

Towards Datalog on Quantum Annealers: Compiling Recursive Logic Programs with Bottom-up Semantics to 2-local Ising Models

Bruno Rucy Carneiro Alves de Lima^{1,2,*}, Victor Henrique Cabral Pinheiro¹,
Evgenii Dolzhkov¹ and Joseph Haske³

¹University of Tartu, Tartu, Estonia

²Lucidarium Systems, <https://lucidarium.systems>

³Independent Researcher

Abstract

Quantum annealers solve problems by finding the lowest-energy (ground) state of a programmable physical system, a 2-local Ising model, whose energy function is the Hamiltonian. We compile recursive Datalog programs into such models so that the ground state projects onto the program's minimal Herbrand model. The compiler has four stages: binarization, grounding, reduction to a Min-Ones SAT formula, and Ising encoding. Each rule becomes an energy penalty on the one assignment that violates it, and a small uniform cost on every true atom selects the minimal model. We contribute both in theory and in practice with per-stage correctness lemmas and a correspondence theorem, verified in Lean 4, establishing that the ground state of the compiled model projects onto the program's minimal Herbrand model. We map the compiled models onto the topologies of commercial annealers and characterize, under classical and simulated-quantum annealing, whether and when that certified ground state is attained.

Keywords

Datalog, Quantum Annealing, Ising Model, Logic Programming, Bottom-up Evaluation, Minimal Model

1. Introduction

A quantum annealer [1, 2] solves one kind of problem: given an energy function over binary variables, find an assignment of minimum energy. Programming the machine means choosing that energy function, its *Hamiltonian*.

Nonrecursive, query-directed (top-down) logic programs, specifically a subset of Prolog [3], have been compiled to *2-local Ising models* [4]: energy functions over ± 1 -valued variables, called *spins*, whose interactions are at most pairwise, the form that D-Wave's annealers minimize [5].

A Datalog [6] program has a least model [7], and an Ising model has least-energy (ground) states. We show that the two minima can be made to coincide: the least model of a program becomes the ground state of a compiled Ising model. Bottom-up evaluation stays tractable in data complexity, so the optimum of every compiled instance is known exactly and solvers are measured against it, not against a sampler's estimate.

We compile bottom-up *recursive* Datalog to 2-local Ising models. Our contributions are:

1. a compiler in which each rule becomes an energy penalty on the one assignment that violates it, alternative rules deriving the same head require no dedicated construct, and minimality follows from a single uniform cost on every true atom
2. per-stage correctness lemmas and a correspondence theorem establishing that the ground state of the compiled model projects onto the program's minimal Herbrand model, with proofs in

Datalog 2.0 2026: 6th International Workshop on the Resurgence of Datalog in Academia and Industry, September 7, 2026, Klagenfurt, Austria

*Corresponding author.

✉ bruno.rucy.carneiro.alves.de.lima@ut.ee (B. R. C. A. d. Lima); victor.pinheiro@ut.ee (V. H. C. Pinheiro); evgenii.dolzhkov@ut.ee (E. Dolzhkov); joseph.l.haske@gmail.com (J. Haske)

ORCID 0000-0002-2679-6639 (B. R. C. A. d. Lima); 0000-0003-1112-2620 (V. H. C. Pinheiro)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Lean 4 [8]¹

3. a reference implementation in Python that compiles arbitrary programs and exactly solves instances small enough for exhaustive verification, with an empirical evaluation of its hardware footprint and of solver recovery under classical and simulated-quantum annealing.¹

Related work. Before quantum hardware, propositional satisfiability was already modelled as energy minimization: formulas map to quadratic energy functions whose minima are the satisfying assignments [9], and logic programs were encoded so that the minima are their models, with no preference among them [10]. On annealers, Ising formulations are catalogued for many NP problems [11], and SAT and MaxSAT encodings target sparse annealer hardware, with the maximum-weight assignments at minimum energy [12]. A nonrecursive fragment of Prolog compiles to Hamiltonians whose ground states are the answers to a single query [5]. Answer Set Programming [13] has also been paired with quantum computation, though by Grover search [14] on gate-based machines, with no energy landscape involved [15]. On the classical side, computing the minimal models of a propositional theory is a well-studied problem, with efficient algorithms near the Horn fragment [16].

In each energy encoding above, the bottom of the landscape holds a solution set: the satisfying assignments, the models, or the answers to a query. None distinguishes the least model, the object bottom-up Datalog computes. Our propositional objective is itself standard: the compiler reduces the ground program to Min-Ones SAT (satisfiability with the fewest true atoms), a problem whose approximability admits a complete classification [17]. Horn formulas are one of the three polynomial-time cases of that classification because the least model is the cardinality minimum, and the compiler grew out of that observation. We are not aware of any prior mapping of Datalog onto Min-Ones SAT, and the contribution is the certified compiler built around that framing. The compiler starts from recursive Datalog, whose rules carry variables where the encodings above are propositional, and makes grounding an explicit, model-preserving stage, so the ground state projects onto the query-independent least Herbrand model of the source program. We claim no advance in grounding itself: the stage is kept simple on purpose, it is certified because the correctness theorem must span it, and the optimized grounders of Answer Set Programming’s ground-and-solve architecture would shrink it and apply here unchanged. The Ising realization uses only standard penalty energies and auxiliary spins [12, 18].

2. Background

2.1. Datalog

Syntax. A Datalog program P consists of *rules* $h \leftarrow b_1, \dots, b_n$ where the head h and each body atom b_i are predicates applied to terms. Terms are *variables* (X, Y) or *constants* (a, b). Every rule is *safe*: each head variable also occurs in the body. A rule is *recursive* when its head predicate occurs in a body, and a *ground fact* is a rule with no variables and an empty body, such as $Edge(a, b)$. The transitive closure program used throughout is

$$Reach(X, Y) \leftarrow Edge(X, Y) \quad Reach(X, Z) \leftarrow Edge(X, Y), Reach(Y, Z).$$

Semantics. Given an *extensional database* (EDB) $\mathcal{D}$ of input facts, the *Herbrand universe* $\mathcal{U}$ is the set of constants in $P \cup \mathcal{D}$ and the *Herbrand base* $\mathfrak{H}$ is the set of ground atoms over $\mathcal{U}$. A program is a set of definite (Horn) clauses and has a unique *minimal model*, the least Herbrand model, $\mathfrak{M}_P(\mathcal{D}) \subseteq \mathfrak{H}$: the smallest set such that

1. $\mathcal{D} \subseteq \mathfrak{M}_P(\mathcal{D})$ and
2. for every rule and every substitution θ from the rule’s variables to $\mathcal{U}$, if $\{b_1\theta, \dots, b_n\theta\} \subseteq \mathfrak{M}_P(\mathcal{D})$ then $h\theta \in \mathfrak{M}_P(\mathcal{D})$.

¹The artifact is public at <https://github.com/brurucy/quantum-datalog>, with the Lean development in `lean/`, the compiler in `src/`, and the notebooks and generated data beside them.

