Deciding Boundedness of Monadic Sirups

Stanislav Kikot
Institute for Information Transmission Problems
Moscow, Russia
staskikotx@gmail.com

Agi Kurucz
King's College London
London, UK
agi.kurucz@kcl.ac.uk

Vladimir V. Podolskii
HSE University
Moscow, Russia
vpodolskii@hse.ru

Michael Zakharyaschev
Birkbeck, University of London, UK &
HSE University, Moscow, Russia
michael@dcs.bbk.ac.uk

ABSTRACT

We show that deciding boundedness (aka FO-rewritability) of monadic single rule datalog programs (sirups) is 2ExpTime-hard, which matches the upper bound known since 1988 and finally settles a long-standing open problem. We obtain this result as a byproduct of an attempt to classify monadic 'disjunctive sirups'—Boolean conjunctive queries $\boldsymbol{q}$ with unary and binary predicates mediated by a disjunctive rule $T(x) \lor F(x) \leftarrow A(x)$—according to the data complexity of their evaluation. Apart from establishing that deciding FO-rewritability of disjunctive sirups with a dag-shaped $\boldsymbol{q}$ is also 2ExpTime-hard, we make substantial progress towards obtaining a complete FO/L-hardness dichotomy of disjunctive sirups with ditree-shaped $\boldsymbol{q}$.

CCS CONCEPTS

• **Information systems → Query languages**; • **Theory of computation → Complexity theory and logic**; **Description logics**; • **Computing methodologies → Knowledge representation and reasoning**.

KEYWORDS

Boundedness, monadic datalog, first-order rewritability, ontology-mediated query.

ACM Reference Format:
Stanislav Kikot, Agi Kurucz, Vladimir V. Podolskii, and Michael Zakharyaschev. 2021. Deciding Boundedness of Monadic Sirups. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 23 pages. https://doi.org/10.1145/3452021.3458332

1 INTRODUCTION

There have been two waves in the investigation of boundedness or first-order rewritability of various types of recursive queries. The first one started in the mid 1980s, when the deductive database community was analysing recursion in datalog queries with the aim of optimising and parallelising their execution. One of the fundamental issues was the problem of deciding whether the depth of recursion required to evaluate a given datalog query could be bounded independently of the input data. By 2000, among other remarkable results, it had been discovered that

– boundedness of linear datalog queries with binary predicates and of ternary linear datalog queries with a single recursive rule is undecidable [27, 33];
– deciding program boundedness is 2ExpTime-complete for monadic programs [9, 18], PSpace-complete for linear monadic programs [18, 36], and NP-complete for linear monadic and dyadic single rule programs [37].

Interestingly, the exact complexity of deciding boundedness of monadic datalog programs with a single recursive rule, known as *sirups* since [19], has remained open so far, somewhere between NP and 2ExpTime, to be more precise. To clarify the 'status [of boundedness] for sirups' is part of Open Problem 4.2.10 in [29]. According to [4], Kanellakis and Papadimitriou, who were interested in datalog programs computable in NC, and so parallelisable, 'have investigated the case of unary sirups, and have made progress towards a complete characterization'. Alas, that work appears to have never been completed and published.

In this paper, we finally settle the boundedness problem for monadic sirups by showing that it is 2ExpTime-hard, which matches the upper bound for deciding boundedness of arbitrary monadic datalog programs [18] (and which should be compared with the NP–PSpace gap between deciding boundedness of *linear* sirups and non-sirups.)

We obtained this result while surfing the second wave, which was triggered in the mid 2010s by the theory and practice of ontology-based data access (OBDA) [15, 35, 38] (recently rebranded to virtual knowledge graphs [39]). In OBDA, a typical ontology-mediated query (OMQ) takes the form $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q})$ with a description logic (DL) ontology $\mathcal{O}$ and a conjunctive query (CQ) $\boldsymbol{q}$. A fundamental problem in this setting is to decide whether a given OMQ $\boldsymbol{Q}$ is FO-rewritable, in which case finding certain answers to $\boldsymbol{Q}$ can be done by evaluating a non-recursive SQL-query using a standard RDBMS.

The ontology language *OWL 2 QL* for OBDA systems (such as Mastro[1] or Ontop[2]), standardised by the W3C in 2009, is based on

ACM ISBN 978-1-4503-8381-3/21/06…$15.00
https://doi.org/10.1145/3452021.3458332

[1]https://www.obdasystems.com
[2]https://ontopic.biz

DL-Lite that uniformly guarantees FO-rewritability of all OMQs with an *OWL 2 QL* ontology. Uniformly FO-rewritable tgds, aka Datalog$^{\pm}$ or existential rules, have also been identified; see, e.g., [17, 23, 30]. As an inevitable consequence, however, all of these ontology languages are very inexpressive.

The FO-rewritability problem for OMQs in more expressive ontology languages was attacked in [11] via a reduction to CSPs. It has been shown, among other results, that

– deciding FO-rewritability of OMQs with ontologies in expressive DLs such as $\mathcal{ALC}$ (notational variant of multi-modal logic $\mathbf{K}_n$) and atomic CQs is NExpTime-complete [11], which becomes 2NExpTime-complete in the case of (non-atomic) CQs and also monadic disjunctive datalog queries [14, 20];
– any OMQ with a (Horn) $\mathcal{EL}$ ontology and a CQ is either FO-, or linear-datalog-, or datalog-rewritable, and deciding this trichotomy is ExpTime-complete [31, 32]; see also [6, 10] for complexity results on deciding FO-rewritability of OMQs with more expressive Horn description logic ontologies and frontier-guarded existential rules.

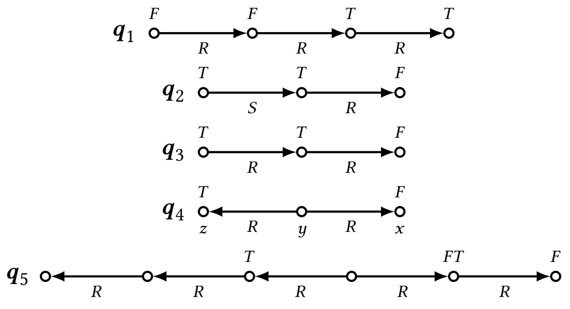
In [22], aiming to single out and classify possible causes of non-FO- or non-(linear)-datalog-rewritability of OMQs, we considered (in the DL setting) a disjunctive analogue of monadic sirups, namely, monadic disjunctive datalog programs $\Delta_{\boldsymbol{q}}$ of the form

$$T(x) \lor F(x) \leftarrow A(x) \tag{1}$$

$$\boldsymbol{G} \leftarrow \boldsymbol{q} \tag{2}$$

where $\boldsymbol{q}$ is a (Boolean) CQ with unary predicates $T(x)$, $F(y)$ and arbitrary binary predicates, and $\boldsymbol{G}$ is a nullary (goal) predicate. In DL and conceptual modelling, rule (1) is known as a *covering axiom* (or constraint) $A \sqsubseteq T \sqcup F$ (as in 'class Animal is covered by classes Male and Female'). We illustrate the zoo of 'monadic disjunctive sirups' by an example, where CQs are given as digraphs with labelled edges and (partially) labelled nodes.

Example 1. Consider the CQs $\boldsymbol{q}_1, \dots, \boldsymbol{q}_5$ shown below:



For instance, in full, rule (2) in the program $\Delta_{\boldsymbol{q}_4}$ looks as

$$\boldsymbol{G} \leftarrow F(x), R(y,x), R(y,z), T(z).$$

Intuitively, the certain answer to the Boolean query $(\Delta_{\boldsymbol{q}_4}, \boldsymbol{G})$ over a data instance $\mathcal{D}$ (given in the form of a labelled graph) is 'yes' iff we can find the pattern $\boldsymbol{q}_4$ in every graph obtained by labelling each of the A-nodes in $\mathcal{D}$ with either T or F. As shown in [22], answering $(\Delta_{\boldsymbol{q}_i}, \boldsymbol{G})$ is coNP-complete for $\boldsymbol{q}_1$, P-complete for $\boldsymbol{q}_2$, NL-complete for $\boldsymbol{q}_3$, L-complete for $\boldsymbol{q}_4$, and, in view of Example 4 below, $\boldsymbol{q}_5$ is FO-rewritable and so in AC^0.

Every disjunctive sirup $\Delta_{\boldsymbol{q}}$, in which $\boldsymbol{q}$ has a single 'solitary' F-node (like in $\boldsymbol{q}_2$–$\boldsymbol{q}_5$), is equivalent to a monadic datalog program $\Pi_{\boldsymbol{q}}$. For instance, $\Delta_{\boldsymbol{q}_4}$ is equivalent to $\Pi_{\boldsymbol{q}_4}$ with three rules

$$\begin{aligned} &\boldsymbol{G} \leftarrow F(x), R(y,x), R(y,z), P(z)\\ &P(x) \leftarrow T(x)\\ &P(x) \leftarrow A(x), R(y,x), R(y,z), P(z) \end{aligned}$$

Furthermore, for certain CQs $\boldsymbol{q}$, boundedness of $\Pi_{\boldsymbol{q}}$ coincides with boundedness of a sirup sub-program of $\Pi_{\boldsymbol{q}}$ (see Sec. 2). In the above example, this sirup, $\Sigma_{\boldsymbol{q}_4}$, comprises the last two rules of $\Pi_{\boldsymbol{q}_4}$, and neither $(\Delta_{\boldsymbol{q}_4}, \boldsymbol{G})$ nor $(\Sigma_{\boldsymbol{q}_4}, P)$ is FO-rewritable.

On the other hand, every disjunctive sirup $\Delta_{\boldsymbol{q}}$ can be encoded as a CQ mediated by a Schema.org[3] ontology. Deciding FO-rewritability of UCQs mediated by Schema.org is known to be PSpace-hard [25].

Our first result in this paper establishes 2ExpTime-hardness of deciding FO-rewritability in all of these cases. In Sec. 3, we show how a computation of an alternating Turing machine can be captured in terms of boundedness of the disjunctive sirup $\Delta_{\boldsymbol{q}}$, datalog program $\Pi_{\boldsymbol{q}}$ or its sirup sub-program $\Sigma_{\boldsymbol{q}}$, for some CQ $\boldsymbol{q}$. Compared to known techniques, which require multiple rules in a program or a union of multiple CQs to check properties of Turing machine computations, we achieve the same aim by means of polynomially-many small Boolean circuits that are 'implemented' by a *single* CQ $\boldsymbol{q}$ and check local properties of binary trees representing the expansions of $\Pi_{\boldsymbol{q}}$.

What causes such high computational costs of recognising FO-rewritability of seemingly very primitive programs? Are there any natural classes of monadic (disjunctive) sirups whose boundedness can be checked by tractable algorithms? The 2ExpTime-hardness proof provides three clues: first, the CQs $\boldsymbol{q}$ used in it are dags; second, each of them has two T-nodes; and, third, they contain many twin FT-nodes (as in $\boldsymbol{q}_5$ above). In [22], we gave a complete classification of monadic disjunctive sirups $\Delta_{\boldsymbol{q}}$ with a path CQ $\boldsymbol{q}$ and an extra disjointness constraint

$$\bot \leftarrow T(x), F(x) \tag{3}$$

(as in 'classes Male and Female are disjoint') according to their data complexity (AC^0/NL/P/coNP) and rewritability type (FO/linear datalog/datalog/disjunctive datalog).

Here, in Sec. 4, we make significant progress towards a complete understanding of FO-rewritability of disjunctive sirups $\Delta_{\boldsymbol{q}}$ with a ditree-shaped CQ $\boldsymbol{q}$. First, we prove that twin-free CQs $\boldsymbol{q}$ as well as those that contain comparable (w.r.t. the tree order in $\boldsymbol{q}$) solitary F- and T-nodes (like in $\boldsymbol{q}_1$–$\boldsymbol{q}_3$ but not $\boldsymbol{q}_4$ and $\boldsymbol{q}_5$) give rise to NL-hard disjunctive sirups. In particular, this yields a tractable FO/NL-hardness dichotomy of the ditree disjunctive sirups with disjointness (3). Second, we obtain a tractable FO/L/NL-completeness trichotomy of ditree disjunctive sirups with one solitary F, one solitary T and any number of FT-twins. (This case corresponds to linear ditree sirups.) Finally, we establish an FO/L-hardness dichotomy for ditree disjunctive sirups with one solitary F and show that this dichotomy can be decided in polynomial time if the number of solitary Ts in the CQs is bounded (like in our 2ExpTime-hardness

[3]https://schema.org: 'Many applications from Google, Microsoft, Pinterest, Yandex and others already use these vocabularies to power rich, extensible experiences'.

proof) and in exponential time otherwise. It follows that deciding FO-rewritability of such disjunctive sirups is fixed-parameter tractable if the number of solitary Ts is regarded as a parameter.

2 PRELIMINARIES

We remind the reader (who can consult [2] for details) that a datalog program is a finite set, Π, of rules of the form

$$\forall \boldsymbol{x}\, (\gamma_0 \leftarrow \gamma_1 \wedge \cdots \wedge \gamma_m) \tag{4}$$

where each γ_i is a (constant- and function-free) atom $Q(\boldsymbol{y})$ with $\boldsymbol{y} \subseteq \boldsymbol{x}$. As usual, we omit $\forall \boldsymbol{x}$ and replace $\wedge$ with a comma. The atom γ_0 is the *head* of the rule, and $\gamma_1, \dots, \gamma_m$ comprise its *body*. The variables in the head must also occur in the body. The predicate in the head of a rule in Π is called an *IDB predicate*; non-IDB predicates in Π are *EDB predicates*. We call a rule *recursive* if its body has at least one IDB predicate; otherwise, it is an *initialisation rule*. The *arity* of Π is the maximum arity of its IDB predicates. Here, we only consider *monadic* datalog programs with at most *binary* EDBs. A *monadic sirup* is a monadic program with a single recursive rule.

A *data instance* for Π is any finite set $\mathcal{D}$ of ground atoms with EDB predicates in Π. The set of constants in $\mathcal{D}$ is denoted by $\mathsf{ind}(\mathcal{D})$. For a unary IDB predicate P, a *certain answer* to the *datalog query* (Π, P) *over* $\mathcal{D}$ is any $a \in \mathsf{ind}(\mathcal{D})$ such that $\mathcal{I} \models P[a]$, for every model $\mathcal{I}$ of Π and $\mathcal{D}$, or, in other words, $P(a)$ is in the closure $\Pi(\mathcal{D})$ of $\mathcal{D}$ under the rules in Π. For a 0-ary IDB $\boldsymbol{G}$ (goal), a *certain answer* to $(\Pi, \boldsymbol{G})$ over $\mathcal{D}$ is 'yes' if $\boldsymbol{G} \in \Pi(\mathcal{D})$, and 'no' otherwise.

A typical monadic datalog program, $\Pi_{\boldsymbol{q}}$, we deal with in this paper is associated with a *conjunctive query* (CQ) $\boldsymbol{q}$, which in our context is just a set of atoms with unary predicates F, T and arbitrary binary predicates. An atom $F(z) \in \boldsymbol{q}$ is *solitary* if $T(z) \notin \boldsymbol{q}$, and symmetrically for $T(z)$; a pair $T(z), F(z) \in \boldsymbol{q}$ is referred to as *twins*.

For a CQ $\boldsymbol{q}$ with a single solitary $F(x)$, possibly multiple solitary $T(y_1), \dots, T(y_n)$, arbitrary twins $T(z), F(z)$ and binary atoms, the program $\Pi_{\boldsymbol{q}}$ comprises the following rules with 0-ary goal $\boldsymbol{G}$:

$$\boldsymbol{G} \leftarrow F(x), \boldsymbol{q}^-, P(y_1), \dots, P(y_n) \tag{5}$$

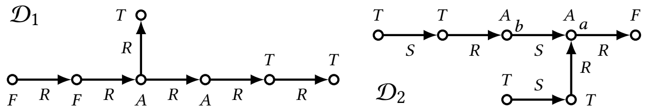
$$P(x) \leftarrow T(x) \tag{6}$$

$$P(x) \leftarrow A(x), \boldsymbol{q}^-, P(y_1), \dots, P(y_n) \tag{7}$$

where $\boldsymbol{q}^- = \boldsymbol{q} \setminus \{F(x), T(y_1), \dots, T(y_n)\}$, and A and P are fresh unary EDB and IDB predicates, respectively. One can show (see [22, 28] for details) that, for any such $\boldsymbol{q}$, called a *1-CQ* henceforth, $(\Pi_{\boldsymbol{q}}, \boldsymbol{G})$ is equivalent to the *disjunctive datalog program* $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ with rules (1) and (2) in the sense that they return the same answer over any data instance $\mathcal{D}$. Here, as usual, a *certain answer* to $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ over $\mathcal{D}$ is 'yes' iff $\mathcal{I} \models \boldsymbol{G}$, for every model $\mathcal{I}$ of $\Delta_{\boldsymbol{q}}$ and $\mathcal{D}$.

The monadic sirups, deciding boundedness of which is proved to be 2ExpTime-hard in Sec. 3, take the form $\Sigma_{\boldsymbol{q}} = \{(6), (7)\}$ with a 1-CQ $\boldsymbol{q}$ and goal predicate P. Adapting a similar terminology, we refer to disjunctive datalog programs $\Delta_{\boldsymbol{q}} = \{(1), (2)\}$ and queries $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$, where $\boldsymbol{q}$ may contain multiple T and F in general, as *monadic disjunctive sirups* or *d-sirups*, for short.

Example 2. Note that recursion in d-sirups is implicit and originates in 'proof by exhaustion' or 'case distinction', which can be seen by evaluating $(\Delta_{\boldsymbol{q}_1}, \boldsymbol{G})$ and $(\Delta_{\boldsymbol{q}_2}, \boldsymbol{G})$ (or the corresponding $(\Pi_{\boldsymbol{q}_2}, \boldsymbol{G})$), with the $\boldsymbol{q}_i$ from Example 1, over the data instances $\mathcal{D}_1$ and, respectively $\mathcal{D}_2$ below.



For instance, let $\mathcal{I}$ be any model of $\Delta_{\boldsymbol{q}_2}$ and $\mathcal{D}_2$. By rule (1), each of the A-nodes a and b in $\mathcal{I}$ is labelled by F or T. If a is an F-node, $\boldsymbol{q}_2$ is embeddable in $\mathcal{I}$ via the vertical R-arrow. So let a be a T-node. If b is a T-node, $\boldsymbol{q}_2$ is embeddable in $\mathcal{I}$ starting from a, and if b is an F-node, there is an embedding starting from b. Thus, $\mathcal{I} \models \boldsymbol{q}_2$.

A monadic (disjunctive) datalog query (Π, Q) is *bounded* or *FO-rewritable* if there is a first-order formula $\Phi(x)$ (a sentence Φ if Q is 0-ary) such that, for any data instance $\mathcal{D}$, a constant $a \in \mathsf{ind}(\mathcal{D})$ (or 'yes') is a certain answer to (Π, Q) over $\mathcal{D}$ iff $\mathcal{D} \models \Phi[a]$ (respectively, $\mathcal{D} \models \Phi$), where $\mathcal{D}$ is regarded as an FO-structure. It is known (see, e.g., [11, 20]) that in this case (Π, Q) is rewritable into a union of conjunctive queries (UCQ). It is also known [34] that FO-rewritability of datalog queries (Π, Q) can be characterised in terms of *Q-expansions*, which are defined inductively below for our special queries $(\Pi_{\boldsymbol{q}}, \boldsymbol{G})$ under the moniker 'cactuses'.

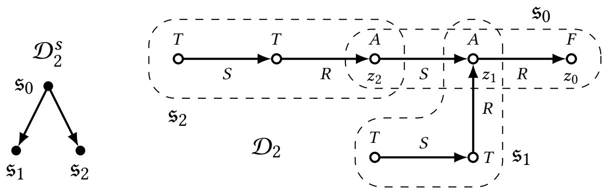
To begin with, we set $C_{\boldsymbol{G}} = \{F(x), \boldsymbol{q}^-, T(y_1), \dots, T(y_n)\} = \{\boldsymbol{q}\}$ and $\mathfrak{K}_{\boldsymbol{q}} = \{C_{\boldsymbol{G}}\}$. Then we take the closure of $\mathfrak{K}_{\boldsymbol{q}}$ under the rule

(bud) if $T(y) \in C \in \mathfrak{K}_{\boldsymbol{q}}$ is solitary, then we add to $\mathfrak{K}_{\boldsymbol{q}}$ the set of atoms obtained from C by replacing $T(y)$ with the atoms $A(x), \boldsymbol{q}^-, T(y_1), \dots, T(y_n)$, in which x is renamed to y and all other variables are given *fresh* names.

The elements of the resulting (infinite if $n \geq 1$) set $\mathfrak{K}_{\boldsymbol{q}}$ are called *cactuses* for $(\Pi_{\boldsymbol{q}}, \boldsymbol{G})$. We represent cactuses as labelled digraphs.

For $C \in \mathfrak{K}_{\boldsymbol{q}}$, we refer to the copies $\mathfrak{s}$ of (maximal subsets of) $\boldsymbol{q}$ that comprise C as *segments* and to the copy of the solitary F-node in $\mathfrak{s}$ as its *focus*. The *skeleton* C^s of C is the ditree whose nodes are the segments $\mathfrak{s}$ of C and edges $(\mathfrak{s}, \mathfrak{s}')$ mean that $\mathfrak{s}'$ was attached to $\mathfrak{s}$ by budding. The *depth of* $\mathfrak{s}$ *in* C (or *in* C^s) is the number of edges on the branch from the root of C^s to $\mathfrak{s}$. The *depth of* C is the maximum depth of its segments.

Example 3. The data instance $\mathcal{D}_2$ from Example 2 is (isomorphic to) a cactus from $\mathfrak{K}_{\boldsymbol{q}_2}$ obtained by applying **(bud)** to $\boldsymbol{q}_2$ twice. The skeleton $\mathcal{D}_2^s$ along with its three segments $\mathfrak{s}_0, \mathfrak{s}_1, \mathfrak{s}_2$ and their respective focuses z_0, z_1, z_2 are illustrated below:



In the remainder of this section, we establish a connection between boundedness of $(\Pi_{\boldsymbol{q}}, \boldsymbol{G})$ and $(\Sigma_{\boldsymbol{q}}, P)$, for a 1-CQ $\boldsymbol{q}$, which requires a few definitions. Every cactus $C \in \mathfrak{K}_{\boldsymbol{q}}$ has exactly one F-node. We call it the *root-focus* of C and denote it by r. By replacing the F-label of r in C with A, we obtain a digraph C°; the set of all such C°, for $C \in \mathfrak{K}_{\boldsymbol{q}}$, is denoted by $\mathfrak{K}_{\boldsymbol{q}}^\circ$. The following proposition is proved by a standard induction on the derivation length:

Proposition 1. *For any data instance $\mathcal{D}$ and any $a \in \mathsf{ind}(\mathcal{D})$,*

- *$\boldsymbol{G} \in \Pi_{\boldsymbol{q}}(\mathcal{D})$ iff there is a homomorphism from some cactus $C \in \mathfrak{K}_{\boldsymbol{q}}$ to $\mathcal{D}$;*

– *$P(a) \in \Sigma_{\boldsymbol{q}}(\mathcal{D})$ iff either $T(a) \in \mathcal{D}$ or there is a homomorphism h from some $C^\circ \in \mathfrak{K}^\circ_{\boldsymbol{q}}$ to $\mathcal{D}$ such that $h(r) = a$.*

A 1-CQ $\boldsymbol{q}$ is called *focused* if the following condition holds:

(foc) for any cactuses $C, C' \in \mathfrak{K}_{\boldsymbol{q}}$, if there is a homomorphism $h \colon C \to C'$, then $h(r) = r$.

The significance of this notion is shown by Example 4 below, and by the following characterisation of boundedness; cf. [34]:

Proposition 2. *For every focused 1-CQ $\boldsymbol{q}$ with solitary $F(x)$, $T(y_1), \dots, T(y_n)$, the following conditions are equivalent:*

(a) $(\Sigma_{\boldsymbol{q}}, P)$ is bounded;

(b) $(\Pi_{\boldsymbol{q}}, G)$ is bounded;

(c) there exists $d < \omega$ such that, for every $C \in \mathfrak{K}_{\boldsymbol{q}}$, there is a homomorphism $h \colon C' \to C$, for some $C' \in \mathfrak{K}_{\boldsymbol{q}}$ of depth $\leq d$.

Conditions (b) and (c) are equivalent for every (not necessarily focused) 1-CQ $\boldsymbol{q}$, in which case (a) is equivalent to (c) with an additional requirement that $h(r) = r$.

Proof. (a) $\Rightarrow$ (b) If $\Phi(x)$ is an FO-rewriting of $(\Sigma_{\boldsymbol{q}}, P)$, then

$$\exists x, y_1, \dots, y_n, \boldsymbol{z} \,\big(F(x) \land \boldsymbol{q}' \land \Phi(y_1) \land \dots \land \Phi(y_n)\big)$$

is an FO-rewriting of $(\Pi_{\boldsymbol{q}}, G)$, where $\boldsymbol{z}$ comprises the variables in $\boldsymbol{q}'$ different from $x, y_1, \dots, y_n$.

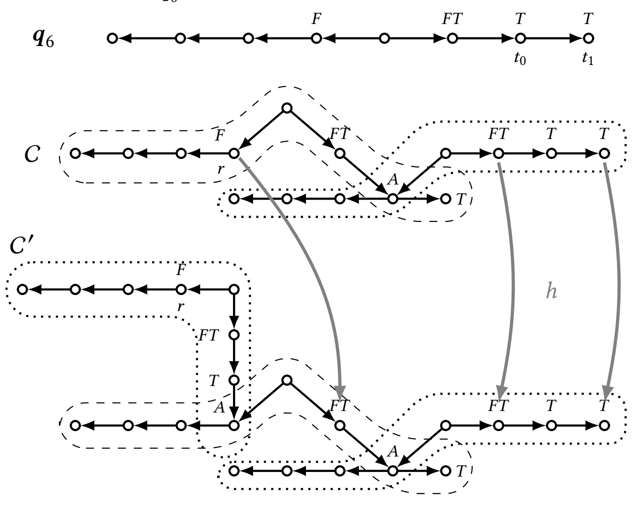
(b) $\Rightarrow$ (c) Let $\exists \boldsymbol{y}\,(\boldsymbol{q}_1 \lor \dots \lor \boldsymbol{q}_m)$ be a UCQ-rewriting of $(\Pi_{\boldsymbol{q}}, G)$, where the $\boldsymbol{q}_i$ are CQs and $\boldsymbol{y}$ comprises their variables. Treating the $\boldsymbol{q}_i$ as data instances, we obviously have $G \in \Pi_{\boldsymbol{q}}(\boldsymbol{q}_i)$, and so, for every i, $1 \leq i \leq m$, there is a homomorphism from some $C_i \in \mathfrak{K}_{\boldsymbol{q}}$ to $\boldsymbol{q}_i$. Let d be the maximum depth of the C_i, $i = 1, \dots, m$. Consider any $C \in \mathfrak{K}_{\boldsymbol{q}}$. Then there are homomorphisms $C_i \to \boldsymbol{q}_i \to C$, for some i, $1 \leq i \leq m$, the composition of which is the required h.

(c) $\Rightarrow$ (a) By Prop. 1 and (c), the sentence $\exists r, \boldsymbol{y}\,(C_1 \lor \dots \lor C_m)$, where the C_i are all of the cactuses of depth $\leq d$ with root-focus r and the remaining variables $\boldsymbol{y}$, is an FO-rewriting of $(\Pi_{\boldsymbol{q}}, G)$. We show that the formula $\Phi(r) = T(r) \lor \exists \boldsymbol{y}\,\big(C_1^\circ \lor \dots \lor C_m^\circ\big)$ is an FO-rewriting of $(\Sigma_{\boldsymbol{q}}, P)$. Let $P(a) \in \Sigma_{\boldsymbol{q}}(\mathcal{D})$, for some $\mathcal{D}$ and $a \in \mathsf{ind}(\mathcal{D})$. By Prop. 1, either $T(a) \in \mathcal{D}$, in which case $\mathcal{D} \models \Phi[a]$, or there is a homomorphism h from some $C^\circ \in \mathfrak{K}_{\boldsymbol{q}}$ to $\mathcal{D}$ such that $h(r) = a$. By (c), there is a homomorphism $g \colon C_i \to C$, for some $i \leq m$. As $\boldsymbol{q}$ is focused, $g(r) = r$, and so we can regard g as a $C_i^\circ \to C^\circ$ homomorphism. But then we obtain a homomorphism $hg \colon C_i^\circ \to \mathcal{D}$ with $hg(r) = a$, from which $\mathcal{D} \models \exists \boldsymbol{y}\, C_i^\circ[a]$. That $\mathcal{D} \models \Phi[a]$ implies $P(a) \in \Sigma_{\boldsymbol{q}}(\mathcal{D})$ is trivial. ❑

The next example illustrates the difference between focused and unfocused 1-CQs $\boldsymbol{q}$ as far as boundedness of $(\Pi_{\boldsymbol{q}}, G)$ and $(\Sigma_{\boldsymbol{q}}, P)$ is concerned.

