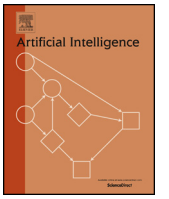
Artificial Intelligence 309 (2022) 103738

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



A tetrachotomy of ontology-mediated queries with a covering axiom

Olga Gerasimova [a], Stanislav Kikot [b], Agi Kurucz [c,*], Vladimir Podolskii [d,a], Michael Zakharyaschev [e]

[a] HSE University, Moscow, Russia
[b] Institute for Information Transmission Problems, Moscow, Russia
[c] Department of Informatics, King's College London, UK
[d] Steklov Mathematical Institute, Moscow, Russia
[e] Department of Computer Science and Information Systems, Birkbeck, University of London, UK

ARTICLE INFO

Article history:
Received 19 July 2020
Received in revised form 4 May 2022
Accepted 5 May 2022
Available online 13 May 2022

Keywords:
Ontology-mediated query
Description logic
Datalog
Disjunctive datalog
First-order rewritability
Data complexity

ABSTRACT

Our concern is the problem of efficiently determining the data complexity of answering queries mediated by description logic ontologies and constructing their optimal rewritings to standard database queries. Originated in ontology-based data access and datalog optimisation, this problem is known to be computationally very complex in general, with no explicit syntactic characterisations available. In this article, aiming to understand the fundamental roots of this difficulty, we strip the problem to the bare bones and focus on Boolean conjunctive queries mediated by a simple covering axiom stating that one class is covered by the union of two other classes. We show that, on the one hand, these rudimentary ontology-mediated queries, called disjunctive sirups (or d-sirups), capture many features and difficulties of the general case. For example, answering d-sirups is Π_2^p-complete for combined complexity and can be in AC^0 or L-, NL-, P-, or coNP-complete for data complexity (with the problem of recognising FO-rewritability of d-sirups being 2ExpTime-hard); some d-sirups only have exponential-size resolution proofs, some only double-exponential-size positive existential FO-rewritings and single-exponential-size nonrecursive datalog rewritings. On the other hand, we prove a few partial sufficient and necessary conditions of FO- and (symmetric/linear-) datalog rewritability of d-sirups. Our main technical result is a complete and transparent syntactic AC^0/NL/P/coNP tetrachotomy of d-sirups with disjoint covering classes and a path-shaped Boolean conjunctive query. To obtain this tetrachotomy, we develop new techniques for establishing P- and coNP-hardness of answering non-Horn ontology-mediated queries as well as showing that they can be answered in NL.

* Corresponding author.
E-mail addresses: ogerasimova@hse.ru (O. Gerasimova), staskikotx@gmail.com (S. Kikot), agi.kurucz@kcl.ac.uk (A. Kurucz), podolskii@mi.ras.ru (V. Podolskii), michael@dcs.bbk.ac.uk (M. Zakharyaschev).

https://doi.org/10.1016/j.artint.2022.103738

1. Introduction

1.1. The ultimate question

The general research problem we are concerned with in this article can be formulated as follows: for any given ontology-mediated query (OMQ, for short) $\mathbf{Q} = (\mathcal{O}, \boldsymbol{q})$ with a description logic ontology $\mathcal{O}$ and a conjunctive query $\boldsymbol{q}$,

(data complexity) determine the computational complexity of answering $\mathbf{Q}$ over any input data instance $\mathcal{A}$ under the open world semantics and, if possible,

(rewritability) reduce the task of finding certain answers to $\mathbf{Q}$ over any input $\mathcal{A}$ to the task of evaluating a conventional database query $\mathbf{Q}'$ with optimal data complexity directly over $\mathcal{A}$ (the query $\mathbf{Q}'$ is then called a *rewriting* of the OMQ $\mathbf{Q}$).

Ontology-based data access Answering queries mediated by a description logic (DL) ontology has been known as an important reasoning problem in knowledge representation since the early 1990s [1]. The proliferation of DLs and their applications [2,3], the development of the (DL-underpinned) Web Ontology Language OWL,[1] and especially the paradigm of ontology-based data access (OBDA) [4–6] (proposed in the mid 2000s and recently rebranded to the virtual knowledge graph (VKG) paradigm [7]), have made theory and practice of answering ontology-mediated queries (OMQs) a hot research area lying at the crossroads of Knowledge Representation and Reasoning, Semantic Technologies and the Semantic Web, Knowledge Graphs, and Database Theory and Technologies.

In a nutshell, the idea underlying OBDA is as follows. The users of an OBDA system (such as Mastro[2] or Ontop[3]) may assume that the data they want to query is given in the form of a directed graph whose nodes are labelled with concepts (unary predicates or classes) and whose edges are labelled with roles (binary predicates or properties)—even though, in reality, the data can be physically stored in different and possibly heterogeneous data sources—hence the moniker VKG. The concept and role labels come from an ontology, designed by a domain expert, and should be familiar to the intended users who, on the other hand, do not have to know anything about the real data sources. Apart from providing a user-friendly vocabulary for queries and a high-level conceptual view of the data, an important role of the ontology is to enrich possibly incomplete data with background knowledge. To illustrate, imagine that we are interested in the life of 'scientists' and would like to satisfy our curiosity by querying the data available on the Web (it may come from the universities' databases, publishing companies, personal web pages, social networks, etc.). An ontology $\mathcal{O}$ about scientists, provided by an OBDA system, might contain the following 'axioms' (given, for readability, both as DL concept inclusions and first-order sentences):

$$\mathit{BritishScientist} \sqsubseteq \exists\, \mathit{affiliatedWith}.\mathit{UniversityInUK} \tag{1}$$

$$\forall x\, [\mathit{BritishScientist}(x) \to \exists y\, (\mathit{affiliatedWith}(x, y) \wedge \mathit{UniversityInUK}(y))]$$

$$\exists\, \mathit{worksOnProject} \sqsubseteq \mathit{Scientist} \tag{2}$$

$$\forall x\, [\exists y\, \mathit{worksOnProject}(x, y) \to \mathit{Scientist}(x)]$$

$$\mathit{Scientist} \sqcap \exists\, \mathit{affiliatedWith}.\mathit{UniversityInUK} \sqsubseteq \mathit{BritishScientist} \tag{3}$$

$$\forall x\, [(\mathit{Scientist}(x) \wedge \exists y\, (\mathit{affiliatedWith}(x, y) \wedge \mathit{UniversityInUK}(y))) \to \mathit{BritishScientist}(x)]$$

$$\mathit{BritishScientist} \sqsubseteq \mathit{Brexiteer} \sqcup \mathit{Remainer} \tag{4}$$

$$\forall x\, [\mathit{BritishScientist}(x) \to (\mathit{Brexiteer}(x) \vee \mathit{Remainer}(x))]$$

Now, to find, for example, British scientists, we could execute a simple OMQ $\mathbf{Q}(x) = (\mathcal{O}, \boldsymbol{q}(x))$ with the query

$$\boldsymbol{q}(x) \;=\; \mathit{BritishScientist}(x)$$

mediated by the ontology $\mathcal{O}$. The OBDA system is expected to return the members of the concept *BritishScientist* that are extracted from the original datasets by 'mappings' (database queries connecting the data with the ontology vocabulary and virtually populating its concepts and roles) and also deduced from the data and axioms in $\mathcal{O}$ such as (3). It is this latter reasoning task that makes OMQ answering non-trivial and potentially intractable both in practice and from the complexity-theoretic point of view.

[1] https://www.w3.org/TR/owl2-overview/.
[2] https://www.obdasystems.com.
[3] https://ontopic.biz.

Uniform approach To ensure theoretical and practical tractability, the OBDA paradigm presupposes that the users' OMQs are reformulated—or rewritten—by the OBDA system into conventional database queries over the original data sources, which have proved to be quite efficiently evaluated by the existing database management systems. Whether or not such a rewriting is possible and into which query language naturally depends on the OMQ in question. One way to *uniformly* guarantee the desired rewritability is to delimit the language for OMQ ontologies and queries. Thus, the *DL-Lite* family of description logics [5] and the *OWL 2 QL* profile[4] of *OWL 2* were designed so as to guarantee rewritability of *all* OMQs with a *DL-Lite* ontology and a conjunctive query (CQ) into first-order (FO) queries, that is, essentially SQL queries [8]. In complexity-theoretic terms, FO-rewritability of an OMQ means that it can be answered in LogTime uniform AC^0, one of the smallest complexity classes [9]. In our example above, only axioms (1) and (2) are allowed by *OWL 2 QL*. Various dialects of tuple-generating dependencies (tgds), aka datalog$^{\pm}$ or existential rules, that admit FO-rewritability and extend *OWL 2 QL* have also been identified; see, e.g., [10–13].

Any OMQ with an $\mathcal{EL}$, *OWL 2 EL* or *Horn*$\mathcal{SHIQ}$ ontology is datalog-rewritable [14–17], and so can be answered in P—polynomial time in the size of data—using various datalog engines, say GraphDB,[5] LogicBlox[6] or RDFox.[7] Axioms (1)–(3) are admitted by the $\mathcal{EL}$ syntax. On the other hand, OMQs with an $\mathcal{ALC}$ (a notational variant of the multimodal logic $\mathbf{K}_n$ [18]) ontology and a CQ are in general coNP-complete [1], and so often regarded as intractable and not suitable for OBDA, though they can be rewritten to disjunctive datalog [19–21] supported by systems such as DLV[8] or clasp.[9] For example, coNP-complete is the OMQ $(\{(4)\}, \boldsymbol{q}_1)$ with the CQ

$$\boldsymbol{q}_1 = \exists w, x, y, z\, [\mathit{Brexiteer}(w) \land \mathit{hasCoAuthor}(w, x) \land \mathit{Remainer}(x) \land \mathit{hasCoAuthor}(x, y) \land \mathit{Brexiteer}(y) \land \mathit{hasCoAuthor}(y, z) \land \mathit{Remainer}(z)]$$

(see also the representation of $\boldsymbol{q}_1$ as a labelled graph below). It might be of interest to note that by making the role *hasCoAuthor* symmetric using, for example, the role inclusion axiom

$$\mathit{hasCoAuthor} \sqsubseteq \mathit{hasCoAuthor}^- \qquad (5)$$

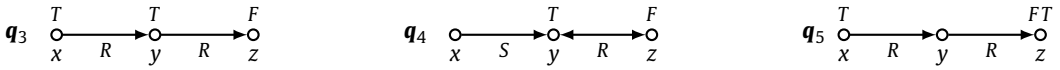
$$\forall x, y\, [\mathit{hasCoAuthor}(x, y) \to \mathit{hasCoAuthor}(y, x)]$$

we obtain the OMQ $(\{(4), (5)\}, \boldsymbol{q}_1)$, which is rewritable to a symmetric datalog query, and so can be answered by a highly parallelisable algorithm in the complexity class L (logarithmic space).

For various reasons, many existing ontologies do not comply with the restrictions imposed by the standard languages for OBDA. Notable examples include the large-scale medical ontology SNOMED CT,[10] which is mostly but not entirely in $\mathcal{EL}$, and the oil and gas NPD FactPages[11] ontology and the Subsurface Exploration Ontology [22], both of which fall outside *OWL 2 QL* by a whisker, in particular because of *covering axioms* like (4) that are quite typical in conceptual modelling. One way to (partially) resolve this issue is to compute an approximation of a given ontology within the required ontology language, which is an interesting and challenging reasoning problem by itself; see, e.g., [23–26] and references therein. In practice, the non-complying axioms are often simply omitted from the ontology in the hope that not too many answers to OMQs will be lost. An attempt to figure out whether it was indeed the case for the OMQs with the Subsurface Exploration Ontology and geologists' queries from [22] was the starting point of research that led to this article.

Non-uniform approach An ideal alternative to the uniform approach to OBDA discussed above would be to admit OMQs in a sufficiently expressive language and supply the OBDA system with an algorithm that recognises the data complexity of each given OMQ and rewrites it to a database query in the corresponding target language. For example, while answering the OMQ $(\{(4)\}, \boldsymbol{q}_1)$ is coNP-complete, we shall see later on in this paper that $(\{(4)\}, \boldsymbol{q}_2)$ with the same ontology and the CQ $\boldsymbol{q}_2$ shown in the picture below is P-complete and datalog-rewritable, $(\{(4)\}, \boldsymbol{q}_3)$ is NL- (non-deterministic logarithmic space) complete and linear-datalog-rewritable, $(\{(4)\}, \boldsymbol{q}_4)$ is L-complete and symmetric-datalog-rewritable, while $(\{(4)\}, \boldsymbol{q}_5)$ is in AC^0 and FO-rewritable. In the picture, $F(u)$ stands for $\mathit{Brexiteer}(u)$, $T(u)$ for

[4] https://www.w3.org/TR/owl2-profiles/.
[5] https://graphdb.ontotext.com.
[6] https://developer.logicblox.com.
[7] https://www.oxfordsemantic.tech.
[8] http://www.dlvsystem.com.
[9] https://potassco.org/clasp/.
[10] https://bioportal.bioontology.org/ontologies/SNOMEDCT.
[11] https://factpages.npd.no.



Remainer(u), $R(u, v)$ for *hasCoAuthor*(u, v), $S(u, v)$ for *hasBoss*(x, y), and all of the variables w, x, y, z are assumed to be existentially quantified. Another example is the experiments with the NPD FactPages and Subsurface Exploration ontologies used for testing OBDA in industry [15,22,27]. Although the ontologies contain covering axioms of the form $A \sqsubseteq B_1 \sqcup \cdots \sqcup B_n$ not allowed in *OWL 2 QL*, one can show that the concrete queries provided by the end-users do not 'feel' those dangerous axioms and are FO-rewritable. Note also the experiments in [28] showing that rewriting non-Horn OMQs to datalog can significantly improve the efficiency of answering by means of existing engines.

Is it possible to efficiently recognise the data complexity of answering any given OMQ and construct its optimal rewriting? The database community has been investigating these questions in the context of datalog optimisation since the 1980s; see Section 1.3 for details and references. For various families of DLs, a complexity-theoretic analysis of the **(data complexity)** problem was launched by Lutz and Wolter [29] and Bienvenu et al. [30]. Incidentally, the latter discovered a close connection with another important and rapidly growing area of Computer Science and AI: constraint satisfaction problems (CSPs), for which a P/NP-dichotomy, conjectured by Feder and Vardi [31], has recently been established [32,33]. We briefly survey the current state of the art in Section 1.3 below. Here, it suffices to say that recognising FO-rewritability is ExpTime-complete for OMQs with a 'lightweight' $\mathcal{EL}$ ontology [34,35] and 2NExpTime-complete for OMQs with a 'full-fledged' $\mathcal{ALC}$ ontology [36]. In either case, the problem seems to be too complex for a universal algorithmic solution, although experiments in [37] demonstrated that many real-life atomic OMQs in $\mathcal{EL}$ can be efficiently rewritten to non-recursive datalog by the ExpTime algorithm.

A more practical take on the **(rewritability)** problem, started by Motik [19], exploits the datalog connection mentioned above. In a nutshell, the idea is as follows. OMQs with a Horn DL ontology are rewritten to datalog queries, which could further be treated by the datalog optimisation techniques for removing or linearising recursion or partial FO-rewriting algorithms such as [38]. Non-Horn OMQs are transformed to (possibly exponential-size [20]) disjunctive datalog queries to which partial datalog rewriting algorithms such as the ones in [28] can be applied. It is to be emphasised, however, that tractable datalog optimisation and rewriting techniques cannot be complete.

In this article, we propose to approach the ultimate question from a different, bottom-up direction. In order to see the wood for the trees, we isolate some major sources of difficulty with **(data complexity)** and **(rewritability)** within a syntactically simple yet highly non-trivial class of OMQs. Apart from unearthing the fundamental roots of high complexity, this will allow us to obtain explicit syntactic rewritability conditions and even complete classifications of OMQs according to their data complexity and rewritability type. (Note that similar approaches were taken for analysing datalog programs and CSPs; see Sections 1.2 and 1.3.)

1.2. Our contribution

We investigate the **(data complexity)** and **(rewritability)** problems for OMQs $\boldsymbol{Q}$ of a very simple form:

(d-sirup) $\boldsymbol{Q} = (\mathsf{cov}_A, \boldsymbol{q})$, where $\mathsf{cov}_A = \{ A \sqsubseteq F \sqcup T \}$ and $\boldsymbol{q}$ is a Boolean CQ with unary predicates F, T and arbitrary binary predicates.

Our ultimate aim is to understand how the interplay between the *covering axiom* $A \sqsubseteq F \sqcup T$ and the structure of $\boldsymbol{q}$ determines the complexity and rewritability properties of $\boldsymbol{Q}$. By regarding $\boldsymbol{q}$ and data instances as labelled directed graphs (like in the picture above), we can formulate the problem of answering $\boldsymbol{Q}$ in plain graph-theoretic terms:

Instance: any labelled directed graph (digraph, for short) $\mathcal{A}$;
Problem: decide whether each digraph obtained by labelling every A-node in $\mathcal{A}$ with either F or T contains a homomorphic image of $\boldsymbol{q}$ (in which case the certain answer to $\boldsymbol{Q}$ over $\mathcal{A}$ is 'yes').

By definition (see, e.g., [39]), this can be done in coNP as $\boldsymbol{q}$ is *fixed*, and so the existence of a homomorphism from $\boldsymbol{q}$ to any labelling of $\mathcal{A}$ can be checked in polynomial time by inspecting all possible $|\mathcal{A}|^{|\boldsymbol{q}|}$-many maps from $\boldsymbol{q}$ to $\mathcal{A}$. In practice, we could try to solve this problem using, say, a resolution-based prover (see Example 2) or by evaluating the disjunctive datalog program {(6), (7)} below over $\mathcal{A}$, both of which would require finding proofs of exponential size in general (see Theorem 3). The **(data complexity)** and **(rewritability)** problems ask whether there exists a more efficient algorithmic solution for the given $\boldsymbol{Q}$ in principle and whether it can be realised as a standard (linear, symmetric) datalog or FO-query evaluated over the input graphs $\mathcal{A}$.

The OMQ $\boldsymbol{Q} = (\mathsf{cov}_A, \boldsymbol{q})$ is equivalent to the *monadic disjunctive datalog query*

$$T(x) \vee F(x) \leftarrow A(x) \tag{6}$$

$$\boldsymbol{G} \leftarrow \boldsymbol{q} \tag{7}$$

with a nullary goal predicate $\boldsymbol{G}$. In the 1980s, trying to understand boundedness (FO-rewritability) and linearisability (linear-datalog-rewritability) of datalog queries, the database community introduced the notion of *sirup*—standing for '*datalog query with a single recursive rule*' [40–42]—which was thought to be crucial for understanding datalog recursion and optimising datalog programs [43, Problem 4.2.10]. Our OMQs $\boldsymbol{Q}$ or disjunctive datalog queries $(\{(6),(7)\},\boldsymbol{G})$—which henceforth are referred to as (*monadic*) *disjunctive sirups* or simply *d-sirups*—play the same fundamental role for understanding OMQs with expressive ontologies and monadic disjunctive datalog queries.

Looking pretty trivial syntactically, d-sirups form a very sophisticated class of OMQs. For example, deciding FO-rewritability of d-sirups (even those of them that are equivalent to monadic datalog sirups) turns out to be 2ExpTime-hard [44]—as complex as deciding program boundedness of arbitrary monadic datalog programs [45,46]. Interestingly, one of the sources of this unexpectedly high complexity is 'twin' FT-labels of nodes in CQs like $\boldsymbol{q}_5$ above. We can eliminate this source by imposing the standard *disjointness constraint* $F \sqcap T \sqsubseteq \bot$ (or $\bot \leftarrow F(x), T(x)$ in datalog parlance), often used in ontologies and conceptual modelling. Thus, we arrive to *dd-sirups* of the form

(dd-sirup) $\boldsymbol{Q} = (\mathsf{cov}_A^\bot, \boldsymbol{q})$, where $\mathsf{cov}_A^\bot = \{\, A \sqsubseteq F \sqcup T,\ \ F \sqcap T \sqsubseteq \bot \,\}$.

The complexity and rewritability of both d- and dd-sirups only depend on the structure of the CQs $\boldsymbol{q}$, which suggests a research programme of classifying (d)d-sirups by the type of the graph underlying $\boldsymbol{q}$—directed path, tree, their undirected variants, etc.—and characterising the data complexity and rewritability of OMQs in the resulting classes. Thus, in the context of datalog sirups, Afrati and Papadimitriou [47] gave a complete characterisation of *binary chain* sirups that are computable in NC, and so parallelisable. Actually, according to [47], Kanellakis and Papadimitriou 'have investigated the case of unary sirups, and have made progress towards a complete characterization'. Unfortunately, that work has never been published.[12] (As shown later on in this article and [44], unary datalog sirups are closely connected to d-sirups.)

The main achievement of this article is a complete characterisation of dd-sirups with a path-shaped CQ (like $\boldsymbol{q}_1$–$\boldsymbol{q}_3$ and $\boldsymbol{q}_5$ above). Syntactically, the obtained characterisation, a tetrachotomy, is transparent and easily checkable: for any dd-sirup $\boldsymbol{Q} = (\mathsf{cov}_A^\bot, \boldsymbol{q})$ with a path-shaped CQ $\boldsymbol{q}$,

(AC^0) $\boldsymbol{Q}$ is FO-rewritable and can be answered in AC^0 iff $\boldsymbol{q}$ contains an FT-twin or has no F-nodes or no T-nodes;

otherwise,

(NL) $\boldsymbol{Q}$ is linear-datalog-rewritable and answering it is NL-complete if $\boldsymbol{q}$ is a 'periodic' CQ with a single F-node or a single T-node;
(P) $\boldsymbol{Q}$ is datalog-rewritable and answering it is P-complete if $\boldsymbol{q}$ is an 'aperiodic' CQ with a single F- or T-node;
(coNP) answering $\boldsymbol{Q}$ is coNP-complete if $\boldsymbol{q}$ has at least two F-nodes and at least two T-nodes.

(Assuming that NL $\neq$ P $\neq$ coNP, the three 'if' above can be replaced by 'iff'.) From the technical point of view, however, to establish this first complete syntactic characterisation of OMQs with disjunctive axioms, we require an adaptation of known methods from description logic [34,35] and datalog [45,46] as well as developing novel techniques for proving P- and especially coNP-hardness. As a (cruel) exercise, the reader might be tempted to consider the dd-sirup with $\boldsymbol{q}_1$ above and then permute the Fs and Ts in it. The known techniques of encoding NP-complete problems such as 2+2CNF or graph 3-colouring in terms of OMQ answering are not applicable in this case as $\mathsf{cov}_A^\bot$ is not capable of any reasoning bar *binary* case distinction and $\boldsymbol{q}_1$ has only one binary relation. (To compare, the first coNP-hard d-sirup found by Schaerf [1] has five roles that are used to encode clauses and their literals in 2+2-CNFs.) An even harder problem is to find a unified construction for arbitrary path-shaped CQs as different types of them require different treatment.

Structure of the article In the remainder of this section, we briefly review the related work. Section 2 contains the necessary background definitions. It also shows (by reduction of the mutilated chessboard problem [48,49]) that answering d-sirups using resolution-based provers requires finding proofs of exponential size in general.

In Section 3, we make an initial scan of the 'battleground' and obtain a few relatively simple complexity and rewritability results for arbitrary (not necessarily path-shaped) d- and dd-sirups. First, we show (by reduction of $\forall\exists$SAT) that answering (d)d-sirups is Π_2^p-complete for combined complexity (in the size of $\boldsymbol{q}$ and $\mathcal{A}$), that is, harder than answering *DL-Lite* and $\mathcal{EL}$ OMQs [50,15] (unless NP $= \Pi_2^p$, and so NP $=$ PSpace). This result is an improvement on Π_2^p-hardness of answering OMQs with a Schema.org ontology [51], which are more expressive than (d)d-sirups. Then we start classifying d-sirups in terms of occurrences of F and T in the CQs $\boldsymbol{q}$. Those without occurrences of a solitary F (like $\boldsymbol{q}_5$) or a solitary T are readily seen to be FO-rewritable. All other twinless d-sirups are shown to be L-hard, with certain symmetric d-sirups with one solitary F and one solitary T being rewritable to symmetric datalog, and so L-complete. D-sirups with a single solitary F or a single solitary T (and possibly with twins) are shown to be rewritable to monadic datalog queries (which also

[12] https://en.wikipedia.org/wiki/Paris_Kanellakis.

follows from [28]). This observation allows us to use datalog expansions [52] (called *cactuses* in our context) and automata-theoretic techniques [45] to analyse FO- and linear-datalog-rewritability of the corresponding d-sirups. In [44], we used the criterion of FO-rewritability in terms of cactuses to prove that deciding FO-rewritability of d-sirups with a single solitary F or T as well as that of monadic datalog sirups is 2ExpTime-complete. Here, we show that nonrecursive datalog, positive existential and UCQ-rewritings of such d-sirups are of at least single-, double- and triple-exponential size in the worst case, respectively.

As far as we are aware, there is no known semantic or syntactic criterion distinguishing between datalog programs in NL and P, though Lutz and Sabellek [34,35] gave a nice semantic characterisation of OMQs with an $\mathcal{EL}$ ontology. In Section 4, we combine their ideas with the automata-theoretic technique of Cosmadakis et al. [45] and prove a useful graph-theoretic sufficient condition for d-sirups to be linear-datalog-rewritable (and so in NL). Note that every d-sirup whose CQ $\boldsymbol{q}$ is a ditree with a single solitary F (or T) at the root can be rewritten to an atomic OMQ in $\mathcal{EL}$, to which the ExpTime-complete trichotomy of [34] is applicable.

Finally, in Sections 5 and 6, we obtain the tetrachotomy of the path-shaped dd-sirups discussed above. Items **($\mathbf{AC}^0$)** and **(NL)** and the upper bound in **(P)** follow from the previous sections. By far the hardest part of the tetrachotomy is establishing P- and coNP-hardness. To prove the former, we assemble AND- and OR-gates from copies of a given aperiodic CQ and then use those gates to construct ABoxes that 'compute' arbitrary monotone Boolean circuits, which is known to be P-complete. The structure of the gates and circuits is uniform for each type of aperiodicity. In the proof of coNP-hardness, building 3CNFs from copies of a given CQ $\boldsymbol{q}$ is not uniform as various parts of the construction subtly depend on the order of and the distances between the F- and T-nodes in $\boldsymbol{q}$. We are not aware of any even remotely similar methods in the literature, and believe that our novel 'bike technique' can be used for showing coNP-hardness of many other classes of OMQs. (It might be of interest to note that the coNP-hardness result in our tetrachotomy implies completeness of the datalog rewriting algorithm from [28] for path-shaped dd-sirups.)

In Section 7, we summarise the obtained results and formulate a few open problems for future research.

An extended abstract [53] with some of the results from this article has been presented at the 17th International Conference on Principles of Knowledge Representation and Reasoning.

1.3. Related work

There have been two big waves of research related to **(data complexity)** and **(rewritability)** of ontology-mediated queries. The first one started in the mid 1980s, when the database community was working on optimisation and parallelisation of datalog programs, which was hoped to be done by 'intelligent compilers' (see, e.g., [54,40,55–59], surveys [43,60] and references therein). One of the fundamental problems considered was to decide whether the depth of recursion required to evaluate a given datalog query could be bounded independently of the input data, which implies FO-rewritability of the datalog query. Boundedness was shown to be decidable in P for some classes of linear programs [61,54], NP-complete for linear monadic and dyadic single rule programs [41], PSpace-complete for linear monadic programs [45,62], and 2ExpTime-complete for arbitrary monadic programs [45,46]; see also [63]. On the other hand, boundedness of linear datalog queries with binary predicates and of ternary linear datalog queries with a single recursive rule was proved to be undecidable [64,65] along with many other semantic properties of datalog programs including linearisability, being in L or being in NC [66]. The computational complexity of evaluating datalog sirups (of arbitrary arity) as well as their descriptive complexity were studied in [42].

The second wave was largely caused by the apparent success story of the DL-underpinned Web Ontology Language OWL and the OBDA paradigm, both in theory and practice. On the one hand, as we mentioned earlier, large families of DLs that guarantee FO-rewritability [5,67] (the *DL-Lite*-family) and datalog-rewritability [68–70] (the $\mathcal{EL}$-family) and [14,20] (the Horn DL-family) were designed and investigated. Other types of rule-based languages with FO-rewritability have also been identified [71,72,11–13]. On the other hand, various methods for rewriting expressive OMQs to (disjunctive) datalog were suggested and implemented [16,17,73,28,74,75]. For example, the PAGOdA system combines the datalog reasoner RDFox and the OWL 2 reasoner HermiT [24]. A partial FO-rewriting algorithm for OMQs with an $\mathcal{ELU}$ ontology (allowing disjunction in $\mathcal{EL}$) and an atomic query was suggested in [38], and a sound and complete but not necessarily terminating algorithm for OMQs with existential rules in [13].

Complexity-theoretic investigations of the **(data complexity)** and **(rewritability)** problems for DL OMQs fall into two categories depending on whether the ontology language is Horn or not. FO-rewritability of OMQs with an ontology given in a Horn DL between $\mathcal{EL}$ and *Horn*$\mathcal{SHIF}$ was studied in [76,37,77], which provided semantic characterisations and established, using automata-theoretic techniques, the complexity of deciding FO-rewritability ranging from ExpTime via NExpTime to 2ExpTime. A complete characterisation of OMQs with an $\mathcal{EL}$-ontology was obtained in [34,35], establishing an AC^0/NL/P data complexity trichotomy, which corresponds to an FO-/linear-datalog-/datalog-rewritability trichotomy. Deciding this trichotomy was shown to be ExpTime-complete. FO-rewritability of OMQs whose ontology is a set of (frontier-)guarded existential rules was investigated in [78].

For non-Horn ontology languages (allowing disjunctive axioms), a crucial step in understanding **(data complexity)** and **(rewritability)** was the discovery in [30,36] of a connection between OMQs and non-uniform constraint satisfaction problems (CSPs) with a fixed template via MMSNP of [31]. It was used to show that deciding FO- and datalog-rewritability of OMQs with an ontology in any DL between $\mathcal{ALC}$ and $\mathcal{SHIU}$ and an atomic query is NExpTime-complete. The Feder-Vardi

dichotomy of CSPs [32,33] implies a P/coNP dichotomy of such OMQs, which is decidable in NExpTime. For monadic disjunctive datalog and OMQs with an $\mathcal{ALCI}$ ontology (that is, $\mathcal{ALC}$ with inverse roles) and a CQ, deciding FO-rewritability rises and becomes 2NExpTime-complete; deciding whether such an OMQ is rewritable to monadic datalog is between 2NExpTime and 3NExpTime [79,36]. Deciding FO-rewritability of OMQs with a Schema.org[13] ontology (which admits inclusions between concept and role names as well as covering axioms for role domains and ranges) and a union of CQs (UCQ) is PSpace-hard; for acyclic UCQs, it can be done in NExpTime [51]. The data complexity and rewritability of OMQs whose ontology is given in the guarded fragment of first-order logic were considered in [80].

Despite the discovery of general algebraic, automata- and graph-theoretic and semantic characterisations of data complexity and rewritability—which are usually very hard to check—there are very few explicit and easily checkable, possibly partial and applicable to limited OMQ families, sufficient and/or necessary conditions let alone complete classifications. Notable examples include (non-)linearisability conditions for chain datalog queries [81–83], the markability condition of datalog rewritability for disjunctive datalog programs and DL ontologies [28], and explicit NC/P-dichotomy of datalog chain sirups [47]. Classifications and dichotomies of various CSPs have been intensively investigated since Schaefer's classification theorem [84]; see, e.g., [85–88,32,33] and references therein.

The natural idea [30] of translating OMQs to CSPs and then using the algorithms and techniques developed for checking their complexity looks hardly viable in general: for instance, as reported in [89], the Polyanna program [90], designed to check tractability of CSPs, failed to recognise coNP-hardness of the very simple OMQ obtained by swapping the last F- and T-labels in $(\{(4)\}, \boldsymbol{q}_1)$ above because the CSP translation is unavoidably exponential.

2. Preliminaries

Using the standard description logic syntax and semantics [3], we consider *ontology-mediated queries* (OMQs) of the form $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q})$, where $\mathcal{O}$ is one of the two ontologies

$$\mathsf{cov}_A = \{ A \sqsubseteq F \sqcup T \}, \qquad \mathsf{cov}_A^{\bot} = \{ A \sqsubseteq F \sqcup T,\ \ F \sqcap T \sqsubseteq \bot \}$$

and $\boldsymbol{q}$ is a *Boolean conjunctive query* (*CQ*, for short): an FO-sentence $\boldsymbol{q} = \exists \boldsymbol{x} \varphi(\boldsymbol{x})$, in which φ is a conjunction of (constant- and function-free) atoms with variables from $\boldsymbol{x}$. We often think of $\boldsymbol{q}$ as the *set* of its atoms. In the context of this paper, CQs may only contain two unary predicates F, T and arbitrary binary predicates. As in the previous section, OMQs $\boldsymbol{Q} = (\mathsf{cov}_A, \boldsymbol{q})$ are also called *d-sirups* and $\boldsymbol{Q} = (\mathsf{cov}_A^{\bot}, \boldsymbol{q})$ *dd-sirups*.

Occasionally, we set $A = \top$, in which case $A \sqsubseteq F \sqcup T$ becomes the *total covering axiom* $F \sqcup T$. It is to be noted that this axiom is *domain dependent* [8], and so regarded to be *unsafe* and disallowed in disjunctive datalog. In general, answering a d-sirup $(\mathsf{cov}_A, \boldsymbol{q})$ could be harder than answering the corresponding OMQ $(\mathsf{cov}_\top, \boldsymbol{q})$ as shown by Example 20. We only consider OMQs $(\mathsf{cov}_\top, \boldsymbol{q})$ in examples and when proving some lower complexity bounds.

An *ABox* (data instance), $\mathcal{A}$, is a finite set of ground atoms with unary or binary predicates. We denote by $\mathsf{ind}(\mathcal{A})$ the set of constants (individuals) in $\mathcal{A}$. An *interpretation* is a structure of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with a domain $\Delta^{\mathcal{I}} \neq \emptyset$ and an interpretation function $\cdot^{\mathcal{I}}$ such that $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for any constant a, $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}} = \emptyset$, $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for any unary predicate P, and $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for any binary P. The truth-relation $\mathcal{I} \models \boldsymbol{q}$, for any CQ $\boldsymbol{q}$, is defined as usual in first-order logic. The interpretation $\mathcal{I}$ is a *model* of $\mathcal{O}$ if $A^{\mathcal{I}} \subseteq F^{\mathcal{I}} \cup T^{\mathcal{I}}$ and, for $\mathcal{O} = \mathsf{cov}_A^{\bot}$, also $F^{\mathcal{I}} \cap T^{\mathcal{I}} = \emptyset$; it is a *model* of $\mathcal{A}$ if $P(a) \in \mathcal{A}$ implies $a^{\mathcal{I}} \in P^{\mathcal{I}}$ and $P(a, b) \in \mathcal{A}$ implies $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$.

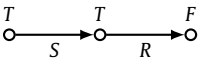
The *certain answer* to an OMQ $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q})$ over an ABox $\mathcal{A}$ is 'yes' if $\mathcal{I} \models \boldsymbol{q}$ for all models $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$—in which case we write $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$—and 'no' otherwise. A model of $\mathcal{O}$ and $\mathcal{A}$ is *minimal* if, for each *undecided* A-*individual* a, for which $A(a)$ is in $\mathcal{A}$ but neither $F(a)$ nor $T(a)$ is, exactly one of $a^{\mathcal{I}} \in F^{\mathcal{I}}$ or $a^{\mathcal{I}} \in T^{\mathcal{I}}$ holds. Clearly, $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$ iff $\mathcal{I} \models \boldsymbol{q}$ for every minimal model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$. So, from now on, 'model' always means 'minimal model'.

It is often convenient to regard CQs, ABoxes and interpretations as digraphs with labelled edges and partially labelled nodes (by F, T in CQs and F, T, A in ABoxes and interpretations). It is straightforward to see that the truth-relation $\mathcal{I} \models \boldsymbol{q}$ is equivalent to the existence of a digraph homomorphism $h \colon \boldsymbol{q} \to \mathcal{I}$ preserving the labels of nodes and edges. Without loss of generality, we assume that CQs are *connected* as undirected graphs. This graph-theoretic perspective allows us to consider special classes of CQs such as *tree-shaped* CQs, in which the underlying *undirected* graph is a tree, or *ditree-shaped* CQs, in which the underlying directed graph is a tree with all edges pointing away from the root, or dag-shaped CQs, which contain no directed cycles, etc. In particular, by a *path-shaped CQ* $\boldsymbol{q}$ (or *path CQ*, for short) we mean a (simple) directed path each of whose edges is labelled by one binary predicate. In other words, the binary atoms in $\boldsymbol{q}$ form a sequence $R_1(x_1, x_2), R_2(x_2, x_3), \dots, R_n(x_n, x_{n+1})$, where the x_i are all pairwise distinct variables in $\boldsymbol{q}$ (the R_i are not necessarily distinct).

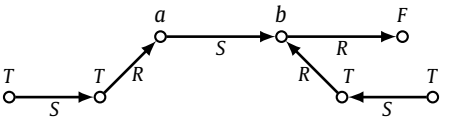
We illustrate the reasoning required to find the certain answer to a d-sirup over an ABox both on intuitive and formal levels. Our first example shows that, unsurprisingly, answering d-sirups can be done by a plain proof by cases.

Example 1. Consider the OMQ $\boldsymbol{Q} = (\mathsf{cov}_\top, \boldsymbol{q})$ with $A = \top$ and the path CQ $\boldsymbol{q}$ shown in the picture below:

[13] https://schema.org.



By analysing the four possible cases for $a, b \in F^{\mathcal{I}}, T^{\mathcal{I}}$ in an arbitrary model $\mathcal{I}$ of $\mathsf{cov}_\top$ and the ABox below, one can readily see that each of them contains $\boldsymbol{q}$ as a subgraph, and so the certain answer to $\mathbf{Q}$ over this ABox is 'yes'.



Indeed, if $a^{\mathcal{I}} \in F^{\mathcal{I}}$, then $\boldsymbol{q}$ is homomorphically embeddable into the S–R path on the left-hand side of $\mathcal{I}$. Otherwise $a^{\mathcal{I}} \in T^{\mathcal{I}}$. If $b^{\mathcal{I}} \in F^{\mathcal{I}}$, then $\boldsymbol{q}$ is homomorphically embeddable into the bottom S–R path on the right-hand side of $\mathcal{I}$. In the remaining case $b^{\mathcal{I}} \in T^{\mathcal{I}}$, there is a homomorphism from $\boldsymbol{q}$ into the S–R path on the top of $\mathcal{I}$.

Such proofs can be given as formal resolution refutations (derivations of the empty clause) in clausal logic.

Example 2. The certain answer to a d-sirup $\mathbf{Q} = (\mathsf{cov}_A, \boldsymbol{q})$ over an ABox $\mathcal{A}$ is 'yes' iff the following set $\mathcal{S}_{\mathbf{Q},\mathcal{A}}$ of clauses is unsatisfiable:

$$\mathcal{S}_{\mathbf{Q},\mathcal{A}} = \big\{ \neg A(y) \vee F(y) \vee T(y), \quad \bigvee_{P(\boldsymbol{x}) \in \boldsymbol{q}} \neg P(\boldsymbol{x}) \big\} \cup \mathcal{A}.$$

(For a dd-sirup $\mathbf{Q} = (\mathsf{cov}_A^{\perp}, \boldsymbol{q})$, the set $\mathcal{S}_{\mathbf{Q},\mathcal{A}}$ also contains the clause $\neg F(z) \vee \neg T(z)$.) In other words, the certain answer to $\mathbf{Q}$ over $\mathcal{A}$ is 'yes' iff there is a derivation of the empty clause from $\mathcal{S}_{\mathbf{Q},\mathcal{A}}$ in classical first-order resolution calculus [91]. By grounding $\mathcal{S}_{\mathbf{Q},\mathcal{A}}$, that is, by uniformly substituting individuals in $\mathsf{ind}(\mathcal{A})$ for the variables $\boldsymbol{x}$, y and z in the first two clauses of $\mathcal{S}_{\mathbf{Q},\mathcal{A}}$, we obtain a set $\bar{\mathcal{S}}_{\mathbf{Q},\mathcal{A}}$ of essentially propositional clauses with $|\bar{\mathcal{S}}_{\mathbf{Q},\mathcal{A}}|$ polynomial in $|\mathcal{A}|$. Again, the certain answer to $\mathbf{Q}$ over $\mathcal{A}$ is 'yes' iff there is a derivation of the empty clause from $\bar{\mathcal{S}}_{\mathbf{Q},\mathcal{A}}$ using propositional resolution. We now show that, in general, such derivations are of exponential size in $|\mathcal{A}|$.

Theorem 3. *There exist a CQ $\boldsymbol{q}$ and a sequence $\mathcal{A}_n$, $n > 0$, of ABoxes such that $|\mathcal{A}_n|$ is polynomial in n and any resolution refutation of $\mathcal{S}_{\mathbf{Q},\mathcal{A}_n}$ or $\bar{\mathcal{S}}_{\mathbf{Q},\mathcal{A}_n}$, for $\mathbf{Q} = (\mathsf{cov}_A, \boldsymbol{q})$, is of size $2^{\Omega(n)}$.*

Proof. We show that the *mutilated chessboard problem* can be solved by answering a d-sirup $\mathbf{Q}$ over certain ABoxes $\mathcal{A}_n$. The problem is as follows: given a chessboard of size $2n \times 2n$, for $n > 0$, with two white corner squares removed, prove that it *cannot* be covered by domino tiles (rectangles with two squares). This problem was encoded as a set of propositional clauses of size linear in n [48,49], any resolution proof of which is of size $2^{\Omega(n)}$ [48, Theorem 2.1]. We encode the same problem by the set $\mathcal{S}_{\mathbf{Q},\mathcal{A}_n}$, for some d-sirup $\mathbf{Q} = (\mathsf{cov}_A, \boldsymbol{q})$ and ABox $\mathcal{A}_n$. Since our encoding and the one in [48] are 'locally' translatable to each other and in view of [49, Proposition 3.4]), any resolution refutation of $\mathcal{S}_{\mathbf{Q},\mathcal{A}_n}$ or $\bar{\mathcal{S}}_{\mathbf{Q},\mathcal{A}_n}$ is also of size $2^{\Omega(n)}$.

