Tractable Ontology-Mediated Query Answering with Datatypes

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by Julio C. Lazzarini Lemos

April 17, 2019

Summary

Adding datatypes to ontology-mediated queries (OMQs) often makes query answering hard, even for lightweight languages. As a consequence, the use of datatypes in ontologies, e.g. in OWL 2 QL, has been severely restricted. We propose a new, non-uniform, way of analyzing the data-complexity of OMQ answering with datatypes. Instead of restricting the ontology language we aim at a classification of the patterns of datatype atoms in OMQs into those that can occur in non-tractable OMQs and those that only occur in tractable OMQs. To this end we establish a close link between OMQ answering with datatypes and constraint satisfaction problems (CSPs) over the datatypes. Given that query answering in this setting is undecidable in general already for very simple datatypes, we introduce, borrowing from the database literature, a property of OMQs called the Bounded Match Depth Property (BMDP). We apply the link to CSPs– using results and techniques in universal algebra and model theory–to prove PTIME/co-NP dichotomies for OMQs with the BDMP over Horn-$\mathcal{ALCHI}$ extended with (1) all finite datatypes, (2) rational numbers with linear order and (3) certain families of datatypes over the integers with the successor relation.

Contents

1 Introduction

In recent years, querying data using ontologies has become one of the main applications of description logics (DLs). The general idea is that an ontology is used to enrich incomplete and heterogenous data with a semantics and with background knowledge and thereby serves as an interface for querying data that also allows the derivation of additional facts. In this area called ontology-based data management (OBDM) one of the main research problems is to identify ontology languages and queries for which query answering scales to large amounts of data [33, 13]. Since the size of the data is typically very large compared to the size of the ontology and the size of the query, the central measure for such scalability is provided by data complexity; that is, the complexity of the computational problem is measured where both the ontology and the query are fixed, and only the data varies.

As in most Computer Science applications, in standard ontology languages such as the Ontology Web Language 2 (OWL 2),[1] ontologies can refer to data values such as integers, rationals and strings. These classes of values are called *datatypes*,[2] and were introduced in DLs in 1991 (see [9]) and then investigated intensively [74]. The well-kown relevance of datatypes for applications is the main motivation of the present contribution.

In DL, ontologies take the form of a TBox, data is stored in an ABox, and the most important class of queries are (unions of) conjunctive queries, or simply (U)CQs. A basic observation regarding this setup is that even for DLs from the DL-Lite family that have been designed for tractable OBDM the addition of datatypes to the TBoxes or the UCQs easily leads to non-tractable query answering problems [5, 94]. As a consequence of this, the use of datatypes in TBoxes and query languages for OBDM has been severely restricted. For example, the OWL2 QL standard admits datatypes with unary predicates only. Nevertheless, in applications, there is clearly a need for expressive datatypes both in TBoxes and in queries. And, in particular, datatypes with predicates of higher arity.

The aim of this thesis is to revisit OBDM with expressive datatypes from a new, non-uniform perspective. Instead of the standard approach that aims at the definition of DLs $\mathcal{L}$ and query languages $\mathcal{Q}$ such that for any TBox $\mathcal{T}$ in $\mathcal{L}$ and any query q in $\mathcal{Q}$, answering q under $\mathcal{T}$ is tractable in data complexity, we aim at describing the complexity of query answering with datatypes at a more fine-grained level— by taking into account the way in which datatype atoms can occur in TBoxes and in queries. To this end, we establish a close link between the complexity of query answering and of constraint satisfaction problems (CSPs) over the datatype. This link enables us to transfer complexity results from the CSP world to the world of OBDM and leads, in some

[1] https://www.w3.org/TR/owl2-syntax/

[2] https://www.w3.org/TR/owl2-syntax/#Datatype_Maps

cases, to complete classifications of the complexity of query answering into PTIME and coNP-complete classes.

In more detail, we consider TBoxes in Horn DLs extended with concept inclusions that contain attribute restrictions qualified by unary datatype atoms on their right hand side and UCQs that contain datatype atoms of arbitrary arity. If $\mathcal{T}$ is such a TBox over datatype $\mathcal{D}$ and q such a UCQ over $\mathcal{D}$, then $Q = (\mathcal{T}, q)$ is called an ontology-mediated query (OMQ) over $\mathcal{D}$. We aim at understanding the complexity of query answering for this very broad class of OMQs. A first observation is that query answering becomes undecidable for many important datatypes $\mathcal{D}$ including $(\mathbb{Q}, <)$, $(\mathbb{Z}, <)$ and $(\mathbb{Z}, \leq)$. To restore decidability and enable a polynomial reduction to the complement of CSPs over $\mathcal{D}$ we utilise the bounded match depth property (BMDP), borrowed from the database literature, a property of OMQs that ensures that answers to OMQs can be determined based on a bounded subset of the standard chase of a knowledge base in a Horn DL. Many practical OMQs have the BMDP. For example, all OMQs with either TBoxes whose chase always terminates (which is often the case in practice) or with rooted UCQs whose variables are all connected via non-datatype variables to an answer variable (which covers a broad class of UCQs). If the datatype $\mathcal{D}$ is homogeneous (as is the case for $(\mathbb{Q}, <)$ and $(\mathbb{Q}, \leq)$), then the latter condition can be relaxed even further to certain Boolean UCQs. As the CSP of many important datatypes is in NP, it follows that query answering for OMQs with the BMDP over such datatypes is in coNP, a significant improvement compared to the undecidable OMQs without the BMDP. To sharpen the link between OMQ answering and CSP further, we also provide a converse polynomial reduction of CSPs over a datatype $\mathcal{D}$ to the complement of answering OMQs with BMDP over $\mathcal{D}$. This converse reduction can thus be used to transfer NP-hardness results from the CSP world to co-NP-hardness results for OMQ answering. More importantly, however, we now have a framework for transfering *complexity classification results* from CSP to OMQ answering.

We illustrate the power of this framework for all finite datatypes, datatypes $(\mathbb{Q}, \leq)$, and a family of datatypes with a first order definition in $(\mathbb{Z}, \overline{succ})$, where $\overline{succ}$ is the complement of the successor relation $succ = \{(a, b) \mid a + 1 = b\}$.

Note first that even without qualified attribute restrictions OMQs over datatype $(\mathbb{Q}, \leq)$ can express many interesting queries.

Example 1.0.1. Let $\mathcal{T} = \{\exists p \sqsubseteq \exists U, \exists p^- \sqsubseteq \exists U\}$ be a TBox, where p is a role name and U an attribute. Then the Boolean CQ

$$q \leftarrow p(x, y) \wedge U(x, u) \wedge U(y, v) \wedge u \leq v$$

is entailed by a KB $(\mathcal{T}, \mathcal{A})$ over $(\mathbb{Q}, \leq)$ such that $\mathcal{A}$ is an ABox containing no assertions using U iff $\mathcal{A}$ contains a p-cycle. Thus, answering the OMQ $(\mathcal{T}, q)$ is NLOGSPACE-complete. We also construct OMQs over $(\mathbb{Q}, \leq)$ that are PTIME-complete and, respectively, co-NP-complete.

Our main result is a PTIME/coNP-dichotomy for OMQs over $(\mathbb{Q}, \leq)$ with the BMDP. To formulate the dichotomy we associate with every OMQ $Q = (\mathcal{T}, q)$ a datatype pattern

$\mathsf{dtype}(Q) = (\theta_{\mathcal{T}}, \theta_q)$ such that $\theta_{\mathcal{T}}$ contains the datatype atoms in $\mathcal{T}$ and θ_q contains the datatype atoms in q. In Example 1.0.1, $\theta_{\mathcal{T}} = \emptyset$ and $\theta_q = \{u \leq v\}$. Then, based on the framework introduced above and a recent PTIME/NP-dichotomy result for temporal CSPs [19] we show that for any datatype pattern θ exactly one of the following two conditions holds (unless PTIME=co-NP):

- Evaluating OMQs Q with BMDP and $\mathsf{dtype}(Q) = \theta$ is always in PTime.
- There exists an OMQ Q with BMDP and $\mathsf{dtype}(Q) = \theta$ whose evaluation problem is coNP-hard.

In addition, our dichotomy comes with a purely syntactic description of the datatype patterns that lead to OMQs that are in PTIME. For example, the datatype pattern in Example 1.0.1 will always lead to an OMQ in PTime.

* * *

We now sum up the main contributions of this thesis, and then present the chapter organisation.

In this thesis, we obtain the following results:

- A framework for transfering complexity and dichotomy results from CSPs to OMQ answering, and vice-versa;
- A small correction of the dichotomy of temporal CSPs in [19];
- Undecidability results for OMQ answering over numerical datatypes with inequality and linear order;
- A decidability condition entirely based on the BMDP, a notion borrowed from the database literature;
- Based on results in the CSP literature, a series of instantiations of the transfer framework above, namely for all finite datatypes, dense linear orders, and integers with distance relations (i.e., certain datatypes using the successor relation).

In Chapter 2 we introduce the framework for this thesis. Namely, we define datatypes and then introduce the expressive language $\mathcal{ALCHI}$ and the Horn DL languages used throughout this work: the *DL-Lite* family and our main language Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$, where $\mathcal{D}$ is a datatype. Next we define the relevant query languages and their semantics. In particular, we introduce CQs and UCQs. We then discuss universal models and a generalization we call universal pre-models. Recall that $\mathcal{M}$ is a universal model of a knowledge base consisting of a TBox $\mathcal{T}$ and ABox $\mathcal{A}$ if the following holds for every UCQ q: $\mathcal{T}, \mathcal{A}$ entails q (i.e., q is satisfied in all models of $\mathcal{T}, \mathcal{A}$) iff $\mathcal{M} \models q$. Thus universal models reduce a deduction problem to an evaluation problem in a model. An ontology language $\mathcal{L}$ has the universal model property iff every satisfiable knowledge base in the language $\mathcal{L}$ has a universal model. Standard Horn DLs have the universal model property.

This fact is fundamental for analyzing the complexity of query answering in Horn DLs and for designing practical algorithms. In the presence of expressive datatypes, Horn DLs do not have the universal model property. We thus rely on a surrogate notion of models called pre-models. These are interpretations where the extensions of concepts and roles are fixed, but the value of attributes are left open using placeholders. We then introduce the notion of a completion function, one which assigns data values to such placeholders. Finally, we show that for query answering it suffices to consider particular pre-interpretations called universal pre-models (of satisfiable KBs) that exhibit the desired "universality" property.

Next in Chapter 3 we introduce CSPs as a means for proving complexity results in query answering; our transfer results will come in Chapter 5. Due to the presence of ABox elements in the query answering problem, with such results in mind we embark in the study of CSPs *with constants*. In the case of finite CSPs, we are able to establish equivalence with CSPs without constants by applying well-known and also very recent results– in particular, the proof of the Feder-Vardi conjecture. (This will allow us to prove a dichotomy (assuming PTIME $\neq$ NP) for query answering over *all* finite datatypes in Chapter 6.) We then embark on a thorough exploration of temporal CSPs, i.e., CSPs of templates with a first order definition in $(\mathbb{Q}, <)$.

In Chapter 4, we look in detail at the PTIME/NP dichotomy for temporal CSPs proved by Bodirsky and Kára and correct a minor error in the proof of the main result. This will allow us, using the results of Chapter 5, to show a PTIME/co-NP dichotomy for query answering over the datatype $(\mathbb{Q}, \leq)$.

Chapter 5 also explores decidability. We begin by showing, as mentioned above, that the general problem of query answering for $\mathit{DL\text{-}Lite}_{\mathcal{R}}^{attrib}(\mathcal{D})$ is already undecidable for typical datatypes. In more detail, we do a reduction from unrestricted tiling of the discrete plane into query answering over $\mathcal{D} \in \{(\mathbb{Z}, \neq), (\mathbb{Q}, \neq)\}$ with TBoxes in the language above. Then we reduce the later problem into query answering over $(\mathbb{Z}, <)$ and $(\mathbb{Q}, <)$. Finally, we show that query answering over $(\mathbb{Z}, \leq)$ is also undecidable. That motivates another topic of this chapter, the bounded match depth property (BMDP), already mentioned above. This notion comes from the database literature([30]. OMQs with that property are decidable since in order to check whether a query is entailed by a KB one only needs to consider a bounded subset of the chase: the subinterpretation induced by the elements reachable from ABox individuals in a fixed finite number of steps. This holds for any OMQ with a *DL-Lite* TBox such that the chase always terminates, for any OMQ whose UCQ is rooted, and for any safe OMQ (a weaker notion than rootedness that applies to many Boolean queries) over homogeneous datatypes. We present a complete proof for the latter case. Finally, we prove– and that is the main contribution of this chapter– the transfer theorem from query answering (for OMQs with the BMDP) into CSPs with constants and vice-versa; both reductions are polynomial time.

We then instantiate this transfer framework for a series of datatypes in Chapter 6, as stated ealier: all finite datatypes, dense linear orders and a large family of datatypes with a first order definition in $(\mathbb{Z}, \overline{succ})$. A complication with such instantiations of datatypes with infinite domain is that, in order to use results in CSPs, one has to first find a bridge from, and to CSPs with constants. This cannot be done in general as one has to present

a method for eliminating the constants from the inputs to CSPs that depends on the structure of the datatype. In the case of dense linear order we not only obtain the desired dichotomy for query answering, but we also classify tractable and (possibly) intractable cases based on the shape of the part of the query containing datatype atoms, as outlined above.

We close the thesis with a detailed overview of related work in Chapter 7.

2 Framework

2.1 Introduction

In this section we present the general framework for query answering over lightweight Description Logics with datatypes.

2.2 Datatypes

A *relational signature* Σ is a set $\{R_1, R_2, \ldots\}$ where each R_i is a relation symbol along with a function *arity*, where $arity(R_i) = k_i$ with $k_i > 0$. Also a *relational structure* over a (relational) signature Σ, also referred to as a Σ-*structure*, is a tuple $\Gamma = (D, R_1, R_2, \ldots)$ where D is a non-empty set, termed the *domain* of Γ, and each $R_i \subseteq D^{k_i}$ with R_i taken from Σ. Given one such structure Γ, the tuples in each R_i are fixed by its interpretation. Usually we use the same name to refer to a relation symbol and to the relation itself; when necessary, we use R^Γ to refer to the relation R as interpreted by a structure Γ. We also use $\mathrm{dom}(\Gamma)$ to refer to the domain of Γ.

Throughout this work a *datatype* is a relational structure.

Example 2.2.1. Typical datatypes are numerical datatypes such as $(A, <, \leq, >, \geq, =)$, where $A \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ and $<, \leq, >, \geq =$ are defined as usual; other common examples are strings with some ordering (e.g. lexicographical), the Boolean type, time instants and dates.

The *complement* of a datatype $\mathcal{D}$, denoted by $\overline{\mathcal{D}}$, is obtained by replacing each k_i-ary relation R_i in $\mathcal{D}$ by its complement $\overline{R_i} := D^{k_i} \setminus R_i$. In connection with relational structures we will use ω to denote the smallest infinite cardinal.

The following notion from first order logic will also be needed in connection with datatypes. A *primitive positive* (PP) formula φ over a datatype $\mathcal{D}$ is a logical formula of the form

$$\exists x_1, \ldots, x_m . \psi_1 \wedge \ldots \wedge \psi_\ell,$$

where each ψ_i, $1 \leq i \leq \ell$, is of the form $R(x_{i_1}, \ldots, x_{i_k})$ or $x_{i_1} = x_{i_2}$, with R a relation from $\mathcal{D}$ with arity k. When φ does not contain free variables it is called a *PP sentence*. When constants are allowed, we refer to the formulae obtained as *PP formulae with constants*, written PP_c.

2.3 The Description Logic $\mathcal{ALCHI}$, extensions and fragments

Here we define the Description Logics (DLs) [8] used in this work.

For all languages, we assume countably infinite and mutually disjoint sets N_C, N_R, N_U, N_A, of *concept names*, *role names*, *attribute names* and *individual names*, respectively. We next define the syntax and the semantics of the languages we consider.

Syntax The DL $\mathcal{ALCHI}$ is the well-studied [8] extension of $\mathcal{ALC}$ (first introduced in [96]) with inverse roles ("$\mathcal{I}$", in the usual naming convention) and role inclusions ("$\mathcal{H}$"), as defined next.

A role r is defined by the grammar

$$r := p \mid p^-,$$

where p ranges over all role names and p^- is called the *inverse role of* p. An $\mathcal{ALCI}$-concept C is defined by the following grammar:

$$C, D, E := A \mid \top \mid \bot \mid \neg C \mid D \sqcap E \mid C \sqcup D \mid \exists r.C \mid \forall r.C$$

where A ranges over all concept names. Then an $\mathcal{ALCHI}$-TBox is a finite set of *concept inclusions* of the form $C \sqsubseteq D$, where C, D are $\mathcal{ALCI}$-concepts, and *role inclusions* of the form $r_1 \sqsubseteq r_2$, where r_1, r_2 are roles.

We define, and consider throughout this work, two extensions of $\mathcal{ALCHI}$ with attributes over a fixed datatype $\mathcal{D}$. The first of them is $\mathcal{ALCHI}^{attrib}(\mathcal{D})$, where concepts C can also take the form of an *attribute restriction* $\exists U$. Moreover, a $\mathcal{ALCHI}^{attrib}(\mathcal{D})$-TBox, in addition to concept and role inclusions, also supports *attribute inclusions* of the form $U_1 \sqsubseteq U_2$. The second of them is $\mathcal{ALCHI}^{qattrib}(\mathcal{D})$, where, in addition to attribute restrictions and attribute inclusions, *qualified attribute restriction* are also allowed. These attribute restrictions are constructors of the form $\exists U.\varphi, \forall U.\varphi$, where U is an attribute and φ is a PP formula over $\mathcal{D}$ with constants and a single free variable x.