Example 4. Consider the 1-CQ $\boldsymbol{q}_5$ from Example 1. Let C_k be the cactus obtained by applying **(bud)** k-times to $C_0 = \boldsymbol{q}_5$. There are homomorphisms $h \colon C_1 \to C_k$, for $k \geq 2$, and so both $(\Pi_{\boldsymbol{q}_5}, G)$ and $(\Delta_{\boldsymbol{q}_5}, G)$ are rewritable to the UCQ $C_0 \lor C_1$. For each such h, we have $h(r) = r$, so $\boldsymbol{q}_5$ is focused and the sirup $(\Sigma_{\boldsymbol{q}_5}, P)$ is bounded.

Now, consider the 1-CQ $\boldsymbol{q}_6$ below, where all of the arrows are labelled by R. It is not hard to see that, for every $C' \in \mathfrak{K}_{\boldsymbol{q}_6}$ of depth ≥ 2, there exist $C \in \mathfrak{K}_{\boldsymbol{q}_6}$ of depth ≤ 1 and a homomorphism $h \colon C \to C'$, so $(\Pi_{\boldsymbol{q}_6}, G)$ and $(\Delta_{\boldsymbol{q}_6}, G)$ are FO-rewritable. However, every such h maps the root-focus F-node r to an FT-node, and so $\boldsymbol{q}_6$ is not focused. In the picture below, C is obtained by budding at t_0, and C' by budding first at t_1 and then at t_0. Using Prop. 2, one can show that $(\Sigma_{\boldsymbol{q}_6}, P)$ is not bounded.



3 DECIDING BOUNDEDNESS OF SIRUPS

In this section, we prove the following:

Theorem 3. *The problems of deciding boundedness of monadic sirups $(\Sigma_{\boldsymbol{q}}, P)$ and monadic d-sirups $(\Delta_{\boldsymbol{q}}, G)$ are both 2ExpTime-hard.*

Before diving into technical details, we put this theorem into the context of related work.

3.1 Related results

That deciding program boundedness of arbitrary monadic datalog queries can be done in 2ExpTime was shown in 1988 using an automata-theoretic technique [18]. A matching lower bound for monadic queries with multiple recursive rules was finally settled in 2015 [9] using a construction from [8], which is based on the encoding of Turing machine computations from [12, 13]. For monadic sirups, the NP lower bound for the linear case [37] has remained so far the best known result (though, in view of Prop. 2 and the proof of [22, Theorem 9], it can be raised to PSpace).

Establishing a higher lower bound for monadic sirups is difficult for two obvious reasons: monadicity and singularity. The impact of arity and the number of recursive rules on deciding boundedness of datalog programs has been studied in great detail; see [26, 33] and further references therein. For example, boundedness was shown to be undecidable first for linear datalog programs of arity 4 [21], then for those of arity 2 with multiple recursive rules [37], which were encoded in a single rule at the expense of higher arity [1]; finally, boundedness was proved to be undecidable already for linear sirups of arity 3 [33].

Intuitively, the proofs of the lower bounds mentioned above use different rules in a datalog program in order to detect and exclude different ‘defects’ in possible computations of a Turing machine. Our task in the proof of Theorem 3 will be to design such an encoding of computations that can be verified by a single CQ.

3.2 Proof idea

To achieve this, similarly to [7–9, 12, 13], we represent computations of a Turing machine by means of annotated binary trees. The design of the tree-representation of computations is such that its structure can be connected with expansions (cactuses) of a given sirup via a series of small Boolean circuits, which is the main innovation of our construction.

More precisely, we use the criterion of Prop. 2 for testing boundedness. Our aim is, given any alternating Turing machine (ATM) $\boldsymbol{M}$ deciding a language in AExpSpace = 2ExpTime and an input $\boldsymbol{w}$, to construct a (dag-shaped) focused 1-CQ $\boldsymbol{q}$ of polynomial size such that the following holds:

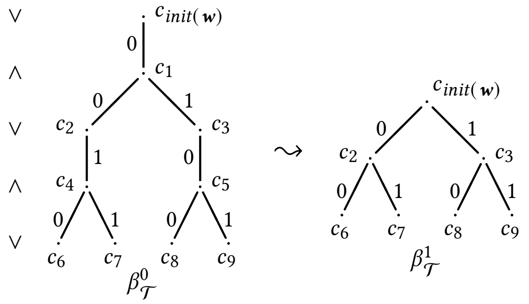
Lemma 4. *$\boldsymbol{M}$ rejects $\boldsymbol{w}$ iff there is $K < \omega$ such that every cactus $C \in \mathfrak{K}_{\boldsymbol{q}}$ contains a homomorphic image of some $C^- \in \mathfrak{K}_{\boldsymbol{q}}$ of depth at most K.*

We represent both the computation space of $\boldsymbol{M}$ on $\boldsymbol{w}$ and $\boldsymbol{q}$-cactuses by 01-*trees*: binary ditrees whose edges are labelled by 0 or 1, with siblings having different labels. On the one hand, we encode the computation space of $\boldsymbol{M}$ in such a way that checking whether an arbitrary 01-tree represents a rejecting computation-tree on $\boldsymbol{w}$ can be done by means of polynomially-many polynomial-size Boolean circuits (in fact, formulas). On the other hand, the 1-CQ $\boldsymbol{q}$ we associate with $\boldsymbol{M}$ and $\boldsymbol{w}$ has two solitary T-nodes, t_0 and t_1. Thus, we can regard the skeleton C^s of any cactus $C \in \mathfrak{K}_{\boldsymbol{q}}$ as a 01-tree, indicating which of t_0 or t_1 were budded. The 1-CQ $\boldsymbol{q}$ is assembled from gadgets implementing the Boolean circuits used for checking the above properties of computations.

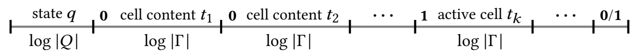
3.3 Connecting computations and cactuses

3.3.1 Encoding computations by 01*-trees.* We assume that we are given an ATM $\boldsymbol{M} = (Q, \Gamma, \delta, q_{init}, q_{accept}, q_{reject}, g)$ with states Q including q_{init}, q_{accept}, q_{reject}, tape alphabet Γ, transition function δ, and $g\colon Q \longrightarrow \{\wedge, \vee\}$. For any input $\boldsymbol{w} \in \Gamma^*$, a *configuration* of $\boldsymbol{M}$ is a triple containing information about the current state, the current position of the head, and the current content of the $2^{\boldsymbol{p}(|\boldsymbol{w}|)} = 2^{\boldsymbol{p}}$ tape-cells, for some polynomial $\boldsymbol{p}$. If its current state is q, then we call the configuration a *q-configuration.* The *full computation space* $\mathcal{T}_{\boldsymbol{M},\boldsymbol{w}}$ is a finite tree whose nodes are (labelled by) configurations, with its root being the initial configuration $c_{init(\boldsymbol{w})}$ (in state q_{init} reading the leftmost symbol of $\boldsymbol{w}$), the descendants generated by δ, and each leaf being either a q_{accept}- or a q_{reject}-configuration (a *halting configuration*). We assume that the depth of $\mathcal{T}_{\boldsymbol{M},\boldsymbol{w}}$ is $2^{\boldsymbol{p}(|\boldsymbol{w}|)}$, $q_{init}, q_{accept}, q_{reject}$ are $\vee$-states, every non-leaf has branching 2, and $\wedge$- and $\vee$-configurations alternate on each branch. A *computation-tree* (*of $\boldsymbol{M}$ on $\boldsymbol{w}$*) is a substructure $\mathcal{T}$ of $\mathcal{T}_{\boldsymbol{M},\boldsymbol{w}}$, which is a tree with root $c_{init(\boldsymbol{w})}$ such that every non-leaf $\wedge$-node ($\vee$-node) in $\mathcal{T}$ has both (respectively, exactly one) of its children from $\mathcal{T}_{\boldsymbol{M},\boldsymbol{w}}$ in $\mathcal{T}$. The tree $\mathcal{T}$ is *rejecting* if it has a q_{reject}-leaf and *accepting* otherwise. *$\boldsymbol{M}$ rejects $\boldsymbol{w}$* iff all computation-trees of $\boldsymbol{M}$ on $\boldsymbol{w}$ are rejecting, and *accepts $\boldsymbol{w}$* otherwise.

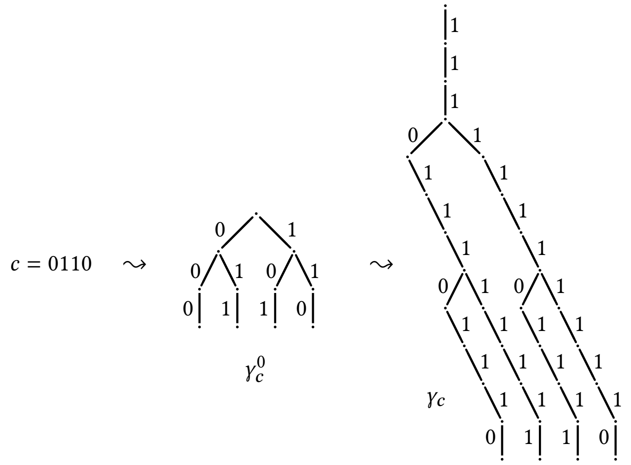
We encode a computation-tree $\mathcal{T}$ by an *infinite* 01-tree $\beta^+_{\mathcal{T}}$ via a series of steps as follows. First, by our assumption on binary branching, $\mathcal{T}$ can be considered as a (finite) 01-tree $\beta^0_{\mathcal{T}}$ (with its nodes still labelled by configurations). Next, we take the full binary 'substructure' $\beta^1_{\mathcal{T}}$ of the $\vee$-configurations in $\beta^0_{\mathcal{T}}$ as shown below:



(So the depth of $\beta^1_{\mathcal{T}}$ is $2^{\boldsymbol{p}-1}$.) The information about which child of each $\vee$-configuration is taken in $\beta^0_{\mathcal{T}}$ is provided in the encoding of the subsequent $\vee$-configuration. To achieve this, we fine-tune the 'configurations-as-binary-tree-leaves' representation of [12, 13] for our purpose. Let $\boldsymbol{d} = \boldsymbol{d}(|\boldsymbol{w}|) > \boldsymbol{p}(|\boldsymbol{w}|) = \boldsymbol{p}$ be a polynomial in $|\boldsymbol{w}|$ such that configurations can be encoded by a 01-sequence of length $2^{\boldsymbol{d}}$. We represent each $\vee$-configuration c by the 01-sequence

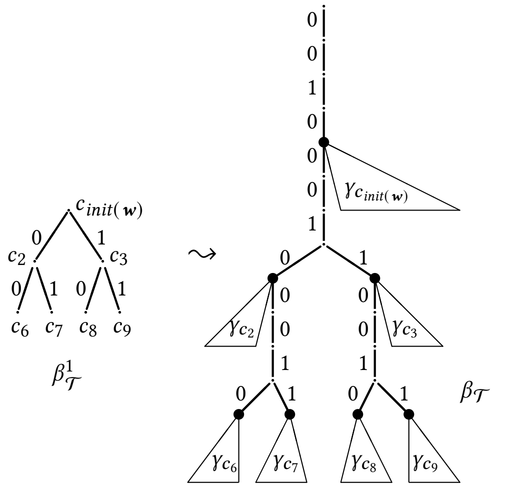


where the last bit is 0 (1) iff c's parent $\wedge$-configuration is a 0-child (1-child) of its parent. (By imposing some restrictions on Q and Γ, one can ensure that, given a $2^{\boldsymbol{d}}$-long 01-sequence, it is 'easy' to locate the first bit of each 'cell-representation' in it.) We encode the digits of this sequence as the leaves of a 01-tree γ^0_c of depth $\boldsymbol{d} + 1$ by taking first a full binary tree of depth d, and for each of its $2^{\boldsymbol{d}}$ leaves, taking a $*$-child whenever the corresponding digit in the sequence is $*$. (Throughout, we use $*$ in 01-sequences as a wildcard for 0 or 1.) Finally, we turn γ^0_c to a 01-tree γ_c of depth $4\boldsymbol{d} + 4$ by adding an incoming edge-pattern 111 above each node:



We call γ_c a *c-tree* (or, a *configuration-tree*, in general).

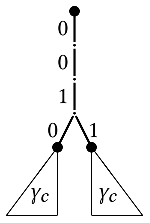
Next, we take the full binary 01-tree $\beta^1_{\mathcal{T}}$ above (whose nodes are labelled by $\vee$-configurations), and turn it to a 01-tree $\beta_{\mathcal{T}}$ (now without node labels) as follows. We add an incoming edge-pattern 0010 above the root, stick the root of a c-tree to each node labelled by some c, and add an outgoing edge-pattern 001 below each node before branching; see Fig. 1. Note that $\beta_{\mathcal{T}}$ is of depth $\boldsymbol{e} = \boldsymbol{e}(|\boldsymbol{w}|)$, for some exponential function $\boldsymbol{e}$. For any configuration c, if the c-tree γ_c is a substructure of $\beta_{\mathcal{T}}$, then we call the root node of γ_c a *main node* (*of c*) and say that it *represents c in $\beta_{\mathcal{T}}$*; see •-nodes in Fig. 1.



$c_{init(\boldsymbol{w})}$
c_2
c_3
c_6 c_7 c_8 c_9
$\beta^1_{\mathcal{T}}$
$\gamma_{c_{init(\boldsymbol{w})}}$
γ_{c_2} γ_{c_3} γ_{c_6} γ_{c_7} γ_{c_8} γ_{c_9}
$\beta_{\mathcal{T}}$

Figure 1: The 01-tree $\beta_{\mathcal{T}}$.

We also consider an infinite 'version' of $\beta_{\mathcal{T}}$. We obtain the infinite 01-tree $\beta^+_{\mathcal{T}}$ from $\beta_{\mathcal{T}}$ by repeatedly sticking the following pattern to the main node of each halting configuration c:



In other words, we assume δ to be such that after reaching a halting configuration c, c is repeated forever on every branch of $\mathcal{T}_{\boldsymbol{M},\boldsymbol{w}}$.

An infinite 01-tree β is *ideal* if it can be constructed by starting with $\beta^+_{\mathcal{T}_0}$, for some computation-tree $\mathcal{T}_0$, and then by repeatedly attaching to each of the leaves (that must be leaves of some configuration-tree) the root of some $\beta^+_{\mathcal{T}}$, where each $\mathcal{T}$ can be any computation-tree.

Observe that each branch of an ideal tree is infinite. We are interested in finite 'middle-bits' of ideal trees. We call a subtree of an ideal tree having a main node (of not necessarily $c_{init(\boldsymbol{w})}$) as root a *desired tree*. Given some $M < \omega$ and a 01-tree β, by an *M-cut* of β we mean the 01-tree obtained by cutting all longer than M branches in β at depth M. The pretty baroque design above ensures that there is a polynomial list of polynomially detectable properties that identify desired trees up to isomorphism (see Claim 4.1 below). In the next subsection, we discuss these properties.

3.3.2 Characterising exponential computations polynomially.

We investigate certain polynomial neighbourhoods of nodes in 01-trees, and collect a polynomial list of their properties that fully characterise those situations that can occur in a desired tree. For each of the properties P below, in Sec. 3.4 we describe in detail how to give a small Boolean circuit φ_P having specific *input-types* such that, when evaluated at a node $\mathfrak{a}$ of some 01-tree β, P fails at $\mathfrak{a}$ iff there is some 01-sequence $\boldsymbol{b}$ such that $\boldsymbol{b}$ is gathered from the neighbourhood of $\mathfrak{a}$ in β according to the input-types of φ_P and $\varphi_P[\boldsymbol{b}] = 1$.

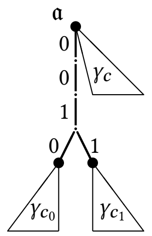
Given $n < \omega$ and a node $\mathfrak{a}$ of depth $\leq n$ in a 01-tree β, for any $k \leq n$, we denote by $P^k_{\mathfrak{a}}$ the k-long suffix of the path ending at $\mathfrak{a}$ in β. To begin with, observe that every path in a desired tree that is longer than $4\boldsymbol{d} + 6$ must contain a main node, and main nodes can be identified by the property 'the path leading to the node ends with a 001∗-pattern'. So, given a node $\mathfrak{a}$ in a 01-tree β, we say that $\mathfrak{a}$ *is good in* β, if either the depth of $\mathfrak{a}$ in β is $< 4\boldsymbol{d} + 11$, or $P^{4\boldsymbol{d}+11}_{\mathfrak{a}}$ contains a 001∗-pattern; see Sec. 3.4.1.

Next, we describe proper branching-patterns in a desired tree. It is easy to see that if the path leading to a node $\mathfrak{a}$ does contain a 001∗-pattern, then there exist unique k, ℓ and w such that $4 \leq k \leq 4\boldsymbol{d}+11$, $P^k_{\mathfrak{a}} = 001{*}(111{*})^{\ell} w$, and either $\ell \leq \boldsymbol{d}$ and w is a prefix of 001, or $\ell < \boldsymbol{d}$ and w is a prefix of 111. Moreover, ℓ and w characterise the children of $\mathfrak{a}$. We call $\mathfrak{a}$ *properly branching in* β if the following conditions **(pb1)**–**(pb4)** hold:

- **(pb1)** if either w is empty and $\ell = 0$, or $w = 001$, or $w = 111$ and $\ell < \boldsymbol{d} - 1$, then $\mathfrak{a}$ has two children;
- **(pb2)** if either w is empty and $0 < \ell < \boldsymbol{d}$, or $w = 1$, or $w = 11$, or $w = 00$, then $\mathfrak{a}$ has no 0-child;
- **(pb3)** if either w is empty and $\ell = \boldsymbol{d}$, or $w = 0$, then $\mathfrak{a}$ has no 1-child;
- **(pb4)** if $w = 111$ and $\ell = \boldsymbol{d} - 1$, then $\mathfrak{a}$ has only one child;

see Sec. 3.4.2. Note that leaves are never properly branching.

Next, we ensure that the 'building-block' computation-trees in an ideal tree are properly represented in a 01-tree β (provided that all of its nodes are properly branching). First, after each leaf of a configuration-tree, the representation of a proper computation-tree from $c_{init(\boldsymbol{w})}$ should start. In such a case, the main node $\mathfrak{a}$ of $c_{init(\boldsymbol{w})}$ can be identified by an incoming path ending with a 111∗001∗-pattern. Then detecting whether the c-tree with root $\mathfrak{a}$ is not a $c_{init(\boldsymbol{w})}$-tree requires checking polynomial information. We call $\mathfrak{a}$ *properly initialising in* β if whenever the depth of $\mathfrak{a}$ in β is ≥ 8, $P^8_{\mathfrak{a}}$ is of the form 111∗001∗, and $\mathfrak{a}$ is the root of a c-tree, then $c = c_{init(\boldsymbol{w})}$; see Sec. 3.4.4. Second, the computation steps described by δ should be properly represented. We call $\mathfrak{a}$ *properly computing in* β if, whenever the following pattern is present at $\mathfrak{a}$ in β



then the triple (c, c_0, c_1) of ∨-configurations 'matches' the transition function δ of $\boldsymbol{M}$. In order to detect that this is not the case, one needs to check polynomial information 'around' $\mathfrak{a}$ in β; see Sec. 3.4.3.

We call $\mathfrak{a}$ *correct in* β if $\mathfrak{a}$ is good, properly branching, properly initialising and properly computing in β. Otherwise, $\mathfrak{a}$ is called *incorrect in* β. Now it is straightforward to show that the collected properties of $\mathfrak{a}$-neighbourhoods characterise desired trees:

Claim 4.1. *For any $M < \omega$, any 01-tree β and any node $\mathfrak{a}$ with $P^4_{\mathfrak{a}} = 001{*}$, the M-cut $\beta^M_{\mathfrak{a}}$ of the subtree of β with root $\mathfrak{a}$ is isomorphic to the M-cut of a desired tree iff every node of depth $< M$ in $\beta^M_{\mathfrak{a}}$ is correct in $\beta^M_{\mathfrak{a}}$.*

We also need to detect the presence of nodes representing q_{reject}-configurations in computation-trees; see Sec. 3.4.5.

3.3.3 Cactus homomorphisms vs rejecting computations. As our 1-CQ $\boldsymbol{q}$ will have one solitary F-node and two solitary T-nodes t_0 and t_1, there are four possible kinds of non-root segments in any cactus $C \in \mathfrak{K}_{\boldsymbol{q}}$ denoted $\boldsymbol{q}^-_{TT}$, $\boldsymbol{q}^-_{AT}$, $\boldsymbol{q}^-_{TA}$ and $\boldsymbol{q}^-_{AA}$. For example, $\boldsymbol{q}^-_{TT}$ is obtained by replacing the F-label of the solitary F-node in $\boldsymbol{q}$ by A; in cactuses different from $\boldsymbol{q}$, leaf segments are of this form. The segment $\boldsymbol{q}^-_{TA}$ is obtained by replacing both the F-label of the solitary F-node and the T-label of the t_1-node by A. As $\boldsymbol{q}$ itself does not contain A, if $h\colon C \to C'$ is a homomorphism, for some $C, C' \in \mathfrak{K}_{\boldsymbol{q}}$, then the focus of every non-root segment $\mathfrak{s}$ in C (labelled by A) is mapped by h to the focus of some non-root segment $\mathfrak{s}'$ in C'. Our $\boldsymbol{q}$ will also satisfy **(foc)**: for every homomorphism $h\colon C \to C'$ between cactuses $C, C' \in \mathfrak{K}_{\boldsymbol{q}}$, h maps the only solitary F-node in C (the focus of its root segment) to the only solitary F-node in C'. So we say that h *maps* a segment $\mathfrak{s}$ into a segment $\mathfrak{s}'$ if h maps the focus of $\mathfrak{s}$ to the focus of $\mathfrak{s}'$.

Now the proof of Theorem 3 can be completed as follows: Using Claim 4.1, we prove in Appendix A that to obtain Lemma 4 it suffices to construct a 1-CQ $\boldsymbol{q}$ such that **(foc)** holds and, for every $C \in \mathfrak{K}_{\boldsymbol{q}}$,

(leaf) there is a homomorphism $h\colon \boldsymbol{q}^-_{TT} \to C$ mapping $\boldsymbol{q}^-_{TT}$ into some non-leaf segment $\mathfrak{s}$ of C iff either $\mathfrak{s}$ is incorrect or $\mathfrak{s}$ represents a q_{reject}-configuration in the skeleton C^s of C;

(branch) if h maps $\boldsymbol{q}^-_{TT}$ into a non-leaf segment $\mathfrak{s}$ that is not properly branching in C^s due to violating **(pb1)**, but $\mathfrak{s}$ is correct in C^s according to the other properties, then
- $h(t_0) = t_0$ and $h(t_1) \neq t_0$, if $\mathfrak{s} = \boldsymbol{q}^-_{TA}$;
- $h(t_1) = t_1$ and $h(t_0) \neq t_1$, if $\mathfrak{s} = \boldsymbol{q}^-_{AT}$.

After defining the focused 1-CQ $\boldsymbol{q}$ in Secs. 3.5.1–3.5.3, we show in Sec. 3.5.4 that, for every $C \in \mathfrak{K}_{\boldsymbol{q}}$, **(leaf)** and **(branch)** are satisfied, completing the proof of Lemma 4.

3.4 Boolean formulas

We describe polynomially-many polynomial-size Boolean circuits (in fact, Boolean formulas) that test the (failure) of the properties of a node $\mathfrak{a}$ in a 01-tree β, given in Sec. 3.3.2. For each such formula, we also define some *input-types*, describing where around the tested node $\mathfrak{a}$ the input 01-sequence for the formula should be 'gathered' from. In defining the input-types we use the following terminology: for $n < \omega$, the *n-long uppath* (*of* $\mathfrak{a}$ *in* β) is the reverse of the n-long suffix of the path ending at $\mathfrak{a}$ in β; while an *n-long downpath* is the n-long prefix of some path starting at $\mathfrak{a}$ in β.

3.4.1 Checking goodness. One can clearly define a Boolean formula $\textsc{Good}(x_1, \dots, x_{4\boldsymbol{d}+11})$ such that, for any $4\boldsymbol{d} + 11$-long 01-sequence $\boldsymbol{b}$, $\textsc{Good}[\boldsymbol{b}] = 1$ iff $\boldsymbol{b}$ does not contain the reverse of a 001$*$-pattern. The input should be gathered from the $4\boldsymbol{d} + 11$-long uppath.

3.4.2 Checking proper branching-patterns. For each of conditions **(pb1)**–**(pb4)** in Sec. 3.3.2, we have a different family of formulas.

(pb1) For every k with $4 \leq k \leq 4\boldsymbol{d} + 11$, we define a Boolean formula $\textsc{MustBranch}^k(x_1, \dots, x_k)$ such that, for any k-long 01-sequence $\boldsymbol{b} = (b_1, \dots, b_k)$, $\varphi^k[\boldsymbol{b}] = 1$ iff $\boldsymbol{b}$ is the reverse of a sequence of the form $001{*}(111{*})^\ell w$, where either w is empty and $\ell = 0$, or $w = 001$, or $w = 111$ and $\ell < \boldsymbol{d} - 1$. For example, if $k = 4$ then we have

$$\textsc{MustBranch}^4[\boldsymbol{b}] = 1 \quad \text{iff} \quad \boldsymbol{b} \text{ is the reverse of } 001{*}.$$

The input should be gathered from the k-long uppath.

(pb2) For every k with $4 \leq k \leq 4\boldsymbol{d} + 11$, we define a Boolean formula $\textsc{NoBranch}^k_0(x_1, \dots, x_{k+1})$ such that, for any $k+1$-long 01-sequence $\boldsymbol{b} = (b_1, \dots, b_{k+1})$, $\textsc{NoBranch}^k_0[\boldsymbol{b}] = 1$ iff $b_{k+1} = 0$ and $(b_1, \dots, b_k)$ is the reverse of a sequence of the form $001{*}(111{*})^\ell w$, where either w is empty and $0 < \ell < \boldsymbol{d}$, or $w = 1$, or $w = 11$, or $w = 00$. The input for $(x_1, \dots, x_k)$ should be gathered from the k-long uppath, and for x_{k+1} from a 1-long downpath.

(pb3) For every k with $4 \leq k \leq 4\boldsymbol{d} + 11$, we define a polynomial size Boolean formula $\textsc{NoBranch}^k_1(x_1, \dots, x_{k+1})$ such that, for any $k + 1$-long 01-sequence $\boldsymbol{b} = (b_1, \dots, b_{k+1})$, $\textsc{NoBranch}^k_1[\boldsymbol{b}] = 1$ iff $b_{k+1} = 1$ and $(b_1, \dots, b_k)$ is the reverse of a sequence of the form $001{*}(111{*})^\ell w$, where either w is empty and $\ell = \boldsymbol{d}$, or $w = 0$. The input for $(x_1, \dots, x_k)$ should be gathered from the k-long uppath, and for x_{k+1} from a 1-long downpath.

(pb4) For every k with $4 \leq k \leq 4\boldsymbol{d} + 11$, we define a polynomial size Boolean formula $\textsc{NoBranch}^k(x_1, \dots, x_{k+2})$ such that, for any $k + 2$-long 01-sequence $\boldsymbol{b} = (b_1, \dots, b_{k+2})$, $\textsc{NoBranch}^k[\boldsymbol{b}] = 1$ iff $b_{k+1} \neq b_{k+2}$ and $(b_1, \dots, b_k)$ is the reverse of a sequence of the form $001{*}(111{*})^\ell w$, where $w = 111$ and $\ell = \boldsymbol{d} - 1$. The input for $(x_1, \dots, x_k)$ should be gathered from the k-long uppath, and for each of x_{k+1} and x_{k+2} from a 1-long downpath.