Our CQ $\boldsymbol{q}$ is shown on the left-hand side of Fig. 1. The mutilated $2n \times 2n$ chessboard is turned to an ABox $\mathcal{A}_n$ by replacing each of its squares with the pattern shown in the middle of Fig. 1. The encodings of the different squares are connected via their four *contacts*, depicted as $\circ$-nodes. Each of these contacts is labelled by A or F, depending on whether it is in-between two squares, or at the boundary of the board; see the right-hand side of Fig. 1. All of the binary edges in $\boldsymbol{q}$ and $\mathcal{A}_n$ are assumed to be labelled by R. Labels w, x, y, z are just pointers and not parts of $\boldsymbol{q}$ or $\mathcal{A}_n$.

We call a model I of cov_A and $\mathcal{A}_n$ *covering* if exactly one contact in the encoding of each square is in $T^{\mathcal{I}}$. Covering models are clearly in one-to-one correspondence with domino-coverings (with each contact being in $T^{\mathcal{I}}$ iff it is between two squares covered by the same domino). We show that a model $\mathcal{I}$ of cov_A and $\mathcal{A}_n$ is covering iff $\mathcal{I} \not\models \boldsymbol{q}$. This implies the correctness of our encoding: the $2n \times 2n$ mutilated chessboard cannot be covered by dominos iff the answer to $\mathbf{Q}$ over $\mathcal{A}_n$ is 'yes', that is, there exist resolution refutations of both $\mathcal{S}_{\mathbf{Q},\mathcal{A}_n}$ and $\bar{\mathcal{S}}_{\mathbf{Q},\mathcal{A}_n}$.

($\Rightarrow$) If $\mathcal{I}$ is covering, then at least one of the four contacts of each square is not labelled by F. Thus, node x of $\boldsymbol{q}$ can be homomorphically mapped only to node w of the encoding of some square, and so y should be mapped to z. But then the two T-nodes in $\boldsymbol{q}$ should be mapped to two different contacts of the same square, contrary to the fact that in covering models only one such contact is labelled by T. Therefore, there is no $\boldsymbol{q} \to \mathcal{I}$ homomorphism. ($\Leftarrow$) If $\mathcal{I}$ is not covering because there is a square none of whose contacts is labelled by T, then by mapping x to z in the encoding of that square we can obtain a $\boldsymbol{q} \to \mathcal{I}$ homomorphism. And if there is a square such that at least two of its contacts are labelled by T, we can obtain a $\boldsymbol{q} \to \mathcal{I}$ homomorphism by mapping x to w and y to z. □

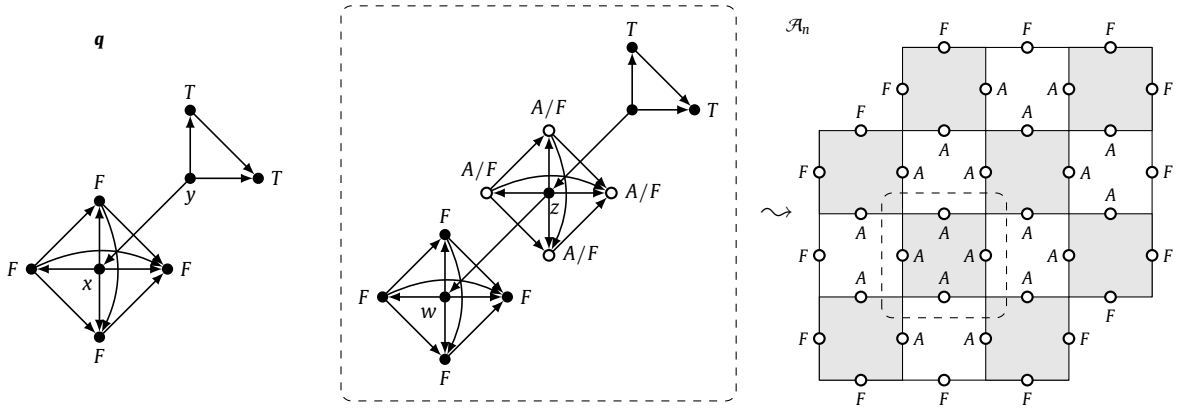


Fig. 1. Encoding the mutilated chessboard problem as $\mathcal{S}_{(\mathsf{cov}_A, \boldsymbol{q}), \mathcal{A}_n}$.

Our concern in the remainder of this article is the *combined* and *data complexity* of deciding, for a given (d)d-sirup $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q})$ and an ABox $\mathcal{A}$, whether $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$. In the former case, both $\boldsymbol{q}$ and $\mathcal{A}$ are regarded as input; in the latter one, $\boldsymbol{q}$ is fixed. It should be clear that $\Pi_2^p = \mathsf{coNP}^{\mathsf{NP}}$ is an upper bound for the combined complexity of our problem, which amounts to checking that, for every model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$, there exists a homomorphism $\boldsymbol{q} \to \mathcal{I}$, with the latter being NP-complete. For data complexity, when $\boldsymbol{q}$ is fixed, checking the existence of a homomorphism $\boldsymbol{q} \to \mathcal{I}$ can be done in P, and so the whole problem is in coNP.

We are also interested in various types of rewritability of (d)d-sirups. An OMQ $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q})$ is called *FO-rewritable* if there is an FO-sentence Φ such that $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$ iff Φ is true in $\mathcal{A}$ given as an FO-structure [9]. In terms of circuit complexity, FO-rewritability is equivalent to answering $\boldsymbol{Q}$ in logtime-uniform AC^0 [9].

Recall from, say [8], that a *datalog program*, Π, is a finite set of *rules* of the form $\forall \boldsymbol{x} (\gamma_0 \leftarrow \gamma_1 \wedge \dots \wedge \gamma_m)$, where each γ_i is a (constant- and function-free) atom $Q(\boldsymbol{y})$ with $\boldsymbol{y} \subseteq \boldsymbol{x}$. As usual, we omit $\forall \boldsymbol{x}$. The atom γ_0 is the *head* of the rule, and $\gamma_1, \dots, \gamma_m$ its *body*. All of the variables in the head must occur in the body. The predicates in the heads of rules are called *IDB predicates*, the rest *EDB predicates*. The *arity* of Π is the maximum arity of its IDB predicates; 1-ary Π is called *monadic*. A *datalog query* in this article takes the form $(\Pi, \boldsymbol{G})$ with a 0-ary (goal) atom $\boldsymbol{G}$. The *answer* to $(\Pi, \boldsymbol{G})$ over an ABox $\mathcal{A}$ is 'yes' if $\boldsymbol{G}$ is true in the structure $\Pi(\mathcal{A})$ obtained by closing $\mathcal{A}$ under the rules in Π, in which case we write $\Pi, \mathcal{A} \models \boldsymbol{G}$. We call $(\Pi, \boldsymbol{G})$ a *datalog-rewriting* of an OMQ $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q})$ in case $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$ iff $\Pi, \mathcal{A} \models \boldsymbol{G}$, for any ABox $\mathcal{A}$ containing EDB predicates of Π only. If $\boldsymbol{Q}$ is datalog-rewritable, then it can be answered in P for data complexity [60].

If there is a rewriting of $\boldsymbol{Q}$ to a $(\Pi, \boldsymbol{G})$ with a *linear* program Π, having at most one IDB predicate in the body of each of its rules, then $\boldsymbol{Q}$ can be answered in NL (non-deterministic logarithmic space). The NL upper bound also holds for datalog queries with a linear-stratified program, which is defined as follows. A *stratified* program [8] is a sequence $\Pi = (\Pi_0, \dots, \Pi_n)$ of datalog programs, called the *strata* of Π, such that each predicate in Π can occur in the head of a rule only in one stratum Π_i and can occur in the body of a rule only in strata Π_j with $j \geq i$. If, in addition, the body of each rule in Π contains at most one occurrence of a head predicate from the same stratum, Π is called *linear-stratified*. Every linear-stratified program can be converted to an equivalent linear datalog program [83], and so datalog queries with a linear-stratified program can be answered in NL for data complexity.

A linear program Π is *symmetric* if, for any recursive rule $I(\boldsymbol{x}) \leftarrow J(\boldsymbol{y}) \wedge E(\boldsymbol{z})$ in Π (except the goal rules), where J is an IDB predicate and $E(\boldsymbol{z})$ is the conjunction of the EDBs of the rule, its symmetric counterpart $J(\boldsymbol{y}) \leftarrow I(\boldsymbol{x}) \wedge E(\boldsymbol{z})$ is also a rule in Π. It is known (see, e.g., [92]) that symmetric programs can be evaluated in L (deterministic logarithmic space) for data complexity. Thus, if $\boldsymbol{Q}$ is rewritable to a symmetric datalog query, it can be answered in L.

The complexity classes we deal with in this article form the chain

$$\mathsf{AC}^0 \subsetneq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{coNP} \subseteq \Pi_2^p$$

(whether any of the inclusions $\subseteq$ is strict is a major open problem in complexity theory). The P/NP dichotomy for CSPs [32, 33] and the reductions from [30,36] imply that every (d)d-sirup is either in P or coNP-complete. However, as far as we know, at the moment there are no other established dichotomies for OMQs with disjunctive axioms. On the other hand, as mentioned above, OMQ answering in AC^0 is equivalent to FO-rewritability, but whether OMQ answering in L (NL or P) implies symmetric-datalog-rewritability (respectively, linear-datalog- or datalog-rewritability) also remains open.

3. Initial observations

In this section, we obtain a number of relatively simple complexity and rewritability results that are applicable to arbitrary (not necessarily path) d- and dd-sirups.

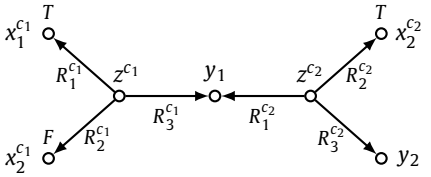
3.1. Combined complexity

Our first result pushes to the limit [51, Theorem 5] according to which answering OMQs with a Schema.org ontology is Π_2^p-complete for combined complexity (the hardness proof of that theorem uses an ontology with an enumeration definition $E = \{0, 1\}$ and additional concept names, i.e., unary predicates, none of which is available in our case).

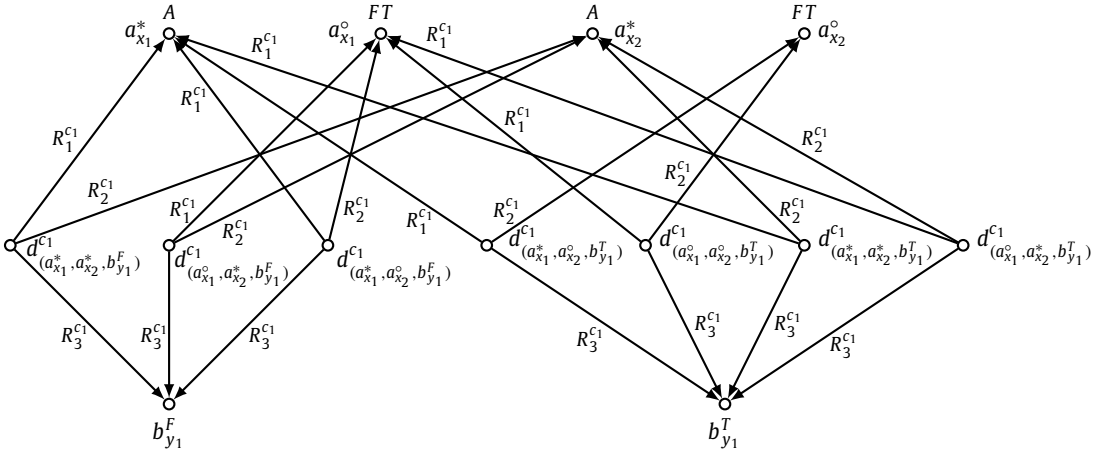
Theorem 4. (*i*) *Answering d-sirups* $(\mathsf{cov}_A, \boldsymbol{q})$ *and dd-sirups* $(\mathsf{cov}_A^{\perp}, \boldsymbol{q})$ *is* Π_2^p*-complete for combined complexity.*
(*ii*) *Answering d- and dd-sirups with a tree-shaped CQ* $\boldsymbol{q}$ *is* coNP*-complete for combined complexity.*

Proof. As mentioned in Section 2, deciding whether $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$, given a (d)d-sirup $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q})$ and an ABox $\mathcal{A}$, can be done by a coNP Turing machine (checking all models $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$) with an NP-oracle (checking the existence of a homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$); for tree-shaped $\boldsymbol{q}$, a P-oracle is enough (see, e.g., [93] and further references therein). The lower bound in (*ii*) follows from Theorem 27.

For (*i*), we prove it by reduction of Π_2^p-complete ∀∃3SAT [94]. We remind the reader that a propositional formula $\psi(\boldsymbol{x}, \boldsymbol{y})$ with tuples $\boldsymbol{x}$ and $\boldsymbol{y}$ of propositional variables is a *3CNF* if it is a conjunction of *clauses* of the form $\ell_1 \vee \ell_2 \vee \ell_3$, where each ℓ_i is a *literal* (a propositional variable or a negation thereof). The decision problem ∀∃3SAT asks whether the fully quantified propositional formula $\varphi = \forall \boldsymbol{x} \exists \boldsymbol{y}\, \psi(\boldsymbol{x}, \boldsymbol{y})$ is true, for any given 3CNF ψ. We may assume that each clause contains each variable at most once. Denote by $\boldsymbol{q}_\varphi$ the CQ that, for each clause $c = \ell_1 \vee \ell_2 \vee \ell_3$ in ψ, contains atoms $R_i^c(z^c, u_i^c)$, $i = 1, 2, 3$, with $u_i^c = y$ if $y \in \boldsymbol{y}$ occurs in ℓ_i and $u_i^c = x^c$ if $x \in \boldsymbol{x}$ occurs in ℓ_i; in the latter case, $\boldsymbol{q}_\varphi$ also contains $T(x^c)$ if $\ell_i = x$ and $F(x^c)$ if $\ell_i = \neg x$. For example, clauses $c_1 = x_1 \vee \neg x_2 \vee y_1$ and $c_2 = \neg y_1 \vee x_2 \vee y_2$ contribute the following atoms to $\boldsymbol{q}_\varphi$:



For $\mathcal{O} = \mathsf{cov}_A$, the ABox $\mathcal{A}_\varphi$ is defined as follows. For $x \in \boldsymbol{x}$, we take individuals a_x^* and a_x° and, for $y \in \boldsymbol{y}$, individuals b_y^F and b_y^T. $\mathcal{A}_\varphi$ contains the atoms $A(a_x^*)$, $F(a_x^\circ)$, $T(a_x^\circ)$, for $x \in \boldsymbol{x}$. For each $c = \ell_1 \vee \ell_2 \vee \ell_3$, we define a set E^c of triples of the above individuals: $(e_1, e_2, e_3) \in E^c$ iff (*i*) for $i = 1, 2, 3$, $e_i \in \{a_x^*, a_x^\circ\}$ whenever $x \in \boldsymbol{x}$ occurs in ℓ_i, (*ii*) for $i = 1, 2, 3$, $e_i \in \{b_y^F, b_y^T\}$ whenever $y \in \boldsymbol{y}$ occurs in ℓ_i, and (*iii*) there is $i \in \{1, 2, 3\}$ such that either $e_i = a_x^*$, or $e_i = b_y^\nu$ and the assignment $y = \nu$ makes ℓ_i true. Now, for any c and (e_1, e_2, e_3) in E^c, we take a fresh individual $d^c_{(e_1,e_2,e_3)}$—the *centre* of the pair $\big(c, (e_1, e_2, e_3)\big)$—and add three atoms $R_i^c(d^c_{(e_1,e_2,e_3)}, e_i)$, $i = 1, 2, 3$, to $\mathcal{A}_\varphi$. To illustrate, for $c_1 = x_1 \vee \neg x_2 \vee y_1$, the set E^{c_1} contains all triples of the form $(a_{x_1}^{\mu_1}, a_{x_2}^{\mu_2}, b_{y_1}^\nu)$ except $(a_{x_1}^\circ, a_{x_2}^\circ, b_{y_1}^F)$ and gives the following fragment of $\mathcal{A}_\varphi$:



For $\mathcal{O} = \mathsf{cov}_A^{\perp}$, we take a_x^F and a_x^T instead of each a_x°, add the atoms $F(a_x^F)$, $T(a_x^T)$ instead of $F(a_x^\circ)$, $T(a_x^\circ)$, and replace item (*i*) in the definition of E^c with (*i*)′ for $i = 1, 2, 3$, $e_i \in \{a_x^*, a_x^F, a_x^T\}$ whenever $x \in \boldsymbol{x}$ occurs in ℓ_i.

The number of atoms in $\mathcal{A}_\varphi$ is polynomial in the size of φ.

Claim 4.1. *Suppose* $\mathfrak{a}\colon \boldsymbol{x} \to \{F, T\}$ *is any assignment and*

$$\mathcal{A}_\varphi^{\mathfrak{a}} \;=\; \mathcal{A}_\varphi \cup \{\, T(a_x^*) \mid \mathfrak{a}(x) = T,\ x \in \boldsymbol{x}\,\} \cup \{\, F(a_x^*) \mid \mathfrak{a}(x) = F,\ x \in \boldsymbol{x}\,\}.$$

There exists an assignment $\mathfrak{b}\colon \boldsymbol{y} \to \{F, T\}$ *that makes* $\psi(\mathfrak{a}(\boldsymbol{x}), \mathfrak{b}(\boldsymbol{y}))$ *true iff* $\mathcal{A}_\varphi^{\mathfrak{a}} \models \boldsymbol{q}_\varphi$.

Proof. ($\Rightarrow$) Suppose $\mathfrak{b}$ is such that $\psi(\mathfrak{a}(\boldsymbol{x}), \mathfrak{b}(\boldsymbol{y}))$ is true. We need to show that there is a homomorphism $h\colon \boldsymbol{q}_\varphi \to \mathcal{A}_\varphi^{\mathfrak{a}}$.

Case cov_A: For any clause $c = \ell_1 \vee \ell_2 \vee \ell_3$ in ψ and for any $i = 1, 2, 3$, we define e_i^c as follows. We let (*i*) $e_i^c = a_x^*$ if $x \in \boldsymbol{x}$ occurs in ℓ_i and $\mathfrak{a}$ makes ℓ_i true, (*ii*) $e_i^c = a_x^\circ$ if $x \in \boldsymbol{x}$ occurs in ℓ_i and $\mathfrak{a}$ makes ℓ_i false, and (*iii*) $e_i^c = b_y^{\mathfrak{b}(y)}$ if $y \in \boldsymbol{y}$ occurs in ℓ_i. As $\psi(\mathfrak{a}(\boldsymbol{x}), \mathfrak{b}(\boldsymbol{y}))$ is true, (e_1^c, e_2^c, e_3^c) is in E^c. Then we define a map h by taking $h(z^c)$ to be the centre of $\big(c, (e_1^c, e_2^c, e_3^c)\big)$ and $h(u_i^c) = e_i^c$. It follows from the construction that h is well-defined and a homomorphism from $\boldsymbol{q}_\varphi$ to $\mathcal{A}_\varphi$ with respect to the binary atoms. We show that it preserves the unary atoms as well. Indeed, for each c and each $x \in \boldsymbol{x}$ occurring in some literal ℓ_i in c, there are two cases: (1) If x^c is labelled by T in $\boldsymbol{q}_\varphi$, then $\ell_i = x$. So if $\mathfrak{a}$ makes ℓ_i true, then $e_i^c = a_x^*$ is labelled by T in $\mathcal{A}_\varphi^{\mathfrak{a}}$. And if $\mathfrak{a}$ makes ℓ_i false, then $e_i^c = a_x^\circ$ is labelled by both T and F in $\mathcal{A}_\varphi^{\mathfrak{a}}$. (2) If x^c is labelled by F in $\boldsymbol{q}_\varphi$, then $\ell_i = \neg x$. So if $\mathfrak{a}$ makes ℓ_i true, then $e_i^c = a_x^*$ is labelled by F in $\mathcal{A}_\varphi^{\mathfrak{a}}$. And if $\mathfrak{a}$ makes ℓ_i false, then $e_i^c = a_x^\circ$ is labelled by both T and F in $\mathcal{A}_\varphi^{\mathfrak{a}}$.

Case $\mathsf{cov}_A^\perp$: In the definition of e_i^c, we replace (*ii*) with (*ii*)′ $e_i^c = a_x^T$ if $\ell_i = x$ for some $x \in \boldsymbol{x}$ and $\mathfrak{a}(x) = F$, and (*ii*)″ $e_i^c = a_x^F$ if $\ell_i = \neg x$ for some $x \in \boldsymbol{x}$ and $\mathfrak{a}(x) = T$. Again, we claim that h as defined above preserves the unary atoms. Indeed, for each c and for each $x \in \boldsymbol{x}$ occurring in some literal ℓ_i in c, there are two cases: (1) If x^c is labelled by T in $\boldsymbol{q}_\varphi$, then $\ell_i = x$. So if $\mathfrak{a}$ makes ℓ_i true, then $e_i^c = a_x^*$ is labelled by T in $\mathcal{A}_\varphi^{\mathfrak{a}}$. And if $\mathfrak{a}$ makes ℓ_i false, then $e_i^c = a_x^T$ is labelled by T in $\mathcal{A}_\varphi^{\mathfrak{a}}$. (2) If x^c is labelled by F in $\boldsymbol{q}_\varphi$, then $\ell_i = \neg x$. So if $\mathfrak{a}$ makes ℓ_i true, then $e_i^c = a_x^*$ is labelled by F in $\mathcal{A}_\varphi^{\mathfrak{a}}$. And if $\mathfrak{a}$ makes ℓ_i false, then $e_i^c = a_x^F$ is labelled by F in $\mathcal{A}_\varphi^{\mathfrak{a}}$.

($\Leftarrow$) Suppose $h\colon \boldsymbol{q}_\varphi \to \mathcal{A}_\varphi^{\mathfrak{a}}$. Then, for any $y \in \boldsymbol{y}$, we have $h(y) = b_y^\nu$ for some $\nu \in \{F, T\}$. We then set $\mathfrak{b}(y) = \nu$. We claim that $\psi(\mathfrak{a}(\boldsymbol{x}), \mathfrak{b}(\boldsymbol{y}))$ is true. Indeed, for every clause $c = \ell_1 \vee \ell_2 \vee \ell_3$ in ψ, there is $(e_1, e_2, e_3) \in E^c$ such that h maps the 'contribution' of c in $\boldsymbol{q}_\varphi$ onto the 'star' with centre $d^c_{(e_1,e_2,e_3)}$. If (e_1, e_2, e_3) is in E^c because $e_i = a_x^*$, for some $i \in \{1, 2, 3\}$, $x \in \boldsymbol{x}$, then the label of a_x^* in $\mathcal{A}_\varphi^{\mathfrak{a}}$ is $\mathfrak{a}(x)$. As h is a homomorphism, the label of x^c in $\boldsymbol{q}_\varphi$ is also $\mathfrak{a}(x)$, and so $\mathfrak{a}$ makes c true by the definition of $\boldsymbol{q}_\varphi$. And if (e_1, e_2, e_3) is in E^c because $e_i = b_y^{\mathfrak{b}(y)}$, for some $i \in \{1, 2, 3\}$, $y \in \boldsymbol{y}$ with $\mathfrak{b}(y)$ making ℓ_i true, then c is clearly true as well. □

Finally, we prove that φ is satisfiable iff $\mathcal{O}, \mathcal{A}_\varphi \models \boldsymbol{q}_\varphi$ iff $\mathcal{I} \models \boldsymbol{q}_\varphi$ for every model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}_\varphi$. ($\Rightarrow$) Given $\mathcal{I}$, define an assignment $\mathfrak{a}_\mathcal{I}\colon \boldsymbol{x} \to \{F, T\}$ by taking $\mathfrak{a}_\mathcal{I}(x) = T$ if $a_x^* \in T^\mathcal{I}$ and $\mathfrak{a}_\mathcal{I}(x) = F$ if $a_x^* \in F^\mathcal{I}$. Then $\mathcal{I} = \mathcal{A}_\varphi^{\mathfrak{a}_\mathcal{I}}$, and so we are done by Claim 4.1. The implication ($\Leftarrow$) also follows from Claim 4.1, as $\mathcal{A}_\varphi^{\mathfrak{a}}$ is a model of $\mathcal{O}$ and $\mathcal{A}_\varphi$, for every assignment $\mathfrak{a}\colon \boldsymbol{x} \to \{F, T\}$. □

3.2. *Data complexity:* AC^0 *and* L

From now on, we focus on the *data* complexity of answering d-sirups $(\mathsf{cov}_A, \boldsymbol{q})$ and dd-sirups $(\mathsf{cov}_A^\perp, \boldsymbol{q})$. We start classifying (d)d-sirups in terms of occurrences of F and T in the CQs $\boldsymbol{q}$. Atoms $F(x), T(x) \in \boldsymbol{q}$, for some variable x, are referred to as FT*-twins* in $\boldsymbol{q}$. If $\boldsymbol{q}$ does not contain FT-twins, we call it *twinless*, and similarly for ABoxes. By a *solitary* F or T we mean a non-twin F- or, respectively, T-node.

Observe that answering $(\mathsf{cov}_A, \boldsymbol{q})$ is not easier than answering $(\mathsf{cov}_A^\perp, \boldsymbol{q})$: If a given ABox $\mathcal{A}$ contains FT-twins, then there is no model of $\mathsf{cov}_A^\perp$ and $\mathcal{A}$, and so $\mathsf{cov}_A^\perp, \mathcal{A} \models \boldsymbol{q}$. Also,

$$\mathsf{cov}_A, \mathcal{A} \models \boldsymbol{q} \quad \text{iff} \quad \mathsf{cov}_A^\perp, \mathcal{A} \models \boldsymbol{q}, \quad \text{for any twinless ABox } \mathcal{A}. \tag{8}$$

If $\boldsymbol{q}$ contains FT-twins, then $\exists x \big(F(x) \wedge T(x)\big)$ is an FO-rewriting of $(\mathsf{cov}_A^\perp, \boldsymbol{q})$. In general, any rewriting of $(\mathsf{cov}_A, \boldsymbol{q})$ can be converted to a rewriting of $(\mathsf{cov}_A^\perp, \boldsymbol{q})$ into the same language. For example, if $(\Pi, \boldsymbol{G})$ is a (symmetric/linear) datalog rewriting of $(\mathsf{cov}_A, \boldsymbol{q})$, then $(\Pi \cup \{\bot \leftarrow F(x), T(x)\}, \boldsymbol{G})$ is a (symmetric/linear) datalog rewriting of $(\mathsf{cov}_A^\perp, \boldsymbol{q})$.

Aiming to identify FO-rewritable (d)d-sirups, we consider first those CQs that do not have a solitary F or a solitary T, calling them 0-*CQs*. Queries of this type are quite common, only asking about one of the covering predicates (as in 'are there any undergraduate students who take symbolic AI courses?' and 'what about the postgraduate ones?' provided that students are either undergraduate or postgraduate). The following theorem establishes a complexity dichotomy between 0-CQs and non-0-CQs, which contain occurrences of both covering predicates (as in 'are there both undergraduate and postgraduate students in the College's University Challenge team?').

Theorem 5. (*i*) *If* $\boldsymbol{q}$ *is a* 0*-CQ, then both* $(\mathsf{cov}_A, \boldsymbol{q})$ *and* $(\mathsf{cov}_A^\perp, \boldsymbol{q})$ *can be answered in* AC^0*, with* $\boldsymbol{q}$ *being an FO-rewriting of each of them.*

(*ii*) *If* $\boldsymbol{q}$ *is twinless and contains at least one solitary* F *and at least one solitary* T*, then answering* $(\mathsf{cov}_\top, \boldsymbol{q})$ *and* $(\mathsf{cov}_\top^\perp, \boldsymbol{q})$*, and so* $(\mathsf{cov}_A, \boldsymbol{q})$ *and* $(\mathsf{cov}_A^\perp, \boldsymbol{q})$ *is* L*-hard.*

Proof. (i) Let $\mathcal{O}$ be one of cov_A or $\mathsf{cov}_A^\perp$. We show that $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$ iff $\mathcal{A} \models \boldsymbol{q}$, and so $\boldsymbol{q}$ is an FO-rewriting of $(\mathcal{O}, \boldsymbol{q})$. ($\Rightarrow$) Suppose $\mathcal{A} \not\models \boldsymbol{q}$ and $\boldsymbol{q}$ has no solitary F (the other case is similar). Let $\mathcal{A}'$ be the result of adding a label F to every undecided A-node in $\mathcal{A}$. Clearly, $\mathcal{A}'$ is a model of $\mathcal{O}$ and $\mathcal{A}$ with $\mathcal{A}' \not\models \boldsymbol{q}$. ($\Leftarrow$) is trivial.

(ii) The proof is by an FO-reduction of the L-complete reachability problem for undirected graphs. Denote by $\boldsymbol{q}'$ the CQ obtained by gluing together all the T-nodes and by gluing together all the F-nodes in $\boldsymbol{q}$. Thus, $\boldsymbol{q}'$ contains a single T-node, x, and a single F-node, y. Clearly, there is a homomorphism $h\colon \boldsymbol{q} \to \boldsymbol{q}'$. Let $\boldsymbol{q}'' = \boldsymbol{q}' \setminus \{T(x), F(y)\}$.

Suppose $G = (V, E)$ is a graph with $\mathfrak{s}, \mathfrak{t} \in V$. We regard G as a directed graph such that $(\mathfrak{u}, \mathfrak{v}) \in E$ iff $(\mathfrak{v}, \mathfrak{u}) \in E$, for any $\mathfrak{u}, \mathfrak{v} \in V$. Construct a twinless ABox $\mathcal{A}_G$ from G in the following way. Replace each edge $e = (\mathfrak{u}, \mathfrak{v}) \in E$ by a copy $\boldsymbol{q}''_e$ of $\boldsymbol{q}''$ such that, in $\boldsymbol{q}''_e$, node x is renamed to $\mathfrak{u}$, y to $\mathfrak{v}$, and all other nodes z to some fresh copy $\mathfrak{z}_e$. Then $\mathcal{A}_G$ comprises all such $\boldsymbol{q}''_e$, for $e \in E$, as well as atoms $T(\mathfrak{s})$ and $F(\mathfrak{t})$. We show that there is a path from $\mathfrak{s}$ to $\mathfrak{t}$ in G ($\mathfrak{s} \to_G \mathfrak{t}$, in symbols) iff $\mathsf{cov}_\top, \mathcal{A}_G \models \boldsymbol{q}$ iff $\mathsf{cov}_\top^\perp, \mathcal{A}_G \models \boldsymbol{q}$ (cf. (8)).

($\Rightarrow$) Suppose there is a path $\mathfrak{s} = \mathfrak{v}_0, \dots, \mathfrak{v}_n = \mathfrak{t}$ in G with $e_i = (\mathfrak{v}_i, \mathfrak{v}_{i+1}) \in E$, for $i < n$. Consider an arbitrary model $\mathcal{I}$ of $\mathsf{cov}_\top$ and $\mathcal{A}_G$. Since $\mathcal{I} \models \mathsf{cov}_\top$, and $T(\mathfrak{s})$ and $F(\mathfrak{t})$ are in $\mathcal{A}_G$, we can find some $i < n$ such that $\mathfrak{v}_i \in T^\mathcal{I}$ and $\mathfrak{v}_{i+1} \in F^\mathcal{I}$. As $\boldsymbol{q}''_{e_i}$ is an isomorphic copy of $\boldsymbol{q}''$, we obtain $\mathcal{I} \models \boldsymbol{q}''$, and so $\mathcal{I} \models \boldsymbol{q}$.

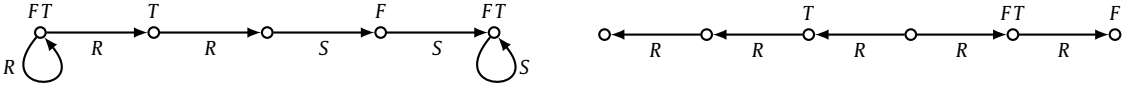
($\Leftarrow$) Suppose $\mathfrak{s} \not\to_G \mathfrak{t}$. Then, by the construction, $\mathfrak{t}$ is not reachable from $\mathfrak{s}$ in $\mathcal{A}_G$ (not even via an undirected path). Define a model $\mathcal{I}$ of $\mathsf{cov}_\top^\perp$ and $\mathcal{A}_G$ by taking $T^\mathcal{I}$ to be the set of nodes in $\mathcal{A}_G$ that are reachable from $\mathfrak{s}$ (via an undirected path) and $F^\mathcal{I}$ its complement. Clearly, no connected component of $\mathcal{A}_G$ (as undirected graph) contains both $T^\mathcal{I}$- and $F^\mathcal{I}$ nodes. Since $\boldsymbol{q}$ is connected and contains at least one T and at least one F, it follows that $\mathcal{I} \not\models \boldsymbol{q}$. □

As $\mathsf{AC}^0 \subsetneq \mathsf{L}$ and $\exists x\,(F(x) \wedge T(x))$ is an FO-rewriting of $(\mathsf{cov}_A^\perp, \boldsymbol{q})$ in which $\boldsymbol{q}$ contains a twin, Theorem 5 gives a sufficient and necessary criterion of FO-rewritability for dd-sirups:

Corollary 6. *A dd-sirup $\mathbf{Q} = (\mathsf{cov}_A^\perp, \boldsymbol{q})$ can be answered in AC^0 iff $\boldsymbol{q}$ is a 0-CQ or contains a twin.*

Characterising FO-rewritable d-sirups with a CQ containing twins turns out to be a much harder problem, which will be discussed in Section 3.4.

Example 7. Meanwhile, the reader is invited to show that the d-sirups with the CQs below are FO-rewritable (see also Example 14). Note that each of these CQs is *minimal*, that is, not equivalent to any of its proper sub-CQs.



The lower bound result in Theorem 5 (ii) is complemented by the following simple sufficient condition. To formulate it, we require non-Boolean CQs $\boldsymbol{q}(\boldsymbol{x})$ that apart from existentially quantified variables may also contain *free* variables $\boldsymbol{x}$ called *answer* or *distinguished variables*. Such a CQ $\boldsymbol{q}'(x, y)$ is *symmetric* if, for any ABox $\mathcal{A}$ and any $a, b \in \mathsf{ind}(\mathcal{A})$, we have $\mathcal{A} \models \boldsymbol{q}'(a, b)$ iff $\mathcal{A} \models \boldsymbol{q}'(b, a)$, where $\mathcal{A}$ is regarded as an FO-structure and $\models$ is the usual first-order truth relation.

Theorem 8. *Let $\mathcal{O}$ be one of cov_A or $\mathsf{cov}_A^\perp$ and let $\boldsymbol{q}$ be a Boolean CQ that is equivalent to*

$$\exists x, y\, \big(F(x) \wedge \boldsymbol{q}'_1(x) \wedge \boldsymbol{q}'(x, y) \wedge \boldsymbol{q}'_2(y) \wedge T(y)\big),$$

where (a) CQs $\boldsymbol{q}'_1(x)$, $\boldsymbol{q}'(x, y)$ and $\boldsymbol{q}'_2(y)$ do not contain solitary T and F, (b) $\boldsymbol{q}'(x, y)$ is symmetric, and (c) $\boldsymbol{q}'_1(x)$ and $\boldsymbol{q}'_2(y)$ are disjoint, with x and y being their only common variables with $\boldsymbol{q}'(x, y)$. Then $(\mathcal{O}, \boldsymbol{q})$ is rewritable to a symmetric datalog program, and so can be answered in L.

Proof. Suppose $\mathcal{O} = \mathsf{cov}_A$. We claim that $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$ iff there exist $n \geq 1$ and $v_0, v_1, \dots, v_n \in \mathsf{ind}(\mathcal{A})$ such that

(S1) $F(v_0), A(v_1), \dots, A(v_{n-1}), T(v_n) \in \mathcal{A}$,
(S2) $\mathcal{A} \models \boldsymbol{q}'(v_i, v_{i+1})$, for $0 \leq i < n$,
(S3) $\mathcal{A} \models \boldsymbol{q}'_1(v_i)$, for $0 \leq i < n$,
(S4) $\mathcal{A} \models \boldsymbol{q}'_2(v_i)$, for $1 \leq i \leq n$.

Indeed, suppose there are $v_0, v_1, \dots, v_n \in \mathsf{ind}(\mathcal{A})$ such that (S1)–(S4) hold. Consider any model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$. By (S1), there is $i < n$ with $v_i \in F^\mathcal{I}$ and $v_{i+1} \in T^\mathcal{I}$. Then (S2)–(S4) guarantee that $\mathcal{I} \models \boldsymbol{q}$. Conversely, suppose $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$ for some ABox $\mathcal{A}$. For $P \in \{F, T, A\}$, let $P^\mathcal{A} = \{a \in \mathsf{ind}(\mathcal{A}) \mid P(a) \in \mathcal{A}\}$. Define inductively sets F_j and F'_j, for $j \geq 0$, by setting $F_0 = F^\mathcal{A}$, $F'_j = \{b \mid \mathcal{A} \models \boldsymbol{q}'_1(a) \wedge \boldsymbol{q}'(a, b) \wedge \boldsymbol{q}'_2(b)$ for some $a \in F_j\}$ and $F_{j+1} = A^\mathcal{A} \cap F'_j$. Let $\mathcal{I}$ be a model of $\mathcal{O}$ and $\mathcal{A}$ with $F^\mathcal{I} = \bigcup_{j=0}^\infty F_j$ and $T^\mathcal{I} = T^\mathcal{A} \cup \big(A^\mathcal{A} \setminus \bigcup_{j=1}^\infty F_j\big)$. By our assumption, there is a homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. Thus, $h(x) \in F_j$ and $h(y) \in F'_j$ for

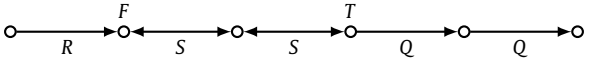
some j. Then $h(y) \in T^{\mathcal{A}}$, for otherwise $h(y) \in F_{j+1}$, contrary to $h(y) \in T^{\mathcal{I}}$. Now, let $v_{n-1} = h(x)$ and $v_n = h(y)$. If $j = 0$ then we are done with $n = 1$. If $j > 0$ then $h(x) \in A^{\mathcal{A}} \cap F'_{j-1}$, and so there is $v_{n-2} \in F_{j-1}$ such that $\mathcal{A} \models \boldsymbol{q}'_1(v_{n-2}) \wedge \boldsymbol{q}'(v_{n-2}, v_{n-1}) \wedge \boldsymbol{q}'_2(v_{n-1})$. By iterating this process, we obtain $v_0, v_1, \dots, v_n \in \mathsf{ind}(\mathcal{A})$ as required.

It remains to observe that checking whether there are $v_0, v_1, \dots, v_n \in \mathsf{ind}(\mathcal{A})$ such that (S1)–(S4) hold can be done by the following symmetric datalog program, in which $B(x) = A(x) \wedge \boldsymbol{q}'_1(x) \wedge \boldsymbol{q}'_2(x)$:

$$\begin{aligned}
\boldsymbol{G} &\leftarrow \boldsymbol{q} \\
\boldsymbol{G} &\leftarrow F(x), \boldsymbol{q}'_1(x), \boldsymbol{q}'(x, y), P(y) \\
P(x) &\leftarrow B(x), \boldsymbol{q}'(x, y), \boldsymbol{q}'_2(y), T(y) \\
P(x) &\leftarrow B(x), \boldsymbol{q}'(x, y), P(y), B(y)
\end{aligned}$$

where, by the symmetry of $\boldsymbol{q}'(x, y)$, the only recursive rule $P(x) \leftarrow B(x), \boldsymbol{q}'(x, y), P(y), B(y)$ is equivalent to its symmetric counterpart. If $\mathcal{O} = \mathsf{cov}^{\perp}_{A}$, we add the non-recursive rule $\boldsymbol{G} \leftarrow F(x), T(x)$ to the program. □

Example 9. By Theorems 8 and 5 (*ii*), the d-sirup $(\mathsf{cov}_{\top}, \boldsymbol{q})$ with $\boldsymbol{q}$ shown below is L-complete.



3.3. Datalog rewritability of d-sirups with a 1-CQ

In this section, we introduce a technical tool that can be used to show datalog rewritability of (d)d-sirups whose CQ contains exactly one solitary F and at least one solitary T (or exactly one solitary T and at least one solitary F). We refer to such CQs as *1-CQs*. The tool is an adaptation of the known (disjunctive) datalog technique of *expansions* [54,45,52]. We use this tool to observe that every (d)d-sirup with a 1-CQ can be rewritten to a very simple datalog query—nearly a sirup in the sense of [40,41], and so can be answered in P. Note that a more general *markability* technique (tracing dependencies on disjunctive predicates in the program rules) for rewriting disjunctive datalog programs into datalog was developed in [28]. In Section 3.4, we also adapt the datalog expansion technique to characterise FO-rewritability of those datalog queries semantically.

Throughout this section, we assume that $\boldsymbol{q}$ is a 1-CQ, with $F(x)$ and $T(y_1), \dots, T(y_n)$ being all of the solitary occurrences of F and T in $\boldsymbol{q}$. As before, we let $\mathcal{O}$ be one of cov_A or $\mathsf{cov}^{\perp}_{A}$. For each dd-sirup $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q})$, we define a monadic (that is, having at most unary IDB predicates) datalog program $\Pi_{\boldsymbol{Q}}$ with nullary goal $\boldsymbol{G}$ and four rules

$$\boldsymbol{G} \leftarrow F(x), \boldsymbol{q}', P(y_1), \dots, P(y_n) \tag{9}$$

$$P(x) \leftarrow T(x) \tag{10}$$

$$P(x) \leftarrow A(x), \boldsymbol{q}', P(y_1), \dots, P(y_n) \tag{11}$$

$$\boldsymbol{G} \leftarrow F(x), T(x) \tag{12}$$

where $\boldsymbol{q}' = \boldsymbol{q} \setminus \{F(x), T(y_1), \dots, T(y_n)\}$ and P is a fresh predicate symbol that never occurs in ABoxes. Thus, the body of rule (11) is obtained from $\boldsymbol{q}$ by replacing $F(x)$ with $A(x)$ and each $T(y_i)$ with $P(y_i)$. If $\mathcal{O} = \mathsf{cov}_A$, rule (12) is omitted.

