Quantum Verification of Compressed Proofs through DLDS-to-Circuit Compilation

Lorenzo Saraiva[1], **Edward Hermann Haeusler**[1]

[1]Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro, RJ – Brazil

{lsaraiva,hermann}@inf.puc-rio.br

Abstract. *Quantum computation provides a natural mechanism for verifying global properties of exponentially large combinatorial spaces when these spaces admit compact representations. We present a quantum verification architecture for compressed proof objects based on Dag-Like Derivability Structures (DLDS), polynomial-size directed acyclic graphs obtained by horizontally compressing Natural Deduction proofs. A single DLDS may encode exponentially many tree-shaped derivations, called* readings*, so explicit reading-by-reading verification is exponential in the number of branching points. We develop a polynomial-time compiler that translates the reading semantics of a restricted class of DLDSs into a reversible quantum oracle. The admissible regime covered by the correctness proof consists of normalized implicational DLDSs with a reading-independent introduction suffix, handled through a deferred-discharge criterion. The resulting oracle is compatible with Grover-style amplitude amplification over the reading space. We prove correctness for this class, establish $O(n^2)$ bounds for qubits and gates, and validate the construction using Qiskit simulations on synthetic and pipeline-derived DLDS instances. The construction provides concrete quantum verification infrastructure motivated by the conjectured inclusion* $\mathrm{coNP} \subseteq \mathrm{QCMA}$*, while leaving the underlying proof-theoretic certificate question as a separate open problem.*

1. Introduction

Quantum computation is particularly well suited for applying structured tests coherently over large combinatorial spaces when these spaces admit compact representations. By preparing a superposition of possible configurations and applying a reversible oracle, a quantum verifier can mark the configurations that violate a desired property without explicitly enumerating them. Amplitude amplification techniques such as Grover search can then increase the probability of observing a marked configuration [Grover 1996, Boyer et al. 1998]. This paradigm becomes especially relevant when a polynomial-size structure implicitly represents exponentially many classical objects. In such situations, classical verification may require enumerating all possibilities, while a quantum algorithm can query a verifier coherently over the corresponding superposition.

In this paper we investigate this paradigm in the context of proof verification. We consider *Dag-Like Derivability Structures* (DLDS), polynomial-size DAG representations of Natural Deduction proofs obtained via the Horizontal Compression algorithm (HC) of [Gordeev and Haeusler 2019, Haeusler et al. 2025]. A DLDS is a polynomial-size directed acyclic graph (DAG) in which nodes correspond to formulas and edges represent

applications of inference rules. Because identical sub-derivations are merged, a single DLDS may encode exponentially many tree-shaped derivations, which we call *readings*. A DLDS with k branching points represents up to 2^k possible readings.

The verification problem studied here is reading-based: a DLDS is accepted when *all* of its readings correspond to valid Natural Deduction derivations. Explicit reading-by-reading inspection is exponential in the branching parameter k, because the DLDS represents a family of up to 2^k tree-like proofs. In this setting, a DLDS may be viewed as a compact classical witness encoding an exponentially large family of derivations. Our goal is to provide an explicit quantum verification architecture for such witnesses: a reversible oracle whose computational branches coincide with the possible readings of the compressed proof object.

We show that quantum computation provides a natural verification mechanism for this setting. By encoding the branching choices of a DLDS in a register of k qubits prepared in uniform superposition, a quantum circuit can apply the same verification oracle coherently across all readings. Grover amplification can then detect invalid readings, if they exist, with high probability using only $O(\sqrt{2^k})$ oracle calls.

Beyond its algorithmic construction, this work is motivated by the broader question of quantum verification of classical certificates. In our setting, a DLDS plays the role of a compact classical witness, while the compiled quantum circuit acts as a verifier for its reading semantics. This provides concrete quantum verification machinery for a protocol motivated by the conjectured inclusion $\mathrm{coNP} \subseteq \mathrm{QCMA}$, but it should not be read as a proof of that inclusion. For the family of non-Hamiltonicity tautologies α_G, whose formulas have size $\mathcal{O}(|G|^3)$, the number of branching points satisfies $k = O(\log n)$ where n is the number of vertices of G [Gordeev and Haeusler 2019]. Under such a branching bound, Grover search over the reading space requires only $\mathrm{poly}(n)$ oracle calls. Whether suitable polynomial-size DLDS certificates exist for all such tautologies remains an open proof-theoretic question; we refer to [Gordeev and Haeusler 2022, Gordeev and Haeusler 2019] for further discussion. The goal of this article is therefore narrower: to provide an explicit reversible quantum verification architecture for the reading semantics of the compressed proof objects considered here.

To realize this approach, we develop a polynomial-time compiler that maps a DLDS into a quantum verification circuit.[1] The compiler constructs an oracle that propagates hypothesis dependencies through the DAG using reversible operations, routes dependencies across branching points through MUX-controlled gates, and performs a root discharge check identifying invalid derivations. The resulting circuit serves directly as the oracle in a Grover-style search.