3.4.3 Checking proper computation steps. This is an adaptation of the technique of [12, 13] to our representation. Suppose that

- n_Q is such that n_Q-long 01-sequences are in one-to-one correspondence with the states in Q,
- n_Γ is such that $(n_\Gamma - 1)$-long 01-sequences are in one-to-one correspondence with the symbols in Γ,
- $n_Q + 2^{\boldsymbol{p}} \cdot n_\Gamma + 1 = 2^{\boldsymbol{d}}$

(see the picture in Sec. 3.3.1 on representing configurations with $2^{\boldsymbol{d}}$-long 01-sequences).

First, we define a Boolean formula $\textsc{Head}(x_1, \dots, x_{4(\boldsymbol{d}+1)})$ such that, for any $4(\boldsymbol{d} + 1)$-long 01-sequence $\boldsymbol{b}$, $\textsc{Head}[\boldsymbol{b}] = 1$ iff $\boldsymbol{b}$ describes a path in a desired tree starting at a main node with 1, and ending at the first bit of the representation of some cell-content of some configuration c (that is, it is the $(n_Q + i \cdot n_\Gamma + 1)$th bit of the 01-sequence representing c, for some $i < 2^{\boldsymbol{p}}$). Similarly, for $* = 0, 1$, we define a Boolean formula $\textsc{Head}^*(x_1, \dots, x_{4(\boldsymbol{d}+1)+4})$ such that, for any $4(\boldsymbol{d}+1)+4$-long 01-sequence $\boldsymbol{b}$, $\textsc{Head}^*[\boldsymbol{b}] = 1$ iff $\boldsymbol{b}$ describes a path in a desired tree starting at a main node with $001{*}1$, and ending at the first bit of the representation of some cell-content of some configuration.

Next, we define a Boolean formula

$$\textsc{SameCell}(x_1, \dots, x_{4(\boldsymbol{d}+1)}, y_1, \dots, y_{4(\boldsymbol{d}+1)+4}, z_1, \dots, z_{4(\boldsymbol{d}+1)+4})$$

such that, for any $4(\boldsymbol{d}+1)$-long 01-sequence $\boldsymbol{b}$ and $4(\boldsymbol{d}+1)+4$-long 01-sequences $\boldsymbol{b}^0, \boldsymbol{b}^1$, we have $\textsc{SameCell}[\boldsymbol{b}, \boldsymbol{b}^0, \boldsymbol{b}^1] = 1$ iff $\textsc{Head}[\boldsymbol{b}] = 1$, $\textsc{Head}^*[\boldsymbol{b}^*] = 1$, and the three paths $\boldsymbol{b}, \boldsymbol{b}^0, \boldsymbol{b}^1$ end at the same cell of a configuration and its two children-configurations. (The number i of this cell, for some $i < 2^{\boldsymbol{p}}$, can be identified from $\boldsymbol{b}$.)

Next, we define a Boolean formula $\textsc{State}(x_1, \dots, x_{4(\boldsymbol{d}+1)\cdot n_Q})$ such that, for any $4(\boldsymbol{d}+1)$-long 01-sequences $\boldsymbol{b}_1, \dots, \boldsymbol{b}_{n_Q}$,

$$\textsc{State}[\boldsymbol{b}_1, \dots, \boldsymbol{b}_{n_Q}] = 1$$

iff for every $j \leq n_Q$, $\boldsymbol{b}_j$ describes a $4(\boldsymbol{d}+1)$-long path in a desired tree starting at a main node with 1, and ending at the jth bit of the representation of some configuration c. (So, whenever $\boldsymbol{b}_j = (b_j^1, \dots, b_j^{4(\boldsymbol{d}+1)})$ for each $j \leq n_Q$, then $(b_1^{4(\boldsymbol{d}+1)}, \dots, b_{n_Q}^{4(\boldsymbol{d}+1)})$ encodes a state in Q.) Similarly, for $* = 0, 1$, we define a Boolean formula $\textsc{State}^*(x_1, \dots, x_{(4(\boldsymbol{d}+1)+4)\cdot n_Q})$ such that, for any $4(\boldsymbol{d}+1)+4$-long 01-sequences $\boldsymbol{b}_1 \dots, \boldsymbol{b}_{n_Q}$, $\textsc{State}^*[\boldsymbol{b}_1 \dots, \boldsymbol{b}_{n_Q}] = 1$ iff for every $j \leq n_Q, \boldsymbol{b}_j$ describes a path in a desired tree starting at a main node with $001{*}1$, and ending at the jth bit of the representation of some configuration.

Next, we define a Boolean formula $\textsc{Cell}(x_1, \dots, x_{4(\boldsymbol{d}+1)\cdot n_\Gamma})$ such that, for any $4(\boldsymbol{d}+1)$-long 01-sequences $\boldsymbol{b}_1 \dots, \boldsymbol{b}_{n_\Gamma}$,

$$\textsc{Cell}[\boldsymbol{b}_1 \dots, \boldsymbol{b}_{n_\Gamma}] = 1$$

iff there is $i < 2^{\boldsymbol{p}}$ such that, for every $j \leq n_\Gamma$, $\boldsymbol{b}_j$ describes a path in a desired tree starting at a main node with 1, and ending at the jth bit of the representation of the ith cell's content in some configuration. (So, $\textsc{Head}[\boldsymbol{b}_1] = 1$, and if $\boldsymbol{b}_j = (b_j^1, \dots, b_j^{4(\boldsymbol{d}+1)})$ for each $j \leq n_\Gamma$, then $(b_2^{4(\boldsymbol{d}+1)}, \dots, b_{n_\Gamma}^{4(\boldsymbol{d}+1)})$ encodes a symbol in Γ.) Similarly, for $* = 0, 1$, we define a polynomial size Boolean formula $\textsc{Cell}^*(x_1, \dots, x_{(4(\boldsymbol{d}+1)+4)\cdot n_\Gamma})$ such that, for any $4(\boldsymbol{d}+1)+4$-long 01-sequences $\boldsymbol{b}_1 \dots, \boldsymbol{b}_{n_\Gamma}$, $\textsc{Cell}^*[\boldsymbol{b}_1 \dots, \boldsymbol{b}_{n_\Gamma}] = 1$ iff there is $i < 2^{\boldsymbol{p}}$ such that, for every $j \leq n_\Gamma$, $\boldsymbol{b}_j$ describes a path in a desired tree starting at a main node with $001{*}1$, and ending at the jth bit of the representation of the ith cell's content in some configuration (In particular, $\textsc{Head}^*[\boldsymbol{b}_1] = 1$.)

Next, for $z \in \{0, 1\}$, we take the following tuples of variables:

- $\boldsymbol{s} = (s_1, \dots, s_{4(\boldsymbol{d}+1)\cdot n_Q})$, which is to be gathered from n_Q-many $4(\boldsymbol{d}+1)$-long downpaths (representing the $\vee$-state in c);
- $\boldsymbol{v} = (v_1, \dots, v_{4(\boldsymbol{d}+1)\cdot n_\Gamma})$, which needs to be gathered from n_Γ-many $4(\boldsymbol{d}+1)$-long downpaths (representing the active cell's content in c);
- $\boldsymbol{s}^0 = (s_1^0, \dots, s^0_{(4(\boldsymbol{d}+1)+4)\cdot n_Q})$, $\boldsymbol{s}^1 = (s_1^1, \dots, s^1_{(4(\boldsymbol{d}+1)+4)\cdot n_Q})$, each of which needs to be gathered from n_Q-many $4(\boldsymbol{d}+1)+4$-long downpaths (representing the $\vee$-states in c_0, c_1);
- $\boldsymbol{t}^z = (t_1^z, \dots, t^z_{4(\boldsymbol{d}+1)\cdot n_\Gamma})$, $\boldsymbol{t}^{z0} = (t_1^{z0}, \dots, t^{z0}_{(4(\boldsymbol{d}+1)+4)\cdot n_\Gamma})$, and $\boldsymbol{t}^{z1} = (t_1^{z1}, \dots, t^{z1}_{(4(\boldsymbol{d}+1+4))\cdot n_\Gamma})$, for $z \in \{0, 1\}$, where $\boldsymbol{t}^z$ is to be gathered from n_Γ-many $4(\boldsymbol{d}+1)$-long downpaths, and each of $\boldsymbol{t}^{z0}$ and $\boldsymbol{t}^{z1}$ is to be gathered from n_Γ-many $4(\boldsymbol{d}+1)+4$-long downpaths ($\boldsymbol{t}^z, \boldsymbol{t}^{z0}, \boldsymbol{t}^{z1}$ represent the ith cell's contents in c, c_0, c_1, for some $i < 2^{\boldsymbol{p}}$, when the z $\wedge$-child of c is taken in $\mathcal{T}_{\boldsymbol{M},x}$);
- $\boldsymbol{z}^0 = (z_1^0, \dots, z^0_{4(\boldsymbol{d}+1)+4})$ and $\boldsymbol{z}^1 = (z_1^1, \dots, z^1_{4(\boldsymbol{d}+1)+4})$, each of which needs to be gathered from a $4(\boldsymbol{d}+1)+4$-long downpath (representing the respective bits identifying the parent $\wedge$-configuration of c_0 and c_1).

For $z \in \{0, 1\}$, we can define a Boolean formula $\textsc{Step}^z$ such that, for any 01-sequence $\boldsymbol{b} = (\boldsymbol{s}, \boldsymbol{v}, \boldsymbol{s}^0, \boldsymbol{s}^1, \boldsymbol{t}, \boldsymbol{t}^0, \boldsymbol{t}^1, \boldsymbol{z}^0, \boldsymbol{z}^1)$, $\textsc{Step}^z[\boldsymbol{b}] = 1$ iff $\boldsymbol{z}^0 = 001011\dots1z$, $\boldsymbol{z}^1 = 001111\dots1z$, $\textsc{State}[\boldsymbol{s}] = 1$, $\textsc{Cell}[\boldsymbol{v}] = 1$, $\textsc{State}^0[\boldsymbol{s}^0] = 1$, $\textsc{State}^1[\boldsymbol{s}^1] = 1$, $\textsc{Cell}[\boldsymbol{t}] = 1$, $\textsc{Cell}^0[\boldsymbol{t}^0] = 1$, $\textsc{Cell}^1[\boldsymbol{t}^1] = 1$, and

$$\textsc{SameCell}[t_1, \dots, t_{4(\boldsymbol{d}+1)}, t_1^0, \dots, t^0_{4(\boldsymbol{d}+1)+4}, t_1^1, \dots, t^1_{4(\boldsymbol{d}+1)+4}] = 1,$$

but the information provided by $\boldsymbol{b}$ is inconsistent with the transition function δ in the sense that when the z $\wedge$-child of c is chosen as the common parent of c_0 and c_1 in the computation-tree $\mathcal{T}$, the content-triple of the ith cells of c, c_0, c_1 is wrong, where i is identified from the input in $(t_1, \dots, t_{4(\boldsymbol{d}+1)}, t_1^0, \dots, t^0_{4(\boldsymbol{d}+1)+4}, t_1^1, \dots, t^1_{4(\boldsymbol{d}+1)+4})$.

Finally, we define $\textsc{Step}(\boldsymbol{s}, \boldsymbol{v}, \boldsymbol{s}^0, \boldsymbol{s}^1, \boldsymbol{t}^0, \boldsymbol{t}^{00}, \boldsymbol{t}^{01}, \boldsymbol{t}^1, \boldsymbol{t}^{10}, \boldsymbol{t}^{11}, \boldsymbol{z}^0, \boldsymbol{z}^1)$ as the disjunction

$$\textsc{Step}^0(\boldsymbol{s}, \boldsymbol{v}, \boldsymbol{s}^0, \boldsymbol{s}^1, \boldsymbol{t}^0, \boldsymbol{t}^{00}, \boldsymbol{t}^{01}, \boldsymbol{z}^0, \boldsymbol{z}^1) \vee \\ \textsc{Step}^1(\boldsymbol{s}, \boldsymbol{v}, \boldsymbol{s}^0, \boldsymbol{s}^1, \boldsymbol{t}^1, \boldsymbol{t}^{10}, \boldsymbol{t}^{11}, \boldsymbol{z}^0, \boldsymbol{z}^1).$$

It is not hard to see that there is $\boldsymbol{b}$ with $\textsc{Step}[\boldsymbol{b}] = 1$ iff the information about the configuration-triple (c, c_0, c_1) encoded in $\boldsymbol{b}$ is inconsistent with δ.

3.4.4 Checking proper initialisation. We take the following tuples of variables:

- $\boldsymbol{y} = (y_1, \dots, y_8)$, which is to be gathered from the 8-long uppath (representing the last 8-bits of the path leading to the main node of a configuration c);
- $\boldsymbol{s} = (s_1, \dots, s_{4(\boldsymbol{d}+1)\cdot n_Q})$, to be gathered from n_Q-many $4(\boldsymbol{d}+1)$-long downpaths (representing the state in c);
- $\boldsymbol{w}^j = (w_1^j, \dots, w^j_{4(\boldsymbol{d}+1)\cdot n_\Gamma})$, for $j < |\boldsymbol{w}|$, each of which needs to be gathered from n_Γ-many $4(\boldsymbol{d}+1)$-long downpaths (representing the contents of the first $|\boldsymbol{w}|$-many cells in c);
- $\boldsymbol{t} = (t_1, \dots, t_{4(\boldsymbol{d}+1)\cdot n_\Gamma})$, which needs to be gathered from n_Γ-many $4(\boldsymbol{d}+1)$-long downpaths (representing the contents of some cell in c).

Then we can define a Boolean formula $\textsc{Init}$ such that, for any 01-sequence $\boldsymbol{b} = (\boldsymbol{y}, \boldsymbol{s}, \boldsymbol{w}^1, \dots, \boldsymbol{w}^{|\boldsymbol{w}|}, \boldsymbol{t})$, $\textsc{Init}[\boldsymbol{b}] = 1$ iff $\boldsymbol{y}$ is the reverse of some pattern $111{*}001{*}$, $\textsc{State}[\boldsymbol{s}] = 1$, for all $1 \leq j \leq |x|$, $\textsc{Cell}[\boldsymbol{w}^j] = 1$ and $(w_1^j, \dots, w^j_{4(\boldsymbol{d}+1)})$ ends at the $(n_Q + (j-1)\cdot n_\Gamma + 1)$th bit of the 01-sequence representing configurations, $\textsc{Cell}[\boldsymbol{t}] = 1$, but the information provided by $\boldsymbol{b}$ is inconsistent with c being $c_{init(\boldsymbol{w})}$ (at the cell identified by the prefix $(t_1, \dots t_{4(\boldsymbol{d}+1)})$ of $\boldsymbol{t}$).

3.4.5 Representing q_{reject}-configurations. This can clearly be done by a formula $\textsc{Reject}(s_1, \dots, s_{4(\boldsymbol{d}+1)\cdot n_Q})$ for which $\textsc{Reject}[\boldsymbol{s}] = 1$ iff $\textsc{State}[\boldsymbol{s}] = 1$ and the sequence

$$(s_{4(\boldsymbol{d}+1)}, s_{4(\boldsymbol{d}+1)\cdot 2}, \dots, s_{4(\boldsymbol{d}+1)\cdot n_Q})$$

encodes q_{reject}. The input should be gathered from n_Q-many $4(\boldsymbol{d}+1)$-long downpaths.

3.5 Query design

The dag-shaped 1-CQ $\boldsymbol{q}$ having one solitary F-node, two solitary T-nodes and many FT-twins will be such that properties **(foc)**, **(leaf)** and **(branch)** given in Sec. 3.3.3 hold, for all $\mathcal{C}, \mathcal{C}' \in \mathfrak{K}_{\boldsymbol{q}}$.

3.5.1 Overall query structure. To simplify notation, in our pictures we omit the R-labels from R-arrows, and use extra labels (different from F, T) on nodes, say B on a, as a shorthand for a B-arrow (a, a') to a *fresh* node a'. Letters other than upper case italics (greek, lower case italics and bold) are used as pointers and are not part of $\boldsymbol{q}$.

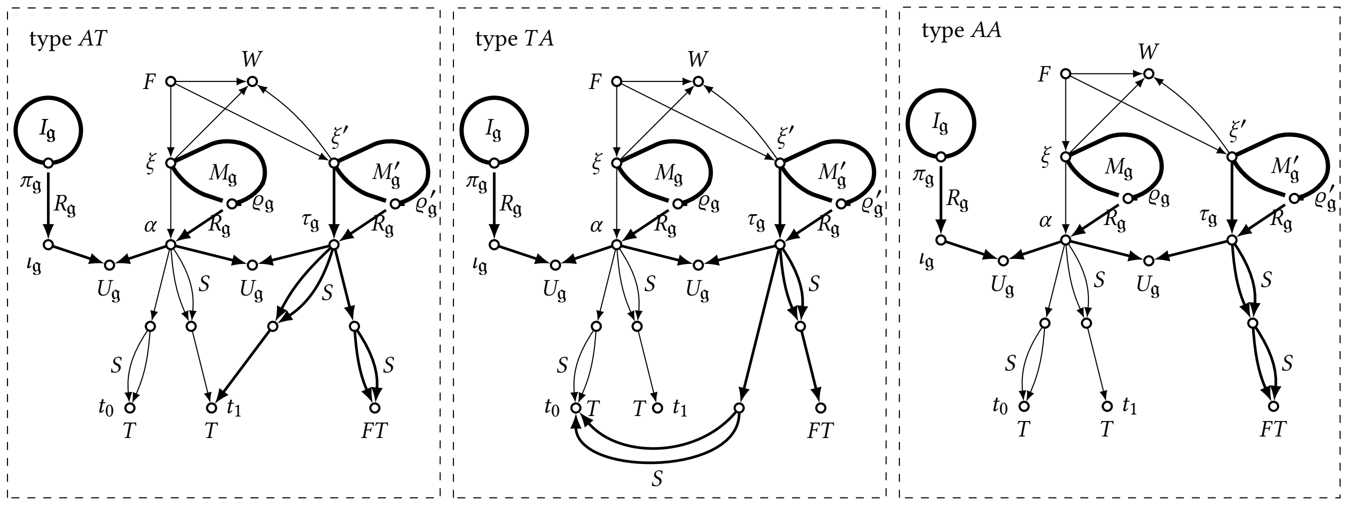
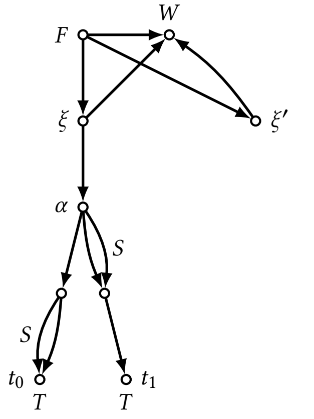


Figure 2: Frames of type AT, TA **and** AA.

The 1-CQ $\boldsymbol{q}$ has the following simple *base block* containing all of the solitary F- and T-nodes of $\boldsymbol{q}$:



Wired to the base in $\boldsymbol{q}$ are *gadgets* $\mathfrak{g}$ that *implement* the Boolean formulas $\varphi_\mathfrak{g}$ defined in Sec. 3.4. Each gadget $\mathfrak{g}$ has four components: two isomorphic copies of its *main block* $M_\mathfrak{g}$ and $M'_\mathfrak{g}$, an *input block* $I_\mathfrak{g}$ and a *frame*. The frame wires the gadget to the base and can be of one of the three *types* AT, TA and AA, which are shown in Fig. 2 (with the base block being indicated in each case by thin lines). We say that $\mathfrak{g}$ is of type Z if its frame is of type Z. The frame of $\mathfrak{g}$ has a few distinguished nodes: $\pi_\mathfrak{g}$ and $\iota_\mathfrak{g}$ via which $I_\mathfrak{g}$ is $R_\mathfrak{g}$-wired to the base block, $\varrho_\mathfrak{g}$ via which $M_\mathfrak{g}$ is $R_\mathfrak{g}$-wired to the base block, $\varrho'_\mathfrak{g}$ and $\tau_\mathfrak{g}$ via which $M'_\mathfrak{g}$ is $R_\mathfrak{g}$-wired to the base block (where the edge-labelling $R_\mathfrak{g}$ is also unique for gadget $\mathfrak{g}$), the single FT-twin of $\mathfrak{g}$ (none of $I_\mathfrak{g}$, $M_\mathfrak{g}$ and $M'_\mathfrak{g}$ contains any FT-twins), and two nodes labelled by $U_\mathfrak{g}$.

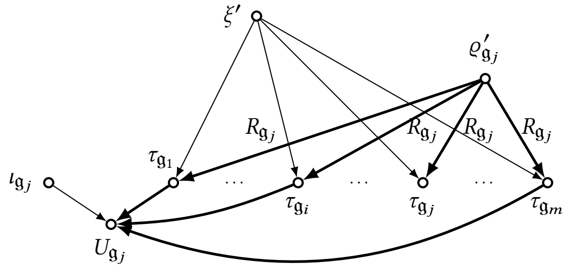
It is easy to see that $\boldsymbol{q}$ satisfies **(foc)**: its F-node has successors, while none of the FT-nodes does. Further, we observe that if $h\colon \boldsymbol{q}^-_{TT} \to C$ is a homomorphism mapping $\boldsymbol{q}^-_{TT}$ into some non-leaf segment $\mathfrak{s}$, then there must be a gadget $\mathfrak{g}$ such that $h(\alpha) = \tau_\mathfrak{g}$, for the α-node in $\boldsymbol{q}^-_{TT}$ and the $\tau_\mathfrak{g}$-node in $\mathfrak{s}$. Then, because of the $U_\mathfrak{g}$-nodes, $h(\iota_\mathfrak{g}) = \alpha$ must hold, for the $\iota_\mathfrak{g}$-node in $\boldsymbol{q}^-_{TT}$ and the α-node in $\mathfrak{s}$. Therefore, $h(\pi_\mathfrak{g}) = \varrho_\mathfrak{g}$ and the $I_\mathfrak{g}$-block of $\boldsymbol{q}^-_{TT}$ must be mapped by h to the $M_\mathfrak{g}$-block of $\mathfrak{s}$, forcing the input to interact with the formula.

Given a gadget $\mathfrak{g}$ and a homomorphism $h\colon \boldsymbol{q}^-_{TT} \to C$ mapping $\boldsymbol{q}^-_{TT}$ into a non-leaf segment $\mathfrak{s}$ of some cactus C, we say that $\mathfrak{g}$ *is triggered by h at* $\mathfrak{s}$ if $h(\iota_\mathfrak{g}) = \alpha$, for the $\iota_\mathfrak{g}$-node in $\boldsymbol{q}^-_{TT}$ and the α-node in $\mathfrak{s}$. We say that $\mathfrak{g}$ *is triggered at* $\mathfrak{s}$ if there is a homomorphism h triggering $\mathfrak{g}$ at $\mathfrak{s}$. Observe that if $\mathfrak{s}$ is of the form $\boldsymbol{q}^-_Z$, for some $Z \in \{AT, TA, AA\}$, and $\mathfrak{g}$ is triggered at $\mathfrak{s}$, then $\mathfrak{g}$ is either of type AA or of type Z.

Each gadget $\mathfrak{g}$ in $\boldsymbol{q}$ 'implements' some Boolean formula $\varphi_\mathfrak{g}(\boldsymbol{y})$ checking some property of desired trees at node $\mathfrak{s}$ in the skeleton 01-tree C^s of the cactus C. So the input values for the variables in $\boldsymbol{y}$ are 'collected' from an environment of $\mathfrak{s}$ in C^s. This collection process is regulated by the input-types of each $\varphi_\mathfrak{g}$. We have the following gadgets in $\boldsymbol{q}$, each implementing some formula described in Sec. 3.4:

- **(g1)** a type AA gadget implementing Good;
- **(g2)** for every k with $4 \le k \le 4\boldsymbol{d} + 11$ and every $Z \in \{AT, TA\}$, a type Z gadget implementing MustBranch^k;
- **(g3)** for every k with $4 \le k \le 4\boldsymbol{d} + 11$ and every $* \in \{0, 1\}$, a type AA-gadget implementing NoBranch$^k_*$;
- **(g4)** for every k with $4 \le k \le 4\boldsymbol{d} + 11$, a type AA-gadget implementing NoBranch^k;
- **(g5)** a type AA gadget implementing Step;
- **(g6)** a type AA gadget implementing Init;
- **(g7)** a type AA gadget implementing Reject.

Now suppose $\mathfrak{g}_1, \dots, \mathfrak{g}_m$ are all of the gadgets in $\boldsymbol{q}$. We want to ensure that when a gadget $\mathfrak{g}_j$ is triggered by some h, then the other gadgets are not triggered (that is, the $\iota_{\mathfrak{g}_i}$-node for every $i \neq j$ can be mapped by h to itself). So, in addition to the above, for every j, we connect the $\iota_{\mathfrak{g}_j}$-node via an $U_{\mathfrak{g}_j}$-labelled node to the $\tau_{\mathfrak{g}_i}$-node, for all $i \neq j$. We also want to ensure that when $\mathfrak{g}_j$ is triggered by some h, then the $M_{\mathfrak{g}_i}$-block can be h-mapped to the $M'_{\mathfrak{g}_i}$-block for every i (not just for j). So we not only $R_{\mathfrak{g}_j}$-connect $\varrho'_{\mathfrak{g}_j}$ with $\tau_{\mathfrak{g}_j}$, but also add $R_{\mathfrak{g}_j}$-arrows connecting $\varrho'_{\mathfrak{g}_j}$ with all of the $\tau_{\mathfrak{g}_i}$:



The proof of the following claim is provided in Appendix B:

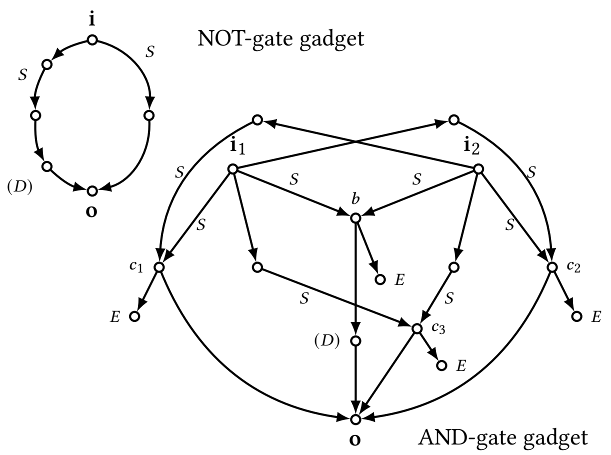
CLAIM 4.2. *A gadget $\mathfrak{g}$ in $\boldsymbol{q}$ is triggered at $\mathfrak{s}$ iff there is $\boldsymbol{b}^{\mathfrak{s}}_{\mathfrak{g}}$ such that $\boldsymbol{b}^{\mathfrak{s}}_{\mathfrak{g}}$ is gathered from 'around' $\mathfrak{s}$ in C^s according to the input-types for $\varphi_{\mathfrak{g}}$ and $\varphi_{\mathfrak{g}}[\boldsymbol{b}^{\mathfrak{s}}_{\mathfrak{g}}] = 1$.*

Claim 4.2 will be used in Sec. 3.5.4 to show that every $C \in \mathfrak{K}_{\boldsymbol{q}}$ satisfies properties **(leaf)** and **(branch)** given in Sec. 3.3.3.

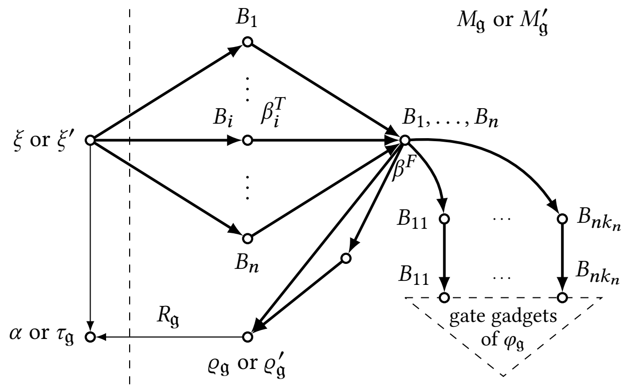
3.5.2 Main blocks in gadgets. Here we give a uniform description of the main block $M_{\mathfrak{g}}$ of each gadget $\mathfrak{g}$ in $\boldsymbol{q}$. Apart from the label W, which is uniform through the gadgets, for each particular $\mathfrak{g}$, there are a few additional labels on some nodes in $M_{\mathfrak{g}}$, $M'_{\mathfrak{g}}$ and $I_{\mathfrak{g}}$. These are always specific to $\mathfrak{g}$, but we omit indicating this to simplify notation.