The minimal model is the least fixpoint of the immediate-consequence operator T_P , which exists by Knaster–Tarski since T_P is monotone [19]. This least-model reading of a definite program is standard [7, 20].

2.2. Bottom-up vs. Top-down Evaluation

Top-down. Given a *query* such as $\text{Reach}(a, d)?$, evaluation searches backward: it finds rules whose head unifies with the query, recursively proves the body, and backtracks on failure.

Bottom-up. Evaluation starts from $\mathcal{D}$, applies all rules to derive new facts, and repeats to a fixpoint. For $\mathcal{D}_0 = \{\text{Edge}(a, b), \text{Edge}(b, c), \text{Edge}(c, d)\}$: Iteration 1 (rule 1) derives $\text{Reach}(a, b), \text{Reach}(b, c), \text{Reach}(c, d)$. Iteration 2 (rule 2) adds $\text{Reach}(a, c), \text{Reach}(b, d)$, iteration 3 adds $\text{Reach}(a, d)$, and iteration 4 adds nothing (fixpoint). $\mathfrak{M}_P(\mathcal{D}_0)$ contains these 6 facts and the 3 edges, independently of any query.

2.3. 2-local Ising Models

An Ising model over n binary spins $\sigma = (\sigma_1, \dots, \sigma_n) \in \{-1, +1\}^n$ has Hamiltonian

$$H(\sigma) = \sum_{i=1}^n \eta_i \sigma_i + \sum_{i < j} J_{ij} \sigma_i \sigma_j,$$

with *local fields* $\eta \in \mathbb{R}^n$ and *couplings* $J \in \mathbb{R}^{n \times n}$, realized in hardware as programmable couplers. The *ground state* $\sigma^* = \arg \min_{\sigma} H(\sigma)$ has minimal energy. Here *2-local* means that H has at most pairwise interactions, a bound on the degree of H .

Physically, the model is a collection of n magnets, each free to point up ($\sigma_i = +1$) or down ($\sigma_i = -1$). A local field biases one magnet: at $\eta_i < 0$ the term $\eta_i \sigma_i$ is lowest when $\sigma_i = +1$, so the field pulls the spin up, and at $\eta_i > 0$ it pulls the spin down. A coupling ties a pair: at $J_{ij} < 0$ the term $J_{ij} \sigma_i \sigma_j$ is lowest when the two spins agree, and at $J_{ij} > 0$ when they differ. The Hamiltonian adds up these local preferences, which in general conflict, and the ground state is their best global compromise. A quantum annealer is a physical machine whose spins relax toward this minimum, so programming it consists of choosing η and J [1, 2].

Hardware constrains that choice in two ways. The first is expressiveness: a constraint coupling three or more spins at once has no direct 2-local form, so encoding it requires auxiliary variables, which Section 3 introduces at the rule level and at the energy level. The second is connectivity: the nonzero couplings of a compiled model form a graph over its spins, and this graph is dense, while a chip offers couplers only between qubits physically adjacent in its fixed layout of bounded degree (Chimera, Pegasus, Zephyr [21, 22]). The bridge is *minor-embedding*: each logical spin is realized as a *chain* of physical qubits coupled strongly enough to act as one spin, laid out so that every required coupling meets an available coupler [23]. A chain spends several qubits on one logical spin, so embedding inflates the qubit count. We use the spin encoding $\sigma_f = +1$ for *true* and $\sigma_f = -1$ for *false*.

3. Compilation Framework

We realize our compiler as four typed stages, each a transformation between categories of decreasing abstraction:

$$\mathcal{P} \xrightarrow{\mathcal{C}_1} \mathcal{P}^{\text{bin}} \xrightarrow{\mathcal{C}_2} \mathcal{P}^{\text{gnd}} \xrightarrow{\mathcal{C}_3} \Phi \xrightarrow{\mathcal{C}_4} (\eta, J).$$

A program $P \in \mathcal{P}$ binarizes to a program $\mathcal{C}_1(P) \in \mathcal{P}^{\text{bin}}$ whose rule bodies hold at most two atoms. Grounding sends that to a variable-free program in $\mathcal{P}^{\text{gnd}}$, Stage 3 reads off a propositional formula Φ in conjunctive normal form (CNF), and Stage 4 emits the local fields η and couplings J of a 2-local Ising model. Every object denotes the same minimal Herbrand model $\mathfrak{M}_P(\mathcal{D})$, and we design each stage to preserve that denotation.

We develop the four stages in order, each with its preservation lemma, and state two correctness theorems. We illustrate the stages with the transitive closure program and $\mathcal{D}_0$.

3.1. Stage 1: Binarization ($\mathcal{C}_1: \mathcal{P} \rightarrow \mathcal{P}^{\text{bin}}$)

This stage rewrites rules with $n > 2$ body atoms into rules with at most 2 body atoms by introducing fresh auxiliary predicates. For $r = (h, b_1, \dots, b_n)$, fold the body right-to-left:

$$\begin{aligned} \mathcal{B}(r) = & \{(\text{Aux}_1(V_1), b_{n-1}, b_n)\} \\ & \cup \{(\text{Aux}_k(V_k), b_{n-k}, \text{Aux}_{k-1}(V_{k-1})) \mid 2 \leq k \leq n-2\} \\ & \cup \{(h, b_1, \text{Aux}_{n-2}(V_{n-2}))\} \end{aligned}$$

where Aux_k summarises $b_{n-k} \wedge \dots \wedge b_n$ and carries exactly the still-needed variables, with $\text{vars}(\cdot)$ the set of variables occurring in its argument:

$$V_k = \text{vars}(\{b_{n-k}, \dots, b_n\}) \cap (\text{vars}(h) \cup \text{vars}(\{b_1, \dots, b_{n-k-1}\})),$$

preserving variable dependencies and rule safety (every head variable still occurs in the body). Each step of the fold is a *split*: it replaces the two-atom tail block by one fresh auxiliary atom, so a rule with n body atoms takes $n-2$ splits.

Example. Both rules are already binary, so $\mathcal{C}_1(P_0) = P_0$, and the two splits of $Q(X, W) \leftarrow S(X, Y), T(Y, Z), U(Z, W), V(W)$ give

$$\text{Aux}_1(Z, W) \leftarrow U(Z, W), V(W) \quad \text{Aux}_2(Y, W) \leftarrow T(Y, Z), \text{Aux}_1(Z, W) \quad Q(X, W) \leftarrow S(X, Y), \text{Aux}_2(Y, W).$$

Lemma 1 (Binarization Correctness). *For any program P and EDB $\mathcal{D}$ whose own predicates avoid the auxiliary names, given one fresh auxiliary predicate name per split, restricted to the non-auxiliary predicates, $\mathfrak{M}_P(\mathcal{D}) = \mathfrak{M}_{\mathcal{C}_1(P)}(\mathcal{D})$.*

PROOF (SKETCH). Each split replaces the current two-atom tail block by a fresh atom $\text{Aux}(V_k)$, where V_k are the block variables that also occur elsewhere in the rule or in the head, and adds the single rule deriving $\text{Aux}(V_k)$ from that block. The proof turns on the soundness of this projection: a grounding satisfies the original body exactly when some extension over the block-local variables makes $\text{Aux}(V_k)$ derivable, which is their existential elimination. The auxiliary predicate is fresh, so the added rules derive only auxiliary atoms and the step is a conservative extension. Induction over the fold, one split per step, carries the equality to the fully binarized program, and the minimal model is unchanged on the non-auxiliary predicates.