Finally, a *$\mathcal{D}$-ABox*, or simply ABox if the context is clear, consists of *assertions* of the form $A(a)$, $p(a, b)$, and $U(a, u)$, where A is a concept name, p is a role name, U is an attribute name, a, b are individual names, and $u \in \mathrm{dom}(\mathcal{D})$. A *$\mathcal{L}(\mathcal{D})$-Knowledge Base ($\mathcal{L}(\mathcal{D})$-KB)* is a pair $(\mathcal{T}, \mathcal{A})$ consisting of a $\mathcal{L}(\mathcal{D})$-TBox $\mathcal{T}$ and a $\mathcal{D}$-ABox $\mathcal{A}$, for $\mathcal{L} \in \{\mathcal{ALCHI}^{attrib}, \mathcal{ALCHI}^{qattrib}\}$. When necessary we use $\Sigma(\mathcal{T})$, for $\mathcal{T}$ a TBox, to refer to the symbols used in $\mathcal{T}$.

Example 2.3.1. Let $\mathcal{D} = (\mathbb{Q}, \leq)$. The KB $(\mathcal{T}_1, \mathcal{A})$ defined as

$$\begin{aligned}\mathcal{T}_1 = \{&\mathsf{Orc} \sqsubseteq \mathsf{Warrior},\ \mathsf{Warrior} \sqsubseteq \exists \mathsf{hitpoints}.(30 \leq x \leq 100),\\ &\mathsf{Warrior} \sqsubseteq \forall \mathsf{hitpoints}.(50 \leq x \leq 100),\\ &\exists \mathsf{profession}.\mathsf{Wizard} \sqsubseteq \forall \mathsf{hitpoints}.(10 \leq x \leq 50)\},\\ \mathcal{A} = \{&\mathsf{profession}(a, c), \mathsf{Wizard}(c),\ \mathsf{hitpoints}(a, 40),\ \mathsf{Orc}(b)\}\end{aligned}$$

is a $\mathcal{ALCHI}(\mathbb{Q}, \leq)$-KB.

Name	Syntax	Semantics
atomic concept	A	$A^{\mathcal{I}}$
top concept	$\top$	$\Delta^{\mathcal{I}}_{\mathsf{ind}}$
bottom concept	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^{\mathcal{I}}_{\mathsf{ind}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid \exists b \text{ with } (a,b) \in r^{\mathcal{I}} \;\&\; b \in C^{\mathcal{I}}\}$
universal restriction	$\forall r.C$	$\{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid (a,b) \in r^{\mathcal{I}} \text{ implies } b \in C^{\mathcal{I}}\}$
attribute restriction	$\exists U$	$\{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid \exists v((a,v) \in U^{\mathcal{I}})\}$
$\exists$-qualified attribute restriction	$\exists U.\varphi$	$\{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid \exists v\ ((a,v) \in U^{\mathcal{I}} \;\&\; \mathcal{D} \models \varphi(v))\}$
$\forall$-qualified attribute restriction	$\forall U.\varphi$	$\{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid \forall v\ ((a,v) \in U^{\mathcal{I}} \;\Rightarrow \mathcal{D} \models \varphi(v))\}$
concept inclusion	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
role inclusion	$r_1 \sqsubseteq r_2$	$r_1^{\mathcal{I}} \subseteq r_2^{\mathcal{I}}$
attribute inclusion	$U_1 \sqsubseteq U_2$	$U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}}$

Table 2.1: Syntax and semantics of $\mathcal{ALCHI}^{qattrib}(\mathcal{D})$ and fragments

Semantics An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ over a datatype $\mathcal{D}$ consists of a non-empty domain $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}}_{\mathsf{ind}} \cup \mathrm{dom}(\mathcal{D})$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns to each concept name A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_{\mathsf{ind}}$, to each role name p a relation $p^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_{\mathsf{ind}} \times \Delta^{\mathcal{I}}_{\mathsf{ind}}$, and to each attribute name U a relation $U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_{\mathsf{ind}} \times \mathrm{dom}(\mathcal{D})$. The elements in $\Delta^{\mathcal{I}}_{\mathsf{ind}}$ are called *individuals*, whereas the elements in $\mathrm{dom}(\mathcal{D})$ are called *data values*. We use $\mathrm{dom}(\mathcal{A})$ to refer to the set of individuals occurring in an ABox $\mathcal{A}$. We assume that $\Delta^{\mathcal{I}}_{\mathsf{ind}}$ and $\mathrm{dom}(\mathcal{D})$ are disjoint. Throughout this work, we make the *standard name assumption*: if $\mathcal{I}$ is an interpretation, then we set $a^{\mathcal{I}} := a$ for all individual names a. We also set $u^{\mathcal{I}} := u$ for each $u \in \mathrm{dom}(\mathcal{D})$, and $R^{\mathcal{I}} := R$ for each relation R of $\mathcal{D}$. The interpretation $\mathcal{I}$ induces the interpretations $C^{\mathcal{I}}$ and $r^{\mathcal{I}}$ for each concept C and role r as defined in Table 2.3. Attribute restrictions and qualified attribute restrictions are interpreted as follows:

$$
\begin{aligned}
(\exists U)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid \exists v\ ((a,v) \in U^{\mathcal{I}})\} \\
(\exists U.\varphi)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid \exists v\ ((a,v) \in U^{\mathcal{I}} \;\&\; \mathcal{D} \models \varphi(v))\}, \\
(\forall U.\varphi)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid \forall v\ ((a,v) \in U^{\mathcal{I}} \Rightarrow \mathcal{D} \models \varphi(v))\}.
\end{aligned}
$$

Table 2.3 presents the complete syntax and semantics of languages studied throughout this work.

The interpretation $\mathcal{I}$ is called a *model* of a $\mathcal{D}$-ABox $\mathcal{A}$ if $a \in A^{\mathcal{I}}$, $(a,b) \in p^{\mathcal{I}}$, and $(a,u) \in U^{\mathcal{I}}$ for all assertions $A(a)$, $p(a,b)$, and $U(a,u)$ in $\mathcal{A}$. It is called *model* of a $\mathcal{L}(\mathcal{D})$ TBox $\mathcal{T}$ if $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$ for every inclusion $X \sqsubseteq Y \in \mathcal{T}$.

An interpretation is a model of a $\mathcal{D}$-KB $(\mathcal{T}, \mathcal{A})$ if it is a model of both $\mathcal{A}$ and $\mathcal{T}$. A $\mathcal{D}$-KB $(\mathcal{T}, \mathcal{A})$ is *satisfiable* if it has a model; in this case we say that $\mathcal{A}$ is *satisfiable relative to*, or w.r.t., $\mathcal{T}$. We say KBs $\mathcal{K}, \mathcal{K}'$ are *equisatisfiable* if $\mathcal{K}$ is satisfiable if, and only if, $\mathcal{K}'$ is satisfiable.

We define a basic notion of homomorphisms for DLs without attributes. It will be extended later for more expressive languages. Given an $\mathcal{ALCHI}$-TBox $\mathcal{T}$, a *homomorphism* from an interpretation $\mathcal{I}$ to an interpretation $\mathcal{J}$ is a mapping $h\colon \Delta^{\mathcal{I}}_{\mathsf{ind}} \to \Delta^{\mathcal{J}}_{\mathsf{ind}}$ such that all invididual names are mapped to themselves and such that $h(a) \in A^{\mathcal{J}}$ if $a \in A^{\mathcal{I}}$ for all concept names $A \in \Sigma(\mathcal{T})$, and $(h(a), h(a')) \in r^{\mathcal{J}}$ if $(a, a') \in r^{\mathcal{I}}$ for all role names $r \in \Sigma(\mathcal{T})$.

2.4 Queries

Conjunctive queries (CQs) q over $\mathcal{D}$ take the form $q(\bar{x}) \leftarrow \varphi$, where $\bar{x}$ is the tuple of *answer variables* of q, and φ is a conjunction of atomic formulae of the form $A(y)$, $p(y, z)$, $U(y, u)$, or $R(u_1, \dots, u_k)$, where A, p, U, and R range over concept names, role names, attribute names, and relation names in $\mathcal{D}$, respectively; each y, z and each $u, u_1, \dots, u_k$ is a *variable*. The variables $u, u_1, \dots, u_k$ are called *data variables*. We use the term *non-data variable (non-data answer variable)* to refer to a variable (answer variable) that is not a data variable. As usual, all variables of $\bar{x}$ must occur in some atom of φ. A *match* of q in an interpretation $\mathcal{I}$ is a mapping μ from the variables of φ to $\Delta^{\mathcal{I}}$ such that for each atom $X(t_1, \dots, t_k)$ of φ we have $(\mu(t_1), \dots, \mu(t_k)) \in X^{\mathcal{I}}$.

Definition 2.4.1. Let $\mathcal{I}$ be an interpretation, q a query and $\bar{c}$ a tuple of individual names and data values. Then $\bar{c}$ is an *answer* to q in $\mathcal{I}$ if there is a match μ of q in $\mathcal{I}$ such that $\mu(\bar{x}) = \bar{c}$. This is denoted by $\mathcal{I} \models q(\bar{c})$.

A *union of conjunctive queries (UCQ)* q over $\mathcal{D}$ takes the form $q_1(\bar{x}), \dots, q_n(\bar{x})$, where each $q_i(\bar{x})$ is a CQ over $\mathcal{D}$. The q_i are called *disjuncts* of q. A tuple $\bar{c}$ of individual names and data values is an *answer* to q in an interpretation $\mathcal{I}$, denoted by $\mathcal{I} \models q(\bar{c})$, if $\bar{c}$ is an answer to some disjunct of q in $\mathcal{I}$.

Definition 2.4.2. Given an $\mathcal{L}(\mathcal{D})$-KB $(\mathcal{T}, \mathcal{A})$, an UCQ q over $\mathcal{D}$, and a tuple $\bar{c}$ of individual names and data values, we write $\mathcal{T}, \mathcal{A} \models q(\bar{c})$ if $\bar{c}$ is an answer to q in every model of $(\mathcal{T}, \mathcal{A})$.

An *ontology-mediated query (OMQ)* over $\mathcal{D}$ takes the form $Q = (\mathcal{T}, q)$, where $\mathcal{T}$ is a $\mathcal{L}(\mathcal{D})$-TBox and q is a UCQ over $\mathcal{D}$. Given a $\mathcal{D}$-ABox $\mathcal{A}$ and a tuple $\bar{c}$, we write $\mathcal{A} \models Q(\bar{c})$ if $\mathcal{T}, \mathcal{A} \models q(\bar{c})$.

Example 2.4.3. Let $\mathcal{T}_1, \mathcal{A}$ be as in Example 2.3.1, and let

$$\begin{aligned}\mathcal{T}_2 = \{&\mathsf{Orc} \sqsubseteq \mathsf{Warrior},\ \mathsf{Warrior} \sqsubseteq \exists \mathsf{hitpoints}.(50 \leq x \leq 100),\\ &\mathsf{Warrior} \sqsubseteq \forall \mathsf{hitpoints}.(50 \leq x \leq 100),\\ &\exists \mathsf{profession}.\mathsf{Wizard} \sqsubseteq \forall \mathsf{hitpoints}.(10 \leq x \leq 40)\}.\end{aligned}$$

Also let q be a UCQ defined as

$$q(x) \leftarrow \mathsf{Orc}(x), \mathsf{profession}(y, z), \mathsf{Wizard}(z), \mathsf{hitpoints}(x, u), \mathsf{hitpoints}(y, w), w \leq u.$$

. Then it can be checked that $Q_1 = (\mathcal{T}_1, q)$ and $Q_2 = (\mathcal{T}_2, q)$ are OMQs with $\mathcal{A} \models Q_2$ and $\mathcal{A} \not\models Q_1$.

2.5 Horn Description Logics

Rather then the full languages above, we focus on their so-called *Horn* versions. The name stems from the fact that these logics can be embedded into the Horn fragment of first-order logic; see [61, 39]. Such DLs were introduced in [62], although there the name "Horn DL" refered to the more expressive Horn-$\mathcal{SHIQ}$ only; see [68] for a thorough study of Horn DLs. Here we define Horn-$\mathcal{L}$ using the often used notion of positive and negative polarity of occurences of $\mathcal{L}$-concepts, where $\mathcal{L} \in \{\mathcal{ALCHI}, \mathcal{ALCHI}^{attrib}, \mathcal{ALCHI}^{qattrib}\}$. We define polarity in an inductive way:

1. A concept C occurs *positively* in C;

2. If C occurs positively in C', then C occurs positively in $C' \sqcap D, C' \sqcup D, \exists r.C', \forall r.C'$ and it occurs negatively in $\neg C'$;

3. If C occurs negatively in C', then C occurs negatively in $C' \sqcap D, C' \sqcup D, \exists r.C', \forall r.C'$ and it occurs positively in $\neg C'$.

We say that a concept C occurs positively in an inclusion $D \sqsubseteq C'$, and negatively in an inclusion $C' \sqsubseteq D$, if it occurs positively in C'. Also, we say that a concept C occurs negatively in an inclusion $D \sqsubseteq C'$, and positively in an inclusion $C' \sqsubseteq D$, if it occurs negatively in C'. Finally, we say that a concept C occurs positively (negatively) in a TBox $\mathcal{T}$ if C occurs positively (negatively) in some inclusion in $\mathcal{T}$.[1]

Now we are ready to define:

Definition 2.5.1. Let $\mathcal{T}$ be an $\mathcal{L}$-TBox. Then $\mathcal{T}$ is called *Horn* if in $\mathcal{T}$ the concepts $\exists U.\varphi$ and $\forall U.\varphi$ occur only positively, no concept of the form $C \sqcup D$ occurs positively, and no concept of the form $\neg C$ or $\forall r.C$ occurs negatively.

We define now two Horn languages which are fragments of Horn-$\mathcal{ALCHI}^{qattrib}$, namely *DL-Lite*$_{\mathcal{R}}^{attrib}$ and *DL-Lite*$_{\mathcal{R}}^{qattrib}$. They are well known members of the *DL-Lite* family [33, 2], in particular extensions of *DL-Lite*$_{\mathcal{R}}$. A concept $B := A \mid \exists r.\top \mid \forall r.\top \mid \exists U$, for A a concept name, r a role name and U an attribute name, is here called a *basic concept*. A *DL-Lite*$_{\mathcal{R}}^{attrib}(\mathcal{D})$ *TBox* is a finite set of *role inclusions*, *attribute inclusions*, and *concept inclusions*. The concept inclusions are restricted to ones of the form $B_1 \sqsubseteq B_2$ and $B_1 \sqsubseteq \neg B_2$ where B_1, B_2 are basic concepts. In TBoxes of the extension *DL-Lite*$_{\mathcal{R}}^{qattrib}(\mathcal{D})$ of *DL-Lite*$_{\mathcal{R}}^{attrib}(\mathcal{D})$, in addition, qualified attribute restrictions are allowed on the right hand side of inclusions.

It is more transparent to deal with TBoxes that have a reduced number of axiom types and nestings of concepts. With that purpose in mind, we define:

[1] One should be aware that it is possible for a concept to occur both positively and negatively in an axiom or in a TBox.

Definition 2.5.2. Let $\mathcal{T}$ be a TBox all whose inclusions are of the form

$$\begin{aligned}
&A \sqsubseteq \bot,\ A \sqsubseteq B,\ A_1 \sqcap A_2 \sqsubseteq B,\ \top \sqsubseteq A,\\
&A \sqsubseteq \exists r.B,\ A \sqsubseteq \forall r.B,\ \exists r.A \sqsubseteq B,\\
&\exists U \sqsubseteq A,\ A \sqsubseteq \exists U,\ A \sqsubseteq \exists U.\varphi,\ A \sqsubseteq \forall U.\varphi,\\
&r_1 \sqsubseteq r_2,\ U_1 \sqsubseteq U_2,
\end{aligned}$$

where A, A_1, A_2, B are concept names, r, r_1, r_2 are role names and U, U_1, U_2 are attribute names. Then we say that $\mathcal{T}$ is in *normal form*.

From now on we assume

$$\mathcal{L} \in \{\mathcal{ALCHI}, \mathcal{ALCHI}^{attrib}, \mathcal{ALCHI}^{qattrib}, \textit{DL-Lite}_{\mathcal{R}}^{attrib}, \textit{DL-Lite}_{\mathcal{R}}^{qattrib}\}.$$

We present a polynomial-time algorithm for transforming any Horn-$\mathcal{L}$-TBox into a Horn-$\mathcal{L}$ TBox $\mathcal{T}'$ that is in normal form. This transformation will preserve a desirable property of the original KB; namely, the resulting KB will be equivalent to the input *modulo* the new symbols introduced. This will be done via the notion of a (model-theoretic) conservative extension.

We use a strategy similar to the one used in [65]. Given a Horn-$\mathcal{L}$ TBox $\mathcal{T}$, let $\mathit{Sub}(\mathcal{T})$ denote the set of all concepts and subconcepts occuring in all inclusions in $\mathcal{T}$. Now for each $C \in \Sigma(\mathcal{T})$ (note that C can also be a subconcept in $\mathcal{T}$) we introduce a fresh concept name A_C. Let τ be an operation on $\mathit{Sub}(\mathcal{T})$ given by:

$$\begin{aligned}
\tau(A) &= A,\\
\tau(\top) &= \top,\\
\tau(\bot) &= \bot,\\
\tau(\neg C) &= \neg A_C,\\
\tau(C \sqcap D) &= A_C \sqcap A_D,\\
\tau(C \sqcup D) &= A_C \sqcup A_D,\\
\tau(\exists U) &= \exists U,\\
\tau(\exists U.\varphi) &= \exists U.\varphi,\\
\tau(\forall U.\varphi) &= \forall U.\varphi,\\
\tau(\exists r.C) &= \exists r.A_C,\\
\tau(\forall r.C) &= \forall r.A_C,
\end{aligned}$$

where A is a concept name, C, D are concepts, r a role name, U an attribute and φ a PP formula with one free variable. We define a new TBox $\mathcal{T}'$, starting with the role and attribute inclusions of $\mathcal{T}$. As a first step we add the following new inclusions:

Rule 1: $A_C \sqsubseteq \tau(C)$ for every concept C that occurs positively in $\mathcal{T}$;

Rule 2: $\tau(C) \sqsubseteq A_C$ for every concept C that occurs negatively in $\mathcal{T}$;

Rule 3: $A_C \sqsubseteq A_D$ for every concept inclusion $C \sqsubseteq D$ in $\mathcal{T}$.