A Boolean formula $\varphi_{\mathfrak{g}}(\boldsymbol{y})$ is regarded as a ditree whose vertices are called *gates*. Leaf gates are labelled by the variables from the list $\boldsymbol{y} = (y_1, \dots, y_n)$, with each y_i labelling k_i-many leaves of $\varphi_{\mathfrak{g}}(\boldsymbol{y})$. Each non-leaf g is either an AND-gate (having 2 children) or a NOT-gate (having 1 child), with the outgoing edge(s) leading to the *input(s) of g*. Given an assignment $\boldsymbol{b}$ of 0 or 1 to the *input-variables* y_i, we compute the value of each gate in $\varphi_{\mathfrak{g}}$ under $\boldsymbol{b}$ as usual in Boolean logic.

We encode the gate-structure of $\varphi_{\mathfrak{g}}(\boldsymbol{y})$ by the $M_{\mathfrak{g}}$-block (and also by its copy $M'_{\mathfrak{g}}$) as follows. With each non-leaf gate g in $\varphi_{\mathfrak{g}}$ we associate a fresh copy of its gadget shown below (where D in brackets means that D is only present when the gate in question is the root gate of $\varphi_{\mathfrak{g}}$):

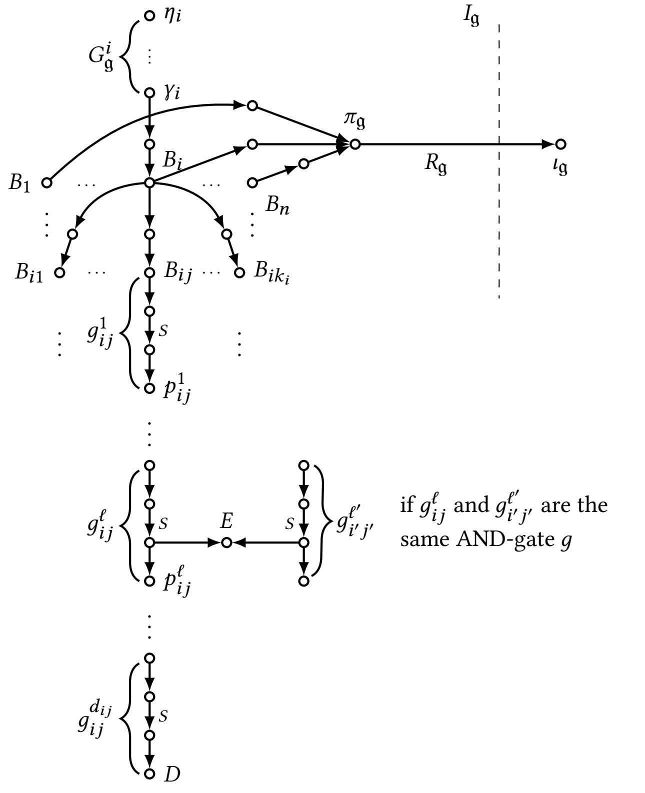


Each branch of $\varphi_{\mathfrak{g}}$ is characterised by a pair (i, j) such that the leaf node of the branch is labelled by the jth copy y^j_i of the variable y_i, for some i, j with $1 \le i \le n$ and $1 \le j \le k_i$. For each pair (i, j), we introduce a label B_{ij}. Suppose that g_1 and g_2 are the inputs of an AND-gate g. Then, for each $m = 1, 2$, if g_m is a non-leaf gate, then we merge node $\mathbf{o}$ of the g_m-gadget with node $\mathbf{i}_m$ of the g-gadget; and if g_m is labelled by y^j_i, we merge node $\mathbf{i}_m$ of the g-gadget with the lower B_{ij} -node in $M_{\mathfrak{g}}$. We proceed similarly with NOT-gates. The picture below shows how $M_{\mathfrak{g}}$ (and its copy $M'_{\mathfrak{g}}$) looks like (where, apart from the B_{ij}, we also label some nodes with B_i, for $1 \le i \le n$):



B_1
$M_{\mathfrak{g}}$ or $M'_{\mathfrak{g}}$
B_i
β^T_i
$B_1, \dots, B_n$
ξ or ξ'
β^F
B_{11}
B_{nk_n}
B_n
B_{11}
B_{nk_n}
$R_{\mathfrak{g}}$
gate gadgets of $\varphi_{\mathfrak{g}}$
α or $\tau_{\mathfrak{g}}$
$\varrho_{\mathfrak{g}}$ or $\varrho'_{\mathfrak{g}}$

3.5.3 Input blocks in gadgets. Given a Boolean formula $\varphi_{\mathfrak{g}}(\boldsymbol{y})$ with $\boldsymbol{y} = (y_1, \dots, y_n)$, its input block $I_{\mathfrak{g}}$ consists of a uniformly describable part (depending on $\varphi_{\mathfrak{g}}$ and n) and a *gathering block* $G^i_{\mathfrak{g}}$, for each i with $1 \le i \le n$ (depending on the input-types of $\varphi_{\mathfrak{g}}(\boldsymbol{y})$). For each branch of $\varphi_{\mathfrak{g}}$ characterised by (i, j), let $g^1_{ij}, \dots, g^{d_{ij}}_{ij}$ be the sequence of non-leaf gates from leaf to root on the branch with leaf y^j_i. The structure of the input block $I_{\mathfrak{g}}$ is shown below:

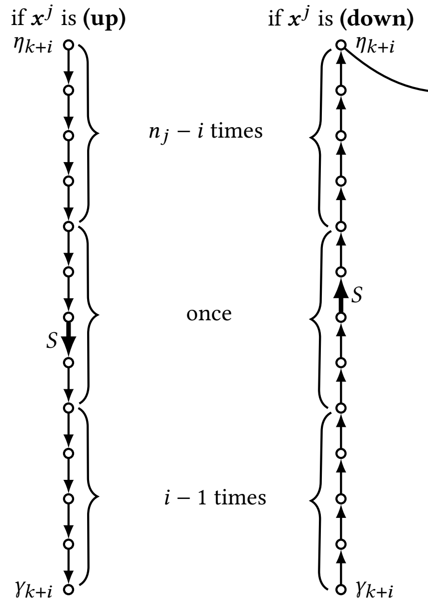


Finally, we describe the gathering blocks $G^i_\mathfrak{g}$ in $I_\mathfrak{g}$. The Boolean formula in $\mathfrak{g}$ takes the form $\varphi_\mathfrak{g}(\boldsymbol{y}) = \varphi_\mathfrak{g}(\boldsymbol{x}^1, \dots, \boldsymbol{x}^m)$ where each tuple $\boldsymbol{x}^j = (x^j_1, \dots, x^j_{n_j})$ of variables can be of two input-types:

(up) either $\boldsymbol{x}^j$ is gathered from the (unique) n_j*-long uppath* (the reverse of the suffix of the path ending at $\mathfrak{s}$ in C^s);

(down) or $\boldsymbol{x}^j$ is gathered from an n_j*-long downpath* (the prefix of a path starting at $\mathfrak{s}$ in C^s).

So suppose $y_{k+1}, \dots, y_{k+n_j}$ are among the variables of $\varphi_\mathfrak{g}$ such that $(y_{k+1}, \dots, y_{k+n_j}) = \boldsymbol{x}^j$ for some j. Then, for each i with $1 \le i \le n_j$, $G^{k+i}_\mathfrak{g}$ is shown below:



In case $\boldsymbol{x}^j$ is like in **(down)**, the W-node (of the base block) is a common successor of the η_{k+i}-nodes, for every $i = 1, \dots, n_j$, which ensures that the input bits for $y_{k+1}, \dots, y_{k+n_j}$ are all gathered from the same n_j-long downpath; see Appendix B for an example.

3.5.4 Proving that $\boldsymbol{q}$ *satisfies* **(leaf)** *and* **(branch)**. Suppose C is a cactus in $\mathfrak{K}_{\boldsymbol{q}}$ and $\mathfrak{s}$ is a non-leaf segment in the skeleton C^s of C. Then $\mathfrak{s}$ is of the form $\boldsymbol{q}^-_{Z_\mathfrak{s}}$ for some $Z_\mathfrak{s} \in \{AT, TA, AA\}$.

(leaf) ($\Rightarrow$) Suppose $h\colon \boldsymbol{q}^-_{TT} \to C$ is a homomorphism mapping $\boldsymbol{q}^-_{TT}$ into $\mathfrak{s}$. Then there is a gadget $\mathfrak{g}$ that is triggered by h at $\mathfrak{s}$ (that is, the $h(\iota_\mathfrak{g}) = \alpha$ for the $\iota_\mathfrak{g}$-node in $\boldsymbol{q}^-_{TT}$ and the α-node in $\mathfrak{s}$). By Claim 4.2, there is $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ such that $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ is gathered from 'around' $\mathfrak{s}$ in C^s according to the input-types for $\varphi_\mathfrak{g}$ and $\varphi_\mathfrak{g}[\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}] = 1$. Now we have a case distinction, depending on the gadget $\mathfrak{g}$, as listed in Sec. 3.5.1. Each gadget implements a formula whose input-types and behaviour are described in Sec. 3.4:

(g1) $\mathfrak{g}$ is the type AA gadget implementing Good (cf. Sec. 3.4.1). By the input-types of Good, $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ is the $4\boldsymbol{d} + 11$-long uppath, and it does not contain the reverse of a $001*$-pattern. Thus, $\mathfrak{s}$ is not good in C^s.

(g2) There exist some k with $4 \le k \le 4\boldsymbol{d} + 11$ and $Z \in \{AT, TA\}$ such that $\mathfrak{g}$ is the type Z gadget implementing MustBranch^k (cf. Sec. 3.4.2). By the input-types of MustBranch^k, $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ is the k-long uppath, and it is the reverse of a sequence of the form $001*(111*)^\ell w$, where either w is empty and $\ell = 0$, or $w = 001$, or $w = 111$ and $\ell < \boldsymbol{d} - 1$. On the other hand, $Z_\mathfrak{s} = Z$ must hold, and so C^s is not branching at $\mathfrak{s}$. As branching at $\mathfrak{s}$ is required in condition **(pb1)** of being properly branching, it follows that $\mathfrak{s}$ is not properly branching, and so it is incorrect in C^s.

(g3) There exist some k with $4 \le k \le 4\boldsymbol{d} + 11$ and $* \in \{0, 1\}$ such that $\mathfrak{g}$ is the type AA gadget implementing NoBranch$^k_*$ (cf. Sec. 3.4.2). Suppose, say, that $* = 0$ (the case when $* = 1$ is similar). By the input-types of NoBranch$^k_*$, $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g} = (\boldsymbol{e}^\mathfrak{s}, b^\mathfrak{s})$, where $\boldsymbol{e}^\mathfrak{s}$ is the k-long uppath and $b^\mathfrak{s}$ is a 1-long downpath. Also, $b^\mathfrak{s} = 0$ and $\boldsymbol{e}^\mathfrak{s}$ is the reverse of a sequence of the form $001*(111*)^\ell w$, where either w is empty and $0 < \ell < \boldsymbol{d}$, or $w = 1$, or $w = 11$, or $w = 00$. As having a 0-child is forbidden in condition **(pb2)** of being properly branching, it follows that $\mathfrak{s}$ is not properly branching, and so it is incorrect in C^s.

(g4) There exists some k with $4 \le k \le 4\boldsymbol{d} + 11$ such that $\mathfrak{g}$ is the type AA gadget implementing NoBranch^k (cf. Sec. 3.4.2). By the input-types of NoBranch^k, $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g} = (\boldsymbol{e}^\mathfrak{s}, b^\mathfrak{s}_1, b^\mathfrak{s}_2)$, where $\boldsymbol{e}^\mathfrak{s}$ is the k-long uppath and each of $b^\mathfrak{s}_1$ and $b^\mathfrak{s}_2$ is a 1-long downpath. Also, $b^\mathfrak{s}_1 \ne b^\mathfrak{s}_2$, and $\boldsymbol{e}^\mathfrak{s}$ is the reverse of a sequence of the form $001*(111*)^\ell w$, where $w = 111$ and $\ell = \boldsymbol{d} - 1$. As having two children is forbidden in condition **(pb4)** of being properly branching, it follows that $\mathfrak{s}$ is not properly branching, and so $\mathfrak{s}$ is incorrect in C^s.

(g5) $\mathfrak{g}$ is the type AA gadget implementing Step. By the input-types of Step, $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ should have gathered data about some $\vee$-configuration c and its two 'subsequent' $\vee$-configurations c_0, c_1. As explained in Sec. 3.4.3, (c, c_0, c_1) is inconsistent with δ, and so $\mathfrak{s}$ is not properly computing in C^s. Thus, it is incorrect in C^s.

(g6) $\mathfrak{g}$ is the type AA gadget implementing Init. By the input-types of Init, $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ should have gathered data about the 8-long uppath and some $\vee$-configuration c. As explained in Sec. 3.4.4, the part of $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ gathered from the 8-long uppath is the reverse of some pattern $111*001*$, but $c \ne c_{init(\boldsymbol{w})}$. Thus, $\mathfrak{s}$ is not properly initialising in C^s, and so it is incorrect in C^s.

(g7) $\mathfrak{g}$ is the type AA gadget implementing Reject. As explained in Sec. 3.4.5, by the input-types of Reject, $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ should have gathered data about some state q, and $q = q_{reject}$ must hold. Therefore, $\mathfrak{s}$ represents a q_{reject}-configuration in C^s, as required.

(leaf) ($\Leftarrow$) Again, we have cases **(g1)**–**(g7)**. In each case, we have a formula $\varphi_\mathfrak{g}$ for which some input $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ can be gathered from 'around' $\mathfrak{s}$ in C^s according to its input-types and for which $\varphi_\mathfrak{g}[\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}] = 1$. So by Claim 4.2, there is a homomorphism $h\colon \boldsymbol{q}^-_{TT} \to C$ mapping $\boldsymbol{q}^-_{TT}$ into $\mathfrak{s}$ and triggering $\mathfrak{g}$ at $\mathfrak{s}$.

(branch) Suppose there is a homomorphism $h\colon \boldsymbol{q}^-_{TT} \to C$ mapping $\boldsymbol{q}^-_{TT}$ into some non-leaf segment $\mathfrak{s}$. Then some gadget $\mathfrak{g}$ is triggered by h at $\mathfrak{s}$. By our assumption on $\mathfrak{s}$ and by the proof of the ($\Rightarrow$) direction of **(leaf)** above, it follows that $\mathfrak{g}$ can only be the Z type gadget implementing MustBranch^k, where $Z \in \{AT, TA\}$ is such that $\mathfrak{s} = \boldsymbol{q}^-_Z$. An inspection of Fig. 2 shows that $h(t_0) = t_0$ and $h(t_1) \ne t_0$ whenever $Z = TA$, and $h(t_1) = t_1$ and $h(t_0) \ne t_1$ whenever $Z = AT$. Therefore, **(branch)** holds.

This completes the proof that $\boldsymbol{q}$ satisfies **(leaf)** and **(branch)**.

3.6 OMQs with Schema.org and DL-Lite$_{bool}$

Schema.org, founded by Google, Microsoft, Yahoo and Yandex and developed by an open community process, comprises a set of rules

$P(\boldsymbol{x}) \leftarrow Q(\boldsymbol{x})$, for unary or binary predicates P and Q, together with domain and range constraints such as

$$T(x) \lor F(x) \leftarrow S(x,y) \tag{8}$$

$$T(y) \lor F(y) \leftarrow R(x,y) \tag{9}$$

For example, according to the Schema.org ontology, the range of the binary relation $\mathsf{musicBy}(x,y)$ is covered by the union of $\mathsf{MusicGroup}(y)$ and $\mathsf{Person}(y)$. In the syntax of description logic *DL-Lite$_{bool}$* [5], rules (8) and (9) are written as

$$\exists S \sqsubseteq T \sqcup F \qquad \text{and} \qquad \exists R^- \sqsubseteq T \sqcup F$$

Given any d-sirup $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$, denote by $\Delta'_{\boldsymbol{q}}$ the 'Schema.org ontology' obtained by replacing (1) in $\Delta_{\boldsymbol{q}}$ with rule (9), for a fresh R.

Proposition 5. *A d-sirup $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ is FO-rewritable iff $(\Delta'_{\boldsymbol{q}}, \boldsymbol{G})$ is FO-rewritable.*

Proof. $(\Rightarrow)$ Suppose Φ is a UCQ-rewriting of $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ and Φ' the result of replacing every $A(y)$ in Φ with $\exists x\, R(x,y)$. We claim that Φ' is an FO-rewriting of $(\Delta'_{\boldsymbol{q}}, \boldsymbol{G})$. Indeed, suppose $\mathcal{D}'$ is any data instance for $(\Delta'_{\boldsymbol{q}}, \boldsymbol{G})$. Without loss of generality we may assume that it does not contain atoms $A(a)$. Let $\mathcal{D}$ be the result of adding $A(b)$ to $\mathcal{D}'$ whenever $R(a,b) \in \mathcal{D}'$. Then $\Delta_{\boldsymbol{q}}, \mathcal{D} \models \boldsymbol{G}$ iff $\Delta'_{\boldsymbol{q}}, \mathcal{D}' \models \boldsymbol{G}$, and also $\mathcal{D} \models \Phi$ iff $\mathcal{D}' \models \Phi'$, from which $\Delta'_{\boldsymbol{q}}, \mathcal{D}' \models \boldsymbol{G}$ iff $\mathcal{D}' \models \Phi'$.

$(\Leftarrow)$ Suppose Φ' is a UCQ-rewriting of $(\Delta'_{\boldsymbol{q}}, \boldsymbol{G})$ and Φ is the result of replacing every $R(x,y)$ in Φ' with $A(y)$. Let $\mathcal{D}$ be a data instance for $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$. Without loss of generality we may assume that it does not contain atoms of the form $R(a,b)$. Let $\mathcal{D}'$ be the result of adding $R(a,b)$, for a fresh a, to $\mathcal{D}$ whenever $A(b) \in \mathcal{D}$. Then $\Delta_{\boldsymbol{q}}, \mathcal{D} \models \boldsymbol{G}$ iff $\Delta'_{\boldsymbol{q}}, \mathcal{D}' \models \boldsymbol{G}$, and also $\mathcal{D} \models \Phi$ iff $\mathcal{D}' \models \Phi'$, from which $\Delta'_{\boldsymbol{q}}, \mathcal{D}' \models \boldsymbol{G}$ iff $\mathcal{D}' \models \Phi'$. ❑

As a consequence of Theorem 3 and Proposition 5, we obtain the following theorem, which is an improvement on [25, Theorem 11] showing PSpace-hardness of deciding FO-rewritability of UCQs mediated by Schema.org ontologies.

Theorem 6. *Deciding FO-rewritability of CQs mediated by a Schema.org or DL-Lite$_{bool}$ ontology is 2ExpTime-hard.*

4 MONADIC D-SIRUPS WITH A DITREE CQ

The high lower bound obtained in the previous section can be regarded as a formal confirmation of the empirical fact that finding transparent syntactic, let alone practical criteria of FO-rewritability for sufficiently general classes of monadic (d-)sirups is a notoriously difficult problem. The only positive results in this direction we know of are the syntactic NC/P dichotomy of binary *chain* sirups [4] (see also [3]) and the complete AC^0/NL/P/coNP tetrachotomy of monadic *path* d-sirups without twins [22].

The CQs used in the proof of Theorem 3 were directed acyclic graphs with one solitary F-node, two solitary T-nodes, and multiple FT-twins. The question we try to answer in this section is whether the restriction of the set of CQs admitted in d-sirups to those that are *rooted directed trees* as graphs (*ditree CQs*, for short) makes deciding FO-rewritability of d-sirups $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ easier, having in mind a complete syntactic classification of such d-sirups as an ultimate (possibly unrealistic) aim. Note for starters that, by Example 1, the data complexity of evaluating d-sirups with a ditree CQ ranges from AC^0 to L, NL, P, and coNP.

A CQ $\boldsymbol{q}$ is *minimal* if there is no $\boldsymbol{q} \to \boldsymbol{q}'$ homomorphism, for any proper subCQ $\boldsymbol{q}'$ of $\boldsymbol{q}$. As well-known, checking minimality of tree-shaped CQs can be done in polynomial time; see, e.g., [16]. We denote the root node of $\boldsymbol{q}$ by $\mathfrak{r}$ and write $x \preceq_{\boldsymbol{q}} y$ to say that there is a (directed) path from x to y in $\boldsymbol{q}$, and $x \prec_{\boldsymbol{q}} y$ if $x \preceq_{\boldsymbol{q}} y$ and $x \neq y$. A pair (x,y) is *$\prec_{\boldsymbol{q}}$-comparable* if either $x \preceq_{\boldsymbol{q}} y$ or $y \preceq_{\boldsymbol{q}} x$, otherwise (x,y) is *$\prec_{\boldsymbol{q}}$-incomparable*. If $x \preceq_{\boldsymbol{q}} y$ then $\delta_{\boldsymbol{q}}(x,y)$ is the number of edges between x and y. The *distance* between any x and y is $\partial_{\boldsymbol{q}}(x,y) = \delta_{\boldsymbol{q}}\big(\inf_{\boldsymbol{q}}(x,y), x\big) + \delta_{\boldsymbol{q}}\big(\inf_{\boldsymbol{q}}(x,y), y\big)$, where $\inf_{\boldsymbol{q}}(x,y)$ is the unique node such that $\inf_{\boldsymbol{q}}(x,y) \preceq_{\boldsymbol{q}} x$, $\inf_{\boldsymbol{q}}(x,y) \preceq_{\boldsymbol{q}} y$ and $z \preceq_{\boldsymbol{q}} \inf_{\boldsymbol{q}}(x,y)$ whenever $z \preceq_{\boldsymbol{q}} x$ and $z \preceq_{\boldsymbol{q}} y$. The subscript $\boldsymbol{q}$ in $\preceq_{\boldsymbol{q}}$, $\prec_{\boldsymbol{q}}$, $\delta_{\boldsymbol{q}}$, $\inf_{\boldsymbol{q}}$ and $\partial_{\boldsymbol{q}}$ will be dropped if understood.

If t is a solitary T-node and f is solitary F-node, we call (t,f) a *solitary pair*. We say that a solitary pair (t,f) is of *minimal distance*, if $\partial(t,f) \geq \partial(\mathsf{t},\mathsf{f})$ for any solitary pair (t,f). A $\prec$-incomparable solitary pair (t,f) is called *symmetric* if the CQ obtained by removing the labels F, T from f, t and cutting the branches below them is symmetric with respect to $\mathfrak{r}$ (see $\boldsymbol{q}_4$ in Example 1). A ditree $\boldsymbol{q}$ is *quasi-symmetric* if it has no $\prec$-comparable solitary pairs, and every solitary pair (t,f) of minimal distance is symmetric.

As follows from [22] (where F and T are interchangeable),

(a) if $\boldsymbol{q}$ has no solitary F, then $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ is FO-rewritable;
(b) if $\boldsymbol{q}$ has one solitary F, then $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ is datalog-rewritable (and so in P for data complexity);
(c) if $\boldsymbol{q}$ has one solitary F and one solitary T, then $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ is linear-datalog-rewritable (and so in NL);
(d) if $\boldsymbol{q}$ has one solitary F, one solitary T and is quasi-symmetric, then $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ is symmetric-linear-datalog-rewritable (and so in L).

The following result identifies a large and tractable class of d-sirups with a ditree CQ whose evaluation is NL-hard:

Theorem 7. *Suppose $\boldsymbol{q}$ is a minimal ditree CQ with at least one solitary F, at least one solitary T and such that either*

(i) *there is a $\prec$-comparable solitary pair (t,f) or*
(ii) *$\boldsymbol{q}$ is not quasi-symmetric and has no FT-twins.*

Then evaluating the d-sirup $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ is NL-hard.

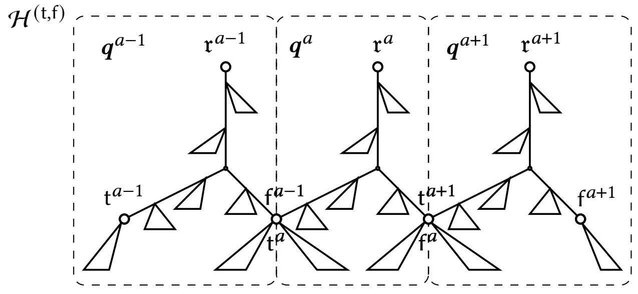
Proof. The proof is by reduction of the NL-complete reachability problem for dags. Given a dag $G = (V,E)$ with nodes $\mathfrak{s}, \mathfrak{t} \in V$, we construct a data instance $\mathcal{D}_G$ as follows. We pick a solitary pair (t,f) such that, in case (i), (t,f) is $\prec$-comparable and there is no solitary T- or F-node between t and f; and, in case (ii), (t,f) is of minimal distance, $\prec$-incomparable, and not symmetric. Then, in both cases, we replace each $e = (\mathfrak{u},\mathfrak{v}) \in E$ by a fresh copy $\boldsymbol{q}^e$ of $\boldsymbol{q}$ in which t^e is renamed to $\mathfrak{u}$ with $T(\mathfrak{u})$ replaced by $A(\mathfrak{u})$, and f^e is renamed to $\mathfrak{v}$ with $F(\mathfrak{v})$ replaced by $A(\mathfrak{v})$. The dag $\mathcal{D}_G$ comprises the $\boldsymbol{q}^e$, for $e \in E$, as well as $T(\mathfrak{s})$ and $F(\mathfrak{t})$. We show that $\mathfrak{s} \to_G \mathfrak{t}$ iff the answer to $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ over $\mathcal{D}_G$ is 'yes'.

$(\Rightarrow)$ If $\mathfrak{s} = \mathfrak{v}_0, \dots, \mathfrak{v}_n = \mathfrak{t}$ is a path in G with $e_i = (\mathfrak{v}_i, \mathfrak{v}_{i+1}) \in E$, for $i < n$, then for any model $\mathcal{I}$ of $\Delta_{\boldsymbol{q}}$ and $\mathcal{D}_G$, there is some $i < n$ such that $\mathcal{I} \models T(\mathfrak{v}_i)$ and $\mathcal{I} \models F(\mathfrak{v}_{i+1})$, and so the identity map from $\boldsymbol{q}$ to its copy $\boldsymbol{q}^{e_i}$ is a $\boldsymbol{q} \to \mathcal{I}$ homomorphism.

$(\Leftarrow)$ If $\mathfrak{s} \not\to_G \mathfrak{t}$, we define a model $\mathcal{I}$ of $\Delta_{\boldsymbol{q}}$ and $\mathcal{D}_G$ by labelling with T the A-nodes in $\mathcal{D}_G$ that (as nodes of G) are reachable from $\mathfrak{s}$

(via a directed path in G) and with F the remaining ones. We claim that if one of (i) or (ii) holds, then there is no homomorphism from $\boldsymbol{q}$ to $\mathcal{I}$, and so the answer to $(\Delta_{\boldsymbol{q}}, G)$ over $\mathcal{D}_G$ is 'no'. Indeed, take any map h from $\boldsymbol{q}$ to $\mathcal{I}$, and consider the substructure $\mathcal{H}^{(\mathsf{t},\mathsf{f})}$ of $\mathcal{D}_G$ comprising those copies $\boldsymbol{q}^{e_1}, \dots, \boldsymbol{q}^{e_n}$ of $\boldsymbol{q}$ that have a non-empty intersection with $h(\boldsymbol{q})$. To simplify notation, we set $\boldsymbol{q}^j = \boldsymbol{q}^{e_j}$. Then $\mathcal{I}$ can be regarded as a model of $\mathcal{H}^{(\mathsf{t},\mathsf{f})}$. The (quite arduous case-distinction) proof in Appendix C shows that h cannot be a homomorphism from $\boldsymbol{q}$ to $\mathcal{I}$.

Here, we only sketch the proof for case (ii) when $\boldsymbol{q}$ is not quasi-symmetric, and we may also assume that $\boldsymbol{q}$ has no $\prec$-comparable solitary pairs. Suppose $h\colon \boldsymbol{q} \to \mathcal{I}$ is a homomorphism, and let a be such that $h(\mathfrak{r}) \in \boldsymbol{q}^a$. As (t,f) is $\prec$-incomparable, $\mathcal{H}^{(\mathsf{t},\mathsf{f})}$ consists of (at most) three copies $\boldsymbol{q}^a$, $\boldsymbol{q}^{a-1}$ and $\boldsymbol{q}^{a+1}$ of $\boldsymbol{q}$, and looks as shown in the picture below:



As (t,f) is not symmetric, $\mathcal{I}$ is such that the 'contacts' between the $\boldsymbol{q}$-copies are either both in $F^{\mathcal{I}}$ or both in $T^{\mathcal{I}}$.