2. Background

Natural Deduction and Proof Compression. Natural Deduction, introduced by Gentzen [Gentzen 1935] and refined by Prawitz [Prawitz 1965], is a proof system for propositional and first-order logic. In the fragment with implication ($\supset$), proofs are tree-structured derivations built from two rules:

[1]Implementation available at `https://github.com/lorenzosaraiva/dlds-pipeline`.

$$\frac{\begin{array}{c}[A]^i\\ \vdots\\ B\end{array}}{A \supset B}\,{\supset}\mathrm{I}_i \qquad \frac{A \qquad A \supset B}{B}\,{\supset}\mathrm{E}$$

Implication Introduction (${\supset}\mathrm{I}_i$) discharges hypothesis A labeled i, binding it into the implication $A \supset B$. Implication Elimination (${\supset}$E, modus ponens) combines a minor premise A with a major premise $A \supset B$ to derive B.

Remark 1. *The purely implicational fragment, equipped solely with implication introduction and implication elimination, polynomially simulates any propositional logic with the sub-formula property. That is, derivations in arbitrary propositional systems can be translated into implicational derivations with at most polynomial increase in size.*

This logic is not classical, and therefore the above result should not be conflated with the classical interdefinability of Boolean connectives (cf. [Haeusler 2015]); the statement concerns proof-theoretic simulation, not classical functional completeness.

Normal proofs of certain tautologies, such as those arising from the Statman speedup theorem [Statman 1979], can have exponentially large proof trees. Haeusler's Horizontal Compression algorithm [Haeusler et al. 2025, Gordeev and Haeusler 2019] addresses this by collapsing shared sub-derivations into a single DAG node, producing a polynomial-size sub-structure with labeled edges that distinguish between multiple uses of the same node. Throughout this paper we work in the normalized implicational setting with greedy discharge, so that the introduction phase forms a common suffix below the elimination and branching structure exploited by the compiler.

Dag-Like Derivability Structures. To track which hypotheses are active at each point in a derivation, we associate each node with a *dependency bitvector*: a binary string of length t (the number of hypotheses), where bit i is set if hypothesis H_i feeds into the sub-derivation at that node. Under implication elimination, the dependency set at the conclusion is the bitwise OR of the two premises' dependency sets. Under implication introduction discharging hypothesis H_i, bit i is cleared. A derivation is closed (all hypotheses discharged) if and only if the root's dependency bitvector is all-zeros. Equivalently, a derivation is closed when every hypothesis that appears in the dependency set at the root is subsequently discharged by a ${\supset}$I rule. In Section 3, we exploit the layered structure of DLDS to verify this condition without explicitly implementing the discharge steps.

Definition 2 (DLDS)**.** *A Dag-Like Derivability Structure $\mathcal{D}$ is a directed acyclic graph (DAG) where each node v carries a formula $\varphi(v)$, a dependency bitvector $d_v \in \{0,1\}^t$ (where t is the number of hypotheses, i.e., top formulas), and labeled edges to child nodes, with labels (colors) indicating distinct uses.*

When the HC algorithm collapses two nodes with the same formula at the same derivation level, the resulting DAG loses the unique tree-like structure of the original proof. To preserve the logical content, HC introduces two mechanisms:

Colored deduction edges: after compression, deduction edges carry ordinal labels (colors) distinguishing which original branch they belong to; **ancestor edges:** auxiliary edges labeled with ordinal sequences record the provenance of collapsed nodes. An ancestor edge label $[c_1; c_2]$ means "follow edges of color c_1, then color c_2," encoding a path through the original derivation.

Together, colored edges and ancestor edges allow the compressed DAG to be read as encoding a family of tree-like derivations. Each such derivation is called a *reading*: it is obtained by selecting one color at each branching point, thereby recovering one tree-shaped derivation represented by the compressed structure. For k binary branching points, there are 2^k possible readings. The DLDS is valid if and only if every reading properly discharges all hypotheses. In this paper the semantics of a DLDS is precisely this family of readings. Our compiler therefore targets reading-based validity directly: the quantum circuit accepts exactly when every reading induces a closed tree-like derivation.

Example. Figure 1 shows the elimination and branching prefix of a cascaded DLDS for the tautology

$$(A_3 \supset (A_4 \supset A_5)) \supset (A_2 \supset (A_3 \supset A_4)) \supset$$
$$(A_1 \supset (A_2 \supset A_3)) \supset (A_1 \supset A_2) \supset (A_1 \supset A_5).$$

The common $\supset$I suffix discharging the five hypotheses is omitted and handled by deferred discharge (Section 3). The displayed prefix has $t = 5$ hypotheses and $k = 3$ branching points, hence $2^3 = 8$ readings. The node A_5 is the root. At each shaded branching node, the reading qubit controls which child receives the node's dependency vector. All readings yield $d_{A_5} = 11111$.