It can be checked that as a result we obtain only inclusions of the form $A \sqsubseteq A'$, $A \sqsubseteq \tau(C_+)$, $\tau(C_-) \sqsubseteq A$, where:

- C_+ occurs positively in $\mathcal{T}$, that is, it is of the form

$$A',\ \top,\ \bot,\ \neg C,\ C \sqcap D,\ \exists r.C,\ \forall r.C,\ \exists U,\ \exists U.\varphi,\ \text{or } \forall U.\varphi;$$

- C_- occurs negatively in $\mathcal{T}$, that is, it is of the form

$$\top,\ \bot,\ A,\ C \sqcap D,\ A \sqcup D,\ \text{or } \exists r.C.$$

As a result we get only inclusions in normal form, except for ones of the form $A \sqsubseteq \top, A \sqsubseteq \neg A'$ and $A \sqsubseteq A_1 \sqcap A_2$. So as a second step we modify the inclusions above as follows. The inclusions of the form $A \sqsubseteq \top$ are deleted. As regards the ones of the form $A \sqsubseteq \neg A'$, they are replaced by the equivalent $A \sqcap A' \sqsubseteq \bot$; we then introduce a new concept name B together with the inclusions $A \sqcap A' \sqsubseteq B$ and $B \sqsubseteq \bot$, which are in normal form. Finally, for the ones of the form $A \sqsubseteq A_1 \sqcap A_2$, we delete them and introduce equivalent inclusions of the form $A \sqsubseteq A_1, A \sqsubseteq A_2$, which are in normal form. The whole transformation can be seen to take only polynomially many steps in the size of the input $\mathcal{T}$.

We now introduce the notion of model-theoretic conservative extension. As mentioned above, it will be used to prove, in a straightforward way, the property of KBs (preserved by TBoxes in normal form) which is crucial for query answering.

Definition 2.5.3. Let $\mathcal{T}, \mathcal{T}'$ be TBoxes. Then $\mathcal{T}'$ *is a model-theoretic conservative extension of* $\mathcal{T}$ if the following holds:

1. $\mathcal{T}' \models \mathcal{T}$;
2. for every model $\mathcal{I}$ of $\mathcal{T}$ there exists a model $\mathcal{I}'$ of $\mathcal{T}'$ that coincides with $\mathcal{I}$ regarding the interpretation of symbols in $\Sigma(\mathcal{T})$.

Example 2.5.4. Suppose, say, $\exists r.A_1 \sqsubseteq A_2 \sqcap A_3 \in \mathcal{T}$. We show that $\mathcal{T}' \models \exists r.A_1 \sqsubseteq A_2 \sqcap A_3$. By the third rule of the first stage of the transformation, $A_{\exists r.A_1} \sqsubseteq A_{A_2 \sqcap A_3} \in \mathcal{T}'$. The following inclusions are also in $\mathcal{T}'$:

$$\begin{aligned}
\tau(\exists r.A_1) = \exists r.A_{A_1} \sqsubseteq A_{\exists r.A_1} &\ \text{(rule 2)},\\
A_{A_2 \sqcap A_3} \sqsubseteq \tau(A_2 \sqcap A_3) = A_{A_2} \sqcap A_{A_3} &\ \text{(rule 1)},\\
\tau(A_1) = A_1 \sqsubseteq A_{A_1} &\ \text{(rule 2)},\\
A_{A_2} \sqsubseteq \tau(A_2) = A_2 &\ \text{(rule 1)},\\
A_{A_3} \sqsubseteq \tau(A_3) = A_3 &\ \text{(rule 1)},\\
A_{A_2 \sqcap A_3} \sqsubseteq A_{A_2} &\ \text{(second stage of transformation)},\\
A_{A_2 \sqcap A_3} \sqsubseteq A_{A_3} &\ \text{(second stage of transformation)}.
\end{aligned}$$

Let $\mathcal{I}$ be a model of $\mathcal{T}'$. Now assume $(a, b) \in r^{\mathcal{I}}$ with $b \in (A_1)^{\mathcal{I}}$ for individuals $a, b \in \Delta^{\mathcal{I}}$. Since $A_1 \sqsubseteq A_{A_1} \in \mathcal{T}'$, we have $b \in (A_{A_1})^{\mathcal{I}}$. Also $\exists r.A_{A_1} \sqsubseteq A_{\exists r.A_1} \in \mathcal{T}'$ implies $a \in (A_{\exists r.A_1})^{\mathcal{I}}$. Thus $a \in (A_{A_2 \sqcap A_3})^{\mathcal{I}}$. By the four last axioms in the list above, we obtain $a \in (A_2)^{\mathcal{I}}$ and $a \in (A_3)^{\mathcal{I}}$ as required.

We now prove:

Lemma 2.5.5. *Let $\mathcal{T}$ be a Horn-$\mathcal{L}$ TBox and $\mathcal{T}'$ the result of translating $\mathcal{T}$ into normal form. Then $\mathcal{T}'$ is a model-theoretic conservative extension of $\mathcal{T}$.*

Proof. To show (1) of Definition 2.5.3, it suffices to note that any model of $\mathcal{T}'$ is also model of $\mathcal{T}$: $\mathcal{T}$ and $\mathcal{T}'$ share the same role and attribute inclusions and the original concept inclusions are all entailed by $\mathcal{T}'$ due to the way the transformation is defined. This is done by structural induction on the axioms of $\mathcal{T}$.

Let $C \sqsubseteq D$ be an inclusion in $\mathcal{T}$. Then $A_C \sqsubseteq A_D \in \mathcal{T}'$ by the third transformation rule. Take a model $\mathcal{M}$ of $\mathcal{T}'$. We show that $\mathcal{M} \models C \sqsubseteq D$ by a structural induction over $C \sqsubseteq D$. Assume there is an individual $a \in \Delta^{\mathcal{M}}$ with $a \in C^{\mathcal{M}}$.

Base case for the first induction: $C := A$ (that is, C is a concept name which we rename to A). Then A occurs positively in A and negatively in $A \sqsubseteq D$. By the second rule, $\tau(A) = A \sqsubseteq A_A \in \mathcal{T}'$, so $\mathcal{M} \models A \sqsubseteq A_A$. Then $a \in A_A^{\mathcal{M}}$. By the third rule, $A_A \sqsubseteq A_D \in \mathcal{T}'$, so that $a \in A_D^{\mathcal{M}}$. Base case for the second induction: let $D := A'$. Since A' occurs positively in A', and positively in $A \sqsubseteq A'$, by the first rule we have $A'_A \sqsubseteq \tau(A') = A'$. Therefore $a \in (A')^{\mathcal{M}} = D^{\mathcal{M}}$ and we are through with the base case for both inductions.

For the IH, assume we have shown that $a \in D^{\mathcal{M}}$ for subconcepts $C', C_1, C_2, D', D_1, D_2$ of C and D, respectively. We have to consider the cases where (*) C is $\exists r.C', C_1 \sqcap C_2$ or $C_1 \sqcup C_2$ (first induction); and (**) D is $\neg D$, $\exists r.D'$, $\forall r.D'$, $\exists U$, $\forall U$, $\exists U.\varphi$, $\forall U.\varphi$ or $D_1 \sqcap D_2$ (second induction). We do one case for each.

As for (*), suppose C is $\exists r.C'$. Then there exists a $b \in \Delta^{\mathcal{M}}$ with $(a, b) \in r^{\mathcal{M}}$ and $b \in (C')^{\mathcal{M}}$. Since C occurs negatively in $C \sqsubseteq D$, by rule 2 we have $\tau(\exists r.C') = \exists r.A_{C'} \sqsubseteq A_{\exists r.C'} \in \mathcal{T}'$. Since C' occurs negatively in $\exists r.C' \sqsubseteq D$, by rule 2 we have that $\tau(C') \sqsubseteq A_{C'} \in \mathcal{T}'$. By the IH of the first induction, $b \in A_{C'}^{\mathcal{M}}$. Thus $a \in A_{\exists r.C'}^{\mathcal{M}}$. Since $A_{\exists r.C'} \sqsubseteq A_D \in \mathcal{T}'$ by rule 3, we have $a \in A_D^{\mathcal{M}}$. The remaining cases are similar.

As for (**), assume $a \in A_D^{\mathcal{M}}$. Suppose D is $\neg D'$. Since $\neg D'$ occurs positively in $C \sqsubseteq \neg D'$, we have by rule 1 $A_{\neg D'} \sqsubseteq \tau(\neg D') = \neg A_{D'}$. Since D' occurs negatively in $C \sqsubseteq \neg D'$, by rule 2 we have $\tau(D') \sqsubseteq A_{D'}$. By the IH, $a \notin (D')^{\mathcal{M}}$. The remaining cases are similar.

To show (2) of Definition 2.5.3, we need to prove that for every model $\mathcal{I}$ of $\mathcal{T}$ there exists a model $\mathcal{I}'$ of $\mathcal{T}'$ that coincides with $\mathcal{I}$ regarding the interpretation of symbols in $\Sigma(\mathcal{T})$. In order to find a suitable interpretation $\mathcal{I}'$, we require that $\mathcal{I}'$ interprets all old concepts of $\Sigma(\mathcal{T})$ exactly as in $\mathcal{I}$. As for the new concepts A_C in $\Sigma(\mathcal{T}') \setminus \Sigma(\mathcal{T})$ introduced by the transformation, it suffices to set $A_C^{\mathcal{I}'} := C^{\mathcal{I}}$. By the construction hinted it is clear that (2) of Definition 2.5.3 is satisfied.

Thus $\mathcal{T}'$ is a model-theoretic conservative extension of $\mathcal{T}$, as required. □

The result now follows:

Corollary 2.5.6. *Let $\mathcal{T}$ be a Horn-$\mathcal{L}$ TBox and $\mathcal{T}'$ the result of transforming $\mathcal{T}$ into normal form. Then for all ABoxes $\mathcal{A}$ consistent w.r.t. $\mathcal{T}$:*

1. *for all inclusions* $C \sqsubseteq D$ *that only use symbols from* $\Sigma(\mathcal{T})$*, we have* $(\mathcal{T}, \mathcal{A}) \models C \sqsubseteq D$ *iff* $(\mathcal{T}', \mathcal{A}) \models C \sqsubseteq D$*;*
2. *for all UCQs* q *that only use symbols from* $\Sigma(\mathcal{T})$*, we have* $(\mathcal{T}, \mathcal{A}) \models q$ *iff* $(\mathcal{T}', \mathcal{A}) \models q$*.*

Given that the transformation gives us a KB that is equivalent modulo the fresh concept names, from now on we can safely assume $\mathcal{L}$-TBoxes are given in normal form.

2.6 Universal models of Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$-KBs

In this section we present techniques for tackling the problem of query answering over Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$ Knowledge Bases and then prove useful results that rely on those techniques. They basically consist in extending the notion of universal models of KBs (defined next). These models can be generated by using a (possibly) infinite procedure. Most of the results in this section are extensions of known results for less expressive languages–namely, lightweight DLs without attributes.

Hom-initial models and the existence of universal models We introduce two notions that are central to the current inquiry.

Definition 2.6.1. Given a description logic $\mathcal{L}$ and a consistent $\mathcal{L}$-KB $(\mathcal{T}, \mathcal{A})$, suppose that there exists a model $\mathcal{M}$ of $(\mathcal{T}, \mathcal{A})$ such that there is a homomorphism from $\mathcal{M}$ to every model $\mathcal{M}'$ of $(\mathcal{T}, \mathcal{A})$. Then $\mathcal{M}'$ is called a *hom-initial model of* $(\mathcal{T}, \mathcal{A})$.

Remark 2.6.2. In the database literature, hom-initial models are known as universal models [44, 48]. Here we use more recent DL terminology (e.g. [78, 77]), reserving the latter term for query answering (see below).

The guaranteed existence of hom-initial models for an arbitrary consistent KB formulated in a language $\mathcal{L}$ depends on the properties of $\mathcal{L}$ itself. For example, it is known that, given a Datalog program $\mathbf{P}$, that is, a set of *function-free Horn clauses* [38], and a database D, i.e., a set of ground facts, there always exists a unique and finite "minimal" model of $(\mathbf{P}, D)$ [38]; see [91], section 4.1, for the general case of logic programs. It can then be shown that any DL KB that can be translated into a pair consisting of a Datalog program and a database has a hom-initial model which is, in addition, *finite*. Unfortunatelly, as we will see next, even for slightly more expressive languages finite hom-initial models do not exist in general.

For query answering purposes, we also adopt the following notion:

Definition 2.6.3. Let $(\mathcal{T}, \mathcal{A})$ be an $\mathcal{L}$-KB, for $\mathcal{L}$ a Description Logic. Suppose there exists a model $\mathcal{M}$ of $(\mathcal{T}, \mathcal{A})$ such that for all UCQs q, $(\mathcal{T}, \mathcal{A}) \models q$ if, and only if, $\mathcal{M} \models q$. Then $\mathcal{M}$ is called an *universal model of* $(\mathcal{T}, \mathcal{A})$.

We now show:

Proposition 2.6.4. *A hom-initial model of an* $\mathcal{L}$*-KB* $(\mathcal{T}, \mathcal{A})$ *is also an universal model of* $(\mathcal{T}, \mathcal{A})$*.*

Proof. Let $\mathcal{M}$ be a hom-initial model of $(\mathcal{T}, \mathcal{A})$ and $q(\bar{x})$ be a UCQ. We show that Definition 2.6.3 applies. So suppose $(\mathcal{T}, \mathcal{A}) \models q(\bar{x})$. Now, since in particular $\mathcal{M} \models (\mathcal{T}, \mathcal{A})$, we have $\mathcal{M} \models q(\bar{x})$. For the converse, assume $\mathcal{M} \models q(\bar{x})$. Let $\bar{a}$ be a tuple of individual names and data values from $\mathcal{A}$ and μ be a match of q in $\mathcal{M}$ such that $\mu(\bar{x}) = \bar{a}$. Given that $\mathcal{M}$ is hom-initial, there exists a homomorphism h from $\mathcal{M}$ into any model of $(\mathcal{T}, \mathcal{A})$. Let $\mathcal{M}'$ be one such model. We now compose μ and h into a homomorphism h' to obtain that $h'(\bar{x}) = \bar{a}$ so that $\mathcal{M}' \models q(\bar{a})$. Therefore $(\mathcal{T}, \mathcal{A}) \models q(\bar{x})$, and we have verified that $\mathcal{M}$ is a universal model of $(\mathcal{T}, \mathcal{A})$. This concludes the proof. □

The obvious practical consequence of the existence of universal models $\mathcal{M}$ of $\mathcal{L}$-KBs $(\mathcal{T}, \mathcal{A})$, for a given language $\mathcal{L}$, is that in answering queries over $(\mathcal{T}, \mathcal{A})$ one only has to consider one representative model, namely $\mathcal{M}$.

The canonical model construction for Horn-$\mathcal{ALCHI}$ via the chase We will now look at the well known canonical model construction for Horn-$\mathcal{ALCHI}$ KBs, which gives us (possibly) infinite hom-initial models. This construction will be later considerably extended to define suitable ("pre"-)models for Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$ for a datatype $\mathcal{D}$.

The construction uses the notion of a chase, originated in database theory as a tool for reasoning about data dependencies and checking query containment (see [1], chapters 8 and 10). Usually "chase" denotes both the procedure and its output. The procedure is defined nondeterministically and can be nonterminating. A modern definition of the chase for tractable DLs can be seen in [31].

Let $(\mathcal{T}, \mathcal{A})$ be a KB where $\mathcal{T}$ is a Horn-$\mathcal{ALCHI}$ TBox. Recall that we assume $\mathcal{T}$ to be in normal form. To present the chase procedure it is convenient to regard an interpretation simply as a set of assertions. As every ABox can be converted into an interpretation we will often not distinguish between the two. For example, we write $\exists r(a) \in \mathcal{A}$ for an ABox $\mathcal{A}$ if $a \in (\exists r)^{\mathcal{I}}$ for the interpretation $\mathcal{I}$ corresponding to $\mathcal{A}$ (which is the case if there exists b with $r(a, b) \in \mathcal{A}$). To simplify presentation, we assume that for any ABox $\mathcal{A}$, $p(a, b) \in \mathcal{A}$ iff $p^-(b, a) \in \mathcal{A}$.

So let $\mathcal{A}$ be the an ABox. Fresh individuals, i.e., individuals not occuring in $\mathcal{A}$, are called *nulls*. We assume a countably infinite number of nulls are given from a disjoint set. A rule is said to be *applicable* if its precondition is satisfied. Rules are applied exaustively as follows. We define the *chase step* as the application of one of the rules (when one of them is applicable) defined in Figure 2.1 below, to $\mathcal{A}$.

Definition 2.6.5. The *chase procedure* S for a Horn-$\mathcal{L}$-KB $(\mathcal{T}, \mathcal{A})$ in normal form is the sequence $S_0, S_1, \ldots$ where $S_0 = \mathcal{A}$ and each S_{i+1} is a set obtained by applying a chase step to S_i.

A chase procedure $S = S_0, S_1, \ldots$ is said to be *fair* if for all rules ρ which are applicable to some assertion α in S_i, $i \in \mathbb{N}$, there exists a chase step at S_j, with $j \geq i$, such that ρ is applied to α. We always assume that the chase procedure is fair. Notice that in Figure 2.1 A stands for a concept name and C for an arbitrary $\mathcal{ALCI}$-concept.

