+1} \longrightarrow}}{\Gamma, \ldots, \Gamma \longrightarrow \Delta, \ldots, \Delta}}{\Gamma \longrightarrow \Delta}$$

□

6.3. **From $\mathsf{LK}^+(\mathsf{eLNDT})$ to eLNDT.** Just like in the deterministic case, we obtain a polynomial simulation of $\mathsf{LK}^+(\mathsf{eLNDT})$ by eLNDT by the Boolean constructions of [BDK20, Section 5] (see also [BDK19]):

Proposition 6.11. *If $\mathsf{LK}^+(\mathsf{eLNDT})$ has a size N proof over $\mathcal{E}$ of an eNDT sequent $\Gamma \longrightarrow \Delta$ then eLNDT has a $\mathrm{poly}(N)$ size proof of $\Gamma \longrightarrow \Delta$ over some $\mathcal{E}' \supseteq \mathcal{E}$.*

In fact the above result is also obtained as a special case of the development of positive decisions in Section 5.1: setting $\mathcal{E}' := \mathcal{E}^+$, we can define $A \wedge B := 0A(0 \vee B)$, as given in Definition 5.1, and we can duly derive the $\wedge$-l and $\wedge$-r rules of Eq. (4.1), under this interpretation, by appealing to the truth conditions for positive decisions from Lemma 5.2.

As promised the proof of the main result of this section is now obtained by composing the various polynomial simulations we have obtained:

Proof of Theorem 6.1. Immediate from Propositions 6.7, 6.9 and 6.11. □

7. A proof complexity theoretic analogue of *co*$\mathbf{NL} = \mathbf{NL}$

In this section, we use our non-uniform formalisation of the Immerman-Szelepcsényi result to obtain a bona fide proof complexity theoretic version of *co*$\mathbf{NL} = \mathbf{NL}$.

To formulate the statement we want, it is not enough to equate eLNDT with some 'dual' version, reasoning over co-NBPs: as sequents are two-sided, having the negation of a eNDT formula A on one side is the same as having A on the other side. The corresponding analogue we shall formalise is rather akin to "the Logspace Hierarchy collapses to $\mathbf{NL}$". To be concrete, a $\exists\forall\mathbf{BP}$ is just an alternating branching program with only one alternation, starting with non-deterministic states (see, e.g., [Weg00] for a detailed exposition). We shall show in this section that eLNDT polynomially simulates a corresponding system for $\exists\forall$BPs.

7.1. **Systems for co-nondeterministic and ∃∀-alternating branching programs.** First let us define the appropriate systems. In terms of notation, we shall make use of explicit metavariables to delineate alternations of $\vee$ and $\wedge$ in alternating branching programs. All notation should be construed as self-contained to this section, to avoid any clashes with previous sections. As we have already seen several examples of systems, representations and proofs in related systems, we shall remain as brief as possible in this subsection.

A **co-eNDT formula**, written U,V etc., is generated by,

$$U,V,\ldots \quad ::= \quad 0 \quad | \quad 1 \quad | \quad UpV \quad | \quad U \wedge V \quad | \quad u_i$$

where $u_0,u_1,\ldots$ is a fresh set of **co-eNDT extension variables**.

From here an **e∃∀DT formula**, written X,Y etc., is generated by,

$$X,Y,\ldots \quad ::= \quad 0 \quad | \quad 1 \quad | \quad U \quad | \quad XpY \quad | \quad X \vee Y \quad | \quad x_i$$

where $x_0,x_1,\ldots$ is a fresh set of **e∃∀DT extension variables**.

Extension axiom sets for both types of formula above are defined as expected, i.e. sets of the form $\mathcal{U} = \{u_i \leftrightarrow U_i\}_{i<n}$ or $\mathcal{X} = \{x_i \leftrightarrow X_i\}_{i<n}$, with each U_i or X_i, respectively, containing only extension variables among $u_0,\ldots,u_{i-1}$ or $x_0,\ldots,x_{i-1}$, respectively. It is not hard to see that e∃∀DT formulas, over e∃∀DT extension axiom sets, represent ∃∀BPs, just like eNDT formulas for NBPs.