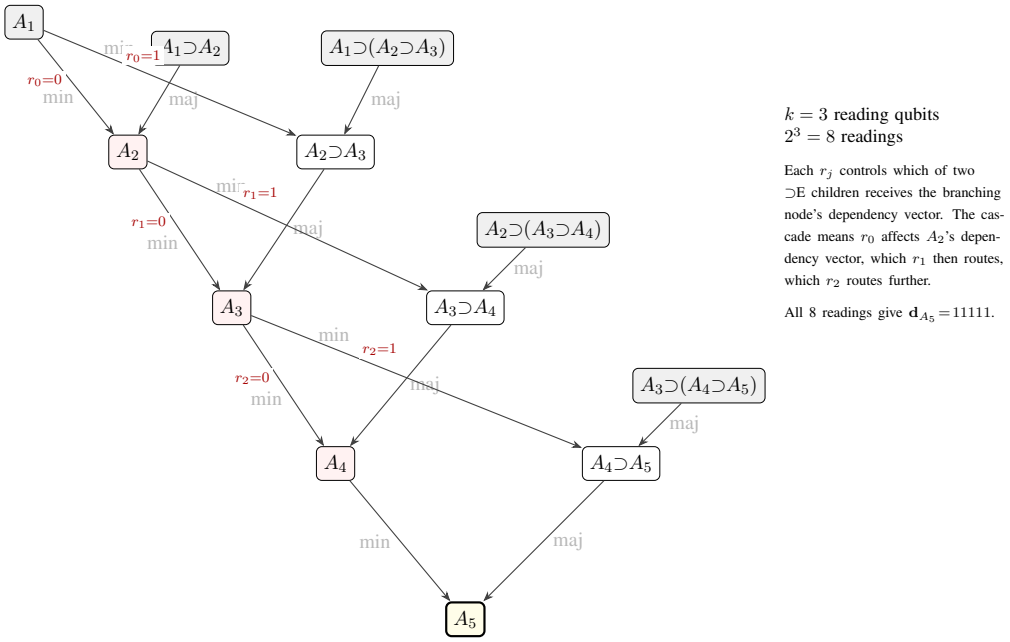

Figure 1. Cascaded DLDS with 5 hypotheses and $k = 3$ reading qubits. Each $\supset$E node has a minor and major premise (labeled), and A_5 is the root of the displayed DAG prefix. At the branching nodes A_1, A_2, and A_3 (shaded), the reading qubit r_j selects which of two children receives the node's dependency vector. The cascade means earlier reading choices affect the dependency vectors available to later branching decisions. All 8 readings discharge all hypotheses at the root.

Grover's Algorithm. Grover's search algorithm [Grover 1996] finds a marked item in an unstructured database of N items using $O(\sqrt{N})$ oracle queries, achieving a quadratic speedup over classical search. Given an oracle O_f that marks states $|x\rangle$ with $f(x) = 1$ via a phase flip ($|x\rangle \mapsto (-1)^{f(x)}|x\rangle$), Grover's algorithm iteratively applies O_f followed by a diffusion operator $D = 2|\psi\rangle\langle\psi| - I$ (where $|\psi\rangle$ is the uniform superposition). Af-

ter $O(\sqrt{N/M})$ iterations, where M is the number of marked items, the probability of measuring a marked item approaches 1.

When $M > N/2$, that is, the majority of items are marked, the rotation overshoots after one iteration and begins amplifying the unmarked minority. This observation is useful when comparing alternative marking conventions for DLDS readings.

3. DLDS-to-Circuit Compilation

We now describe the compiler that transforms a DLDS $\mathcal{D}$ with n nodes, t hypotheses, and k binary branching points into a quantum verification oracle. The construction targets the reading-based semantics described in Section 2: each assignment of the reading register determines one reading, and the oracle marks exactly those readings that fail the deferred-discharge criterion.

Circuit Architecture. The circuit operates on four kinds of registers: the **reading register** $\mathbf{r} = (r_0, \ldots, r_{k-1})$, consisting of k qubits encoding the reading choice and initialized in uniform superposition via Hadamard gates; **dependency registers** d_v for each node v, consisting of t-bit registers tracking which hypotheses feed into the sub-derivation at v and initialized to 0; **ancilla registers**, consisting of fresh t-bit registers for each reversible OR operation and initialized to 0; and an **output qubit** out, set to 1 if the root discharge check fails.

The total qubit count is $O(k + n \cdot t + e \cdot t + 1)$ where e is the number of $\supset$E nodes. Since $k, e, t \leq n$, the overall count is $O(n^2)$.