3.2. Stage 2: Grounding ($\mathcal{C}_2: \mathcal{P}^{\text{bin}} \rightarrow \mathcal{P}^{\text{gnd}}$)

This stage instantiates variables to constants in $\mathfrak{U}$. The atoms that grounding can ever need form the *Herbrand base subset*, or *relevant base*,

$$\mathfrak{B} = \mathcal{D} \cup \{p(\vec{c}) \mid p \text{ a head predicate of } P, \vec{c} \in \mathfrak{U}^{ar(p)}\},$$

the EDB facts together with every ground atom of an intensional (head) predicate. An extensional predicate contributes only its facts. An intensional predicate is instantiated over $\mathfrak{U}$, so $|\mathfrak{B}|$ is $|\mathfrak{U}|^{\Theta(ar)}$ whether or not the atoms are derivable, the standard combined-complexity blowup [24] that demand-driven grounding in the magic-sets tradition [25] would control. For each rule r we keep the *relevant* ground instances

$$\mathcal{G}(r) = \{r\theta \mid \theta: \text{vars}(r) \rightarrow \mathfrak{U}, \text{ every body atom of } r\theta \text{ lies in } \mathfrak{B}, \text{ and the head of } r\theta \text{ is not among its body atoms}\},$$

and emit each EDB fact as a bodiless ground rule. A ground instance with a body atom outside $\mathfrak{B}$, for instance an extensional atom that is not a fact, can never fire and is omitted. A ground instance whose head occurs in its own body fires only when its head already holds, so it derives nothing and is omitted.

Restricting to $\mathfrak{B}$ is lossless. The decisive fact is what can *never* appear: every atom produced during bottom-up evaluation is an EDB fact or carries an intensional head predicate, so it lies in $\mathfrak{B}$, and no

atom outside $\mathfrak{B}$ is ever derived. A ground instance discarded by the first condition has, by construction, a body atom outside $\mathfrak{B}$, an atom no iterate can contain, so it never fires in either the relevant or the full grounding.

Example. The stage receives the binarized program of Stage 1, which for the running example is P_0 unchanged. On $\mathcal{D}_0$ the Herbrand universe is $\{a, b, c, d\}$ and $\mathfrak{B}$ holds 19 atoms (the 3 edges and the 16 ground *Reach* atoms). Grounding then produces 18 ground rules, the constraints over those 19 atoms: the 3 EDB facts, the 3 instances of r_1 (one per edge), and the 12 instances of r_2 (each edge $Edge(x, y)$ paired with a target z), among them

$$Reach(a, b) \leftarrow Edge(a, b) \quad Reach(a, d) \leftarrow Edge(a, b), Reach(b, d).$$

The unrestricted grounding $\{r\theta \mid \theta: vars(r) \rightarrow \mathfrak{U}\}$ would instead list $|\mathfrak{U}|^{\#vars(r)}$ instances per rule, $16 + 64 + 3 = 83$ in all, almost every one dropped by the two conditions of $\mathcal{G}(r)$.

Lemma 2 (Grounding Correctness). *Grounding preserves the minimal model: $\mathfrak{M}_{\mathcal{C}_2(P^{bin})}(\mathcal{D}) = \mathfrak{M}_{P^{bin}}(\mathcal{D})$. Grounding introduces no auxiliary predicates, so the equality holds on the full Herbrand base.*

PROOF (SKETCH). In two steps. (i) *Full grounding preserves models.* A ground instance $r\theta$ is satisfied by an interpretation exactly when the binary rule r is satisfied by that interpretation under θ , so an interpretation models the full grounding iff it models P^{bin} , and the two share every model, hence the least one. (ii) *Relevance loses nothing.* The relevant grounding $\mathcal{C}_2(P^{bin}) = \bigcup_r \mathcal{G}(r)$ and the full grounding generate *term-by-term identical* bottom-up chains: both stay inside $\mathfrak{B}$, and on any interpretation I contained in $\mathfrak{B}$ their one-step closures $I \cup T(I)$ agree, writing T for each grounding's immediate-consequence operator. An instance present in the full but absent from the relevant grounding is dropped for one of two reasons. Either a body atom lies outside $\mathfrak{B}$, which such an interpretation cannot contain, so the instance never fires, or the head sits among its own body atoms, so whatever the instance fires from I was already in I . Identical chains reach the identical least fixpoint. Composing (i) and (ii) gives $\mathfrak{M}_{\mathcal{C}_2(P^{bin})}(\mathcal{D}) = \mathfrak{M}_{P^{bin}}(\mathcal{D})$. No auxiliary predicates are introduced, so the equality holds on the full Herbrand base.

3.3. Stage 3: Ground Program $\rightarrow$ Boolean Formula ($\mathcal{C}_3: \mathcal{P}^{gnd} \rightarrow \text{CNF}$)

Input to this stage is the *relevant* ground program $\mathcal{P}^{gnd} = \mathcal{C}_2(P^{bin})$ of Stage 2 (the 18-rule program for the running example, not the unrestricted grounding), which Lemma 2 certified has the same minimal model as P^{bin} . We record it as an explicit propositional formula. Each ground rule $h \leftarrow b_1, \dots, b_k$ becomes the Horn clause $\neg b_1 \vee \dots \vee \neg b_k \vee h$, a fact $h \leftarrow$ becomes the unit clause h , and $\mathcal{C}_3(P)$ is their conjunction Φ , a CNF over the ground atoms. Here the meaning of the object shifts from derivation to satisfaction, as a clause carries polarities and has no privileged head. An assignment satisfies Φ exactly when it models the ground program, so the minimal model is the assignment that satisfies Φ while setting the fewest atoms true. We therefore read the bottom-up answer as a **Min-Ones SAT** instance.

Min-Ones SAT is NP-complete in general [17], yet on Horn formulas the minimum coincides with the unique least model, computable in linear time [26]. What remains for Stage 4 is to build an energy whose unique minimizer is that least model.

The lemma is well known, and we record it in the form the correspondence proof composes.