Definition 7.1 (Systems for co-eNDT and e∃∀DT formulas)**.** The system co-eLNDT is defined like eLNDT, cf. Definition 2.8 only over co-eNDT formulas instead of eNDT formulas, using $\wedge$-rules from (4.1) instead of the $\vee$-rules of eLNDT.

The system eL∃∀DT is defined just like eLNDT, cf. Definition 2.8, only over e∃∀DT formulas instead of eNDT formulas, so including $\wedge$-rules from (4.1) as well as the rules of eLNDT, over appropriate formulas.

Remark 7.2 (Arbitrary alternation)**.** It is not hard to see from here how to develop a syntax for more general alternating branching programs, but we shall remain within the restricted systems here defined to avoid cumbersome metavariable bookkeeping.

The main result of this section is:

Theorem 7.3 (Proof complexity theoretic analogue of *co***NL** = **NL**)**.** eLNDT *polynomially simulates* eL∃∀DT*, over NDT sequents.*

To prove this we define a series of translations, according to different types of BPs, and combine them appropriately. Like earlier results, we could refine this result to account for conclusions with extension variables, over corresponding sets of extension axioms. However, as we shall work with many systems in this section over different languages, we shall refrain from doing this to simplify theorem statements and their proofs.

7.2. eLNDT **polynomially simulates co-**eLNDT**.** First, as a warm up result, and for later use, let us establish the expected polynomial equivalence between eLNDT and co-eLNDT over simple sequents. We state the result in only the one direction that we shall need (the other being entirely dual):

Proposition 7.4. eLNDT *polynomially simulates co-*eLNDT *over DT sequents.*

The result follows by 'dualising' a proof; let us make this notion formal:

Definition 7.5 (Negation)**.** For each co-eNDT extension variable u_j let its **negation** $\bar{u}_j$ be a fresh eNDT extension variable. We extend negation to all co-eNDT formulas by,

$$\begin{aligned}\bar{0} &:= 1\\ \bar{1} &:= 0\\ \overline{UpV} &:= \bar{U}p\bar{V}\\ \overline{U\wedge V} &:= \bar{U}\vee\bar{V}\end{aligned}$$

We write $\bar{\Gamma} := \{\bar{U} : U \in \Gamma\}$. We further extend this notation to sets of co-eNDT extension axioms: if $\mathcal{U} = \{u_i \leftrightarrow U_i\}_{i<m}$ then $\bar{\mathcal{U}} := \{\bar{u}_i \leftrightarrow \bar{U}_i\}_{i<m}$.

Note that, for any co-eNDT formula U, its negation $\bar{U}$ is an eNDT formula. Moreover, it is not hard to see that if U is interpreted over some extension axioms $\mathcal{U}$, then $\bar{U}$ computes its negation, over $\bar{\mathcal{U}}$, in the expected way. We can make this observation proof-relevant too:

Lemma 7.6. *For each co-*eLNDT *proof π over $\mathcal{U}$ of a co-eNDT sequent $\Gamma \longrightarrow \Delta$ there is an* eLNDT *proof $\bar{\pi}$ over $\bar{\mathcal{U}}$ of $\bar{\Delta} \longrightarrow \bar{\Gamma}$ of size polynomial in $|\pi|$.*

Proof. Construct $\bar{\pi}$ by replacing all co-eNDT formulas U of the LHS of the sequent with $\bar{U}$ in the RHS and vice versa. We justify each resulting step of inference as follows:

- Identity and cut steps on U are translated to identity and cut steps, respectively, on $\bar{U}$.
- Extension hypotheses remain correct by definition.
- Each left structural step s-l is translated to a right structural step s-r and vice-versa.
- 0-l and $\wedge$-l steps are respectively translated to 1-r and $\vee$-r steps, and vice-versa. E.g.:

$$\wedge\text{-}l\,\frac{\Gamma, U, V}{\Gamma, U\wedge V \longrightarrow \Delta} \quad\rightsquigarrow\quad \vee\text{-}r\,\frac{\bar{\Delta} \longrightarrow \bar{\Gamma}, \bar{U}, \bar{V}}{\bar{\Delta} \longrightarrow \bar{\Gamma}, \bar{U}\vee\bar{V}}$$