We also define by induction a class $\mathfrak{K}_{\boldsymbol{Q}}$ of ABoxes called *cactuses for* $\boldsymbol{Q}$. We start by setting $\mathfrak{K}_{\boldsymbol{Q}} = \{\boldsymbol{q}\}$, regarding $\boldsymbol{q}$ as an ABox, and then recursively apply to $\mathfrak{K}_{\boldsymbol{Q}}$ the following 'budding' rule:

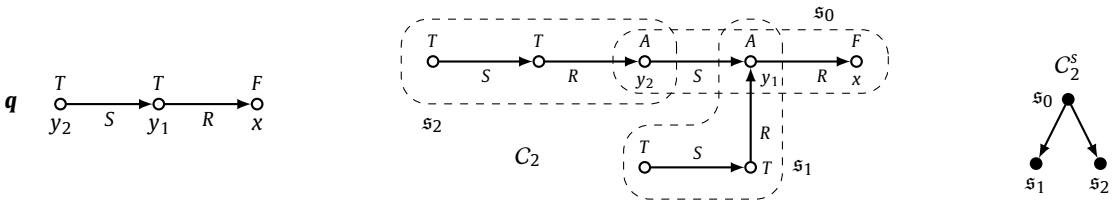
(bud) if $T(y) \in C \in \mathfrak{K}_{\boldsymbol{Q}}$ with solitary $T(y)$, then we add to $\mathfrak{K}_{\boldsymbol{Q}}$ the ABox obtained by replacing $T(y)$ in C with the set $(\boldsymbol{q} \setminus \{F(x)\}) \cup \{A(x)\}$, in which x is renamed to y and all other variables are given *fresh* names.

It is straightforward to see by structural induction that

$$\mathcal{O}, C \models \boldsymbol{q}, \quad \text{for every } C \in \mathfrak{K}_{\boldsymbol{Q}}. \tag{13}$$

For $C \in \mathfrak{K}_{\boldsymbol{Q}}$, we refer to the copies $\mathfrak{s}$ of (maximal subsets of) $\boldsymbol{q}$ comprising C as *segments*. 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 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 10. In the picture below, the cactus C_2 is obtained by applying **(bud)** to the 1-CQ $\boldsymbol{q}$ twice. Its skeleton C_2^{s} with three segments $\mathfrak{s}_0, \mathfrak{s}_1, \mathfrak{s}_2$ is shown on the right-hand side of the picture.



Theorem 11. *For any (d)d-sirup $\mathbf{Q} = (\mathcal{O}, \boldsymbol{q})$ with a 1-CQ $\boldsymbol{q}$ and any ABox $\mathcal{A}$, the following conditions are equivalent*:

(*i*) $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$,
(*ii*) $\Pi_{\mathbf{Q}}, \mathcal{A} \models \boldsymbol{G}$,
(*iii*) *there exists a homomorphism* $h\colon \mathcal{C} \to \mathcal{A}$, *for some* $\mathcal{C} \in \mathfrak{K}_{\mathbf{Q}}$, *or* $\mathcal{O} = \mathrm{cov}_A^{\perp}$ *and* $\mathcal{A}$ *contains an FT-twin.*

Proof. We show the implications (*i*) $\Rightarrow$ (*ii*) $\Rightarrow$ (*iii*) $\Rightarrow$ (*i*).

(*i*) $\Rightarrow$ (*ii*) If $\mathcal{O} = \mathrm{cov}_A^{\perp}$ and $\mathcal{A}$ contains a node labelled by both T and F, then $\boldsymbol{G}$ holds in the closure $\Pi_{\mathbf{Q}}(\mathcal{A})$ of $\mathcal{A}$ under $\Pi_{\mathbf{Q}}$ by rule (12). In any other case, we define a model $\mathcal{I}$ based on $\mathcal{A}$ by labelling each 'undecided' A-node a by T if $P(a)$ holds in $\Pi_{\mathbf{Q}}(\mathcal{A})$, and by F otherwise. As $\mathcal{I}$ is a model of $\mathcal{O}$ and $\mathcal{A}$, there is a homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. Then $h(y_i) \in T^{\mathcal{I}}$, and so $P\big(h(y_i)\big)$ holds in $\Pi_{\mathbf{Q}}(\mathcal{A})$, for every $i \leq n$ (by rule (10) and the definition of $\mathcal{I}$). We claim that $h(x)$ is an F-node in $\Pi_{\mathbf{Q}}(\mathcal{A})$, and so $\boldsymbol{G}$ holds in $\Pi_{\mathbf{Q}}(\mathcal{A})$ by rule (9). Indeed, otherwise by $h(x) \in F^{\mathcal{I}}$ and the definition of $\mathcal{I}$, $h(x)$ is an A-node but not a P-node in $\Pi_{\mathbf{Q}}(\mathcal{A})$, contrary to rule (11).

(*ii*) $\Rightarrow$ (*iii*) Suppose $\mathcal{O} = \mathrm{cov}_A$ or $\mathcal{A}$ does not contain a node labelled by both T and F. Then rule (12) is either not in $\Pi_{\mathbf{Q}}$ or not used. We define inductively (on the applications of rule (11) in the derivation of $\boldsymbol{G}$) a cactus $\mathcal{C} \in \mathfrak{K}_{\mathbf{Q}}$ and a homomorphism $h\colon \mathcal{C} \to \mathcal{A}$. To begin with, there are objects $x^a, y_1^a, \dots, y_n^a$ for which rule (9) was triggered. Thus, x^a is an F-node in $\Pi_{\mathbf{Q}}(\mathcal{A})$, and so it is an F-node in $\mathcal{A}$. Take a function $h_0\colon \boldsymbol{q} \to \mathcal{A}$ that preserves binary predicates, with $h_0(x) = x^a$ and $h_0(y_i) = y_i^a$ for $i \leq n$. If y_i^a is a T-node in $\mathcal{A}$ for every $i \leq n$, then $h = h_0$ is the required homomorphism from $\boldsymbol{q} \in \mathfrak{K}_{\mathbf{Q}}$ to $\mathcal{A}$. If y_i^a is not a T-node in $\mathcal{A}$, for some i, then y_i^a is a P-node in $\Pi_{\mathbf{Q}}(\mathcal{A})$ obtained by rule (11), and so y_i^a is an A-node in $\mathcal{A}$. Also, there are $x^b = y_i^a$ and $y_1^b, \dots, y_n^b$ such that rule (11) was triggered for $x^b, y_1^b, \dots, y_n^b$. Let $\mathcal{C}$ be the cactus obtained from $\boldsymbol{q}$ by budding at y_i. We extend h_0 to a function $h_1\colon \mathcal{C} \to \mathcal{A}$ such that it preserves binary predicates and $h_1(y_j^{\mathfrak{s}}) = y_j^b$ for all T-nodes $y_j^{\mathfrak{s}}$ of the new segment $\mathfrak{s}$. If y_j^b is a T-node in $\mathcal{A}$ for every $j \leq n$, then $h = h_1$ is the required homomorphism from $\mathcal{C} \in \mathfrak{K}_{\mathbf{Q}}$ to $\mathcal{A}$. Otherwise, we bud $\mathcal{C}$ again and repeat the above argument. As the derivation of $\boldsymbol{G}$ from $\mathcal{A}$ using $\Pi_{\mathbf{Q}}$ is finite, sooner or later the procedure stops with a cactus and a homomorphism.

(*iii*) $\Rightarrow$ (*i*) If $\mathcal{O} = \mathrm{cov}_A^{\perp}$ and $\mathcal{A}$ contains a node labelled by both T and F, then $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$ obviously holds. Otherwise, take an arbitrary model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{A}$. We define a model $\mathcal{I}^+$ of $\mathcal{O}$ and $\mathcal{C}$ by 'pulling back $\mathcal{I}$' via the homomorphism h: for every node x in $\mathcal{C}$, $x \in A^{\mathcal{I}^+}$ iff $h(x) \in A^{\mathcal{I}}$. By (13), there is a homomorphism $g\colon \boldsymbol{q} \to \mathcal{I}^+$. Thus, the composition of g and h is a $\boldsymbol{q} \to \mathcal{I}$ homomorphism, as required. □

Corollary 12. *Any (d)d-sirup $(\mathcal{O}, \boldsymbol{q})$ with a 1-CQ $\boldsymbol{q}$ is datalog-rewritable, and so can be answered in* P.

As mentioned in the introduction, the problems of FO-rewritability (aka boundedness in the datalog literature) and linear-datalog-rewritability (aka linearisability) of datalog queries have been thoroughly investigated since the 1980s. In Sections 3.4 and 4, we discuss these questions for (d)d-sirups with a 1-CQ.

3.4. Deciding FO-rewritability of d- and dd-sirups with a 1-CQ

A key to understanding FO-rewritability of d- and dd-sirups with a 1-CQ is the following semantic criterion, which is well-known in the datalog setting; see, e.g., [54,45]:

Theorem 13. *A (d)d-sirup $\mathbf{Q} = (\mathcal{O}, \boldsymbol{q})$ with a 1-CQ $\boldsymbol{q}$ is FO-rewritable iff there exists a $d < \omega$ such that every cactus $\mathcal{C} \in \mathfrak{K}_{\mathbf{Q}}$ contains a homomorphic image of some cactus $\mathcal{C}^- \in \mathfrak{K}_{\mathbf{Q}}$ of depth $\leq d$, in which case a disjunction of the cactuses of depth $\leq d$, regarded as Boolean CQs, is an FO-rewriting of $\mathbf{Q}$.*

Proof. ($\Rightarrow$) By [30, Proposition 5.9], $\mathbf{Q}$ has an FO-rewriting of the form $\boldsymbol{q}_1 \vee \dots \vee \boldsymbol{q}_n$, where the $\boldsymbol{q}_i$ are CQs. Treating the $\boldsymbol{q}_i$ as ABoxes, we obviously have $\mathcal{O}, \boldsymbol{q}_i \models \boldsymbol{q}$, and so, by Theorem 11, there is a homomorphism from some $\mathcal{C}_i \in \mathfrak{K}_{\mathbf{Q}}$ to $\boldsymbol{q}_i$. Now let d be the maximum of the depths of the $\mathcal{C}_i$. Consider any $\mathcal{C} \in \mathfrak{K}_{\mathbf{Q}}$ of depth $> d$. Then there are homomorphisms $\mathcal{C}_i \to \boldsymbol{q}_i \to \mathcal{C}$, for some i, $1 \leq i \leq n$, as required.

($\Leftarrow$) Given $d < \omega$, we take all of the cactuses $\mathcal{C}_1, \dots, \mathcal{C}_n$ of depth $\leq d$ (up to isomorphism). Now we consider each $\mathcal{C}_i$ as a CQ. Then $\mathcal{C}_1 \vee \dots \vee \mathcal{C}_n$ is an FO-rewriting of $\mathbf{Q}$. Indeed, if $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$ then there are homomorphisms $\mathcal{C}_i \to \mathcal{C} \to \mathcal{A}$, for some $\mathcal{C}$ and i, again by Theorem 11. □

Example 14. Let $\boldsymbol{Q}_1$ be the d-sirup with the first CQ from Example 7. It is not hard to verify that every cactus for $\boldsymbol{Q}_1$ contains a homomorphic image of this CQ, which is therefore an FO-rewriting of $\boldsymbol{Q}_1$. Now, let $\boldsymbol{Q}_2$ be the d-sirup with the second CQ from Example 7. Let C_k be the cactus obtained by applying **(bud)** k-times to the original cactus C_0 (isomorphic to the given CQ). There are homomorphisms $h\colon C_1 \to C_k$, for $k \geq 2$, and so $\boldsymbol{Q}_2$ is rewritable to $C_0 \vee C_1$.

As follows from [45], which considered arbitrary monadic datalog queries, checking the criterion of Theorem 13 can be done in 2ExpTime. A matching lower bound for monadic datalog queries with multiple recursive rules was established in [46]. It has recently been shown that already deciding FO-rewritability of monadic datalog sirups of the form $(\{(10), (11)\}, P(x))$ and also of d-sirups with a 1-CQ is 2ExpTime-hard [44]. Thus, we obtain:

Theorem 15 *([45,36,44]). Deciding FO-rewritability of d-sirups can be done in 2*NExpTime*. Deciding FO-rewritability of d-sirups with a 1-CQ is 2*ExpTime*-complete. It follows that deciding FO-rewritability of CQs mediated by a Schema.org or DL-Lite$_{bool}$ [67] ontology can be done in 2*NExpTime*, and is 2*ExpTime*-hard.*

The exact complexity of deciding FO-rewritability of d-sirups (2NExpTime or 2ExpTime) remains an open problem. Another important issue for OBDA and datalog optimisation is the *succinctness problem* for FO-rewritings [95,50]. It is not known if every FO-rewritable d-sirup has a polynomial-size FO-rewriting. However, we can show that this is not the case for the UCQ-, PE- and NDL-rewritings, which are standard in OBDA systems. We remind the reader (see [50] for details and further references) that a *UCQ-rewriting* takes the form of disjunction (union) of CQs, while a *positive existential* (*PE*) *rewriting* is built from atoms using $\exists$, $\wedge$ and $\vee$ in an arbitrary way. A *nonrecursive datalog* (*NDL*) *rewriting* is a datalog query $(\Pi, \boldsymbol{G})$ such that the dependency digraph of Π is acyclic, where a predicate P *depends* on a predicate P' in Π if Π has a clause with P in the head and P' in the body.

Theorem 16. *There is a sequence of FO-rewritable d-sirups $\boldsymbol{Q}_n = (\mathsf{cov}_A, \boldsymbol{q}_n)$ of polynomial size in $n > 0$ such that any UCQ-, PE- and NDL-rewritings of $\boldsymbol{Q}_n$ are of at least triple, double and single exponential size in n, respectively.*

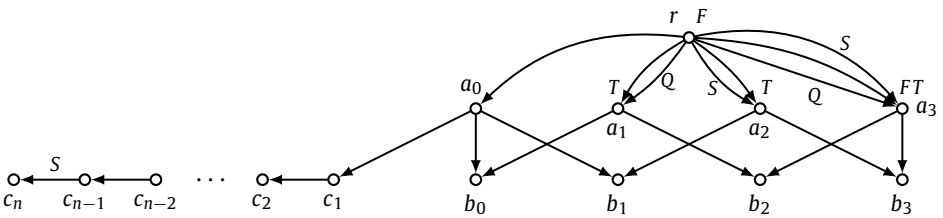
Proof. Consider an alternating Turing machine (ATM) $\boldsymbol{M}_n$ that works as follows on any input of length $\leq n$. Its tape of size exponential in n is used as a counter from 0 to 2^{2^n}. The tape also has two extra cells a and b. $\boldsymbol{M}_n$ begins in a $\vee$-state by writing 0 and 1 in cell a in two alternative branches of the full computation space. Then $\boldsymbol{M}_n$ continues, in a $\wedge$-state, by writing 0 and 1 in cell b in two alternative branches of the full computation space. If the bits in a and b in a given branch of the tree are distinct, $\boldsymbol{M}_n$ enters an accepting state. Otherwise, the counter is increased by 1 and the ATM repeats the previous two steps. If the counter exceeds 2^{2^n}, $\boldsymbol{M}_n$ enters a rejecting state. Thus, $\boldsymbol{M}_n$ rejects every input. Moreover, given any input $\boldsymbol{w}$, every computation tree of $\boldsymbol{M}_n$ on $\boldsymbol{w}$ contains exactly one rejecting configuration, which is the leaf of a branch of length double-exponential in n.

We now use the ATMs $\boldsymbol{M}_n$ and any input $\boldsymbol{w}$ of length $\leq n$ to construct, as described in [44], polynomial-size 1-CQs $\boldsymbol{q}_n$. Then, by the ($\Rightarrow$) direction of [44, Lemma 4], the d-sirups $\boldsymbol{Q}_n = (\mathsf{cov}_A, \boldsymbol{q}_n)$ are FO-rewritable. On the other hand, one can show similarly to the proof of the ($\Leftarrow$) direction of [44, Lemma 4] that any computation tree of $\boldsymbol{M}_n$ on $\boldsymbol{w}$ corresponds to a cactus $C \in \mathfrak{K}_{\boldsymbol{Q}_n}$ of triple-exponential size in n such that no smaller cactus is homomorphically embeddable to C. It follows that *any* UCQ-rewritings of $\boldsymbol{Q}_n$ must be of at least triple-exponential size.

Any PE-rewritings of $\boldsymbol{Q}_n$ are of at least double-exponential size. Indeed, given a PE-rewriting in prenex form of size s (the number of atoms in the formula), we can transform its matrix (the quantifier-free part) to DNF and obtain a UCQ rewriting of size $\leq s2^s$. So if s were sub-double-exponential, then the size of this UCQ-rewriting would be less than triple-exponential. A similar argument shows that it is impossible to obtain NDL-rewritings of subexponential size because otherwise we could transform them to sub-double-exponential PE-rewritings. □

The proof above does not provide us with any lower bound on the size of FO-rewritings because, by a result of Gurevich and Shelah [96], there is a potentially non-elementary blow-up in length from a homomorphism invariant FO-sentence to its shortest equivalent PE-sentence. We illustrate Theorem 16 by a simple example of an FO-rewritable d-sirup whose UCQ-rewritings are of at least double-exponential size.

Example 17. Consider the d-sirups $\boldsymbol{Q}_n = (\mathsf{cov}_A, \boldsymbol{q}_n)$, where $\boldsymbol{q}_n$, for $n \geq 2$, is the 1-CQ depicted below, with the omitted labels on the edges being all R (and r, a_i, b_i, c_i being pointers rather than labels in $\boldsymbol{q}_n$).



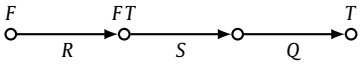
For any cactus $C \in \mathfrak{K}_{\boldsymbol{q}_n}$ and any node x in $\boldsymbol{q}_n$, let x^C denote the copy of x in the root segment of C. Observe that C is of depth $\geq n$ iff C contains an R-path π that starts at r^C and has $\geq n$ A-nodes, the first of which is either a_1^C or a_2^C. We show first that if the depth of C is $\geq n$, then there is a homomorphism $h\colon \boldsymbol{q}_n \to C$. Indeed, if the first A-node of π is a_1^C, then we can define h by taking $h(r) = r^C$, $h(a_0) = a_1^C$, $h(a_i) = a_3^C$ for $i = 1, 2, 3$, $h(b_j) = b_2^C$ for $j = 0, 1$, $h(b_j) = b_3^C$ for $j = 2, 3$, $c_1, \dots, c_{n-1}$ are h-mapped to the next $n-1$ A-nodes in π, and $h(c_n)$ is the FT-node in the segment with root $h(c_{n-1})$. (The case when the first A-node of π is a_2^C is similar.) So Theorem 13 implies that $\boldsymbol{Q}_n$ is FO-rewritable.

On the other hand, we claim that if C, C' are cactuses of depths $< n$ and there is a homomorphism $h\colon C \to C'$, then $C = C'$. We show this by induction on the depth of C (which cannot exceed the depth of C'). Observe first that, for any x in $\boldsymbol{q}_n$, we must have $h(x^C) = x^{C'}$: This holds for r^C because the FT-node a_3 has no S-successors, for a_0^C because the depth of C' is less than n, $h(a_1^C) \neq a_2^{C'}$ because a_2 has no Q-predecessor, $h(a_1^C) \neq a_3^{C'}$ because a_0 and a_1 have a common successor, while a_0 and a_3 do not, we have a similar argument for $h(a_2^C)$, and then we clearly have $h(x^C) = x^{C'}$ for $x = c_1, \dots, c_{n-1}, a_3$. It follows that if $C = \boldsymbol{q}_n$ then $C' = \boldsymbol{q}_n$ must also hold, otherwise h does not preserve T. If the depth of C is > 0 then, for $i = 1, 2$, let $\mathfrak{s}_i$ be the segment in C^s having a_i^C as its root node, and let C_i^- be the 'subcactus' of C whose skeleton is the subtree of C^s with root $\mathfrak{s}_i$. We define $C_i'^-$ from C' similarly. An inspection of $\boldsymbol{q}_n$ shows that we must have homomorphisms $h_1\colon C_1^- \to C_1'^-$ and $h_2\colon C_2^- \to C_2'^-$. Thus, we have $C_1^- = C_1'^-$ and $C_2^- = C_2'^-$ by the induction hypothesis (IH). Therefore, $C = C'$ follows, and so the UCQ rewriting Φ_n of $\boldsymbol{Q}_n$ provided by Theorem 13 contains all different cactuses of depth $< n$, the number of which is $2^{2^{O(n)}}$. It follows that *any* UCQ-rewritings Φ_n' of $\boldsymbol{Q}_n$ have at least $2^{2^{O(n)}}$ disjuncts. For otherwise, by the pigeonhole principle, there exist different disjuncts C and C' in Φ_n and $C \to D$ and $C' \to D$ homomorphisms, for some disjunct D of Φ_n'. On the other hand, there is a $D \to C''$ homomorphism, for some disjunct C'' in Φ_n, and so, as shown above, $C = C' = C''$, which is a contradiction.

One can readily transform Φ_n to an equivalent PE-rewriting of exponential size at the expense of nested $\wedge$ and $\vee$. But, by the proof of Theorem 16, there are no PE-rewritings of subexponential size. On the other hand, the datalog program {(9)–(11)} can be converted to an NDL-program describing cactuses of depth $< n$ and containing $O(n)$ rules.

Finding an explicit syntactic characterisation of FO-rewritable d-sirups turns out to be nearly as hard as characterising FO-rewritable OMQs in fully-fledged expressive DLs and monadic disjunctive datalog queries. Notice, however, that the 1-CQs used in Example 17 and the construction of [44] (underlying Theorem 16) are quite involved dags with multiple edges and possibly multiple FT-twins. So one could hope that by restricting the shape of CQs and/or by disallowing FT-twins we would obtain less impenetrable yet practically useful classes of d-sirups. Indeed, for d-sirups $\boldsymbol{Q}$ whose 1-CQ is a *ditree* with its unique solitary F-node as root, the program $\Pi_{\boldsymbol{Q}}$ can be reformulated as an $\mathcal{EL}$-ontology, and so one can use the AC^0/NL/P trichotomy of [34,35], which is checkable in ExpTime.

Example 18. To illustrate, consider the 1-CQ $\boldsymbol{q}$ below:



We have $\mathsf{cov}_A, \mathcal{A} \models \boldsymbol{q}$ iff $\mathcal{E}, \mathcal{A} \models \exists x\, B(x)$, where $\mathcal{E}$ is the $\mathcal{EL}$ TBox $\{F \sqcap C_{\boldsymbol{q}} \sqsubseteq B,\ T \sqsubseteq P,\ A \sqcap C_{\boldsymbol{q}} \sqsubseteq P\}$ with $C_{\boldsymbol{q}} = \exists R.(F \sqcap T \sqcap \exists S.\exists Q.P)$ and the DL syntax illustrated in terms of first-order logic by (1)–(4) in Section 1.

Further, as shown in [44], any d-sirup with a ditree 1-CQ, not necessarily having an F-labelled root, is either FO-rewritable or L-hard, and deciding this dichotomy is fixed-parameter tractable if we regard the number of solitary T-nodes as a parameter. Moreover, for dd-sirups with an arbitrary ditree CQ, there is an explicit syntactic trichotomy: each of them is either FO-rewritable or L-complete, or NL-hard. On the other hand, there is no readily available machinery for explicitly characterising NL-completeness, P- and coNP-hardness of (d)d-sirups (let alone more general types of OMQs). We are going to fill in this gap to some extent in the remainder of the article. To begin with, we combine some ideas from [45,34] to prove a general sufficient condition of linearisability for d-sirups with a 1-CQ.

4. Linear-datalog-rewritability of d- and dd-sirups with a 1-CQ

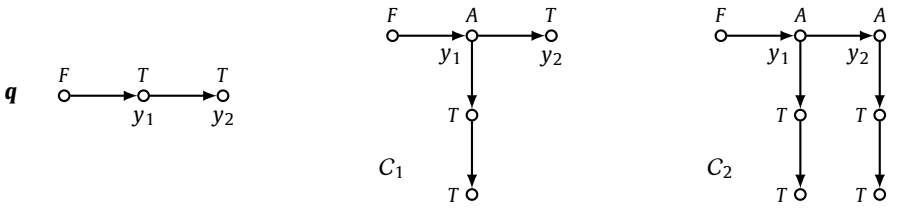
We require a few new definitions, assuming as before that $\mathcal{O}$ is one of cov_A or $\mathsf{cov}_A^\perp$. First, we extend the class $\mathfrak{K}_{\boldsymbol{Q}}$ of *cactuses* for any (d)d-sirup $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q})$ to a wider class $\mathfrak{K}_{\boldsymbol{Q}}^+$ by adding another inductive rule to its definition. We define $\mathfrak{K}_{\boldsymbol{Q}}^+$ as the class of structures obtained from $\boldsymbol{q}$ by recursively applying **(bud)** and the following 'pruning' rule:

(prune) if $C \in \mathfrak{K}_{\boldsymbol{Q}}^+$ and $\mathcal{O}, C^- \models \boldsymbol{q}$, where $C^- = C \setminus \{T(y)\}$, for some solitary $T(y)$ in C, then we add C^- to $\mathfrak{K}_{\boldsymbol{Q}}^+$.

If C^- is obtaining from C by **(prune)**, we define the skeleton $(C^-)^s$ of C^- to be C^s. We continue to call members of $\mathfrak{K}_{\boldsymbol{Q}}^+$ *cactuses*. We write $C' \subseteq C$ to say that, when regarded as ABoxes (sets of atoms), the cactus C' is (isomorphic to) a subset of

the cactus C. A cactus $C \in \mathfrak{K}^{+}_{\mathbf{Q}}$ is *minimal* if, for every $C' \in \mathfrak{K}^{+}_{\mathbf{Q}}$, $C' \subseteq C$ implies $C' = C$. The class of minimal cactuses in $\mathfrak{K}^{+}_{\mathbf{Q}}$ is denoted by $\mathfrak{K}^{min}_{\mathbf{Q}}$. It should be clear that (13) holds for $\mathfrak{K}^{min}_{\mathbf{Q}}$ in place of $\mathfrak{K}_{\mathbf{Q}}$.

Example 19. Consider the d-sirup $\mathbf{Q} = (\mathsf{cov}_A, \boldsymbol{q})$ with $\boldsymbol{q}$ shown on the left-hand side of the picture below (the R-labels on the edges are omitted). The cactus C_1 is obtained by budding y_1 in $\boldsymbol{q}$, and C_2 is obtained by budding y_2 in C_1.



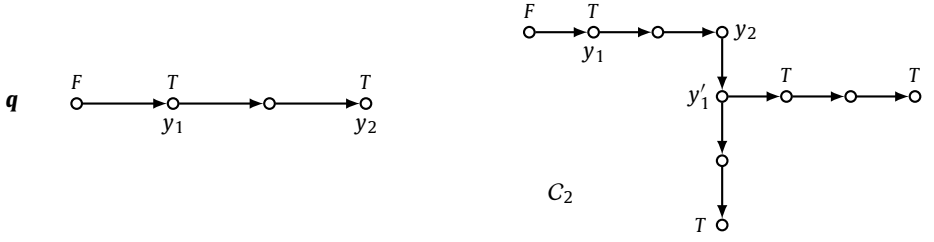
Let C_1^{-} be the result of removing $T(y_2)$ from C_1. Then $\mathsf{cov}_A, C_1^{-} \models \boldsymbol{q}$, the pruned cactus C_1^{-} is minimal, while $C_2 \supset C_1^{-}$ is not. Based on this observation, one can show that the skeleton of each cactus in $\mathfrak{K}^{min}_{\mathbf{Q}}$ has only one branch.

The branching number [34] of a rooted tree $\mathfrak{T}$ is defined as follows. For any node u in $\mathfrak{T}$, we compute inductively its *branching rank* $\boldsymbol{br}(u)$ by taking $\boldsymbol{br}(u) = 0$ if u is a leaf and, for a non-leaf u,

$$\boldsymbol{br}(u) = \begin{cases} m+1, & \text{if } u \text{ has} \geq 2 \text{ children of branching rank } m; \\ m, & \text{otherwise,} \end{cases} \tag{14}$$

where m is the maximum of the branching ranks of u's children. The *branching number of* $\mathfrak{T}$ is the branching rank of its root node. (In other words, the branching number of $\mathfrak{T}$ is $\boldsymbol{b}$ if the largest full binary tree that is a minor of $\mathfrak{T}$ is of depth $\boldsymbol{b}$.) The *branching number of a cactus* $C \in \mathfrak{K}^{+}_{\mathbf{Q}}$ is the branching number of C^{s}. We call $\mathfrak{K}^{min}_{\mathbf{Q}}$ *boundedly branching* if there is some $\boldsymbol{b} < \omega$ such that $\mathfrak{K}^{min}_{\mathbf{Q}}$ contains a cactus with branching number $\boldsymbol{b}$ but no cactus of greater branching number. Otherwise, we call $\mathfrak{K}^{min}_{\mathbf{Q}}$ *unboundedly branching*.

Example 20. The branching number of each cactus in $\mathfrak{K}^{min}_{\mathbf{Q}}$ from Example 19 is 0; however, there are cactuses in $\mathfrak{K}_{\mathbf{Q}}$ with an arbitrarily large branching number $\boldsymbol{b} < \omega$. As another instructive example, consider the 1-CQ $\boldsymbol{q}$ depicted on the left-hand side below. The cactus C_2 for $\mathbf{Q} = (\mathsf{cov}_\top, \boldsymbol{q})$ on the right-hand side, obtained by first budding y_2 and then



y_1', can be pruned at y_1 by removing $T(y_1)$ (since *every* node in a model of $\mathsf{cov}_\top$ is labelled by F or T). Using this observation, one can show that every cactus in $\mathfrak{K}^{min}_{\mathbf{Q}}$ has branching number ≤ 1. On the other hand, if $\mathbf{Q} = (\mathsf{cov}_A, \boldsymbol{q})$, then $\mathfrak{K}^{min}_{\mathbf{Q}}$ is unboundedly branching as follows from Theorems 21 and 26.

Theorem 21. *Every (d)d-sirup* $\mathbf{Q} = (\mathcal{O}, \boldsymbol{q})$ *with a* 1*-CQ* $\boldsymbol{q}$ *and boundedly branching* $\mathfrak{K}^{min}_{\mathbf{Q}}$ *is linear-datalog-rewritable, and so can be answered in* NL.

Proof. Similarly to [45], we represent cactus-like ABoxes as terms of a tree alphabet and construct a tree automaton $\mathfrak{A}_{\mathbf{Q}}$ such that (*i*) cactuses in $\mathfrak{K}^{min}_{\mathbf{Q}}$ are accepted by $\mathfrak{A}_{\mathbf{Q}}$, and (*ii*) for every ABox $\mathcal{A}$ accepted by $\mathfrak{A}_{\mathbf{Q}}$, we have $\mathcal{O}, \mathcal{A} \models \boldsymbol{q}$. Then, using ideas of [34], we show that if $\mathfrak{K}^{min}_{\mathbf{Q}}$ is boundedly branching, then the automaton $\mathfrak{A}_{\mathbf{Q}}$ can be transformed into a (monadic) linear-stratified datalog rewriting of $\mathbf{Q}$. As shown in [83], such a rewriting can further be converted into a linear datalog rewriting (at the expense of increasing the arity of IDB predicates in the program).

We only consider the case $\mathcal{O} = \mathsf{cov}_A$ leaving a similar proof for $\mathsf{cov}_A^{\perp}$ to the reader. As before, we assume that $\boldsymbol{q}$ is a 1-CQ such that $F(x)$ and $T(y_1), \dots, T(y_n)$ are all of the solitary occurrences of F and T in $\boldsymbol{q}$.

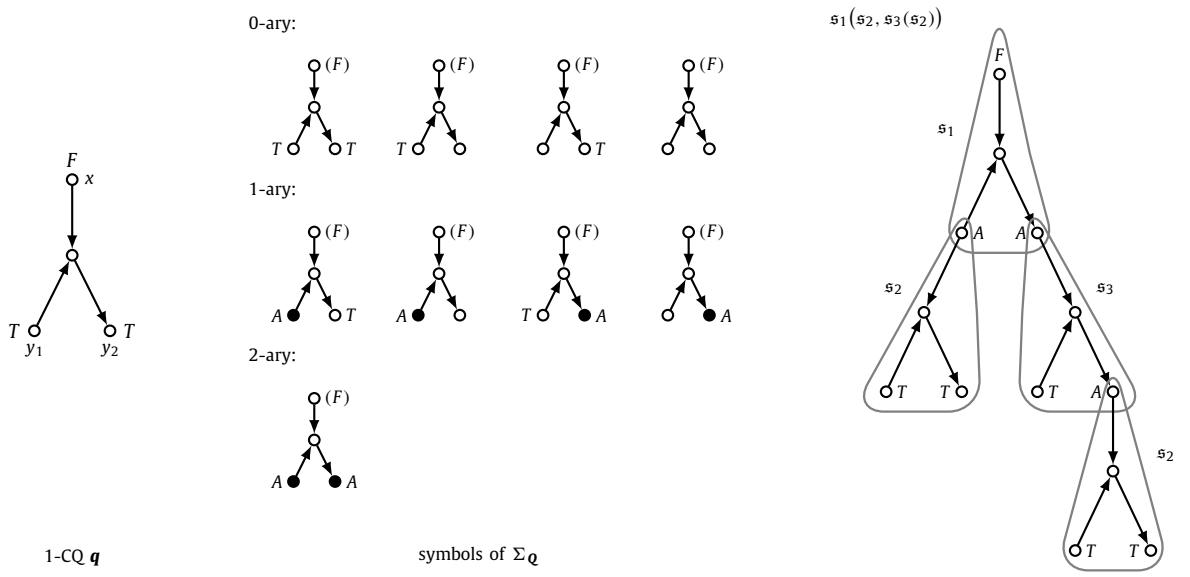


Fig. 2. An example of a tree alphabet $\Sigma_{\boldsymbol{q}}$ and a cactus as a $\Sigma_{\boldsymbol{q}}$-tree.

Recall from [97] that a *tree alphabet* is a finite set Σ of symbols, each of which is associated with a natural number, its *arity*. A Σ-*tree* is any ground term built up inductively, using the symbols of Σ as functions: 0-ary symbols in Σ are Σ-trees and, for any k-ary $\mathfrak{a}$ in Σ and Σ-trees $C_1, \dots, C_k$, the term $\mathfrak{a}(C_1, \dots, C_k)$ is a Σ-tree. The *branching number of a* Σ-*tree* is that of its parse tree. We define a tree alphabet $\Sigma_{\boldsymbol{q}}$ as follows. Consider cactus-like ABoxes that are built from $\boldsymbol{q}$ using **(bud)** and **(prune)**, with applications of the latter also allowed when $\mathsf{cov}_A, C^- \not\models \boldsymbol{q}$ for the resulting ABox C^-, and extend the notions of skeleton, branching number and segments to these in the natural way. The symbols of $\Sigma_{\boldsymbol{q}}$ are the segments $\mathfrak{s}$ of such ABoxes, with the arity of $\mathfrak{s}$ being the number of its budding nodes, and with the x-node of $\mathfrak{s}$ being either labelled by F or not. Then each cactus in $\mathfrak{K}^+_{\boldsymbol{q}}$ can be encoded by some $\Sigma_{\boldsymbol{q}}$-tree; see Fig. 2 for an example. On the other hand, every $\Sigma_{\boldsymbol{q}}$-tree represents some cactus-like ABox. So, with a slight abuse of terminology, from now on by a $\Sigma_{\boldsymbol{q}}$-tree we mean either the corresponding term or ABox.

However, such an ABox C is not necessarily a cactus in $\mathfrak{K}^+_{\boldsymbol{q}}$ for two possible reasons: either $\mathsf{cov}_A, C \not\models \boldsymbol{q}$ or C having F-nodes in some 'wrong' segments (every cactus has a unique F-node, viz., the x-node of its root segment). We are interested in those $\Sigma_{\boldsymbol{q}}$-trees C for which $\mathsf{cov}_A, C \models \boldsymbol{q}$. To capture them, we use tree automata [97]. A *nondeterministic finite tree automaton* (NTA) over a tree alphabet Σ is a quadruple $\mathfrak{A} = (Q, Q_f, \Delta, \Sigma)$, where Q is a finite set of *states*, $Q_f \subseteq Q$ is a set of *final* states, and Δ is a set of *transitions* of the form $q_1, \dots, q_k \Rightarrow^{\mathfrak{a}} q$, where $k \geq 1$ is the arity of $\mathfrak{a} \in \Sigma$ and $q_1, \dots, q_k, q \in Q$; for symbols $\mathfrak{a}$ of arity 0, we have *initial transitions* in Δ of the form $\Rightarrow^{\mathfrak{a}} q$. A *run* of $\mathfrak{A}$ on a Σ-tree C is a labelling function r from the subterms of C to Q satisfying the following condition: for any subterm $C^- = \mathfrak{a}(C_1, \dots, C_k)$ of C, there is a transition $q_1, \dots, q_k \Rightarrow^{\mathfrak{a}} q$ in Δ such that $r(C_1) = q_1, \dots, r(C_k) = q_k$ and $r(C^-) = q$ (in which case we say that the transition is *used in* r). A Σ-tree C is *accepted* by $\mathfrak{A}$ if there is a run of $\mathfrak{A}$ on C that labels C with a final state. Let $L(\mathfrak{A})$ be the set of all Σ-trees accepted by $\mathfrak{A}$. A set L of Σ-trees is called a *regular tree language* if $L = L(\mathfrak{A})$, for some NTA $\mathfrak{A}$ over Σ.

Claim 21.1. $L_{\boldsymbol{q}} = \{C \mid C \text{ is a } \Sigma_{\boldsymbol{q}}\text{-tree with } \mathsf{cov}_A, C \models \boldsymbol{q}\}$ *is a regular tree language.*

Proof. We proceed via a series of steps. In the construction, we use Theorem 11 for describing $L_{\boldsymbol{q}}$ by means of the datalog program $\Pi_{\boldsymbol{q}} = \{(9), (10), (11)\}$. We extend the tree alphabet $\Sigma_{\boldsymbol{q}}$ to a tree alphabet $\Sigma^e_{\boldsymbol{q}}$ as follows. For each symbol $\mathfrak{s}$ in $\Sigma_{\boldsymbol{q}}$, we label some (possibly none) of the nodes in segment $\mathfrak{s}$ by P. We call each resulting 'segment' $\mathfrak{s}^e$ an *extension of* $\mathfrak{s}$. (Each symbol in $\Sigma_{\boldsymbol{q}}$ might have several extensions, and each of them has the same arity as $\mathfrak{s}$.) Let $\Sigma^e_{\boldsymbol{q}}$ consist of all possible extensions of every $\mathfrak{s}$ in $\Sigma_{\boldsymbol{q}}$. We say that a $\Sigma^e_{\boldsymbol{q}}$-tree C^e is an *extension* of a $\Sigma_{\boldsymbol{q}}$-tree C if they have isomorphic tree structures, and each symbol $\mathfrak{s}^e$ in C^e is an extension of the corresponding symbol $\mathfrak{s}$ in C. For example, the closure $\Pi_{\boldsymbol{q}}(C)$ of any $\Sigma_{\boldsymbol{q}}$-tree C under $\Pi_{\boldsymbol{q}}$ is an extension of C.

For any $\Sigma^e_{\boldsymbol{q}}$-tree C^e, we write $C^e \models \boldsymbol{G}$, for the goal predicate $\boldsymbol{G}$ of $\Pi_{\boldsymbol{q}}$, if there is a homomorphism from $\boldsymbol{q}^e$ to C^e, where $\boldsymbol{q}^e = \boldsymbol{q} \setminus \{T(y_1), \dots, T(y_n)\} \cup \{P(y_1), \dots, P(y_n)\}$. We claim that each of the following is a regular tree language:

(a) the set of $\Sigma^e_{\boldsymbol{q}}$-trees C^e with $C^e \neq \Pi_{\boldsymbol{q}}(C^e)$;

(b) the set of $\Sigma^e_{\mathbf{Q}}$-trees C^e with $C^e \models \boldsymbol{G}$;
(c) the set of $\Sigma_{\mathbf{Q}}$-trees C that have some extension C^e with $C^e = \Pi_{\mathbf{Q}}(C^e)$ and $C^e \not\models \boldsymbol{G}$;
(d) the set of $\Sigma_{\mathbf{Q}}$-trees C with $\Pi_{\mathbf{Q}}, C \models \boldsymbol{G}$.

Indeed, to show (a), we need an NTA 'detecting a pattern' in the ABox C^e falsifying one of rules (10)–(11) in $\Pi_{\mathbf{Q}}$. Similarly, to show (b), we need an NTA 'detecting a pattern' in C^e corresponding to an application of rule (9) in $\Pi_{\mathbf{Q}}$. Now, (c) follows from (a), (b) and the fact that regular tree languages are closed under taking complements, intersections and linear homomorphisms [97] (as the 'forgetting' function substituting $\mathfrak{s}$ for each $\mathfrak{s}^e$ is a linear tree homomorphism from $\Sigma^e_{\mathbf{Q}}$-trees to $\Sigma_{\mathbf{Q}}$-trees, mapping any extension C^e to C). To show (d), take the complement of (c), and observe that $\Pi_{\mathbf{Q}}, C \models \boldsymbol{G}$ iff, for every extension C^e of C, whenever $C^e = \Pi_{\mathbf{Q}}(C^e)$ then $C^e \models \boldsymbol{G}$.

Finally, it follows from (d) and Theorem 11 that $L_{\mathbf{Q}}$ is a regular tree language. □

An NTA $\mathfrak{A} = (Q, Q_f, \Delta, \Sigma)$ is *linear-stratified* if there is a function $\boldsymbol{st}\colon Q \to \omega$ such that, for any transition $q_1, \dots, q_k \Rightarrow^{\mathfrak{a}} q$ in Δ,

- $\boldsymbol{st}(q_i) \leq \boldsymbol{st}(q)$, for every i, $1 \leq i \leq k$, and
- there is at most one i such that $1 \leq i \leq k$ and $\boldsymbol{st}(q_i) = \boldsymbol{st}(q)$.