1. If $A \sqsubseteq \bot \in \mathcal{T}$ and $A(a) \in S$, then terminate and output $\mathsf{unsatisfiable}$;
2. If $C \sqsubseteq A \in \mathcal{T}$ and $C(a) \in S$ and $A(a) \notin S$, then add $A(a)$ to S.
3. If $C \sqsubseteq \exists r.A \in \mathcal{T}$ and $C(a) \in S$ and there is no $r(a,c) \in S$ with $A(c) \in S$, then add $r(a,c)$ and $A(c)$ to S, for c a fresh null;
4. If $C \sqsubseteq \forall r.A \in \mathcal{T}$ and $C(a) \in S$ and $r(a,b) \in S$ but $A(b) \notin S$, then add $A(b)$ to S;
5. If $r_1 \sqsubseteq r_2 \in \mathcal{T}$ and $r_1(a,u) \in S$ and $r_2(a,u) \notin S$, then add $r_2(a,b)$ to S;

Figure 2.1: Chase rules for Horn-$\mathcal{ALCHI}$-KBs

Example 2.6.6. Let $(\mathcal{T}, \mathcal{A})$ be a KB where

$$\begin{aligned}\mathcal{T} =&\{\mathsf{Orc} \sqsubseteq \mathsf{Warrior},\ \mathsf{Warrior} \sqsubseteq \exists \mathsf{weapon},\\ &\exists \mathsf{weapon} \sqsubseteq \exists \mathsf{lefthand.Shield}\},\\ \mathcal{A} =&\{\mathsf{Orc}(a)\}.\end{aligned}$$

Then the following sequence is a chase procedure for $(\mathcal{T}, \mathcal{A})$:

$$\begin{aligned}S =&\{\mathsf{Orc}(a)\},\\ &\{\mathsf{Orc}(a), \mathsf{Warrior}(a)\},\\ &\{\mathsf{Orc}(a), \mathsf{Warrior}(a), \mathsf{weapon}(a,c)\},\\ &\{\mathsf{Orc}(a), \mathsf{Warrior}(a), \mathsf{weapon}(a,c), \mathsf{lefthand}(a,d), \mathsf{Shield}(d)\}.\end{aligned}$$

We then define, for an $\mathcal{L}$-KB $(\mathcal{T}, \mathcal{A})$, the *chase output of* $(\mathcal{T}, \mathcal{A})$ (or simply *the chase of* $(\mathcal{T}, \mathcal{A})$), as $\text{chase}(\mathcal{T}, \mathcal{A}) := \bigcup_{i \in \mathbb{N}} S_i$.

For the cases where we do not obtain the output $\mathsf{unsatisfiable}$, we use the assertions contained in $\text{chase}(\mathcal{T}, \mathcal{A})$ to define the model $\mathcal{M}_S = (\Delta^{\mathcal{M}_S}, \cdot^{\mathcal{M}_S})$, called the *canonical model* of $(\mathcal{T}, \mathcal{A})$, as follows:

- $\Delta^{\mathcal{M}_S}_{\mathsf{ind}} := \{a \mid a \text{ is an individual or null occurring in } \text{chase}(\mathcal{T}, \mathcal{A})\}$;
- $A^{\mathcal{M}_S} := \{a \mid A(a) \in \text{chase}(\mathcal{T}, \mathcal{A})\}$;
- $r^{\mathcal{M}_S} := \{(a,b) \mid r(a,b) \in \text{chase}(\mathcal{T}, \mathcal{A})\}$.

We obtain the following result:

Proposition 2.6.7. *Let $(\mathcal{T}, \mathcal{A})$ be a Horn-$\mathcal{ALCHI}$-KB. Then*

- *$(\mathcal{T}, \mathcal{A})$ is unsatisfiable iff the chase procedure of $(\mathcal{T}, \mathcal{A})$ outputs* $\mathsf{unsatisfiable}$*;*
- *if $(\mathcal{T}, \mathcal{A})$ is satisfiable, then $\mathcal{M}_S$ is hom-initial for $(\mathcal{T}, \mathcal{A})$.*

Proof. See [49]. □

We will prove a stronger result later, which subsumes the above.

In [25] a finite generating structure for Horn-$\mathcal{ALCHI}$-KBs $(\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is consistent w.r.t. $\mathcal{A}$, is defined which can be constructed in time $|\mathcal{A}| \times 2^{p(|\mathcal{T}|)}$, for p a polynomial. The interpretation defined by unravelling this structure is then shown to homomorphically map to every model of $(\mathcal{T}, \mathcal{A})$ as in Proposition 2.6.7, (b).

Dealing with datatypes Differently from attribute-free Horn-$\mathcal{ALCHI}$, the more expressive languages studied here often do not have universal models. Intuitively, the reason for this is what follows. When generating a model for "abstract" axioms of the form $C \sqsubseteq \exists r$ one is often forced to introduce anonymous individuals (technically, nulls). It so happens that two anonymous individuals in such a model, in and of themselves, do not have any distinctive properties. Not so with concrete domains, which come with a fixed, built-in semantics. Under the open world semantics, different interpretations of anonymous attribute values yield different data values, and so the relationship that holds between them as per the built-in semantics is entirely open. This is particularly important in the case of query answering– hence for investigating the existence of universal models–, as the following example illustrates.

Example 2.6.8. Let $\mathcal{D} = (\mathbb{N}, \leq)$ and $(\mathcal{T}, \mathcal{A})$ be a Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$-KB where

$$\begin{aligned}\mathcal{T} = \{&\mathsf{Orc} \sqsubseteq \mathsf{Warrior},\ \mathsf{Hobgoblin} \sqsubseteq \mathsf{Warrior},\\ &\mathsf{Warrior} \sqsubseteq \exists \mathsf{defence}.(10 \leq x \leq 50),\\ &\mathsf{Warrior} \sqsubseteq \exists \mathsf{attack}.(10 \leq x \leq 50)\}.\\ \mathcal{A} = \{&\mathsf{Orc}(a),\ \mathsf{Hobgoblin}(b)\}.\end{aligned}$$

Also consider the CQs

$$\begin{aligned}q_1(x, y) &\leftarrow \mathsf{Warrior}(x), \mathsf{Warrior}(y), \mathsf{attack}(x, u_1), \mathsf{defence}(y, u_2), u_1 \leq u_2,\\ q_2(x, y) &\leftarrow \mathsf{Warrior}(x), \mathsf{Warrior}(y), \mathsf{attack}(x, u_1), \mathsf{defence}(y, u_2), u_2 \leq u_1.\end{aligned}$$

To show that $q_1(a, b)$ is not entailed by $(\mathcal{T}, \mathcal{A})$ we construct an interpretation $\mathcal{I}$ satisfying both $\mathcal{T}$ and $\mathcal{A}$ where e.g. $\mathsf{defence}^{\mathcal{I}} = \{(a, 10), (b, 10)\}$ and $\mathsf{attack}^{\mathcal{I}} = \{(a, 50), (b, 50)\}$. We do the same for obtaining $\mathcal{T}, \mathcal{A} \not\models q_2(a, b)$: we let $\mathcal{I}'$ be any model of $(\mathcal{T}, \mathcal{A})$ with e.g. $\mathsf{defence}^{\mathcal{I}} = \{(a, 50), (b, 50)\}$ and $\mathsf{attack}^{\mathcal{I}} = \{(a, 10), (b, 10)\}$. Yet, one will fail in constructing an universal model $\mathcal{J}$ (see Definition 2.6.3) of $\mathcal{T}$ and $\mathcal{A}$ with $\mathcal{J} \not\models q_1$ and $\mathcal{J} \not\models q_2$, for clearly no such model exists.

The simple counterexample above shows that Horn-$\mathcal{L}(\mathcal{D})$ does not have the universal model property. To remedy this, as a first step we introduce the notion of *pre-interpretations* (and *pre-models*), that is, interpretations that fix the extension of concepts and roles but leave open those of attributes, except for imposing labels with constraints on their possible values according to attribute restrictions occurring in the TBox. Those anonymous individuals will be referred to as *data nulls*. Moreover, the definition of *universal pre-models* of such KBs and *completions* thereof, i.e., assignments of data values that satisfy all the constraints labelling the nulls, will enable us, as a second step, to define a suitable extension of the notion of query entailment.

Pre-interpretations, pre-models and completions We now define pre-interpretations.

Definition 2.6.9. A *pre-interpretation* $\mathcal{I}$ over a datatype $\mathcal{D}$ is a triple $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, Z^{\mathcal{I}})$ where

- $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}}_{\mathsf{ind}} \cup \Delta^{\mathcal{I}}_{\mathsf{data}}$ is the domain of $\mathcal{I}$, where $\Delta^{\mathcal{I}}_{\mathsf{ind}}$ is a non-empty set of individuals, $\Delta^{\mathcal{I}}_{\mathsf{data}} = \mathrm{dom}(\mathcal{D}) \cup \Delta^{\mathcal{I}}_{\mathsf{null}}$ is a set of *data elements*, and $\Delta^{\mathcal{I}}_{\mathsf{null}}$ is a set of *data nulls*;
- $\cdot^{\mathcal{I}}$ is a function assigning to each concept name A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_{\mathsf{ind}}$, to each role name p a relation $p^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_{\mathsf{ind}} \times \Delta^{\mathcal{I}}_{\mathsf{ind}}$, and to each attribute name U a relation $U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_{\mathsf{ind}} \times \Delta^{\mathcal{I}}_{\mathsf{data}}$;
- $Z^{\mathcal{I}}$ maps each $u \in \Delta^{\mathcal{I}}_{\mathsf{data}}$ to a set $Z^{\mathcal{I}}(u)$ of PP formulae $\varphi(x)$ with constants over $\mathcal{D}$ such that
  (i) $\mathcal{D} \models \exists x(\bigwedge_{\varphi \in Z^{\mathcal{I}}(u)} \varphi(x))$ for every $u \in \Delta^{\mathcal{I}}_{\mathsf{null}}$; and
  (ii) if $d \in \mathrm{dom}(\mathcal{D})$, then $(x = d) \in Z^{\mathcal{I}}(d)$.

We extend the definition of the extension $C^{\mathcal{I}}$ of a concept C from interpretations to pre-interpretations by defining the extension of the concepts C not involving attributes in the straightforward way and by setting for any pre-interpretation $\mathcal{I}$ for $\mathcal{D}$:

- $(\exists U)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid \text{there exists } v \in \Delta^{\mathcal{I}}_{\mathsf{data}} \text{ with } (a, v) \in U^{\mathcal{I}}\}$;
- $(\exists U.\varphi)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid \text{there exists } v \in \Delta^{\mathcal{I}}_{\mathsf{data}} \text{ with } (a, v) \in U^{\mathcal{I}} \text{ and } \varphi \in Z^{\mathcal{I}}(v)\}$;
- $(\forall U.\varphi)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}}_{\mathsf{ind}} \mid \text{for all } v \in \Delta^{\mathcal{I}}_{\mathsf{data}} \text{ with } (a, v) \in U^{\mathcal{I}} \text{ we have } \varphi \in Z^{\mathcal{I}}(v)\}$;

For any inclusion $X \sqsubseteq Y$ we set $\mathcal{I} \models X \sqsubseteq Y$ if $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$. In this case we say that $\mathcal{I}$ satisfies $X \sqsubseteq Y$. $\mathcal{I}$ is a *pre-model* of a TBox if it satisfies all its inclusions. $\mathcal{I}$ is a *pre-model* of an ABox $\mathcal{A}$ if $a \in A^{\mathcal{I}}$ for all $A(a) \in \mathcal{A}$, $(a, b) \in p^{\mathcal{I}}$ for all $p(a, b) \in \mathcal{A}$, and $(a, d) \in U^{\mathcal{I}}$ for all $U(a, d) \in \mathcal{A}$. $\mathcal{I}$ is a *pre-model* of a KB $(\mathcal{T}, \mathcal{A})$ if it is a pre-model of $\mathcal{T}$ and $\mathcal{A}$.

A completion of a pre-interpretation replaces data nulls by suitable values from $\mathrm{dom}(\mathcal{D})$.

Definition 2.6.10. Let $\mathcal{J}$ be a pre-interpretation over $\mathcal{D}$. Then

1. a *completion function* f for $\mathcal{J}$ is a mapping $f\colon \Delta^{\mathcal{J}}_{\mathsf{null}} \to D$ such that $\mathcal{D} \models \varphi(f(u))$ for all $u \in \Delta^{\mathcal{J}}_{\mathsf{null}}$ and $\varphi(x) \in Z^{\mathcal{J}}(u)$.
2. The *completion* of $\mathcal{J}$ given by f, written $f(\mathcal{J})$, is the interpretation $\mathcal{I}$ defined by setting:
   - $\Delta^{\mathcal{I}}_{\mathsf{ind}} = \Delta^{\mathcal{J}}_{\mathsf{ind}}$;
   - $A^{\mathcal{I}} = A^{\mathcal{J}}$ for all $A^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}}_{\mathsf{ind}}$;
   - $r^{\mathcal{I}} = r^{\mathcal{J}}$ for all $r^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}}_{\mathsf{ind}} \times \Delta^{\mathcal{J}}_{\mathsf{ind}}$;
   - $U^{\mathcal{I}} = (U^{\mathcal{J}} \cap (\Delta^{\mathcal{J}}_{\mathsf{ind}} \times \mathrm{dom}(\mathcal{D}))) \cup \{(b, f(v)) \mid (b, v) \in U^{\mathcal{J}}, v \in \Delta^{\mathcal{J}}_{\mathsf{null}}\}$ for all $U^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}}_{\mathsf{ind}} \times \Delta^{\mathcal{J}}_{\mathsf{data}}$.

Example 2.6.11. A pre-interpretation $\mathcal{I}$ for the $(\mathbb{N}, \leq)$-KB $(\mathcal{T}, \mathcal{A})$ in example 2.6.8 is given by setting $\Delta^{\mathcal{I}} = \{a, b, u_1, u_2, v_1, v_2\}$; $\mathsf{Orc}^{\mathcal{I}} = \{a\}$, $\mathsf{Hobgoblin}^{\mathcal{I}} = \{b\}$; $\mathsf{Warrior}^{\mathcal{I}} = \{a, b\}$; $\mathsf{attack}^{\mathcal{I}} = \{(a, u_1), (b, v_1)\}$ and $\mathsf{defence}^{\mathcal{I}} = \{(a, u_2), (b, v_2)\}$. Notice that the data null u_1 is constrained, due to the inclusions in $\mathcal{T}$, to take some value within the range defined by $Z^{\mathcal{I}}(u_1) = \{(10 \leq x \leq 50)\}$. The same goes for the remaining data nulls. Let f_0 be a completion function defined as $f_0(u_1) = f_0(v_1) = 50$ and $f_0(u_2) = f_0(v_2) = 10$. Then f_0 gives attack interpreted in $f_0(\mathcal{I})$ as $\{(a, 50), (b, 50)\}$ and $\mathsf{defence}$ as $\{(a, 10), (b, 10)\}$.

Given a pre-model of a $\mathcal{D}$-KB $(\mathcal{T}, \mathcal{A})$, any completion function for $(\mathcal{T}, \mathcal{A})$ will give us a model of $(\mathcal{T}, \mathcal{A})$.

Lemma 2.6.12. *If $\mathcal{J}$ is a pre-model of a KB $(\mathcal{T}, \mathcal{A})$ and f is a completion function for $\mathcal{J}$, then $f(\mathcal{J})$ is a model of $(\mathcal{T}, \mathcal{A})$.*

Proof. Let $\mathcal{J}$ be a pre-model of $(\mathcal{T}, \mathcal{A})$. We define a completion function f for $\mathcal{J}$. By the definition of a pre-model, all assertions in $\mathcal{A}$, and all axioms in $\mathcal{T}$, are satisfied by $\mathcal{J}$. W.r.t. all assertions, as well as all axioms not involving attributes, nothing needs to be done. Axioms of the forms (1) $C \sqsubseteq \exists U.\varphi$ and (2) $C \sqsubseteq \forall U.\varphi$ are the only ones one needs to take care of. As for (1), for each $a \in C^{\mathcal{J}}$ we have $a \in (\exists U.\varphi)^{\mathcal{J}}$, so we do as follows. Recall that $a \in (\exists U.\varphi)^{\mathcal{J}}$ implies there exists $u \in \Delta^{\mathcal{J}}_{\mathsf{data}}$ with $(a, u) \in U^{\mathcal{J}}$ where $\varphi \in Z^{\mathcal{J}}(u)$. If $u = d$ for some $d \in \mathrm{dom}(\mathcal{D})$, by definition of $\mathcal{J}$ (being a pre-interpretation) we have $Z^{\mathcal{J}}(d) = \{x = d\}$; we thus set $f(d) = d$. Otherwise u is a data null. In this case we pick any d with $\mathcal{D} \models \varphi(d)$ thus setting $f(u) = d$. Either way we obtain that there exists $v \in \mathrm{dom}(\mathcal{D})$ with $(a, v) \in U^{\mathcal{I}}$ and $\mathcal{D} \models \varphi(v)$. Thus $\mathcal{I} \models C \sqsubseteq \exists U.\varphi$. As for (2), we obtain that $a \in (\forall U.\varphi)^{\mathcal{J}}$ implies for all $u \in \Delta^{\mathcal{J}}_{\mathsf{data}}$ with $(a, u) \in U^{\mathcal{J}}$ we have $\varphi \in Z^{\mathcal{J}}(u)$. If $U^{\mathcal{J}} = \emptyset$ we are done. Otherwise we have $U^{\mathcal{J}} = \{(a, u_1), \ldots, (a, u_n)\}$ where for each u_i, $1 \leq i \leq n$, we have $\varphi \in Z(u_i)$. Whenever $u_i = d$ we set $f(u_i) = d$, otherwise we fix some value d with $\mathcal{D} \models \varphi(d)$ and set $f(u_i) = d$. Either way we obtain that for all $v \in \mathrm{dom}(\mathcal{D})$ with $(a, v) \in U^{\mathcal{I}}$, $\mathcal{D} \models \varphi(v)$ follows. Thus $\mathcal{I} \models C \sqsubseteq \forall U.\varphi$. This altogether yields $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$. □

The existence of completion functions is also an important, if straightforward, fact which will be used later.