$$\wedge\text{-}r\,\frac{\Gamma \longrightarrow \Delta, U \quad \Gamma \longrightarrow \Delta, V}{\Gamma \longrightarrow \Delta, U\wedge V} \quad\rightsquigarrow\quad \vee\text{-}l\,\frac{\bar{\Delta}, \bar{U} \longrightarrow \bar{\Gamma} \quad \bar{\Delta}, \bar{V} \longrightarrow \bar{\Gamma}}{\bar{\Delta}, \bar{U}\vee\bar{V} \longrightarrow \bar{\Gamma}}$$

- Finally p-l and p-r steps are translated as follows,

$$p\text{-}l\,\frac{\Gamma, U \longrightarrow p, \Delta \quad \Gamma, p, V \longrightarrow \Delta}{\Gamma, UpV \longrightarrow \Delta} \quad\rightsquigarrow\quad p\text{-}r\,\frac{\bar{\Delta}, \bar{p} \longrightarrow \bar{U}, \bar{\Gamma}}{\bar{\Delta} \longrightarrow p, \bar{U}, \bar{\Gamma}}\circ \quad \frac{\bar{\Delta} \longrightarrow \bar{p}, \bar{V}, \bar{\Gamma}}{\bar{\Delta}, p \longrightarrow \bar{V}, \bar{\Gamma}}\bullet}{\bar{\Delta} \longrightarrow \bar{U}p\bar{V}, \bar{\Gamma}}$$

$$p\text{-}r\,\frac{\Gamma \longrightarrow p, U, \Delta \quad \Gamma, p \longrightarrow V, \Delta}{\Gamma \longrightarrow UpV, \Delta} \quad\rightsquigarrow\quad p\text{-}l\,\frac{\bar{\Delta}, \bar{U}, \bar{p} \longrightarrow \bar{\Gamma}}{\bar{\Delta}, \bar{U} \longrightarrow p, \bar{\Gamma}}\circ \quad \frac{\bar{\Delta}, \bar{V} \longrightarrow \bar{p}, \bar{\Gamma}}{\bar{\Delta}, p, \bar{V} \longrightarrow \bar{\Gamma}}\bullet}{\bar{\Delta}, \bar{U}p\bar{V} \longrightarrow \bar{\Gamma}}$$

  where the steps marked $\circ$ are obtained by cutting against constant-size proofs of $\longrightarrow p, \bar{p}$, and the steps marked $\bullet$ are obtained by cutting against constant-size proofs of $p, \bar{p} \longrightarrow$. □

From here, by cutting against identities on eDT formulas, we immediately have a proof of our aforementioned simulation.

Proof of Proposition 7.4. For a eDT formula A, it is routine to construct polynomial-size LDT-proofs of $\longrightarrow A, \bar{A}$ and $A, \bar{A} \longrightarrow$, by induction on the structure of A. Now, given a co-eLNDT proof of a DT sequent $\Gamma \longrightarrow \Delta$, first apply Lemma 7.6 to obtain an eLNDT-proof of $\bar{\Delta} \longrightarrow \bar{\Gamma}$, then apply linearly many cuts against $\longrightarrow A, \bar{A}$ for each $A \in \Delta$ and against $B, \bar{B} \longrightarrow$ for each $B \in \Gamma$, to obtain a proof of $\Gamma \longrightarrow \Delta$. □

7.3. **Simulating $\mathsf{eL}\exists\forall\mathsf{DT}$ relative to a fixed number of true variables.** Towards a proof of Theorem 7.3, let us now fix, for the remainder of this section, an $\mathsf{eL}\exists\forall\mathsf{DT}$ proof π of an NDT sequent $\Sigma \longrightarrow \Pi$ over extension axioms $\mathcal{X} = \{x_j \leftrightarrow X_j\}_{j<n}$.