The following 'structural' claim (tracking the possible locations of $h(\mathsf{f})$ and $h(\mathsf{t})$) is proved in Appendix C (it is also used in the proof of Theorem 11 below):

Claim 7.1. *Suppose (t,f) is $\prec$-incomparable and of minimal distance (though $\boldsymbol{q}$ might contain FT-twins). If $\mathsf{m} = \inf_{\boldsymbol{q}}(\mathsf{t},\mathsf{f})$ then $h(\mathsf{m})$ is in $\boldsymbol{q}^a$, and one of the following holds:*

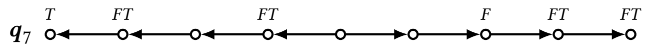
1. $\mathsf{m}^a \prec_{\boldsymbol{q}^a} h(\mathsf{m}) \prec_{\boldsymbol{q}^a} \mathsf{t}^a$, *$h(\mathsf{t})$ is in $\boldsymbol{q}^{a-1}$ with* $\mathsf{f}^{a-1} \prec_{\boldsymbol{q}^{a-1}} h(\mathsf{t})$, *and* $h(\mathsf{f}) = \mathsf{t}^a$;
2. $\mathsf{m}^a \prec_{\boldsymbol{q}^a} h(\mathsf{m}) \prec_{\boldsymbol{q}^a} \mathsf{f}^a$, *$h(\mathsf{f})$ is in $\boldsymbol{q}^{a+1}$ with* $\mathsf{t}^{a+1} \prec_{\boldsymbol{q}^{a+1}} h(\mathsf{f})$, *and* $h(\mathsf{t}) = \mathsf{f}^a$;
3. $h(\mathsf{m}) = \mathsf{m}^a$, $h(\mathsf{f}) = \mathsf{f}^a$, *and $h(\mathsf{t})$ is in $\boldsymbol{q}^a$ with* $h(\mathsf{t}) \prec_{\boldsymbol{q}^a} \mathsf{f}^a$;
4. $h(\mathsf{m}) = \mathsf{m}^a$, $h(\mathsf{t}) = \mathsf{t}^a$, *and $h(\mathsf{f})$ is in $\boldsymbol{q}^a$ with* $h(\mathsf{f}) \prec_{\boldsymbol{q}^a} \mathsf{t}^a$.

However, if $\boldsymbol{q}$ contains neither FT-twins nor $\prec$-comparable solitary pairs, none of (1)–(4) in Claim 7.1 can happen. ❑

Denote by $\Delta^+_{\boldsymbol{q}}$ the d-sirup $(\Delta_{\boldsymbol{q}}, G)$ extended by an extra rule $\bot \leftarrow T(x), F(x)$ saying that the predicates F and T are disjoint, and so $\Delta^+_{\boldsymbol{q}}$ with $\boldsymbol{q}$ containing an FT-twin is inconsistent. As shown in [22], $(\Delta_{\boldsymbol{q}}, G)$ is L-hard when $\boldsymbol{q}$ has at least one solitary F and at least one solitary T but no FT-twins. So we have:

Corollary 8. *Every d-sirup $(\Delta^+_{\boldsymbol{q}}, G)$ with a ditree $\boldsymbol{q}$ is either FO-rewritable (if $\boldsymbol{q}$ contains FT-twins), or L-hard (if $\boldsymbol{q}$ is quasi-symmetric without FT-twins), or NL-hard (otherwise).*

The non-quasi-symmetric CQs $\boldsymbol{q}$ that are outside the scope of Theorem 7 are those that have FT-twins and only contain $\prec$-incomparable solitary pairs. That Theorem 7 does not hold for such CQs is demonstrated by $\boldsymbol{q}_5$ in Example 1 (cf. Claim 7.1 (3)), $\boldsymbol{q}_6$ in Example 4 (cf. Claim 7.1 (4)), and $\boldsymbol{q}_7$, $\boldsymbol{q}_8$ below (cf. Claim 7.1 (1)), for all of which $(\Delta_{\boldsymbol{q}}, G)$ is FO-rewritable. As before, the omitted labels on the arrows are all R.



Our next result, used in tandem with Theorem 7, gives an FO/L-hardness dichotomy for d-sirups $(\Delta_{\boldsymbol{q}}, G)$ with a ditree 1-CQ $\boldsymbol{q}$ (having a single solitary F). If such a $\boldsymbol{q}$ has k-many solitary T-nodes, each of which is $\prec$-incomparable with the F-node, we call it a *Λ-CQ of span k*.

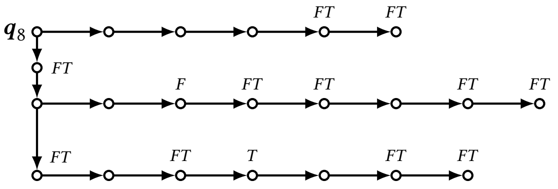
Theorem 9. (i) *For any Λ-CQ $\boldsymbol{q}$, either the d-sirup $(\Delta_{\boldsymbol{q}}, G)$ is FO-rewritable or evaluating it is L-hard.*

(ii) *For Λ-CQs of span k, deciding this FO/L-dichotomy can be done in time $p(|\boldsymbol{q}|)2^{p'(k)}$, for some polynomials p and p'. Thus, deciding FO-rewritability of d-sirups with a Λ-CQ is fixed-parameter tractable, if the Λ-CQ's span is regarded as a parameter.*

Proof. Let $\boldsymbol{q}$ be a Λ-CQ with solitary T-nodes $T(y_1), \dots, T(y_k)$. By Prop. 2, $(\Delta_{\boldsymbol{q}}, G)$ is FO-rewritable iff there exists $d < \omega$ such that any cactus (for $\boldsymbol{q}$) contains a homomorphic image of some cactus of depth $\leq d$. The *neighbourhood* of a segment $\mathfrak{s}$ in a cactus C consists of $\mathfrak{s}$ itself and those segments that, in the skeleton C^s, are the children, parent and siblings of $\mathfrak{s}$—at most $2k + 1$ segments in total. Since $\boldsymbol{q}$ is a ditree, in which the F-node is $\prec$-incomparable with any T-node, the following holds for any cactuses C, C' (for $\boldsymbol{q}$):

Claim 9.1. *Suppose $h\colon C \to C'$ is a homomorphism that maps the root of a segment $\mathfrak{s}$ in C to a node in a segment $\mathfrak{s}'$ in C'. Then the nodes in $\mathfrak{s}$ are mapped by h to nodes in the neighbourhood of $\mathfrak{s}'$.*

Example 5. Consider the Λ-CQ $\boldsymbol{q}_8$ of span 1 below. We invite the reader to verify that there is a homomorphism $h\colon C_2 \to C_i$, for $i \geq 3$ (where C_i is obtained by i-many applications of **(bud)** to $C_0 = \boldsymbol{q}_8$) such that the h-image of the leaf segment in C_2 intersects three segments in C_i. It is not hard to see that $(\Delta_{\boldsymbol{q}}, G)$ is FO-rewritable to $\exists z\, (C_0 \lor C_1 \lor C_2)$.



In any skeleton C^s, we label by $i \in \{1, \dots, k\}$ every edge that results from budding $T(y_i)$. The neighbourhood of any segment $\mathfrak{s}$ is given by the triple $t = (P, i_{\mathfrak{s}}, C)$, where $P \subseteq \{1, \dots, k\}$ comprises the labels on the edges from the parent $\mathfrak{s}'$ of $\mathfrak{s}$, $i_{\mathfrak{s}}$ is the label on $(\mathfrak{s}', \mathfrak{s})$, and $C \subseteq \{1, \dots, k\}$ are the labels on the edges to $\mathfrak{s}$'s children. If $\mathfrak{s}$ is the root of C^s, $P = \emptyset$ and we set $i_{\mathfrak{s}} = 0$; if $\mathfrak{s}$ is a leaf, $C = \emptyset$. We refer to $\mathfrak{s}$ as the *central segment of* t, and to t as the *type* of $\mathfrak{s}$; we call it a *root type* if $\mathfrak{s}$ is the root of C^s, and a *leaf type* if $\mathfrak{s}$ is a leaf. A cactus C is *acyclic* if none of the branches in C^s has two nodes of the same type.

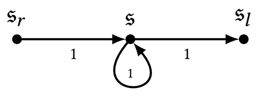
Let $\mathfrak{G}$ be the digraph whose nodes are all possible types and there is an edge (t, t') labelled by $i \in \{1, \dots, k\}$ iff some skeleton C^s has an edge $(\mathfrak{s}, \mathfrak{s}')$ labelled by i with $\mathfrak{s}$ being of type t and $\mathfrak{s}'$ of type t'. Let χ_C be the *canonical homomorphism* of C^s to $\mathfrak{G}$ (mapping the segments of C^s to their types). For a subgraph $\mathfrak{H}$ of $\mathfrak{G}$ denote by $\bar{\mathfrak{H}}$

the result of replacing the types in $\mathfrak{H}$ with their central segments and glueing them at A-nodes as indicated by the types and edges in $\mathfrak{H}$, mimicking **(bud)**. We call this operation the $\bar{\cdot}$-*closure* of $\mathfrak{H}$.

A node v of type (P, i, C) in a subgraph $\mathfrak{H}$ of $\mathfrak{G}$ is *realisable in* $\mathfrak{H}$ if v has exactly one outgoing edge labelled by j in $\mathfrak{H}$, for each $j \in C$. We call $\mathfrak{H}$ *realisable* if it has exactly one *source* (a node without incoming edges) of root type and all nodes in $\mathfrak{H}$ are realisable.

A *periodic structure* is a triple $\mathfrak{P} = (\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{E})$ satisfying the following conditions. First, we take some realisable subgraph $\mathfrak{H}$ of $\mathfrak{G}$ and define the *pre-periodic* part $\boldsymbol{B}$ to be the subgraph of $\mathfrak{H}$ induced by those nodes v in $\mathfrak{H}$, for which there are no arbitrarily long paths from the source to v. The *periodic* part $\boldsymbol{P}$ is induced by the remaining nodes in $\mathfrak{H}$. Finally, the *post-periodic part* $\boldsymbol{E}$ comprises a set R of nodes in $\boldsymbol{P}$, intersecting any directed cycle in $\boldsymbol{P}$, and a family of *acyclic* subgraphs $\mathfrak{H}_v$, $v \in R$, of $\mathfrak{G}$ with unique source v and such that all of the $\mathfrak{H}_v$'s nodes are realisable in $\mathfrak{H}_v$, for any $v \in R$. Denote by $\bar{\mathfrak{P}} = (\bar{\boldsymbol{B}}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{E}})$ the triple obtained by taking the $\bar{\cdot}$-closure of the components $\boldsymbol{B}$, $\boldsymbol{P}$ and $\boldsymbol{E}$ in $\mathfrak{P}$.

To illustrate, for $k = 1$, in the only periodic structure with non-empty $\boldsymbol{P}$ shown below, $\boldsymbol{B}$ comprises the root segment $\mathfrak{s}_r$, $\boldsymbol{P}$ the segment $\mathfrak{s}$ (with two A-nodes), and $\boldsymbol{E}$ the leaf segment $\mathfrak{s}_l$. There are also three 'degenerate' periodic structures with empty $\boldsymbol{P}$ and $\boldsymbol{E}$.



The *acyclic version* of a rooted digraph G is constructed as follows. We consider each path π starting in the root and ending at the first repeating node v on π with the last edge (u, v). For all such π, v and (u, v), we add to G a fresh node v' and replace (u, v) by (u, v').

The proof of the following criterion can be found in Appendix D:

Claim 9.2. *The d-sirup $(\Delta_{\boldsymbol{q}}, G)$ is FO-rewritable iff, for any periodic structure $\mathfrak{P} = (\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{E})$ with $\boldsymbol{P} \neq \emptyset$, one of the following holds:*

- **(h1)** *there is a homomorphism from some cactus to the $\bar{\cdot}$-closure of the acyclic version of $\boldsymbol{B} \cup \boldsymbol{P}$;*
- **(h2)** *there is a homomorphism from the root segment of some cactus to $\bar{\boldsymbol{P}}$;*
- **(h3)** *there is a homomorphism from the root segment of one of the $\bar{\mathfrak{H}}_v$ to $\bar{\boldsymbol{E}}$.*

On the other hand, we have the following claim, which is proved in Appendix E and establishes an FO/L-hardness dichotomy of d-sirups with a Λ-CQ:

Claim 9.3. *If none of conditions* **(h1)**–**(h3)** *holds, then evaluating $(\Delta_{\boldsymbol{q}}, G)$ is L-hard.*

In Appendix F, we show that checking the criterion of Claim 9.2 for Λ-CQs of span k can be done in time $p(|\boldsymbol{q}|)2^{p'(k)}$, for some polynomials p and p'. ❑

As a consequence of Theorems 7 and 9, we obtain the dichotomy:

Corollary 10. *Any d-sirup $(\Delta_{\boldsymbol{q}}, G)$ with a ditree 1-CQ $\boldsymbol{q}$ is either FO-rewritable or L-hard. Deciding this dichotomy, parameterised by the number of solitary T-nodes in CQs, is fixed-parameter tractable.*

This result is in sharp contrast to 2ExpTime-completeness of deciding FO-rewritability of d-sirups $(\Delta_{\boldsymbol{q}}, G)$ with a dag 1-CQ $\boldsymbol{q}$ having two solitary T-nodes. We hope that, using the techniques of [22, 31], this dichotomy can be extended to a complete FO/L/NL/P-tetrachotomy of all d-sirups with a ditree 1-CQ. As a first step, we obtain the following trichotomy:

Theorem 11. *For any a ditree CQ $\boldsymbol{q}$ with one solitary F and one solitary T, $(\Delta_{\boldsymbol{q}}, G)$ is either FO-rewritable, or L-complete, or NL-complete. Deciding this trichotomy can be done in polynomial time.*

Proof. We use the results of [22] listed as items (c) and (d) on page 12. Let t and f be the solitary T- and F-nodes in $\boldsymbol{q}$. If (t, f) is $\prec$-comparable then $(\Delta_{\boldsymbol{q}}, G)$ is NL-complete by (c) and Theorem 7 (i). If $\boldsymbol{q}$ is quasi-symmetric, then $(\Delta_{\boldsymbol{q}}, G)$ is in L by (d); L-hardness is shown in Appendix G by a reduction of graph reachability (using a construction that is similar to the one in the proof of Theorem 7).

Otherwise, we consider two models $\mathcal{I}$ over the structure $\mathcal{H}^{(\mathsf{t},\mathsf{f})}$ (defined in the proof-sketch of Theorem 7 (ii)): one has both 'contacts' in $F^{\mathcal{I}}$, the other in $T^{\mathcal{I}}$. We check whether there exists a homomorphism from $\boldsymbol{q}$ to either of these models: If neither, then $(\Delta_{\boldsymbol{q}}, G)$ is NL-hard by the the proof of Theorem 7 (ii). If at least one of them is possible, then $(\Delta_{\boldsymbol{q}}, G)$ is FO-rewritable by Prop. 2 (as one can use the $\boldsymbol{q} \to \mathcal{I}$ homomorphism to define homomorphisms from some depth ≤ 2 cactus to any larger cactus). Details can be found in Appendix G. ❑

5 CONCLUSIONS

In this paper, we settled the long-standing open problem on the complexity of deciding boundedness of monadic single rule datalog programs. Namely, we proved this problem to be 2ExpTime-complete—that is, as hard as deciding program boundedness of arbitrary monadic datalog programs [18]. The main innovation of our proof is that we look at the computations of ATMs and the expansions of sirups through the lens of Boolean circuits and show how these circuits can be 'implemented' in dag-shaped CQs to verify the correctness of computations encoded by the expansions.

We obtained this result while trying to classify a somewhat different type of basic recursive programs called monadic disjunctive sirups. The disjunctive rule $F(x) \lor T(x) \leftarrow A(x)$ can make answering a Boolean CQ it mediates in the d-sirup range between AC^0 and coNP. Deciding FO-rewritability of monadic d-sirups (as well as of Schema.org and *DL-Lite$_{bool}$* ontology-mediated queries) was shown to be between 2ExpTime and 2NExpTime, and so a complete classification of monadic d-sirups according to their data complexity can be as illusory as the classification of monadic sirups, which has been challenging the datalog community since the 1980s.

On the other hand, this paper shows that d-sirups with ditree CQs are less impenetrable, and we believe a complete classification is possible, though it could be quite tricky and laborious. This problem as well as pinpointing the exact complexity of deciding FO-rewritability of monadic d-sirups (2ExpTime vs 2NexpTime) are left for future work.

ACKNOWLEDGMENTS

This work was supported by the UK EPSRC grant EP/S032282, HSE University Basic Research Program, and Russian Science Foundation 20-11-20203 (Section 4). Thanks are due to the anonymous reviewers for their comments and constructive suggestions.

REFERENCES

[1] Serge Abiteboul. 1989. Boundedness is Undecidable for Datalog Programs with a Single Recursive Rule. *Inf. Process. Lett.* 32, 6 (1989), 281–287. https://doi.org/10.1016/0020-0190(89)90019-7

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley. http://webdam.inria.fr/Alice/

[3] Foto N. Afrati, Manolis Gergatsoulis, and Francesca Toni. 2003. Linearisability on datalog programs. *Theor. Comput. Sci.* 308, 1-3 (2003), 199–226. https://doi.org/10.1016/S0304-3975(02)00730-2

[4] Foto N. Afrati and Christos H. Papadimitriou. 1993. The Parallel Complexity of Simple Logic Programs. *J. ACM* 40, 4 (1993), 891–916. https://doi.org/10.1145/153724.153752

[5] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. 2009. The DL-Lite Family and Relations. *J. Artif. Intell. Res.* 36 (2009), 1–69. https://doi.org/10.1613/jair.2820

[6] Pablo Barceló, Gerald Berger, Carsten Lutz, and Andreas Pieris. 2018. First-Order Rewritability of Frontier-Guarded Ontology-Mediated Queries. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, Jérôme Lang (Ed.). ijcai.org, 1707–1713. https://doi.org/10.24963/ijcai.2018/236

[7] Michael Benedikt, Pierre Bourhis, Georg Gottlob, and Pierre Senellart. 2020. Monadic Datalog, Tree Validity, and Limited Access Containment. *ACM Trans. Comput. Log.* 21, 1 (2020), 6:1–6:45.

[8] Michael Benedikt, Pierre Bourhis, and Pierre Senellart. 2012. Monadic Datalog Containment. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 7392)*, Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer (Eds.). Springer, 79–91. https://doi.org/10.1007/978-3-642-31585-5_11

[9] Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. 2015. The Complexity of Boundedness for Guarded Logics. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*. IEEE Computer Society, 293–304. https://doi.org/10.1109/LICS.2015.36

[10] Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. 2016. First Order-Rewritability and Containment of Conjunctive Queries in Horn Description Logics. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, Subbarao Kambhampati (Ed.). IJCAI/AAAI Press, 965–971. http://www.ijcai.org/Abstract/16/141

[11] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. 2014. Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP. *ACM Trans. Database Syst.* 39, 4 (2014), 33:1–33:44. https://doi.org/10.1145/2661643

[12] Henrik Björklund, Wim Martens, and Thomas Schwentick. 2008. Optimizing Conjunctive Queries over Trees Using Schema Information. In *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings (Lecture Notes in Computer Science, Vol. 5162)*, Edward Ochmanski and Jerzy Tyszkiewicz (Eds.). Springer, 132–143. https://doi.org/10.1007/978-3-540-85238-4_10

[13] Henrik Björklund, Wim Martens, and Thomas Schwentick. 2018. Conjunctive query containment over trees using schema information. *Acta Informatica* 55, 1 (2018), 17–56. https://doi.org/10.1007/s00236-016-0282-1

[14] Pierre Bourhis and Carsten Lutz. 2016. Containment in Monadic Disjunctive Datalog, MMSNP, and Expressive Description Logics. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*, Chitta Baral, James P. Delgrande, and Frank Wolter (Eds.). AAAI Press, 207–216. http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12847

[15] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2007. Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. *J. Autom. Reason.* 39, 3 (2007), 385–429. https://doi.org/10.1007/s10817-007-9078-x

[16] Chandra Chekuri and Anand Rajaraman. 2000. Conjunctive query containment revisited. *Theor. Comput. Sci.* 239, 2 (2000), 211–229. https://doi.org/10.1016/S0304-3975(99)00220-0

[17] C. Civili and R. Rosati. 2012. A Broad Class of First-Order Rewritable Tuple-Generating Dependencies. In *Proc. of the 2nd Int. Datalog 2.0 Workshop (Lecture Notes in Computer Science, Vol. 7494)*. Springer, 68–80.

[18] Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. 1988. Decidable Optimization Problems for Database Logic Programs (Preliminary Report). In *STOC*. 477–490.

[19] Stavros S. Cosmadakis and Paris C. Kanellakis. 1986. Parallel Evaluation of Recursive Rule Queries. In *Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 24-26, 1986, Cambridge, Massachusetts, USA*, Avi Silberschatz (Ed.). ACM, 280–293. https://doi.org/10.1145/6012.15421

[20] Cristina Feier, Antti Kuusisto, and Carsten Lutz. 2019. Rewritability in Monadic Disjunctive Datalog, MMSNP, and Expressive Description Logics. *Logical Methods in Computer Science* 15, 2 (2019). https://doi.org/10.23638/LMCS-15(2:15)2019

[21] Haim Gaifman, Harry G. Mairson, Yehoshua Sagiv, and Moshe Y. Vardi. 1987. Undecidable Optimization Problems for Database Logic Programs. In *Proceedings of the Symposium on Logic in Computer Science (LICS '87), Ithaca, New York, USA, June 22-25, 1987*. IEEE Computer Society, 106–115.

[22] Olga Gerasimova, Stanislav Kikot, Agi Kurucz, Vladimir V. Podolskii, and Michael Zakharyaschev. 2020. A Data Complexity and Rewritability Tetrachotomy of Ontology-Mediated Queries with a Covering Axiom. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, Diego Calvanese, Esra Erdem, and Michael Thielscher (Eds.). 403–413. https://doi.org/10.24963/kr.2020/41

[23] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. 2014. Query Rewriting and Optimization for Ontological Databases. *ACM Trans. Database Syst.* 39, 3 (2014), 25:1–25:46. https://doi.org/10.1145/2638546

[24] Martin Grohe. 2007. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM* 54, 1 (2007), 1:1–1:24. https://doi.org/10.1145/1206035.1206036

[25] André Hernich, Carsten Lutz, Ana Ozaki, and Frank Wolter. 2015. Schema.org as a Description Logic. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Qiang Yang and Michael J. Wooldridge (Eds.). AAAI Press, 3048–3054. http://ijcai.org/Abstract/15/430

[26] Gerd G. Hillebrand, Paris C. Kanellakis, Harry G. Mairson, and Moshe Y. Vardi. 1991. Tools for Datalog Boundedness. In *Proceedings of the Tenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 29-31, 1991, Denver, Colorado, USA*, Daniel J. Rosenkrantz (Ed.). ACM Press, 1–12. https://doi.org/10.1145/113413.113414

[27] Gerd G. Hillebrand, Paris C. Kanellakis, Harry G. Mairson, and Moshe Y. Vardi. 1995. Undecidable Boundedness Problems for Datalog Programs. *J. Log. Program.* 25, 2 (1995), 163–190. https://doi.org/10.1016/0743-1066(95)00051-K

[28] Mark Kaminski, Yavor Nenov, and Bernardo Cuenca Grau. 2016. Datalog rewritability of Disjunctive Datalog programs and non-Horn ontologies. *Artif. Intell.* 236 (2016), 90–118. https://doi.org/10.1016/j.artint.2016.03.006

[29] Paris C. Kanellakis. 1990. Elements of Relational Database Theory. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, Jan van Leeuwen (Ed.). Elsevier and MIT Press, 1073–1156. https://doi.org/10.1016/b978-0-444-88074-1.50022-6

[30] Mélanie König, Michel Leclère, Marie-Laure Mugnier, and Michaël Thomazo. 2015. Sound, complete and minimal UCQ-rewriting for existential rules. *Semantic Web* 6, 5 (2015), 451–475. https://doi.org/10.3233/SW-140153

[31] Carsten Lutz and Leif Sabellek. 2017. Ontology-Mediated Querying with the Description Logic EL: Trichotomy and Linear Datalog Rewritability. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 1181–1187. https://doi.org/10.24963/ijcai.2017/164

[32] Carsten Lutz and Leif Sabellek. 2019. A Complete Classification of the Complexity and Rewritability of Ontology-Mediated Queries based on the Description Logic EL. *CoRR* abs/1904.12533 (2019). arXiv:1904.12533 http://arxiv.org/abs/1904.12533

[33] Jerzy Marcinkowski. 1999. Achilles, Turtle, and Undecidable Boundedness Problems for Small DATALOG Programs. *SIAM J. Comput.* 29, 1 (1999), 231–257. https://doi.org/10.1137/S0097539797322140

[34] Jeffrey F. Naughton. 1986. Data Independent Recursion in Deductive Databases. In *Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, March 24-26, 1986, Cambridge, Massachusetts, USA*, Avi Silberschatz (Ed.). ACM, 267–279. https://doi.org/10.1145/6012.15420

[35] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking Data to Ontologies. *J. Data Semant.* 10 (2008), 133–173. https://doi.org/10.1007/978-3-540-77688-8_5

[36] Ron van der Meyden. 2000. Predicate Boundedness of Linear Monadic Datalog is in PSPACE. *Int. J. Found. Comput. Sci.* 11, 4 (2000), 591–612. https://doi.org/10.1142/S0129054100000351

[37] Moshe Y. Vardi. 1988. Decidability and Undecidability Results for Boundedness of Linear Recursive Queries. In *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 21-23, 1988, Austin, Texas, USA*, Chris Edmondson-Yurkanan and Mihalis Yannakakis (Eds.). ACM, 341–351. https://doi.org/10.1145/308386.308470

[38] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. 2018. Ontology-Based Data Access: A Survey. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, Jérôme Lang (Ed.). ijcai.org, 5511–5519. https://doi.org/10.24963/ijcai.2018/777

[39] Guohui Xiao, Linfang Ding, Benjamin Cogrel, and Diego Calvanese. 2019. Virtual Knowledge Graphs: An Overview of Systems and Use Cases. *Data Intell.* 1, 3 (2019), 201–223. https://doi.org/10.1162/dint_a_00011

A PROOF OF LEMMA 4 FROM (FOC), (LEAF) AND (BRANCH)

Let $\boldsymbol{q}$ be a 1-CQ having two solitary T-nodes t_0 and t_1 such that, for every $\mathcal{C}, \mathcal{C}' \in \mathfrak{K}_{\boldsymbol{q}}$,

(foc) if $h\colon \mathcal{C} \to \mathcal{C}'$ is a homomorphism then h maps the root segment of $\mathcal{C}$ into the root segment of $\mathcal{C}'$;

(leaf) there is a homomorphism $h\colon \boldsymbol{q}_{TT}^{-} \to \mathcal{C}$ mapping $\boldsymbol{q}_{TT}^{-}$ into some non-leaf segment $\mathfrak{s}$ of $\mathcal{C}$ iff either $\mathfrak{s}$ is incorrect or $\mathfrak{s}$ represents a q_{reject}-configuration in $\mathcal{C}^s$;

(branch) if h maps $\boldsymbol{q}_{TT}^{-}$ into a non-leaf segment $\mathfrak{s}$ that is not properly branching in $\mathcal{C}^s$ due to violating **(pb1)**, but $\mathfrak{s}$ is correct in $\mathcal{C}^s$ according to the other properties, then

- $h(t_0) = t_0$ and $h(t_1) \neq t_0$, if $\mathfrak{s} = \boldsymbol{q}_{TA}^{-}$;
- $h(t_1) = t_1$ and $h(t_0) \neq t_1$, if $\mathfrak{s} = \boldsymbol{q}_{AT}^{-}$.