Lemma 2.6.13. *If $\mathcal{I}$ is a model of a Horn-$\mathcal{L}(\mathcal{D})$-KB $(\mathcal{T}, \mathcal{A})$, then there exists a pre-model $\mathcal{J}$ of $(\mathcal{T}, \mathcal{A})$ and a completion function f for $\mathcal{J}$ such that $f(\mathcal{J}) = \mathcal{I}$.*

Proof. Let $\mathcal{I}$ be a model of a Horn-$\mathcal{L}(\mathcal{D})$-KB $(\mathcal{T}, \mathcal{A})$. We define a completion function f for a particular pre-model $\mathcal{J}$ such that $f(\mathcal{J}) = \mathcal{I}$. To obtain $\mathcal{J}$, replace all data values in $\Delta^{\mathcal{I}}_{\mathsf{data}}$ not occuring in $\mathcal{A}$ by data nulls u from a (countably infinite) disjoint set and construct the sets $Z^{\mathcal{J}}(u)$.[2] It is easy to see that $\mathcal{J}$ is a pre-model of $(\mathcal{T}, \mathcal{A})$. Now define a function $f\colon \Delta^{\mathcal{J}}_{\mathsf{null}} \to \mathrm{dom}(\mathcal{D})$ given by $f(u) = d$ where d is the value in $\Delta^{\mathcal{I}}_{\mathsf{data}}$ which u replaces. Therefore given $u \in \Delta^{\mathcal{J}}_{\mathsf{null}}$ we have $\mathcal{D} \models \varphi(f(u))$ for all $\varphi \in Z^{\mathcal{J}}(u)$. Moreover, given the identity between data values from $\Delta^{\mathcal{I}}_{\mathsf{data}} \setminus \mathrm{dom}(\mathcal{A})$ assigned via attributes to individuals and the image of f, we have $f(\mathcal{J}) = \mathcal{I}$. □

[2] One way of constructing such sets is using the extension of the chase procedure presented later on in this section.

We now define hom-initial and universal *pre-models*.

Definition 2.6.14. A pre-model $\mathcal{I}$ of a KB $(\mathcal{T}, \mathcal{A})$ is *hom-initial* for $(\mathcal{T}, \mathcal{A})$ if for all models $\mathcal{I}'$ of $(\mathcal{T}, \mathcal{A})$, there is a homomorphism from $\mathcal{I}$ to $\mathcal{I}'$.

Definition 2.6.15. Let $\mathcal{I}$ be a pre-model of a KB $(\mathcal{T}, \mathcal{A})$ such that for any UCQ q and tuple $\bar{c}$, $(\mathcal{T}, \mathcal{A}) \models q(\bar{c})$ if, and only if, $f(\mathcal{I}) \models q(\bar{c})$ for all completion functions f. Then $\mathcal{I}$ is called a *universal pre-model of* $(\mathcal{T}, \mathcal{A})$.

We now show that any pre-model of a KB that is "initial" regarding homomorphisms is a universal pre-model of the KB.

Previously we defined homomorphisms from *interpretations to interpretations* for languages without existential and universal attribute restrictions. Here we define homomorphisms between a *pre-interpretation* for $\mathcal{D}$ and an interpretation for $\mathcal{D}$. Let $\mathcal{I}$ be a pre-interpretation and $\mathcal{J}$ be an interpretation for $\mathcal{D}$. A *homomorphism* from $\mathcal{I}$ to $\mathcal{J}$ is a mapping $h\colon \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ such that all invididual names and all $d \in \mathrm{dom}(\mathcal{D})$ are mapped to themselves and

(a) $h[\Delta^{\mathcal{I}}_{\mathsf{ind}}] \subseteq \Delta^{\mathcal{J}}_{\mathsf{ind}}$ such that $h(a) \in A^{\mathcal{J}}$ if $a \in A^{\mathcal{I}}$, and $(h(a), h(a')) \in r^{\mathcal{J}}$ if $(a, a') \in r^{\mathcal{I}}$;

(b) $h[\Delta^{\mathcal{I}}_{\mathsf{data}}] \subseteq \mathrm{dom}(\mathcal{D})$ such that if $(a, v) \in U^{\mathcal{I}}$, then $(h(a), h(v)) \in U^{\mathcal{J}}$ and $\mathcal{D} \models \varphi(h(v))$ for all $\varphi \in Z^{\mathcal{I}}(v)$.

Observe that if $\mathcal{I}$ is actually an interpretation (rather than a pre-interpretation only) then we obtain a homomorphism in the standard sense and it follows that for any UCQ q and tuple $\bar{c}$ of individual names and data values in $\mathcal{I}$ we have $\mathcal{J} \models q(\bar{c})$ whenever $\mathcal{I} \models q(\bar{c})$.

We then adapt the notion of hom-initial (defined previously for the canonical model construction) simply by using the newly defined homomorphisms from pre-interpretations to interpretations.

We show that such hom-initial models are universal pre-models.

Theorem 2.6.16. *Let $\mathcal{I}$ be a pre-model of a KB $(\mathcal{T}, \mathcal{A})$ which is hom-initial. Then $\mathcal{I}$ is a universal pre-model of $(\mathcal{T}, \mathcal{A})$.*

Proof. Let $\mathcal{I}$ be a hom-initial pre-model of a Horn-$\mathcal{L}(\mathcal{D})$-KB $(\mathcal{T}, \mathcal{A})$. We show that $\mathcal{I}$ satisfies Definition 2.6.15, that is, for any UCQ q and tuple $\bar{c}$, $(\mathcal{T}, \mathcal{A}) \models q(\bar{c})$ if, and only if, $f(\mathcal{I}) \models q(\bar{c})$ for all completion functions f.

So let $q(\bar{x})$ be a UCQ and $\bar{c}$ be a tuple of individuals and data values from $\mathcal{A}$.

First, assume for all models $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$ we have $\mathcal{I} \models q(\bar{c})$. Let f be a completion function for $\mathcal{I}$. By Lemma 2.6.12, $f(\mathcal{I})$ is a model of $(\mathcal{T}, \mathcal{A})$. Thus $f(\mathcal{I}) \models q(\bar{c})$ as desired.

Conversely, assume $f(\mathcal{I}) \models q(\bar{c})$ for all completion functions f and let $\mathcal{J}$ be a model of $(\mathcal{T}, \mathcal{A})$. We have to show that $\mathcal{J} \models q(\bar{c})$. By assumption there is a homomorphism h_0 from $\mathcal{I}$ to $\mathcal{J}$.

As shown in Figure 2.2, construct a completion function f_0 for $\mathcal{I}$ using h_0 by setting $f_0 := h_0 \restriction_{\Delta^{\mathcal{I}}_{\mathsf{data}}}$. f_0 is a completion function since $\mathcal{D} \models \varphi(h_0(u))$ for all $\varphi \in Z^{\mathcal{I}}(u)$, by definition of homomorphisms.

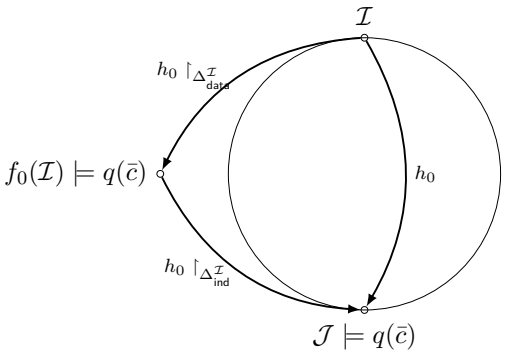

Figure 2.2: Theorem 2.6.16: construction of completion function f_0

Now the mapping $h : f_0(\mathcal{I}) \to \mathcal{J}$ defined by setting $h(a) = h_0(a)$ for all $a \in \Delta^{\mathcal{I}}_{\mathsf{ind}}$ and $h(d) = d$ for all $d \in \mathrm{dom}(\mathcal{D})$ is a homomorphism from $f_0(\mathcal{I})$ to $\mathcal{J}$. But then $\mathcal{J} \models q(\bar{c})$ since $f_0(\mathcal{I}) \models q(\bar{c})$. □

Computing the chase of Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$-KBs Previously we constructed a chase procedure for Horn-$\mathcal{ALCHI}$ KBs. Recall the general ideas and definitions on the chase presented there. We now extend the chase procedure to one that constructs for every satisfiable Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$-KB a hom-initial pre-model of the KB.

Here we regard a *pre-interpretation* as a set of assertions that can contain, in addition to ABox assertions, assertions of the form $U(a, u)$ with u a data null and sets $Z(u)$ of PP_c formulae for u a data value or data null. We refer to the resulting set of assertions as a *pre-ABox*.

Assume a Horn-$\mathcal{L}(\mathcal{D})$-KB $(\mathcal{T}, \mathcal{A})$ is given and let S be the pre-ABox obtained from $\mathcal{A}$ by adding $Z(d) = \{x = d\}$ for every data value d in $\mathcal{A}$. We now use as rules for the chase step the union of the sets of rules defined in Figures 2.1 and 2.3. Again define $\mathrm{chase}(\mathcal{T}, \mathcal{A}) = \bigcup_{i \in \mathbb{N}} S_i$. Notice that each S_i contains, together with assertions, a set $Z_i(u)$ for each data value or data null u.

Example 2.6.17. (Example of a finite chase.) Let $\mathcal{T} := \{A_1 \sqsubseteq \exists r.A_2, A_2 \sqsubseteq \exists U_1, U_1 \sqsubseteq U_2\}$ and $\mathcal{A} := \{A_1(a)\}$. Then

$$\begin{aligned} S = &\{A_1(a)\}, \\ &\{A_1(a), r(a,c), A_2(c)\}, \\ &\{A_1(a), r(a,c), A_2(c), U_1(c,u)\}, \\ &\{A_1(a), r(a,c), A_2(c), U_1(c,u), U_2(c,u)\}. \end{aligned}$$

In this case S is finite and, in addition, unique.

6. If $C \sqsubseteq \exists U \in \mathcal{T}$ and $C(a) \in S$ and there is no data value or data null v with $U(c, v) \in S$ then add $U(a, u)$ to S and set $Z(u) = \emptyset$, where u is a fresh data null.

7. If $C \sqsubseteq \exists U.\varphi \in \mathcal{T}$ and $C(a) \in S$ and there is no data value or data null v with $U(c, v) \in S$ and $\varphi \in Z(v)$, then add $U(a, u)$ to S and set $Z(u) = \{\varphi(u)\}$, where u is a fresh data null. Terminate and output unsatisfiable if $\mathcal{D} \not\models \exists x \varphi(x)$.

8. If $C \sqsubseteq \forall U.\varphi \in \mathcal{T}$ and $C(a) \in S$ and $U(c, u) \in S$ for u a data value or data null such $\varphi \notin Z(u)$, then add φ to $Z(u)$. Terminate and output unsatisfiable if $\mathcal{D} \not\models \exists x \bigwedge_{\varphi \in Z(u)} \varphi(x)$.

9. If $U_1 \sqsubseteq U_2 \in \mathcal{T}$ and $U_1(a, u) \in S$ and $U_2(a, u) \notin S$, then add $U_2(a, u)$ to S.

Figure 2.3: Additional chase rules for Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$

Example 2.6.18. (Example of an infinite chase.) Let $\mathcal{T} := \{A \sqsubseteq \exists r.A\}$ and $\mathcal{A} := \{A(a)\}$. We use nulls from a set $\{c_0, c_1, \ldots\}$. Then

$$\begin{aligned} S = &\{A(a)\}, \\ &\{A(a), r(a, c_0), A(c_0)\}, \\ &\{A(a), r(a, c_0), A(c_0), r(c_0, c_1), A(c_1)\}, \\ &\{A(a), r(a, c_0), A(c_0), r(c_0, c_1), A(c_1), r(c_1, c_2), A(c_2)\}, \\ &\ldots \end{aligned}$$

Therefore

$$\text{chase}(\mathcal{T}, \mathcal{A}) = \{A(a), r(a, c_0)\} \cup \{A(c_i) \mid i \in \mathbb{N}\} \cup \{r(c_i, c_{i+1}) \mid i \in \mathbb{N}\}.$$

Now we are ready to define:

Definition 2.6.19. The *canonical pre-model* of a satisfiable Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$-KB $(\mathcal{T}, \mathcal{A})$, denoted $\text{can}(\mathcal{T}, \mathcal{A})$, is the pre-interpretation obtained from $\bigcup_{i \in \mathbb{N}} S_i$ where for each data element $u \in \Delta_{\mathsf{data}}^{\text{can}(\mathcal{T}, \mathcal{A})}$ we set $Z^{\text{can}(\mathcal{T}, \mathcal{A})}(u) := \bigcup_{i \in \mathbb{N}} Z_i(u)$.

The canonical pre-model then has the desired property:

Lemma 2.6.20. *Let $(\mathcal{T}, \mathcal{A})$ be a satisfiable KB. Then $\text{can}(\mathcal{T}, \mathcal{A})$ is a pre-model of $(\mathcal{T}, \mathcal{A})$.*

Proof. Given that all assertions in $\mathcal{A}$ occur in $\text{chase}(\mathcal{T}, \mathcal{A})$ by the definition of the pre-ABox S, we immediately get $\text{can}(\mathcal{T}, \mathcal{A}) \models \mathcal{A}$. Now we show that $\text{can}(\mathcal{T}, \mathcal{A}) \models \mathcal{T}$. Given any axiom δ in $\mathcal{T}$, a simple argument shows that $\text{can}(\mathcal{T}, \mathcal{A}) \models \delta$ depending on the axiom's form. We prove that the claim holds for two representative cases:

- Let $\delta = C \sqsubseteq \exists U.\varphi$. Then if $C(a)$ occurs in S_j for some $j \in \mathbb{N}$, rule $C \sqsubseteq U.\varphi$ is applied at some step $i \geq j$ yielding $U(a,u) \in S_{i+1}$ and $\varphi \in Z_{i+1}(u)$. Thus $a \in C^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ implies there exists $u \in \Delta_{\mathsf{data}}^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ with $(a,u) \in U^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ and $\varphi \in Z^{\mathrm{can}(\mathcal{T},\mathcal{A})}(u)$. Therefore by definition $\mathrm{can}(\mathcal{T},\mathcal{A}) \models C \sqsubseteq \exists U.\varphi$.

- Let $\delta = C \sqsubseteq \forall U.\varphi$. If $C(a), U(a,v) \in S_j$ with $\varphi \notin Z_j(v)$, we have that rule $C \sqsubseteq \forall U.\varphi$ is applied at some step $i \geq j$ to the effect that $\varphi \in Z_{i+1}(v)$. Thus inductively $a \in C^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ implies for all $v \in \Delta_{\mathsf{data}}^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ with $(a,v) \in U^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ we have $\varphi \in Z^{\mathrm{can}(\mathcal{T},\mathcal{A})}(v)$. Therefore by definition we obtain that $\mathrm{can}(\mathcal{T},\mathcal{A}) \models C \sqsubseteq \forall U.\varphi$.

The arguments are similar for the remaining forms of axioms.

□

The notions and techniques used in the lemma that follows are similar to the ones used in the data exchange framework [49]; in particular see Lemma 3.4 in [49].

Lemma 2.6.21. *Let $(\mathcal{T},\mathcal{A})$ be a Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$-KB in normal form. Then*

1. *$(\mathcal{T},\mathcal{A})$ is unsatisfiable if, and only if, the chase procedure outputs* unsatisfiable*;*
2. *If $(\mathcal{T},\mathcal{A})$ is satisfiable, then $\mathrm{can}(\mathcal{T},\mathcal{A})$ is hom-initial for $(\mathcal{T},\mathcal{A})$.*

Proof. Let $(\mathcal{T},\mathcal{A})$ be a Horn-$\mathcal{ALCHI}^{qattrib}$-KB. We show the following:

A. if the chase of $(\mathcal{T},\mathcal{A})$ does not output unsatisfiable, then $(\mathcal{T},\mathcal{A})$ has a model.

B. if $(\mathcal{T},\mathcal{A})$ is satisfiable, then the chase of $(\mathcal{T},\mathcal{A})$ does not output unsatisfiable and $\mathrm{can}(\mathcal{T},\mathcal{A})$ is hom-initial for $(\mathcal{T},\mathcal{A})$.

Ad A., assume the chase of $(\mathcal{T},\mathcal{A})$ does not output unsatisfiable. Inspecting the rules of the chase (Figures 2.1 and 2.3), it is checked that the following holds:

I Rule 1 is not ever triggered;

II Whenever Rule 6 is triggered, the latter condition is not satisfied;

III Whenever Rule 7 is triggered, the latter condition is not satisfied.

To obtain that $(\mathcal{T},\mathcal{A})$ is satisfiable we only need to show that there exists an interpretation $\mathcal{I}$ with $\mathcal{I} \models (\mathcal{T},\mathcal{A})$. A simple inductive argument shows this. We start with an interpretation $\mathcal{I}_0$ with $\mathcal{I}_0 \models \mathcal{A}$ and at each step $i+1$ we extend (or repair) the previous interpretation $\mathcal{I}_i$ by taking an axiom of $\mathcal{T}$ into account. By the forms of axioms that can occur in $\mathcal{T}$, this procedure will only be blocked at some step j if either (i) there is an individual a with $a \in A^{\mathcal{I}_j}$ for some concept name A, and $A \sqsubseteq \bot \in \mathcal{T}$; or (ii) there is an individual a with $a \in C^{\mathcal{I}_j}$, for a concept C and $C \sqsubseteq \exists U.\varphi \in \mathcal{T}$, for U an attribute name, but $\mathcal{D} \not\models \exists x \varphi(x)$; or (iii) there is an individual a with $a \in C^{\mathcal{I}_j}$, $(a,v) \in U^{\mathcal{I}_j}$ for v a data value, C a concept and U an attribute name, and $C \sqsubseteq \forall U.\psi \in \mathcal{T}$, but

$\mathcal{D} \not\models \exists x \bigwedge_{\varphi \in \Phi} \varphi(x)$, where Φ contains ψ together with all other PP_c formulae (in universal U-attribute restrictions) in $\mathcal{T}$ that constrain v.[3] However, this is guaranteed not to happen, for either case would contradict (I), (II) or (III) above.