Let $\mathbf{U} = U_0, \ldots, U_{N-1}$ enumerate the co-eNDT formulas in π that are *not* eDT formulas. Write $\bar{\mathbf{U}} = \bar{U}_0, \ldots, \bar{U}_{N-1}$ and $U_i^k := 1\bar{U}_i^k 0$. Now we can speicalise Theorem 5.7 to obtain extension axiom sets $\mathcal{D}_i^k \supseteq \mathcal{T}_{\mathcal{B}}^{\mathbf{B}}$, for $i < N$ and $k \in \mathbb{Z}$, such that:

Corollary 7.7. *There are polynomial size* eLNDT *proofs of,*

- $t_k^{\bar{\mathbf{U}}}, \bar{U}_i, U_i^k \longrightarrow t_{k+1}^{\bar{\mathbf{U}}}$
- $t_k^{\bar{\mathbf{U}}} \longrightarrow \bar{U}_i, U_i^k, t_{k+1}^{\bar{\mathbf{U}}}$

over $\mathcal{D}_i^k$, for $i < N$ and $k \in \mathbb{Z}$.

Proof sketch. From Theorem 5.7 we have polynomial size proofs of $t_k^{\bar{\mathbf{U}}}, \bar{U}_i, U_i^k \longrightarrow 0, t_{k+1}^{\bar{\mathbf{U}}}$ and $t_k^{\bar{\mathbf{U}}}, 1 \longrightarrow \bar{U}_i, U_i^k, t_{k+1}^{\bar{\mathbf{U}}}$, over $\mathcal{D}_i^k$, whence the two sequents above follow by cutting against the initial sequents 0-l and 1-r respectively. □

We want a sort of generalisation of this result to cover all e$\exists\forall$DT-theorems. Let us first extend the definition of $\cdot^k$ appropriately:

Definition 7.8 (k-translation)**.** We extend the notation U_i^k to all e$\exists\forall$DT formulas of π, defining X^k for each X in π as follows,

$$\begin{aligned} (XpY)^k &:= X^k p Y^k \\ (X \vee Y)^k &:= X^k \vee Y^k \\ x_j^k &:= e_j^k \end{aligned}$$

where e_j^k is a fresh eNDT extension variable. We also write $\Gamma^k := \{A^k : A \in \Gamma\}$, for cedents Γ of π, and $\mathcal{X}^k := \{e_j^k \leftrightarrow X_j^k\}_{j<n}$.

Again, let us point out that each X^k is an eNDT formula. Writing $\mathcal{D}^k := \mathcal{D}_0^k \cup \cdots \cup \mathcal{D}_{N-1}^k$, let us first establish the following result:

Lemma 7.9. *There are polynomial-size* eLNDT *proofs of $t_k^{\bar{\mathbf{U}}}, \Sigma \longrightarrow \Pi, t_{k+1}^{\bar{\mathbf{U}}}$ over $\mathcal{X}^k \cup \mathcal{D}^k$, for each $k \in \mathbb{Z}$.*

Proof. We construct the required proof π^k from π as follows:

- Add $t_k^{\bar{\mathbf{U}}}$ to the LHS of each line and $t_{k+1}^{\bar{\mathbf{U}}}$ to the RHS of each line.
- Replace each each U_i in the LHS (or RHS) by $\bar{U}_i$ in the RHS (respectively LHS).
- Replace each e$\exists\forall$DT formula X that is not a co-eNDT formula (and so also not a eDT formula) by X^k.

Notice that (even extended) NDT formulas are unaffected by the formula translation described above, so π^k certainly ends with the desired sequent, $t_k^{\bar{\mathbf{U}}}, \Sigma \longrightarrow \Pi, t_{k+1}^{\bar{\mathbf{U}}}$. Moreover, π^k is almost a correct eLNDT proof, it only remains to repair certain inference steps:

- Identities and other initial sequents remain derivable with additional weakenings for $t_k^{\bar{\mathbf{U}}}$ on the LHS and $t_{k+1}^{\bar{\mathbf{U}}}$ on the RHS.
- Cuts remain intact by the transformation above.
- Any structural steps remain intact by the transformation above.
- Any logical or extension step on eDT formulas remains intact, as the transformation does not affect them.