Claim 21.2. *For any NTA $\mathfrak{A}$ and any $\boldsymbol{b} < \omega$, there is a linear-stratified NTA $\mathfrak{A}^s$ such that*

$$\{C \in L(\mathfrak{A}) \mid \textit{the branching number of } C \textit{ is} \leq \boldsymbol{b}\} \subseteq L(\mathfrak{A}^s) \subseteq L(\mathfrak{A}). \tag{15}$$

Proof. Suppose $\mathfrak{A} = (Q, Q_f, \Delta, \Sigma)$. We define $\mathfrak{A}^s = (Q^s, Q^s_f, \Delta^s, \Sigma)$ as follows. First, set $Q^s = Q \times \{0, \dots, \boldsymbol{b}\}$ and $Q_f = Q_f \times \{0, \dots, \boldsymbol{b}\}$. Then, for any transition of the form $\Rightarrow^{\mathfrak{a}} q$ in Δ, we add the transition $\Rightarrow^{\mathfrak{a}} (q, 0)$ to Δ^s. For any transition $q_1, \dots, q_k \Rightarrow^{\mathfrak{a}} q$ in Δ and any $m \leq \boldsymbol{b}$, we add to Δ^s all transitions $(q_1, m_1), \dots, (q_k, m_k) \Rightarrow^{\mathfrak{a}} (q, m)$ such that

- either $m_1, \dots, m_k < m$ and $m_i = m_j = m - 1$, for some $i \neq j$;
- or $m_i = m$, for some i, and $m_j < m$, for all $j \neq i$.

$\mathfrak{A}^s$ is linear-stratified as one can set $\boldsymbol{st}\big((q, m)\big) = m$, for $q \in Q$, $m \leq \boldsymbol{b}$. To show (15), observe that $L(\mathfrak{A}^s) \subseteq L(\mathfrak{A})$ since from every run r of $\mathfrak{A}^s$ on C we obtain a run of $\mathfrak{A}$ on C by replacing each $(q_1, m_1), \dots, (q_k, m_k) \Rightarrow^{\mathfrak{a}} (q, m)$ used in r with $q_1, \dots, q_k \Rightarrow^{\mathfrak{a}} q$. For the other inclusion, given a run r of $\mathfrak{A}$ on some C with branching number $\leq \boldsymbol{b}$, we obtain a run of $\mathfrak{A}^s$ on C by labelling each subterm C^- of C with state $\big(r(C^-), \boldsymbol{b}^-\big)$, where $\boldsymbol{b}^-$ is the branching number of C^-. □

We can now complete the proof of Theorem 21. Indeed, suppose that every cactus in $\mathfrak{K}^{min}_{\mathbf{Q}}$ has branching number at most $\boldsymbol{b} < \omega$. By Claims 21.1 and 21.2, there is a linear-stratified NTA $\mathfrak{A} = (Q, Q_f, \Delta, \Sigma_{\mathbf{Q}})$ such that

$$\{C \in L_{\mathbf{Q}} \mid \text{the branching number of } C \text{ is at most } \boldsymbol{b}\} \subseteq L(\mathfrak{A}) \subseteq L_{\mathbf{Q}}.$$

Using $\mathfrak{A}$, we construct a (monadic) linear-stratified program $\Pi_{\mathfrak{A}}$ with goal predicate $\boldsymbol{G}_{\mathfrak{A}}$ as follows. For every $q \in Q$, we introduce a fresh unary predicate P_q. For every final state $q \in Q_f$, the program $\Pi_{\mathfrak{A}}$ contains the rule

$$\boldsymbol{G}_{\mathfrak{A}} \leftarrow P_q(x). \tag{16}$$

For every transition $q_1, \dots, q_k \Rightarrow^{\mathfrak{s}} q$ in Δ, where the budding nodes in the k-ary segment $\mathfrak{s}$ are $y_{i_1}, \dots, y_{i_k}$, $\Pi_{\mathfrak{A}}$ contains

$$P_q(x) \leftarrow \mathfrak{s}, P_{q_1}(y_{i_1}), \dots, P_{q_k}(y_{i_k}). \tag{17}$$

As $\mathfrak{A}$ is linear-stratified, it is easy to see that the program $\Pi_{\mathfrak{A}}$ is linear-stratified. We claim that $(\Pi_{\mathfrak{A}}, \boldsymbol{G}_{\mathfrak{A}})$ is a datalog-rewriting of $\mathbf{Q}$, that is, for any ABox $\mathcal{A}$ (without the P_q), we have $\Pi_{\mathfrak{A}}, \mathcal{A} \models \boldsymbol{G}_{\mathfrak{A}}$ iff $\mathrm{cov}_A, \mathcal{A} \models \boldsymbol{q}$.

($\Leftarrow$) By Theorem 11, there is a homomorphism $h\colon C \to \mathcal{A}$, for some $C \in \mathfrak{K}_{\mathbf{Q}}$. As C always contains some $C' \in \mathfrak{K}^{min}_{\mathbf{Q}}$, we may assume that $C \in \mathfrak{K}^{min}_{\mathbf{Q}}$, and so C has branching number $\leq \boldsymbol{b}$. As $\mathrm{cov}_A, C \models \boldsymbol{q}$ clearly holds for every $C \in \mathfrak{K}^{min}_{\mathbf{Q}}$, it follows that $C \in L_{\mathbf{Q}}$, and so $C \in L(\mathfrak{A})$. Let r be an accepting run of $\mathfrak{A}$ on C. We construct a derivation of $\boldsymbol{G}_{\mathfrak{A}}$ in $\Pi_{\mathfrak{A}}(\mathcal{A})$ by induction on C as a $\Sigma_{\mathbf{Q}}$-tree, moving from leaves to the root. For every segment $\mathfrak{s}$ in C, if the transition $q_1, \dots, q_k \Rightarrow^{\mathfrak{s}} q$ is used in r then we apply (17) with the substitution of $h(z)$ for any node z in $\mathfrak{s}$. Also, if $r(C) = q$, for some final state q of $\mathfrak{A}$, then we apply (16) with the substitution $h(x_{\mathfrak{s}_0})$, where $x_{\mathfrak{s}_0}$ is the x-node of the root segment $\mathfrak{s}_0$ in C. It follows that $\Pi_{\mathfrak{A}}, \mathcal{A} \models \boldsymbol{G}_{\mathfrak{A}}$.

($\Rightarrow$) By induction on a derivation of $\boldsymbol{G}_{\mathfrak{A}}$, we construct a $\Sigma_{\mathbf{Q}}$-tree $\mathcal{B}$, an accepting run r of $\mathfrak{A}$ on $\mathcal{B}$, and a homomorphism $f\colon \mathcal{B} \to \mathcal{A}$. To begin with, there is an object x^a for which (16) was triggered for some $q \in Q_f$. Then $P_q(x^a)$ was deduced by an application of (17) for some $\mathfrak{s}$. If this $\mathfrak{s}$ is 0-ary, then s is a $\Sigma_{\mathbf{Q}}$-tree (of depth 0), the function r labelling $\mathfrak{s}$ with q is an accepting run on $\mathfrak{s}$, and the substitution f_0 used in (17) is a homomorphism from $\mathfrak{s}$ to $\mathcal{A}$. If $\mathfrak{s}$ is k-ary, for some $k > 0$, then there are $y^a_{i_1}, \dots, y^a_{i_k}$ for which (17) was triggered. For each $j = 1, \dots, k$, consider the rule

$$P_{q_j}(x) \leftarrow \mathfrak{s}^j, P_{q_1^j}(y_{i_1}), \dots, P_{q_{k_j}^j}(y_{i_{k_j}})$$

by which $P_{q_j}(y^a_{i_j})$ was deduced. Take the ABox $\mathcal{B}$ built up by glueing the x node of each segment $\mathfrak{s}^j$ to the y_{i_j} node of $\mathfrak{s}$, extend r by labelling each $\mathfrak{s}^j$ with q_j, and extend f_0 to a $\mathcal{B} \to \mathcal{A}$ homomorphism by taking the substitutions used in the rules. Now, if every $\mathfrak{s}^j$ is 0-ary, then $\mathcal{B}$ is a $\Sigma_{\boldsymbol{q}}$-tree and we are done. Otherwise, repeat the above procedure for the 'arguments' of each $\mathfrak{s}^j$ of arity > 0. As the derivation of $\boldsymbol{G}_{\mathfrak{A}}$ is finite, sooner or later the procedure stops, as required.

As $\mathcal{B} \in L(\mathfrak{A}) \subseteq L_{\boldsymbol{q}}$, by Theorem 11 there exists a homomorphism $h\colon C \to \mathcal{B}$, for some cactus $C \in \mathfrak{K}_{\boldsymbol{q}}$. Then the composition of h and f is a homomorphism from C to $\mathcal{A}$, and so $\mathsf{cov}_A, \mathcal{A} \models \boldsymbol{q}$ by Theorem 11, as required. □

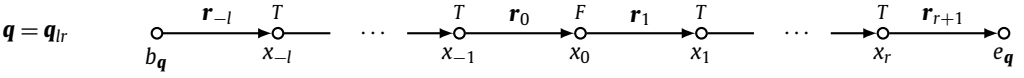
We do not know if the sufficient condition in Theorem 21 of linear-datalog-rewritability of (d)d-sirups with a 1-CQ is also a necessary one. As follows from [34,35], this is so for ditree 1-CQs with root labelled by F; see Example 18. We use Theorem 21 in the next section to show that answering path-shaped dd-sirups with a certain periodic structure can be done in NL.

5. AC^0 / NL / P / coNP-tetrachotomy of dd-sirups with a path CQ

We now obtain the main result of this article: a complete syntactic classification of dd-sirups $(\mathsf{cov}^{\perp}_A, \boldsymbol{q})$ with a path-shaped CQ $\boldsymbol{q}$ according to their data complexity and rewritability type. While the AC^0/NL part of this AC^0/NL/P/coNP-tetrachotomy follows from our earlier results, proving P- and especially coNP-hardness turns out to be tough and requires the development of novel machinery.

From now on, we only consider path CQs $\boldsymbol{q}$ (whose digraph is path-shaped). Solitary F- and T-nodes will simply be called F- and T-*nodes*, respectively. We denote the first (root) node in $\boldsymbol{q}$ by $b_{\boldsymbol{q}}$ and the last (leaf) node by $e_{\boldsymbol{q}}$. Given nodes x and y, we write $x \prec y$ to say that there is a directed path from x to y in $\boldsymbol{q}$; as usual, $x \preceq y$ means $x \prec y$ or $x = y$. For $x \preceq y$, the set $[x, y]$ comprises those atoms in $\boldsymbol{q}$ whose variables are in the interval $\{z \mid x \preceq z \preceq y\}$ and $(x, y) = [x, y] \setminus \{T(x), F(x), T(y), F(y)\}$. For $\boldsymbol{i} = (x, y)$, we let $|\boldsymbol{i}|$ be the length of the path from x to y, and $|\boldsymbol{q}| = |(b_{\boldsymbol{q}}, e_{\boldsymbol{q}})|$.

We divide path CQs into three disjoint classes: the 0-CQs and the 1-CQs defined earlier, and the 2-*CQs* that contain at least two F-nodes and at least two T-nodes. As we saw in Section 3.2, dd-sirups $(\mathsf{cov}^{\perp}_A, \boldsymbol{q})$ with $\boldsymbol{q}$ containing FT-twins are always FO-rewritable. We split twinless 1-CQs further into *periodic* and *aperiodic* ones, only considering 1-CQs with a single F-node and at least one T-node (as the case with a single T-node and at least one F-node is symmetric). Given such a twinless 1-CQ $\boldsymbol{q}$ and natural numbers l, r with $l + r \geq 1$, we write $\boldsymbol{q} = \boldsymbol{q}_{lr}$ to say that $\boldsymbol{q}$ has l-many T-nodes $x_{-l} \prec \dots \prec x_{-1}$ that $\prec$-precede its only F-node x_0, and r-many T-nodes $x_1 \prec \dots \prec x_r$ that $\prec$-succeed x_0. For every i with $-l \leq i \leq r + 1$, we define a set $\boldsymbol{r}_i$ of binary atoms by taking $\boldsymbol{r}_i = (x_{i-1}, x_i)$, where $x_{-l-1} = b_{\boldsymbol{q}}$ and $x_{r+1} = e_{\boldsymbol{q}}$. Note that $\boldsymbol{r}_i \neq \emptyset$ for $-l < i < r + 1$, but $\boldsymbol{r}_{-l} = \emptyset$ if $b_{\boldsymbol{q}} = x_{-l}$ and $\boldsymbol{r}_{r+1} = \emptyset$ if $x_r = e_{\boldsymbol{q}}$.



Each $\boldsymbol{r}_i$ determines a finite sequence $\langle \boldsymbol{r}_i \rangle$ of binary predicate symbols. We call $\boldsymbol{q}$ *right-periodic* if $\boldsymbol{q} = \boldsymbol{q}_{0r}$ and either $r = 1$ or $\langle \boldsymbol{r}_i \rangle = \langle \boldsymbol{r}_1 \rangle$ for all $i = 1, \dots, r$ and $\langle \boldsymbol{r}_{r+1} \rangle = \langle \boldsymbol{r}_1 \rangle^* \lambda$ for some (possibly empty) prefix λ of $\langle \boldsymbol{r}_1 \rangle$. By taking a mirror image of this definition, we obtain the notion of *left-periodic* 1-CQ, in which case $\boldsymbol{q} = \boldsymbol{q}_{l0}$ and either $l = 1$ or $\langle \boldsymbol{r}_{-i} \rangle = \langle \boldsymbol{r}_0 \rangle$ for all $i = 1, \dots, l - 1$ and $\langle \boldsymbol{r}_{-l} \rangle = \lambda \langle \boldsymbol{r}_0 \rangle^*$ for some (possibly empty) suffix λ of $\langle \boldsymbol{r}_0 \rangle$. A twinless 1-CQ $\boldsymbol{q}$ is called *periodic* if it is either right- or left-periodic, and *aperiodic* otherwise.

Theorem 22 (***tetrachotomy***). *Let $\boldsymbol{Q}$ be any d-sirup with a twinless path CQ $\boldsymbol{q}$ or any dd-sirup with a path CQ $\boldsymbol{q}$. Then the following tetrachotomy holds* (*where the three 'if' can be replaced by 'iff' provided that* NL $\neq$ P $\neq$ coNP):

(AC^0) $\boldsymbol{Q}$ *is FO-rewritable and can be answered in* AC^0 *iff $\boldsymbol{q}$ is a* 0*-CQ or contains an FT-twin*; *otherwise,*
(NL) $\boldsymbol{Q}$ *is linear-datalog-rewritable and answering it is* NL*-complete if $\boldsymbol{q}$ is a periodic* 1*-CQ*;
(P) $\boldsymbol{Q}$ *is datalog-rewritable and answering it is* P*-complete if $\boldsymbol{q}$ is an aperiodic* 1*-CQ*;
(coNP) *answering* $\boldsymbol{Q}$ *is* coNP*-complete if $\boldsymbol{q}$ is a* 2*-CQ.*

The first item follows from Theorem 5 and the fact that AC$^0 \neq$ L. The upper bounds in the remaining three are given by Theorem 24, Corollary 12, and Theorem 4, respectively. The matching lower bounds are established by Theorems 23, 26 and 27 to be proved below. We begin with the following criterion:

Theorem 23. *If $\boldsymbol{q}$ is a twinless path* 1*-CQ, then answering* $(\mathsf{cov}_A, \boldsymbol{q})$ *and* $(\mathsf{cov}^{\perp}_A, \boldsymbol{q})$ *is* NL*-hard.*

Proof. The proof is by an FO-reduction of the NL-complete reachability problem for directed acyclic graphs (dags). We assume that there exist a T-node x and an F-node y in $\boldsymbol{q}$ with $x \prec y$ (the other case is symmetric) and without any F- or T-nodes between them. Given a dag $G = (V, E)$ with nodes $\mathfrak{s}, \mathfrak{t} \in V$, we construct a twinless ABox $\mathcal{A}_G$ as follows. Replace each edge $e = (\mathfrak{u}, \mathfrak{v}) \in E$ by a fresh copy $\boldsymbol{q}^e$ of $\boldsymbol{q}$ such that node x in $\boldsymbol{q}^e$ is renamed to $\mathfrak{u}$ with $T(\mathfrak{u})$ replaced by $A(\mathfrak{u})$, and node y is renamed to $\mathfrak{v}$ with $F(\mathfrak{v})$ replaced by $A(\mathfrak{v})$. Then $\mathcal{A}_G$ 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 $\mathsf{cov}_A, \mathcal{A}_G \models \boldsymbol{q}$ iff $\mathsf{cov}^{\perp}_A, \mathcal{A}_G \models \boldsymbol{q}$ (cf. (8)).

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

($\Leftarrow$) Suppose $\mathfrak{s} \not\to_G \mathfrak{t}$. Define a model $\mathcal{I}$ of cov_A and $\mathcal{A}_G$ by labelling with T the undecided A-nodes in $\mathcal{A}_G$ that are reachable from $\mathfrak{s}$ (via a directed path) and with F the remaining ones. By excluding all possible locations where the T-node x could be mapped, we see that there is no homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. Indeed, suppose $h(x)$ is in a copy $\boldsymbol{q}^e$ for some edge $e = (\mathfrak{u}, \mathfrak{v}) \in E$. Then $h(x)$ cannot precede $\mathfrak{u}$ or succeed $\mathfrak{v}$ in $\boldsymbol{q}^e$, otherwise there is not enough room in $\boldsymbol{q}^e$ for the rest of $\boldsymbol{q}$ to be mapped. And $h(x)$ cannot be between $\mathfrak{u}$ and $\mathfrak{v}$ either, as $\boldsymbol{q}$ is twinless and there are no T-nodes between x and y in $\boldsymbol{q}$. If $h(x) = \mathfrak{u}$, then we exclude all options where the F-node y could be mapped: $h(y)$ cannot succeed $\mathfrak{u}$ in $\boldsymbol{q}^{(\mathfrak{u}', \mathfrak{u})}$ for any edge $(\mathfrak{u}', \mathfrak{u})$ because of the lack of room in $\boldsymbol{q}^{(\mathfrak{u}', \mathfrak{u})}$, and $h(y) = \mathfrak{v}'$ cannot hold in $\boldsymbol{q}^{(\mathfrak{u}, \mathfrak{v}')}$ for any edge $(\mathfrak{u}, \mathfrak{v}') \in E$ because such a $\mathfrak{v}'$ is labelled by T in $\mathcal{I}$. For similar reasons, $h(x) = \mathfrak{v}$ cannot happen either. □

A generalisation of this theorem to d-sirups with ditree-shaped 1-CQs possibly containing FT-twins has been proved in [44] using a much more involved construction; see also Example 25 below.

By Corollary 12, all (d)d-sirups with a 1-CQ are datalog-rewritable and can be answered in P. Our next task is to establish an NL/P dichotomy for d-sirups with a twinless path 1-CQ.

Theorem 24. *If $\boldsymbol{q}$ is a periodic twinless path* 1*-CQ, then both* $(\mathsf{cov}_A, \boldsymbol{q})$ *and* $(\mathsf{cov}^{\perp}_A, \boldsymbol{q})$ *are linear-datalog-rewritable, and so can be answered in* NL.

Proof. We use the notation above, and only consider the case when $\boldsymbol{Q} = (\mathsf{cov}_A, \boldsymbol{q})$ and $\boldsymbol{q} = \boldsymbol{q}_{0r}$ is a right-periodic twinless path 1-CQ with a single F-node x_0 and T-nodes $x_1, \dots, x_r$. We show that every cactus in $\mathfrak{K}^{min}_{\boldsymbol{Q}}$ has branching number at most 1 and use Theorem 21. If $r = 1$, then the cactuses in $\mathfrak{K}^{min}_{\boldsymbol{Q}}$ have branching number 0.

So suppose $r \geq 2$ and $C \in \mathfrak{K}^{min}_{\boldsymbol{Q}}$. For nodes u, v in C, we write $u \prec_C v$ to say that there is a directed path from u to v in the (acyclic) digraph C. We call a node in C a *T-copy* if it is a copy of a T-node x_i of $\boldsymbol{q}$ for some $i = 1, \dots, r$. There can be three kinds of T-copies: those that were budded while constructing C are labelled by A, those that were pruned have no label, and the rest are labelled by T. Observe first that

$$\text{if some } T\text{-copy } u \text{ is unlabelled in } C, \text{ then there is no } T\text{-copy } v \text{ such that } u \prec_C v \text{ and } v \text{ is labelled by } T. \tag{18}$$

Indeed, consider any model $\mathcal{I}$ of cov_A and C in which all A-nodes u' with $u \prec_C u'$ are in $T^{\mathcal{I}}$. As $\mathsf{cov}_A, C \models \boldsymbol{q}$, there is a homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. As x_0 is an F-node in $\boldsymbol{q}$ and its copy $x_0^{\mathfrak{s}_0}$ in the root segment $\mathfrak{s}_0$ of C is the only F-node in C, it follows from our assumption on $\mathcal{I}$ that $u \not\prec_C h(x_0)$. We show that $u \not\prec_C h(x_i)$, for any $i \leq r$. Indeed, this is clear if $h(x_0) \not\prec_C u$. So suppose $h(x_0) \prec_C u \prec_C h(x_r)$. Then, either $h(x_0) = x_0^{\mathfrak{s}_0}$ or $h(x_0)$ is a budded T-copy $\prec_C$-preceding u. By $\boldsymbol{q}$ being right-periodic, every T-copy on the path from $h(x_0)$ to $h(x_r)$ in C different from $h(x_0)$ must be labelled by T. However, this is not the case for u, which is a contradiction. As $u \not\prec_C h(x_i)$ for any $i \leq r$, by using **(bud)** and **(prune)** we can construct a cactus $C_1 \in \mathfrak{K}^{+}_{\boldsymbol{Q}}$ that is the same as C apart from all T-labelled T-copies u' with $u \prec_C u'$ being unlabelled. Then $C_1 \subseteq C$, and so $C \in \mathfrak{K}^{min}_{\boldsymbol{Q}}$ implies that $C = C_1$, proving (18).

Next, consider any branch $\mathfrak{s}_0, \dots, \mathfrak{s}_{n-1}, \mathfrak{s}_n$ in the skeleton C^s of C such that there are no A-nodes in the segment $\mathfrak{s}_{n-1}$ $\prec_C$-succeeding the A-labelled T-copy w that has been budded to obtain the leaf segment $\mathfrak{s}_n$. Let π be the path in C from the root node of $\mathfrak{s}_0$ to the leaf node of $\mathfrak{s}_n$. We claim that

$$\text{all } T\text{-copies in } \pi \text{ are labelled by either } T \text{ or } A. \tag{19}$$

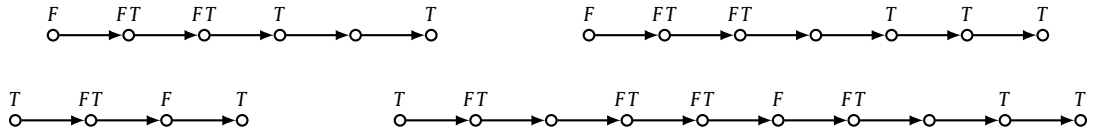
Indeed, by (18), it is enough to show that all T-copies in $\mathfrak{s}_n$ are labelled by T. Suppose on the contrary that at least one of them is not. Let $\mathcal{I}$ be a model of cov_A and C where the A-node w is labelled by F. Then there is a homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$ such that $h(x_i) \neq w$ for any $i = 0, \dots, r$. Thus, by using **(bud)** and **(prune)** we can construct a cactus $C_2 \in \mathfrak{K}^{+}_{\boldsymbol{Q}}$ that is the same as C apart from w in $\mathfrak{s}_{n-1}$ not being budded but pruned (and so w is unlabelled in C_2 and $\mathfrak{s}_n$ is not a segment in C_2^s). Then $C_2 \subseteq C$ but $C_2 \neq C$, contrary to $C \in \mathfrak{K}^{min}_{\boldsymbol{Q}}$.

Now (19) and $\boldsymbol{q}$ being right-periodic imply that, for any model $\mathcal{I}$ of cov_A and C, there is a homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$ mapping x_0 and $(x_0, \mathbf{e}_{\boldsymbol{q}})$ into π. Indeed, take $h(x_0) = z$ where z is the $\prec_C$-last F-labelled A-node in π if there is such, and $x_0^{\mathfrak{s}_0}$ otherwise. By the definition of budding, the part of $\boldsymbol{q}$ $\prec$-preceding x_0 can be mapped to $\mathcal{I}$ as well, possibly covering some parts of C not in π but in a child-segment of some of the $\mathfrak{s}_i$. Therefore, by using **(bud)** and **(prune)** we can construct a cactus $C_3 \in \mathfrak{K}^{+}_{\boldsymbol{Q}}$ whose skeleton consists of the branch $\mathfrak{s}_0, \dots, \mathfrak{s}_n$ and all other children of the segments $\mathfrak{s}_i$ for $i = 0, \dots, n-1$,

the T-copies that were labelled by A and budded further in C in some of these children are unlabelled in C_3, and all other T-copies are the same in C_3 and C. Then $C_3 \subseteq C$ and the branching number of C_3 is at most 1. As $C \in \mathfrak{K}_{\boldsymbol{q}}^{min}$, $C = C_3$ follows. □

One can generalise the proof of Theorem 24 to various path 1-CQs *with* FT-twins. Here are some examples.

Example 25. We invite the reader to show that answering the d-sirups with the following 1-CQs is NL-complete:



We next show that answering any d-sirup with twinless path 1-CQs not covered by Theorem 24 is P-hard.

Theorem 26. *If* $\boldsymbol{q}$ *is an aperiodic twinless path* 1*-CQ, then answering both* $(\mathsf{cov}_A, \boldsymbol{q})$ *and* $(\mathsf{cov}_A^{\perp}, \boldsymbol{q})$ *is* P*-hard.*

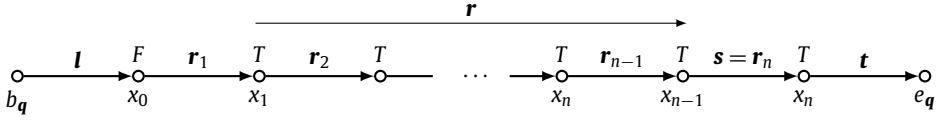
Proof. The theorem is proved by an FO-reduction of the monotone circuit evaluation problem, which is known to be P-complete [98]. We remind the reader that a *monotone Boolean circuit* is a directed acyclic graph $\boldsymbol{C}$ whose vertices are called *gates*. Gates with in-degree 0 are *input gates*. Each non-input gate g is either an AND-gate or an OR-gate, and has in-degree 2 (with the two edges coming in from gates we call the *inputs of* g). One of the non-input gates is distinguished as the *output gate*. Given an assignment α of F and T to the input gates of $\boldsymbol{C}$, we compute the value of each gate in $\boldsymbol{C}$ under α as usual in Boolean logic. The *output* $\boldsymbol{C}(\alpha)$ *of* $\boldsymbol{C}$ *on* α is the truth-value of the output gate. For every monotone Boolean circuit $\boldsymbol{C}$ and every assignment α, we construct a twinless ABox $\mathcal{A}_{\boldsymbol{C},\alpha}$ whose size is polynomial in the sizes of $\boldsymbol{q}$ and $\boldsymbol{C}$, and then show that $\boldsymbol{C}(\alpha) = T$ iff $\mathsf{cov}_A, \mathcal{A}_{\boldsymbol{C},\alpha} \models \boldsymbol{q}$ iff $\mathsf{cov}_A^{\perp}, \mathcal{A}_{\boldsymbol{C},\alpha} \models \boldsymbol{q}$ (cf. (8)).

We prove the theorem for aperiodic 1-CQs with a single F-node (the other case is symmetric). Suppose $\boldsymbol{q} = \boldsymbol{q}_{lr}$ for some l, r with $l + r \geq 1$. Then there can be three reasons for $\boldsymbol{q}$ being aperiodic: (i) $l = 0$ and $\boldsymbol{q}$ is not right-periodic, (ii) $r = 0$ and $\boldsymbol{q}$ is not left-periodic, or (iii) $l, r \geq 1$. In each of the three cases (i)–(iii), we give a different reduction.

(i) If $\boldsymbol{q} = \boldsymbol{q}_{0r}$ and $\boldsymbol{q}$ is not right-periodic, then $r \geq 2$. We let

$$n = \begin{cases} r, & \text{if } \langle \boldsymbol{r}_1 \rangle = \langle \boldsymbol{r}_2 \rangle = \cdots = \langle \boldsymbol{r}_r \rangle; \\ \min\{i \mid 1 < i \leq r \text{and} \langle \boldsymbol{r}_i \rangle \neq \langle \boldsymbol{r}_1 \rangle\}, & \text{otherwise.} \end{cases}$$

Then $n \geq 2$. The ABox $\mathcal{A}_{\boldsymbol{C},\alpha}$ is built up from isomorphic copies of the following intervals: $\boldsymbol{l} = (b_{\boldsymbol{q}}, x_0)$, $\boldsymbol{r}_1 = (x_0, x_1)$, $\boldsymbol{r} = (x_1, x_{n-1})$, $\boldsymbol{s} = (x_{n-1}, x_n)$, and $\boldsymbol{t} = (x_n, e_{\boldsymbol{q}})$. Note that $\boldsymbol{s}$ is nonempty and has no T-nodes. On the other hand, $\boldsymbol{l}$ can be empty when $b_{\boldsymbol{q}} = x_0$, $\boldsymbol{r}$ can be empty when $n = 2$, and $\boldsymbol{t}$ can be empty when $x_n = e_{\boldsymbol{q}}$.



We use the gadgets in Fig. 3 to simulate the AND- and OR-gates. For AND-gates, we distinguish between two cases $|\boldsymbol{s}| > |\boldsymbol{r}_1|$ and $|\boldsymbol{s}| \leq |\boldsymbol{r}_1|$, while the gadget for OR-gates is the same in both cases. Throughout, in our pictures of ABoxes, lower case letters like $a, b, z, \ldots$ are just pointers, not actual labels of nodes. In Fig. 3, if $\boldsymbol{r} = \emptyset$ then $z = a'$ and $z' = b$ are labelled only by A.

Given a monotone circuit $\boldsymbol{C}$ and an assignment α, we construct $\mathcal{A}_{\boldsymbol{C},\alpha}$ as follows. With each non-input gate g we associate a fresh copy of its gadget. When the inputs of g are gates g_a and g_b then, for each $i = a, b$, if g_i is a non-input gate, then we merge node c of the gadget for g_i with node i in the gadget for g; and if g_i is an input gate, we replace the label A of i and i' (if available) in the gadget for g with $\alpha(g_i)$. Finally, we replace the label A of node c in the gadget for the output gate with F. We claim that $\mathsf{cov}_A, \mathcal{A}_{\boldsymbol{C},\alpha} \models \boldsymbol{q}$ iff $\boldsymbol{C}(\alpha) = T$.

$(\Leftarrow)$ is proved by induction on the number of non-input gates in $\boldsymbol{C}$. The basis is obvious. For the induction step, suppose the output gate g in $\boldsymbol{C}$ is an AND-gate with inputs g_a and g_b, at least one of which is a non-input gate. Let $\mathcal{I}$ be an arbitrary model of cov_A and $\mathcal{A}_{\boldsymbol{C},\alpha}$. If both a and b in the gadget for g are in $T^{\mathcal{I}}$, then it is easy to check that we always have a $\boldsymbol{q} \to \mathcal{I}$ homomorphism, no matter what the labels of a' and b' (if available) are. It remains to consider the case when either a or b is in $F^{\mathcal{I}}$, and so the corresponding g_i is not an input gate. Take the subcircuit $\boldsymbol{C}^-$ of $\boldsymbol{C}$ whose output gate is g_i. Then $\mathcal{A}_{\boldsymbol{C}^-,\alpha}$ is the sub-ABox of $\mathcal{A}_{\boldsymbol{C},\alpha}$ with node c in the gadget for g_i as its topmost node, and $A(c)$ replaced by $F(c)$. Now, if $\mathcal{I}^-$ is the restriction of $\mathcal{I}$ to $\mathcal{A}_{\boldsymbol{C}^-,\alpha}$ (and so $c \in F^{\mathcal{I}^-}$), then by IH there is a $\boldsymbol{q} \to \mathcal{I}^-$ homomorphism, and so $\mathcal{I} \models \boldsymbol{q}$ as well. The case when the output gate g in $\boldsymbol{C}$ is an OR-gate is similar.

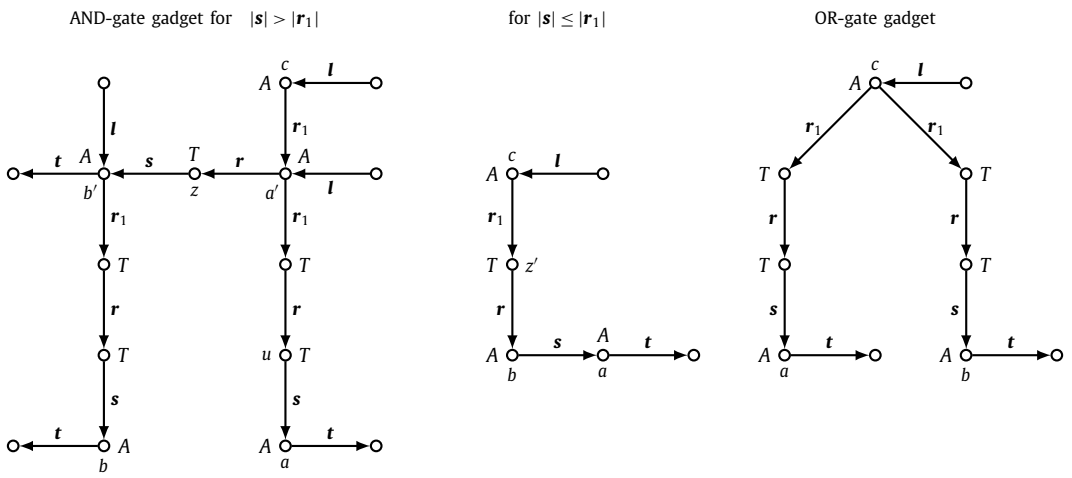


Fig. 3. Gate gadgets in case (*i*).

($\Rightarrow$) Suppose $\boldsymbol{C}(\alpha) = F$. To show $\mathsf{cov}_A, \mathcal{A}_{\boldsymbol{C},\alpha} \not\models \boldsymbol{q}$, we define a model $\mathcal{I}$ of cov_A and $\mathcal{A}_{\boldsymbol{C},\alpha}$ inductively by labelling the A-nodes in the gadget for each non-input gate g of $\boldsymbol{C}$ by F or T as follows: node c is labelled by the truth-value of g under α, while node i (and node i' if applicable), for $i = a, b$, is labelled by the truth-value of g_i under α, where g_a and g_b are the inputs of g. Suppose, on the contrary, that there is a homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. We exclude all options for the image $h(\boldsymbol{q})$ of $\boldsymbol{q}$. To this end, we track possible locations for $h(x_0) \in F^{\mathcal{I}}$. Let the non-input gate g be such that $h(x_0)$ is in the gadget for g and the inputs of g are gates g_a and g_b. We may assume that $h(x_0)$ is different from nodes a and b, because if $h(x_0) = i$ for $i \in \{a, b\}$ then g_i must be a non-input gate (otherwise there is no room for $h(\boldsymbol{q})$ in $\mathcal{A}_{\boldsymbol{C},\alpha}$), and so $h(x_0) = c$ in the gadget for g_i.

Suppose first that g is an AND-gate and $|\boldsymbol{s}| > |\boldsymbol{r}_1|$. We have the following cases:

$a, a' \in T^{\mathcal{I}}$, $b, b', c \in F^{\mathcal{I}}$**:** If $h(x_0) = c$, then $h(x_1) = a'$ and, since $b' \in F^{\mathcal{I}}$, $h(\boldsymbol{q})$ cannot continue 'horizontally' towards b'. But then, since $|\boldsymbol{s}| > |\boldsymbol{r}_1|$, the node $h(x_n)$ must be strictly between u and a which is impossible because there are no T-nodes in $\boldsymbol{s}$. We cannot have $h(x_0) = b'$ because $b \in F^{\mathcal{I}}$ and there is no room for $h(\boldsymbol{q})$ in $\boldsymbol{t}$.

$a, a', c \in F^{\mathcal{I}}$, $b, b' \in T^{\mathcal{I}}$**:** If $h(x_0) = a'$ then, since $a \in F^{\mathcal{I}}$, $h(\boldsymbol{q})$ cannot continue 'vertically' towards a. Then $h(x_1)$ is the central T-node. But then, since $|\boldsymbol{s}| > |\boldsymbol{r}_1|$, the node $h(x_{n-1})$ must be strictly between b' and z, which is impossible because there are no T-nodes in $\boldsymbol{s}$.

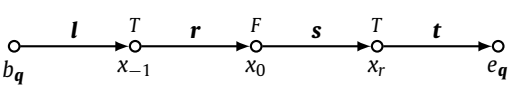
$a, a', b, b', c \in F^{\mathcal{I}}$**:** This case is covered by the previous ones.

Suppose next that g is an AND-gate and $|\boldsymbol{s}| \le |\boldsymbol{r}_1|$. Then $h(x_0) = c$ and $h(x_{n-1}) = b$, provided that $b \in T^{\mathcal{I}}$ (otherwise such h is impossible), which means that $a \in F^{\mathcal{I}}$, and so $h(x_n)$ is located in some other gadget g' whose node c is merged with the current $b = h(x_{n-1})$. However, this is impossible because of the following. In every gadget, the 'edges' leaving node c are all labelled by $\boldsymbol{r}_1$. As x_{n-1} is '$\boldsymbol{s}$-connected' to x_n and $h(x_{n-1}) = c$ in the gadget for g', if $|\boldsymbol{s}| < |\boldsymbol{r}_1|$ then $h(x_n)$ must be strictly between c and the end-node of an $\boldsymbol{r}_1$-edge, but there are no T-nodes there. So suppose $|\boldsymbol{s}| = |\boldsymbol{r}_1|$. Then $h(x_n)$ is the end-node of an $\boldsymbol{r}_1$-edge in the gadget for g', and so $\langle \boldsymbol{s} \rangle = \langle \boldsymbol{r}_1 \rangle$. Now it follows from the definition of n and $\boldsymbol{s}$ that $n = r$ and $\langle \boldsymbol{r}_1 \rangle = \cdots = \langle \boldsymbol{r}_r \rangle = \langle \boldsymbol{s} \rangle$. As $h(x_r) = h(x_n)$ is the end-node of an $\boldsymbol{r}_1$-edge starting at c in the gadget for g', an inspection of the gate-gadgets shows that $\boldsymbol{r}_{r+1} = (x_r, e_{\boldsymbol{q}}) = \boldsymbol{t}$ must be mapped to a non-empty sequence of $\boldsymbol{r}_1$-intervals followed by $\boldsymbol{t}$ (either in the gadget for g', or in some subsequent gadgets). So $\langle \boldsymbol{r}_{r+1} \rangle$ must be a possibly empty sequence of $\langle \boldsymbol{r}_1 \rangle$s, possibly followed by a non-empty proper prefix of $\langle \boldsymbol{r}_1 \rangle$, contrary to $\boldsymbol{q}$ being not right-periodic.

Finally, if g is an OR-gate and $h(x_0) = c$ in the gadget for g, then both a and b of the gadget are in F^I, and so $h(x_n) \in F^{\mathcal{I}}$, which is a contradiction.

The proof of (*ii*) is a mirror image of the previous one.

(*iii*) If $\boldsymbol{q} = \boldsymbol{q}_{lr}$ and $l, r \ge 1$, then $\mathcal{A}_{\boldsymbol{C},\alpha}$ is built up from isomorphic copies of the following intervals: $\boldsymbol{l} = (b_{\boldsymbol{q}}, x_{-1})$, $\boldsymbol{r} = (x_{-1}, x_0)$, $\boldsymbol{s} = (x_0, x_r)$, and $\boldsymbol{t} = (x_r, e_{\boldsymbol{q}})$. Note that $\boldsymbol{r}$ is not empty and has no T-nodes, while $\boldsymbol{l}$ and $\boldsymbol{t}$ may be empty.



We use the gadgets in Fig. 4 to simulate the AND- and OR-gates. The number of A-nodes in the gadget for a non-output AND-gate exceeds $|\boldsymbol{q}| + 2$.

Given a monotone circuit $\boldsymbol{C}$ and an assignment α, we construct $\mathcal{A}_{\boldsymbol{C},\alpha}$ as follows. With each non-input gate g we associate a fresh copy of its gadget. When the inputs of g are gates g_a and g_b then, for each $i = a, b$, if g_i is a non-input gate, then

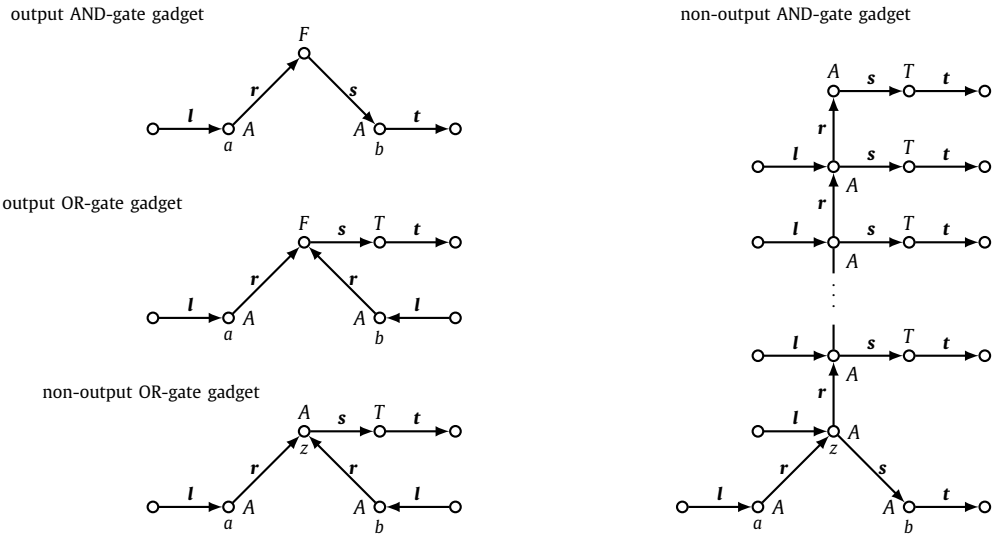


Fig. 4. Gate gadgets in case (*iii*).

we merge the topmost A-node of the gadget for g_i with node i in the gadget for g; and if g_i is an input gate, we replace the label A of i in the gadget for g with $\alpha(g_i)$. We claim that $\mathrm{cov}_A, \mathcal{A}_{\boldsymbol{C},\alpha} \models \boldsymbol{q}$ iff $\boldsymbol{C}(\alpha) = T$.