We prove the following:

Lemma 4. *$\boldsymbol{M}$ rejects $\boldsymbol{w}$ iff there is $K < \omega$ such that every $\mathcal{C} \in \mathfrak{K}_{\boldsymbol{q}}$ contains a homomorphic image of some $\mathcal{C}^{-} \in \mathfrak{K}_{\boldsymbol{q}}$ of depth $\leq K$.*

($\Leftarrow$) Suppose $\boldsymbol{M}$ accepts $\boldsymbol{w}$, and so there is an accepting computation-tree $\mathcal{T}_{accept}$. We construct an ideal tree β^{∞} by using only $\beta^{+}_{\mathcal{T}_{accept}}$ in every step, and then take a subtree $\beta^{\infty}_{\mathfrak{a}}$ whose root $\mathfrak{a}$ is the main node of some $c_{init(\boldsymbol{w})}$. Given $K < \omega$, we take the $(K{+}1)$-cut β_K of the desired tree $\beta^{\infty}_{\mathfrak{a}}$. By the ($\Rightarrow$) direction of Claim 4.1, every node of depth $\leq K$ in β_K is correct in β_K. By the construction of β^{∞} from the accepting computation-tree $\mathcal{T}_{accept}$, no node in β_K represents a q_{reject}-configuration. Let $\mathcal{C}_K$ be the cactus such that $\mathcal{C}_K^s = \beta_K$. Then, by the ($\Rightarrow$) direction of **(leaf)**, there is no homomorphism from $\boldsymbol{q}_{TT}^{-}$ to $\mathcal{C}_K$ mapping $\boldsymbol{q}_{TT}^{-}$ into some non-leaf segment of $\mathcal{C}_K$.

It follows that no cactus $\mathcal{C}^{-}$ of depth $\leq K$ can be homomorphically mapped to $\mathcal{C}_K$. Indeed, suppose on the contrary that there is a homomorphism h from $\mathcal{C}^{-}$ to $\mathcal{C}_K$. By **(foc)**, h must map the root segment of $\mathcal{C}^{-}$ into the root segment of $\mathcal{C}_K$. Thus, any leaf segment $\mathfrak{s}$ of $\mathcal{C}^{-}$ (and so $\boldsymbol{q}_{TT}^{-}$) should be mapped by h into some segment $\mathfrak{s}'$ of $\mathcal{C}_K$ whose depth is $\leq K$ in $\mathcal{C}_K^s$, and so $\mathfrak{s}'$ is a non-leaf segment of $\mathcal{C}_K$, which is a contradiction.

($\Rightarrow$) Suppose $\boldsymbol{M}$ rejects $\boldsymbol{w}$, and so every computation-tree is rejecting. Let $K = \boldsymbol{e} + 8\boldsymbol{d} + 19$ and take some cactus $\mathcal{C}$ of depth $> K$. We will cut each of the long branches of $\mathcal{C}^s$ at some depth $\leq K$, and show that the resulting cactus $\mathcal{C}^{-}$ can be mapped homomorphically into $\mathcal{C}$.

To this end, take a branch $\mathcal{B}$ of $\mathcal{C}^s$ longer than K. We will cut $\mathcal{B}$ (and possibly some other branches) at some depth $\leq K$, and show that the resulting cactus $\mathcal{C}'$ can be mapped homomorphically into $\mathcal{C}$. There are two cases: either the $4\boldsymbol{d} + 11$-long prefix of $\mathcal{B}$ does not contain a $001{*}$-sequence, or it does. In the former case, the segment $\mathfrak{s}$ at the end of the $4\boldsymbol{d} + 11$-long prefix of $\mathcal{B}$ (which is a non-leaf segment of $\mathcal{C}^s$) is not good in $\mathcal{C}^s$, and so it is incorrect in $\mathcal{C}^s$. Thus, by the ($\Leftarrow$) direction of **(leaf)**, there is a homomorphism $h\colon \boldsymbol{q}_{TT}^{-} \to \mathcal{C}$ mapping $\boldsymbol{q}_{TT}^{-}$ into $\mathfrak{s}$. We cut $\mathcal{B}$ at $\mathfrak{s}$. Let $\mathfrak{s}'$ denote the leaf segment corresponding to $\mathfrak{s}$ in the resulting cactus $\mathcal{C}'$. Then by mapping $\mathfrak{s}' = \boldsymbol{q}_{TT}^{-}$ according to h and taking the isomorphism on any other segment of $\mathcal{C}'$, we obtain a homomorphism from $\mathcal{C}'$ to $\mathcal{C}$.

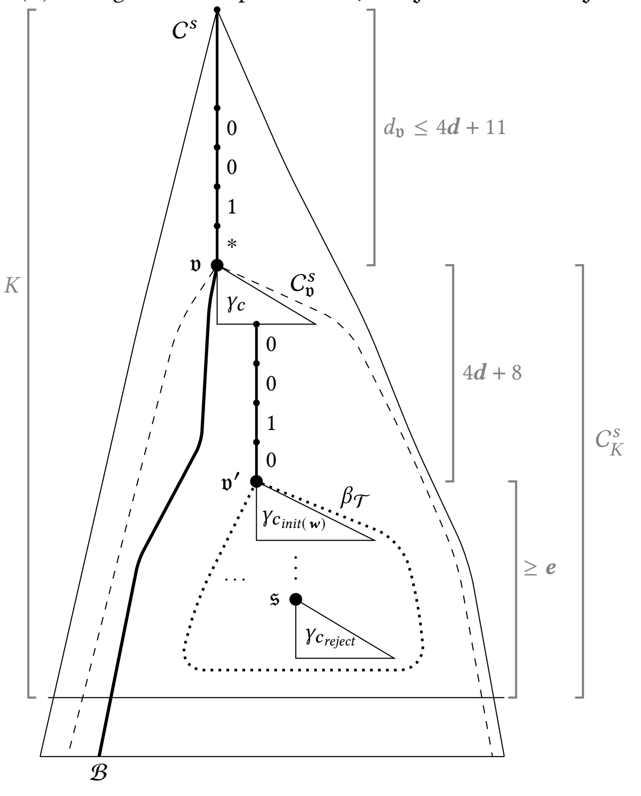
Now consider the latter case. We take some $001{*}$-sequence in the $4\boldsymbol{d} + 11$-long prefix of $\mathcal{B}$, and let $\mathfrak{v}$ be the segment at the end of this $001{*}$-sequence. We consider the subtree $\mathcal{C}^s_{\mathfrak{v}}$ of $\mathcal{C}^s$ with root $\mathfrak{v}$. We claim that there is a segment $\mathfrak{s}$ in $\mathcal{C}^s_{\mathfrak{v}}$ whose depth in $\mathcal{C}^s$ is $\leq K$ and such that

(correct) every segment on the path from $\mathfrak{v}$ to $\mathfrak{s}$ is correct in $\mathcal{C}^s$, and there is a homomorphism $h_0\colon \boldsymbol{q}_{TT}^{-} \to \mathcal{C}$ mapping $\boldsymbol{q}_{TT}^{-}$ into $\mathfrak{s}$.

Indeed, denote by $d_{\mathfrak{v}}$ the depth of $\mathfrak{v}$ in $\mathcal{C}^s$ (then $d_{\mathfrak{v}} \leq 4\boldsymbol{d}+11$). There are two cases:

(i) There is some segment of depth $< K - d_{\mathfrak{v}}$ in $\mathcal{C}^s_{\mathfrak{v}}$ that is incorrect in $\mathcal{C}^s_{\mathfrak{v}}$. Then we choose such a segment $\mathfrak{s}$ for which every segment on the path from $\mathfrak{v}$ to $\mathfrak{s}$ is correct in $\mathcal{C}^s_{\mathfrak{v}}$ (and so in $\mathcal{C}^s$). If $\mathfrak{s}$ is a leaf of $\mathcal{C}^s_{\mathfrak{v}}$ (and so of $\mathcal{C}^s$), then the isomorphism from $\boldsymbol{q}_{TT}^{-}$ to $\mathfrak{s}$ is a homomorphism from $\boldsymbol{q}_{TT}^{-}$ to $\mathcal{C}$ mapping $\boldsymbol{q}_{TT}^{-}$ into $\mathfrak{s}$. Otherwise, by the ($\Leftarrow$) direction of **(leaf)**, there is a homomorphism from $\boldsymbol{q}_{TT}^{-}$ to $\mathcal{C}$ mapping $\boldsymbol{q}_{TT}^{-}$ into the non-leaf segment $\mathfrak{s}$ (whose depth in $\mathcal{C}^s$ is $\leq K$).

(ii) All segments of depth $< K - d_{\mathfrak{v}}$ in $\mathcal{C}^s_{\mathfrak{v}}$ are correct in $\mathcal{C}^s_{\mathfrak{v}}$.



Then let $\mathcal{C}^s_K$ be the $(K - d_{\mathfrak{v}})$-cut of $\mathcal{C}^s_{\mathfrak{v}}$. As $\mathcal{C}^s_K$ is a substructure of $\mathcal{C}^s_{\mathfrak{v}}$, all segments of depth $< K - d_{\mathfrak{v}}$ in $\mathcal{C}^s_K$ are correct in $\mathcal{C}^s_K$. So, by the ($\Leftarrow$) direction of Claim 4.1, $\mathcal{C}^s_K$ is isomorphic to the $(K - d_{\mathfrak{v}})$-cut of some desired tree, and therefore $\mathfrak{v}$ represents some configuration c. Whichever configuration c is, there is a segment $\mathfrak{v}'$ of depth $\leq 4\boldsymbol{d}+8$ in $\mathcal{C}^s_K$ that is the main node of $c_{init(\boldsymbol{w})}$. As $K - d_{\mathfrak{v}} - (4\boldsymbol{d} + 8) \geq \boldsymbol{e}$, it follows that there is a computation-tree $\mathcal{T}$ such that $\beta_{\mathcal{T}}$ is a substructure of the subtree of $\mathcal{C}^s_K$ with root $\mathfrak{v}'$ (as the depth of each $\beta_{\mathcal{T}}$ is $\boldsymbol{e}$). Thus, there is a segment $\mathfrak{s}$ in $\mathcal{C}^s_K$ representing a q_{reject}-configuration c_{reject} in $\mathcal{C}^s_K$ (because every computation-tree is rejecting, and so $\mathcal{T}$ is rejecting). As $\mathfrak{s}$ is a non-leaf segment in $\mathcal{C}^s_K$ and $\mathcal{C}^s_K$ is a substructure of $\mathcal{C}^s$, $\mathfrak{s}$ is a non-leaf segment representing c_{reject} in $\mathcal{C}^s$ whose depth is $\leq K$ in $\mathcal{C}^s$. Thus, by the ($\Leftarrow$) direction of **(leaf)**, there is a homomorphism from $\boldsymbol{q}^-_{TT}$ to C mapping $\boldsymbol{q}^-_{TT}$ into $\mathfrak{s}$.

So in both cases (i) and (ii), we have shown that there is a segment $\mathfrak{s}$ of depth $\leq K$ in C^s such that **(correct)** holds. However, $\mathfrak{s}$ is not necessarily in the branch $\mathcal{B}$. Let $\mathfrak{s}_m$ be the last ancestor of $\mathfrak{s}$ in $\mathcal{B}$, and list the segments $\mathfrak{s} = \mathfrak{s}_0, \mathfrak{s}_1, \dots, \mathfrak{s}_m$ on the path leading upwards from $\mathfrak{s}$ to $\mathfrak{s}_m$. Let C' be obtained from C by cutting at $\mathfrak{s}_i$ every branch of C^s going through $\mathfrak{s}_i$ other than the one going to $\mathfrak{s}$, for every $i \leq m$. (In particular, $\mathcal{B}$ is cut at $\mathfrak{s}_m$ which is of depth $\leq K$.) Let $\mathfrak{s}^\star_i$ denote the segment corresponding to $\mathfrak{s}_i$ in C'. Then $\mathfrak{s}^\star_0$ is a leaf in C', and so $\mathfrak{s}^\star_0 = \boldsymbol{q}^-_{TT}$. Also, for each $i > 0$,

– either $\mathfrak{s}^\star_i = \mathfrak{s}_i$
– or $\mathfrak{s}_i = \boldsymbol{q}^-_{AA}$ and $\mathfrak{s}^\star_i$ is either $\boldsymbol{q}^-_{AT}$ or $\boldsymbol{q}^-_{TA}$.

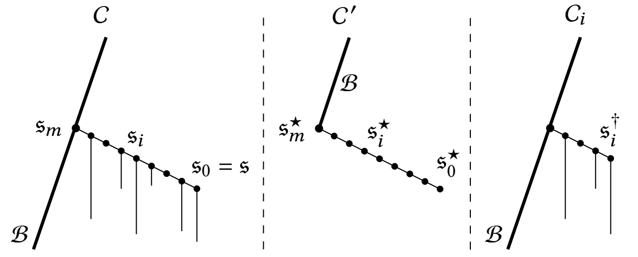
We claim that, for every $i \leq m$, there is some homomorphism $h_i\colon \mathfrak{s}^\star_i \to C$ mapping $\mathfrak{s}^\star_i$ into $\mathfrak{s}_i$ and such that

if $\mathfrak{s}^\star_{i-1}$ is the j-child of $\mathfrak{s}^\star_i$, for $j = 0, 1$, then

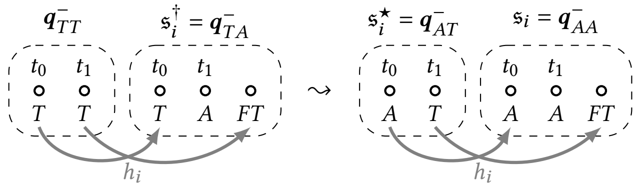
$$h_i \text{ maps the } t_j\text{-node of } \mathfrak{s}^\star_i \text{ to the } t_j\text{-node of } \mathfrak{s}_i. \qquad (10)$$

This will be enough for building a homomorphism from C' to C: we take these h_i on each $\mathfrak{s}^\star_i$, and the isomorphism on any other segment.

Indeed, if $i = 0$ then the h_0 in **(correct)** is suitable. If $i > 0$ and $\mathfrak{s}^\star_i = \mathfrak{s}_i$, then the isomorphism is suitable for h_i. So suppose that $\mathfrak{s}^\star_i \neq \mathfrak{s}_i$ (so $\mathfrak{s}_i = \boldsymbol{q}^-_{AA}$). We consider the case when $\mathfrak{s}^\star_i = \boldsymbol{q}^-_{AT}$, that is, $\mathfrak{s}^\star_{i-1}$ is a 0-child of $\mathfrak{s}^\star_i$ (the case when $\mathfrak{s}^\star_i = \boldsymbol{q}^-_{TA}$ is similar). Let C_i be obtained from C by cutting at $\mathfrak{s}_i$ the branch leading to $\mathfrak{s}$. Let $\mathfrak{s}^\dagger_i$ denote the segment corresponding to $\mathfrak{s}_i$ in C_i, that is, $\mathfrak{s}^\dagger_i = \boldsymbol{q}^-_{TA}$.



By **(correct)**, $\mathfrak{s}_i$ is correct in C^s, and so $\mathfrak{s}_i$ is properly branching in C^s. Thus, $\mathfrak{s}^\dagger_i$ is incorrect in C^s_i because it violates condition **(pb1)** in Sec. 3.3.2. On the other hand, $\mathfrak{s}^\dagger_i$ is correct in C^s_i in all the other aspects (this is because apart from $\mathfrak{s}_i$ and some of its descendants, every other segment is the same in both cactuses C and C_i). Therefore, by the ($\Leftarrow$) direction of **(leaf)**, there is a homomorphism $h_i\colon \boldsymbol{q}^-_{TT} \to C_i$ mapping $\boldsymbol{q}^-_{TT}$ into $\mathfrak{s}^\dagger_i$. Also, by **(branch)**, the same h_i is a homomorphism from $\mathfrak{s}^\star_i$ to C, mapping $\mathfrak{s}^\star_i$ to $\mathfrak{s}_i$ and such that (10) holds:



$\boldsymbol{q}^-_{TT}$
$\mathfrak{s}^\dagger_i = \boldsymbol{q}^-_{TA}$
$\mathfrak{s}^\star_i = \boldsymbol{q}^-_{AT}$
$\mathfrak{s}_i = \boldsymbol{q}^-_{AA}$
t_0
t_1
T
A
FT
$\rightsquigarrow$
h_i

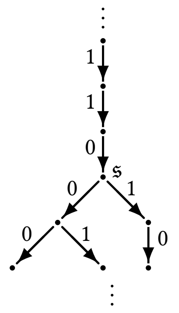
So in any case we showed that there exists a $C' \to C$ homomorphism, for some subcactus C' of C where branch $\mathcal{B}$ is cut at some depth $\leq K$. If C' still has branches longer than K, we repeat the above process for a long branch in C' to obtain a $C'' \to C'$ homomorphism for some C'', and so on. At the end, we obtain a cactus C^- of depth $\leq K$ homomorphically mapping into C, which completes the proof of Lemma 4.

B PROOF OF CLAIM 4.2

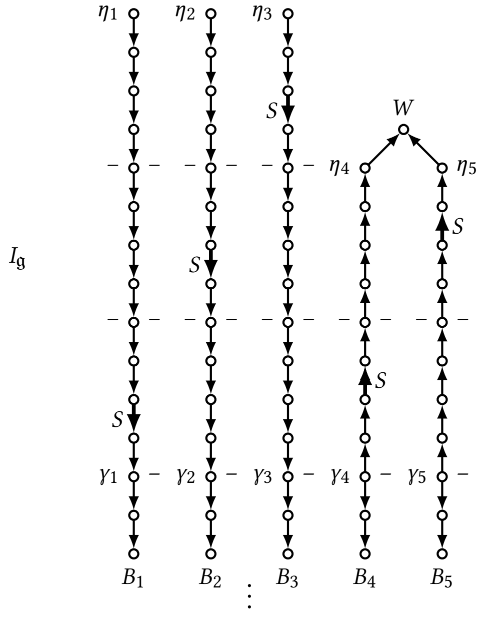
($\Rightarrow$) Suppose that, for some C and $\mathfrak{s}$, a gadget $\mathfrak{g}$ implementing a formula $\varphi_\mathfrak{g}(y_1, \dots, y_n)$ is triggered at $\mathfrak{s}$. Then there is a $h\colon \boldsymbol{q}^-_{TT} \to C$ homomorphism mapping the $I_\mathfrak{g}$-block in $\boldsymbol{q}^-_{TT}$ to the $M_\mathfrak{g}$-block in $\mathfrak{s}$. In particular, $h(\iota_\mathfrak{g}) = \alpha$, and so $h(\pi_\mathfrak{g}) = \varrho_\mathfrak{g}$. Thus, for every $i \leq n$, the B_i-node in $I_\mathfrak{g}$ must also be mapped to one of the two B_i-nodes in the $M_\mathfrak{g}$-block of $\mathfrak{s}$ (either β^T_i or β^F). However, which of these two B_i-nodes is the image depends on the truth-value $b^\mathfrak{s}_i$ of the gathered input $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g} = (b^\mathfrak{s}_1, \dots, b^\mathfrak{s}_n)$ on the variable y_i. We claim that

(i) if $b^\mathfrak{s}_i = 0$, then the B_i-node in $I_\mathfrak{g}$ is mapped by h to β^F;
(ii) if $b^\mathfrak{s}_i = 1$, then the B_i-node in $I_\mathfrak{g}$ is mapped by h to β^T_i.

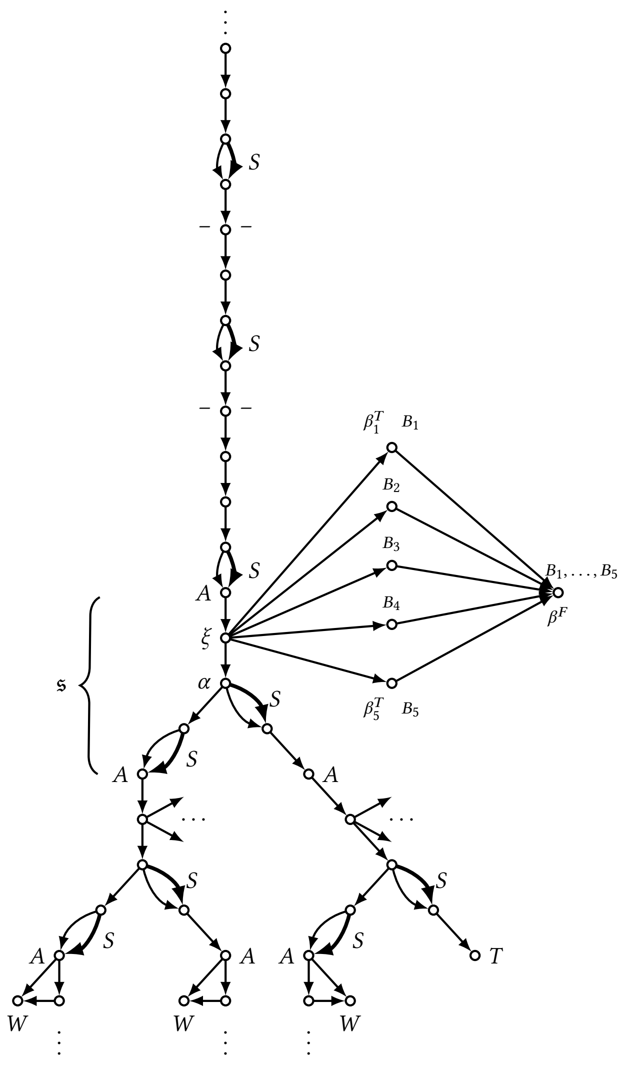
Instead of proving (i) and (ii), here we give an illustrative example. Suppose $\varphi_\mathfrak{g}(y_1, \dots, y_5)$ is such that (y_1, y_2, y_3) should be gathered from the 3-long uppath, and (y_4, y_5) from a 2-long downpath. Suppose the 'environment' of $\mathfrak{s}$ in C^s looks like this:



Then if h is a homomorphism triggering $\mathfrak{g}$ at $\mathfrak{s}$, then the possible inputs $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ that can be gathered are 01100, 01101, or 01110, because h should map the pattern



to the pattern shown below:



(We are also using that the parts of gadgets that are not depicted above do not contain W-nodes, so the h-image cannot 'stray' there when taking a downpath.)

It remains to see how h maps the remaining part of the $I_\mathfrak{g}$-block into the $M_\mathfrak{g}$-block of $\mathfrak{s}$. We claim that for every non-leaf gate g in $\varphi_\mathfrak{g}$, if g^ℓ_{ij} is an occurrence of g on some branch, then the end-node p^ℓ_{ij} of the RSR-pattern corresponding to g^ℓ_{ij} in $I_\mathfrak{g}$ is mapped in such a way that

(iii) $h(p^\ell_{ij})$ is the $\mathbf{o}$-node of the gadget for g, whenever the value of g under $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ is 0;

(iv) $h(p^\ell_{ij})$ is the (D)-node of the gadget for g, whenever the value of g under $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ is 1.

We prove this by induction on the tree-structure of $\varphi_\mathfrak{g}$, going from leaves to root. Take some gate g, and let g^ℓ_{ij} be an occurrence of g.

First, suppose that g is an AND-gate. There are many cases, depending on the truth-values of g and its two inputs g_1 and g_2 under $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$, and also on whether each of the g_i is a leaf gate or not. We consider just two cases, the other ones are similar.

– Suppose that the value of g under $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ is 0, $\ell = 1$ (and so g_1 is a leaf labelled by y_i), and $b^\mathfrak{s}_i = 1$. Suppose that g_2 is also a leaf gate, and so g_2 has value 0 under $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$. Let $g^1_{i'j'}$ be an occurrence of g_2. By (ii), the B_{ij}-node in $I_\mathfrak{g}$ is mapped by h to the upper B_{ij}-node in the $M_\mathfrak{g}$-block of $\mathfrak{s}$. So the first R-edge of the RSR-pattern corresponding to g^1_{ij} is mapped to the R-edge connecting the two B_{ij}-nodes. Thus, the S-edge of the RSR-pattern corresponding to g^1_{ij} must be mapped to an S-edge starting at the $\mathbf{i}_1$-node of the g-gadget. Similarly, by (i), the $B_{i'j'}$-node in $I_\mathfrak{g}$ is mapped by h to the lower $B_{i'j'}$-node in the $M_\mathfrak{g}$-block of $\mathfrak{s}$. So the S-edge of the RSR-pattern corresponding to $g^1_{i'j'}$ must be mapped to an S-edge following an R-edge starting at the $\mathbf{i}_2$-node of the g-gadget. As h preserves E, the end-nodes of these two S-edges in the g-gadget must coincide, and so it must be node c_1. So $h(p^1_{ij})$ is the $\mathbf{o}$-node of the g-gadget.

– Suppose that the value of g under $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ is 1, and both of its inputs are non-leaf gates having value 1 under $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$. Suppose $g^{\ell-1}_{ij}$ is an occurrence of g_1 and $g^{\ell'}_{i'j'}$ is an occurrence of g_2. By the IH, $h(p^{\ell-1}_{ij})$ is the (D)-node of the gadget for g_1, and $h(p^{\ell'}_{i'j'})$ is the (D)-node of the gadget for g_2. Then the S-edges of the RSR-patterns corresponding to $g^{\ell-1}_{ij}$ and $g^{\ell'}_{i'j'}$ must be mapped, respectively, to S-edges starting at the $\mathbf{i}_1$- and $\mathbf{i}_2$-nodes of the g-gadget. As h preserves E, the end-nodes of these two S-edges in the g-gadget must coincide, and so it must be node b. So $h(p^\ell_{ij})$ is the (D)-node of the g-gadget, as required.

The case when g is a NOT-gate can be handled similarly, thereby completing the proof of (iii) and (iv). As h preserves D, it follows that $\varphi_\mathfrak{g}[\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}] = 1$.

($\Leftarrow$) If there is $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ such that $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ is gathered from 'around' $\mathfrak{s}$ in C^s according to the input-types for $\varphi_\mathfrak{g}$ and $\varphi_\mathfrak{g}[\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}] = 1$, then we define a function $h\colon \boldsymbol{q}^-_{TT} \to C$ by taking

– $h(\alpha) = \tau_\mathfrak{g}$ for the $\tau_\mathfrak{g}$-node of $\mathfrak{s}$,
– $h(\iota_\mathfrak{g}) = \alpha$ for the α-node of $\mathfrak{s}$,

and mapping

– the $I_\mathfrak{g}$-block to the $M_\mathfrak{g}$-block of $\mathfrak{s}$ following the structure of $\boldsymbol{b}^\mathfrak{s}_\mathfrak{g}$ and $\varphi_\mathfrak{g}$ as described above,
– the $I_{\mathfrak{g}_i}$-block of every gadget $\mathfrak{g}_i$ different from $\mathfrak{g}$ to the $I_{\mathfrak{g}_i}$-block of $\mathfrak{s}$.
– the $M_{\mathfrak{g}_i}$-block of every gadget $\mathfrak{g}_i$ to the $M'_{\mathfrak{g}_i}$-block of $\mathfrak{s}$, and
– the $M'_{\mathfrak{g}_i}$-block of every gadget $\mathfrak{g}_i$ also to the $M'_{\mathfrak{g}_i}$-block of $\mathfrak{s}$.

Using the interaction-regulating mechanism between different gadgets described in Sec. 3.5.1, it is easy to see that h is a homomorphism, and $\mathfrak{g}$ is triggered by h at $\mathfrak{s}$.

C PROOF OF THEOREM 7

We prove the following:

Theorem 7. *Suppose $\boldsymbol{q}$ is a minimal ditree CQ with at least one solitary F, at least one solitary T and such that either*

(i) *there is a $\prec$-comparable solitary pair* (t, f) *or* (*ii*) $\boldsymbol{q}$ *is not quasi-symmetric and has no* FT*-twins.*

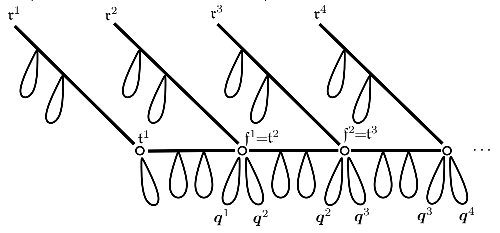
Then evaluating the d-sirup $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ *is NL-hard.*

The proof is by reduction of the NL-complete reachability problem for dags. Given a dag $G = (V, E)$ with nodes $\mathfrak{s}, \mathfrak{t} \in V$, we construct a data instance $\mathcal{D}_G$ as follows. We pick a solitary pair $(\mathfrak{t}, \mathfrak{f})$ such that, in case (i), $(\mathfrak{t}, \mathfrak{f})$ is $\prec$-comparable and there is no solitary T- or F-node between $\mathfrak{t}$ and $\mathfrak{f}$; and, in case (ii), $(\mathfrak{t}, \mathfrak{f})$ is of minimal distance, $\prec$-incomparable, and not symmetric. Then, in both cases, we replace each $e = (\mathfrak{u}, \mathfrak{v}) \in E$ by a fresh copy $\boldsymbol{q}^e$ of $\boldsymbol{q}$ in which $\mathfrak{t}^e$ is renamed to $\mathfrak{u}$ with $T(\mathfrak{u})$ replaced by $A(\mathfrak{u})$, and $\mathfrak{f}^e$ is renamed to $\mathfrak{v}$ with $F(\mathfrak{v})$ replaced by $A(\mathfrak{v})$. The dag $\mathcal{D}_G$ comprises the $\boldsymbol{q}^e$, for $e \in E$, as well as $T(\mathfrak{s})$ and $F(\mathfrak{t})$. We show that $\mathfrak{s} \to_G \mathfrak{t}$ iff the answer to $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ over $\mathcal{D}_G$ is 'yes'.