Lemma 3 (Min-Ones SAT = minimal model). *For a ground program P and EDB $\mathcal{D}$, the satisfying assignments of $\Phi = \mathcal{C}_3(P)$ are exactly the models of P , and $\mathfrak{M}_P(\mathcal{D})$ is the unique Min-Ones SAT solution, since it satisfies Φ and is a proper subset of every other satisfying assignment.*

PROOF (SKETCH). A Horn clause $\neg b_1 \vee \dots \vee \neg b_k \vee h$ holds at I exactly when $\{b_1, \dots, b_k\} \subseteq I$ implies $h \in I$, so $I \models \Phi$ iff I models P . The minimal model is itself a model and lies inside every model, by the least-model property [7]. A proper subset of a finite set has strictly smaller cardinality, which makes $\mathfrak{M}_P(\mathcal{D})$ the unique Min-Ones solution.

Table 1

Gate energies before scaling by W , the uniform weight that scales every gate against the cardinality field. Each gate rests at its floor on every assignment that satisfies its rule and rises by the gap Δ on the unique violating assignment, marked *. The 2-IMPLY energies have the ancilla z minimized out.

$\text{FACT } (h \leftarrow)$		$\text{1-IMPLY } (h \leftarrow b)$			$\text{2-IMPLY } (h \leftarrow l, r)$			
h	E	b	h	E	l	r	h	E
F	+5 *	F	F	-1	F	F	F	-2
T	-5	F	T	-1	F	F	T	-2
		T	F	+3 *	F	T	F	-2
		T	T	-1	F	T	T	-2
					T	F	F	-2
					T	F	T	-2
					T	T	F	+2 *
					T	T	T	-2

3.4. Stage 4: 2-local Ising Encoding ($\mathcal{C}_4: \Phi \rightarrow \mathbb{R}^V \times \mathbb{R}^{V \times V}$)

We turn Φ into an energy whose unique minimizer is the least model that Stage 3 identified, and this is the main technical step of the compiler. Each clause of Φ is the Horn clause of a ground rule $h \leftarrow b_1, \dots, b_k$ with $k \leq 2$, that is, the implication $(b_1 \wedge \dots \wedge b_k) \rightarrow h$. Exactly one assignment of its atoms *violates* the rule: every body atom true and the head false. The rule's *gate* is an energy term over those atoms that takes one constant value, its *floor*, on every assignment satisfying the rule, and rises by a gap $\Delta > 0$ on the violating assignment. Three gate shapes cover $k \leq 2$, all standard penalty encodings [12], with the degree-lowering reduction supplying the auxiliary spin [18]. Table 1 lists their energies, and, writing $s_x = \sigma_x$ for the spin of atom x and η_x for the local field on that spin, the coefficients realizing them are:

FACT ($h \leftarrow$), **0-IMPLY**: $\eta_h := -5$, energy $-5 s_h$ ($\Delta = 10$). Forces h true.

1-IMPLY ($h \leftarrow b$): $\eta_b := +1$, $\eta_h := -1$, $J_{b,h} := -1$, energy $s_b - s_h - s_b s_h$ ($\Delta = 4$). Pairwise, with no extra spin needed.

2-IMPLY ($h \leftarrow l, r$): the violation $l \wedge r \wedge \neg h$ is a degree-3 term, lowered to a quadratic form with one *ancilla* spin z [18]: $\eta_z := +1$, $\eta_h := -1$, $J_{l,r} := +1$, $J_{z,l} = J_{z,r} = J_{z,h} := -1$ ($\Delta = 4$, with z minimised out).

For the running example's ground instance $\text{Reach}(a, d) \leftarrow \text{Edge}(a, b), \text{Reach}(b, d)$, the 2-IMPLY gate ties the two body spins, the head spin, and one fresh ancilla with these coefficients. The energy rests at the floor on every assignment except the violating one, $\text{Edge}(a, b)$ and $\text{Reach}(b, d)$ true with $\text{Reach}(a, d)$ false, where it rises by $\Delta = 4$.

Disjunction. A head derivable by several rules contributes one gate per rule on the shared head spin. Because $(a \rightarrow h) \wedge (b \rightarrow h)$ equals $(a \vee b) \rightarrow h$, the encoding expresses this disjunction of alternative bodies without a dedicated construct.

Minimality. An implication gate binds the head only when the rule's body holds, so a head with no satisfied body stays otherwise free. We add a uniform field $\eta_f += \varepsilon/2$ with $\varepsilon > 0$ on every atom spin. A fact atom is pinned by its gate, so its field term is a constant shift that moves no ground state. The reference compiler implements exactly this energy.

As spins are ± 1 , a field $\eta_f s_f$ swings by $2\eta_f$ between true and false, and $\eta_f = \varepsilon/2$ makes each true atom cost ε relative to false. Among the models of P , the objective becomes ε times the number of true atoms. The minimal model is a proper subset of every other model (Lemma 3), so it alone has the fewest true atoms and the field selects it.

Precision. Accumulated field offsets must stay below the penalty of any single violation, so we scale the gates by a weight W and keep $\varepsilon N < W$ for some N bounding the minimal model's size. Each gate's gap is at least $\Delta_{\min} = 4$ (Table 1), so a violation costs at least $4W$. Since $\mathfrak{M}_P(\mathcal{D}) \subseteq \mathfrak{B}$, the static choice $N = |\mathfrak{B}|$ works and is known before any evaluation. The compiler tightens it to $N = n_{\text{atoms}}$, the atom-spin count, which still bounds the minimal model because every minimal-model atom occurs

in the compiled program, and deploys $\varepsilon = 1$ with $W = n_{\text{atoms}} + 1$, so $N < W$ holds by construction. On every family evaluated here $n_{\text{atoms}} = |\mathfrak{B}|$. Relative to the gate scale, the selecting field is then $\varepsilon/(W \Delta_{\min}) = \Theta(1/n_{\text{atoms}})$: it shrinks as the atom count grows, and the evaluation quantifies what that costs on hardware.

3.5. Correctness

Theorem 1 (Ising Encoding, ε -minimality). *Let H be the Hamiltonian of Stage 4 for a ground program P at the deployed $\varepsilon = 1$, under $N < W$ for an integer weight W and some $N \geq |\mathfrak{M}_P(\mathcal{D})|$. Then H has a unique minimizing true-set (set of atoms assigned true), and the atom-projection of every ground state equals the minimal model: $\sigma_f = +1 \iff f \in \mathfrak{M}_P(\mathcal{D})$.*

PROOF (SKETCH). The model carries one atom spin per ground atom of P and one private ancilla per 2-IMPLY gate. For a true-set I write $\text{cost}(I) = \min_z H(\sigma_I, z)$ for the energy with the atom spins fixed by I and the ancillas minimized out, and let c_0 collect the per-gate floors and the constant part $-n_{\text{atoms}}/2$ of the uniform field. The constant c_0 is independent of I , shifts every energy equally, and so does not affect the argmin. No gate is violated exactly when I models P , by Lemma 3, in which case every gate rests at its floor and $\text{cost}(I) = c_0 + |I|$. When I fails to model P , some gate is violated and $\text{cost}(I) \geq c_0 + W$ (the physical rise is at least $W \Delta_{\min}$). The minimal model then sits strictly below every non-model, $\text{cost}(\mathfrak{M}_P(\mathcal{D})) = c_0 + |\mathfrak{M}_P(\mathcal{D})| \leq c_0 + N < c_0 + W \leq \text{cost}(I)$, so no non-model can be a minimizer.