($\Leftarrow$) is proved by induction on the number of non-input gates in $\boldsymbol{C}$. The basis (when $\boldsymbol{C}$ has one non-input gate) is obvious. For the induction step, suppose the output gate g in $\boldsymbol{C}$ is an OR-gate with inputs g_a and g_b, at least one of which is a non-input gate. Let $\mathcal{I}$ be an arbitrary model of cov_A and $\mathcal{A}_{\boldsymbol{C},\alpha}$. If at least one of a or b in the gadget for g is in $T^{\mathcal{I}}$, then clearly $\mathcal{I} \models \boldsymbol{q}$. It remains to consider the case when a and b are both in $F^{\mathcal{I}}$. Let i be such that g_i is a non-input gate. There are two cases. (*a*) If node z in the gadget for g_i is in $F^{\mathcal{I}}$, consider the subcircuit $\boldsymbol{C}^-$ of $\boldsymbol{C}$ whose output gate is g_i. Then $\mathcal{A}_{\boldsymbol{C}^-,\alpha}$ is the sub-ABox of $\mathcal{A}_{\boldsymbol{C},\alpha}$ with z as its topmost node, and $A(z)$ replaced by $F(z)$. Now, if $\mathcal{I}^-$ is the restriction of $\mathcal{I}$ to $\mathcal{A}_{\boldsymbol{C}^-,\alpha}$ (and so $z \in F^{\mathcal{I}^-}$) then, by IH, there is a $\boldsymbol{q} \to \mathcal{I}^-$ homomorphism, and so $\mathcal{I} \models \boldsymbol{q}$ as well. (*b*) If $z \in T^{\mathcal{I}}$ then g_i is an AND-gate and, as the topmost A-node in the gadget for g_i is in $F^{\mathcal{I}}$, there is an A-node in the gadget for g_i that is in $T^{\mathcal{I}}$ while the next A-node above it is in F^{I}. So we have a $\boldsymbol{q} \to \mathcal{I}$ homomorphism. The case when the output gate of $\boldsymbol{C}$ is an AND-gate is similar.

($\Rightarrow$) Suppose $\boldsymbol{C}(\alpha) = F$. To show $\mathrm{cov}_A, \mathcal{A}_{\boldsymbol{C},\alpha} \not\models \boldsymbol{q}$, we define a model $\mathcal{I}$ of cov_A and $\mathcal{A}_{\boldsymbol{C},\alpha}$ by putting the A-nodes of the gadget for any gate g in $\boldsymbol{C}$ to $F^{\mathcal{I}}$ (or $T^{\mathcal{I}}$) if the truth-value of g under α is F (or, respectively, T). Suppose, on the contrary, that there is a homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. We track the possible locations of $h(x_0) \in F^{\mathcal{I}}$:

- If the output gate is an AND-gate, then $h(x_0)$ cannot be the F-node of its gadget because then $h(x_{-1}) = a$ and $h(x_r) = b$, and so at least one of them would be in F^I, which is a contradiction.
- If the output gate is an OR-gate, then $h(x_0)$ cannot be the F-node of its gadget because then either $h(x_{-1}) = a$ or $h(x_{-1}) = b$, and so $h(x_{-1})$ would be in F^I, a contradiction.
- So suppose $h(x_0)$ is an A-node in a gadget for a non-input and non-output gate g. If g is an OR-gate, then either $h(x_{-1}) = a$ or $h(x_{-1}) = b$ in the gadget for g, and so $h(x_{-1})$ would be in F^I, a contradiction. So suppose g is an AND-gate, and consider the gadget for g. Then $h(x_0)$ cannot be any A-node located above z, because otherwise $h(x_{-1})$ would be the previous A-node, and so in $F^{\mathcal{I}}$, a contradiction. Finally, if $h(x_0) = z$ then, as the vertical line comprised of the $\boldsymbol{r}$ is longer than $\boldsymbol{q}$ and contains no T-nodes, $h(x_1) \in T^{\mathcal{I}}$ must also be in the gadget for g, and it must be in one of the horizontal $\boldsymbol{s}$. But this is impossible because $\boldsymbol{r}$ is non-empty, and so the distance between $z = h(x_0)$ and $h(x_1)$ in the gadget would be greater than the distance between x_0 and x_1 in $\boldsymbol{q}$.

Thus, we cannot have a homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. □

The proof above bears some superficial similarities to the construction of Afrati and Papadimitriou [47] in their classification of binary chain sirups. One could also draw some parallels with the proof of P-hardness for OMQs with an $\mathcal{EL}$ ontology given by Lutz and Sabellek [34,35], who used a reduction of path systems accessibility (PSA) rather than monotone circuit evaluation.

The most difficult part of our tetrachotomy is proving coNP-hardness of dd-sirups with path 2-CQs. Despite the abundance of results on algorithmic aspects of graph homomorphisms [87], we failed to find any known technique applicable to our case. In the remainder of the article, we develop a new method for establishing coNP-hardness of disjunctive OMQs.

6. Proving coNP-hardness: the bike technique

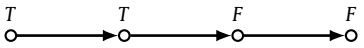
Theorem 27. *If $\boldsymbol{q}$ is a twinless path 2-CQ, then answering both $(\mathsf{cov}_A, \boldsymbol{q})$ and $(\mathsf{cov}_A^{\perp}, \boldsymbol{q})$ is* coNP*-hard.*

We prove Theorem 27 by a polynomial reduction of the complement of NP-complete 3SAT [98]. Recall that a *3CNF* is a conjunction of *clauses* of the form $\ell_1 \vee \ell_2 \vee \ell_3$, where each ℓ_i is a *literal* (a propositional variable or a negation thereof). The decision problem 3SAT asks whether a given 3CNF ψ is satisfiable. For any 3CNF ψ, we construct a twinless ABox $\mathcal{A}_{\boldsymbol{q},\psi}$ whose size is polynomial in the sizes of $\boldsymbol{q}$ and ψ, and show that ψ is satisfiable iff $\mathsf{cov}_A, \mathcal{A}_{\boldsymbol{q},\psi} \not\models \boldsymbol{q}$ iff $\mathsf{cov}_A^{\perp}, \mathcal{A}_{\boldsymbol{q},\psi} \not\models \boldsymbol{q}$ (cf. (8)). The construction, called the *bike technique*, builds $\mathcal{A}_{\boldsymbol{q},\psi}$ from many copies of $\boldsymbol{q}$ via three major steps:

1. First, we represent the truth-values of the *literals* in ψ by gadgets called cogwheels.
2. Next, we connect cogwheels to represent *negation* properly by gadgets called bikes.
3. Finally, we connect bikes to represent the interaction of the *clauses* in ψ and obtain $\mathcal{A}_{\boldsymbol{q},\psi}$.

These steps will be defined and investigated in detail in Sections 6.1–6.3. But before that we explain the underlying ideas and illustrate them by an example. Each cogwheel $\mathcal{W}$ in Step 1 has many A-nodes (the number depends on $|\boldsymbol{q}|$ and the number of clauses in ψ) where the different copies of $\boldsymbol{q}$ meet. Each $\mathcal{W}$ is such that, for every model $\mathcal{I}$ of cov_A and $\mathcal{W}$, we have $\mathcal{I} \not\models \boldsymbol{q}$ iff the A-nodes in $\mathcal{W}$ are all in $T^{\mathcal{I}}$ or are all in $F^{\mathcal{I}}$. If a variable p occurs in ψ, then in Step 2 a bike $\mathcal{B}$, representing the pair $\{p, \neg p\}$ of literals, is assembled from two disjoint cogwheels by connecting them via A-nodes using two further copies of $\boldsymbol{q}$. We have pairwise disjoint bikes for all variables occurring in ψ. Each bike $\mathcal{B}$ is constructed in such a way that, for every model $\mathcal{I}$ of cov_A and $\mathcal{B}$, we have $\mathcal{I} \not\models \boldsymbol{q}$ iff the truth-values in $\mathcal{I}$ represented by the two cogwheels of $\mathcal{B}$ are opposites of each other. Finally, in Step 3, for each clause $c = \ell_1 \vee \ell_2 \vee \ell_3$ in ψ, we use a further copy $\boldsymbol{q}^c$ of $\boldsymbol{q}$ to connect, via three A-nodes, three cogwheels from the bikes representing $\{\ell_1, \neg\ell_1\}$, $\{\ell_2, \neg\ell_2\}$ and $\{\ell_3, \neg\ell_3\}$ in such a way that for every model $\mathcal{I}$ of cov_A and the resulting ABox $\mathcal{A}_{\boldsymbol{q},\psi}$, we have $\mathcal{I} \not\models \boldsymbol{q}$ iff the labels of the three 'c-connection' A-nodes in $\mathcal{I}$ define an assignment satisfying c. If in Step 1 we choose the number of A-nodes in the cogwheels to be large enough, then in each cogwheel we can use different 'c-connection' A-nodes for different clauses, and they can also be different from those A-nodes that are used for constructing the bikes from the cogwheels.

Example 27.1. Consider the d-sirup $(\mathsf{cov}_A, \boldsymbol{q})$ with the 2-CQ $\boldsymbol{q}$ shown in the picture below (with R on edges omitted).



Let $\psi = c_1 \wedge c_2 \wedge c_3$, where $c_1 = \neg p \vee q \vee \neg r$, $c_2 = p \vee q \vee \neg r$, and $c_3 = p \vee \neg q \vee r$. Fig. 5 shows the steps of the construction of $\mathcal{A}_{\boldsymbol{q},\psi}$. In Step 1, we construct six cogwheels, each from four copies of $\boldsymbol{q}$, representing one of $p, \neg p, q, \neg q, r, \neg r$. Then in Step 2 we construct three bikes, representing the pairs $\{p, \neg p\}$, $\{q, \neg q\}$ and $\{r, \neg r\}$. Finally, we connect the bikes in Step 3 to obtain $\mathcal{A}_{\boldsymbol{q},\psi}$. Given an assignment $\mathfrak{a}\colon \{p, q, r\} \to \{T, F\}$, we define a model $\mathcal{I}_{\mathfrak{a}}$ of cov_A and $\mathcal{A}_{\boldsymbol{q},\psi}$ as follows: for $v \in \{p, q, r\}$, if $\mathfrak{a}(v) = T$ then the A-nodes in the v-cogwheel are in $T^{\mathcal{I}_{\mathfrak{a}}}$ and in the $\neg v$-cogwheel are in $F^{\mathcal{I}_{\mathfrak{a}}}$; and if $\mathfrak{a}(v) = F$ then the A-nodes in the v-cogwheel are in $F^{\mathcal{I}_{\mathfrak{a}}}$ and in the $\neg v$-cogwheel are in $T^{\mathcal{I}_{\mathfrak{a}}}$. It is tedious but not hard to check that $\mathcal{I}_{\mathfrak{a}} \not\models \boldsymbol{q}$ iff $\mathfrak{a}$ satisfies ψ.

It is far from obvious what exactly are the particular properties of this construction that can be generalised to *arbitrary* twinless path 2-CQs (just consider some permutations of the F- and T-nodes in $\boldsymbol{q}$ above). On the one hand, it is easy to identify what is needed for the 'if $\mathsf{cov}_A, \mathcal{A}_{\boldsymbol{q},\psi} \not\models \boldsymbol{q}$ then ψ is satisfiable' direction to hold. However, the main obstacle in proving the converse implication is that, given a model $\mathcal{I}$ determined by an assignment satisfying ψ, we need to exclude *all* $\boldsymbol{q} \to \mathcal{I}$ homomorphisms, not just those that map $\boldsymbol{q}$ onto one of its copies in $\mathcal{A}_{\boldsymbol{q},\psi}$. At each of the three steps, there can be such 'parasite' $\boldsymbol{q} \to \mathcal{I}$ homomorphisms, and there is no single, universal way of correctly assembling the $\mathcal{A}_{\boldsymbol{q},\psi}$ for all $\boldsymbol{q}$ and ψ. In the remainder of the article, we show how to overcome this obstacle.

We fix some twinless path 2-CQ $\boldsymbol{q}$ and use the following notation. For any k, we let t_k (f_k) denote the kth T-node (F-node) in $\boldsymbol{q}$. In particular, t_1, t_{last-1}, and t_{last} denote, respectively, the first, the last but one, and the last T-node in $\boldsymbol{q}$. Given any path CQ $\boldsymbol{q}'$, we write $\prec_{\boldsymbol{q}'}$ and $\preceq_{\boldsymbol{q}'}$ for the ordering of nodes in $\boldsymbol{q}'$, and $\delta_{\boldsymbol{q}'}$ for the distance in $\boldsymbol{q}'$, that is, $\delta_{\boldsymbol{q}'}(x, y)$ is the number of edges in the path from x to y whenever $x \preceq_{\boldsymbol{q}'} y$. As before, we omit the subscripts when $\boldsymbol{q}' = \boldsymbol{q}$, and set $|\boldsymbol{q}| = \delta(b_{\boldsymbol{q}}, e_{\boldsymbol{q}})$, for the first (root) node $b_{\boldsymbol{q}}$ and the last (leaf) node $e_{\boldsymbol{q}}$ in $\boldsymbol{q}$.

Throughout, when proving statements of the form $\mathsf{cov}_A, \mathcal{A} \not\models \boldsymbol{q}$ for some ABox $\mathcal{A}$, we use a generalisation of homomorphisms, which allows us to regard our CQs as if they contained a *single* binary predicate only. Given a model $\mathcal{I}$ of an ABox $\mathcal{A}$, we call a map $h\colon \boldsymbol{q} \to \mathcal{I}$ a *subhomomorphism* if the following conditions hold:

- $h(x) \in T^{\mathcal{I}}$, for every T-node x in $\boldsymbol{q}$, and $h(x) \in F^{\mathcal{I}}$, for every F-node x in $\boldsymbol{q}$;
- for any nodes x, y in $\boldsymbol{q}$, if $R(x, y)$ is in $\boldsymbol{q}$ for some R, then $S\big(h(x), h(y)\big)$ is in $\mathcal{A}$ for some S.

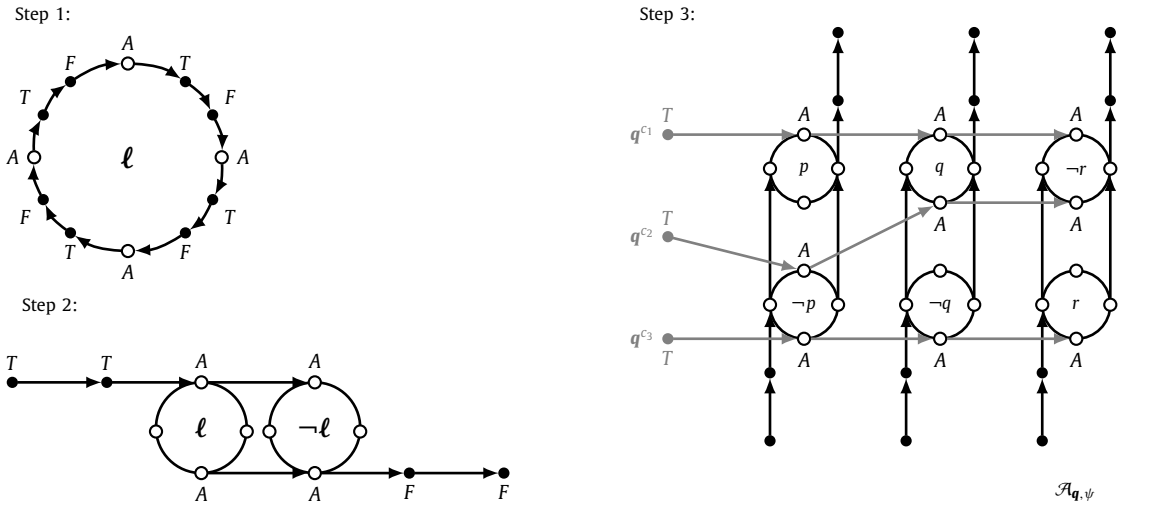


Fig. 5. An example of the bike-technique.

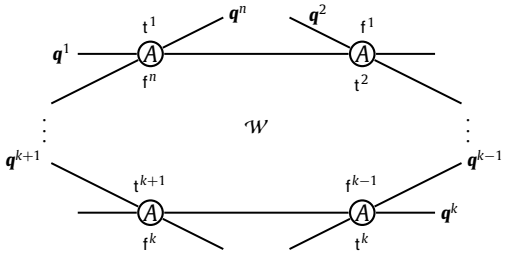


Fig. 6. An n-cogwheel $\mathcal{W}$ for $\boldsymbol{q}$.

The ABoxes $\mathcal{A}$ we build from copies of $\boldsymbol{q}$ will contain cycles, but these cycles will be large compared to $|\boldsymbol{q}|$. Thus, for any subhomomorphism h mapping $\boldsymbol{q}$ to some model $\mathcal{I}$ of $\mathcal{A}$, $h(\boldsymbol{q})$ can always be regarded as a path CQ, and we have the following obvious '*h-shift*' property:

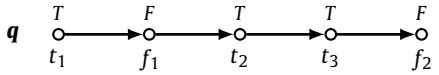
$$\delta(y,z) = \delta_{h(\boldsymbol{q})}\big(h(y), h(z)\big), \text{ for all nodes } y \text{ and } z \text{ in } \boldsymbol{q}. \tag{20}$$

6.1. Representing the truth-values of literals by cogwheels

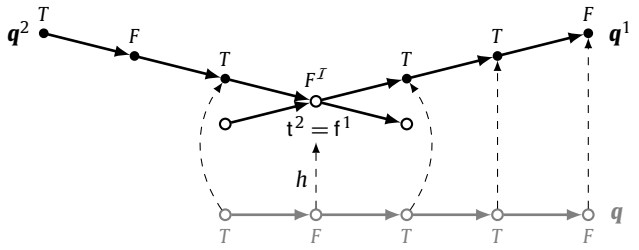
For $n \geq |\boldsymbol{q}|$, we take n disjoint copies $\boldsymbol{q}^1, \dots, \boldsymbol{q}^n$ of $\boldsymbol{q}$. For any j, $1 \leq j \leq n$, and any node x in $\boldsymbol{q}$, let x^j denote the copy of x in $\boldsymbol{q}^j$. For each j, we pick a T-node t^j and an F-node f^j in $\boldsymbol{q}^j$, calling the selected nodes *contacts*. We replace the T- and F-labels of all the contacts with A, and then glue f^j together with t^{j+1} for every j, $1 \leq j \leq n$, with $\pm$ being understood throughout modulo n. We call the resulting ABox $\mathcal{W}$ an *n-cogwheel (for $\boldsymbol{q}$)*; see Fig. 6. Given two contacts $\mathsf{c}_1 = \mathsf{f}^i = \mathsf{t}^{i+1}$ and $\mathsf{c}_2 = \mathsf{f}^j = \mathsf{t}^{j+1}$, we define the *contact-distance between c_1 and c_2 in $\mathcal{W}$* as $\min\big(|i-j|, n-|i-j|\big)$.

As shown in Lemma 27.3 below, it is straightforward to see that, for any n-cogwheel $\mathcal{W}$, if $\mathcal{I}$ is a model of cov_A and $\mathcal{W}$ with $\mathcal{I} \not\models \boldsymbol{q}$, then either all contacts of $\mathcal{W}$ are in $T^{\mathcal{I}}$ or all contacts of $\mathcal{W}$ are in $F^{\mathcal{I}}$. We want the converse implication to hold as well, in which case $\mathcal{W}$ would 'represent' a truth-value. In order to achieve this, we need to choose the contacts in such a way that all possible locations in $\mathcal{W}$ for the image $h(\boldsymbol{q})$ of a potential homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$ are excluded. The following example shows an improper choice of contacts.

Example 27.2. Consider the 2-CQ shown in the picture below.



Take two copies $\boldsymbol{q}^1$ and $\boldsymbol{q}^2$ of $\boldsymbol{q}$ in $\mathcal{W}$. If we choose the contacts $\mathsf{t}^1 = t_1^1$, $\mathsf{f}^1 = f_1^1$, $\mathsf{t}^2 = t_3^2$, $\mathsf{f}^2 = f_2^1$, and $\mathcal{I}$ is such that all contacts of $\mathcal{W}$ are in $F^{\mathcal{I}}$, then we do have the following $h\colon \boldsymbol{q} \to \mathcal{I}$ homomorphism:



To make the search space for contacts smaller and exclude cases like in Example 27.2, we make the following assumptions. To begin with, we assume that

$$t_1 \prec f_1 \tag{21}$$

(as the other case is symmetric). We also assume that the contacts of the n-cogwheel $\mathcal{W}$ have the following properties:

$$\mathsf{t}^j \prec_{\boldsymbol{q}^j} \mathsf{f}^j, \text{ for every } j \text{ with } 1 \leq j \leq n; \tag{22}$$

$$\text{if } \mathsf{t}^{j+1} = t^{j+1} \text{ and } \mathsf{f}^j = f^j, \text{ then } t \prec f, \text{ for all } j \text{ with } 1 \leq j \leq n. \tag{23}$$

(Note that (23) does not hold in Example 27.2, as $t_3 \not\prec f_1$.) For each j, the nodes preceding t^j in $\boldsymbol{q}^j$ form its *initial cog*, while the nodes succeeding f^j in $\boldsymbol{q}^j$ form its *final cog*.

The following general criterion still gives us quite some flexibility in designing cogwheels:

Lemma 27.3. *Suppose $\mathcal{W}$ is an n-cogwheel for some $n \geq |\boldsymbol{q}|$ satisfying (22) and (23). For any model $\mathcal{I}$ of cov_A and $\mathcal{W}$, we have $\mathcal{I} \not\models \boldsymbol{q}$ iff the contacts in $\mathcal{I}$ are either all in $T^{\mathcal{I}}$ or all in $F^{\mathcal{I}}$.*

Proof. ($\Rightarrow$) Suppose the contact $\mathsf{f}^{i-1} = \mathsf{t}^i$ is in $T^{\mathcal{I}}$. Since $\mathcal{I} \not\models \boldsymbol{q}$, the 'clockwise next' contact $\mathsf{f}^i = \mathsf{t}^{i+1}$ is also in $T^{\mathcal{I}}$. It follows by induction that all of the contacts in $\mathcal{W}$ are in $T^{\mathcal{I}}$. If the contact $\mathsf{f}^{i-1} = \mathsf{t}^i$ is in $F^{\mathcal{I}}$, then the 'anti-clockwise next' contact $\mathsf{f}^{i-2} = \mathsf{t}^{i-1}$ is also in $F^{\mathcal{I}}$, from which it follows by induction that all of the contacts are in $F^{\mathcal{I}}$.

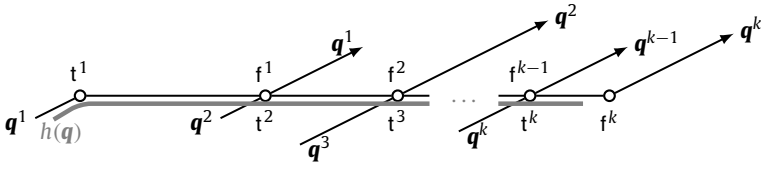
($\Leftarrow$) First, suppose that $\mathcal{I}$ is a model of cov_A and $\mathcal{W}$ such that all contacts in $\mathcal{I}$ are in $F^{\mathcal{I}}$. The proof is via excluding all possible locations in $\mathcal{W}$ for the image $h(\boldsymbol{q})$ of a potential subhomomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. As $n \geq |\boldsymbol{q}|$, we may consider $h(\boldsymbol{q})$ as a path CQ, so $\preceq_{h(\boldsymbol{q})}$ and $\delta_{h(\boldsymbol{q})}$ are well-defined. Observe that if t and f are such that $\mathsf{t}^j = t^j$ and $\mathsf{f}^j = f^j$ for some j then, by the definition of the minimal model $\mathcal{I}$, we clearly cannot have that $h(t) = \mathsf{t}^j$ and $h(f) = \mathsf{f}^j$. In particular, there is no $\boldsymbol{q} \to \mathcal{I}$ subhomomorphism mapping $\boldsymbol{q}$ onto any of the $\boldsymbol{q}^j$, and so $h(\boldsymbol{q})$ must intersect with at least two copies of $\boldsymbol{q}$ in $\mathcal{W}$. Further, by (23), there is not enough room for $h(\boldsymbol{q})$ to start in an initial cog and then, after reaching a contact, to finish in a final cog (like in Example 27.2).

So, without loss of generality, we may assume that there is some k such that $2 \leq k < n$,

$$h(\boldsymbol{q}) \text{ intersects each of the copies } \boldsymbol{q}^1, \dots, \boldsymbol{q}^k, \tag{24}$$

$$b^1_{\boldsymbol{q}} \prec_{\boldsymbol{q}^1} h(b_{\boldsymbol{q}}) \prec_{\boldsymbol{q}^1} \mathsf{f}^1, \text{ and} \tag{25}$$

$$h(\boldsymbol{q}) \cap \boldsymbol{q}^k \neq \{\mathsf{t}^k\}. \tag{26}$$



Now, consider the sub-ABox $\mathcal{H}$ of $\mathcal{W}$ consisting of the copies $\boldsymbol{q}^1, \dots, \boldsymbol{q}^k$. For any ℓ with $1 \leq \ell \leq k$, let $\iota^\ell\colon \boldsymbol{q}^\ell \to \boldsymbol{q}$ be the isomorphism mapping each x^ℓ to x. We define a function $g_{\leftarrow}\colon \mathsf{ind}(\mathcal{H}) \to \mathsf{ind}(\mathcal{H})$ 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 $\mathsf{c} = \mathsf{f}^\ell = \mathsf{t}^{\ell+1}$, for $1 \leq \ell < k$, as a node in $\boldsymbol{q}^{\ell+1}$, that is, $g_{\leftarrow}(\mathsf{c}) = g_{\leftarrow}(\mathsf{t}^{\ell+1}) = h\big(\iota^{\ell+1}(\mathsf{t}^{\ell+1})\big)$. Throughout, we use the following property of $g_{\leftarrow}$, which is a straightforward consequence of the h-shift in (20) and the similar property of the isomorphism ι^ℓ: for every ℓ with $1 \leq \ell \leq k$,

$$\text{if } y, z \text{ are both in the same copy } \boldsymbol{q}^\ell,\ y, z \neq \mathsf{f}^\ell \text{ whenever } \ell < k, \text{ and } y \preceq_{\boldsymbol{q}^\ell} z,$$
$$\text{then } g_{\leftarrow}(y) \preceq_{h(\boldsymbol{q})} g_{\leftarrow}(z) \text{ and } \delta_{\boldsymbol{q}^\ell}(y, z) \;=\; \delta_{h(\boldsymbol{q})}\big(g_{\leftarrow}(y), g_{\leftarrow}(z)\big). \tag{27}$$

As $\mathcal{H}$ is finite, there exists a 'fixpoint' of $g_{\leftarrow}$: a node x in $\mathcal{H}$ and a number $N > 0$ such that $g^N_{\leftarrow}(x) = x$. We 'shift this fixpoint-cycle to the left.' More precisely, we claim that

$$\text{there is a contact } \mathsf{c} \text{ with } g^N_{\leftarrow}(\mathsf{c}) = \mathsf{c}. \tag{28}$$

Indeed, let $y_0 = x$, $y_1 = g_{\leftarrow}(x)$, $y_2 = g^2_{\leftarrow}(x), \dots, y_{N-1} = g^{N-1}_{\leftarrow}(x)$. Then $g^N_{\leftarrow}(y_j) = y_j$ for every $j < N$, and so if one of the y_j is a contact, we are done with (28). So suppose otherwise. We cannot have that every y_j is in $\boldsymbol{q}^1$, as otherwise, by (27) and (25), for every $j \leq N$,

$$\delta_{\boldsymbol{q}^1}(b^1_{\boldsymbol{q}}, y_j) = \delta_{h(\boldsymbol{q})}\big(g_{\leftarrow}(b^1_{\boldsymbol{q}}), g_{\leftarrow}(y_j)\big) = \delta_{h(\boldsymbol{q})}\big(h(b_{\boldsymbol{q}}), y_{j+1}\big) = \delta_{\boldsymbol{q}^1}\big(h(b_{\boldsymbol{q}}), y_{j+1}\big) < \delta_{\boldsymbol{q}^1}\big(b^1_{\boldsymbol{q}}, y_{j+1}\big) \tag{29}$$

(here $+$ is modulo N). Therefore, by (26),

$$\text{there exists } j < N \text{ such that } y_j \text{ is in } \boldsymbol{q}^{\ell_j} \text{ and } \mathsf{t}^{\ell_j} \prec_{\boldsymbol{q}^{\ell_j}} y_j, \text{ for some } \ell_j > 1. \tag{30}$$

(When $\ell_j = 1$, such a contact t^{ℓ_j} does not necessarily exist.) For $j < N$ with $\ell_j > 1$, we set $d_j = \delta_{\boldsymbol{q}^{\ell_j}}\big(\mathsf{t}^{\ell_j}, y_j\big)$. Let $K < N$ be such that

$$d_K = \min\{d_j \mid j < N \text{ and } \ell_j > 1\}$$

(which is well-defined by (30)), and set $\mathsf{c} = \mathsf{f}^{\ell_K - 1} = \mathsf{t}^{\ell_K}$. By (24), we have $y_j \prec_{\boldsymbol{q}^{\ell_j}} \mathsf{f}^{\ell_j}$ whenever $1 < \ell_j < k$. Thus, by the definition of K and (27), for every $j \leq N$, $g^j_{\leftarrow}(\mathsf{c})$ belongs to the same copy $\boldsymbol{q}^{\ell_{K+j}}$ as y_{K+j}, $g^j_{\leftarrow}(\mathsf{c}) \preceq_{\boldsymbol{q}^{\ell_{K+j}}} y_{K+j}$, and

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

It follows, in particular, that $g^N_{\leftarrow}(\mathsf{c})$ belongs to the same copy $\boldsymbol{q}^{\ell_K}$ as y_K, and $\delta_{\boldsymbol{q}^{\ell_K}}\big(g^N_{\leftarrow}(\mathsf{c}), y_K\big) = \delta_{\boldsymbol{q}^{\ell_K}}(\mathsf{c}, y_K)$. Therefore, $g^N_{\leftarrow}(\mathsf{c}) = \mathsf{c}$, as required in (28).

It remains to show that (28) leads to a contradiction. Indeed, $\mathsf{c} \in F^{\mathcal{I}}$ by our assumption, and so c cannot be in $T^{\mathcal{I}}$ by the minimality of $\mathcal{I}$. On the other hand, we show by induction on $j \geq 1$ that $g^j_{\leftarrow}(\mathsf{c}) \in T^{\mathcal{I}}$, and so $\mathsf{c} = g^N_{\leftarrow}(\mathsf{c}) \in T^{\mathcal{I}}$. If $j = 1$ then $g_{\leftarrow}(\mathsf{c}) = h\big(\iota^{\ell}(\mathsf{t}^{\ell})\big)$ for some ℓ, and so $g_{\leftarrow}(\mathsf{c}) \in T^{\mathcal{I}}$ as $\iota^{\ell}(\mathsf{t}^{\ell})$ is a T-node in $\boldsymbol{q}$ and h is a subhomomorphism. If $j > 1$ then $g^{j-1}_{\leftarrow}(\mathsf{c}) \in T^{\mathcal{I}}$ by IH. Thus, $g^{j-1}_{\leftarrow}(\mathsf{c})$ is not a contact and $\iota^{\ell}\big(g^{j-1}_{\leftarrow}(\mathsf{c})\big)$ must be a T-node in $\boldsymbol{q}$ for some ℓ. Therefore, $g^j_{\leftarrow}(\mathsf{c}) = h\big(\iota^{\ell}\big(g^{j-1}_{\leftarrow}(\mathsf{c})\big)\big)$ is in $T^{\mathcal{I}}$, as h is a subhomomorphism.

The case of $\mathcal{I}$ with contacts in $T^{\mathcal{I}}$ is similar. Now we define a function $g_{\rightarrow} : \mathsf{ind}(\mathcal{H}) \to \mathsf{ind}(\mathcal{H})$ by taking again $g_{\rightarrow}(x) = h\big(\iota^{\ell}(x)\big)$ whenever x is a node in $\boldsymbol{q}^{\ell}$, but now we consider each contact $\mathsf{c} = \mathsf{f}^{\ell} = \mathsf{t}^{\ell+1}$ as a node in $\boldsymbol{q}^{\ell}$, that is, $g_{\rightarrow}(\mathsf{c}) = g_{\rightarrow}(\mathsf{f}^{\ell}) = h\big(\iota^{\ell}(\mathsf{f}^{\ell})\big)$. Then, in the proof of (28) for $g_{\rightarrow}$, we 'shift the fixpoint-cycle to the right'. □

Remark 27.4. It is to be noted that if we make more specialised assumptions on the choice of contacts, then Lemma 27.3 can have a more straightforward proof. For example, suppose that the n-cogwheel $\mathcal{W}$ satisfies (22) and $\mathsf{f}^j = f^j_1$, for all j with $1 \leq j \leq n$. Given a model $\mathcal{I}$ such that all contacts of $\mathcal{W}$ have the same truth-value, we can show that no subhomomorphism $h : \boldsymbol{q} \to \mathcal{I}$ exists by excluding the possible locations of $h(f_1)$:

- $h(f_1)$ cannot be a contact f^j, otherwise $h(t)$ is also a contact, for the T-node t with $\mathsf{t}^j = t^j$;
- $h(f_1)$ cannot be in the final cog of some $\boldsymbol{q}^j$, otherwise there is no room for $h(\boldsymbol{q})$ in that cog; and
- there are no other options for $h(f_1)$, as there is no F-node preceding f_1 in $\boldsymbol{q}$.

Unfortunately, as illustrated in Example 27.6 below, we cannot always assume our n-cogwheels to be that simple.

6.2. Representing negation by bikes

For each variable in the 3CNF ψ, we take a fresh pair of cogwheels $\mathcal{W}_{\bullet}$ and $\mathcal{W}_{\circ}$ and connect them using two more fresh copies of $\boldsymbol{q}$ in a special way. We want to achieve that, for any model $\mathcal{I}$ of cov_A and the resulting 'two-wheel' ABox, we have $\mathcal{I} \not\models \boldsymbol{q}$ iff the two cogwheels $\mathcal{W}_{\bullet}$ and $\mathcal{W}_{\circ}$ 'represent' opposite truth-values:

$$\text{either all contacts of } \mathcal{W}_{\bullet} \text{ are in } F^{\mathcal{I}} \text{ and all contacts of } \mathcal{W}_{\circ} \text{ are in } T^{\mathcal{I}},$$
$$\text{or all contacts of } \mathcal{W}_{\bullet} \text{ are in } T^{\mathcal{I}} \text{ and all contacts of } \mathcal{W}_{\circ} \text{ are in } F^{\mathcal{I}}. \tag{31}$$

To this end, suppose $\mathcal{W}_{\bullet}$ and $\mathcal{W}_{\circ}$ are two disjoint n-cogwheels, for some $n > 4|\boldsymbol{q}| + 2$, built up from the $\boldsymbol{q}$-copies ${}^{\bullet}\boldsymbol{q}^1, \dots, {}^{\bullet}\boldsymbol{q}^n$ and ${}^{\circ}\boldsymbol{q}^1, \dots, {}^{\circ}\boldsymbol{q}^n$, respectively. For $i = 1, \dots, n$ and $* = \bullet, \circ$, we denote the contacts in ${}^{*}\boldsymbol{q}^i$ by ${}^{*}\mathsf{t}^i$ and ${}^{*}\mathsf{f}^i$; and for any node x in $\boldsymbol{q}$, we denote by ${}^{*}x^i$ the copy of x in ${}^{*}\boldsymbol{q}^i$. We pick two contacts ${}^{\bullet}\mathsf{f}^{i_{\bullet}} = {}^{\bullet}\mathsf{t}^{i_{\bullet}+1}$ and ${}^{\bullet}\mathsf{f}^{j_{\bullet}} = {}^{\bullet}\mathsf{t}^{j_{\bullet}+1}$ in $\mathcal{W}_{\bullet}$

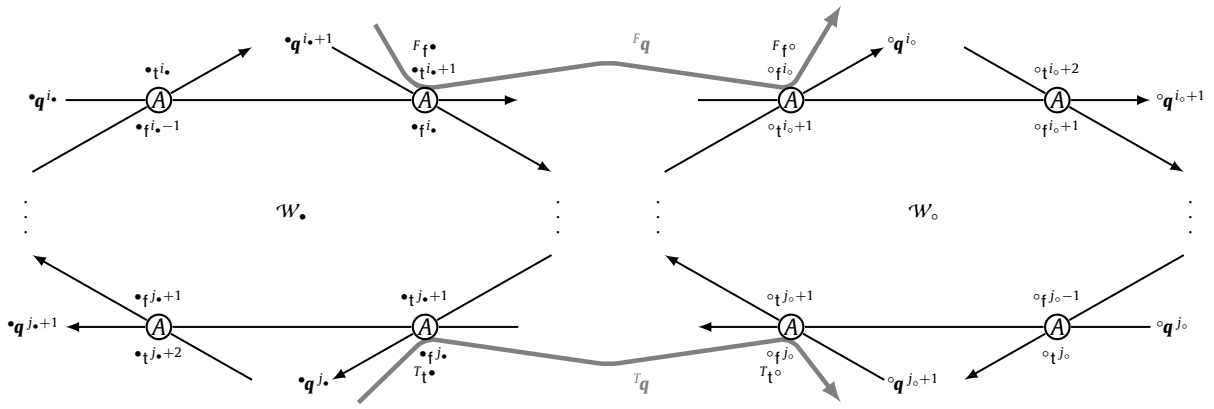


Fig. 7. An n-bike $\mathcal{B}$ for $\boldsymbol{q}$.

that are 'far' from each other in the sense that the contact-distance between them in $\mathcal{W}_\bullet$ is $> 2|\boldsymbol{q}|$. Similarly, we pick two contacts ${}^\circ\mathsf{f}^{i_\circ} = {}^\circ\mathsf{t}^{i_\circ+1}$ and ${}^\circ\mathsf{f}^{j_\circ} = {}^\circ\mathsf{t}^{j_\circ+1}$ in $\mathcal{W}_\circ$ such that the contact-distance between them in $\mathcal{W}_\circ$ is $> 2|\boldsymbol{q}|$.

Next, let ${}^F\boldsymbol{q}$, ${}^T\boldsymbol{q}$ be two more fresh and disjoint copies of $\boldsymbol{q}$. For $Z = F, T$ and any node x in $\boldsymbol{q}$, we denote by ${}^Z x$ the copy of x in ${}^Z\boldsymbol{q}$. We connect $\mathcal{W}_\bullet$ and $\mathcal{W}_\circ$ via ${}^F\boldsymbol{q}$ and ${}^T\boldsymbol{q}$ as follows. First, we pick two F-nodes $\mathsf{f}^\bullet$ and f° with $\mathsf{f}^\bullet \prec \mathsf{f}^\circ$ in $\boldsymbol{q}$, and replace their F-labels by A. Then we glue together node ${}^F\mathsf{f}^\bullet$ of ${}^F\boldsymbol{q}$ with the contact ${}^\bullet\mathsf{f}^{i_\bullet} = {}^\bullet\mathsf{t}^{i_\bullet+1}$ of $\mathcal{W}_\bullet$, and also glue together ${}^F\mathsf{f}^\circ$ with the contact ${}^\circ\mathsf{f}^{i_\circ} = {}^\circ\mathsf{t}^{i_\circ+1}$ of $\mathcal{W}_\circ$. Finally, we pick two T-nodes $\mathsf{t}^\bullet$ and t° with $\mathsf{t}^\bullet \prec \mathsf{t}^\circ$ in $\boldsymbol{q}$, and replace their T-labels by A. Then we glue together node ${}^T\mathsf{t}^\bullet$ of ${}^T\boldsymbol{q}$ with the contact ${}^\bullet\mathsf{f}^{j_\bullet} = {}^\bullet\mathsf{t}^{j_\bullet+1}$ of $\mathcal{W}_\bullet$, and also glue together ${}^T\mathsf{t}^\circ$ with the contact ${}^\circ\mathsf{f}^{j_\circ} = {}^\circ\mathsf{t}^{j_\circ+1}$ of $\mathcal{W}_\circ$. The resulting ABox $\mathcal{B}$ is called an *n-bike (for $\boldsymbol{q}$)*, see Fig. 7. We call the contacts ${}^F\mathsf{f}^\bullet = {}^\bullet\mathsf{f}^{i_\bullet} = {}^\bullet\mathsf{t}^{i_\bullet+1}$ and ${}^F\mathsf{f}^\circ = {}^\circ\mathsf{f}^{i_\circ} = {}^\circ\mathsf{t}^{i_\circ+1}$ *F-connections* in $\mathcal{B}$; the *F-neighbourhood* of $\mathcal{B}$ consists of those contacts whose contact-distance from an F-connection is $\leq |\boldsymbol{q}|$. Similarly, the contacts ${}^T\mathsf{t}^\bullet = {}^\bullet\mathsf{f}^{j_\bullet} = {}^\bullet\mathsf{t}^{j_\bullet+1}$ and ${}^T\mathsf{t}^\circ = {}^\circ\mathsf{f}^{j_\circ} = {}^\circ\mathsf{t}^{j_\circ+1}$ *T-connections* in $\mathcal{B}$, and the *T-neighbourhood* of $\mathcal{B}$ consists of those contacts whose contact-distance from a T-connection is $\leq |\boldsymbol{q}|$.

Using Lemma 27.3 and the fact that the F-connections are F-nodes in ${}^F\boldsymbol{q}$ while the T-connections are T-nodes in ${}^T\boldsymbol{q}$, it is straightforward to see that, for any n-bike $\mathcal{B}$, if $\mathcal{I}$ is a model of cov_A and $\mathcal{B}$ with $\mathcal{I} \not\models \boldsymbol{q}$, then (31) holds.