($\Rightarrow$) If $\mathfrak{s} = \mathfrak{v}_0, \dots, \mathfrak{v}_n = \mathfrak{t}$ is a path in G with $e_i = (\mathfrak{v}_i, \mathfrak{v}_{i+1}) \in E$, for $i < n$, then for any model $\mathcal{I}$ of $\Delta_{\boldsymbol{q}}$ and $\mathcal{D}_G$, there is some $i < n$ such that $\mathcal{I} \models T(\mathfrak{v}_i)$ and $\mathcal{I} \models F(\mathfrak{v}_{i+1})$, and so the identity map from $\boldsymbol{q}$ to its copy $\boldsymbol{q}^{e_i}$ is a $\boldsymbol{q} \to \mathcal{I}$ homomorphism.

($\Leftarrow$) If $\mathfrak{s} \not\to_G \mathfrak{t}$, we define a model $\mathcal{I}$ of $\Delta_{\boldsymbol{q}}$ and $\mathcal{D}_G$ by labelling with T the A-nodes in $\mathcal{D}_G$ that (as nodes of G) are reachable from $\mathfrak{s}$ (via a directed path in G) and with F the remaining ones. We call these A-nodes *contacts*. We claim that if one of (i) or (ii) holds, then there is no homomorphism from $\boldsymbol{q}$ to $\mathcal{I}$, and so the answer to $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ over $\mathcal{D}_G$ is 'no'. Indeed, take any map h from $\boldsymbol{q}$ to $\mathcal{I}$, and consider the substructure $\mathcal{H}^{(\mathfrak{t},\mathfrak{f})}$ of $\mathcal{D}_G$ comprising those copies $\boldsymbol{q}^{e_1}, \dots, \boldsymbol{q}^{e_n}$ of $\boldsymbol{q}$ that have a non-empty intersection with $h(\boldsymbol{q})$. To simplify notation, we set $\boldsymbol{q}^j = \boldsymbol{q}^{e_j}$. Then $\mathcal{I}$ can be regarded as a model of $\mathcal{H}^{(\mathfrak{t},\mathfrak{f})}$. We show that h cannot be a homomorphism from $\boldsymbol{q}$ to $\mathcal{I}$. We prove the two cases (i) and (ii) separately. Throughout, for any ditree CQ $\boldsymbol{q}'$ and node x in it, we denote by $\boldsymbol{q}'_x$ the sub-ditree of $\boldsymbol{q}'$ with root x.

In case (i), we picked a solitary pair $(\mathfrak{t}, \mathfrak{f})$ such that it is $\prec$-comparable and there is no solitary T- or F-node between $\mathfrak{t}$ and $\mathfrak{f}$. Suppose that $\mathfrak{t} \prec \mathfrak{f}$ (the other case is similar). Then $\mathcal{H}^{(\mathfrak{t},\mathfrak{f})}$ is as follows:



We may assume that the model $\mathcal{I}$ over $\mathcal{H}^{(\mathfrak{t},\mathfrak{f})}$ is such that the contacts between the $\boldsymbol{q}$-copies are either all in $F^{\mathcal{I}}$ or all in $T^{\mathcal{I}}$. We track the location of $h(\mathfrak{t})$. Let a be such that $h(\mathfrak{t})$ is in $\boldsymbol{q}^a$ and $h(\mathfrak{t}) \neq \mathfrak{f}^a$. We consider several cases, and show that none of them is possible.

(1) $h(\mathfrak{t}) \prec_{\boldsymbol{q}^a} \mathfrak{t}^a$.
Then $h(\mathfrak{r})$ is also in $\boldsymbol{q}^a$, and so there is not enough room for $h(\boldsymbol{q})$ in $\boldsymbol{q}^a$.

(2) $h(\mathfrak{t})$ and $\mathfrak{t}^a$ are $\prec_{\boldsymbol{q}^a}$-incomparable.
Then there is x such that $\mathfrak{r} \preceq x \prec \mathfrak{t}$ and $x^a = \inf_{\boldsymbol{q}^a}\big(h(\mathfrak{t}), \mathfrak{t}^a\big)$. So $h(x) \prec_{\boldsymbol{q}^a} h(\mathfrak{t})$. If $h(x) \prec_{\boldsymbol{q}^a} x^a$ as well, then there is not enough room for $h(\boldsymbol{q})$ in $\boldsymbol{q}^a$. So $x^a \preceq_{\boldsymbol{q}_a} h(x) \prec_{\boldsymbol{q}^a} h(\mathfrak{t})$. If $x^a \prec_{\boldsymbol{q}_a} h(x)$ then $\boldsymbol{q}_x$ is mapped by h into $\boldsymbol{q}^a_{h(x)}$, but there is no room for that. So it follows that $h(x) = x^a$. Let y be the child of x in $\boldsymbol{q}$ with $y \preceq \mathfrak{t}$. Then $h(y)$ is the child of x^a in $\boldsymbol{q}^a$ that is $\preceq_{\boldsymbol{q}^a} h(\mathfrak{t})$, and so $y^a \neq h(y)$. Thus, $\boldsymbol{q}^a_{y^a}$ and $\boldsymbol{q}^a_{h(y)}$ are disjoint subtrees of $\boldsymbol{q}^a$, and so $\boldsymbol{q}_y$ and $\boldsymbol{q}_{\iota^a(h(y))}$ are disjoint subtrees of $\boldsymbol{q}$. Therefore, the following function f is a $\boldsymbol{q} \to (\boldsymbol{q} \setminus \boldsymbol{q}_y)$ homomorphism:

$$f(z) = \begin{cases} \iota^a\big(h(z)\big), & \text{if } z \text{ is in } \boldsymbol{q}_y,\\ z & \text{otherwise,}\end{cases}$$

contrary to the minimality of $\boldsymbol{q}$.

(3) $h(\mathfrak{t}) = \mathfrak{t}^a$.
Then all contacts are in $T^{\mathcal{I}}$, and so $h(\mathfrak{f}) \neq \mathfrak{f}^a$. Thus, there is x such that $\mathfrak{t} \preceq x \prec \mathfrak{f}$ and $h(x) = x^a = \inf_{\boldsymbol{q}^a}\big(\mathfrak{f}^a, h(\mathfrak{f})\big)$. Let y be the child of x with $y \preceq \mathfrak{f}$. Then $h(y)$ is the child of x^a in $\boldsymbol{q}^a$, that is $\preceq_{\boldsymbol{q}^a} h(\mathfrak{f})$, and so $y^a \neq h(y)$. Thus, $\boldsymbol{q}^a_{y^a}$ and $\boldsymbol{q}^a_{h(y)}$ are disjoint subtrees of $\boldsymbol{q}^a$, and so $\boldsymbol{q}_y$ and $\boldsymbol{q}_{\iota^a(h(y))}$ are disjoint subtrees of $\boldsymbol{q}$. Therefore, the following function f is a $\boldsymbol{q} \to (\boldsymbol{q} \setminus \boldsymbol{q}_y)$ homomorphism:

$$f(z) = \begin{cases} \iota^a\big(h(z)\big), & \text{if } z \text{ is in } \boldsymbol{q}_y,\\ z & \text{otherwise,}\end{cases}$$

contrary to the minimality of $\boldsymbol{q}$.

(4) $\mathfrak{t}^a \prec_{\boldsymbol{q}^a} h(\mathfrak{t})$.
Since there are no T-nodes between $\mathfrak{t}^a$ and $\mathfrak{f}^a$, there are three cases: either (a) $\mathfrak{f}^a \preceq_{\boldsymbol{q}^a} h(\mathfrak{t})$, or (b) $(h(\mathfrak{t}), \mathfrak{f}^a)$ is $\prec_{\boldsymbol{q}^a}$-incomparable, or (c) $h(\mathfrak{t}) = \vartheta^a_*$ for some FT-twin ϑ_* with $\mathfrak{t} \prec \vartheta_* \prec \mathfrak{f}$.

(a) Then $\boldsymbol{q}_{\mathfrak{t}}$ is mapped by h into $\boldsymbol{q}^a_{\mathfrak{f}}$, but there is no room for that as $\mathfrak{t} \prec \mathfrak{f}$.

(b) Then $\boldsymbol{q}_{\mathfrak{t}}$ is mapped by h into $\boldsymbol{q}^a_x$ for some x with $\mathfrak{t} \prec x$, but there is no room for that.

(c) Then the set $X = \{\vartheta^\ell \in h(\boldsymbol{q}) \mid \vartheta \text{ is an } FT\text{-twin with } \mathfrak{t} \prec \vartheta\}$ is not empty. We define a function $g_\leftarrow \colon h(\boldsymbol{q}) \to h(\boldsymbol{q})$ by taking $g_\leftarrow(x) = h\big(\iota^\ell(x)\big)$ whenever x is a node in $\boldsymbol{q}^\ell$, where we consider each contact $\mathfrak{c} = \mathfrak{f}^i = \mathfrak{t}^{i+1}$, for $1 \le i < n$, as a node in $\boldsymbol{q}^{i+1}$, that is, $g_\leftarrow(\mathfrak{c}) = g_\leftarrow(\mathfrak{t}^{i+1}) = h\big(\iota^{i+1}(\mathfrak{t}^{i+1})\big) = h(\mathfrak{t})$. Throughout, we use the following obvious 'shift' property of $g_\leftarrow$: for every ℓ, $1 \le \ell \le n$,

if y, z are both in the same copy $\boldsymbol{q}^\ell$,
$y, z \neq \mathfrak{f}^\ell$ whenever $\ell < n$, and $y \preceq_{\boldsymbol{q}^\ell} z$,
then $g_\leftarrow(y) \preceq_{h(\boldsymbol{q})} g_\leftarrow(z)$
and $\delta_{\boldsymbol{q}^\ell}(y, z) = \delta_{h(\boldsymbol{q})}\big(g_\leftarrow(y), g_\leftarrow(z)\big)$. (11)

It is straightforward to see that the restriction $g\,|_X$ of $g_\leftarrow$ to X is an $X \to X$ function. As X is finite, there exists a 'fixpoint' of $g\,|_X$: a node ϑ in X and a number $N > 0$ such that $(g\,|_X)^N(\vartheta) = \vartheta$. We regard this fixpoint-cycle as a fixpoint-cycle of $g_\leftarrow$, and will 'shift it to the left.' More precisely, we claim that

$$\text{there is a contact } \mathfrak{c} \text{ with } g^N_\leftarrow(\mathfrak{c}) = \mathfrak{c}. \tag{12}$$

Indeed, let $y_0 = \vartheta, y_1 = g_\leftarrow(\vartheta), y_2 = g^2_\leftarrow(\vartheta), \dots, y_{N-1} = g^{N-1}_\leftarrow(\vartheta)$. For every $j < N$, we have $y_j \in X$, and so there is a contact $\mathfrak{t}^{\ell_j}$ with $\mathfrak{t}^{\ell_j} \prec_{\boldsymbol{q}^{\ell_j}} y_j$. We let $d_j = \delta_{\boldsymbol{q}^{\ell_j}}\big(\mathfrak{t}^{\ell_j}, y_j\big)$. Let $K < N$ be such that $d_K = \min\{d_j \mid j < N\}$, and set $\mathfrak{c} = \mathfrak{f}^{\ell_K - 1} = \mathfrak{t}^{\ell_K}$. As each y_j is an FT-twin, we have $y_j \neq \mathfrak{f}^{\ell_j}$ whenever $j < N$. Thus, by the definition of K

and (11), for every $j \leq N$, $g_{\leftarrow}^{j}(\mathsf{c})$ belongs to the same copy $\boldsymbol{q}^{\ell_{K+j}}$ as y_{K+j}, $g_{\leftarrow}^{j}(\mathsf{c}) \preceq_{\boldsymbol{q}^{\ell_{K+j}}} y_{K+j}$, and

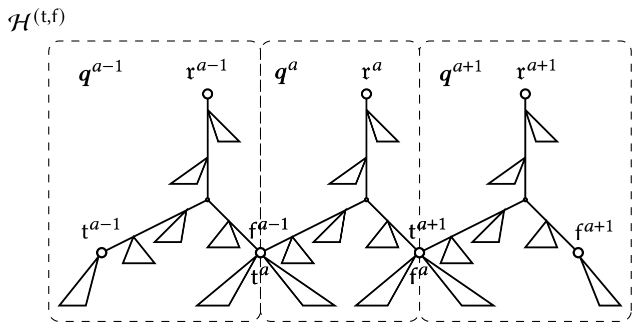
$$d_K = \delta_{\boldsymbol{q}^{\ell_{K+j}}}\big(g_{\leftarrow}^{j}(\mathsf{c}), y_{K+j}\big).$$

It follows, in particular, that $g_{\leftarrow}^{N}(\mathsf{c})$ belongs to the same copy $\boldsymbol{q}^{\ell_K}$ as y_K, and

$$\delta_{\boldsymbol{q}^{\ell_K}}\big(g_{\leftarrow}^{N}(\mathsf{c}), y_K\big) = \delta_{\boldsymbol{q}^{\ell_K}}(\mathsf{c}, y_K).$$

Therefore, $g_{\leftarrow}^{N}(\mathsf{c}) = \mathsf{c}$, as required in (12).
It remains to show that (12) leads to a contradiction. Indeed, we have either $\mathsf{c} \notin F^{\mathcal{I}}$ or $\mathsf{c} \notin T^{\mathcal{I}}$. On the other hand, by our assumption $g_{\leftarrow}^{j}(\mathsf{c}) = h(\mathsf{t}) = \vartheta_*^a$ for some FT-twin ϑ_*, and so $g_{\leftarrow}^{j}(\mathsf{c})$ is an FT-twin for every j with $1 \leq j \leq N$.

In case (ii), $\boldsymbol{q}$ is not quasi-symmetric, and we may also assume that $\boldsymbol{q}$ has no $\prec$-comparable solitary pairs. In this case, we picked a solitary pair (t, f) such that it is of minimal distance, $\prec$-incomparable, and not symmetric. Suppose $h \colon \boldsymbol{q} \to \mathcal{I}$ is a homomorphism, and let a be such that $h(\mathfrak{r}) \in \boldsymbol{q}^a$. As (t, f) is $\prec$-incomparable, $\mathcal{H}^{(\mathsf{t},\mathsf{f})}$ consists of (at most) three copies $\boldsymbol{q}^a$, $\boldsymbol{q}^{a-1}$ and $\boldsymbol{q}^{a+1}$ of $\boldsymbol{q}$, and looks as shown in the picture below:



As (t, f) is not symmetric, $\mathcal{I}$ is such that the contacts between the $\boldsymbol{q}$-copies are either both in $F^{\mathcal{I}}$ or both in $T^{\mathcal{I}}$.

The following 'structural' claim (tracking the possible locations of $h(\mathsf{f})$ and $h(\mathsf{t})$) is also used in the proof of Theorem 11:

Claim 7.1. *Suppose* (t, f) *is* $\prec$*-incomparable and of minimal distance (though* $\boldsymbol{q}$ *might contain FT-twins). If* $\mathsf{m} = \inf_{\boldsymbol{q}}(\mathsf{t}, \mathsf{f})$ *then* $h(\mathsf{m})$ *is in* $\boldsymbol{q}^a$*, and one of the following holds*:

(1) $\mathsf{m}^a \prec_{\boldsymbol{q}^a} h(\mathsf{m}) \prec_{\boldsymbol{q}^a} \mathsf{t}^a$, $h(\mathsf{t})$ *is in* $\boldsymbol{q}^{a-1}$ *with* $\mathsf{f}^{a-1} \prec_{\boldsymbol{q}^{a-1}} h(\mathsf{t})$, *and* $h(\mathsf{f}) = \mathsf{t}^a$;
(2) $\mathsf{m}^a \prec_{\boldsymbol{q}^a} h(\mathsf{m}) \prec_{\boldsymbol{q}^a} \mathsf{f}^a$, $h(\mathsf{f})$ *is in* $\boldsymbol{q}^{a+1}$ *with* $\mathsf{t}^{a+1} \prec_{\boldsymbol{q}^{a+1}} h(\mathsf{f})$, *and* $h(\mathsf{t}) = \mathsf{f}^a$;
(3) $h(\mathsf{m}) = \mathsf{m}^a$, $h(\mathsf{f}) = \mathsf{f}^a$, *and* $h(\mathsf{t})$ *is in* $\boldsymbol{q}^a$ *with* $h(\mathsf{t}) \prec_{\boldsymbol{q}^a} \mathsf{f}^a$;
(4) $h(\mathsf{m}) = \mathsf{m}^a$, $h(\mathsf{t}) = \mathsf{t}^a$, *and* $h(\mathsf{f})$ *is in* $\boldsymbol{q}^a$ *with* $h(\mathsf{f}) \prec_{\boldsymbol{q}^a} \mathsf{t}^a$.

Proof. If $h(\mathsf{m}) \in \boldsymbol{q}^{a-1} \setminus \boldsymbol{q}^a$ then $\boldsymbol{q}_{\mathsf{m}}$ is mapped by h into $\boldsymbol{q}^{a-1}_{\mathsf{f}^{a-1}}$, but there is no room for that. Similarly, $h(\mathsf{m}) \in \boldsymbol{q}^{a+1} \setminus \boldsymbol{q}^a$ then $\boldsymbol{q}_{\mathsf{m}}$ is mapped by h into $\boldsymbol{q}^{a+1}_{\mathsf{t}^{a+1}}$, but there is no room for that. So $h(\mathsf{m})$ is in $\boldsymbol{q}^a$. We consider several cases:

– $h(\mathsf{m}) \prec_{\boldsymbol{q}^a} \mathsf{m}^a$.
Then there is not enough room for $h(\boldsymbol{q})$ in $\boldsymbol{q}^a$.
– $(h(\mathsf{m}), \mathsf{m}^a)$ is $\prec_{\boldsymbol{q}^a}$-incomparable.
Then let $x = \inf(h(\mathsf{m}), \mathsf{m}^a)$. If $h(x) = x^a$ then $\boldsymbol{q}$ is not minimal. If $x^a \prec_{\boldsymbol{q}^a} h(x) \prec_{\boldsymbol{q}^a} h(\mathsf{m})$ then $\boldsymbol{q}_x$ is mapped by h into $\boldsymbol{q}^a_{h(x)}$, but there is no room for that.
– $\mathsf{m}^a \prec_{\boldsymbol{q}^a} h(\mathsf{m})$, and both $(h(\mathsf{m}), \mathsf{t}^a)$ and $(h(\mathsf{m}), \mathsf{f}^a)$ are $\prec_{\boldsymbol{q}^a}$-incomparable.
Then $\boldsymbol{q}_{\mathsf{m}}$ is mapped by h into $\boldsymbol{q}^a_{h(\mathsf{m})}$, but there is no room for that.
– $h(\mathsf{m}) = \mathsf{t}^a$.
Then $h(\mathsf{t})$ cannot be in $\boldsymbol{q}^a$, as there is no room for that. So $h(\mathsf{t})$ is in $\boldsymbol{q}^{a-1}$, and so depth of $\boldsymbol{q}_{\mathsf{f}}$ > depth of $\boldsymbol{q}_{\mathsf{t}}$. Also, $h(\mathsf{f})$ cannot be in $\boldsymbol{q}^{a-1}$, as there is no room for that. So $h(\mathsf{f})$ is in $\boldsymbol{q}^a$, and so depth of $\boldsymbol{q}_{\mathsf{t}}$ > depth of $\boldsymbol{q}_{\mathsf{f}}$, a contradiction.
– $h(\mathsf{m}) = \mathsf{f}^a$.
This is similar to the previous case.

It follows that either $\mathsf{m}^a \prec_{\boldsymbol{q}^a} h(\mathsf{m}) \prec_{\boldsymbol{q}^a} \mathsf{t}^a$ or $\mathsf{m}^a \preceq_{\boldsymbol{q}^a} h(\mathsf{m}) \prec_{\boldsymbol{q}^a} \mathsf{f}^a$. Consider first the case when $\mathsf{m}^a \prec_{\boldsymbol{q}^a} h(\mathsf{m}) \prec_{\boldsymbol{q}^a} \mathsf{t}^a$. First, we track the location of $h(\mathsf{t})$.

– $h(\mathsf{t})$ is in $\boldsymbol{q}^a$ and $(h(\mathsf{t}), \mathsf{t}^a)$ is $\prec_{\boldsymbol{q}^a}$-incomparable.
Then let u be such that $u^a = \inf(h(\mathsf{t}), \mathsf{t}^a)$. As $u^a \prec_{\boldsymbol{q}^a} h(u) \prec_{\boldsymbol{q}^a} h(\mathsf{t})$, $\boldsymbol{q}_u$ is mapped by h into $\boldsymbol{q}^a_{h(u)}$, but there is no room for that.
– $h(\mathsf{t})$ is in $\boldsymbol{q}^a$, and $\mathsf{t}^a \prec_{\boldsymbol{q}^a} h(\mathsf{t})$.
Then $\boldsymbol{q}_{\mathsf{t}}$ is mapped by h into $\boldsymbol{q}^a_{h(\mathsf{t})}$, but there is no room for that.

Therefore, it follows that $h(\mathsf{t})$ is in $\boldsymbol{q}^{a-1}$ with $\mathsf{f}^{a-1} \prec_{\boldsymbol{q}^{a-1}} h(\mathsf{t})$. Thus,

$$\text{depth of } \boldsymbol{q}_{\mathsf{f}} \ > \text{depth of } \boldsymbol{q}_{\mathsf{t}}. \tag{13}$$

We claim that

$$h(\mathsf{f}) = \mathsf{t}^a, \tag{14}$$

as required in item (1) of Claim 7.1. Indeed, suppose $h(\mathsf{f}) \neq \mathsf{t}^a$. Then

$$h(\mathsf{f}) \text{ is an } FT\text{-twin}, \tag{15}$$

as there are no $\prec$-comparable solitary T- and F-nodes, and (t, f) is of minimal distance by assumption. We cannot have that $h(\mathsf{f}) \in \boldsymbol{q}^{a-1}$ with $\mathsf{f}^{a-1} \prec_{\boldsymbol{q}^{a-1}} h(\mathsf{f})$, as then there is no room for $h(\boldsymbol{q}_{\mathsf{f}})$. We cannot have that $h(\mathsf{f}) \in \boldsymbol{q}^a_{\mathsf{t}^a}$ by (13). Thus, there is x such that $x^a = \inf\big(\mathsf{t}^a, h(\mathsf{f})\big)$ and $\mathsf{m}^a \preceq_{\boldsymbol{q}^a} x^a \prec_{\boldsymbol{q}^a} \mathsf{t}^a$. We will show that this case is not possible either. We define a function $g_{\leftarrow} \colon h(\boldsymbol{q}) \to h(\boldsymbol{q})$ by taking $g_{\leftarrow}(x) = h\big(\iota^{\ell}(x)\big)$ whenever x is a node in $\boldsymbol{q}^{\ell}$, where we consider the contact $\mathsf{c} = \mathsf{t}^a = \mathsf{f}^{a-1}$ as a node in $\boldsymbol{q}^{a-1}$, that is, $g_{\leftarrow}(\mathsf{c}) = h\big(\iota^{a-1}(\mathsf{f}^{a-1})\big) = h(\mathsf{f})$ (the definition of $g_{\leftarrow}(\mathsf{c}')$ when the other contact c' is in $h(\boldsymbol{q})$ does not matter). Let g be the restriction of $g_{\leftarrow}$ to $\boldsymbol{q}^a_{x^a} \cup \boldsymbol{q}^{a-1}_{\mathsf{f}^{a-1}}$. We will use the following obvious 'shift' property of g: for every $\ell \in \{a, a-1\}$,

$$\text{if } y, z \text{ are both in the same copy } \boldsymbol{q}^{\ell}, z \neq \mathsf{t}^a, \text{ and } y \preceq_{\boldsymbol{q}^{\ell}} z,$$
$$\text{then } g(y) \preceq_{h(\boldsymbol{q})} g(z) \text{ and } \delta_{\boldsymbol{q}^{\ell}}(y, z) \ = \ \delta_{h(\boldsymbol{q})}\big(g(y), g(z)\big). \tag{16}$$

Now it is not hard to see that $g\big(\boldsymbol{q}^a_{x^a} \cup \boldsymbol{q}^{a-1}_{\mathsf{f}^{a-1}}\big) \subseteq \boldsymbol{q}^a_{x^a} \cup \boldsymbol{q}^{a-1}_{\mathsf{f}^{a-1}}$. Moreover, using (16) one can also show that, for every FT-twin ϑ,

$$\text{if } \vartheta^a, h(\vartheta) \in \boldsymbol{q}^a_{x^a}, \text{ then } \delta_{\boldsymbol{q}^a}(x^a, \vartheta^a) < \delta_{\boldsymbol{q}^a}\big(x^a, h(\vartheta)\big), \text{ and} \tag{17}$$
$$\text{if } \vartheta^{a-1}, h(\vartheta) \in \boldsymbol{q}^{a-1}_{\mathsf{f}^{a-1}},$$
$$\text{then } \delta_{\boldsymbol{q}^{a-1}}(\mathsf{f}^{a-1}, \vartheta^{a-1}) > \delta_{\boldsymbol{q}^{a-1}}\big(\mathsf{f}^a, h(\vartheta)\big). \tag{18}$$

Now let $X = \{\vartheta^{\ell} \in \boldsymbol{q}^a_{x^a} \cup \boldsymbol{q}^{a-1}_{\mathsf{f}^{a-1}} \mid \vartheta \text{ is an } FT\text{-twin}\}$. Then the restriction $g|_X$ of g to X is an $X \to X$ function. As X is finite, there exists a 'fixpoint' of $g\,|_X$: a node ϑ^ℓ in X and a number $N > 0$ such that $(g\,|_X)^N(\vartheta^\ell) = g^N(\vartheta^\ell) = \vartheta^\ell$. Now it follows from (17) and (18) that $N > 1$ and there is some $j \le N$ with $g^j(\vartheta^\ell) \in \boldsymbol{q}^{a-1}$. Let $j \le N$ be such that the distance $\delta_{\boldsymbol{q}^{a-1}}\big(\mathsf{f}^a, g^j(\vartheta^\ell)\big)$ is minimal, that is, $g^{j+1}(\vartheta^\ell) \notin \boldsymbol{q}^{a-1}$, and let $\eta^{a-1} = g^j(\vartheta^\ell)\big)$. We 'shift up' the fixpoint-cycle with the distance $\delta_{\boldsymbol{q}^{a-1}}\big(\mathsf{f}^a, g^j(\vartheta^\ell)\big)$: By (16), we have $g^N(\mathsf{f}^{a-1}) = \mathsf{f}^{a-1}$ and $g(\mathsf{f}^{a-1}) = h(\mathsf{f})$ is in $\boldsymbol{q}^a$. By (15), $h(\mathsf{f})$ is an FT-twin, and so $g^N(\mathsf{f}^{a-1}) = g^{N-1}\big(g(\mathsf{f}^{a-1})\big)$ is an FT-twin as well. But f^{a-1} is a contact, and so it cannot be both in $F^{\mathcal{I}}$ and $T^{\mathcal{I}}$, a contradiction, proving (14).

The case when $\mathsf{m}^a \prec_{\boldsymbol{q}^a} h(\mathsf{m}) \prec_{\boldsymbol{q}^a} \mathsf{f}^a$ is similar, and it follows that $h(\mathsf{f})$ is in $\boldsymbol{q}^{a+1}$ with $\mathsf{t}^{a+1} \prec_{\boldsymbol{q}^{a+1}} h(\mathsf{f})$, and $h(\mathsf{t}) = \mathsf{f}^a$, as required in item (2) of Claim 7.1.