Among the models, the minimal model is a proper subset of every other, so its cardinality term is strictly smaller and it is the unique minimizing true-set. Each 2-IMPLY ancilla takes whichever value minimizes its own gate. On a gate whose head is true with exactly one true body atom the two values tie, so the full configuration need not be unique, though the atom-projection is.

Foundedness on a cycle. The uniform field is what rejects self-supporting derivations, a behavior the acyclic running example cannot exhibit. Take the ground program $p \leftarrow q, q \leftarrow p$ with empty EDB. It has two models, $\emptyset$ and $\{p, q\}$. The second satisfies both rules, since each atom supports the other, yet it is unfounded, resting on nothing outside the cycle, and the least model is $\emptyset$. The compiled Hamiltonian, with the deployed $\varepsilon = 1$ and $W = N + 1 = 3$, assigns energy -7 to $\emptyset$ and -5 to $\{p, q\}$, so its unique ground state is $\emptyset$, ahead of $\{p, q\}$ by exactly 2ε , the cardinality gap times the field (verified with `dimod.ExactSolver`).

The Clark completion [27] reads each rule as a biconditional, here $p \leftrightarrow q$, and still admits both models. The ε field selects the founded model directly, with no completion or loop formulas [28]. For a definite program the minimal model, the well-founded model [29], and the unique stable model [13] coincide, so on this fragment the field reproduces those semantics rather than extending them.

Composing this encoding theorem with the three preservation lemmas carries the guarantee back to the source program.

Theorem 2 (Correspondence). *Write $\mathcal{I} = \mathcal{C}_4 \circ \mathcal{C}_3 \circ \mathcal{C}_2 \circ \mathcal{C}_1$. For every program P and EDB $\mathcal{D}$ whose Herbrand universe is nonempty, with the fresh-name supply of Lemma 1, if σ is a ground state of the Ising model $\mathcal{I}(P, \mathcal{D})$, then for every non-auxiliary ground atom f , $\sigma_f = +1 \iff f \in \mathfrak{M}_P(\mathcal{D})$. Theorem 1 applied to the binarized program fixes the auxiliary atoms as well, each to its value in the least model of the ground binarized program $\mathcal{C}_2(\mathcal{C}_1(P))$. Only the 2-IMPLY ancilla spins are unconstrained off this projection: a gate whose head is true with exactly one true body atom has two tying ancilla values, so the full ground-state configuration need not be unique.*

PROOF (SKETCH). The pipeline carries $\mathcal{D}$ inside the program as bodiless ground rules (the Stage 2 convention), and a verified absorption lemma shows this lossless: the extended program's least model over the empty EDB is exactly $\mathfrak{M}_P(\mathcal{D})$. The deployed weight $W = |\mathfrak{B}^{\text{bin}}| + 1$, computed on the binarized ground program ($\mathfrak{B}^{\text{bin}} = \mathfrak{B}$ whenever no rule needs binarization), meets the bound of Theorem 1 with $N = |\mathfrak{B}^{\text{bin}}|$, so the theorem applies. The atom-projection of every ground state of

Table 2

Paper results and their Lean identifiers in the verified development. The namespace of each identifier is its stage file, while the semantic layer (`DataLog.lean`) and the witness module (`Nonvacuity.lean`) export top-level names.

Result	Lean identifier
Lemma 1 (Binarization)	<code>C1.correct</code>
Lemma 2 (Grounding)	<code>C2.correct</code>
Lemma 3 (Min-Ones SAT)	<code>C3.sat</code> , <code>C3.minOnes</code>
Theorem 1 (ε -minimality)	<code>C4.encode</code>
Theorem 2 (Correspondence)	<code>C4.pipeline_correct_edb</code>
2-locality of $H((\eta, J)$ extraction)	<code>C4.twolocal</code>

Table 3

The evaluated programs. n is the size parameter, and *reach* takes a hardware graph $G = (V, E)$ and a seed node as data.

Family	Rules	EDB $\mathcal{D}$
<i>tc_path</i> (n)	$Reach(X, Y) \leftarrow Edge(X, Y)$ $Reach(X, Z) \leftarrow Edge(X, Y), Reach(Y, Z)$	$Edge(i, i+1)$ for $0 \leq i < n-1$
<i>tc_cycle</i> (n)	as <i>tc_path</i>	$Edge(i, (i+1) \bmod n)$ for $0 \leq i < n$
<i>nonlinear_tc_path</i> (n)	$Reach(X, Y) \leftarrow Edge(X, Y)$ $Reach(X, Z) \leftarrow Reach(X, Y), Reach(Y, Z)$	as <i>tc_path</i>
<i>linear_chain</i> (n)	$P_i(X) \leftarrow P_{i-1}(X)$ for $1 \leq i < n$	$P_0(c)$
<i>reach</i> (G)	$Reach(X) \leftarrow Source(X)$ $Reach(Y) \leftarrow Reach(X), Edge(X, Y)$ $Reach(X) \leftarrow Reach(Y), Edge(X, Y)$	$Edge(u, v)$ for $\{u, v\} \in E$, one <i>Source</i>

$\mathcal{I}(P)$ is then the minimal model of the ground program $\mathcal{C}_2(\mathcal{C}_1(P))$, the Min-Ones SAT solution of Lemma 3. That model pins every ground atom of the binarized program, auxiliary atoms included, which is the auxiliary-atom clause. Lemma 2 and Lemma 1 carry it back through grounding and binarization, so on the non-auxiliary predicates it is $\mathfrak{M}_P(\mathcal{D})$.

Verification. Table 2 maps the results of this paper to their Lean identifiers. The artifact’s `lean/README.md` reproduces those statements and definitions verbatim, and records the development as sorry-free, with the standard axioms, and instanced hypotheses.

4. Evaluation and Limitations

We size the datasets and programs to what could fit a commercially available annealer as of writing. The target is the hardware class, 2-local Ising machines. Concrete device parameters come from D-Wave, whose Pegasus (Advantage) and Zephyr (Advantage2) topologies have degree 15 and 20 and a few thousand qubits [21, 22]. Correctness is settled by Theorem 2, so the question here is whether an instance of that scale could run. Table 3 defines every evaluated program. *tc_path* is the running example’s program over a path EDB and *tc_cycle* is the same program over a cycle. *nonlinear_tc_path* replaces the recursive rule by one that joins two derived atoms, so *Reach* atoms can support one another. The non-recursive *linear_chain* is a sparse contrast, and *reach* computes single-source reachability over a hardware graph given as data, the family of the chip-scale experiments below.