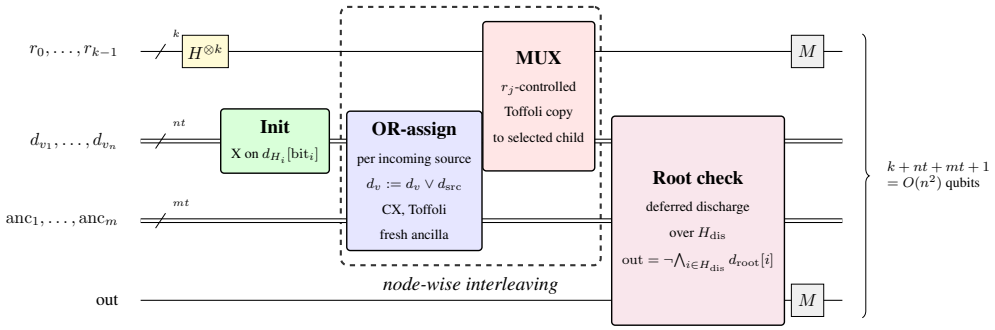

Figure 2. Schematic architecture: one t-bit dependency register per node, a fresh t-bit ancilla per OR-assign (left dirty until uncompute), MUX routing controlled by reading qubits, and a root test over H_{dis}.

Top-Formula Initialization. Each top-formula (hypothesis) H_i initializes its dependency register to the characteristic bitvector e_i, where $e_i[j] = 1$ iff $j = i$. This is implemented by a single X gate on position i of register d_{H_i}.

Reversible OR-Assign. An $\supset$E node v with minor premise v_1 and major premise v_2 computes $d_v = d_{v_1} \vee d_{v_2}$. Since OR is not reversible, we use the identity $a \vee b = a \oplus b \oplus (a \wedge b)$, yielding a three-step reversible circuit per bit position i using a fresh ancilla register a: apply CNOT$(d_v[i], a[i])$ to save the old value, then CNOT$(d_{\mathrm{src}}[i], d_v[i])$ to XOR with the source, then Toffoli$(d_{\mathrm{src}}[i], a[i], d_v[i])$ for the AND correction.

After these three steps, $d_v[i]$ holds $d_v^{\mathrm{old}}[i] \vee d_{\mathrm{src}}[i]$. The ancilla a retains the old value of d_v (it is "dirty"), but since a fresh ancilla is allocated per OR operation, subsequent computations are unaffected. This sequence is its own inverse when applied with

the same ancilla and essential for Grover's uncomputation step.

Ancilla overhead. The fresh-ancilla-per-OR strategy ensures correctness at the cost of increased qubit count: each of the $O(n)$ $\supset$E nodes contributes $O(t)$ ancilla qubits. Future optimizations could recycle ancillas via uncomputation, reducing the total ancilla count at the expense of increased circuit depth.

MUX Branching. When node v has a collapsed edge controlled by reading qubit r_j, the dependency d_v must be routed to child u_0 (when $r_j = 0$) or u_1 (when $r_j = 1$). This is implemented by controlled-copy gates:

$$\text{Color 0:} \quad \text{For each bit } i, \quad \text{Toffoli}(\bar{r}_j, d_v[i], d_{u_0}[i]) \tag{1}$$

$$\text{Color 1:} \quad \text{For each bit } i, \quad \text{Toffoli}(r_j, d_v[i], d_{u_1}[i]) \tag{2}$$

where $\bar{r}_j$ is implemented by bracketing with X gates. When the reading register is in superposition, the controlled-copy gates act coherently on each computational-basis branch: the $|r_j = 0\rangle$ component routes to u_0, while the $|r_j = 1\rangle$ component routes to u_1.

This is the key architectural advantage: the DLDS branching structure directly controls the MUX routing, so the circuit gives a factored reversible representation of the 2^k readings. No explicit path enumeration is needed.

Remark 3 (Multi-Color Decomposition)**.** *If a branching point has $c > 2$ colors, it can be decomposed into $\lceil \log_2 c \rceil$ nested binary branches, each controlled by a fresh reading qubit. This introduces $O(\log c)$ additional qubits per branching point. Every combination of reading qubit values maps to a valid DLDS path. The cascaded example uses only binary branching ($c = 2$); we leave experimental evaluation of the multi-color case to future work.*

Worked Example: Dependency Trace. Table 1 traces the compilation on the cascaded DLDS of Figure 1. Each hypothesis H_i initializes $d_{H_i} = e_i$. At each level, the reading qubit r_j controls where the branching node's dependency vector is routed via MUX; the receiving child then OR-assigns its major premise. Intermediate dependency vectors vary widely across readings, but the OR accumulation at the root guarantees convergence: $d_{A_5} = 11111$ for all 8 readings.

Table 1. Dependency trace for the cascaded DLDS of Figure 1. Each row is one reading; $d_{A_5} = 11111$ in all cases.