However, for the converse implication to hold, we need to choose the contacts that are (a) the F-connections in ${}^F\boldsymbol{q}$, (b) the T-connections in ${}^T\boldsymbol{q}$, and (c) located in the F- and T-neighbourhoods in the two cogwheels of $\mathcal{B}$ carefully, in such a way that all possible locations in $\mathcal{B}$ for the image $h(\boldsymbol{q})$ of a potential homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$ are excluded. So suppose $\mathcal{I}$ is a model of cov_A and $\mathcal{B}$ satisfies (31). We will again try to exclude all $h\colon \boldsymbol{q} \to \mathcal{I}$ subhomomorphisms. To begin with, as $n > 4|\boldsymbol{q}| > |\boldsymbol{q}|$, we may consider the image $h(\boldsymbol{q})$ of $\boldsymbol{q}$ in $\mathcal{I}$ as a path CQ. If we choose all the contacts in (a)–(c) above in such a way that (22) and (23) hold for both cogwheels in $\mathcal{B}$ then, by Lemma 27.3, we know that $h(\boldsymbol{q})$ must intersect with at least one of ${}^F\boldsymbol{q}$ and ${}^T\boldsymbol{q}$. Therefore, the intersection of $h(\boldsymbol{q})$ with any of the two n-cogwheels cannot go beyond their F- and T-neighbourhoods. Further, it is straightforward to see that because of (31),

$$\text{there is no subhomomorphism } h : \boldsymbol{q} \to \mathcal{I} \text{ such that } h(\mathsf{f}^\bullet) = {}^F\mathsf{f}^\bullet \text{ and } h(\mathsf{f}^\circ) = {}^F\mathsf{f}^\circ, \text{ and}$$
$$\text{there is no subhomomorphism } h : \boldsymbol{q} \to \mathcal{I} \text{ such that } h(\mathsf{t}^\bullet) = {}^T\mathsf{t}^\bullet \text{ and } h(\mathsf{t}^\circ) = {}^T\mathsf{t}^\circ. \quad (32)$$

(In particular, there is no $\boldsymbol{q} \to \mathcal{I}$ subhomomorphism mapping $\boldsymbol{q}$ onto ${}^F\boldsymbol{q}$ or onto ${}^T\boldsymbol{q}$.) Because of this, $h(\boldsymbol{q})$ must *properly intersect* with at least one of the two n-cogwheels $\mathcal{W}_\bullet$ or $\mathcal{W}_\circ$ in the sense that the intersection of $h(\boldsymbol{q})$ and the cogwheel is not just a T- or F-connection. As the F-connections are of contact-distance $> 2|\boldsymbol{q}|$ from the T-connections, $h(\boldsymbol{q})$ cannot intersect with both ${}^F\boldsymbol{q}$ and ${}^T\boldsymbol{q}$ at the same time. It is easy to check that, by (32), all options for such a $h(\boldsymbol{q})$ are covered by the eight cases given in Fig. 8.

We aim to show that, for every 2-CQ, suitable contact choices always exist by actually providing an *algorithm* that, given *any* 2-CQ $\boldsymbol{q}$, describes contact choices that are suitable for an n-bike constructed from copies of $\boldsymbol{q}$. For some 2-CQs the suitable contact choices are straightforward (even uniquely determined by (22) and (23)), for some others not so. In general, the different cases in Fig. 8 place different constraints on the suitable contact choices. There might even be some further interaction among these constraints because $h(\boldsymbol{q})$ might intersect, say, the T-neighbourhoods of both cogwheels in $\mathcal{B}$. These interactions, together with constraints (22) and (23), make finding a general solution a tricky cat-and-mouse game. We have tried several different ways of systematising the search for solutions, and ended up with the following choices in our 'heuristics' (with Remark 27.4 motivating (H2)):

(H1) We try to choose all contacts in a way that results in as few cases as possible.
(H2) In excluding possible locations for $h(\boldsymbol{q})$, we aim to track $h(f_1)$. So we aim to choose the contacts in such a way that leaves as few options for $h(f_1)$ as possible.

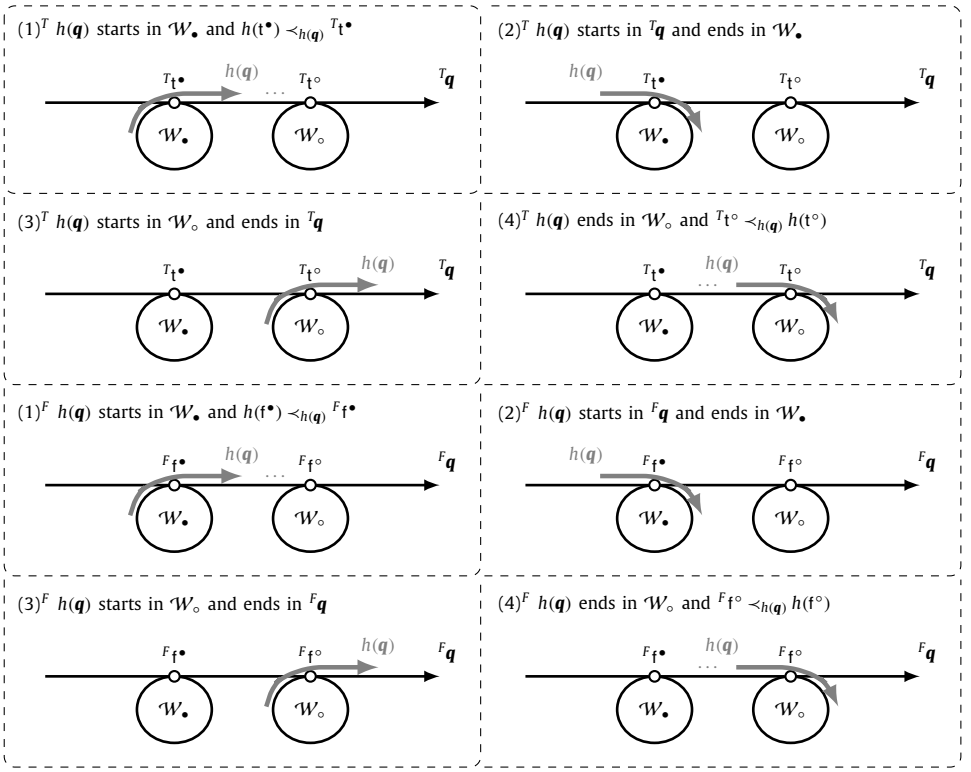
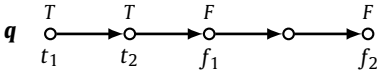


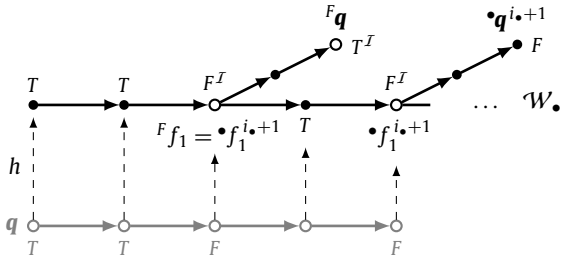
Fig. 8. Possible locations for $h(\boldsymbol{q})$ intersecting ${}^F\boldsymbol{q}$ or ${}^T\boldsymbol{q}$.

In particular, in light of (H1) and (H2), we decided to go for $\mathsf{f}^\bullet = f_1$ and $\mathsf{f}^\circ = f_2$ as F-connections. This leaves us with only two options for $h(f_1)$ in ${}^F\boldsymbol{q}$: its two contacts $\mathsf{f}^\bullet$ or f°. However, we still have to deal with case distinctions in the choices for ${}^\bullet\mathsf{t}^{i_\bullet+1}$ and ${}^\circ\mathsf{t}^{i_\circ+1}$ as illustrated by the following examples.

Example 27.5. (*i*) Consider the 2-CQ

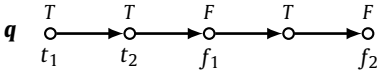


If we choose ${}^\bullet\mathsf{t}^{i_\bullet+1} = {}^\bullet t_1^{i_\bullet+1}$ and ${}^\bullet\mathsf{f}^{i_\bullet+1} = {}^\bullet f_1^{i_\bullet+1}$, and $\mathcal{I}$ is such that all contacts of $\mathcal{W}_\bullet$ are in $F^\mathcal{I}$ and all contacts of $\mathcal{W}_\circ$ are in $T^\mathcal{I}$, then we do have the following $h\colon \boldsymbol{q} \to \mathcal{I}$ homomorphism (see case (2)^F in Fig. 8):

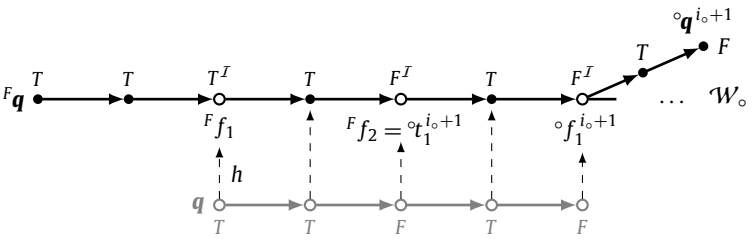


Note that choosing ${}^\bullet\mathsf{f}^{i_\bullet+1} = {}^\bullet f_2^{i_\bullet+1}$ would not help.

(*ii*) Consider the 2-CQ

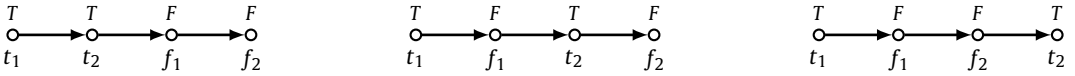


If we choose ${}^\circ\mathsf{t}^{i_\circ+1} = {}^\circ t_1^{i_\circ+1}$ and ${}^\circ\mathsf{f}^{i_\circ+1} = {}^\circ f_1^{i_\circ+1}$, and $\mathcal{I}$ is such that all contacts of $\mathcal{W}_\bullet$ are in $T^\mathcal{I}$ and all contacts of $\mathcal{W}_\circ$ are in $F^\mathcal{I}$, then we do have the following $h\colon \boldsymbol{q} \to \mathcal{I}$ homomorphism (see case $(4)^F$ in Fig. 8):



Note again that choosing ${}^\circ\mathsf{f}^{i_\circ+1} = {}^\circ f_2^{i_\circ+1}$ would not help.

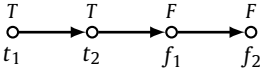
(*iii*) On the other hand, as shown in Lemma 27.7 below, the contact choices of ${}^\bullet\mathsf{t}^{i_\bullet+1} = {}^\bullet t_1^{i_\bullet+1}$, ${}^\bullet\mathsf{f}^{i_\bullet+1} = {}^\bullet f_1^{i_\bullet+1}$, ${}^\circ\mathsf{t}^{i_\circ+1} = {}^\circ t_1^{i_\circ+1}$, and ${}^\circ\mathsf{f}^{i_\circ+1} = {}^\circ f_1^{i_\circ+1}$ are suitable for any of the following three 2-CQs:



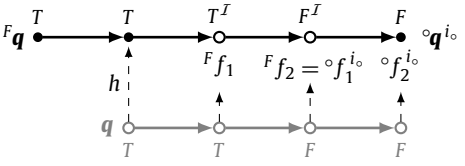
There are other sources of inherent case distinctions. For example, in light of Remark 27.4, it would be tempting to try (the copies of) f_1 as contacts throughout the F-neighbourhoods of $\mathcal{W}_\bullet$ and $\mathcal{W}_\circ$. However, this is not always possible, as illustrated by the following examples.

Example 27.6. (*i*) Take any 2-CQ $\boldsymbol{q}$ that contains only two F-nodes. Thus, we must choose $\mathsf{f}^\bullet = f_1$. If we choose ${}^\bullet\mathsf{f}^{i_\bullet} = {}^\bullet f_1^{i_\bullet}$, then in case $(2)^F$ it is always possible to start $h(\boldsymbol{q})$ in ${}^F\boldsymbol{q}$, map f_1 to ${}^F\mathsf{f}^\bullet = {}^\bullet f_1 = {}^\bullet\mathsf{f}^{i_\bullet} = {}^\bullet f_1^{i_\bullet}$, and finish $h(\boldsymbol{q})$ in the final cog of ${}^\bullet\boldsymbol{q}^{i_\bullet}$.

(*ii*) Consider the 2-CQ



Then we must choose $\mathsf{f}^\bullet = f_1$ and $\mathsf{f}^\circ = f_2$, and so ${}^F\mathsf{f}^\bullet = {}^F f_1$ and ${}^F\mathsf{f}^\circ = {}^F f_2$. If we choose ${}^\circ\mathsf{f}^{i_\circ} = {}^\circ f_1^{i_\circ}$, and $\mathcal{I}$ is such that all contacts of $\mathcal{W}_\bullet$ are in $T^\mathcal{I}$ and all contacts of $\mathcal{W}_\circ$ are in $F^\mathcal{I}$, then we do have the following $h\colon \boldsymbol{q} \to \mathcal{I}$ homomorphism (see case $(4)^F$ in Fig. 8):



There are also examples showing that, unlike the F-connections, the T-connections cannot be chosen uniformly for all possible 2-CQs $\boldsymbol{q}$. (The problems we face are not 'symmetric counterparts' of those with the F-connections because of our overall assumption that $t_1 \prec f_1$; see (21).) We hope that the above examples convince the reader that, even after fixing (H1) and (H2) (or any other heuristics), finding a *general* solution that satisfies (22) and (23) but excludes all cases in Fig. 8 for *any* 2-CQ $\boldsymbol{q}$ is quite a challenge. The following lemma describes such a solution found along the lines of (H1) and (H2):

Lemma 27.7. *Let $\mathcal{B}$ be an n-bike, for some $n \geq 4|\boldsymbol{q}| + 2$, built up from the n-cogwheels $\mathcal{W}_\bullet$ and $\mathcal{W}_\circ$, each satisfying (22) and (23). Suppose $\mathcal{B}$ is such that the following hold for its T-connections*:

$$\mathsf{t}^\bullet = t_1, \qquad \mathsf{t}^\circ = \begin{cases} t_\triangle, & \text{if } f_1 \prec t_{last}, \quad \text{where } t_\triangle \text{ is the first } T\text{-node succeeding } f_1, \\ t_{last}, & \text{if } t_{last} \prec f_1; \end{cases}$$

the following hold for its T-neighbourhood:

$$ {}^{\bullet}\mathsf{t}^{j_\bullet} = {}^{\bullet}t_{\square}^{j_\bullet}, \quad \text{where } t_\square \text{ is the last } T\text{-node preceding } f_1, \qquad {}^{\bullet}\mathsf{f}^{j_\bullet} = \begin{cases} {}^{\bullet}f_1^{j_\bullet}, & \text{if } f_1 \prec t_{last}, \\ {}^{\bullet}f_2^{j_\bullet}, & \text{if } t_{last} \prec f_1, \end{cases} $$

$$ {}^{\circ}\mathsf{t}^{j_\circ} = {}^{\circ}t_1^{j_\circ}, \qquad {}^{\circ}\mathsf{f}^{j_\circ} = \begin{cases} {}^{\circ}f_2^{j_\circ}, & \text{if } t_{last} \prec f_1 \text{ and } \delta(t_{last-1}, t_{last}) = \delta(t_{last}, f_1), \\ {}^{\circ}f_1^{j_\circ}, & \text{otherwise}, \end{cases} $$

$$ {}^{*}\mathsf{t}^{k} = {}^{*}t_1^{k}, \qquad {}^{*}\mathsf{f}^{k} = {}^{*}f_1^{k}, \quad \text{for } * = \bullet, \circ \text{ and for any other } k \text{ with } j_* - |\boldsymbol{q}| \le k \le j_* + |\boldsymbol{q}|; $$

the following hold for its F-connections:

$$ \mathsf{f}^{\bullet} = f_1, \qquad \mathsf{f}^{\circ} = f_2; $$

and the following hold for its F-neighbourhood, for $* = \bullet, \circ$:

$$ {}^{*}\mathsf{t}^{i_*} = {}^{*}t_1^{i_*}, \qquad {}^{*}\mathsf{f}^{i_*} = {}^{*}f_2^{i_*}, $$

$$ {}^{*}\mathsf{t}^{i_*-k} = {}^{*}t_1^{i_*-k}, \qquad {}^{*}\mathsf{f}^{i_*-k} = {}^{*}f_1^{i_*-k}, \quad \text{for } 0 < k \le |\boldsymbol{q}|, $$

$$ {}^{*}\mathsf{t}^{i_*+\ell} = \begin{cases} {}^{*}t_1^{i_*+\ell}, & \text{if } t_{last} \prec f_1 \text{ and } \delta(f_1, f_2) < \delta(t_1, f_1), \\ {}^{*}t_{\square}^{i_*+\ell}, & \text{otherwise, where } t_\square \text{ is the last } T\text{-node preceding } f_1, \end{cases} $$

$$ {}^{*}\mathsf{f}^{i_*+\ell} = \begin{cases} {}^{*}f_2^{i_*+\ell}, & \text{if } t_{last} \prec f_1 \text{ and } \delta(f_1, f_2) \ge \delta(t_1, f_1), \\ {}^{*}f_1^{i_*+\ell}, & \text{otherwise}, \end{cases} $$

for $1 \le \ell \le |\boldsymbol{q}|$.

Then, for any model $\mathcal{I}$ of cov_A and $\mathcal{B}$, we have

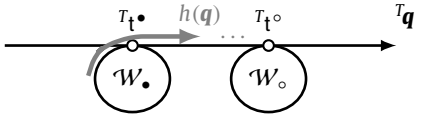
$$ \mathcal{I} \not\models \boldsymbol{q} \quad \text{iff} \quad \begin{array}{l} \text{either all contacts of } \mathcal{W}_\bullet \text{ are in } T^{\mathcal{I}} \text{ and all contacts of } \mathcal{W}_\circ \text{ are in } F^{\mathcal{I}} \\ \text{or all contacts of } \mathcal{W}_\bullet \text{ are in } F^{\mathcal{I}} \text{ and all contacts of } \mathcal{W}_\circ \text{ are in } T^{\mathcal{I}}. \end{array} $$

It is straightforward to check that n-bikes $\mathcal{B}$ satisfying the conditions of the lemma always exist: The F- and T-neighbourhoods of $\mathcal{B}$ can be kept disjoint by taking $> 2|\boldsymbol{q}|$ contact-distance between the F- and T-connections of each of the cogwheels in $\mathcal{B}$ and, by choosing, say, ${}^{*}\mathsf{t}^{k} = {}^{*}t_1^{k}$ and ${}^{*}\mathsf{f}^{k} = {}^{*}f_2^{k}$ for all other k and $* = \bullet, \circ$, conditions (22) and (23) hold in both cogwheels.

Proof. The implication ($\Rightarrow$) of Lemma 27.7 clearly holds for any n-bike $\mathcal{B}$ by the ($\Rightarrow$) direction of Lemma 27.3. To show ($\Leftarrow$), suppose $\mathcal{B}$ is as above, and $\mathcal{I}$ is a model of cov_A and $\mathcal{B}$ such that (31) holds. The proof is via excluding all possible locations in $\mathcal{B}$ for the image $h(\boldsymbol{q})$ of a potential subhomomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. As we discussed above, we have the eight cases in Fig. 8. In line with (H2), in each of these eight cases, we track the location of $h(f_1)$, and exclude all options for it. Throughout, our arguments will use the h-shift property in (20) without explicit reference.

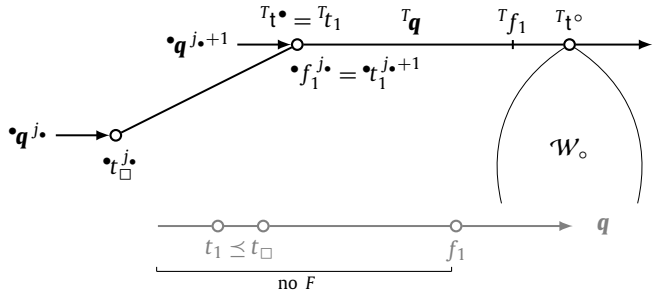
First, we deal with the cases when $h(\boldsymbol{q}) \cap {}^{T}\boldsymbol{q} \neq \emptyset$:

$(1)^{T}$ $h(\boldsymbol{q})$ starts in $\mathcal{W}_\bullet$ and $h({}^{\bullet}\mathsf{t}) \prec_{h(\boldsymbol{q})} {}^{T}\mathsf{t}^{\bullet}$.

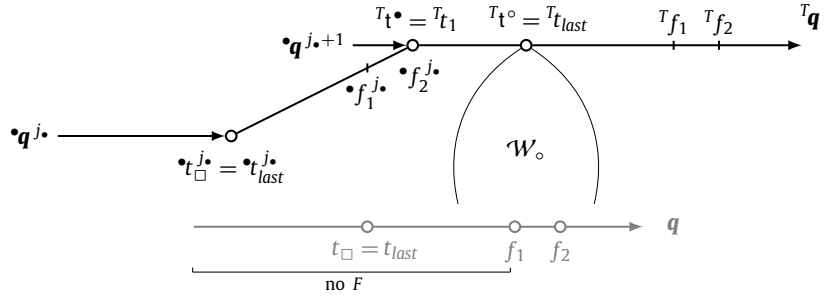


Then $h(\boldsymbol{q})$ definitely properly intersects the T-neighbourhood of $\mathcal{W}_\bullet$, and it might also properly intersect the T-neighbourhood of $\mathcal{W}_\circ$. It follows from $h({}^{\bullet}\mathsf{t}) \prec_{h(\boldsymbol{q})} {}^{T}\mathsf{t}^{\bullet}$ that $h(f_1)$ is in ${}^{T}\boldsymbol{q}$ then $h(f_1) \prec_{{}^{T}\boldsymbol{q}} {}^{T}f_1$. As ${}^{\bullet}\mathsf{t}^{j_\bullet+1} = {}^{\bullet}t_1^{j_\bullet+1}$ and ${}^{\bullet}\mathsf{t}^{j_\bullet-k} \prec_{{}^{\bullet}\boldsymbol{q}^{j_\bullet-k}} {}^{\bullet}f_1^{j_\bullet-k}$ for all $k < |\boldsymbol{q}|$, $h(f_1)$ cannot be in the initial cog of neither ${}^{\bullet}\boldsymbol{q}^{j_\bullet+1}$ nor ${}^{\bullet}\boldsymbol{q}^{j_\bullet-k}$ for any $k \le |\boldsymbol{q}|$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. As ${}^{\bullet}\mathsf{t}^{j_\bullet-k} = {}^{\bullet}t_1^{j_\bullet-k}$ and ${}^{\bullet}\mathsf{f}^{j_\bullet-k} = {}^{\bullet}f_1^{j_\bullet-k}$ for all k with $0 < k \le |\boldsymbol{q}|$, $h(f_1)$ cannot be a contact of $\mathcal{W}_\bullet$ different from ${}^{T}\mathsf{t}^{\bullet}$, otherwise $h(t_1)$ is also a contact of $\mathcal{W}_\bullet$, contradicting (31). For the remaining options, we consider the two cases $f_1 \prec t_{last}$ and $t_{last} \prec f_1$:

If $f_1 \prec t_{last}$ then $\mathsf{t}^{\bullet} = t_1$ and $f_1 \prec \mathsf{t}^{\circ}$, and so ${}^{T}f_1 \prec_{{}^{T}\boldsymbol{q}} {}^{T}\mathsf{t}^{\circ}$. As there is no F-node preceding ${}^{T}f_1$ in ${}^{T}\boldsymbol{q}$, $h(f_1)$ is in $\mathcal{W}_\bullet$. As ${}^{\bullet}\mathsf{t}^{j_\bullet} = {}^{\bullet}t_{\square}^{j_\bullet}$ and ${}^{\bullet}\mathsf{f}^{j_\bullet} = {}^{\bullet}f_1^{j_\bullet}$, $h(f_1)$ cannot be the contact ${}^{T}\mathsf{t}^{\circ} = {}^{\circ}f_1^{j_\circ}$, otherwise $h(t_\square)$ is also a contact of $\mathcal{W}_\bullet$, contradicting (31). As there is no F-node preceding f_1 in $\boldsymbol{q}$, there are no other options for $h(f_1)$ in $\mathcal{W}_\bullet$.



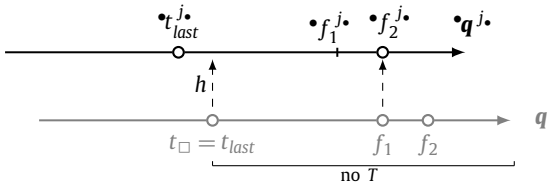
If $t_{last} \prec f_1$ then $\mathrm{t}^\bullet = t_1$ and $\mathrm{t}^\circ = t_{last} = t_\Box$, and so $^T\mathrm{t}^\circ = {}^Tt_{last} \prec_{^T\boldsymbol{q}} {}^Tf_1$. As there is no F-node preceding $^Tf_1$ in $^T\boldsymbol{q}$, $h(f_1)$ is either in $\mathcal{W}_\bullet$ or in $\mathcal{W}_\circ$.



– First, we exclude the remaining options in $\mathcal{W}_\bullet$. As $^\bullet\mathrm{t}^{j_\bullet} = {}^\bullet t_\Box^{j_\bullet}$ and $^\bullet\mathrm{f}^{j_\bullet} = {}^\bullet f_2^{j_\bullet}$, we cannot have $h(f_1) = {}^\bullet f_1^{j_\bullet}$, otherwise both $h(t_\Box)$ and $h(f_2)$ are contacts of $\mathcal{W}_\bullet$, contradicting (31). And if $h(f_1)$ is the contact $^T\mathrm{t}^\bullet = {}^\bullet f_2^{j_\bullet}$, then we track the location of $h(t_{last})$. As $h(\boldsymbol{q})$ starts in $\mathcal{W}_\bullet$ and $h(t_{last}) \prec_{h(\boldsymbol{q})} h(f_1)$, $h(t_{last})$ is in $\mathcal{W}_\bullet$. As $^\bullet\mathrm{t}^{j_\bullet+1} = {}^\bullet t_1^{j_\bullet+1}$ and $t_1 \prec t_{last}$, $h(t_{last})$ cannot be in the initial cog of $^\bullet\boldsymbol{q}^{j_\bullet+1}$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. Thus, we have

$$\delta_{h(\boldsymbol{q})}\big(h(t_{last}), {}^\bullet f_2^{j_\bullet}\big) = \delta_{h(\boldsymbol{q})}\big(h(t_{last}), h(f_1)\big) = \delta(t_{last}, f_1) < \delta(t_{last}, f_2) = \delta_{^\bullet\boldsymbol{q}^{j_\bullet}}\big({}^\bullet t_{last}^{j_\bullet}, {}^\bullet f_2^{j_\bullet}\big),$$

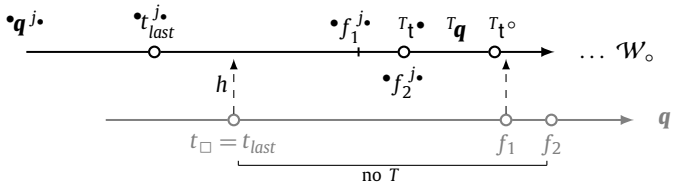
and so $h(t_{last})$ is a node between $^\bullet t_{last}^{j_\bullet} = {}^\bullet t_{t_\Box}^{j_\bullet}$ and $^\bullet f_2^{j_\bullet}$. But there is no such T-node in $^\bullet\boldsymbol{q}^{j_\bullet}$. As the only F-node preceding f_2 in $\boldsymbol{q}$ is f_1, there are no other options for $h(f_1)$ in $\mathcal{W}_\bullet$.



– If $h(f_1)$ is in $\mathcal{W}_\circ$ then $^T\mathrm{t}^\circ = {}^Tt_{last} \preceq_{h(\boldsymbol{q})} h(f_1)$. We track the location of $h(t_{last})$. As $h(\mathrm{t}^\bullet) \prec_{h(\boldsymbol{q})} {}^T\mathrm{t}^\bullet$ by our assumption, we have $h(t_{last}) = h(\mathrm{t}^\circ) \prec_{h(\boldsymbol{q})} {}^T\mathrm{t}^\circ$, and so either $h(t_{last})$ is in $\mathcal{W}_\bullet$ and $h(t_{last}) \prec_{h(\boldsymbol{q})} {}^T\mathrm{t}^\bullet$, or $h(t_{last})$ is in $^T\boldsymbol{q}$. In the former case, as $^\bullet\mathrm{t}^{j_\bullet+1} = {}^\bullet t_1^{j_\bullet+1}$ and $t_1 \prec t_{last}$, $h(t_{last})$ cannot be in the initial cog of $^\bullet\boldsymbol{q}^{j_\bullet+1}$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. Thus, we have

$$\delta_{h(\boldsymbol{q})}\big(h(t_{last}), {}^\bullet f_2^{j_\bullet}\big) = \delta_{h(\boldsymbol{q})}\big(h(t_{last}), {}^T\mathrm{t}^\bullet\big) < \delta_{h(\boldsymbol{q})}\big(h(t_{last}), h(f_1)\big) = \\ \delta(t_{last}, f_1) < \delta(t_{last}, f_2) = \delta_{^\bullet\boldsymbol{q}^{j_\bullet}}\big({}^\bullet t_{last}^{j_\bullet}, {}^\bullet f_2^{j_\bullet}\big),$$

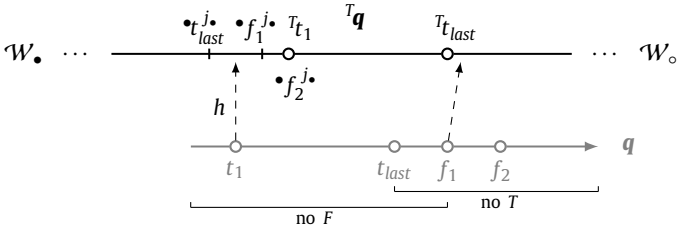
and so $h(t_{last})$ is a node between $^\bullet t_{last}^{j_\bullet} = {}^\bullet t_{t_\Box}^{j_\bullet}$ and $^\bullet f_2^{j_\bullet}$. But there is no such T-node in $^\bullet\boldsymbol{q}^{j_\bullet}$.



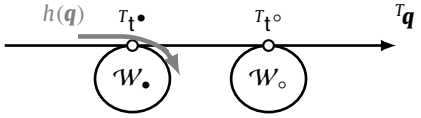
So suppose that $h(\mathrm{t}^\circ) = h(t_{last})$ is in ${}^T\boldsymbol{q}$. Now we track the location of $h(\mathrm{t}^\bullet) = h(t_1)$ in $\mathcal{W}_\bullet$. As $\bullet\mathrm{t}^{j_\bullet+1} = \bullet t_1^{j_\bullet+1}$, $h(t_1)$ cannot be in the initial cog of $\bullet\boldsymbol{q}^{j_\bullet+1}$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. Thus, we have

$$\delta_{h(\boldsymbol{q})}\big(h(t_1), \bullet f_2^{j_\bullet}\big) = \delta_{h(\boldsymbol{q})}\big(h(\mathrm{t}^\bullet), {}^T\mathrm{t}^\bullet\big) = \delta_{{}^T\boldsymbol{q}}\big(h(\mathrm{t}^\circ), {}^T\mathrm{t}^\circ\big) = \delta_{{}^T\boldsymbol{q}}\big(h(t_{last}), {}^T t_{last}\big) \leq$$
$$\delta_{h(\boldsymbol{q})}\big(h(t_{last}), h(f_1)\big) = \delta(t_{last}, f_1) \prec \delta(t_{last}, f_2) = \delta_{\bullet\boldsymbol{q}^{j_\bullet}}\big(\bullet t_{last}^{j_\bullet}, \bullet f_2^{j_\bullet}\big),$$

and so $h(t_1)$ is a node between $\bullet t_{last}^{j_\bullet}$ and $\bullet f_2^{j_\bullet}$. But there is no such T-node in $\bullet\boldsymbol{q}^{j_\bullet}$.

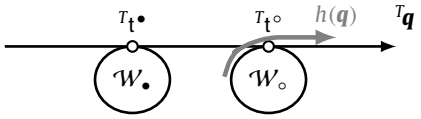


(2)^T $h(\boldsymbol{q})$ starts in ${}^T\boldsymbol{q}$ and ends in $\mathcal{W}_\bullet$.



Then $h(\boldsymbol{q})$ properly intersects the T-neighbourhood of $\mathcal{W}_\bullet$ only. As $\mathrm{t}^\bullet = t_1$ and $t_1 \prec f_1$ by (21), $h(f_1)$ is in $\mathcal{W}_\bullet$ and ${}^T\mathrm{t}^\bullet = {}^T t_1 \preceq_{h(\boldsymbol{q})} h(f_1)$, otherwise there is not enough room for $h(\boldsymbol{q})$ in ${}^T\boldsymbol{q}$. Now we track the location of $h(t_1)$. Again, $h(t_1)$ is in $\mathcal{W}_\bullet$ and ${}^T\mathrm{t}^\bullet = {}^T t_1 \preceq_{h(\boldsymbol{q})} h(t_1)$ and otherwise there is not enough room for $h(\boldsymbol{q})$ in ${}^T\boldsymbol{q}$. As the part of $\boldsymbol{q}$ preceding t_1 is empty (containing no F- or T-nodes), if $h\colon \boldsymbol{q} \to \mathcal{I}$ is a subhomomorphism, then we can modify it to obtain a subhomomorphism from $\boldsymbol{q}$ to the restriction of $\mathcal{I}$ to $\mathcal{W}_\bullet$, which contradicts Lemma 27.3 by (31).

(3)^T $h(\boldsymbol{q})$ starts in $\mathcal{W}_\circ$ and ends in ${}^T\boldsymbol{q}$.

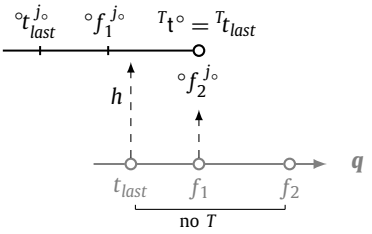


Then $h(\boldsymbol{q})$ properly intersects the T-neighbourhood of $\mathcal{W}_\circ$ only. As ${}^\circ\mathrm{t}^{j_\circ+1} \prec_{\circ\boldsymbol{q}^{j_\circ+1}} {}^\circ f_1^{j_\circ+1}$, $h(f_1)$ cannot be in the initial cog of ${}^\circ\boldsymbol{q}^{j_\circ+1}$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. As ${}^\circ\mathrm{t}^{j_\circ-k} = {}^\circ t_1^{j_\circ-k}$ and ${}^\circ\mathrm{f}^{j_\circ-k} = {}^\circ f_1^{j_\circ-k}$ for all k with $0 < k \leq |\boldsymbol{q}|$, $h(f_1)$ cannot be a contact of $\mathcal{W}_\circ$ different from ${}^T\mathrm{t}^\circ$, otherwise $h(t_1)$ is also a contact, contradicting (31). To exclude the remaining options, we consider the two cases $f_1 \prec t_{last}$ and $t_{last} \prec f_1$:

If $f_1 \prec t_{last}$ then ${}^T f_1 \prec_{{}^T\boldsymbol{q}} {}^T\mathrm{t}^\circ$, and so $h(f_1)$ cannot be in ${}^T\boldsymbol{q}$, otherwise there is not enough room for $h(\boldsymbol{q})$ in ${}^T\boldsymbol{q}$. As ${}^\circ\mathrm{t}^{j_\circ} = {}^\circ t_1^{j_\circ}$ and ${}^\circ\mathrm{f}^{j_\circ} = {}^\circ f_1^{j_\circ}$, $h(f_1)$ cannot be the contact ${}^T\mathrm{t}^\circ = {}^\circ f_1^{j_\circ}$, otherwise $h(t_1)$ is also a contact of $\mathcal{W}_\circ$, contradicting (31). As there is no F-node preceding f_1 in $\boldsymbol{q}$, there are no other options for $h(f_1)$ in $\mathcal{W}_\circ$.

If $t_{last} \prec f_1$ then $\mathrm{t}^\circ = t_{last}$, and so ${}^T\mathrm{t}^\circ = {}^T t_{last} \prec_{{}^T\boldsymbol{q}} {}^T f_1$. As there is no F-node preceding ${}^T f_1$ in ${}^T\boldsymbol{q}$, either $h(f_1) \preceq_{h(\boldsymbol{q})} {}^T\mathrm{t}^\circ$ and $h(f_1)$ is in $\mathcal{W}_\circ$, or $h(f_1) = {}^T f_1$, otherwise there is not enough room for $h(\boldsymbol{q})$ in ${}^T\boldsymbol{q}$. We will exclude both:

- Suppose first that $h(f_1)$ is in $\mathcal{W}_\circ$. As ${}^\circ\mathrm{t}^{j_\circ} = {}^\circ t_1^{j_\circ}$ and ${}^\circ\mathrm{f}^{j_\circ} = {}^T\mathrm{t}^\circ$ is either ${}^\circ f_1^{j_\circ}$ or ${}^\circ f_2^{j_\circ}$, we cannot have $h(f_1) = {}^\circ f_1^{j_\circ}$: If ${}^\circ\mathrm{f}^{j_\circ} = {}^\circ f_1^{j_\circ}$ then because otherwise $h(t_1)$ is also a contact of $\mathcal{W}_\circ$, and if ${}^\circ\mathrm{f}^{j_\circ} = {}^\circ f_2^{j_\circ}$ then because otherwise both $h(t_1)$ and $h(f_2)$ are contacts of $\mathcal{W}_\circ$, contradicting (35) in both cases. As the only F-node preceding f_2 in $\boldsymbol{q}$ is f_1, the only remaining option for $h(f_1)$ being in $\mathcal{W}_\circ$ is when $h(f_1) = {}^\circ\mathrm{f}^{j_\circ} = {}^\circ f_2^{j_\circ}$. Now we track the location of $h(t_{last})$.



As ${}^{\circ}\mathfrak{t}^{j_\circ+1} = {}^{\circ}t_1^{j_\circ+1}$, $h(t_{last})$ cannot be in the initial cog of ${}^{\circ}\boldsymbol{q}^{j_\circ+1}$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. Thus, we have

$$\delta_{h(\boldsymbol{q})}\big(h(t_{last}), {}^{\circ}f_2^{j_\circ}\big) = \delta_{h(\boldsymbol{q})}\big(h(t_{last}), h(f_1)\big) = \delta(t_{last}, f_1) < \delta(t_{last}, f_2) = \delta_{{}^{\circ}\boldsymbol{q}^{j_\circ}}\big({}^{\circ}t^{j_\circ}_{last}, {}^{\circ}f_2^{j_\circ}\big).$$

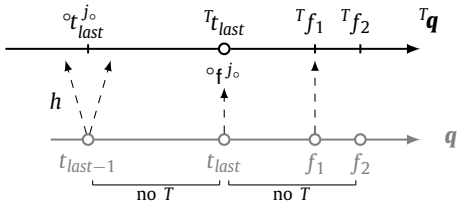
As ${}^{\circ}\mathfrak{t}^{j_\circ} = {}^{\circ}t_1^{j_\circ} \prec_{{}^{\circ}\boldsymbol{q}^{j_\circ}} {}^{\circ}t^{j_\circ}_{last}$, it follows that $h(t_{last})$ is between ${}^{\circ}t^{j_\circ}_{last}$ and ${}^{\circ}f_2^{j_\circ}$. But there is no such T-node in ${}^{\circ}\boldsymbol{q}^{j_\circ}$, and so $h(f_1)$ cannot be in $\mathcal{W}_\circ$.

– If $h(f_1) = {}^{c}f_1$ then $h(t_{last}) = {}^{T}t_{last} = {}^{T}\mathfrak{t}^{\circ}$. We track the location of $h(t_{last-1})$. As ${}^{\circ}\mathfrak{t}^{j_\circ+1} = {}^{\circ}t_1^{j_\circ+1}$, $h(t_{last-1})$ cannot be in the initial cog of ${}^{\circ}\boldsymbol{q}^{j_\circ+1}$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. As ${}^{\circ}t^{j_\circ}_{last} \prec_{{}^{\circ}\boldsymbol{q}^{j_\circ}} {}^{\circ}f_1^{j_\circ} \preceq_{{}^{\circ}\boldsymbol{q}^{j_\circ}} {}^{\circ}\mathfrak{f}^{j_\circ} = h(t_{last})$, we have ${}^{\circ}t_1^{j_\circ} \preceq_{{}^{\circ}\boldsymbol{q}^{j_\circ}} {}^{\circ}t^{j_\circ}_{last-1} \prec_{{}^{\circ}\boldsymbol{q}^{j_\circ}} h(t_{last-1}) \prec_{{}^{\circ}\boldsymbol{q}^{j_\circ}} {}^{\circ}\mathfrak{f}^{j_\circ}$, and so $h(t_{last-1}) = {}^{\circ}t^{j_\circ}_{last}$ must hold (as t_{last} is the only T-node succeeding t_{last-1} in $\boldsymbol{q}$). Therefore,

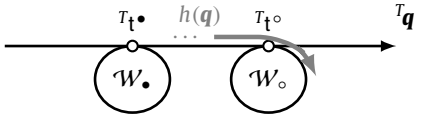
$$\delta(t_{last-1}, t_{last}) = \delta_{h(\boldsymbol{q})}\big(h(t_{last-1}), h(t_{last})\big) =$$

$$= \begin{cases} \delta_{{}^{\circ}\boldsymbol{q}^{j_\circ}}\big({}^{\circ}t^{j_\circ}_{last}, {}^{\circ}f_2^{j_\circ}\big) = \delta(t_{last}, f_2), & \text{if } \delta(t_{last-1}, t_{last}) = \delta(t_{last}, f_1), \\ \delta_{{}^{\circ}\boldsymbol{q}^{j_\circ}}\big({}^{\circ}t^{j_\circ}_{last}, {}^{\circ}f_1^{j_\circ}\big) = \delta(t_{last}, f_1), & \text{if } \delta(t_{last-1}, t_{last}) \neq \delta(t_{last}, f_1), \end{cases}$$

with both cases being impossible.

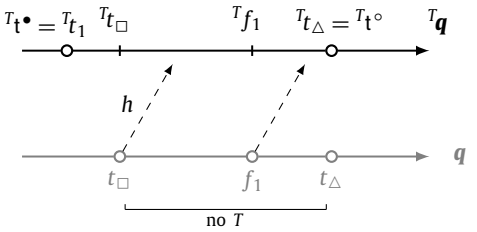


$(4)^T$ $h(\boldsymbol{q})$ ends in $\mathcal{W}_\circ$ and ${}^{T}\mathfrak{t}^{\circ} \prec_{h(\boldsymbol{q})} h(\mathfrak{t}^{\circ})$.