Instances are compiled with the reference compiler at the certified encoding’s $\varepsilon = 1$ and $W = n_{\text{atoms}} + 1$. Soundness needs only $|\mathfrak{M}_P(\mathcal{D})| < W$ (Theorem 1), met at every size. Every rule in Table 3 carries at most two body atoms, so Stage 1’s binarization never fires in these experiments, and the auxiliary-atom path is exercised by the verified witness module instead. Experiments use `dwave-samplers` and `minorminer`, with no QPU involved.

The energy landscape. Because bottom-up evaluation is tractable we know the minimal model, and the

instance we enumerate here, $nonlinear_tc_path(4)$, is small enough to know everything else too. Its recursive rule joins two derived atoms, its EDB is the path $0 \rightarrow 1 \rightarrow 2 \rightarrow 3$, and its compiled form has 19 ground atoms, so there are $2^{19} = 524,288$ atom assignments. We compute the exact energy of every one of them, with each gate’s ancilla at its optimal value. The enumeration certifies that the minimal model is the unique global minimum and that exactly 18 assignments are single-flip local minima.

Figure 1 walks through this space. Each position on the horizontal axis is one assignment, consecutive assignments differ in exactly one atom, and the tick label names the atom added at that step, with the curve interpolating between the plotted states. The vertical axis is the assignment’s energy. The walk starts at $\mathcal{D}$, the assignment holding exactly the three *Edge* facts, which the 0-IMPLY gates force into every low-energy state. The next six steps add the derived *Reach* atoms in fixpoint order through the iterates T^1 and T^2 (the first and second application of T_P to $\mathcal{D}$), arriving at the minimal model, the nine-atom assignment at energy $-1,800.5$, the global minimum. The six steps after it add *Reach*(1, 1), *Reach*(2, 1), *Reach*(2, 2), *Reach*(3, 1), *Reach*(3, 2), and *Reach*(3, 3), an unfounded set whose atoms derive one another circularly. The resulting assignment is a genuine local minimum a mere 6 above the global one, one ε per unfounded atom, while leaving it costs at least 79, since flipping any of the six unfounded atoms back to false violates a gate scaled to $W\Delta_{\min} = 80$. The humps between the two wells are the partial states where a rule body is already satisfied but its head not yet added. These foundedness traps are what energy minimization must avoid, and they set the recovery boundary of the recursive family measured below.

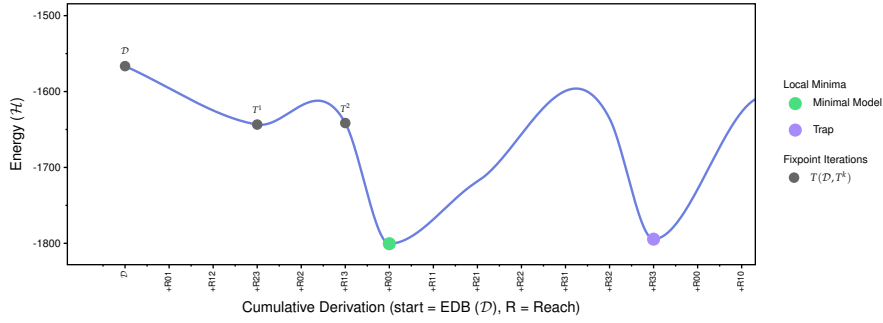


Figure 1: Energy along a one-atom-per-step walk for $nonlinear_tc_path(4)$: from the EDB $\mathcal{D}$ through the fixpoint iterates T^1, T^2 (grey) to the minimal model (green), the global minimum of all 2^{19} assignments by exhaustive enumeration. Adding the six-atom unfounded set (purple) costs one ε per atom, and leaving that local minimum costs at least 79 ($W\Delta_{\min} - \varepsilon$ on the gate scale).

Exact correspondence. On the running example, the path $a \rightarrow b \rightarrow c \rightarrow d$, the compiled model has 31 spins, of which 19 are atom spins (9 true in the minimal model) and 12 are ancillas, one per 2-IMPLY gate, with 51 couplers. An exact solver’s ground state projects exactly onto the 9-atom minimal model, and the cyclic, disjunctive, and multi-derivation cases are instances of Theorem 2’s guarantee.

Logical to physical qubits. The compiled model’s logical interaction graph must be minor-embedded into the annealer’s sparse hardware graph (Section 2.3). We embed the compiled tc_path family with *minorminer* onto the Pegasus and Zephyr generators of *dwave_networkx*. Embedding is randomized, so we report medians over 15 *minorminer* seeds (0–14, an embedding attempt being far cheaper than a recovery run). The running example’s 31 logical spins become a median of 34 physical qubits (range 33–36), a $1.1\times$ overhead with longest chain 2.

Overhead climbs with size as the recursive join drives the logical degree to $O(n)$, which passes Pegasus’s 15 at $n = 8$ and Zephyr’s 20 at $n = 10$. On the Pegasus target, by $n = 9$ the 161-spin model needs a median of 221 qubits (range 201–253), with a longest chain of 7 (range 5–9). Longer chains also spend part of the energy scale on chain strength (the coupling that holds a chain’s qubits aligned), compounding the precision caveat below.

Saturating the shipped machine. Whole-chip topologies are the substrate of D-Wave’s beyond-classical simulations [30], and a topology is itself a graph, so it hosts a natural program: *reach* of Table 3, single-source reachability over the chip’s own graph. The mirrored third rule traverses stored

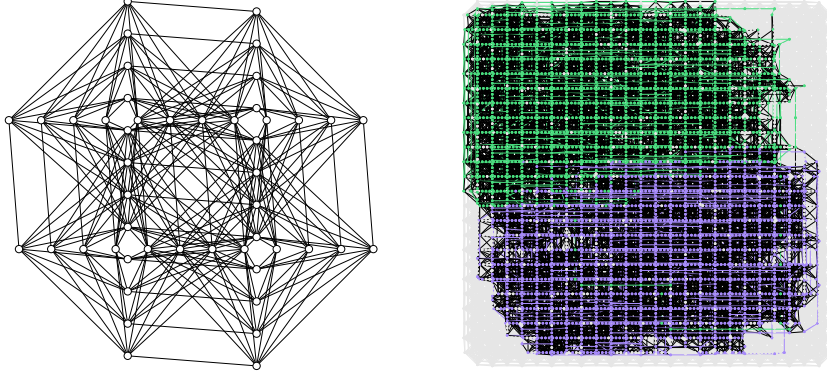


Figure 2: Two independently compiled reachability programs at the shipped machine’s scale. Left: each instance’s EDB, the Zephyr Z1 hardware graph (48 nodes, 280 edges). Right: both compiled programs, 1,778 logical spins, minor-embedded together into the ideal Zephyr Z12 generator (4,800 sites), occupying 3,489 physical qubits with longest chain 36. One instance is green, the other purple, unused qubits are greyed.

edges backward, keeping one *Edge* fact per undirected link. Because *Reach* is unary, the compiled model scales with the data: $V + 3E + 1$ spins for a graph with V nodes and E edges, not $|U|^2$. The instance whose EDB is the Zephyr Z1 graph compiles to 889 logical spins and embeds into the ideal Z12 generator (4,800 sites), the topology class of the shipped Advantage2, on 1,696 sites, with chains up to 26 on the high-degree *Reach* hubs. Larger single instances are out of reach. Z2 compiles to 3,833 spins, above the machine at the observed $2\times$ embedding overhead, and Z3 exceeds it by spin count alone. The machine saturates with independent programs instead: Figure 2 shows two separately compiled Z1 instances embedded side by side on 3,489 sites. Compiled separately, each keeps its own weight W , so the two models are independent Hamiltonians sharing only the chip, and one anneal carries both. On a device the co-tenants would also share the single global coefficient rescaling, harmless here since the two weights are equal.