$r_0r_1r_2$	d_{A_2}	$d_{A_2 \supset A_3}$	d_{A_3}	$d_{A_3 \supset A_4}$	d_{A_4}	$d_{A_4 \supset A_5}$	d_{A_5}
000	11000	00100	11100	00010	11110	00001	11111
001	11000	00100	11100	00010	00010	11101	11111
010	11000	00100	00100	11010	11110	00001	11111
011	11000	00100	00100	11010	11010	00101	11111
100	01000	10100	11100	00010	11110	00001	11111
101	01000	10100	11100	00010	00010	11101	11111
110	01000	10100	10100	01010	11110	00001	11111
111	01000	10100	10100	01010	01010	10101	11111

Deferred Discharge Under Greedy-Discharge Assumptions. In standard Natural Deduction, a derivation is closed when the root's dependency set is empty: each $\supset$I rule clears its hypothesis bit, and the final bitvector is all-zeros. Our compiler does not implement the final $\supset$I suffix explicitly. Instead, it uses a deferred-discharge test at the root of the reading-dependent branching/elimination fragment.

Definition 4 (Admissible DLDS for Deferred Discharge)**.** *A DLDS $\mathcal{D}$ is* admissible for deferred discharge *if every reading of $\mathcal{D}$ decomposes into: (i) a reading-dependent prefix*

consisting only of hypotheses, branching, and ⊃E steps; (ii) a reading-independent suffix σ consisting only of ⊃I steps; and (iii) a fixed set H_{dis} of hypotheses discharged by σ, such that every dependency bit reaching the root of the prefix belongs to H_{dis}. The compiler below is proved correct for this class, not for arbitrary DLDSs.

This is the regime relevant to the normalized, greedy-discharge DLDSs used in our examples. By saying that the introduction steps occur below the branching structure, we mean that after any reading-dependent elimination and routing choices have been fixed, all readings share the same final suffix of ⊃I rules leading to the conclusion formula. Thus the discharged hypotheses are determined by the common conclusion suffix, rather than by the particular reading. In particular, every DLDS produced by HC from a normalized implicational proof with greedy discharge satisfies this condition in the setting considered here, since normalization places the ⊃I phase after the elimination and branching prefix used by the compiler.

Lemma 5 (Deferred-Discharge Equivalence)**.** *Let $\mathcal{D}$ be admissible for deferred discharge, and let d_{root} be the dependency bitvector at the root of the reading-dependent prefix. Then a reading is closed if and only if $d_{\mathrm{root}}[i] = 1$ for every $i \in H_{\mathrm{dis}}$.*

Proof sketch. If some $i \in H_{\mathrm{dis}}$ is missing at the root, then the common introduction suffix attempts to discharge a hypothesis that is absent from that reading, so the reading is not closed. Conversely, if every bit in H_{dis} is present, then each step of the common suffix clears its corresponding hypothesis bit. Because admissibility requires all root dependencies to belong to H_{dis}, no residual bit remains after the suffix, and the reading closes. □

The root check therefore verifies $\forall\, i \in H_{\mathrm{dis}} : d_{\mathrm{root}}[i] = 1$ and sets $out = 1$ if any required bit is missing. This is implemented by applying X gates to each required bit position, a multi-controlled Toffoli targeting out, then X gates to restore the register.

Correctness and Complexity.

Theorem 6 (Oracle Correctness)**.** *For a DLDS $\mathcal{D}$ admissible for deferred discharge, with k binary branching points, the constructed circuit $C_{\mathcal{D}}$ satisfies:*

$$C_{\mathcal{D}}\left(\frac{1}{\sqrt{2^k}}\sum_{r\in\{0,1\}^k}|r\rangle|0\rangle|0\rangle\right) = \frac{1}{\sqrt{2^k}}\sum_{r}|r\rangle|\varphi_r\rangle|f(r)\rangle$$

where φ_r encodes the dependency registers for reading r, and $f(r) = 1$ iff reading r fails the deferred-discharge criterion, equivalently by Lemma 5 iff reading r is not closed.

Proof sketch. By induction on the topological order of nodes. For each node v and reading r, the dependency register d_v in the $|r\rangle$ branch contains the set of hypothesis indices used in the sub-derivation at v under reading r.

Base case: Top-formulas contribute their characteristic bit via X gates, independent of r.

Inductive step: At an ⊃E node v with premises v_1, v_2, the OR-assign computes $d_v = d_{v_1} \vee d_{v_2}$ coherently on each computational-basis branch of the superposition. At a branching node with reading qubit r_j, the controlled-copy gates route d_v to child u_0 in the $|r_j = 0\rangle$ branch and to u_1 in the $|r_j = 1\rangle$ branch. Since the Toffoli gate acts as identity when its control is $|0\rangle$, each branch evolves independently.