Now consider the interpretation obtained in the limit of the procedure, $\mathcal{I} := \bigcup_{i \in \mathbb{N}} \mathcal{I}_i$. It can be checked that $\mathcal{I}$ is a model of $(\mathcal{T}, \mathcal{A})$ as desired.

Ad B., take a model $\mathcal{M}$ of $(\mathcal{T}, \mathcal{A})$ and let $\mathrm{can}(\mathcal{T}, \mathcal{A})$ be the canonical pre-model of $(\mathcal{T}, \mathcal{A})$, that is, the pre-interpretation $(\Delta^{\mathrm{can}(\mathcal{T},\mathcal{A})}, \cdot^{\mathrm{can}(\mathcal{T},\mathcal{A})}, Z^{\mathrm{can}(\mathcal{T},\mathcal{A})})$. By Lemma 2.6.20, $\mathrm{can}(\mathcal{T}, \mathcal{A})$ is a pre-model of $(\mathcal{T}, \mathcal{A})$.

We construct a homomorphism from $\mathrm{can}(\mathcal{T}, \mathcal{A})$ to $\mathcal{M}$ using an intermediary pre-model. By Lemma 2.6.13 there exists a pre-model $\mathcal{M}'$ of $(\mathcal{T}, \mathcal{A})$ and a completion function f such that $f(\mathcal{M}') = \mathcal{M}$. Fix one such completion function f. We define a function

$$h \colon \Delta^{\mathrm{can}(\mathcal{T},\mathcal{A})} \to \Delta^{\mathcal{M}'}$$

by induction on the stepwise construction of $\mathrm{chase}(\mathcal{T}, \mathcal{A}) := \bigcup_{i \in \mathbb{N}} S_i$, used to define $\mathrm{can}(\mathcal{T}, \mathcal{A})$. Whilst presenting the inductive definition we also show that as a consequence a *partial* homomorphism from a pre-interpretation to an interpretation (which satisfies item (a) and partially item (b) of the definition) is constructed. We will then compose f with h to obtain a full homomorphism from $\mathrm{can}(\mathcal{T}, \mathcal{A})$ to $\mathcal{M}$.

For the induction basis, let $i = 0$. We set $h(a) = a$ for all individuals a, and $h(v) = v$ for all $v \in \mathrm{dom}(\mathcal{D})$, that occur in $\mathcal{A}$. It is easy to see that since $S_0 = \mathcal{A}$, if there is an individual $a \in \Delta^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ with $a \in A^{\mathrm{can}(\mathcal{T},\mathcal{A})}$, then $A(a) \in S_0$ and so given that by assumption $\mathcal{M}' \models \mathcal{A}$, we have $h(a) \in A^{\mathcal{M}'}$; also if there are individuals $a, b \in \Delta^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ with $(a, b) \in r^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ for a role name r, then $r(a, b) \in S_0$; thus $\mathcal{M}' \models \mathcal{A}$ yields $(h(a), h(b)) \in r^{\mathcal{M}'}$. Moreover, if there is an individual a and a data value $v \in \mathrm{dom}(\mathcal{D})$ in with $(a, v) \in U^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ for an attribute name U, then $U(a, v) \in S_0$, which together with the assumption that $\mathcal{M}' \models \mathcal{A}$ yields $(h(a), h(v)) \in U^{\mathcal{M}}$.

For the IH, we assume that h has been defined for $i = k \geq 0$ and consider all the possible rule applications at step k of the chase. So let S_{k+1} be obtained from S_k by applying

- Rule 2. Therefore $C \sqsubseteq A$ is applied to $C(a) \in S_k$ at step k and $A(a) \notin S_k$. Therefore $A(a) \in S_{k+1}$, and so $a \in A^{\mathrm{can}(\mathcal{T},\mathcal{A})}$. By the IH, we have $h(a) \in C^{\mathcal{M}}$. Since $\mathcal{M}' \models \mathcal{T}$, $h(a) \in A^{\mathcal{M}'}$.

- Rule 3. Therefore $C \sqsubseteq \exists p.A$ is applied to $C(a) \in S_k$ at step k and there is no individual c with $p(a, c)$, $A(c) \in S_k$. Therefore $p(a, c), A(c) \in S_{k+1}$ where c is a fresh individual, and so $(a, c) \in p^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ with $c \in A^{\mathrm{can}(\mathcal{T},\mathcal{A})}$. By the IH, we have $h(a) \in C^{\mathcal{M}'}$. Since $\mathcal{M}' \models \mathcal{T}$, there is at least one individual b with $(h(a), b) \in p^{\mathcal{M}'}$ and $b \in A^{\mathcal{M}'}$. We set $h(c) = b$. Therefore $(h(a), h(c)) \in p^{\mathcal{M}'}$ with $h(c) \in A^{\mathcal{M}'}$

- Rule 4 Therefore $C \sqsubseteq \forall p.A$ is applied to $C(a) \in S_k$ at step k, $p(a, b) \in S_k$ but $A(b) \notin S_k$ for some individual b. Then $A(b) \in S_{k+1}$. Thus $(a, b) \in r^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ with

[3] That is, we cannot replace v with any other value so as to satisfy all the relevant constraints on U-attributes.

$b \in A^{\mathrm{can}(\mathcal{T},\mathcal{A})}$. By the IH, $h(a) \in C^{\mathcal{M}'}$. Since $\mathcal{M}' \models \mathcal{T}$, for all c with $(h(a), c) \in r^{\mathcal{M}'}$ we have $c \in A^{\mathcal{M}'}$. We pick a c' not already in the image of h and set $h(b) = c'$. Therefore $(h(a), h(b)) \in r^{\mathcal{M}'}$ and $h(b) \in A^{\mathcal{M}'}$.

- Rule 5. Therefore $p \sqsubseteq p'$ is applied to $p(a, b) \in S_k$ at step k and $p'(a, b) \notin S_k$. Thus $p'(a, b) \in S_{k+1}$, and so $(a, b) \in p'^{\mathrm{can}(\mathcal{T},\mathcal{A})}$. By the IH, we have $(h(a), h(b)) \in p^{\mathcal{M}'}$. Then $\mathcal{M}' \models \mathcal{T}$ yields $(h(a), h(b)) \in (p')^{\mathcal{M}'}$.
- Rule 6. Therefore $C \sqsubseteq \exists U$ is applied to $C(a) \in S_k$ at step k and there is no data value or data null v with $U(a, v) \in S_k$. Therefore $U(a, u) \in S_{k+1}$ where u is a data null. Thus $(a, u) \in U^{\mathrm{can}(\mathcal{T},\mathcal{A})}$ where $\varphi \in Z^{\mathrm{can}(\mathcal{T},\mathcal{A})}(u)$. By the IH, we get $h(a) \in C^{\mathcal{M}}$. Since $\mathcal{M}' \models \mathcal{T}$, there is a data null u' with $(h(a), u') \in U^{\mathcal{M}'}$. We set $h(u) = u'$. Therefore $(h(a), h(u)) \in U^{\mathcal{M}'}$.
- Rule 7. Same as rule 6. See † below.
- Rule 8. Do nothing.
- Rule 9. Similar to rule 5 above.

Now set $g := f \circ h \colon \mathrm{can}(\mathcal{T}, \mathcal{A}) \to \mathcal{M}$ and that note that

$$\dagger\ Z^{\mathrm{can}(\mathcal{T},\mathcal{A})}(u) = Z^{\mathcal{M}'}(h(u)) \text{ for all } u \in \Delta_{\mathsf{null}}^{\mathrm{can}(\mathcal{T},\mathcal{A})}.$$

It can be checked that g satisfies both items of the definition of homomorphisms from pre-interpretations to interpretations. Therefore g is a homomorphism from $\mathrm{can}(\mathcal{T}, \mathcal{A})$ to $\mathcal{M}$ and, since $\mathcal{M}$ is arbitrary, by Definition 2.6.14, $\mathrm{can}(\mathcal{T}, \mathcal{A})$ is hom-initial.

Finally, it is also clear that, assuming the chase does output unsatisfiable, we obtain that either Rule 1, or the latter part of Rules 6 or 7 is applied. As a consequence it is clear that $(\mathcal{T}, \mathcal{A})$ is unsatisfiable.

The Lemma now follows from (A) and (B). □

Corollary 2.6.22. *For every satisfable Horn-$\mathcal{L}(\mathcal{D})$-KB $(\mathcal{T}, \mathcal{A})$, the pre-interpretation* $\mathrm{can}(\mathcal{T}, \mathcal{A})$ *is a universal pre-model of* $(\mathcal{T}, \mathcal{A})$.

Proof. Directly from Lemma 2.6.21 and Theorem 2.6.16. □

3 Constraint Satisfaction Problems

3.1 Introduction

In this chapter we present a class of combinatorial problems known as constraint satisfaction problems (CSPs in short). In the following section we provide basic notions and introduce, in detail, techniques used later for proving relevant results.

Then we present a PTIME/NP dichotomy result on CSPs proved in [19] for structures refered to in the literature as *temporal constraint languages*— technically, the structures all whose relations have a first-order definition in $(\mathbb{Q}, <)$—, which will be later used for classifying the complexity of query answering over the datatype $(\mathbb{Q}, \leq)$.

3.2 Definitions and results on CSPs

CSPs have been around for some time under different guises. In this framework, pioneered by Montanari in the 1970s [84], it is possible to model a great variety of real-life problems. They are of both practical and theoretical interest.

Informally, in a CSP we are given a set of constraints with variables over fixed domains and the question is whether there is an assignment of domain elements to the respective variables such that all constraints are satisfied. Natural variations have been proposed and investigated, e.g. the question as to how many assignments satisfy the constraints, or finding an assignment satisfying the greatest number of constraints (exactly or otherwise), but here we restrict ourselves to the above task.

There are several ways of formally defining the decision problem just sketched. We present the formulation that better suits the approach presently undertaken; namely, casting CSPs as the problem of evaluating a certain class of first-order formulae; see e.g. [17]. Next we present basic results on the complexity of CSPs. We close this section with a discussion of relevant dichotomy results and the algebraic approach to proving them.

3.2.1 The computational problem

Recall the definitions introduced in Section 2.2. Let us say we have as inputs PP sentences φ over a structure Γ and the problem is whether φ is satisfied in Γ. Many constraint satisfaction problems can be formalised in this way [17, 18, 21, 15] by choosing an appropriate structure. In this context, the constraints are syntactic objects (the conjuncts of φ). We give a formal definition.

Let Γ be a Σ-structure. Recall that PP formulae φ over Γ are first-order formulae of the form

$$\exists x_1, \ldots, x_p . \psi_1 \wedge \ldots \wedge \psi_\ell,$$

where each ψ_i, $1 \leq i \leq \ell$, is of the form $R(x_1, \ldots, x_k)$ or $x = y$, with R a relation symbol from Σ where $\mathit{arity}(R) = k$. From here on, for a *fixed* relational structure Γ over a signature Σ, here termed a *constraint language* or a *template*,[1] we use the following definition:

$\mathsf{CSP}(\Gamma)$	
Input:	a PP sentence φ over Γ
Question:	$\Gamma \models \varphi$?

A satisfying assignment for an instance $\varphi \in \mathsf{CSP}(\Gamma)$ is called a *solution* for φ.

Another usual way of defining CSPs is in terms of homomorphisms. Let Γ, Γ' be Σ-structures. Then a *homomorphism from* Γ *to* Γ' is a function f from $\mathrm{dom}(\Gamma)$ to $\mathrm{dom}(\Gamma')$ such that for each n-ary relation symbol R in Σ and each n-tuple $\bar{t} \in R^\Gamma$, we have that $(f(a_1), \ldots, f(a_n)) \in R^{\Gamma'}$. If f is injective (surjective, bijective) we say that f is an *injective (surjective, bijective) homomorphism.* A bijective homomorphism from a structure Γ to itself is called an *automorphism of* Γ. If structures Γ, Γ' are such that there is homomorphism from Γ to Γ' and a homomorphism from Γ' to Γ, then we say that Γ *is homomorphically equivalent to* Γ'. An *endomorphism* Γ is a homomorphism from Γ to itself. Given an injective homomorphism h from Γ to Γ' that preserves the complement of each relation, we say that h is an *embedding.* Finally, a structure Γ is called a *core* if all its endomorphisms are also embeddings.

For finite structures, the following results are useful:

Proposition 3.2.1. *([58]) Let Γ be a finite structure. Then Γ is homomorphically equivalent to a unique (up to isomorphism) core Γ'.*

Proposition 3.2.2. *([58]) Let Γ, Δ be finite structures and Γ', Δ' be the cores of Γ and Δ, respectively. Then there is a homomorphism from Γ to Δ if, and only if, there is a homomorphism from Γ' to Δ'.*

We illustrate this with a very simple example.

Example 3.2.3. Let $\Gamma = (\{1, 2, 4, 5\}, \mathit{succ})$, where $\mathit{succ} := \{(x, y) \in (\mathrm{dom}(\Gamma))^2 \mid x + 1 = y\}$. Also let $\Gamma' = (\{1, 2\}, \mathit{succ})$. Then Γ and Γ' are homomorphically equivalent. To check this, let $h\colon \mathrm{dom}(\Gamma) \to \mathrm{dom}(\Gamma')$ such that $h(1) = h(4) = 1, h(2) = h(5) = 2$; and let $g\colon \mathrm{dom}(\Gamma') \to \mathrm{dom}(\Gamma)$ such that $g(1) = 4, g(2) = 5$. Also, Γ' is a core and it is unique, up to isomorphism; that is, Γ' is isomorphic to any other core of Γ.

Now consider the following fundamental algebraic problem. Fix a Σ-structure Γ. Then $\mathsf{CSP}(\Gamma)$ is the problem to decide, given a Σ-structure Γ', whether there is a homomorphism $h\colon \Gamma' \to \Gamma$. This is a traditional way of defining CSPs, see [50]. The following example illustrates this.

[1] We will use the terms interchangeably throughout this work.

Example 3.2.4. Analogously to the definition of homomorphism given above, a *digraph homomorphism* of a directed graph $G = (V_G, E_G)$ to a directed graph $H = (V_H, E_H)$ is a function h from V_G to V_H where for each edge $(v, w) \in E_G$ we have $(h(v), h(w)) \in E_H$. Now fix a directed graph H. Then the digraph homomorphism problem is, given a directed graph G, whether a digraph homomorphism $h\colon G \to H$ can be found. It can be cast as a homomorphism problem as above and is refered to in the combinatorial literature as H-COLOURING [59]. This decision problem generalises k-COLOURING, which is the problem to decide, given an *undirected* graph G, whether we can assign one out of k possible colours to each vertex in such a way that no two adjacent vertices share the same colour. To see this, let K_n denote the complete loopless undirected graph on n vertices. If there is a homomorphism from a graph G to K_n, then G is n-colourable. The converse also holds. Equivalently, G is a "yes"-instance of $\mathsf{CSP}(K_n)$. In other words, a graph G is n-colourable iff there is a homomorphism from G to K_n.

We can recast the problem from Example 3.2.4 again as a CSP for a relational structure and a PP sentence instead. Let $H = (V_H, E_H)$ be a digraph and $G = (V_G, E_G)$ be an input to H-COLOURING. To encode the problem we simply translate H into a relational structure $\Gamma_H = (V_H, E_H)$ and G into a PP sentence φ_G over Γ_H, which we encode using the form $\exists \bar{x} \bigwedge_{(i,j)\in E_G} E_H(x_i, x_j)$. Then G is an instance of H-COLOURING iff $\Gamma_H \models \varphi_G$. For instance, let $H = (V_H, E_H)$ be a digraph with $V_H = \{1, 2, 3, 4\}$ and $E_H = \{(1, 3), (1, 4), (2, 3)\}$ and $G = (V_G, E_G)$ be a digraph with $V_G = \{1, 2, 3\}$ and $E_G = \{(1, 2), (1, 3)\}$. Construct the PP sentence $\varphi_G = \exists x_1, x_2, x_3(E_H(x_1, x_2) \wedge E_H(x_1, x_3))$. Then clearly $\Gamma_H \models \varphi_G$, for e.g. the assignment α given by $\alpha(x_1) = 1,\ \alpha(x_2) = 3,\ \alpha(x_3) = 4$ makes φ_G true in Γ_H.

Here is another typical problem that can be defined in this way.

Example 3.2.5. Let $\Gamma_{lin} = (D, R^1_{lin}, R^2_{lin}, \dots)$ where D is any field (for instance, simply $\mathbb{R}$ or the field of p-adic numbers for p a prime number) and each R^i_{lin} is a k_i-ary relation defined as follows:

$$R^i_{lin} := \{(x^i_1, \dots, x^i_{k_i}) \in D^{k_i} \mid a^i_1 x^i_1 + \dots + a^i_k x^i_{k_i} = r^i\},$$

where all $a^i_1, \dots, a^i_{k_i}, r^i \in D$. An instance of $\mathsf{CSP}(\Gamma_{lin})$ is a system of linear equations $p_1, \dots, p_m$. A solution, when it exists, is a satisfying assignment to the variables; that is, a mapping from all tuples of k_i variables in all p_i to D^{k_i}, for $1 \leq i \leq m$, such that $t_i \in R^i_{lin}$ holds for each resulting k_i-tuple $t_i \subseteq D^{k_i}$.

We finish this section by defining reductions between CSPs, a notion that will be used next.