- A LHS step on a co-eNDT formula U_i is transformed into a RHS step on its dual $\bar{U}_i$, and vice versa, just like in the proof of Lemma 7.6.
- Logical steps on e$\exists\forall$DT formulas (that are not co-eNDT formulas) remain intact, as the $.^k$ translation commutes with decisions, disjunctions and extensions.
- The remaining critical cases are when, bottom-up, a co-eNDT formula 'appears' for the first time, i.e. after a $\vee$-l or $\vee$-r step on a e$\exists\forall$DT formula when at least one auxiliary formula is a co-eNDTformula. Such steps are translated as follows,

$$\vee\text{-}l\,\frac{\Gamma, U_i \to \Delta \quad \Gamma, X \to \Delta}{\Gamma, U_i \vee X \to \Delta} \quad \rightsquigarrow \quad \vee\text{-}l\,\frac{\bullet\,\dfrac{t_k^{\bar{\mathbf{U}}}, \Gamma' \to \bar{U}_i, \Delta', t_{k+1}^{\bar{\mathbf{U}}}}{t_k^{\bar{\mathbf{U}}}, \Gamma', U_i^k \to \Delta', t_{k+1}^{\bar{\mathbf{U}}}} \quad t_k^{\bar{\mathbf{U}}}, \Gamma', X^k \to \Delta}{t_k^{\bar{\mathbf{U}}}, \Gamma', U_i^k \vee X^k \to \Delta', t_{k+1}^{\bar{\mathbf{U}}}}$$

$$\vee\text{-}r\,\frac{\Gamma \to \Delta, U_i, X}{\Gamma \to \Delta, U_i \vee X} \quad \rightsquigarrow \quad \vee\text{-}r\,\frac{\circ\,\dfrac{t_k^{\bar{\mathbf{U}}}, \Gamma', \bar{U}_i \to \Delta', X^k, t_{k+1}^{\bar{\mathbf{U}}}}{t_k^{\bar{\mathbf{U}}}, \Gamma' \to \Delta', U_i^k, X^k, t_{k+1}^{\bar{\mathbf{U}}}}}{t_k^{\bar{\mathbf{U}}}, \Gamma' \to \Delta', U_i^k \vee X^k, \Delta', t_{k+1}^{\bar{\mathbf{U}}}}$$

  where Γ', Δ' are obtained from Γ, Δ by the formula replacement we carried out, and the steps marked $\bullet$ or $\circ$ are derived by cutting against the appropriate duality property for U_i^k from Corollary 7.7. In the case where X is also some U_j, we cut against a further instance of Corollary 7.7 to convert, bottom-up, U_j^k on the LHS or RHS to $\bar{U}_j$ on the RHS or LHS, respectively. □

7.4. **Putting it all together.** We can now assemble the proof of our main result. Recall, from the beginning of the previous subsection, that we already fixed an $\mathsf{eL}\exists\forall\mathsf{DT}$ proof π of an NDT sequent $\Sigma \to \Pi$ over extension axioms $\mathcal{X} = \{x_j \leftrightarrow X_j\}_{j<n}$, with co-eNDT formulas (that are not eDT formulas) among $\mathbf{U}$.

Proof of Theorem 7.3. We have from Lemma 7.9 polynomial-size eLNDT proofs, for each $k \in \mathbb{Z}$, of $t_k^{\bar{\mathbf{U}}}, \Sigma \to \Pi, t_{k+1}^{\bar{\mathbf{U}}}$, over $X^k \cup \mathcal{D}^k$. We can now cut these together to obtain the eLNDT proof,

$$\mathsf{c}\,\frac{(N+2)\mathsf{cut}\,\dfrac{\dfrac{\text{Prop. 5.5}}{\to t_0^{\bar{\mathbf{U}}}} \quad \dfrac{\text{Lem. 7.9}}{t_0^{\bar{\mathbf{U}}}, \Sigma \to \Pi, t_1^{\bar{\mathbf{U}}}} \quad \cdots \quad \dfrac{\text{Lem. 7.9}}{t_N^{\bar{\mathbf{U}}}, \Sigma \to \Pi, t_{N+1}^{\bar{\mathbf{U}}}} \quad \dfrac{\text{Prop. 5.5}}{t_{N+1}^{\bar{\mathbf{U}}} \to}}{\Sigma, \ldots, \Sigma \to \Pi, \ldots, \Pi}}{\Sigma \to \Pi}$$

over extension axioms $\mathcal{X}^0 \cup \mathcal{D}^0 \cdots \cup \mathcal{X}^N \cup \mathcal{D}^N$, as required. □