The root check correctly identifies readings that violate the deferred-discharge criterion. Lemma 5 then identifies these branches exactly with the non-closed readings of $\mathcal{D}$. □

Corollary 7. *Under the hypotheses of Theorem 6, $\mathcal{D}$ is valid iff $f(r) = 0$ for all $r \in \{0,1\}^k$.*

Proposition 8 (Polynomial Construction)**.** *The circuit $C_{\mathcal{D}}$ has $O(n^2)$ qubits, $O(n^2)$ gates, and $O(n^2)$ depth, and is constructible in polynomial time.*

Proof. Each of the $O(n)$ nodes contributes $O(t) = O(n)$ gates (either X gates for initialization, or CX/Toffoli triples for OR-assign, or controlled-copy Toffolis for MUX). The qubit count is $k + n \cdot t + e \cdot t + 1 = O(n^2)$. □

4. Grover Integration

The oracle $C_{\mathcal{D}}$ can be wrapped in a standard Grover loop over the reading register. To use Grover's phase kickback, we prepare the output qubit in $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ so that the oracle's action $|x\rangle|-\rangle \mapsto (-1)^{f(x)}|x\rangle|-\rangle$ applies a phase flip on invalid readings.

Each Grover iteration consists of four steps: **Compute:** apply all oracle gates (initialization, OR-assign, MUX routing, root check); **phase mark:** the root check flips out for invalid readings, and via phase kickback with $|-\rangle$ this becomes a phase flip; **uncompute:** apply all oracle gates in reverse order, which restores all dependency and ancilla registers to $|0\rangle$ because the circuit consists entirely of self-inverse gates (X, CNOT, Toffoli); **diffusion:** apply $2|\psi\rangle\langle\psi| - I$ on the reading register.

Majority/Minority Amplification. If M' of the $N = 2^k$ readings are invalid, standard Grover analysis gives $O(\sqrt{N/M'})$ iterations when M' is known and nonzero. In a nearly-valid DLDS, this means that a small set of bad readings can in principle be amplified from the uniform reading superposition.

The optimal number of iterations is $k_{\text{opt}} = \lfloor \frac{\pi}{4} \cdot \sqrt{N/M'} \rfloor$ when M' is known.

Concrete Example. For the synthetic almost-valid DLDS with $k = 3$ reading variables ($N = 8$) and exactly $M' = 1$ invalid reading, the Grover rotation angle is $\theta = \arcsin\sqrt{M'/N} = \arcsin\sqrt{1/8} \approx 0.3614$ rad, giving $k_{\text{opt}} = \lfloor \pi/(4\theta) \rfloor = 2$. After two iterations the probability of measuring the invalid reading is $\sin^2(5\theta) \approx 0.97$, amplifying the $1/8$ minority to approximately 97% detection probability.

Unknown number of invalid readings. When M' is not known in advance, which is the relevant regime for the $\mathrm{coNP} \subseteq \mathrm{QCMA}$ protocol, where the verifier does not know whether the DLDS is valid, the BBHT strategy [Boyer et al. 1998] can be used with the same oracle by varying the number of Grover iterations adaptively. This changes only the classical control loop, not the compiled circuit itself. We do not evaluate BBHT experimentally in this paper.

5. Experimental Results

We implemented the compiler in Python using Qiskit and validated archived circuits on the AerSimulator with the matrix product state (MPS) method. The reported data come from two complementary sources: hand-built DLDS benchmarks that isolate branching

behavior, and pipeline-derived DLDSs produced from proof-generation and horizontal-compression stages. We emphasize that these experiments validate the raw compiled oracle, not the full Grover-amplified search loop. The reported shot distributions are obtained by preparing the reading register in uniform superposition, applying the oracle, and measuring the output flag and reading register. Thus, one invalid reading among eight produces approximately $1/8$ invalid shots before amplitude amplification, consistent with Section 4.

Synthetic DLDS Benchmarks. The synthetic examples are designed to validate the reading semantics of the compiler independently of proof generation. Two archived instances are particularly informative.

The *almost-valid* DLDS has 17 nodes, 5 dependency bits, and 3 reading variables. Its compiled circuit uses 154 qubits at depth 84. Exactly one reading, $r = 111$, is invalid; the archived simulation report records $3574/4096$ valid shots and $522/4096$ invalid shots, in line with the expected $7/8$ versus $1/8$ split.

The *cascaded-invalid* DLDS has 10 nodes, 4 dependency bits, and 3 reading variables. Its compiled circuit uses 60 qubits at depth 46. Here only one reading is valid; the archived report records $524/4096$ valid shots and $3572/4096$ invalid shots, again matching the intended reading distribution.

Pipeline-Derived HC Instances. To validate the end-to-end relevance of the construction, we also consider instances obtained from the graph-to-proof-to-HC pipeline. These examples are not hand-built: they arise from proof-generation and compression before the quantum compilation stage.