Definition 3.2.6. Given templates Γ, Γ' we say that $\mathsf{CSP}(\Gamma)$ *polynomially reduces to* $\mathsf{CSP}(\Gamma')$, in symbols $\mathsf{CSP}(\Gamma) \leq_p \mathsf{CSP}(\Gamma')$, if there is a polynomial-time algorithm that transforms any instance ψ of $\mathsf{CSP}(\Gamma)$ into an instance ψ' of $\mathsf{CSP}(\Gamma')$ such that $\psi \in \mathsf{CSP}(\Gamma)$ if, and only if, $\psi' \in \mathsf{CSP}(\Gamma')$. We say that $\mathsf{CSP}(\Gamma)$ *is polynomially equivalent to* $\mathsf{CSP}(\Gamma')$, in symbols $\mathsf{CSP}(\Gamma) \equiv_p \mathsf{CSP}(\Gamma)$, if it is the case that $\mathsf{CSP}(\Gamma) \leq_p \mathsf{CSP}(\Gamma')$ and $\mathsf{CSP}(\Gamma') \leq_p \mathsf{CSP}(\Gamma)$.

3.2.2 CSPs with constants

In our study of data complexity of OMQ answering, the goal is to translate such problems into the framework of CSPs. The main difficulty in that scenario is that such translations produce input instances to the CSP that can contain *constants*. In typical CSPs, on the other hand, such parameters are not allowed. We define this problem next.

Recall from Section 2.2 that PP formulae with constants over a Σ-structure Γ are denoted PP_c formulae, or PP_c sentences, when they do not contain free variables. Now, for PP_c sentences as inputs we formulate the problem:

$\mathsf{CSP}_c(\Gamma)$	
Input:	a PP_c sentence φ over Γ
Question:	$\Gamma \models \varphi$?

Related problems in algebra and graph theory have been called *retraction problems* [58]. The retraction problem for a finite domain structure $\mathcal{H}$ is, given a substructure $\mathcal{G}$ of $\mathcal{H}$, whether there is a homomorphism from $\mathcal{G}$ to $\mathcal{H}$ that is the identity on $\mathcal{H}$. Such problems are polynomially equivalent to $\mathsf{CSP}(\mathcal{H}')$ where $\mathcal{H}'$ is $\mathcal{H}$ added with unary relations for all elements of $\mathrm{dom}(\mathcal{H})$ [51]. We explore useful equivalences next.

Remark 3.2.7. Whereas it is clear that all instances of $\mathsf{CSP}(\Gamma)$ are instances of $\mathsf{CSP}_c(\Gamma)$, in symbols $\mathsf{CSP}(\Gamma) \subseteq \mathsf{CSP}_c(\Gamma)$, for all templates Γ, individual inputs to $\mathsf{CSP}_c(\Gamma)$ can be harder than their constant-less counterparts in $\mathsf{CSP}(\Gamma)$. The intuition is that constants in the input (which can be seen as playing the role of partial candidate solutions) can rule out certain solutions at an early stage of computation. E.g., let $\Gamma = (\mathbb{N}, R)$ where $R := \{(a, b) \in \mathbb{N}^2 \mid a = b \vee \sqrt{a} = b\}$, and let $\psi = \exists xy.R(x, y)$. It is easy to check that $\psi \in \mathsf{CSP}(\Gamma)$ by simply picking any $n \in \mathbb{N}$ and setting $x, y =: n$. Now let $\psi' = R(c, d)$ where $c, d \in \mathbb{N}$ with $c \neq d$. Then checking $\psi' \in \mathsf{CSP}_c(\Gamma)$ requires either computing the square root of c or squaring d.

When dealing with templates with finite domain, one can show polynomial equivalence between CSP_c and CSP. To show this, we use two similar (and polynomially equivalent) formulations, one with unary predicates, and another one with partial mappings, in place of constants. The reason for introducing them is to use known results. We say that a Σ-template Γ *admits precolouring* if for all elements a of $\mathrm{dom}(\Gamma)$, Σ contains a unary predicate symbol P_a and $P_a^\Gamma = \{a\}$. Clearly, if a template Γ admits precolouring we can obtain a new template Γ' by simply dropping the unary predicates; for cores, the same problems can then be simulated by using constants instead of the unary predicates. The following is known [70]:

Proposition 3.2.8. *Let Γ be a core. Then there exists a template Γ' that admits precolouring such that $\mathsf{CSP}(\Gamma') \equiv_p \mathsf{CSP}(\Gamma)$.*

We formulate a polynomial-time equivalent version of CSPs for templates that admit precolouring, now in terms of partial mappings. Let Γ be a template, φ a PP sentence and h be a partial mapping from the variables in φ to elements of $\mathrm{dom}(\Gamma)$. We write $\varphi[h]$ to mean φ with assignment h of values to variables of φ. We define:

pre-$\mathsf{CSP}(\Gamma)$	
Input:	a PP sentence φ over Γ and a partial mapping $h\colon \mathit{Var}(\varphi) \to \mathrm{dom}(\Gamma)$
Question:	Is there an extension h' of h such that $\Gamma \models \varphi[h']$?

The following results then connect pre-CSP, CSP_c and CSP for finite cores. They will be used later for proving a dichotomy for OMQ answering over finite datatypes.

Theorem 3.2.9. *([70]) Let Γ be a core with $|\,\mathrm{dom}(\Gamma)| < \omega$. Then pre-$\mathsf{CSP}(\Gamma) \equiv_p \mathsf{CSP}(\Gamma)$.*

Proposition 3.2.10. *For any template Γ we have pre-$\mathsf{CSP}(\Gamma) \equiv_p \mathsf{CSP}_c(\Gamma)$.*

Proof. Let Γ be a template and φ be an instance of pre-$\mathsf{CSP}(\Gamma)$, and h be the partial mapping of the input. The result φ_c of applying h to get the values for the variables in φ and replacing the variables by the corresponding values is clearly an input to $\mathsf{CSP}_c(\Gamma)$. It is then easy to see that $\varphi \in \mathsf{CSP}(\Gamma)$ iff $\varphi_c \in \mathsf{CSP}_c(\Gamma)$. The other direction is shown by a similar argument. □

Using Theorem 3.2.9 and Proposition 3.2.10, we obtain:

Corollary 3.2.11. *$\mathsf{CSP}_c(\Gamma) \equiv_p \mathsf{CSP}(\Gamma)$, for Γ a finite core.*

For finite structures Γ, the following result allows one to focus on the core of Γ.

Proposition 3.2.12. *If Γ is such that $|\,\mathrm{dom}(\Gamma)| < \omega$ and Γ' is the core of Γ, then $\mathsf{CSP}(\Gamma) = \mathsf{CSP}(\Gamma')$.*

These problems prompt the investigation of structures Γ for which the associated CSP is tractable.

3.2.3 Complexity and dichotomies

How hard are CSPs? Let us consider a tractable case, Example 3.2.5 above: such mappings– when they exist– can be found in polynomial time by using the known method called Gaussian elimination.[2] So $\mathsf{CSP}(\Gamma_{lin})$ turns out to be tractable. Actually, a basic fact on the complexity of CSPs is that if we require Γ to be finite (that is, $|\,\mathrm{dom}(\Gamma)| < \omega$), then $\mathsf{CSP}(\Gamma)$ is in NP but, unsurprisingly, not guaranteed to be polynomially solvable, modulo the assumption PTIME $\neq$ NP. On the other hand, in the context of infinite templates it is not hard to come up with undecidable CSPs, as the following example shows.

Example 3.2.13. In contrast to equations in Example 3.2.5, a *Diophantine equation* is any equation that admits only integer solutions. Hilbert's 10th problem is the problem to decide, given an arbitrary Diophantine equation, whether it has a solution. In other words, it asks for an universal algorithm for solving such equations. The problem was proved

[2] In fact, Gaussian elimination generalises to an algebraic method on which some of the known algorithms for solving finite CSPs are based [51].

undecidable by Yuri Matiyasevitch [81]. Now let $R := \{(x, y, z) \in \mathbb{Z}^3 \mid x + y + z = 1\}$ and $S := \{(x, y, z) \in \mathbb{Z}^3 \mid xy = z\}$. Then it can be shown that any Diophantine equation can be represented using only the relations R, S over the integers, so that Hilbert's 10th problem is equivalent to $\mathsf{CSP}(\mathbb{Z}, R, S)$ [14].

In fact, it has been shown that every decision problem has a polynomial-time equivalent CSP over an infinite template [14].[3]

Again assuming PTIME $\neq$ NP, given a CSP over a template Γ, we say that it has a *dichotomy* if $\mathsf{CSP}(\Gamma)$ is either polynomially solvable or NP-complete. In fact, there are important classes for which such a dichotomy has been established; for example Boolean CSPs, that is, CSPs with domain $\{0, 1\}$ [42, 95] and CSPs for templates with three domain elements [28], as well as undirected graphs [59] (see the example below).

Example 3.2.14. Using a known result, a dichotomy can be obtained for $\mathsf{CSP}(\Gamma)$ if Γ ranges over the class of complete loopless graph on n vertices, for $n > 0$. So let $\Gamma = K_n = (V, E)$, where $V = \{1, \ldots, n\}$ and $E := \{(x, y) \in D^2 \mid x \neq y\}$. Notice that any vertex colouring of K_n must use at least n colours; recall Example 3.2.4. Now it suffices to note that, by a very old result, n-colouring is in PTIME if $n \leq 2$, and is NP-complete if $n \geq 3$ [64, 54]. Another example is $\mathsf{CSP}(H)$ from Example 3.2.4. Using a different result it can be shown that if H is either bipartite or has a loop, $\mathsf{CSP}(H)$ is polynomially solvable; otherwise it is NP-complete [59].

That a general dichotomy for finite templates holds was conjectured by Feder and Vardi in 1993.

Theorem 3.2.15. *([50], Conjecture 6)* $\mathsf{CSP}(\Gamma)$*, for all templates* Γ *with* $|\operatorname{dom}(\Gamma)| < \omega$*, is either in* PTIME *or* NP*-complete.*

The conjecture was confirmed recently [29, 100].

By Theorem 3.2.15, assuming PTIME $\neq$ NP, CSPs over finite templates consists of a large class of NP problems which are not NP-intermediate by Ladner's well-known theorem (see [69]). On the other hand, given that CSPs encode a very large class of natural computational problems, it is not surprising that such classifications are hard to obtain, as we will see next.

3.2.4 The algebraic approach to classifying CSPs

A PTIME/NP dichotomy does not exist for the class of infinite templates [14]. On the other hand, as we will see, dichotomies have been established for certain important templates used in applications in Artificial Intelligence.

When possible, given a template Γ, it has been shown useful to look into its algebraic structure, namely, associating an algebra to Γ in which to formulate sufficient and/or necessary conditions for tractability.

We refer the interested reader to [89] for a more detailed and didactic exploration of the general approach and [43], section 11.4, for a textbook treatment of the matter. Here

[3] Meaning there are poly-time Turing reductions between the two problems.

we briefly define some needed notions. Let A be a set. A *function clone* is a set $\mathcal{F}$ of operations of finite arity on A that is preserved by composition and that contains all projections. Formally, $\mathcal{F}$ has to satisfy the following conditions:

1. If an operation $f \in \mathcal{F}$ is n-ary and $g_1, \ldots, g_n \in \mathcal{F}$ are m-ary, then the m-ary operation $f(g_1, \ldots, g_n)$ defined by

$$(x_1, \ldots, x_m) \mapsto f(g_1(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m))$$

   is an element of $\mathcal{F}$;

2. For all $1 \leq k \leq n < \omega$, $\mathcal{F}$ contains the k-th n-ary projection $\pi_k^n \colon A^n \to A$, characterised uniquely by $\pi_k^n(x_1, \ldots, x_n) = x_k$.

A k-ary *polymorphism* of a structure Γ is simply a homomorphism from Γ^k to Γ itself, where $k \geq 1$. A *polymorphism clone* of a structure Γ is the set of all polymorphisms of Γ. Now to every template Γ one can associate a function clone $Pol(\Gamma)$ on $\mathrm{dom}(\Gamma)$. Recall that CSPs can be formulated as the homomorphism problem. Now it turns out that for some *finite* templates Γ, $\mathsf{CSP}(\Gamma)$ is determined by $Pol(\Gamma)$. We call this property the *polymorphism clone property*. For example, the complexity of CSP over the two element template Γ_{Bool} is completely determined by the polymorphism clone $\mathcal{F}$ of Γ_{Bool}: if Γ_{Bool} is preserved by one out of six operations in $\mathcal{F}$, then $\mathsf{CSP}(\Gamma_{Bool})$ is in PTIME; otherwise it is NP-complete [27]. We now exemplify a more complicated problem which cannot be formulated as a CSP over a finite template but is still amenable to algebraic methods; it will be used later:

Example 3.2.16. (Allen's Interval Algebra) Consider the basic binary relations on intervals $x = [x^-, x^+]$ with $x^- < x^+$ (the starting point is less then the end point), for $x^-, x^+ \in \mathbb{R}$. One can form 13 such relations, which are disjoint; they represent all the ways two intervals x and y can be "qualitatively" related. For instance, x *during* y is given by the relation defined by the constraints $x^- > y^-$, $x^- < y^+$, $x^+ > y^-$ and $x^+ < y^+$; see Table 3.1 for all basic relations. If we include the empty relation, taking the possible unions of basic relations gives us $2^{13} = 8192$ "complex" relations. Allen's Algebra $\mathcal{A}$ consists of the set containing those relations together with the operations $\cdot^{-1}, \cap$ and $\circ$, that is, inverse, intersection and composition, respectively. They are defined as follows:

- $\forall x, y \colon xr^{-1}x \iff yrx$,
- $\forall x, y \colon x(r \cap r')y \iff xry \wedge xr'y$,
- $\forall x, y \colon x(r \circ r')y \iff \exists z \colon (xrz \wedge zr'y)$.

Informally, $\mathcal{A}$-SAT is the problem to decide, given a set of variables over intervals in $\mathrm{dom}(\mathcal{A})$ with specified relations between them, whether there exists an assignment of real intervals to the variables that satisfies all such relations. This problem is NP-complete [99]. One way of obtaining tractable problems is to suitably specify subalgebras of $\mathcal{A}$. The

basic relation	in symbols	constraints
x before y y after x	$x \prec y$ $y \succ x$	$\{x^+ < y^-,\ x^- < y^+,$ $x^+ < y^-,\ x^+ < y^+\}$
x meets y y met-by x	xmy ym^-x	$\{x^- < y^-,\ x^- < y^+,$ $x^+ = y^-,\ x^+ < y^+\}$
x overlaps y y overlapped-by x	xoy yo^-x	$\{x^- < y^-,\ x^- < y^+,$ $x^+ > y^-,\ x^+ < y^+\}$
x during y y includes x	xdy yd^-x	$\{x^- > y^-,\ x^- < y^+,$ $x^+ > y^-,\ x^+ < y^+\}$
x starts y y started-by x	xsy ys^-x	$\{x^- = y^-,\ x^- < y^+,$ $x^+ > y^-,\ x^+ < y^+\}$
x finishes y y finished-by x	xfy yf^-x	$\{x^- > y^-,\ x^- < y^+,$ $x^+ > y^-,\ x^+ = y^+\}$
x equals y	$x \equiv y$	$\{x^- = y^-,\ x^- < y^+,$ $x^+ > y^-,\ x^+ = y^+\}$

Table 3.1: Allen's Interval Algebra: the 13 basic relations

dichotomy proved in [67] is in terms of subalgebras: 18 of them exhaust the tractable cases, while any subalgebra not entirely contained in one of them has an NP-complete problem. The *largest* tractable fragment of $\mathcal{A}$ is called ORD-Horn, and will be explored later. We will also see that $\mathcal{A}$-SAT and fragments can be formulated as CSPs.

Three different conditions on *countably infinite* templates, among others, have been proposed which are associated to amenability to tractability characterisations by means of polymorphism clones: homogeneity, ω-categoricity and a certain combinatorial property called the Ramsey property. We define the first two. Homogeneity is here used directly, and ω-categoricity is used in the dichotomy of temporal templates (see the next section). The Ramsey property is only mentioned as it is one of the key elements in dealing with $\mathsf{CSP}_c(\Gamma)$ as long as Γ is homogenous (see [23]).

Definition 3.2.17. Let Γ be a relational structure. Then Γ is said to be *homogeneous* if, for each pair of finite substructures Γ_1, Γ_2 of Γ, every isomorphism $\Gamma_1 \cong \Gamma_2$ can be extended to an automorphism of Γ.

Example 3.2.18. The Erdős-Rényi graph [47] and the structure $(\mathbb{Q}, <)$ are examples of homogeneous structures.[4]

Given a structure Γ, by $Th(\Gamma)$ we denote the first-order theory of Γ.

Definition 3.2.19. Let κ be a cardinal and Γ be a relational structure. Then we say that Γ is κ-categorical if all models of cardinality κ of $Th(\Gamma)$ are isomorphic to Γ.

[4] See also Example 5.3.8 for a description of classes of homogenous datatypes.

Linking the two properties we have the known fact that, given a relational structure Γ, Γ is homogenous only if Γ is ω-categorical, where ω is the countable infinite cardinal.

The starting point for the algebraic method for ω-categorical structures is a property that carries over from finite to a certain class of infinite templates. We say that a k-ary relation R has a primitive positive definition in a template Γ if there is a PP formula φ on k free variables that defines R. Now due to a result in universal algebra proved in [55] and [24], polymorphism clones characterise primitive positive definability. For infinite structures we have the following result:

Theorem 3.2.20. *([22]) Let Γ be an ω-categorical template. Then the relations preserved by Pol(Γ) are exactly those having a primitive positive definition in Γ.*

In other words, PP-definability of Γ, for Γ an ω-categorical template, is characterised by its polymorphism clone. Using Theorem 3.2.20 it can be shown that any chosen "intractable" relation R, that is, a relation encoding an NP-complete problem, captures NP-hardness of an ω-categorical template Γ. That is, it is shown that $\mathsf{CSP}(\Gamma)$ is intractable whenever there is a PP-definition of some relation with an NP-complete problem in Γ. For example, an argument on these lines was used:

- in Jeavon's algebraic proof of Schaeffer's classification of satisfiability problems, see [40];
- in the classification of the ω-categorical template $(\mathbb{Q}, <)$ presented in the next section.

Even though ω-categorical structures are amenable to the algebraic method, neither membership in NP, nor a general dichotomy have been proved for this class. Still, a certain subset of countable homogenous structures called *finitely constrained* are shown to be in NP and conjectured to have a dichotomy [22]. The structures in Example 3.2.18 have known dichotomies; in the next section we present in detail the second one.