8. Conclusions

We proposed Prover-Adversary games NB and DB for reasoning about non-deterministic branching programs (NBPs) and deterministic branching programs (BPs), respectively. We showed that DB and NB correspond to the previously introduced systems eLDT and eLNDT, respectively, from [BDK20, BDK19]. In the deterministic case, let us point out that similar ideas were communicated by Cook but, as far as we can tell, never published (see [Coo01]). For the non-deterministic case we formalised a non-uniform version of the Immerman-Szelepcsényi theorem, $co\mathbf{NL} = \mathbf{NL}$ [Imm88, Sze88] to (partially) negate NBPs.

We applied our main technical Immerman-Szelepcsényi construction to establish a proof complexity theoretic version of $co\mathbf{NL} = \mathbf{NL}$, namely that a system for the second level of the logspace hierarchy collapses to the first level. Precisely, we showed that the system eLNDT reasoning about non-deterministic branching programs polynomially simulates the system $\mathsf{eL}\exists\forall\mathsf{DT}$ reasoning about alternating branching programs with two alternations, $\exists\forall$.

Let us point out that a nonuniform development of the Immerman-Szelepcsényi result, in particular for computing negation of NBPs, has appeared before [Vin96]. The construction in that work is different from ours as our deciders AB_i^kC only work relative to a fixed number k of true inputs among $\mathbf{B}$. On the other hand [Vin96] does not consider a *formalisation* of the construction within a proof system or theory as we have done here, cf. Theorem 5.7.

It would be prudent to establish some *bounded arithmetic* correspondence between these systems and appropriate theories of arithmetic, e.g. $\mathbf{VL}$ and $\mathbf{VNL}$ for $\mathbf{L}$ and $\mathbf{NL}$ (see, e.g., [CN10]). At the same time it would be interesting to understand better the bounded reverse mathematical strength of the Immerman-Szelepcsényi Theorem (cf. [CK04, Per09]). We are aware of ongoing work in this direction by Beckman, Buss, Das and Knop.

We can see eLDT and eLNDT as the canonical 'Frege' systems for $\mathbf{L}$ and $\mathbf{NL}$ respectively. It would therefore also be interesting to use our games to compare (fragments of) $\mathsf{eL(N)DT}$ with recently studied 'OBDD proof systems' in proof complexity, e.g. in [AKV04, BIKS18, CZ09, IR22].

Acknowledgments

We would like to thank Arnold Beckmann, Sam Buss and Sasha Knop for several helpful discussions and related research collaborations that motivated this work. We would also like to thank Meena Mahajan for bringing to our attention some relevant references.

References

[AGP02] Albert Atserias, Nicola Galesi, and Pavel Pudlák. Monotone simulations of non-monotone proofs+. *J. Comput. Syst. Sci.*, 65(4):626–638, 2002.

[AKV04] Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In Mark Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2004. doi:10.1007/978-3-540-30201-8_9.

[BDK19] Sam Buss, Anupam Das, and Alexander Knop. Proof complexity of systems of (non-deterministic) decision trees and branching programs, 2019. URL: https://arxiv.org/abs/1910.08503, doi:10.48550/ARXIV.1910.08503.

[BDK20] Sam Buss, Anupam Das, and Alexander Knop. Proof complexity of systems of (non-deterministic) decision trees and branching programs. In *Proceedings of CSL*, 2020.

[BIKS18] Sam Buss, Dmitry Itsykson, Alexander Knop, and Dmitry Sokolov. Reordering rule makes OBDD proof systems stronger. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 16:1–16:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.16.

[Bus85] Sam Buss. *Bounded arithmetic*. Princeton University, 1985.

[Bus87] Sam Buss. The boolean formula value problem is in ALOGTIME. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 123–131. ACM, 1987. doi:10.1145/28395.28409.

[CB08] Joost-Pieter Katoen Christel Baier. *Principles of Model Checking*. 2008.