The labels `2v_no_hc` and `3v_no_hc` denote built-in non-Hamiltonian test graphs from the pipeline: respectively, a 2-vertex 1-edge graph (a single edge v_1v_2) and the 3-vertex path $P_3 = v_1v_2v_3$ (edges v_1v_2 and v_2v_3), both with no Hamiltonian cycle.

For the archived instance `2v_no_hc`, the compressed DLDS has 29 nodes, 10 dependency bits, and no reading variables. The compiled circuit uses 581 qubits at depth 109, and its unique reading is valid.

For the archived instance `3v_no_hc`, the compressed DLDS has 92 nodes, 29 dependency bits, and 2 reading variables. The compiled circuit uses 5310 qubits at depth 465. All four readings are valid, confirming that the end-to-end pipeline produces a DLDS whose reading semantics is correctly captured by the compiled oracle. The qubit count is dominated by the $n \cdot t + e \cdot t$ dependency-and-ancilla term in the $O(n^2)$ bound of Proposition 8: with $n = 92$ and $t = 29$, the per-node dependency registers alone account for roughly 2700 qubits, with the remainder consumed by per-OR ancillas. This isolates ancilla recycling as the natural target for future optimization. The validation procedure for these pipeline-derived instances was semantic rather than hardware-oriented: we executed each compiled oracle with the MPS simulator and checked the measured output flag over the available readings. In `3v_no_hc`, the two reading variables determine four readings, all of which returned accepting outcomes.

The large qubit counts should not be interpreted as evidence of near-term hardware feasibility, nor do we claim that arbitrary DLDS-compiled circuits of similar size are efficiently simulable by MPS. The experiments validate oracle semantics; hardware

execution and simulator scalability are left for future work.

Table 2. Archived experimental results used in this paper. "Source" distinguishes hand-built DLDS benchmarks from pipeline-derived compressed proofs.

Instance	Source	Nodes	t	k	Qubits	Depth	Validity
Almost-valid	Synthetic	17	5	3	154	84	7/8 valid
Cascaded-invalid	Synthetic	10	4	3	60	46	1/8 valid
`2v_no_hc`	Pipeline	29	10	0	581	109	1/1 valid
`3v_no_hc`	Pipeline	92	29	2	5310	465	4/4 valid

6. Discussion

Complexity-Theoretic Motivation. The construction is relevant to broader investigations on quantum verification with classical compressed proof objects. If a family of tautologies admits polynomial-size DLDS certificates with suitably controlled branching, then the compiler developed here provides a candidate quantum verification primitive for their reading semantics.

At the same time, the present paper is deliberately narrower than any complexity-class claim. We do not prove that broad tautology families admit such DLDS certificates, and we do not settle the status of any proposed global classical validity criterion for DLDSs. Our contribution is the reversible verification architecture itself: given a DLDS whose intended semantics is its family of readings, we show how to compile those readings into a quantum oracle.

6.1. Connection to coNP $\subseteq$ QCMA

Our construction provides the concrete quantum verification machinery for the protocol envisioned in the $\mathrm{coNP} \subseteq \mathrm{QCMA}$ conjecture. In the setting studied here, the DLDS serves as a polynomial-size classical certificate, and our compiler produces a quantum verifier for its reading semantics. If the DLDS encodes a valid proof of a tautology, all readings pass; if any reading fails, Grover can find it with high probability using $O(\sqrt{2^k})$ oracle calls.

The protocol operates as follows: (1) the prover submits a DLDS $\mathcal{D}$ of polynomial size as a classical witness; (2) the verifier compiles $\mathcal{D}$ into a quantum circuit in polynomial time; (3) the verifier runs Grover's algorithm with $O(\sqrt{2^k})$ iterations (using the BBHT protocol when M' is unknown); (4) if an invalid reading is found, the verifier rejects; otherwise, it accepts.

For the family of non-Hamiltonicity tautologies α_G, the number of branching points satisfies $k = O(\log n)$ where $n = |V(G)|$ [Gordeev and Haeusler 2019]. This means the Grover search requires only $O(\sqrt{2^{O(\log n)}}) = O(\mathrm{poly}(n))$ oracle calls, making the verification step efficient. The full inclusion $\mathrm{coNP} \subseteq \mathrm{QCMA}$ additionally requires that every co-tautology admits a polynomial-size DLDS whose validity can be checked by this protocol. This question reduces to the proof-theoretic properties of compressed Natural Deduction, specifically, whether HC always produces polynomial-size DLDS, which is the subject of research [Haeusler et al. 2025, Gordeev and Haeusler 2019, Haeusler 2022]. Our work provides the necessary quantum infrastructure; the sufficiency of DLDS as

proof certificates is a separate, purely classical question.

Limitations. Our experiments run on a classical MPS simulator rather than quantum hardware; the pipeline-derived examples already require 581 and 5310 qubits, so the current evidence supports oracle-semantics validation rather than hardware feasibility. We do not present an experimental BBHT study or systematic hardware-oriented analysis.