Solver recovery. Theorem 2 places the minimal model at the unique ground state, and we measure whether a sampler reaches it. We compile each instance to its Ising model and sample it with two solvers from D-Wave’s SDK: classical simulated annealing (SA, `SimulatedAnnealingSampler`) and D-Wave’s simulated quantum annealing (SQA, `PathIntegralAnnealingSampler`, a path-integral Monte Carlo surrogate for the device). Both are classical simulations, so the recovery rates measured here do not bound a physical anneal. Each runs at a budget of 200 reads (independent anneals) of 1,000 sweeps (update passes over all spins), all other parameters at library defaults. We report the median over five seeds (42–46) with the min–max range, all reproducible from the committed artifact. A run *recovers* the model when its lowest-energy read decodes to the bottom-up least model. The rate p_{succ} is the fraction of a run’s reads that decode to it. Both solvers carry full floating-point coefficients, isolating solver behavior from coupler precision.

Table 4 reports *tc_path*: recovery is near-certain to $n = 6$, then collapses over $n = 7$ –9 (p_{succ} from 0.76 to 0.01) and vanishes by $n = 10$. *tc_cycle* breaks two steps earlier (its last recovery is at $n = 7$), and the non-recursive *linear_chain* recovers throughout the sweep to $n = 30$. The non-linear *nonlinear_tc_path* collapses far sooner: recovery is near-certain at $n = 2$ but already gone by $n = 4$ (p_{succ} 0.97 $\rightarrow$ 0.01 $\rightarrow$ 0), because its unfounded self-supporting sets are the deep traps of Figure 1, basins a sampler falls into and rarely climbs out of.

At logical degree 9 in both, *nonlinear_tc_path*(3) with 11 atoms recovers at 0.01 while *tc_path*(4) with 19 atoms recovers at 0.99, so size alone does not explain the gap. An acyclic linear program admits no self-supporting set. Its shallow incomplete-derivation traps multiply with n and, with the climbing degree, plausibly drive *tc_path*’s later collapse. In these families the wall’s arrival tracks foundedness rather than coupler precision: both solvers carry exact coefficients, yet the energy landscape hides the

Table 4

Solver recovery on *tc_path* (200 reads of 1,000 sweeps): atom spins, minimal-model size, maximum logical degree, and the recovery rate p_{succ} , median over five seeds [min, max], under SA and SQA.

n	atoms	$ \mathfrak{M} $	degree	p_{succ} SA	p_{succ} SQA
4	19	9	9	0.99 [0.98,1.00]	0.98 [0.98,0.99]
5	29	14	11	0.99 [0.98,0.99]	0.98 [0.98,1.00]
6	41	20	13	0.98 [0.98,1.00]	0.96 [0.94,0.98]
7	55	27	15	0.76 [0.73,0.78]	0.60 [0.58,0.64]
8	71	35	17	0.20 [0.17,0.28]	0.12 [0.08,0.15]
9	89	44	19	0.01 [0.01,0.03]	0.01 [0.01,0.01]
10	109	54	21	0.00 [0.00,0.00]	0.00 [0.00,0.01]

program’s polynomial-time Horn structure from local search, and more so as the rules admit more unfounded support.

The wall is calibrated to the budget rather than absolute: $10\times$ and $50\times$ the sweeps lift SA on *tc_path*(9) from 0.01 to 0.175 and 0.55 (and $n = 8$ to 0.985, medians over the same five seeds), so the crossing points shift with effort while the ordering in size and family persists. SQA is never better than SA beyond noise, and is worse on *linear_chain* (0.52 against 0.97 at $n = 30$).

At the machine’s scale, the *reach* program of Table 3 over two disjoint copies of the Z1 graph, with a single *Source* in the first copy, compiles to 1,777 spins, and its minimal model lights *Reach* on the sourced copy only. The count is one spin fewer than Figure 2’s pair of independent programs, whose second copy carries its own *Source* atom. Under the protocol above neither solver recovers it on any read (p_{succ} 0.00 [0.00, 0.00] over five seeds). Every failing read decodes to the same assignment, the minimal model plus all 48 unreached *Reach* atoms, an unfounded set in which the mirrored rule lets the endpoints of every stored edge support each other. That assignment sits 48ε above the ground state, escaping it requires climbing a single-violation barrier of $W\Delta_{\min} = 2,632$, and it survives ten thousand times the sweep budget unchanged. The instance is classically trivial: an incremental Datalog engine, PyDBSP 2.1.0 [31], computes the same minimal model in under 7 milliseconds (median of five runs on an Apple M5 Max laptop CPU).

Precision on a physical device. The solvers above compute with exact coefficients, while a QPU rescales them into a fixed analog range where integrated control errors perturb each programmed value by about 10^{-2} at one standard deviation [32]. Resolving two coefficients reliably needs several standard deviations of separation, leaving roughly 4 to 5 dependable bits. The smallest separation that matters is one atom of cardinality, worth $\varepsilon = 1$, set against the largest programmed coefficient magnitude $c_{\max}$, which the sparse sweep families measure at $4W$ (each fact’s $-5W$ field gains $+W$ from its atom’s role as a 1-IMPLY body), a relative gap $\varepsilon/c_{\max} \sim 1/(4n_{\text{atoms}})$. For *tc_path* a 4-bit device already fails to resolve that gap at the smallest instance (5 atoms) and a 5-bit device at 11 atoms, with the other sparse families crossing within a few atoms of the same points. The non-linear family’s $c_{\max}/W$ grows with n , so its budget is strictly tighter.

5. Conclusion

We present $\mathcal{I} = \mathcal{C}_4 \circ \mathcal{C}_3 \circ \mathcal{C}_2 \circ \mathcal{C}_1$, our compiler from recursive Datalog programs to 2-local Ising models, whose ground state projects onto the least Herbrand model of the source program. The per-stage results carry the minimal model through (Lemmas 1–3 and Theorem 1), and their composition is the ground-state correspondence of Theorem 2. Empirically, recovery under the simulated solvers tracks foundedness rather than instance size, failing completely at chip scale while a classical engine answers in milliseconds. The contribution is the verified bridge itself, and its grounding runs classically. Any quantum advantage would have to come from NP-hard extensions of the source language, from distributing one program across several annealers, or from a richer use of the annealer’s quantum

dynamics than a ground-state read exploits.