Then $h(\boldsymbol{q})$ definitely properly intersects the T-neighbourhood of $\mathcal{W}_\circ$, and it might also properly intersect the T-neighbourhood of $\mathcal{W}_\bullet$. As ${}^{\circ}f_1^{j_\circ+\ell} \preceq_{{}^{\circ}\boldsymbol{q}^{j_\circ+\ell}} {}^{\circ}\mathfrak{f}^{j_\circ+\ell}$ for all $\ell \leq |\boldsymbol{q}|$, $h(f_1)$ cannot be in the final cog of ${}^{\circ}\boldsymbol{q}^{j_\circ+\ell}$ for any $\ell \leq |\boldsymbol{q}|$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. As ${}^{\circ}\mathfrak{t}^{j_\circ+\ell} = {}^{\circ}t_1^{j_\circ+\ell}$ and ${}^{\circ}\mathfrak{f}^{j_\circ+\ell} = {}^{\circ}f_1^{j_\circ+\ell}$ for all ℓ with $1 \leq \ell \leq |\boldsymbol{q}|$, $h(f_1)$ cannot be a contact of $\mathcal{W}_\circ$ different from ${}^{T}\mathfrak{t}^{\circ}$, otherwise $h(t_1)$ is also a contact, contradicting (31). As there is no F-node preceding f_1 in $\boldsymbol{q}$, it follows that $h(f_1)$ must be in ${}^{T}\boldsymbol{q}$. To exclude the remaining options, we consider the two cases $f_1 \prec t_{last}$ and $t_{last} \prec f_1$:

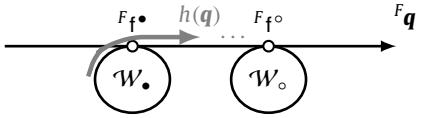
If $f_1 \prec t_{last}$ then $\mathfrak{t}^{\circ} = t_\triangle$, where $t_\triangle$ is the first T-node succeeding f_1. As by our assumption $h(\boldsymbol{q})$ ends in $\mathcal{W}_\circ$ and ${}^{T}\mathfrak{t}^{\circ} \prec_{h(\boldsymbol{q})} h(\mathfrak{t}^{\circ})$, it follows that ${}^{T}f_1 \prec_{{}^{T}\boldsymbol{q}} h(f_1) \preceq_{{}^{T}\boldsymbol{q}} {}^{T}\mathfrak{t}^{\circ} = {}^{T}t_\triangle$. Now we track the location of $h(t_\square)$ for the last T-node $t_\square$ preceding f_1. As $t_1 \preceq t_\square \prec f_1$, it follows that $h(t_\square)$ is between ${}^{T}t_\square$ and $h(f_1)$, and so between ${}^{T}t_\square$ and ${}^{T}t_\triangle$. But there is no such T-node in ${}^{T}\boldsymbol{q}$.



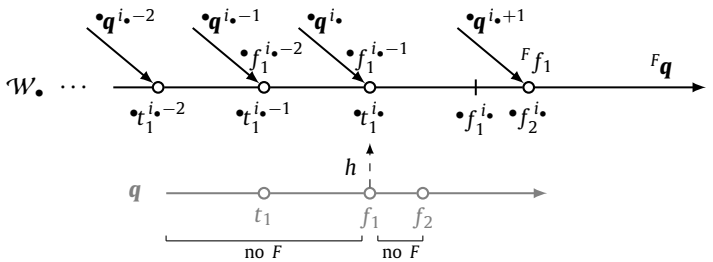
If $t_{last} \prec f_1$ then $\mathsf{t}^\circ = t_{last}$. As $h(\boldsymbol{q})$ ends in $\mathcal{W}_\circ$ and ${}^T\mathsf{t}^\circ \prec_{h(\boldsymbol{q})} h(\mathsf{t}^\circ)$, it follows that ${}^T\mathsf{t}^\circ \prec_{h(\boldsymbol{q})} h(t_{last}) \prec_{h(\boldsymbol{q})} h(f_1)$. Thus, $h(f_1)$ cannot be in ${}^T\boldsymbol{q}$, leaving us no options.

Next, we deal with the cases when $h(\boldsymbol{q}) \cap {}^F\boldsymbol{q} \neq \emptyset$:

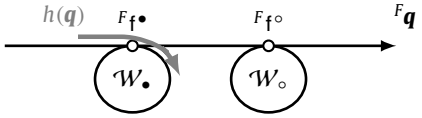
$(1)^F$ $h(\boldsymbol{q})$ starts in $\mathcal{W}_\bullet$ and $h(\mathsf{f}^\bullet) \prec_{h(\boldsymbol{q})} {}^F\mathsf{f}^\bullet$.



Then $h(\boldsymbol{q})$ definitely properly intersects the F-neighbourhood of $\mathcal{W}_\bullet$, and it might also properly intersect the F-neighbourhood of $\mathcal{W}_\circ$. As $\mathsf{f}^\bullet = f_1$, we have $h(f_1) \prec_{h(\boldsymbol{q})} {}^F\mathsf{f}^\bullet$ and $h(f_1)$ is in $\mathcal{W}_\bullet$. As ${}^\bullet\mathsf{t}^{i_\bullet+1} \prec_{{}^\bullet\boldsymbol{q}^{i_\bullet+1}} {}^\bullet f_1^{i_\bullet+1}$ for either choice of ${}^\bullet\mathsf{t}^{i_\bullet+1}$, $h(f_1)$ cannot be in the initial cog of ${}^\bullet\boldsymbol{q}^{i_\bullet+1}$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. As ${}^\bullet\mathsf{t}^{j_\bullet-k} = {}^\bullet t_1^{j_\bullet-k}$ and ${}^\bullet\mathsf{f}^{j_\bullet-k} = {}^\bullet f_1^{j_\bullet-k}$ for all k with $0 < k \leq |\boldsymbol{q}|$, $h(f_1)$ cannot be a contact of $\mathcal{W}_\bullet$ different from ${}^F\mathsf{f}^\bullet$, otherwise $h(t_1)$ is also a contact of $\mathcal{W}_\bullet$, contradicting (31). And as ${}^\bullet\mathsf{t}^{i_\bullet} = {}^\bullet t_1^{i_\bullet}$ and ${}^\bullet\mathsf{f}^{i_\bullet} = {}^\bullet f_2^{i_\bullet}$, we cannot have $h(f_1) = {}^\bullet f_1^{i_\bullet}$, otherwise both $h(t_1)$ and $h(f_2)$ are contacts of $\mathcal{W}_\bullet$, again contradicting (31). As the only F-node preceding f_2 in $\boldsymbol{q}$ is f_1, there are no more options for $h(f_1)$.

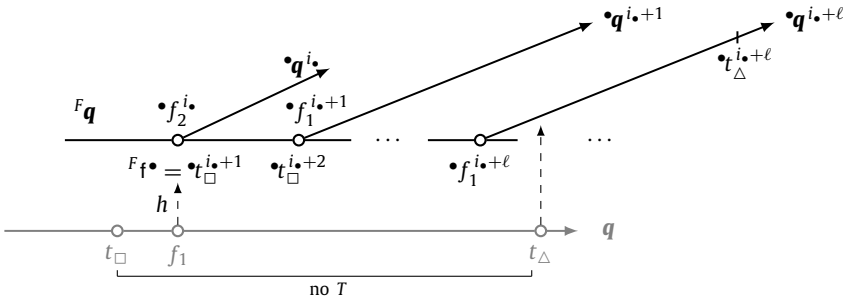


$(2)^F$ $h(\boldsymbol{q})$ starts in ${}^F\boldsymbol{q}$ and ends in $\mathcal{W}_\bullet$.



Then $h(\boldsymbol{q})$ properly intersects the F-neighbourhood of $\mathcal{W}_\bullet$ only. As $\mathsf{f}^\bullet = f_1$, we cannot have $h(f_1) \prec_{h(\boldsymbol{q})} {}^F\mathsf{f}^\bullet = {}^F f_1$, otherwise there is not enough room for $h(\boldsymbol{q})$ in ${}^F\boldsymbol{q}$. Thus, ${}^F\mathsf{f}^\bullet \preceq_{h(\boldsymbol{q})} h(f_1)$ and $h(f_1)$ is in $\mathcal{W}_\bullet$. As ${}^\bullet f_1^{i_\bullet+\ell} \preceq_{{}^\bullet\boldsymbol{q}^{i_\bullet+\ell}} {}^\bullet\mathsf{f}^{i_\bullet+\ell}$ for all $\ell \leq |\boldsymbol{q}|$, $h(f_1)$ cannot be in the final cog of ${}^\bullet\boldsymbol{q}^{i_\bullet}$ for any $\ell \leq |\boldsymbol{q}|$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. To exclude the remaining options, we consider the two cases $f_1 \prec t_{last}$ and $t_{last} \prec f_1$:

If $f_1 \prec t_{last}$ then ${}^\bullet\mathsf{t}^{i_\bullet+\ell} = {}^\bullet t_\square^{i_\bullet+\ell}$ and ${}^\bullet\mathsf{f}^{i_\bullet+\ell} = {}^\bullet f_1^{i_\bullet+\ell}$ for all ℓ with $1 \leq \ell \leq |\boldsymbol{q}|$, where $t_\square$ is the last T-node preceding f_1. Thus, $h(f_1)$ cannot be a contact of $\mathcal{W}_\bullet$ different from ${}^F\mathsf{f}^\bullet$, otherwise $h(t_\square)$ is also a contact, contradicting (31). As there is no F-node preceding f_1 in $\boldsymbol{q}$, the only remaining option for $h(f_1)$ is $h(f_1) = {}^F\mathsf{f}^\bullet$. Then all contacts of $\mathcal{W}_\bullet$ are in $F^{\mathcal{I}}$ by (31). We track the location of $h(t_\triangle)$ for the first T-node $t_\triangle$ succeeding f_1 in $\boldsymbol{q}$.



As $^{\bullet}\mathrm{f}^{i_\bullet} = {}^{\bullet}f_2^{i_\bullet}$, $h(t_\triangle)$ cannot be in the final cog of $^{\bullet}\boldsymbol{q}^{i_\bullet}$, otherwise $h(f_2)$ is also in that cog and there is not enough room for $h(\boldsymbol{q})$ in that cog (unlike in Example 27.6 (i)). Further, $h(t_\triangle)$ cannot be a contact of $\mathcal{W}_\bullet$, as all contacts of $\mathcal{W}_\bullet$ are in $F^{\mathcal{I}}$. As there is no T-node between $t_\square$ and f_1 in $\boldsymbol{q}$, $h(t_\triangle)$ must be in the final cog of $^{\bullet}\boldsymbol{q}^{i_\bullet+\ell}$ for some ℓ with $1 \leq \ell \leq |\boldsymbol{q}|$. Then

$$\delta_{^{\bullet}\boldsymbol{q}^{i_\bullet+\ell}}\big({}^{\bullet}f_1^{i_\bullet+\ell}, h(t_\triangle)\big) < \delta_{h(\boldsymbol{q})}\big({}^{F}\mathrm{f}^{\bullet}, h(t_\triangle)\big) = \delta_{h(\boldsymbol{q})}\big(h(f_1), h(t_\triangle)\big) = \delta(f_1, t_\triangle) = \delta_{^{\bullet}\boldsymbol{q}^{i_\bullet+\ell}}\big({}^{\bullet}f_1^{i_\bullet+\ell}, {}^{\bullet}t_{\triangle}^{i_\bullet+\ell}\big),$$

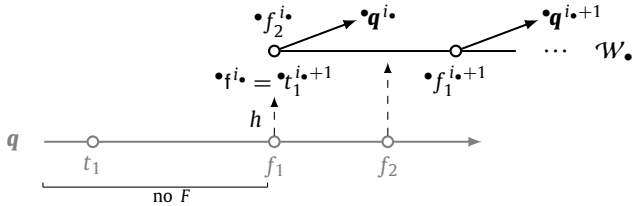
and so $h(t_\triangle)$ is between $^{\bullet}f_1^{i_\bullet+\ell}$ and $^{\bullet}t_{\triangle}^{i_\bullet+\ell}$. But there is no such T-node in $^{\bullet}\boldsymbol{q}^{i_\bullet+\ell}$.

If $t_{last} \prec f_1$ then there are two cases, depending on the relationship between $\delta(f_1, f_2)$ and $\delta(t_1, f_1)$:

– If $\delta(f_1, f_2) < \delta(t_1, f_1)$ then $^{\bullet}\mathrm{t}^{i_\bullet+\ell} = {}^{\bullet}t_1^{i_\bullet+\ell}$ and $^{\bullet}\mathrm{f}^{i_\bullet+\ell} = {}^{\bullet}f_1^{i_\bullet+\ell}$, for all ℓ with $1 \leq \ell \leq |\boldsymbol{q}|$. Thus, $h(f_1)$ cannot be a contact of $\mathcal{W}_\bullet$ different from $^{F}\mathrm{f}^{\bullet}$, otherwise $h(t_1)$ is also a contact, contradicting (31). As there is no F-node preceding f_1 in $\boldsymbol{q}$, the only remaining option for $h(f_1)$ is $h(f_1) = {}^{F}\mathrm{f}^{\bullet}$. Next, we track the location of $h(f_2)$. As $^{\bullet}\mathrm{f}^{i_\bullet} = {}^{\bullet}f_2^{i_\bullet}$, $h(f_2)$ cannot be in the final cog of $^{\bullet}\boldsymbol{q}^{i_\bullet}$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog (unlike in Example 27.6 (i)). Thus,

$$\delta_{h(\boldsymbol{q})}\big({}^{\bullet}t_1^{i_\bullet+1}, h(f_2)\big) = \delta_{h(\boldsymbol{q})}\big({}^{F}\mathrm{f}^{\bullet}, h(f_2)\big) = \delta_{h(\boldsymbol{q})}\big(h(f_1), h(f_2)\big) = \\ \delta(f_1, f_2) < \delta(t_1, f_1) = \delta_{^{\bullet}\boldsymbol{q}^{i_p w+1}}\big({}^{\bullet}t_1^{i_\bullet+1}, {}^{\bullet}f_1^{i_\bullet+1}\big),$$

and so $h(f_2)$ is between $^{\bullet}t_1^{i_\bullet+1}$ and $^{\bullet}f_1^{i_\bullet+1}$. But there is no such F-node in $^{\bullet}\boldsymbol{q}^{i_\bullet+1}$.



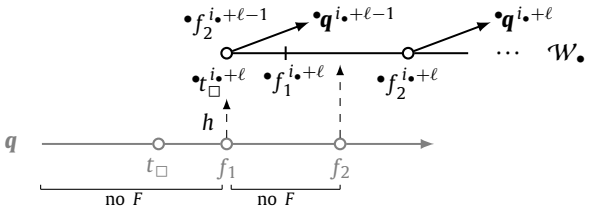
– If $\delta(f_1, f_2) \geq \delta(t_1, f_1)$ then $^{\bullet}\mathrm{t}^{i_\bullet+\ell} = {}^{\bullet}t_{\square}^{i_\bullet+\ell}$ and $^{\bullet}\mathrm{f}^{i_\bullet+\ell} = {}^{\bullet}f_2^{i_\bullet+\ell}$, for all ℓ with $1 \leq \ell \leq |\boldsymbol{q}|$, where $t_\square$ is the last T-node preceding f_1. Thus, $h(f_1) = {}^{\bullet}f_1^{i_\bullet+\ell}$ cannot hold for any ℓ with $1 \leq \ell \leq |\boldsymbol{q}|$, otherwise both $h(t_\square)$ and $h(f_2)$ are contacts of $\mathcal{W}_\bullet$, contradicting (31). As the only F-node preceding f_2 in $\boldsymbol{q}$ is f_1, the only remaining option for $h(f_1)$ is to be a contact of $\mathcal{W}_\bullet$, that is, $h(f_1) = {}^{\bullet}t_{\square}^{i_\bullet+\ell}$ for some ℓ with $1 \leq \ell \leq |\boldsymbol{q}|$. Again, we track the location of $h(f_2)$. As $^{\bullet}\mathrm{f}^{i_\bullet} = {}^{\bullet}f_2^{i_\bullet}$, $h(f_2)$ cannot be in the final cog of $^{\bullet}\boldsymbol{q}^{i_\bullet}$, otherwise there is not enough room for $h(\boldsymbol{q})$ in that cog. Thus,

$$\delta_{h(\boldsymbol{q})}\big({}^{\bullet}t_{\square}^{i_\bullet+\ell}, h(f_2)\big) = \delta_{h(\boldsymbol{q})}\big(h(f_1), h(f_2)\big) = \delta(f_1, f_2) \geq \delta(t_1, f_1) > \delta(t_\square, f_1) = \delta_{^{\bullet}\boldsymbol{q}^{i_\bullet+\ell}}\big({}^{\bullet}t_{\square}^{i_\bullet+\ell}, {}^{\bullet}f_1^{i_\bullet+\ell}\big).$$

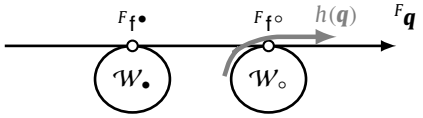
On the other hand,

$$\delta_{h(\boldsymbol{q})}\big({}^{\bullet}t_{\square}^{i_\bullet+\ell}, h(f_2)\big) = \delta_{h(\boldsymbol{q})}\big(h(f_1), h(f_2)\big) = \delta(f_1, f_2) < \delta(t_\square, f_2) = \delta_{^{\bullet}\boldsymbol{q}^{i_\bullet+\ell}}\big({}^{\bullet}t_{\square}^{i_\bullet+\ell}, {}^{\bullet}f_2^{i_\bullet+\ell}\big),$$

and so $h(f_2)$ is between $^{\bullet}f_1^{i_\bullet+\ell}$ and $^{\bullet}f_2^{i_\bullet+\ell}$. But there is no such F-node in $^{\bullet}\boldsymbol{q}^{i_\bullet+\ell}$.

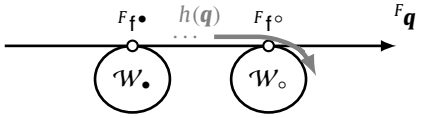


$(3)^F$ $h(\boldsymbol{q})$ starts in $\mathcal{W}_\circ$ and ends in ${}^F\boldsymbol{q}$.



Then $h(\boldsymbol{q})$ properly intersects the F-neighbourhood of $\mathcal{W}_\circ$ only. We have $h(\mathsf{f}^\circ) \preceq_{h(\boldsymbol{q})} {}^F\mathsf{f}^\circ$, as otherwise there is no room for $h(\boldsymbol{q})$ in ${}^F\boldsymbol{q}$. As $\mathsf{f}^\circ = f_2$, we have $h(f_1) \prec_{h(\boldsymbol{q})} h(f_2) \preceq_{h(\boldsymbol{q})} {}^F\mathsf{f}^\circ$ and $h(f_1)$ is in $\mathcal{W}_\circ$. We can exclude all possible locations for $h(f_1)$ by the same argument as in case $(1)^F$, with the F-neighbourhood of $\mathcal{W}_\circ$ in place of the F-neighbourhood of $\mathcal{W}_\bullet$.

$(4)^F$ $h(\boldsymbol{q})$ ends in $\mathcal{W}_\circ$ and ${}^F\mathsf{f}^\circ \prec_{h(\boldsymbol{q})} h(\mathsf{f}^\circ)$.



Then $h(\boldsymbol{q})$ definitely properly intersects the F-neighbourhood of $\mathcal{W}_\circ$, and it might also properly intersect the F-neighbourhood of $\mathcal{W}_\bullet$. As $\mathsf{f}^\bullet = f_1$ and $\mathsf{f}^\circ = f_2$, there is no F-node between ${}^F\mathsf{f}^\bullet$ and ${}^F\mathsf{f}^\circ$ in ${}^F\boldsymbol{q}$, and so ${}^F\mathsf{f}^\circ \preceq_{h(\boldsymbol{q})} h(f_1)$ and $h(f_1)$ is in $\mathcal{W}_\circ$. We can exclude all possible locations for $h(f_1)$ by the same argument as in case $(2)^F$, with the F-neighbourhood of $\mathcal{W}_\circ$ in place of the F-neighbourhood of $\mathcal{W}_\bullet$.

We excluded all possible locations in $\mathcal{B}$ for the image $h(\boldsymbol{q})$ of a potential subhomomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$, which completes the proof of Lemma 27.7. □

6.3. Representing clauses with shared literals

Suppose ψ is a 3CNF with n_ψ clauses of the form $\ell_1 \vee \ell_2 \vee \ell_3$, where each ℓ_i is a literal. We build an ABox $\mathcal{A}_{\boldsymbol{q},\psi}$ as follows. We let $n \geq (n_\psi + 2)(2|\boldsymbol{q}| + 1)$ and, for each propositional variable p in ψ, we take a fresh n-bike $\mathcal{B}^p$ having n-cogwheels $\mathcal{W}^p_\bullet$, $\mathcal{W}^p_\circ$ and satisfying the conditions in Lemma 27.7. We pick three nodes v_1, v_2 and v_3 in $\boldsymbol{q}$ such that each v_z is a T-node or an F-node, and $\mathsf{v}_1 \prec \mathsf{v}_2 \prec \mathsf{v}_3$. We call these three nodes the *special triple* of $\boldsymbol{q}$. Then, for every clause $c = (\ell^c_1 \vee \ell^c_2 \vee \ell^c_3)$ in ψ, we proceed as follows. We take a fresh copy ${}^c\boldsymbol{q}$ of $\boldsymbol{q}$, consider the copies v^c_1, v^c_2 and v^c_3 of the special triple in ${}^c\boldsymbol{q}$, and replace their F- or T-labels with A. Then, for $z = 1, 2, 3$, we glue v^c_z to a contact

(p1) in $\mathcal{W}^p_\bullet$ iff either $\ell^c_z = p$ and v_z is an F-node in $\boldsymbol{q}$, or $\ell^c_z = \neg p$ and v_z is a T-node in $\boldsymbol{q}$;
(p2) in $\mathcal{W}^p_\circ$ iff either $\ell^c_z = p$ and v_z is an T-node in $\boldsymbol{q}$, or $\ell^c_z = \neg p$ and v_z is a F-node in $\boldsymbol{q}$.

For example, if $\boldsymbol{q}$ looks like on the left-hand side of the picture below and $c = (p \vee \neg q \vee r)$, then we obtain the graph shown on the right-hand side of the picture with the n-cogwheels depicted as circles:

We call v^c_1, v^c_2 and v^c_3 *c-connections*, while the *c-neighbourhood* consists of those contacts in each of the three n-cogwheels whose contact-distance from its c-connection is $\leq |\boldsymbol{q}|$. For different clauses c, c', we pick the c- and c'-connections 'sharing' the same n-cogwheel $\mathcal{W}$ in such a way that the c- and c'-neighbourhoods are disjoint from each other and from the F- and T-neighbourhoods in $\mathcal{W}$. (We can do this as $n \geq (n_\psi + 2)(2|\boldsymbol{q}| + 1)$.) We treat the resulting labelled graph as an ABox, call it a (ψ, n)-*gadget* (*for* $\boldsymbol{q}$), and denote it by $\mathcal{A}_{\boldsymbol{q},\psi}$. Clearly, the size of $\mathcal{A}_{\boldsymbol{q},\psi}$ is polynomial in the sizes of $\boldsymbol{q}$ and ψ.

The following lemma is a consequence of the definition of $\mathcal{A}_{\boldsymbol{q},\psi}$, and the 'easy' ($\Rightarrow$) direction of Lemma 27.7.

Lemma 27.8. *If* $\mathsf{cov}_A, \mathcal{A}_{\boldsymbol{q},\psi} \not\models \boldsymbol{q}$, *then* ψ *is satisfiable.*

Proof. Suppose $\mathcal{I}$ is a model of cov_A and $\mathcal{A}_{\boldsymbol{q},\psi}$ such that $\mathcal{I} \not\models \boldsymbol{q}$. As for each variable p in ψ, the n-bike $\mathcal{B}^p$ satisfies the conditions in Lemma 27.7, either all contacts of the n-cogwheel $\mathcal{W}^p_\bullet$ are in $F^{\mathcal{I}}$ and all contacts of $\mathcal{W}^p_\circ$ are in $T^{\mathcal{I}}$, or all contacts of $\mathcal{W}^p_\bullet$ are in $T^{\mathcal{I}}$ and all contacts of $\mathcal{W}^p_\circ$ are in $F^{\mathcal{I}}$. As $\mathcal{I} \not\models \boldsymbol{q}$, for every clause $c = (\ell^c_1 \vee \ell^c_2 \vee \ell^c_3)$ in ψ, there is $z = 1, 2, 3$ such that either v_z is a T-node in $\boldsymbol{q}$ but $\mathsf{v}^c_z \in F^{\mathcal{I}}$, or v_z is an F-node in $\boldsymbol{q}$ but $\mathsf{v}^c_z \in T^{\mathcal{I}}$. Define an assignment $\mathfrak{a}$ by setting $\mathfrak{a}(\ell^c_z) = T$ for each clause c in ψ (and arbitrary otherwise). We claim that $\mathfrak{a}$ is well-defined in the sense that we never set both $\mathfrak{a}(p) = T$ and $\mathfrak{a}(\neg p) = T$. Indeed, suppose otherwise. Suppose also that the former is because of $\ell^{c_1}_{z_1}$ in a clause c_1 and the latter because of $\ell^{c_2}_{z_2}$ in a clause c_2.

Case 1: v_{z_1} is a T-node in $\boldsymbol{q}$ but $\mathsf{v}^{c_1}_{z_1} \in F^{\mathcal{I}}$. As $\mathfrak{a}(p) = T$ implies that $\ell^{c_1}_{z_1} = p$, by (p2) of the construction $\mathsf{v}^{c_1}_{z_1}$ is a contact in the n-cogwheel $\mathcal{W}^p_\circ$. So all contacts in $\mathcal{W}^p_\circ$ are in $F^{\mathcal{I}}$. On the other hand, $\mathfrak{a}(\neg p) = T$ implies that $\ell^{c_2}_{z_2} = \neg p$. If v_{z_2} is a T-node in $\boldsymbol{q}$ but $\mathsf{v}^{c_2}_{z_2} \in F^{\mathcal{I}}$, then $\mathsf{v}^{c_2}_{z_2}$ is a contact in $\mathcal{W}^p_\bullet$ by (p1), and so all contacts in $\mathcal{W}^p_\bullet$ are also in $F^{\mathcal{I}}$, a contradiction. And if v_{z_2} is an F-node in $\boldsymbol{q}$ but $\mathsf{v}^{c_2}_{z_2} \in T^{\mathcal{I}}$, then $\mathsf{v}^{c_2}_{z_2}$ is a contact in $\mathcal{W}^p_\circ$ by (p2), and so all contacts in $\mathcal{W}^p_\circ$ are in $T^{\mathcal{I}}$, a contradiction again.

Case 2: v_{z_1} is an F-node in $\boldsymbol{q}$ but $\mathsf{v}^{c_1}_{z_1} \in T^{\mathcal{I}}$. This case is similar and left to the reader.

Thus, the assignment $\mathfrak{a}$ is well-defined and makes true at least one literal in every clause in ψ. □

It remains to find some conditions on $\mathcal{A}_{\boldsymbol{q},\psi}$ that would guarantee that the converse of Lemma 27.8 also holds. So suppose that ψ is satisfiable under an assignment $\mathfrak{a}$. We define a model $\mathcal{I}_\mathfrak{a}$ of cov_A and $\mathcal{A}_{\boldsymbol{q},\psi}$ as follows:

For every variable p in ψ, we put

$$\begin{aligned} &\text{all contacts of } \mathcal{W}^p_\bullet \text{ to } T^{\mathcal{I}_\mathfrak{a}} \text{ and all contacts of } \mathcal{W}^p_\circ \text{ to } F^{\mathcal{I}_\mathfrak{a}}, \text{ whenever if } \mathfrak{a}(p) = T; \text{ and} \\ &\text{all contacts of } \mathcal{W}^p_\bullet \text{ to } F^{\mathcal{I}_\mathfrak{a}} \text{ and all contacts of } \mathcal{W}^p_\circ \text{ to } T^{\mathcal{I}_\mathfrak{a}}, \text{ whenever if } \mathfrak{a}(p) = F. \end{aligned} \tag{33}$$

We aim to find some conditions on $\mathcal{A}_{\boldsymbol{q},\psi}$ that would imply $\mathcal{I}_\mathfrak{a} \not\models \boldsymbol{q}$. Just like in the case of other ABoxes built up from copies of $\boldsymbol{q}$ before, we are looking for conditions that exclude all possible locations in $\mathcal{A}_{\boldsymbol{q},\psi}$ for the image $h(\boldsymbol{q})$ of a potential subhomomorphism $h\colon \boldsymbol{q} \to \mathcal{I}_\mathfrak{a}$. The definition of $\mathcal{A}_{\boldsymbol{q},\psi}$ allows flexibility

- in the choice of the special triple $\mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_3$ in $\boldsymbol{q}$, and
- also in the choices of the contacts in the c-neighbourhoods, for each clause c.

If we choose all these contacts in such a way that (22), (23) and the conditions of Lemma 27.7 hold then, by (33) and Lemma 27.7, we know that $h(\boldsymbol{q})$ must intersect with at least one ${}^c\boldsymbol{q}$ for some clause c. Therefore, the intersection of $h(\boldsymbol{q})$ with any of its n-cogwheels cannot go beyond its c-neighbourhoods. Further, we claim that,

$$\text{for any clause } c \text{ in } \psi, \text{ there is no subhomomorphism } h : \boldsymbol{q} \to \mathcal{I}_\mathfrak{a} \text{ such that } h(\mathsf{v}_z) = \mathsf{v}^c_z \text{ for all } z = 1, 2, 3. \tag{34}$$

(In particular, there is no $\boldsymbol{q} \to \mathcal{I}_\mathfrak{a}$ subhomomorphism mapping $\boldsymbol{q}$ onto ${}^c\boldsymbol{q}$.) Indeed, suppose on the contrary that there is such a subhomomorphism h for some c. Suppose $\mathfrak{a}(\ell^c_z) = T$ for some a. If $\ell^c_z = p$, then either v_z is an F-node in $\boldsymbol{q}$ but $\mathsf{v}^c_z \in T^{\mathcal{I}_\mathfrak{a}}$ as it is in $\mathcal{W}^p_\bullet$, or v_z is a T-node in $\boldsymbol{q}$ but $\mathsf{v}^c_z \in F^{\mathcal{I}_\mathfrak{a}}$ as it is in $\mathcal{W}^p_\circ$, both are impossible when $h(\mathsf{v}_z) = \mathsf{v}^c_z$. The case of $\ell^c_z = \neg p$ is dually symmetric. It follows that $\mathfrak{a}(\ell^c_z) \neq T$ for any $z = 1, 2, 3$, contrary to $\mathfrak{a}$ satisfying ψ.

By (34), $h(\boldsymbol{q})$ must *properly intersect* with at least one of the three n-cogwheels $\mathcal{W}^c_1$, $\mathcal{W}^c_2$ and $\mathcal{W}^c_3$ glued to ${}^c\boldsymbol{q}$ in the sense that $h(\boldsymbol{q}) \cap \mathcal{W}^c_z \not\subseteq \{\mathsf{v}^c_z\}$ for some $z = 1, 2, 3$. By (33) and Lemma 27.3, we may assume that $h(\boldsymbol{q}) \not\subseteq \mathcal{W}^c_z$ for any $z = 1, 2, 3$. Also by Lemma 27.3, we may assume that if $h(\boldsymbol{q})$ properly intersects with $\mathcal{W}^c_z$, then every node in $h(\boldsymbol{q}) \cap \mathcal{W}^c_z$ is in the c-neighbourhood of $\mathcal{W}^c_z$. As for $c \neq c'$ the c- and c'-neighbourhoods are disjoint, there is a unique c with $h(\boldsymbol{q})$ properly intersecting with one or two of the n-cogwheels $\mathcal{W}^c_1$, $\mathcal{W}^c_2$ and $\mathcal{W}^c_3$ glued to ${}^c\boldsymbol{q}$ (it cannot properly intersect with all three). It is easy to check that, by (34), all options for such a $h(\boldsymbol{q})$ are covered by the six cases $(1)^c$–$(6)^c$ in Fig. 9.

We aim to show that for every 2-CQ suitable contact choices always exist by actually providing an *algorithm* that, given *any* 2-CQ $\boldsymbol{q}$, describes contact choices that, for large enough n, are suitable for any 3CNF ψ and any (ψ, n)-gadget constructed from copies of $\boldsymbol{q}$. Just like in case of bikes, the different potential locations of a homomorphic image place different constraints on our choices. By following our heuristics choices (H1) and (H2) above, we will be able to use the same techniques as for bikes in the proof of Lemma 27.7. In light of (H1), our algorithm chooses $\mathsf{v}_1 = t_1$. This is because our assumption throughout is that $t_1 \prec f_1$ (cf. (21)), and so t_1 is the only node that is followed by at least two other F- or T-nodes (t_{last} and f_1) in every 2-CQ $\boldsymbol{q}$, even if $\boldsymbol{q}$ contains only two F-nodes and two T-nodes. Similarly, v_3 is chosen to be f_2, as in general f_2 is the only node that is preceded by at least two other F- or T-nodes (t_1 and f_1). And then v_2 is chosen from the two 'middle' nodes that are always present, either t_{last} or f_1. The choice of the $\prec$-smaller of t_{last} and f_1 as v_2 is motivated by (H2).

However, now the 3CNF ψ introduces some more 'variables' into our constraint system. In order to reduce the search space, we made some further choices in our heuristics:

(H3) Given any 2-CQ $\boldsymbol{q}$ and any assignment $\mathfrak{a}$ satisfying some 3CNF ψ, we give an algorithm describing choices suitable for achieving $\mathcal{I}_\mathfrak{a} \not\models \boldsymbol{q}$ for any $\boldsymbol{q}$ such that the choices do not depend on ψ and $\mathfrak{a}$, only on $\boldsymbol{q}$.

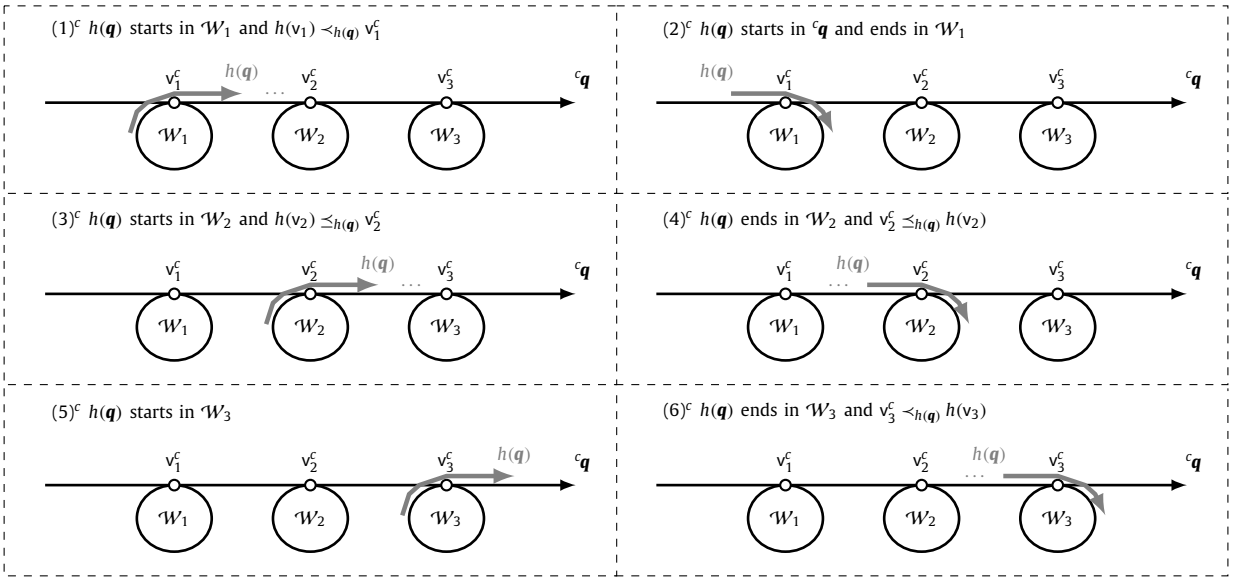
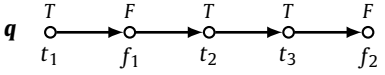


Fig. 9. Possible locations for $h(\boldsymbol{q})$ intersecting ${}^{c}\boldsymbol{q}$.

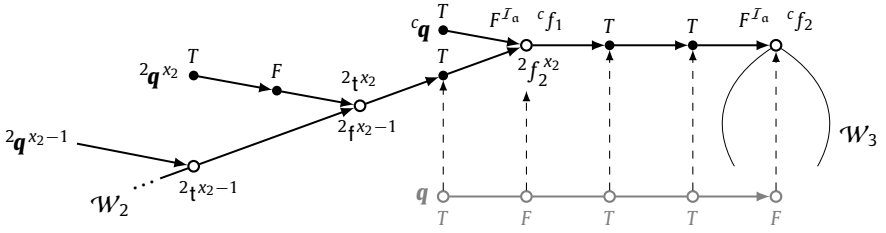
(H4) The algorithm chooses the contacts in the c-neighbourhoods of $\mathcal{A}_{\boldsymbol{q},\psi}$ uniformly, not depending on the particular clause c, but only on $\boldsymbol{q}$.

Yet another difficulty is that (34) is weaker than (32): It does not exclude cases when h 'fixes' two (but not all three) c-connections. Say, in case $(3)^c$ it can happen that $h(\boldsymbol{q})$ intersects $\mathcal{W}_2$ and $\mathcal{W}_3$, at least one of them properly, it does not intersect $\mathcal{W}_1$, and both $h(\mathsf{v}_2) = \mathsf{v}_2^c$ and $h(\mathsf{v}_3) = \mathsf{v}_3^c$ hold. The following example shows how the need for excluding such a situation might 'force' particular contact choices not only for the c-connection of the 'middle' cogwheel, but also *throughout* 'half' of its c-neighbourhood:

Example 27.9. Consider again the 2-CQ from Example 27.2.

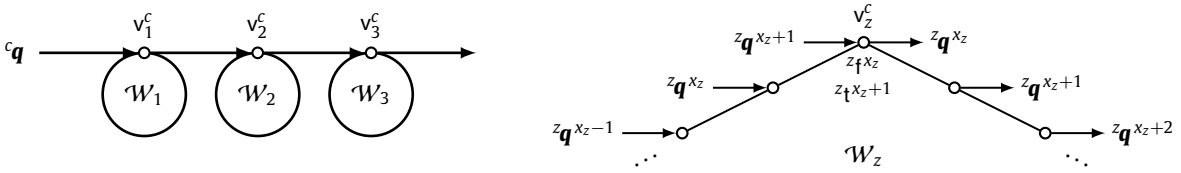


According to the above, as $f_1 \prec t_{last} = t_3$, we choose $\mathsf{v}_2 = f_1$ (and $\mathsf{v}_1 = t_1$, $\mathsf{v}_3 = f_2$). Suppose that, for some clause c, the contact in the 'middle' cogwheel $\mathcal{W}_2$ glued together with $\mathsf{v}_2^c = {}^{c}f_1$ is ${}^{2}\mathsf{f}^{x_2}$, in some copy ${}^{2}\boldsymbol{q}^{x_2}$ of $\boldsymbol{q}$. Then the argument in Example 27.6 (i) shows that we cannot choose ${}^{2}\mathsf{f}^{x_2} = {}^{2}f_1^{x_2}$, and so we must have ${}^{2}\mathsf{f}^{x_2} = {}^{2}f_2^{x_2}$. Now we have three choices for ${}^{2}\mathsf{t}^{x_2}$. However, if we choose either ${}^{2}\mathsf{t}^{x_2} = {}^{2}t_1^{x_2}$ or ${}^{2}\mathsf{t}^{x_2} = {}^{2}t_2^{x_2}$, and $\mathcal{I}_{\mathfrak{a}}$ is such that all contacts of $\mathcal{W}_1$, $\mathcal{W}_2$ and $\mathcal{W}_3$ are in $F^{\mathcal{I}_{\mathfrak{a}}}$, then we do have the following $h\colon \boldsymbol{q} \to \mathcal{I}_{\mathfrak{a}}$ homomorphism (see case $(3)^c$ in Fig. 9):



Therefore, ${}^{2}\mathsf{t}^{x_2} = {}^{2}t_3^{x_2}$ must hold. Let us continue with some other contact choices in the c-neighbourhood of $\mathcal{W}_2$. In light of Remark 27.4, we might want to stick to the 'default' contact choice for ${}^{2}\boldsymbol{q}^{x_2-1}$, and choose ${}^{2}\mathsf{f}^{x_2-1} = {}^{2}f_1^{x_2-1}$. Then, as ${}^{2}\mathsf{t}^{x_2-1} \prec_{{}^{2}\boldsymbol{q}^{x_2-1}} {}^{2}\mathsf{f}^{x_2-1}$ by (22), we must choose ${}^{2}\mathsf{t}^{x_2-1} = {}^{2}t_1^{x_2-1}$. However, in this case (23) fails, and there *is* a $h\colon \boldsymbol{q} \to \mathcal{I}_{\mathfrak{a}}$ homomorphism, as shown in Example 27.2. In fact, by repeating the above argument, we obtain that we must choose ${}^{2}\mathsf{f}^{x_2-k} = {}^{2}f_2^{x_2-k}$ and ${}^{2}\mathsf{t}^{x_2-k} = {}^{2}t_3^{x_2-k}$, for *every* $k \leq |\boldsymbol{q}|$.

In Lemma 27.10 below, we describe a general algorithmic solution to the constraint system along the lines of (H1)–(H4), and show that for this solution the converse of Lemma 27.8 holds. In order to formulate our solution, we need to fix some notation for c-neighbourhoods. With a slight abuse of notation in light of (H4), for any given clause c in ψ, we denote by $\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3$ the three n-cogwheels the node v^c_z of ${}^c\boldsymbol{q}$ is glued to. For each $z = 1, 2, 3$, $\mathcal{W}_z$ is built up from the $\boldsymbol{q}$-copies ${}^z\boldsymbol{q}^1, \dots, {}^z\boldsymbol{q}^n$, and the c-connection of $\mathcal{W}_z$ is obtained by glueing together node v^c_z of ${}^c\boldsymbol{q}$ with the contact ${}^z\mathsf{f}^{x_z} = {}^z\mathsf{t}^{x_z+1}$ of $\mathcal{W}_z$ (throughout, as before, $\pm$ is modulo n).



For any node x in $\boldsymbol{q}$, we denote by ${}^cx$ the copy of x in ${}^c\boldsymbol{q}$; and for $i = 1, \dots, n$ and $z = 1, 2, 3$, we denote by ${}^zx^i$ the copy of x in ${}^z\boldsymbol{q}^i$. Recall that for any k, we let t_k (f_k) denote the kth T-node (F-node) in $\boldsymbol{q}$. In particular, t_{last-1} denotes the last but one T-node in $\boldsymbol{q}$, and t_{last} the last T-node. We again assume that $t_1 \prec f_1$ (cf. (21)), and let $t_\Box$ denote the last T-node preceding f_1.