So suppose that $h(\mathsf{m}) = \mathsf{m}^a$. It is easy to see that $h(\mathsf{t})$ cannot be such that it is in $\boldsymbol{q}^a$ but both $(h(\mathsf{t}), \mathsf{t}^a)$ and $(h(\mathsf{t}), \mathsf{f}^a)$ are $\prec_{\boldsymbol{q}^a}$-incomparable (as otherwise $\boldsymbol{q}$ would not be minimal). Also, $h(\mathsf{t})$ cannot be in $\boldsymbol{q}^{a+1} \setminus \boldsymbol{q}^a$, as there is no room for the h-image of $\boldsymbol{q}_\mathsf{t}$ there. Similarly, $h(\mathsf{f})$ cannot be such that it is in $\boldsymbol{q}^a$ but both $(h(\mathsf{f}), \mathsf{t}^a)$ and $(h(\mathsf{f}), \mathsf{f}^a)$ are $\prec_{\boldsymbol{q}^a}$-incomparable. Also, $h(\mathsf{f})$ cannot be in $\boldsymbol{q}^{a-1} \setminus \boldsymbol{q}^a$, as there is no room for the h-image of $\boldsymbol{q}_\mathsf{f}$ there. It also follows that $\delta(\mathsf{m}, \mathsf{t}) \neq \delta(\mathsf{m}, \mathsf{f})$ (as either both contacts are in $T^{\mathcal{I}}$ or both are in $F^{\mathcal{I}}$, and so $h(\mathsf{t})$ and $h(\mathsf{f})$ cannot both be contacts).

Thus, both $h(\mathsf{t})$ and $h(\mathsf{f})$ are in $\boldsymbol{q}^a$, either $(h(\mathsf{t}), \mathsf{t}^a)$ or $(h(\mathsf{t}), \mathsf{f}^a)$ is $\prec_{\boldsymbol{q}^a}$-comparable, and either $(h(\mathsf{f}), \mathsf{t}^a)$ or $(h(\mathsf{f}), \mathsf{f}^a)$ is $\prec_{\boldsymbol{q}^a}$-comparable.

Next, we show that if $(h(\mathsf{t}), \mathsf{f}^a)$ is $\prec_{\boldsymbol{q}^a}$-comparable, then $h(\mathsf{t}) \prec_{\boldsymbol{q}^a} \mathsf{f}^a$. Indeed, suppose on the contrary that $\mathsf{f}^a \prec_{\boldsymbol{q}^q} h(\mathsf{t})$. Let x be such that $\mathsf{m} \prec x \prec \mathsf{t}$ and $h(x) = \mathsf{f}^a$. If x is not labelled by T in $\boldsymbol{q}$ then h maps a proper subCQ of $\boldsymbol{q}$ to $\boldsymbol{q}$, contradicting the minimality of $\boldsymbol{q}$. If x is labelled by T then $\mathsf{f}^a \in T^{\mathcal{I}}$ should hold. Thus, $h(\mathsf{f}) \neq \mathsf{f}^a$, and so $h(\mathsf{f}) = x$. Therefore, x must be labelled by F too, so it is an FT-twin. But then $h(x)$ cannot be a contact.

Similarly, it can be shown that if $(h(\mathsf{f}), \mathsf{t}^a)$ is $\prec_{\boldsymbol{q}^a}$-comparable, then $h(\mathsf{f}) \prec_{\boldsymbol{q}^a} \mathsf{t}^a$.

Now, suppose first that $(h(\mathsf{t}), \mathsf{f}^a)$ is $\prec_{\boldsymbol{q}^a}$-comparable. Then it follows that $(h(\mathsf{f}), \mathsf{f}^a)$ is $\prec_{\boldsymbol{q}^a}$-comparable, and so $h(\mathsf{f}) = \mathsf{f}^a$, that is, item 3. of Claim 7.1 holds. On the other hand, if $(h(\mathsf{t}), \mathsf{f}^a)$ is $\prec_{\boldsymbol{q}^a}$-incomparable, then $(h(\mathsf{t}), \mathsf{t}^a)$ is $\prec_{\boldsymbol{q}^a}$-comparable, and so $h(\mathsf{t}) = \mathsf{t}^a$. Thus, $(h(\mathsf{f}), \mathsf{f}^a)$ is $\prec_{\boldsymbol{q}^a}$-incomparable, and so $(h(\mathsf{f}), \mathsf{t}^a)$ is $\prec_{\boldsymbol{q}^a}$-comparable. Therefore, $h(\mathsf{f}) \prec_{\boldsymbol{q}^a} \mathsf{t}^a$ holds, as required in item 4. of Claim 7.1. ❑

Now we can complete the proof of Theorem 7 (*ii*) by observing that none of the cases in Claim 7.1 is possible, whenever $\boldsymbol{q}$ contains neither FT-twins nor $\prec$-comparable solitary pairs.

D PROOF OF CLAIM 9.2

We prove the following:

Claim 9.2 *The d-sirup $(\Delta_{\boldsymbol{q}}, G)$ is FO-rewritable iff, for any periodic structure $\mathfrak{P} = (\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{E})$ with $\boldsymbol{P} \neq \emptyset$, one of the following holds:*

(h1) *there is a homomorphism from some cactus to the $\bar{\cdot}$-closure of the acyclic version of $\boldsymbol{B} \cup \boldsymbol{P}$;*

(h2) *there is a homomorphism from the root segment of some cactus to $\bar{\boldsymbol{P}}$;*

(h3) *there is a homomorphism from the root segment of one of the $\bar{\mathfrak{H}}_v$ to $\bar{\boldsymbol{E}}$.*

We say that a skeleton C^s *fits* a periodic structure $\mathfrak{P} = (\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{E})$ if its χ_C-image in $\mathfrak{G}$ is such that each branch of C^s consists of two consecutive parts: the first one is mapped by χ_C into $\mathfrak{H}$ (= $\boldsymbol{B} \cup \boldsymbol{P}$) and the second one into some $\mathfrak{H}_v$ in $\boldsymbol{E}$, for $v \in R$. If C^s fits $\mathfrak{P}$, we denote by $d(C^s, \mathfrak{P})$ the minimum depth of the nodes in C^s that belong to the second part of these branches. Note that, for any periodic structure $\mathfrak{P}$ with non-empty $\boldsymbol{P}$ and any $d < \omega$, there is a cactus C such that C^s fits $\mathfrak{P}$ and $d(C^s, \mathfrak{P}) > d$. (If the source of $\mathfrak{H}$ is $(\emptyset, 0, C)$, then in the root segment of C, we bud $T(y_j)$ iff $j \in C$. We continue budding as prescribed by $\mathfrak{H}$, using the cycles in $\boldsymbol{P}$ to make sure that $d(C^s, \mathfrak{P}) > d$, and then acyclic $\mathfrak{H}_v$, which end in leaf types that give rise to leaf segments in C with all of the $T(y_i)$ unbudded.)

($\Rightarrow$) Suppose $d < \omega$ is such that every cactus contains a homomorphic image of some cactus of depth $\le d$. Let $\mathfrak{P} = (\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{E})$ be a periodic structure with $\boldsymbol{P} \neq \emptyset$ and let C^s fit $\mathfrak{P}$ with 'sufficiently large' $d(C^s, \mathfrak{P})$. Take a minimal cactus C' of depth $\le d$, for which there is a homomorphism $h' \colon C' \to C$. Set $\bar{h} = \bar{\chi}_C h'$ and let $\mathfrak{s}'$ be the root segment in C'.

Case **(h1)**: $\bar{h}(\mathfrak{s}') \cap \bar{\boldsymbol{B}} \neq \emptyset$. Then, since $d(C^s, \mathfrak{P})$ is sufficiently large, $\bar{h}(C') \subseteq \bar{\boldsymbol{B}} \cup \bar{\boldsymbol{P}}$. If $\bar{h}(C')$ goes for more than one cycle in $\bar{\boldsymbol{B}} \cup \bar{\boldsymbol{P}}$, we could cut out one cycle and obtain a homomorphism of a smaller cactus C'' into C, contrary to our assumption. This gives us a homomorphism from a small depth cactus to the $\bar{\cdot}$-closure of the acyclic version of $\boldsymbol{B} \cup \boldsymbol{P}$.

Case **(h2)**: $\bar{h}(\mathfrak{s}') \cap \bar{\boldsymbol{B}} = \emptyset$ and $\bar{h}(\mathfrak{s}') \cap \bar{\boldsymbol{P}} \neq \emptyset$. If $\bar{h}(\mathfrak{s}') \not\subseteq \bar{\boldsymbol{P}}$, then using the fact that $\mathfrak{H}$ is realisable, we can modify $\bar{h}$ and obtain the required homomorphism from the root segment $\mathfrak{s}'$ to $\bar{\boldsymbol{P}}$.

Case **(h3)**: if neither of the previous two cases holds, then we have $\bar{h}(\mathfrak{s}') \subseteq \bar{\mathfrak{H}}_v$, for some $\mathfrak{H}_v$ in $\boldsymbol{E}$ and $v \in R$.

($\Leftarrow$) Consider an arbitrary cactus C, its skeleton C^s and the canonical homomorphism χ_C. Our aim is to define a periodic structure $\mathfrak{P} = (\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{E})$, using which we could construct a 'small' cactus that is homomorphically embeddable into C.

We start from the root and move along each branch of C^s and the χ_C-images of its nodes in $\mathfrak{G}$ until the same type repeats twice, that is, we visit the same node twice in $\mathfrak{G}$. This will give us the pre-periodic $\boldsymbol{B}$ and periodic parts $\boldsymbol{P}$ of $\mathfrak{P}$. We next associate with C^s a certain $\boldsymbol{E}$. For each branch of C^s, consider the node after which the χ_C-image of the branch leaves $\boldsymbol{P}$. Let R be the set of all of such nodes in $\mathfrak{G}$. For each $v \in R$, pick some branch in C^s whose χ_C-image leaves $\boldsymbol{P}$ at v. Let p be the node on this branch such that $\chi_C(p) = v \in \boldsymbol{P}$ but the χ_C-image of the child of p on this branch is not in $\boldsymbol{P}$. Consider the χ_C-image in $\mathfrak{G}$ of the subtree C^s_p of C^s with root p. Cut out from C^s_p the segments between repeating types, if any, so that the χ_C-image $\mathfrak{H}_p$ of the remaining part is acyclic. The constructed pairs $(p, \mathfrak{H}_p)$ form the post-periodic part $\boldsymbol{E}$. It follows from the construction that the resulting $\mathfrak{P} = (\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{E})$ is a periodic structure.

- If **(h1)** is satisfied, we are done because some cactus can be homomorphically embedded into the $\bar{\cdot}$-closure of the acyclic version of $\boldsymbol{B} \cup \boldsymbol{P}$, and so into some initial part of C.
- If **(h2)** holds, then there is a homomorphism from some root segment $\mathfrak{s}$ into a segment in $\bar{\boldsymbol{P}}$. This gives a homomorphism from $\mathfrak{s}$ into C, with the image of the root of $\mathfrak{s}$ being in a non-root segment $\mathfrak{s}'$ in C. Using the fact that $\boldsymbol{P}$ is periodic,

we extract from C a cactus, starting from $\mathfrak{s}'$, of smaller depth that is homomorphically embeddable into C, to which we apply the same argument.

- If **(h3)** holds, we are done by the definition of $\mathfrak{H}_v$.

E PROOF OF CLAIM 9.3

We prove the following:

Claim 9.3. *If none of conditions* **(h1)**–**(h3)** *holds, then evaluating* $(\Delta_{\boldsymbol{q}}, G)$ *is L-hard.*

Let $\mathfrak{P} = (\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{E})$ be a periodic structure. Consider the following additional condition:

(h4) there is a homomorphism from the leaf segment to $\bar{\boldsymbol{B}} \cup \bar{\boldsymbol{P}}$.

We claim that

$$\text{if } \mathfrak{P} = (\boldsymbol{B}, \boldsymbol{P}, \boldsymbol{E}) \text{ is periodic structure with } \boldsymbol{P} \neq \emptyset \text{ for which none of } \textbf{(h1)}\text{–}\textbf{(h3)} \text{ holds, then there is a periodic structure } \mathfrak{P}' \text{ such that none of } \textbf{(h1)}\text{–}\textbf{(h4)} \text{ holds for it.} \tag{19}$$

Indeed, let $\mathfrak{P}$ be a minimal periodic structure, for which none of **(h1)**–**(h3)** holds. Suppose there is a homomorphism h from the leaf segment $\mathfrak{s}$ into $\bar{\boldsymbol{B}} \cup \bar{\boldsymbol{P}}$. Let a be the A-node (focus) in $\mathfrak{s}$. Then $h(a)$ is also an A-node. It corresponds to an edge $(\mathfrak{s}', \mathfrak{s}'')$ in $\mathfrak{P}$. We cut this edge, remove from $\mathfrak{P}$ those segments that are only reachable from $\mathfrak{s}''$, and change $\mathfrak{s}'$ to the segment obtained by replacing the label A on $h(a)$ in $\mathfrak{s}'$ with T. The resulting $\mathfrak{P}'$ is a periodic structure. It is not hard to see that it satisfies none of **(h1)**–**(h3)**, contrary to the minimality of $\mathfrak{P}$, which proves (19).

We are now in a position to show L-hardness of evaluating $(\Delta_{\boldsymbol{q}}, G)$ with a periodic structure, for which none of **(h1)**–**(h4)** holds. The proof is by reduction of undirected reachability. Suppose we are given a graph G, two vertices s and t in it, and we want to decide whether t is reachable from s.

Let $\mathfrak{P}$ be a periodic structure, for which none of **(h1)**–**(h4)** holds. Consider its blow-up $\bar{\mathfrak{P}}$. Replace each node v in G by a fresh copy $\bar{\boldsymbol{P}}_v$ of $\bar{\boldsymbol{P}}$ in $\mathfrak{P}$. For any two connected nodes u and v, if there is a directed edge between nodes α and β in $\bar{\boldsymbol{P}}$, we add the same edge between α_u and β_v, and also between α_v and β_u. We additionally attach $\bar{\boldsymbol{B}}$ to $\bar{\boldsymbol{P}}_s$ and $\bar{\boldsymbol{E}}$ to $\bar{\boldsymbol{P}}_t$ in the same way as in $\mathfrak{P}$. We regard the resulting structure as a data instance $\mathcal{D}$ and show that there is a path from s to t in G iff the answer to $(\Delta_{\boldsymbol{q}}, G)$ over $\mathcal{D}$ is 'yes'.

($\Rightarrow$) Suppose there is a path from s to t. Consider a sufficiently large cactus C with the periodic structure $\mathfrak{P}$. Let h be a homomorphism from C to $\mathcal{D}$, which canonically maps the pre-periodic part of C to $\bar{\boldsymbol{B}}$ and then, having entered $\bar{\boldsymbol{P}}$, it follows the path from s to t in G via the edges of the form (α_u, β_v) and uses $\bar{\boldsymbol{E}}$ at t. The cactus C should be large enough to reach t in this way. Now, taking an arbitrary assignment of T and F to A, we see that we always have a segment in $\mathcal{D}$ (under this assignment) isomorphic to $\boldsymbol{q}$.

($\Leftarrow$) Suppose now there is no path from s to t in G. Denote by V_s the set of nodes reachable from s and by V_t the remainder in G (so there are no edges between V_s and V_t). Consider an assignment under which all A-labels in V_s become F and A in V_t become T. We show that there is no homomorphism from $\boldsymbol{q}$ to the resulting data instance.

Suppose there is $h\colon \boldsymbol{q} \to V_s$. This means that there was a homomorphism from either $\boldsymbol{q}$, or the leaf segment into the data instance before the assignment. Both of these cases are impossible by (19).

Suppose there is $h\colon \boldsymbol{q} \to V_t$. This means that there was a homomorphism from either $\boldsymbol{q}$, or a root segment into the data instance before the assignment, which is again a contradiction with (19).

F COMPLEXITY OF CHECKING THE CRITERION OF CLAIM 9.2

Lemma 12. *For Λ-CQs of span k, one can check the criterion of Claim 9.2 in time $p(|\boldsymbol{q}|)2^{p'(k)}$, for some polynomials p and p'. Thus, deciding FO-rewritability of d-sirups with a Λ-CQ is fixed-parameter tractable, if the CQ's span is regarded as a parameter.*

Proof. We slightly modify the criterion of Claim 9.2. Suppose we have a periodic structure $\mathfrak{P}$ and a homomorphism from a cactus C into $\bar{\mathfrak{P}}$. We say that this homomorphism is *anchored* if the image of the root segment in C includes some A-node of the root segment in $\mathfrak{P}$. We also use the same notion for a homomorphism from a root segment: it is *anchored* if its image includes some A-node of the root segment in $\mathfrak{P}$.

The modified criterion is as follows: $(\Delta_{\boldsymbol{q}}, G)$ is FO-rewritable iff, for any periodic structure $\mathfrak{P}$, there is either (i) an unanchored homomorphism from a root segment to $\bar{\mathfrak{P}}$, or (ii) an anchored homomorphism from an acyclic cactus to $\bar{\mathfrak{P}}$. The proof of this criterion remains essentially the same as the proof of Claim 9.2. Our aim is to show that it can be checked in time $p(|\boldsymbol{q}|)2^{p'(k)}$.

As well-known (see, e.g., [24] and references therein), we can check whether there is a homomorphism from a segment into a type in polynomial time. Clearly, there are $2^{O(k)}$ nodes in the digraph $\mathfrak{G}$.

We call a node (type) t in $\mathfrak{G}$ *black*, if we can homomorphically map some root segment into the blow-up $\bar{t}$ of t. We can find all black nodes in $\mathfrak{G}$ in time $p(|\boldsymbol{q}|)2^{p'(k)}$.

We call a node v *blue* if, for every $\mathfrak{H}_v$ starting from it, there is a homomorphism of a root segment into $\bar{\mathfrak{H}}_v$. We can find all blue nodes in time polynomial in the size of $\mathfrak{G}$. Indeed, consider a game between two players. They move a pebble over the nodes of $\mathfrak{G}$. In each node, the first player picks one outgoing edge for each possible label. The second player picks one of these edges and the players proceed along this edge to the next node. Black positions are winning for the second player. Non-black leaf-types are winning for the first player. It is easy to see that, for a given v, there is $\bar{\mathfrak{H}}_v$ with no root segment embeddings iff the node v is winning for the first player ($\mathfrak{H}_v$ describes a winning strategy). So, to check whether a node is blue, it is enough to check the winner in this node.

If a periodic structure has a black or a blue node, which is not a child of the root, case (i) holds for this structure. Now, we need to check whether every acyclic structure, for which there are no such nodes, satisfies (ii).

By definition, edges in $\mathfrak{G}$ correspond to (budded) A-nodes in $\bar{\mathfrak{G}}$. For an edge (u, v) in $\mathfrak{G}$, we say that we can *cut it at depth d* if, for every periodic structure starting from this edge (more precisely, from the corresponding A node), there is a focused homomorphism (a homomorphism is focused if the focus of a cactus, with F renamed to A, is mapped into the focus of the periodic structure) into it from a cactus $C^{F\to A}$ of depth at most d. The image of $C^{F\to A}$ here is allowed to use any nodes of $\bar{u}$ save its focus. The property 'to be cuttable at depth d' can be checked by recursion on d. For $d = 1$, it just means that there is a homomorphism of a leaf segment into u or v using the A-node corresponding to (u, v). For the recursion step, we need to try all possible extensions of v, not using coloured nodes, and to check all possible homomorphisms of a segment in u or v that map the focus node of a segment into (u, v) and that does not use the focus node of u. If, for any extension of v, there is a homomorphism of a segment that, apart from (u, v), uses only A nodes cuttable at depth $d - 1$, then (u, v) is cuttable at depth d. We keep finding all nodes that can be cut at depth $d = 1, 2, 3, \dots$ until we find d, for which there are no new such nodes. Fix this d. The operation described above can be done in time $p(|\boldsymbol{q}|)2^{p'(k)}$.

Next, for each root segment in $\tilde{\mathfrak{G}}$, we consider all of its neighbourhoods of depth 1 and check whether there is a homomorphism of a root segment in such a neighbourhood such that all A-nodes used by it can be cut at depth d. If this is the case, then in any periodic structure, in which there is no unanchored homomorphism of a root segment, there is an anchored homomorphism of a cactus of depth d (that can be made acyclic in the usual way).

Otherwise, we claim that there is a periodic structure for which neither (i) nor (ii) holds. To show this, we start from a root and its depth one neighbourhood that cannot be cut at depth d and, in each A-node that cannot be cut at depth d, we proceed in such a way that one of the next nodes cannot be cut at depth d. Suppose there is a homomorphism of a cactus of depth into the resulting periodic structure. Consider the smallest cactus with this property. Take an A-node of depth $d + 1$ in this cactus. It is mapped into some edge in our periodic structure. Due to the minimality of the cactus, this A-node cannot be cut at depth d. However, all of its children in the cactus can be cut at depth d, which is a contradiction. ❑

G PROOF OF THEOREM 11

We prove the following:

Theorem 11. *For any a ditree CQ $\boldsymbol{q}$ with one solitary F and one solitary T, $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ is either FO-rewritable, or L-complete, or NL-complete. Deciding this trichotomy can be done in polynomial time.*

First, we show that if $\boldsymbol{q}$ is a ditree CQ with a single solitary F-node f and a single solitary T-node t, and $\boldsymbol{q}$ is quasi-symmetric, then $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ is L-hard. The proof is by an FO-reduction of the L-complete reachability problem for undirected graphs. Given an undirected graph $G = (V, E)$ with nodes $\mathfrak{s}, \mathfrak{t} \in V$, we construct a data instance $\mathcal{D}_G$ as follows. We replace each $e = (\mathfrak{u}, \mathfrak{v}) \in E$ by a fresh copy $\boldsymbol{q}^e$ of $\boldsymbol{q}$ such that node t^e in $\boldsymbol{q}^e$ is renamed to $\mathfrak{u}$ with $T(\mathfrak{u})$ replaced by $A(\mathfrak{u})$, and node f^e is renamed to $\mathfrak{v}$ with $F(\mathfrak{v})$ replaced by $A(\mathfrak{v})$. Then $\mathcal{D}_G$ comprises all such $\boldsymbol{q}^e$, for $e \in E$, as well as $T(\mathfrak{s})$ and $F(\mathfrak{t})$. We show that $\mathfrak{s} \to_G \mathfrak{t}$ iff the certain answer to $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ over $\mathcal{D}_G$ is 'yes'.

$(\Rightarrow)$ Suppose there is a path $\mathfrak{s} = \mathfrak{v}_0, \dots, \mathfrak{v}_n = \mathfrak{t}$ in G, in which $e_i = (\mathfrak{v}_i, \mathfrak{v}_{i+1}) \in E$, for $i < n$. Then, for any model $\mathcal{I}$ of $\Delta_{\boldsymbol{q}}$ and $\mathcal{D}_G$, there is some $i < n$ such that $\mathcal{I} \models T(\mathfrak{v}_i)$ and $\mathcal{I} \models F(\mathfrak{v}_{i+1})$. Thus, the identity map from $\boldsymbol{q}$ to its copy $\boldsymbol{q}^{e_i}$ is a $\boldsymbol{q} \to \mathcal{I}$ homomorphism, as required.

$(\Leftarrow)$ Suppose $\mathfrak{s} \not\to_G \mathfrak{t}$. Define a model $\mathcal{I}$ of $\Delta_{\boldsymbol{q}}$ and $\mathcal{D}_G$ by labelling with T the A-nodes in $\mathcal{D}_G$ that (as nodes in G) are reachable from $\mathfrak{s}$ (via an undirected path in G) and with F the remaining ones. We claim that there is no homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. Indeed, suppose to the contrary that such h exists. Consider the sub-structure $\mathcal{H}^{(\mathsf{t},\mathsf{f})}$ of $\mathcal{D}_G$ comprising those copies of $\boldsymbol{q}$ that have a non-empty intersection with $h(\boldsymbol{q})$. As $\boldsymbol{q}$ is quasi-symmetric, (t, f) is $\prec$-incomparable, and so $\mathcal{H}^{(\mathsf{t},\mathsf{f})}$ looks like in the proof of Theorem 7 (ii). Thus, we have the four cases of Claim 7.1. It follows that $\delta(\mathsf{m}, \mathsf{f}) \neq \delta(\mathsf{m}, \mathsf{t})$. On the other hand, as $\boldsymbol{q}$ is quasi-symmetric and (t, f) is its only solitary pair, (t, f) is symmetric. Thus, we must have $\delta(\mathsf{m}, \mathsf{f}) = \delta(\mathsf{m}, \mathsf{t})$, which is a contradiction.

Next, suppose that (t, f) is not $\prec$-comparable and $\boldsymbol{q}$ is not quasi-symmetric. We consider two models $\mathcal{I}$ over the structure $\mathcal{H}^{(\mathsf{t},\mathsf{f})}$ (defined in the proof of Theorem 7 (ii)): one has both contacts in $F^{\mathcal{I}}$, the other in $T^{\mathcal{I}}$. We check whether there exists a homomorphism from $\boldsymbol{q}$ to either of these models: If neither, then $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ is NL-hard by the proof of Theorem 7 (ii). If at least one of them is possible, then we show that $(\Delta_{\boldsymbol{q}}, \boldsymbol{G})$ is FO-rewritable: We will use the homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$ to define homomorphisms from some depth ≤ 2 cactus to any larger cactus, and then apply the criterion of Prop. 2.

By Claim 7.1, we have four cases for h. When $\boldsymbol{q}$ contains a single solitary F- and a single solitary T-node, then case (2) is a 'symmetrical' version of case (1), and case (4) is a 'symmetrical' version of case (3). So below we deal with cases (1) and (3) only. In each of these cases, we claim that

$$\text{for any } n > 2, \text{ either there is a homomorphism } g\colon C_1 \to C_n, \quad \text{or there is a homomorphism } g\colon C_2 \to C_n, \tag{20}$$

where C_n is the unpruned $\boldsymbol{q}$-cactus (budded at the T-node) whose skeleton is of depth n.

Indeed, we define g in each of the cases. Suppose that the segments of C_n are $\mathfrak{s}_0^n$ (root), $\dots, \mathfrak{s}_n^n$ (leaf).

(1) There are two cases, depending on whether $h(\boldsymbol{q})$ does not intersect with $\boldsymbol{q}^{a+1} - \{\mathsf{f}^a\}$ (like in $\boldsymbol{q}_7$ above), or it does (like in $\boldsymbol{q}_8$ of Example 5). In the former case, we can map C_1 to any larger cactus, while in the latter case only C_2. The map $g\colon C_i \to C_n$ for $i = 1, 2$, is defined as follows: Each of the non-leaf segments of C_i is mapped to the same segment in C_n by its identity map, and the remaining points in $\mathfrak{s}_i^i$ are mapped by taking

$$g(z) = \begin{cases} \iota^a\big(h(z)\big) \text{ in } \mathfrak{s}_{i-1}^n, & \text{if } h(z) \in \boldsymbol{q}^a, \\ \iota^{a-1}\big(h(z)\big) \text{ in } \mathfrak{s}_i^n, & \text{if } h(z) \in \boldsymbol{q}^{a-1}, \\ \iota^{a+1}\big(h(z)\big) \text{ in } \mathfrak{s}_{i-2}^n, & \text{if } i = 2 \text{ and } h(z) \in \boldsymbol{q}^{a+1}. \end{cases}$$

It is straightforward to check that g is well-defined and it is a homomorphism.

(3) We can map C_1 to any larger cactus by the map $g\colon C_1 \to C_n$ defined as follows: The root segment $\mathfrak{s}_0^1$ of C_1 is mapped to the root segment $\mathfrak{s}_0^n$ of C_n by its identity map, and the remaining points in $\mathfrak{s}_1^1$ are mapped by taking

$$g(z) = \begin{cases} \iota^a\big(h(z)\big) \text{ in } \mathfrak{s}_1^n, & \text{if } h(z) \in \boldsymbol{q}^a, \\ \iota^{a-1}\big(h(z)\big) \text{ in } \mathfrak{s}_2^n, & \text{if } h(z) \in \boldsymbol{q}^{a-1}, \\ \iota^{a+1}\big(h(z)\big) \text{ in } \mathfrak{s}_0^n, & \text{if } h(z) \in \boldsymbol{q}^{a+1}. \end{cases}$$

Again, it is straightforward to check that g is well-defined and it is a homomorphism.