Acknowledgments

This work was supported by the Estonian Research Council grant (PRG2764).

Declaration on Generative AI

During the preparation of this work, the authors used Claude (Anthropic) in order to: peer review simulation, paraphrase and reword, citation management. The authors further used it to assist in writing proofs for the Lean 4 mechanization. After using these tools and services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

- [1] T. Kadowaki, H. Nishimori, Quantum annealing in the transverse Ising model, *Physical Review E* 58 (1998) 5355–5363. doi:10.1103/PhysRevE.58.5355.
- [2] M. W. Johnson, et al., Quantum annealing with manufactured spins, *Nature* 473 (2011) 194–198. doi:10.1038/nature10012.
- [3] A. Colmerauer, P. Roussel, The birth of Prolog, in: *History of Programming Languages—II*, ACM, 1996, pp. 331–367. doi:10.1145/234286.1057820.
- [4] E. Ising, Beitrag zur theorie des ferromagnetismus, *Zeitschrift für Physik* 31 (1925) 253–258. doi:10.1007/BF02980577.
- [5] S. Pakin, Performing fully parallel constraint logic programming on a quantum annealer, *Theory and Practice of Logic Programming* 18 (2018) 928–949. doi:10.1017/S1471068418000066.
- [6] S. Ceri, G. Gottlob, L. Tanca, What you always wanted to know about Datalog (and never dared to ask), *IEEE Transactions on Knowledge and Data Engineering* 1 (1989) 146–166. doi:10.1109/69.43410.
- [7] M. H. van Emden, R. A. Kowalski, The semantics of predicate logic as a programming language, *Journal of the ACM* 23 (1976) 733–742. doi:10.1145/321978.321991.
- [8] L. de Moura, S. Ullrich, The Lean 4 theorem prover and programming language, in: *Automated Deduction (CADE)*, volume 12699 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 625–635. doi:10.1007/978-3-030-79876-5_37.
- [9] G. Pinkas, Symmetric neural networks and propositional logic satisfiability, *Neural Computation* 3 (1991) 282–291. doi:10.1162/neco.1991.3.2.282.
- [10] W. A. T. Wan Abdullah, Logic programming on a neural network, *International Journal of Intelligent Systems* 7 (1992) 513–519. doi:10.1002/int.4550070604.
- [11] A. Lucas, Ising formulations of many NP problems, *Frontiers in Physics* 2 (2014) 5. doi:10.3389/fphy.2014.00005.
- [12] Z. Bian, F. Chudak, W. Macready, A. Roy, R. Sebastiani, S. Varotti, Solving SAT (and MaxSAT) with a quantum annealer: Foundations, encodings, and preliminary results, *Information and Computation* 275 (2020) 104609. doi:10.1016/j.ic.2020.104609.
- [13] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, MIT Press, 1988, pp. 1070–1080.
- [14] L. K. Grover, A fast quantum mechanical algorithm for database search, in: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, 1996, pp. 212–219. doi:10.1145/237814.237866.
- [15] R. Romanello, D. D. Giustina, S. Pessotto, C. Piazza, Speeding up answer set programming by quantum computing, in: *Proceedings of the 2024 Workshop on Quantum Search and Information Retrieval*, ACM, 2024, pp. 1–8. doi:10.1145/3660318.3660328.
- [16] R. Ben-Eliyahu, R. Dechter, On computing minimal models, *Annals of Mathematics and Artificial Intelligence* 18 (1996) 3–27. doi:10.1007/BF02136172.
- [17] S. Khanna, M. Sudan, L. Trevisan, D. P. Williamson, The approximability of constraint satisfaction problems, *SIAM Journal on Computing* 30 (2001) 1863–1920. doi:10.1137/S0097539799349948.
- [18] E. Boros, P. L. Hammer, Pseudo-Boolean optimization, *Discrete Applied Mathematics* 123 (2002) 155–225. doi:10.1016/S0166-218X(01)00341-9.
- [19] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, *Pacific Journal of Mathematics* 5 (1955) 285–309.
- [20] S. Abiteboul, R. Hull, V. Vianu, *Foundations of Databases*, Addison-Wesley, 1995.
- [21] K. Boothby, P. Bunyk, J. Raymond, A. Roy, Next-generation topology of D-Wave quantum processors, arXiv:2003.00133, 2020.
- [22] K. Boothby, A. D. King, J. Raymond, Zephyr Topology of D-Wave Quantum Processors, Technical Report 14-1056A-A, D-Wave Systems Inc., 2021.
- [23] V. Choi, Minor-embedding in adiabatic quantum computation: I. the parameter setting problem, *Quantum Information Processing* 7 (2008) 193–209. doi:10.1007/s11128-008-0082-9.
- [24] M. Y. Vardi, The complexity of relational query languages, in: *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC)*, 1982, pp. 137–146. doi:10.1145/800070.802186.
- [25] F. Bancilhon, D. Maier, Y. Sagiv, J. D. Ullman, Magic sets and other strange ways to implement logic programs, in: *Proceedings of the 5th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS)*, 1986, pp. 1–15. doi:10.1145/6012.15399.
- [26] W. F. Dowling, J. H. Gallier, Linear-time algorithms for testing the satisfiability of propositional Horn formulae, *The Journal of Logic Programming* 1 (1984) 267–284. doi:10.1016/0743-1066(84)90014-1.
- [27] K. L. Clark, Negation as failure, in: H. Gallaire, J. Minker (Eds.), *Logic and Data Bases*, Plenum Press, New York, 1978, pp. 293–322. doi:10.1007/978-1-4684-3384-5_11.
- [28] F. Lin, Y. Zhao, ASSAT: Computing answer sets of a logic program by SAT solvers, *Artificial Intelligence* 157 (2004) 115–137. doi:10.1016/j.artint.2004.04.004.

- [29] A. Van Gelder, K. A. Ross, J. S. Schlipf, The well-founded semantics for general logic programs, *Journal of the ACM* 38 (1991) 619–649. doi:10.1145/116825.116838.
- [30] A. D. King, et al., Beyond-classical computation in quantum simulation, *Science* 388 (2025) 199–204. doi:10.1126/science.ad6285.
- [31] B. R. C. A. de Lima, K. Apinis, M. Kramer, K. Micinski, Incremental evaluation of dynamic Datalog programs as a higher-order DBSP program, in: *Datalog-2.0 2024*, volume 3801 of *CEUR Workshop Proceedings*, 2024, pp. 2–16. URL: <https://ceur-ws.org/Vol-3801/>.
- [32] D-Wave Quantum Inc., Integrated control errors (ICE), <https://docs.dwavequantum.com>, 2026.