Lemma 27.10. *Given a 3CNF ψ, let $\mathcal{A}_{\boldsymbol{q},\psi}$ be a (ψ, n)-gadget, for some $n \geq (n_\psi + 2)(2|\boldsymbol{q}| + 1)$, built up from n-bikes, each satisfying the conditions of Lemma 27.7. Suppose $\mathcal{A}_{\boldsymbol{q},\psi}$ is such that the following hold for its special triple*:

$$\mathsf{v}_1 = t_1, \qquad \mathsf{v}_2 = \begin{cases} f_1, & \text{if } f_1 \prec t_{last}, \\ t_{last}, & \text{if } t_{last} \prec f_1, \end{cases} \qquad \mathsf{v}_3 = f_2;$$

and for every clause c in ψ, the following hold for the c-neighbourhood in $\mathcal{W}_1$:

$${}^1\mathsf{t}^{x_1} = {}^1t^{x_1}_\Box, \qquad {}^1\mathsf{f}^{x_1} = \begin{cases} {}^1f_1^{x_1}, & \text{if } f_1 \prec t_{last}, \\ {}^1f_2^{x_1}, & \text{if } t_{last} \prec f_1, \end{cases}$$

$${}^1\mathsf{t}^k = {}^1t_1^k, \qquad {}^1\mathsf{f}^k = {}^1f_1^k, \quad \text{for any other } k \text{ with } x_1 - |\boldsymbol{q}| \leq k \leq x_1 + |\boldsymbol{q}|;$$

the following hold for the c-neighbourhood in $\mathcal{W}_2$, for $k \leq |\boldsymbol{q}|$ and $1 \leq \ell \leq |\boldsymbol{q}|$:

$${}^2\mathsf{t}^{x_2-k} = \begin{cases} {}^2t_\diamond^{x_2-k}, & \text{if } f_1 \prec t_{last} \text{ and there is a } T\text{-node } t_\diamond \text{ with } t_\diamond \prec f_2 \text{ and } \delta(t_\diamond, f_2) = \delta(t_1, f_1), \\ {}^2t_1^{x_2-k}, & \text{otherwise}, \end{cases}$$

$${}^2\mathsf{f}^{x_2-k} = \begin{cases} {}^2f_2^{x_2-k}, & \text{if } f_1 \prec t_{last}, \\ {}^2f_2^{x_2}, & \text{if } k = 0,\ t_{last} \prec f_1 \text{ and } \delta(t_{last-1}, t_{last}) = \delta(t_{last}, f_1), \\ {}^2f_1^{x_2-k}, & \text{otherwise}, \end{cases}$$

$${}^2\mathsf{t}^{x_2+\ell} = \begin{cases} {}^2t_\Box^{x_2+\ell}, & \text{if } f_1 \prec t_{last}, \\ {}^2t_1^{x_2+\ell} & \text{if } t_{last} \prec f_1, \end{cases}$$

$${}^2\mathsf{f}^{x_2+\ell} = {}^2f_1^{x_2+\ell};$$

the following hold for the c-neighbourhood in $\mathcal{W}_3$:

$${}^3\mathsf{t}^{x_3} = {}^3t_1^{x_3}, \qquad {}^3\mathsf{f}^{x_3} = {}^3f_2^{x_3},$$

$${}^3\mathsf{t}^{x_3-k} = {}^3t_1^{x_3-k}, \qquad {}^3\mathsf{f}^{x_3-k} = {}^3f_1^{x_3-k}, \quad \text{for } 0 < k \leq |\boldsymbol{q}|,$$

$${}^3\mathsf{t}^{x_3+\ell} = \begin{cases} {}^3t_1^{x_3+\ell}, & \text{if } t_{last} \prec f_1 \text{ and } \delta(f_1, f_2) < \delta(t_1, f_1), \\ {}^3t_\Box^{x_3+\ell}, & \text{otherwise}, \end{cases}$$

$${}^3\mathsf{f}^{x_3+\ell} = \begin{cases} {}^3f_2^{x_3+\ell}, & \text{if } t_{last} \prec f_1 \text{ and } \delta(f_1, f_2) \geq \delta(t_1, f_1), \\ {}^3f_1^{x_3+\ell}, & \text{otherwise}, \end{cases} \qquad \text{for } 1 \leq \ell \leq |\boldsymbol{q}|.$$

Then $\mathcal{I}_\mathfrak{a} \not\models \boldsymbol{q}$, for any assignment $\mathfrak{a}$ satisfying ψ.

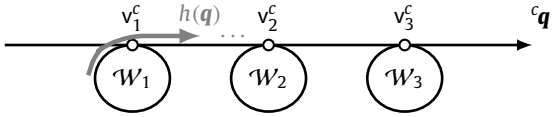
It is straightforward to check that (ψ, n)-gadgets $\mathcal{A}_{\boldsymbol{q},\psi}$ satisfying the conditions of the lemma always exist: As ψ has n_ψ-many clauses and $n \geq (n_\psi + 2)(2|\boldsymbol{q}|+1)$, for different clauses c, c', the c- and c'-neighbourhoods of the same n-cogwheel $\mathcal{W}$ can be kept disjoint from each other and from the F- and T-neighbourhoods of $\mathcal{W}$. Thus, choices for the present lemma do not interfere with the choices for Lemma 27.7. Also, by choosing (the corresponding copies of) t_1 and f_2 as contacts outside the F-, T- and c-neighbourhoods, conditions (22), (23) hold for all cogwheels in $\mathcal{A}_{\boldsymbol{q},\psi}$.

Proof. Suppose $\mathfrak{a}$ is an assignment satisfying ψ, and take the model of cov_A and $\mathcal{A}_{\boldsymbol{q},\psi}$ defined in (33). In light of (H4), we do not use any specifics about the clause c, and so we do not have explicit information about the particular labelings of the c-connections v_1^c, v_2^c and v_3^c in $\mathcal{I}_\mathfrak{a}$. However, (33) still implies that each of the attached cogwheels $\mathcal{W}_1$, $\mathcal{W}_2$ and $\mathcal{W}_3$ 'represents' a truth-value:

$$\text{for each } z = 1, 2, 3, \text{ the contacts of } \mathcal{W}_z \text{ are either all in } T^{\mathcal{I}_\mathfrak{a}} \text{ or all in } F^{\mathcal{I}_\mathfrak{a}}. \tag{35}$$

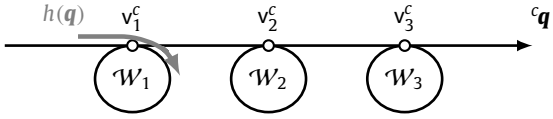
Now the proof of Lemma 27.10 is via excluding all possible locations in $\mathcal{A}_{\boldsymbol{q},\psi}$ for the image $h(\boldsymbol{q})$ of a potential sub-homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}_\mathfrak{a}$. As explained above, by (34), Lemmas 27.3 and 27.7, only the cases $(1)^c$–$(6)^c$ in Fig. 9 remain for the location of $h(\boldsymbol{q})$, and we need to show that none of them is possible. In light of (H2), we always track the location of $h(f_1)$ and, whenever possible, try to reduce the cases to cases in the proof of Lemma 27.7 for bikes:

$(1)^c$ $h(\boldsymbol{q})$ starts in $\mathcal{W}_1$ and $h(\mathsf{v}_1) \prec_{h(\boldsymbol{q})} \mathsf{v}_1^c$.



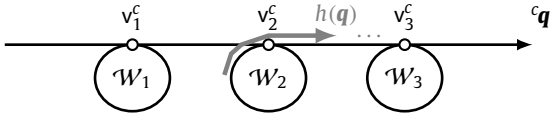
Then $h(\boldsymbol{q})$ definitely properly intersects the c-neighbourhood of $\mathcal{W}_1$, and it might also properly intersect the c-neighbourhoods of $\mathcal{W}_2$ or $\mathcal{W}_3$. It follows from $h(\mathsf{v}_1) \prec_{h(\boldsymbol{q})} \mathsf{v}_1^c$ that if $h(f_1)$ is in ${}^c\boldsymbol{q}$ then $h(f_1) \prec_{{}^c\boldsymbol{q}} {}^c f_1$. We can exclude all possible locations for $h(f_1)$ by the same argument as in case $(1)^T$ in the proof of Lemma 27.7, with the c-neighbourhood of $\mathcal{W}_1$ in place of the T-neighbourhood of $\mathcal{W}_\bullet$.

$(2)^c$ $h(\boldsymbol{q})$ starts in ${}^c\boldsymbol{q}$ and ends in $\mathcal{W}_1$.



Then $h(\boldsymbol{q})$ properly intersects the c-neighbourhood of $\mathcal{W}_1$ only. We can exclude all possible locations for $h(f_1)$ by the same argument as in case $(2)^T$ in the proof of Lemma 27.7, with the c-neighbourhood of $\mathcal{W}_1$ in place of the T-neighbourhood of $\mathcal{W}_\bullet$.

$(3)^c$ $h(\boldsymbol{q})$ starts in $\mathcal{W}_2$ and $h(\mathsf{v}_2) \preceq_{h(\boldsymbol{q})} \mathsf{v}_2^c$.



Then $h(\boldsymbol{q})$ definitely properly intersects the c-neighbourhood of $\mathcal{W}_2$, and it may also properly intersect the c-neighbourhood of $\mathcal{W}_3$. We consider the two cases $f_1 \prec t_{last}$ and $t_{last} \prec f_1$:

If $f_1 \prec t_{last}$ then $\mathsf{v}_2 = f_1$, and so $h(f_1) \preceq_{h(\boldsymbol{q})} {}^c f_1 = \mathsf{v}_2^c$. As ${}^2\mathsf{f}^{x_2-k} = {}^2f_2^{x_2-k}$ for all $k \leq |\boldsymbol{q}|$, $h(f_1) = {}^2f_1^{x_2-k}$ cannot hold for any such k, otherwise both $h(t)$ and $h(f_2)$ are contacts of $\mathcal{W}_2$ for the T-node t with ${}^2\mathsf{t}^{x_2-k} = {}^2t^{x_2-k}$, contradicting (35). Since the only F-node preceding f_2 in $\boldsymbol{q}$ is f_1, the only remaining option for $h(f_1)$ is when $h(f_1) = {}^2f_2^{x_2-k}$ is a contact of $\mathcal{W}_2$ for some $k \leq |\boldsymbol{q}|$. Now we track the location of $h(t_1)$. We have

$$\delta_{h(\boldsymbol{q})}\big(h(t_1), {}^2f_2^{x_2-k}\big) = \delta_{h(\boldsymbol{q})}\big(h(t_1), h(f_1)\big) = \delta(t_1, f_1) = \delta(y, f_2) = \delta_{{}^2\boldsymbol{q}^{x_2-k}}\big({}^2y^{x_2-k}, {}^2f_2^{x_2-k}\big), \tag{36}$$

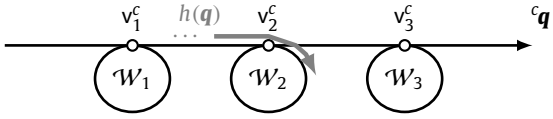
where y is the node in $\boldsymbol{q}$ with $y \prec f_2$ and $\delta(y, f_2) = \delta(t_1, f_1)$. Consider two cases, depending on whether y is a T-node or not:

- If y is a T-node $t_\diamond$, then ${}^2\mathsf{t}^{x_2-k} = {}^2t_\diamond^{x_2-k}$, and so $h(t_1) = {}^2\mathsf{t}^{x_2-k}$ by (36). Thus, $h(t_1)$ is a contact, contradicting (35) and the fact that $h(f_1)$ is also a contact of $\mathcal{W}_2$.
- If y is not a T-node $t_\diamond$ then ${}^2\mathsf{t}^{x_2-k} = {}^2t_1^{x_2-k}$. While $y \preceq f_1$ and $f_1 \prec y$ are both possible, we surely have $t_1 \prec y$, as $\delta(y, f_2) = \delta(t_1, f_1) < \delta(t_1, f_2)$. Then $h(t_1) = {}^2y^{x_2-k}$ follows by (36). But ${}^2y^{x_2-k}$ is not a T-node.

If $t_{last} \prec f_1$ then $\mathsf{v}_2 = t_{last}$. If follows from $h(\mathsf{v}_2) \preceq_{h(\boldsymbol{q})} \mathsf{v}_2^c$ that if $h(f_1)$ is in ${}^c\boldsymbol{q}$ then $h(f_1) \preceq_{{}^c\boldsymbol{q}} {}^c f_1$. As there is no F-node preceding ${}^c f_1$ in ${}^c\boldsymbol{q}$, either $h(f_1)$ is in $\mathcal{W}_2$, or $h(f_1) = {}^c f_1$. We can exclude all possible locations for $h(f_1)$ by

the same argument as in case $(3)^T$ in the proof of Lemma 27.7, with the c-neighbourhood of $\mathcal{W}_2$ in place of the T-neighbourhood of $\mathcal{W}_\circ$.

$(4)^c$ $h(\boldsymbol{q})$ ends in $\mathcal{W}_2$ and $\mathsf{v}_2^c \preceq_{h(\boldsymbol{q})} h(\mathsf{v}_2)$.

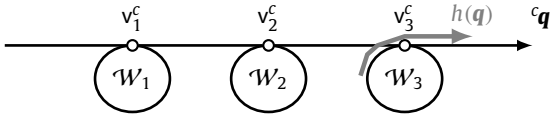


Then $h(\boldsymbol{q})$ definitely properly intersects the c-neighbourhood of $\mathcal{W}_2$, and it might also properly intersect the c-neighbourhood of $\mathcal{W}_1$. We consider the two cases $f_1 \prec t_{last}$ and $t_{last} \prec f_1$:

If $f_1 \prec t_{last}$ then $\mathsf{v}_2 = f_1$, and so $\mathsf{v}_2^c = {}^c f_1 \preceq_{h(\boldsymbol{q})} h(f_1)$ and $h(f_1)$ is in $\mathcal{W}_2$. We can exclude all possible locations for $h(f_1)$ by the same argument as in case $(2)^F$ in the proof of Lemma 27.7, with the c-neighbourhood of $\mathcal{W}_2$ in place of the F-neighbourhood of $\mathcal{W}_\bullet$.

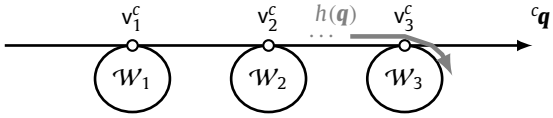
If $t_{last} \prec f_1$ then $\mathsf{v}_2 = t_{last}$. As by our assumption $h(\boldsymbol{q})$ ends in $\mathcal{W}_2$ and $\mathsf{v}_2^c \preceq_{h(\boldsymbol{q})} h(\mathsf{v}_2)$, if follows that $\mathsf{v}_2^c \preceq_{h(\boldsymbol{q})} h(t_{last}) \prec_{h(\boldsymbol{q})} h(f_1)$. We can exclude all possible locations for $h(f_1)$ by the same argument as in case $(4)^T$ in the proof of Lemma 27.7, with the c-neighbourhood of $\mathcal{W}_2$ in place of the T-neighbourhood of $\mathcal{W}_\circ$.

$(5)^c$ $h(\boldsymbol{q})$ starts in $\mathcal{W}_3$.



Then $h(\boldsymbol{q})$ properly intersects the c-neighbourhood of $\mathcal{W}_3$ only. We have $h(\mathsf{v}_3) \preceq_{h(\boldsymbol{q})} \mathsf{v}_3^c$, as otherwise there is no room for $h(\boldsymbol{q})$ in ${}^c\boldsymbol{q}$. As $\mathsf{v}_3 = f_2$, we have $h(f_1) \prec_{h(\boldsymbol{q})} h(f_2) \preceq_{h(\boldsymbol{q})} \mathsf{v}_3^c$ and $h(f_1)$ is in $\mathcal{W}_3$. We can exclude all possible locations for $h(f_1)$ by the same argument as in case $(1)^F$ in the proof of Lemma 27.7, with the c-neighbourhood of $\mathcal{W}_3$ in place of the F-neighbourhood of $\mathcal{W}_\bullet$.

$(6)^c$ $h(\boldsymbol{q})$ ends in $\mathcal{W}_3$ and $\mathsf{v}_3^c \prec_{h(\boldsymbol{q})} h(\mathsf{v}_3)$.



Then $h(\boldsymbol{q})$ definitely properly intersects the c-neighbourhood of $\mathcal{W}_3$, and it may also properly intersect the c-neighbourhood of $\mathcal{W}_1$ or $\mathcal{W}_2$. As $\mathsf{v}_3 = f_2$, we have ${}^c f_2 = \mathsf{v}_3^c \prec_{h(\boldsymbol{q})} h(f_2)$. Therefore, if $h(f_1)$ is in ${}^c\boldsymbol{q}$ then ${}^c f_1 \prec_{{}^c\boldsymbol{q}} h(f_1)$. As $\mathsf{v}_2^c \preceq_{{}^c\boldsymbol{q}} {}^c f_1$ and there is no F-node between ${}^c f_1$ and ${}^c f_2$ in ${}^c\boldsymbol{q}$, it follows that $\mathsf{v}_3^c \preceq_{h(\boldsymbol{q})} h(f_1)$ and $h(f_1)$ is in $\mathcal{W}_3$. We can exclude all possible locations for $h(f_1)$ by the same argument as in case $(2)^F$ in the proof of Lemma 27.7, with the c-neighbourhood of $\mathcal{W}_3$ in place of the F-neighbourhood of $\mathcal{W}_\bullet$.

We excluded all possible locations in $\mathcal{A}_{\boldsymbol{q},\psi}$ for the image $h(\boldsymbol{q})$ of a potential subhomomorphism $h\colon \boldsymbol{q} \to \mathcal{I}_\mathfrak{a}$, which completes the proof of Lemma 27.10. □

To complete the proof of Theorem 27, given a 3CNF ψ with n_ψ clauses, we set $n = (n_\psi + 2)(2|\boldsymbol{q}| + 1)$ and take some (ψ, n)-gadget $\mathcal{A}_{\boldsymbol{q},\psi}$ satisfying the conditions of Lemma 27.10. By Lemmas 27.8 and 27.10, we then obtain: $\mathsf{cov}_A, \mathcal{A}_{\boldsymbol{q},\psi} \not\models \boldsymbol{q}$ iff ψ is satisfiable.

7. Conclusion

This article contributes to the non-uniform approach to ontology-based data access, which—broadly conceived—also includes optimisation of datalog and disjunctive datalog programs. There are three distinctive directions of research in this area (for detailed references, see Section 1.3):

- **(I)** Finding general automata-theoretic, model-theoretic or algebraic characterisations of OMQs with a given data complexity or rewritability type and investigating the computational complexity of checking those characterisations. As it turned out, for many standard DL ontology languages and monadic (disjunctive) datalog programs, the complexity of deciding FO- and datalog-rewritability ranges between ExpTime and 3ExpTime.
- **(II)** Designing practical (possibly incomplete) rewriting and approximation algorithms. For example, the algorithm from [28] either successfully rewrites a given disjunctive datalog program into an equivalent plain datalog program or fails to decide whether the input is datalog rewritable or not.
- **(III)** Obtaining explicit classifications of 'natural' restricted families of OMQs such as, for instance, binary chain datalog sirups [47]. Apart from supplementing **(II)**, results in this direction help pinpoint key sources of the high complexity in **(I)** and thereby identify interesting and better behaved classes of OMQs, as well as develop fine methods of establishing data complexity bounds for OMQ answering.

This article contributes to directions **(I)** and **(III)**. We introduce two classes of rudimentary OMQs, called d- and dd-sirups, and show that they capture many difficulties of both general OMQs with a disjunctive DL ontology and general monadic (plain and disjunctive) datalog queries. Indeed, the syntactically very simple and seemingly inexpressive d-sirups reveal rather complex and unexpected behaviour: (*i*) answering them is Π_2^p-complete for combined complexity and requires finding exponential-size resolution proofs in general; (*ii*) deciding their FO-rewritability turns out to be 2ExpTime-hard [44]—as hard as deciding FO-rewritability of arbitrary monadic datalog queries—with (*iii*) nonrecursive datalog, positive existential, and UCQ rewritings being of at least single-, double- and triple-exponential size in the worst case, respectively. Thus, understanding the behaviour of d-sirups is challenging yet fundamental for developing OBDA with expressive ontologies (note that d-sirups also constitute a new interesting class of CSPs).

The proofs of the 'negative' results mentioned above point to two 'culprits': possibly intersecting classes F and T in the covering axiom $F(x) \lor T(x) \leftarrow A(x)$, and multiple binary relations between the same pair of variables in a query. We demonstrate that elimination of these culprits can lead to non-trivial OMQ classes that admit complete explicit classifications, though may need the development of new methods and quite tricky, laborious proofs. Our main achievement here is an explicit AC^0 / NL/ P/ coNP-tetrachotomy of path-shaped dd-sirups (with disjoint F and T), which required new techniques for establishing membership in NL and for proving P- and especially coNP-hardness. (Incidentally, the bike technique for proving coNP-hardness shows that the algorithm from [28] mentioned in **(II)** is complete for path-shaped dd-sirups.) We believe that these techniques can also be used for wider classes of OMQs, which is witnessed by the AC^0 / L / NL-hardness trichotomy of ditree-shaped dd-sirups in [44].

7.1. Next steps

Interesting and challenging problems arising from our research are abundant; here are some of them.

1. Find complete explicit classifications of the following families of OMQs: (*i*) d-sirups with path CQs (that may contain FT-twins), (*ii*) undirected path-shaped, (*iii*) ditree- and (*iv*) undirected tree-shaped dd- and d-sirups. Also, consider (d)d-sirups $(\mathsf{cov}_\top, \boldsymbol{q})$ and $(\mathsf{cov}_\top^\bot, \boldsymbol{q})$ with total covering $\forall x\,(F(x) \lor T(x))$.
2. Settle the tight complexity of deciding FO- and other types of rewritability for arbitrary (*i*) d-sirups and (*ii*) dd-sirups. We conjecture that (*i*) is harder than (*ii*) in general.
3. Identify the complexity of deciding FO- and other types of rewritability to ontologies in (*i*) Schema.org and (*ii*) *DL-Lite*$_{krom}$ and *DL-Lite*$_{bool}$ [67]. Ontologies in (*i*) allow multiple disjunctions (and so covering by any number of classes); those in (*ii*) allow restricted existential quantification on the right-hand side of implications.
4. Analyse the size of FO-rewritings for OMQs with disjunctive axioms (starting with d- and dd-sirups). Could FO-rewritings be substantially more succinct than NDL- and PE-rewritings (cf. [96, Theorem 6.1])? (Note that the succinctness problem for OMQ rewritings is closely related to circuit complexity [95,50].)
5. Consider the **(data complexity)** and **(rewritability)** problems for d- and dd-sirups with multiple answer variables (which could lead to simpler classifications as indicated by [51]).
6. Investigate interconnections between (d)d-sirups and CSPs (starting from those in [30,36]) with the aim of transferring results from one formalism to the other.
7. Using the techniques developed in this article for establishing lower data complexity bounds, identify classes of OMQs for which rewriting algorithms such as the ones in [38,28] are complete.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The work of O. Gerasimova was funded by RFBR, project number 20-31-90123. The work of V. Podolskii was supported by the HSE University Basic Research Program. The work of M. Zakharyaschev was supported by the EPSRC U.K. grant EP/S032282. We are grateful to Frank Wolter for his remarks that helped us improve the article. Thanks are also due to the anonymous referees for their careful reading, valuable comments and constructive suggestions.

References

[1] A. Schaerf, On the complexity of the instance checking problem in concept languages with existential quantification, J. Intell. Inf. Syst. 2 (1993) 265–278.
[2] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider (Eds.), The Description Logic Handbook, 2 ed., Cambridge University Press, 2007.
[3] F. Baader, I. Horrocks, C. Lutz, U. Sattler, An Introduction to Description Logic, Cambridge University Press, 2017.
[4] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, J. Data Semant. X (2008) 133–173.
[5] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: the DL-Lite family, J. Autom. Reason. 39 (2007) 385–429.

[6] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: a survey, in: J. Lang (Ed.), Proc. IJCAI 2018, 2018, pp. 5511–5519, ijcai.org.
[7] G. Xiao, L. Ding, B. Cogrel, D. Calvanese, Virtual knowledge graphs: an overview of systems and use cases, Data Intell. 1 (2019) 201–223.
[8] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
[9] N. Immerman, Descriptive Complexity, Springer, 1999.
[10] C. Civili, R. Rosati, A broad class of first-order rewritable tuple-generating dependencies, in: Proc. of the 2nd Int. Datalog 2.0 Workshop, in: Lecture Notes in Computer Science, vol. 7494, Springer, 2012, pp. 68–80.
[11] G. Gottlob, G. Orsi, A. Pieris, Query rewriting and optimization for ontological databases, ACM Trans. Database Syst. 39 (2014) 25:1–25:46.
[12] J. Baget, M. Leclère, M. Mugnier, E. Salvat, On rules with existential variables: walking the decidability line, Artif. Intell. 175 (2011) 1620–1654.
[13] M. König, M. Leclère, M. Mugnier, M. Thomazo, Sound, complete and minimal UCQ-rewriting for existential rules, Semant. Web 6 (2015) 451–475.
[14] U. Hustadt, B. Motik, U. Sattler, Data complexity of reasoning in very expressive description logics, in: L.P. Kaelbling, A. Saffiotti (Eds.), Proc. IJCAI 2005, Professional Book Center, 2005, pp. 466–471.
[15] R. Rosati, On conjunctive query answering in EL, in: D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A. Turhan, S. Tessaris (Eds.), Proc. DL 2007, in: CEUR Workshop Proceedings, vol. 250, 2007, CEUR-WS.org.
[16] H. Pérez-Urbina, B. Motik, I. Horrocks, Tractable query answering and rewriting under description logic constraints, J. Appl. Log. 8 (2010) 186–209.
[17] T. Eiter, M. Ortiz, M. Šimkus, T. Tran, G. Xiao, Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules, in: J. Hoffmann, B. Selman (Eds.), Proc. AAAI 2012, AAAI Press, 2012.
[18] D. Gabbay, A. Kurucz, F. Wolter, M. Zakharyaschev, Many-Dimensional Modal Logics: Theory and Applications, Studies in Logic and the Foundations of Mathematics, vol. 148, Elsevier, 2003.
[19] B. Motik, Reasoning in description logics using resolution and deductive databases, Ph.D. thesis, Karlsruhe Institute of Technology, Germany, 2006.
[20] U. Hustadt, B. Motik, U. Sattler, Reasoning in description logics by a reduction to disjunctive datalog, J. Autom. Reason. 39 (2007) 351–384.
[21] B. Cuenca Grau, B. Motik, G. Stoilos, I. Horrocks, Computing datalog rewritings beyond Horn ontologies, in: F. Rossi (Ed.), Proc. IJCAI 2013, IJCAI/AAAI, 2013, pp. 832–838.
[22] D. Hovland, R. Kontchakov, M.G. Skjæveland, A. Waaler, M. Zakharyaschev, Ontology-based data access to Slegge, in: C. d'Amato, M. Fernández, V.A.M. Tamma, F. Lécué, P. Cudré-Mauroux, J.F. Sequeda, C. Lange, J. Heflin (Eds.), Proc. ISWC 2017, Part II, in: Lecture Notes in Computer Science, vol. 10588, Springer, 2017, pp. 120–129.
[23] D. Carral, C. Feier, B. Cuenca Grau, P. Hitzler, I. Horrocks, EL-ifying ontologies, in: S. Demri, D. Kapur, C. Weidenbach (Eds.), Proc. IJCAR 2014, in: Lecture Notes in Computer Science, vol. 8562, Springer, 2014, pp. 464–479.
[24] Y. Zhou, B. Cuenca Grau, Y. Nenov, M. Kaminski, I. Horrocks, PAGOdA: pay-as-you-go ontology query answering using a datalog reasoner, J. Artif. Intell. Res. 54 (2015) 309–367.
[25] E. Botoeva, D. Calvanese, V. Santarelli, D.F. Savo, A. Solimando, G. Xiao, Beyond OWL 2 QL in OBDA: rewritings and approximations, in: D. Schuurmans, M.P. Wellman (Eds.), Proc. AAAI 2016, AAAI Press, 2016, pp. 921–928.
[26] A. Bötcher, C. Lutz, F. Wolter, Ontology approximation in Horn description logics, in: S. Kraus (Ed.), Proc. IJCAI 2019, 2019, pp. 1574–1580, ijcai.org.
[27] E. Kharlamov, D. Hovland, M.G. Skjæveland, D. Bilidas, E. Jiménez-Ruiz, G. Xiao, A. Soylu, D. Lanti, M. Rezk, D. Zheleznyakov, M. Giese, H. Lie, Y.E. Ioannidis, Y. Kotidis, M. Koubarakis, A. Waaler, Ontology based data access in Statoil, J. Web Semant. 44 (2017) 3–36.
[28] M. Kaminski, Y. Nenov, B. Cuenca Grau, Datalog rewritability of disjunctive datalog programs and non-Horn ontologies, Artif. Intell. 236 (2016) 90–118.
[29] C. Lutz, F. Wolter, Non-uniform data complexity of query answering in description logics, in: G. Brewka, T. Eiter, S.A. McIlraith (Eds.), Proc. KR 2012, AAAI Press, 2012.
[30] M. Bienvenu, B. ten Cate, C. Lutz, F. Wolter, Ontology-based data access: a study through disjunctive datalog, CSP, and MMSNP, ACM Trans. Database Syst. 39 (33) (2014) 1–44.
[31] T. Feder, M.Y. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction: a study through datalog and group theory, SIAM J. Comput. 28 (1998) 57–104.
[32] A.A. Bulatov, A dichotomy theorem for nonuniform CSPs, in: C. Umans (Ed.), Proc. FOCS 2017, IEEE Computer Society, 2017, pp. 319–330.
[33] D. Zhuk, A proof of CSP dichotomy conjecture, in: C. Umans (Ed.), Proc. FOCS 2017, IEEE Computer Society, 2017, pp. 331–342.
[34] C. Lutz, L. Sabellek, Ontology-mediated querying with the description logic EL: trichotomy and linear datalog rewritability, in: C. Sierra (Ed.), Proc. IJCAI 2017, 2017, pp. 1181–1187, ijcai.org.
[35] C. Lutz, L. Sabellek, A complete classification of the complexity and rewritability of ontology-mediated queries based on the description logic EL, CoRR, arXiv:1904.12533 [abs], 2019.
[36] C. Feier, A. Kuusisto, C. Lutz, Rewritability in monadic disjunctive datalog, MMSNP, and expressive description logics, Log. Methods Comput. Sci. 15 (2019).
[37] P. Hansen, C. Lutz, I. Seylan, F. Wolter, Efficient query rewriting in the description logic EL and beyond, in: Proc. IJCAI 2015, AAAI, 2015, pp. 3034–3040.
[38] M. Kaminski, B. Cuenca Grau, Sufficient conditions for first-order and datalog rewritability in ELU, in: T. Eiter, B. Glimm, Y. Kazakov, M. Krötzsch (Eds.), Proc. DL, in: CEUR Workshop Proceedings, vol. 1014, 2013, pp. 271–293, CEUR-WS.org.
[39] S. Arora, B. Barak, Computational Complexity: A Modern Approach, 1st ed., Cambridge University Press, 2009.
[40] S.S. Cosmadakis, P.C. Kanellakis, Parallel evaluation of recursive rule queries, in: A. Silberschatz (Ed.), Proc. PODS 1986, ACM, 1986, pp. 280–293.
[41] M.Y. Vardi, Decidability and undecidability results for boundedness of linear recursive queries, in: C. Edmondson-Yurkanan, M. Yannakakis (Eds.), Proc. PODS 1988, ACM, 1988, pp. 341–351.
[42] G. Gottlob, C.H. Papadimitriou, On the complexity of single-rule datalog queries, Inf. Comput. 183 (2003) 104–122.
[43] P.C. Kanellakis, Elements of relational database theory, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, Elsevier and MIT Press, 1990, pp. 1073–1156.
[44] S. Kikot, A. Kurucz, V. Podolskii, M. Zakharyaschev, Deciding boundedness of monadic sirups, in: Proc. PODS 2021, ACM Press, 2021.
[45] S.S. Cosmadakis, H. Gaifman, P.C. Kanellakis, M.Y. Vardi, Decidable optimization problems for database logic programs, in: Proc. STOC 1988, 1988, pp. 477–490.
[46] M. Benedikt, B. ten Cate, T. Colcombet, M. Vanden Boom, The complexity of boundedness for guarded logics, in: Proc. LICS 2015, 2015, pp. 293–304.
[47] F.N. Afrati, C.H. Papadimitriou, The parallel complexity of simple logic programs, J. ACM 40 (1993) 891–916.
[48] S.S. Dantchev, S. Riis, "Planar" tautologies hard for resolution, in: Proc. FOCS 2001, IEEE Computer Society, 2001, pp. 220–229.
[49] M. Alekhnovich, Mutilated chessboard problem is exponentially hard for resolution, Theor. Comput. Sci. 310 (2004) 513–525.
[50] M. Bienvenu, S. Kikot, R. Kontchakov, V. Podolskii, M. Zakharyaschev, Ontology-mediated queries: combined complexity and succinctness of rewritings via circuit complexity, J. ACM 65 (2018) 28:1–28:51.
[51] A. Hernich, C. Lutz, A. Ozaki, F. Wolter, Schema.org as a description logic, in: Q. Yang, M.J. Wooldridge (Eds.), Proc. IJCAI 2015, AAAI Press, 2015, pp. 3048–3054.
[52] J.D. Ullman, Principles of Database and Knowledge-Base Systems, Volume II, Computer Science Press, 1989.
[53] O. Gerasimova, S. Kikot, A. Kurucz, V. Podolskii, M. Zakharyaschev, A data complexity and rewritability tetrachotomy of ontology-mediated queries with a covering axiom, in: D. Calvanese, E. Erdem, M. Thielscher (Eds.), Proc. KR 2020, 2020, pp. 403–413.

[54] J.F. Naughton, Data independent recursion in deductive databases, in: A. Silberschatz (Ed.), Proc. PODS 1986, ACM, 1986, pp. 267–279.
[55] J.D. Ullman, A.V. Gelder, Parallel complexity of logical query programs, Algorithmica 3 (1988) 5–42.
[56] J.F. Naughton, Minimizing function-free recursive inference rules, J. ACM 36 (1989) 69–91.
[57] R. Ramakrishnan, Y. Sagiv, J.D. Ullman, M.Y. Vardi, Proof-tree transformation theorems and their applications, in: A. Silberschatz (Ed.), Proc. PODS 1989, ACM Press, 1989, pp. 172–181.
[58] Y.P. Saraiya, Linearizing nonlinear recursions in polynomial time, in: A. Silberschatz (Ed.), Proc. PODS 1989, ACM Press, 1989, pp. 182–189.
[59] K. Wang, Some positive results for boundedness of multiple recursive rules, in: G. Gottlob, M.Y. Vardi (Eds.), Proc. ICDT 1995, in: Lecture Notes in Computer Science, vol. 893, Springer, 1995, pp. 383–396.
[60] E. Dantsin, T. Eiter, G. Gottlob, A. Voronkov, Complexity and expressive power of logic programming, ACM Comput. Surv. 33 (2001) 374–425.
[61] Y.E. Ioannidis, A time bound on the materialization of some recursively defined views, in: A. Pirotte, Y. Vassiliou (Eds.), Proc. VLDB 1985, Morgan Kaufmann, 1985, pp. 219–226.
[62] R. van der Meyden, Predicate boundedness of linear monadic datalog is in PSPACE, Int. J. Found. Comput. Sci. 11 (2000) 591–612.
[63] J.F. Naughton, Y. Sagiv, A decidable class of bounded recursions, in: M.Y. Vardi (Ed.), Proc. PODS 1987, ACM, 1987, pp. 227–236.
[64] G.G. Hillebrand, P.C. Kanellakis, H.G. Mairson, M.Y. Vardi, Undecidable boundedness problems for datalog programs, J. Log. Program. 25 (1995) 163–190.
[65] J. Marcinkowski, Achilles, turtle, and undecidable boundedness problems for small DATALOG programs, SIAM J. Comput. 29 (1999) 231–257.
[66] H. Gaifman, H.G. Mairson, Y. Sagiv, M.Y. Vardi, Undecidable optimization problems for database logic programs, J. ACM 40 (1993) 683–713.
[67] A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyaschev, The DL-Lite family and relations, J. Artif. Intell. Res. 36 (2009) 1–69.
[68] F. Baader, S. Brandt, C. Lutz, Pushing the EL envelope, in: L.P. Kaelbling, A. Saffiotti (Eds.), Proc. IJCAI 2005, Professional Book Center, 2005, pp. 364–369.
[69] F. Baader, C. Lutz, B. Suntisrivaraporn, Efficient reasoning in EL+, in: B. Parsia, U. Sattler, D. Toman (Eds.), Proc. DL 2006, in: CEUR Workshop Proceedings, vol. 189, 2006, CEUR-WS.org.
[70] F. Baader, S. Brandt, C. Lutz, Pushing the EL envelope further, in: K. Clark, P.F. Patel-Schneider (Eds.), Proc. OWLED 2008 DC Workshop on OWL: Experiences and Directions, 2008.
[71] A. Calì, G. Gottlob, T. Lukasiewicz, A general datalog-based framework for tractable query answering over ontologies, J. Web Semant. 14 (2012) 57–83.
[72] A. Calì, G. Gottlob, A. Pieris, Towards more expressive ontology languages: the query answering problem, Artif. Intell. 193 (2012) 87–128.
[73] M. Kaminski, Y. Nenov, B. Cuenca Grau, Datalog rewritability of disjunctive datalog programs and its applications to ontology reasoning, in: C.E. Brodley, P. Stone (Eds.), Proc. AAAI 2014, AAAI Press, 2014, pp. 1077–1083.
[74] D. Trivela, G. Stoilos, A. Chortaras, G.B. Stamou, Optimising resolution-based rewriting algorithms for OWL ontologies, J. Web Semant. 33 (2015) 30–49.
[75] D. Trivela, G. Stoilos, A. Chortaras, G. Stamou, Resolution-based rewriting for horn-SHIQ ontologies, Knowl. Inf. Syst. 62 (2020) 107–143.
[76] M. Bienvenu, C. Lutz, F. Wolter, First-order rewritability of atomic queries in Horn description logics, in: F. Rossi (Ed.), Proc. IJCAI 2013, IJCAI/AAAI, 2013, pp. 754–760.
[77] M. Bienvenu, P. Hansen, C. Lutz, F. Wolter, First order-rewritability and containment of conjunctive queries in Horn description logics, in: S. Kambhampati (Ed.), Proc. IJCAI 2016, IJCAI/AAAI Press, 2016, pp. 965–971.
[78] P. Barceló, G. Berger, C. Lutz, A. Pieris, First-order rewritability of frontier-guarded ontology-mediated queries, in: J. Lang (Ed.), Proc. IJCAI 2018, 2018, pp. 1707–1713, ijcai.org.
[79] P. Bourhis, C. Lutz, Containment in monadic disjunctive datalog, MMSNP, and expressive description logics, in: C. Baral, J.P. Delgrande, F. Wolter (Eds.), Proc. KR 2016, AAAI Press, 2016, pp. 207–216.
[80] A. Hernich, C. Lutz, F. Papacchini, F. Wolter, Dichotomies in ontology-mediated querying with the guarded fragment, ACM Trans. Comput. Log. 21 (2020) 20:1–20:47.
[81] Y.P. Saraiya, Polynomial-time program transformations in deductive databases, in: D.J. Rosenkrantz, Y. Sagiv (Eds.), Proc. PODS 1990, ACM Press, 1990, pp. 132–144.
[82] W. Zhang, C.T. Yu, D. Troy, Necessary and sufficient conditions to linearize double recursive programs in logic databases, ACM Trans. Database Syst. 15 (1990) 459–482.
[83] F.N. Afrati, M. Gergatsoulis, F. Toni, Linearisability on datalog programs, Theor. Comput. Sci. 308 (2003) 199–226.
[84] T.J. Schaefer, The complexity of satisfiability problems, in: R.J. Lipton, W.A. Burkhard, W.J. Savitch, E.P. Friedman, A.V. Aho (Eds.), Proc. STOC 1978, ACM, 1978, pp. 216–226.
[85] A.A. Bulatov, P. Jeavons, A.A. Krokhin, Classifying the complexity of constraints using finite algebras, SIAM J. Comput. 34 (2005) 720–742.
[86] B. Larose, C. Loten, C. Tardif, A characterisation of first-order constraint satisfaction problems, Log. Methods Comput. Sci. 3 (2007).
[87] P. Hell, J. Nesetril, Colouring, constraint satisfaction, and complexity, Comput. Sci. Rev. 2 (2008) 143–163.
[88] H. Chen, B. Larose, Asking the metaquestions in constraint tractability, ACM Trans. Comput. Theory 9 (2017) 11:1–11:27.
[89] O. Gerasimova, S. Kikot, M. Zakharyaschev, Checking the data complexity of ontology-mediated queries: a case study with non-uniform CSPs and Polyanna, in: C. Lutz, U. Sattler, C. Tinelli, A. Turhan, F. Wolter (Eds.), Description Logic, Theory Combination, and All That, in: Lecture Notes in Computer Science, vol. 11560, Springer, 2019, pp. 329–351.
[90] R. Gault, P. Jeavons, Implementing a test for tractability, Constraints 9 (2004) 139–160.
[91] C.-L. Chang, R.C.-T. Lee, Symbolic Logic and Mechanical Theorem Proving, 1st ed., Academic Press, 1973.
[92] L. Egri, B. Larose, P. Tesson, Symmetric datalog and constraint satisfaction problems in logspace, in: Proc. LICS 2007, IEEE, 2007, pp. 193–202.
[93] M. Grohe, The complexity of homomorphism and constraint satisfaction problems seen from the other side, J. ACM 54 (2007) 1:1–1:24.
[94] L.J. Stockmeyer, The polynomial-time hierarchy, Theor. Comput. Sci. 3 (1976) 1–22.
[95] G. Gottlob, S. Kikot, R. Kontchakov, V. Podolskii, T. Schwentick, M. Zakharyaschev, The price of query rewriting in ontology-based data access, Artif. Intell. 213 (2014) 42–59.
[96] B. Rossman, Homomorphism preservation theorems, J. ACM 55 (2008) 15:1–15:53.
[97] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree automata techniques and applications, Available at http://www.grappa.univ-lille3.fr/tata, 2007.
[98] C. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.