7. Related Work

Aharonov and Naveh [Aharonov and Naveh 2002] introduced QCMA and posed the question of whether $\mathrm{NP} \subseteq \mathrm{QCMA}$; our work is motivated by the dual inclusion $\mathrm{coNP} \subseteq \mathrm{QCMA}$ and studies a concrete DLDS-based verification protocol relevant to that direction. Grover's algorithm [Grover 1996] and its generalization [Boyer et al. 1998] provide the search framework used in our construction. Proof complexity and computational complexity have a long interaction [Cook and Reckhow 1979, Krajíček 1995]; explicit quantum circuit constructions from proof-theoretic structures, however, appear to be novel.

Beyond QCMA, quantum verification of classical certificates has been studied in several settings. Bennett's analysis of reversible computation [Bennett 1989] underlies the uncompute strategy we use for OR-assign and MUX routing, and informs our ancilla management, though our compiler is driven by DLDS structure rather than by a classical program. Within quantum proof verification, prior work has largely targeted the soundness of quantum protocols themselves rather than the compilation of structured classical proof objects into circuits; to our knowledge, an explicit reversible compilation of compressed Natural Deduction proofs has not previously been developed.

On the proof theory side, Statman's speedup theorem [Statman 1979] establishes the exponential gap between cut-free and cut-full proofs that motivates DLDS compression. Haeusler and collaborators [Gordeev and Haeusler 2019, Haeusler et al. 2025] developed the HC algorithm and the DLDS framework, including proposals for global validity criteria based on dependency flow through compressed derivations. Our approach is orthogonal: rather than assuming a specific global criterion, we compile the reading semantics directly into a reversible circuit. To our knowledge, the explicit compilation of proof-theoretic compression parameters such as branching points arising from horizontal compression into quantum-circuit primitives has not been previously explored.

8. Conclusion

We presented a polynomial-time compiler from reading-based DLDS semantics to quantum verification circuits, proved correctness under the deferred-discharge assumptions of Section 3, and validated the construction on synthetic and pipeline-derived simulator benchmarks.

The construction is, to our knowledge, the first explicit reversible compilation of compressed Natural Deduction proofs into quantum verification oracles. The MUX-routing primitive turns out to be a natural match for the colored-edge structure produced by Horizontal Compression: rather than enumerating the 2^k readings, the circuit represents them implicitly in a factored form whose size is polynomial in the compressed proof. We view this as evidence that proof-theoretic compression and reversible circuit compilation interact more cleanly than the asymptotic $O(n^2)$ bound alone would sug-

gest, and that other proof-structural artifacts of compression may admit similar quantum treatments.

Future work includes Grover/BBHT experiments, ancilla reduction, more general discharge patterns, and extensions beyond the implicational fragment.

Declaração sobre uso de IA. Ferramentas de inteligência artificial generativa foram utilizadas durante a preparação deste trabalho para auxiliar na revisão textual e na edição do manuscrito. Todo o conteúdo técnico, resultados, provas e contribuições científicas são de inteira responsabilidade dos autores.

References

Aharonov, D. and Naveh, T. (2002). Quantum np: a survey. arXiv:quant-ph/0210077.

Bennett, C. H. (1989). Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776.

Boyer, M., Brassard, G., Høyer, P., and Tapp, A. (1998). Tight bounds on quantum searching. *Fortschr. Phys.*, 46(4–5):493–505.

Cook, S. A. and Reckhow, R. A. (1979). The relative efficiency of propositional proof systems. *J. Symbolic Logic*, 44(1):36–50.

Gentzen, G. (1935). Investigations into logical deduction. *Math. Z.*, 39:176–210.

Gordeev, L. and Haeusler, E. H. (2019). Proof compression and np versus pspace. *Studia Logica*, 107(1):53–83.

Gordeev, L. and Haeusler, E. H. (2022). Proof compression and np versus pspace ii: Addendum. *Bull. Sect. Log.*, 51(2):197–205.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proc. 28th ACM STOC*, pages 212–219.

Haeusler, E. H. (2015). Propositional logics complexity and the sub-formula property. *EPTCS*, 179:1–16.

Haeusler, E. H. (2022). Exponentially huge natural deduction proofs are redundant: Preliminary results on $m_{\supset}$. *FLAP*, 9(1):287–326.

Haeusler, E. H., Barros Junior, J. F. C., and Callou, R. (2025). On the horizontal compression of dag-derivations in minimal purely implicational logic. arXiv:2206.02300.

Krajíček, J. (1995). *Bounded Arithmetic, Propositional Logic and Complexity Theory*. Cambridge Univ. Press.

Prawitz, D. (1965). *Natural Deduction: A Proof-Theoretical Study*. Almqvist & Wiksell.

Statman, R. (1979). Lower bounds on herbrand's theorem. *Proc. Amer. Math. Soc.*, 75(1):104–107.