[CK04] Stephen A. Cook and Antonina Kolokolova. A second-order theory for NL. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 398–407. IEEE Computer Society, 2004. doi:10.1109/LICS.2004.1319634.

[CN10] Stephen Cook and Phuong Nguyen. *Logical foundations of proof complexity*, volume 11. Cambridge University Press Cambridge, 2010.

[Coo01] Stephen Cook. A survey of complexity classes and their associated propositional proof systems and theories and a proof system for log space, 2001. Slides for Edinburgh talk.http://www.cs.toronto.edu/~sacook/.

[CZ09] Wei Chen and Wenhui Zhang. A direct construction of polynomial-size OBDD proof of pigeon hole problem. *Inf. Process. Lett.*, 109(10):472–477, 2009. doi:10.1016/j.ipl.2009.01.006.

[DD22] Anupam Das and Avgerinos Delkos. Proof complexity of monotone branching programs. In Ulrich Berger, Johanna N. Y. Franklin, Florin Manea, and Arno Pauly, editors, *Revolutions and Revelations in Computability - 18th Conference on Computability in Europe, CiE 2022, Swansea, UK, July 11-15, 2022, Proceedings*, volume 13359 of *Lecture Notes in Computer Science*, pages 74–87. Springer, 2022. doi:10.1007/978-3-031-08740-0_7.

[DD25] Anupam Das and Avgerinos Delkos. Proof complexity of positive branching programs. *Log. Methods Comput. Sci.*, 21(1), 2025. URL: https://doi.org/10.46298/lmcs-21(1:23)2025, doi:10.46298/LMCS-21(1:23)2025.

[Gri91] Michelangelo Grigni. *Structure in monotone complexity*. PhD thesis, 1991. URL: http://www.mathcs.emory.edu/~mic/papers/thesis.ps.gz.

[GS92] Michelangelo Grigni and Michael Sipser. Monotone complexity. In *Poceedings of the London Mathematical Society Symposium on Boolean Function Complexity*, page 57–75, USA, 1992. Cambridge University Press.

[Imm88] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on computing*, 17(5):935–938, 1988.

[IR22] Dmitry Itsykson and Artur Riazanov. Automating OBDD proofs is NP-hard. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 59:1–59:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2022/16857, doi:10.4230/LIPIcs.MFCS.2022.59.

[Kra94] Jan Krajíček. Lower bounds to the size of constant-depth propositional proofs. *The Journal of Symbolic Logic*, 59(1):73–86, 1994. URL: https://doi.org/10.2307/2275250.

[Kra95] Jan Krajíček. *Bounded arithmetic, propositional logic and complexity theory*, volume 60. Cambridge University Press, 1995.

[Kra19] Jan Krajíček. *Proof complexity*, volume 170. Cambridge University Press, 2019.

[PB94] Pavel Pudlák and Sam Buss. How to lie without being (easily) convicted and the length of proofs in propositional calculus. In *Proceedings of CSL*, 1994. doi:10.1007/BFb0022253.

[Per09] Steven Perron. *Power of non-uniformity in proof complexity*. PhD thesis, 2009.

[Spi71] Philip Spira. On time-hardware complexity tradeoffs for boolean functions. In *Proceedings of the 4th Hawaii Symposium on System Sciences, 1971*, pages 525–527, 1971.

[Sze88] Róbert Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988. doi:10.1007/BF00299636.

[Vin96] V. Vinay. Hierarchies of circuit classes that are closed under complement. In Steven Homer and Jin-Yi Cai, editors, *Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity, Philadelphia, Pennsylvania, USA, May 24-27, 1996*, pages 108–117. IEEE Computer Society, 1996. URL: https://urldefense.com/v3/__https://doi.org/10.1109/CCC.1996.507674__;!!CF15FET90Tp8!AirsPZ1esFvc_sF2_ifQNJDTK9R6K2ttfbDVq9BgOq4wflVe2vx2n2-T2fo5IM4XVzv_JYPSIYO_nlPw$, doi:10.1109/CCC.1996.507674.

[Weg00] Ingo Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, 2000. URL: http://ls2-www.cs.uni-dortmund.de/monographs/bdd/.