3.3 Temporal CSPs

In the literature, CSPs where variables denote points in time and each constraint denotes a temporal relationship (e.g. an interval) are usually called *temporal* CSP*s* (see [93] for a survey). We illustrate such CSPs with a classical problem in temporal reasoning.

Example 3.3.1. In certain scheduling scenarios, we are given elements $j_1, \ldots, j_n$, which denote jobs, each to be performed at some point in time, and a list of constraints of the form $(\{j_{i_1}, \ldots, j_{i_m}\}, j_i)$ where $\{j_{i_1}, \ldots, j_{i_m}\} \subset \{j_1, \ldots, j_n\}$ and $j_i \notin \{j_{i_1}, \ldots, j_{i_m}\}$; meaning job j_i is to be done only after one of the jobs $j_{i_1}, \ldots, j_{i_m}$ has been completed. Such constraints are called *AND/OR precedence constraints* [82]. In particular, constraints of the form $(\{i\}, j)$ are called AND-constraints. The ones not of this form are called OR-constraints. The question is, then, whether there is an ordering of $j_1, \ldots, j_n$ such

that no constraint is violated. This feasibility problem, as we will see, can be modelled by temporal CSPs. For instance, let $n = 5$ and the set of constraints be

$$C = \{(\{j_2, j_3\}, j_1), (\{j_3, j_1\}, j_5), (\{j_1\}, j_4), (\{j_5\}, j_4)\}.$$

Then an example solution is the sequence $j_2 j_3 j_1 j_5 j_4$.

We now present the framework and notation used in the present section, which we adapt from [19].

Recall that a k-ary relation R has a first order definition in a structure $\mathcal{D}$ if there is a first-order formula $\varphi(x_1, \dots, x_k)$ over $\mathcal{D}$ such that

$$R := \{(a_1, \dots, a_k) \in (\mathrm{dom}(\mathcal{D}))^k \mid (\mathbb{Q}, <) \models \varphi(a_1, \dots, a_k)\}.$$

We then define:

Definition 3.3.2. A *temporal constraint language* is a relational structure $\Gamma := (\mathbb{Q}, R_1, R_2, \dots)$ where each R_i has a first-order definition in $(\mathbb{Q}, <)$.

By default it is assumed that the formulae defining such relations do not contain elements of $\mathbb{Q}$ as constants. We also say, whenever the property of first-order definability over $(\mathbb{Q}, <)$ holds of a given relation R, that R is a *temporal relation*.

Example 3.3.3. The structure $\Gamma := (\mathbb{Q}, R_{Betw}, R_{Disj}, R_{Cyc}, R_{Sep})$, where

$$\begin{aligned}
R_{Betw} &:= \{(a,b,c) \in \mathbb{Q}^3 \mid a < b < c \vee c < b < a\},\\
R_{Disj} &:= \{(a,b,c) \in \mathbb{Q}^3 \mid a < b \vee b < c\},\\
R_{Cyc} &:= \{(a,b,c) \in \mathbb{Q}^3 \mid a < b < c \vee c < a < b \vee b < c < a\},\\
R_{Sep} &:= \{(a,b,c,d) \in \mathbb{Q}^4 \mid (a < c < b < d) \vee (a < d < b < c) \vee (b < c < a < d)\\
&\qquad \vee (b < d < a < c) \vee (c < a < d < b) \vee (c < b < d < a)\\
&\qquad \vee (d < a < c < b) \vee (d < b < c < a)\}
\end{aligned}$$

is a temporal constraint language. The relations in Γ will be used later. In contrast, the structure $\Gamma' := (\mathbb{Q}, R_+)$ where

$$R_+ := \{(a,b,c) \in \mathbb{Q}^3 \mid a + b = c\},$$

is not a temporal constraint language. There are many ways to show this. One of them is as follows. Assume $\varphi(x, y, z)$ is a first-order definition of R_+ in $(\mathbb{Q}, <)$; we show that this leads to a contradiction. Consider any tuple $(a, b, c) \in R_+$; notice that $a = c - b$ holds by definition of R_+. Also $(\mathbb{Q}, <) \models \varphi(a, b, c)$. It is well known (see [39]) that, for any relational structure $\mathcal{D} = (D, R)$ and all formulae φ on free variables $x_1, \dots, x_n$, if $\beta \colon D \to D$ is an automorphism of $\mathcal{D}$, then

$$(\star)\ \mathcal{D} \models \varphi[x_1 \mapsto a_1, \dots, x_n \mapsto a_n] \text{ iff } \mathcal{D} \models \varphi[x_1 \mapsto \beta(a_1), \dots, x_n \mapsto \beta(a_n)],$$

where $a_j \in D$, $1 \leq j \leq n$. Let $\alpha\colon \mathbb{Q} \to \mathbb{Q}$ be an operation given by $\alpha(x) = x + \epsilon$ for ϵ a fixed non-zero rational number. It is readily checked that α is an automorphism of $(\mathbb{Q}, <)$. Using $(\star)$, we obtain

$$(\mathbb{Q}, <) \models \varphi[x \mapsto a, y \mapsto b, z \mapsto c] \text{ iff } (\mathbb{Q}, <) \models \varphi[x \mapsto \alpha(a), y \mapsto \alpha(b), z \mapsto \alpha(c)].$$

Then there is a tuple (a', b', c') with

$$\begin{aligned} \alpha(a) &= a' = a + \epsilon, \\ \alpha(b) &= b' = b + \epsilon, \\ \alpha(c) &= c' = c + \epsilon, \end{aligned}$$

for $0 \neq \epsilon \in \mathbb{Q}$, such that $(\mathbb{Q}, <) \models \varphi(a', b', c')$. Thus $(a', b', c') \in R_+$. By the definition of R_+, we have $a' + b' = c' = a + \epsilon + b + \epsilon = c + \epsilon$. Thus $a = c - b - \epsilon$. Contradiction.

The following example illustrates the basic framework for temporal constraint languages.

Example 3.3.4. Recall the problem and the problem instance described in Example 3.3.1. All precedence constraints are here relations defined on $(\mathbb{Q}, <)$; conjunctions of atoms $x < y$ are then used for modelling AND-constraints and formulae $x_1 < x_0 \vee \ldots \vee x_m < x_0$ for modelling OR-constraints. So we define the corresponding temporal CSP by setting $\Gamma := (\mathbb{Q}, R_\vee, R_\wedge)$, where

$$R_\vee := \{(a, b, c) \in \mathbb{Q}^3 \mid b < a \vee c < a\},\ R_\wedge := \{(a, b) \in \mathbb{Q}^2 \mid b < a\}$$

encode the types of constraints used in that particular problem. To conclude the encoding of the constraints considered, define the instance to $\mathsf{CSP}(\Gamma)$ to be

$$\varphi_C := \exists x_1, x_2, x_3, x_4, x_5 \Big(R_\vee(x_1, x_2, x_3) \wedge R_\vee(x_5, x_3, x_1) \wedge R_\wedge(x_4, x_1) \wedge R_\wedge(x_4, x_5) \Big).$$

Since we can find a solution in $\mathbb{Q}$, e.g., $x_2 \mapsto 1, x_3 \mapsto 2, x_1 \mapsto 3, x_5 \mapsto 4, x_4 \mapsto 5$ (the sequence $j_2 j_3 j_1 j_5 j_4$ in the original problem), we have $\varphi_C \in \mathsf{CSP}(\Gamma)$.

The algebraic structure of temporal templates

We now present some results on the structure underlying temporal CSPs and definitions used later for presenting complexity results.

The structure $(\mathbb{Q}, <)$ is ω-categorical [37, 60]. Now recall that the set of all automorphisms of a structure Γ forms a permutation group, which we denote $Aut(\mathbb{Q}, <)$. A relation $R \subseteq \mathbb{Q}^k$ is *preserved* by $Aut(\mathbb{Q}, <)$ if, for all $\alpha \in Aut(\mathbb{Q}, <)$, for all tuples $(a_1, \ldots, a_k) \in R$ we have $(\alpha(a_1), \ldots, \alpha(a_k)) \in R$.

The structure $(\mathbb{Q}, <)$ has nice properties, as a consequence of its ω-categoricity.

Theorem 3.3.5. *([60]) Let $R \subseteq \mathbb{Q}^k$ for $1 \leq k \in \mathbb{N}$. Then R has a first-order definition in $(\mathbb{Q}, <)$ if, and only if, R is preserved by $Aut(\mathbb{Q}, <)$.*

Corollary 3.3.6. *A relation $R \subseteq \mathbb{Q}^k$, $1 \leq k \in \mathbb{N}$, is temporal if, and only if, R is preserved by $Aut(\mathbb{Q}, <)$.*

Proof. We have by Theorem 3.3.5 that R is preserved by $Aut(\mathbb{Q}, <)$ iff R has a first-order definition over $(\mathbb{Q}, <)$ iff R is temporal (by definition of temporal relation). □

The result above is used (even if implicitly) in many proofs involving temporal relations.

The approach of [19] was described in the previous section. It consists in, given a temporal constraint language Γ, fully describing the complexity of $\mathsf{CSP}(\Gamma)$ in terms of polymorphism clones. In order to understand the results we need the following definitions.

Throughout this work, given a tuple $t = (a_1, \ldots, a_n)$ we will use $t[i]$ to denote the i-th entry of t.

Definition 3.3.7. Let $s, t \geq 0$ and $f \colon \mathbb{Q}^s \to \mathbb{Q}$.

- Given tuples $\bar{a}_1, \ldots, \bar{a}_s \in \mathbb{Q}^t$, we let $f(\bar{a}_1, \ldots, \bar{a}_s)$ denote the tuple $(b_1, \ldots, b_t) \in \mathbb{Q}^t$ with $b_i = f(\bar{a}_1[i], \ldots, \bar{a}_s[i])$.
- A set $R \subseteq \mathbb{Q}^t$ is *preserved by* f if for all $\bar{a}_1, \ldots, \bar{a}_s \in R$ we have $f(\bar{a}_1, \ldots, \bar{a}_s) \in R$.
- A temporal constraint language $\Gamma = (\mathbb{Q}, R_1, R_2, \ldots)$ is *preserved by* f if each relation R_i in Γ is preserved by f.

Recall the definition of a polymorphism from Section 3.2.4. This property is defined in terms of homomorphisms; i.e., informally, mappings that preserve relations. Let Γ be a relational structure. To illustrate polymorphisms, we inspect polymorphisms from Γ^2 to Γ; i.e., let $s = 2$. In order to obtain from t-tuples $\bar{a}_1, \bar{a}_2$ the t-tuple $\bar{b} = f(\bar{a}_1, \bar{a}_2)$ we apply f componentwise:

$$
\begin{array}{lccccc}
 & \overset{f}{\downarrow} & & \overset{f}{\downarrow} & & \overset{f}{\downarrow} \\
\bar{a}_1 = & (a_1[1], & & a_1[2], & \ldots, & a_1[t]) \\
\bar{a}_2 = & (a_2[1], & & a_2[2], & \ldots, & a_2[t]) \\
\bar{b} = & (f(a_1[1], a_2[1]), & & f(a_1[2], a_2[2]), & \ldots, & f(a_1[t], a_2[t]))
\end{array}
$$

We now define operations which are polymorphisms of $(\mathbb{Q}, <)$. Notice that they are not unique, i.e., they consist of arbitrary operations given by a number of conditions. They are relevant for the classification result in [19].

Definition 3.3.8 ([19])**.**

- $ll \colon \mathbb{Q}^2 \to \mathbb{Q}$, where $ll(a, b) < ll(a', b')$ iff
  - $a \leq 0$ and $a < a'$, or
  - $a \leq 0$ and $a = a'$ and $b < b'$, or
  - $a, a' > 0$ and $b < b'$, or
  - $a > 0$ and $b = b'$ and $a < a'$.

- $pp\colon \mathbb{Q}^2 \to \mathbb{Q}$, where $pp(a,b) \leq pp(a',b')$ iff
  - $a \leq 0$ and $a \leq a'$, or
  - $0 < a, 0 < a'$, and $b \leq b'$.
- $min\colon \mathbb{Q}^2 \to \mathbb{Q}$, where $min(a,b)$ is the minimum of a and b.
- $mi\colon \mathbb{Q}^2 \to \mathbb{Q}$, where

$$mi(a,b) := \begin{cases} \alpha(a), & \text{if } a < b \\ \beta(a), & \text{if } a = b \\ \gamma(b), & \text{if } a > b, \end{cases}$$

  where α, β, γ are unary operations that preserve $<$ such that $\beta(x) < \gamma(x) < \alpha(x) < \beta(x+\epsilon)$ for all $x \in \mathbb{Q}$ and all $\epsilon \in \mathbb{Q}$, $\epsilon > 0$.
- $mx\colon \mathbb{Q}^2 \to \mathbb{Q}$, where

$$mx(a,b) := \begin{cases} \alpha(min(a,b)), & \text{if } a \neq b \\ \beta(a), & \text{if } a = b, \end{cases}$$

  where α and β are unary operations that preserve $<$ such that $\alpha(x) < \beta(x) < \alpha(x+\epsilon)$ for all $x \in \mathbb{Q}$ and all $\epsilon \in \mathbb{Q}$, $\epsilon > 0$.
- The *dual* of an operation $f\colon \mathbb{Q}^k \to \mathbb{Q}$ is the operation $\bar{f}\colon \mathbb{Q}^k \to \mathbb{Q}$ defined by $\bar{f}(a_1, \ldots, a_k) := -f(-a_1, \ldots, -a_k)$.
- A *constant operation* is an operation $f\colon \mathbb{Q}^k \to \mathbb{Q}$, for some integer $k \geq 0$, for which there is a $b \in \mathbb{Q}$ with $f(a_1, \ldots, a_k) = b$ for all $a_1, \ldots, a_k \in \mathbb{Q}$.

Example 3.3.9. The relation

$$R_{min} := \{(a,b,c) \in \mathbb{Q}^3 \mid a > b \vee a > c\}$$

is preserved under ll. Assume $t_1 = (a, a', a'')$, $t_2 = (b, b', b'') \in R_{min}$. We show that $t_3 := ll((a,a',a''),(b,b',b'')) \in R_{min}$. It can be checked that we can assume $a > a'$ w.l.o.g., by symmetry of arguments. If (1) $b' \leq b$ we are done, for in this case $t_3[1] > t_3[2]$. Otherwise (2) $b' > b$, and given that $t_2 \in R_{min}$, we have $b > b''$. If (2.1) $a'' \leq a$ we are done, for then $t_3[3] \leq t_3[1]$ and so $t_3[3] < t_3[1]$ since ll is injective. Finally we have to check the case where (2.2) $a'' > a$. Now, if (2.2.1) $a' > 0$, by assumption we also have $a, a'' > 0$. Given that $b'' < b$, we have $t_3[1] > t_3[3]$. Otherwise (2.2.2) we have $a' \leq 0$. In that case it is easy to check that $t_3[1] > t_3[2]$ given that we have $a' \leq 0 \wedge a > a'$. In all cases above we obtain $t_3 \in R_{min}$. On the other hand, a similar argument shows that the relation

$$R_{max} := \{(a,b,c) \in \mathbb{Q}^3 \mid a < b \vee a < c\}$$

is preserved under dual-ll.

The relations above can be used for modelling AND/OR precedent constraints (see Example 3.3.1). On the other hand, the relation R_{Disj} defined in Example 3.3.3 is not preserved under any of the functions min, mi, mx, ll or their duals. To check this we provide counterexamples:

- Tuples $(0,1,0),\ (1,0,0) \in R_{Disj}$, but $min((0,1,0),(1,0,0)) \notin R_{Disj}$;
- Tuples $(0,1,1),\ (2,0,1) \in R_{Disj}$, but $mi((0,1,1),(2,0,1)) \notin R_{Disj}$;
- Tuples $(1,2,2),\ (2,1,1) \in R_{Disj}$, but $mx((1,2,2),(2,1,1)) \notin R_{Disj}$;
- Tuples $(2,0,1),\ (1,2,2) \in R_{Disj}$, but $ll((2,0,1),(1,2,2)) \notin R_{Disj}$.

Similar counterexamples can be provided for the duals of these functions.

The following result is well-known:

Theorem 3.3.10. *Let $\Gamma = (\mathbb{Q}, R_1, R_2, \dots)$ where each R_i is temporal. Then (1) $\mathsf{CSP}(\Gamma)$ is in* NP*. (2) If Γ contains all temporal relations, then $\mathsf{CSP}(\Gamma)$ is* NP*-hard.*

Proof. For (1) membership in NP, see Proposition 6 in [19]. For (2), assume Γ contains all temporal relations. Therefore Γ contains R_{Betw} and R_{Cyc}, see Example 3.3.3. Then the fact that NP-complete problems Betweenness [88] and Cyclic Ordering [53] can be formulated as CSPs using such relations shows that $\mathsf{CSP}(\Gamma)$ is NP-hard; see [19] for reductions. □

Now we are ready to present the temporal CSP dichotomy:

Theorem 3.3.11. *(Theorem 50 in [19]) A temporal constraint language Γ has a tractable CSP if Γ is preserved by at least one of the following operations: ll, min, mi, mx, their duals, or a constant operation; otherwise, $\mathsf{CSP}(\Gamma)$ is* NP*-complete.*

In the next chapter we address an issue with the proof of this theorem, correcting an error in the technical lemmata used.

3.4 Excursion on maximal tractable temporal languages

To close this chapter, we briefly investigate an important link between temporal CSPs and a classical result in temporal logic for AI.

Example 3.2.16 above introduces Allen's Interval Algebra. There are exactly 18 maximal tractable subalgebras, and any problem formulated in a fragment not contained in one of these subsets is NP-complete [67]. Earlier a famous fragment was presented which consists of the first example of a tractable Allen subalgebra, the so-called *ORD-Horn algebra*, denoted $\mathcal{H}$.

We introduce the essential notions. Since the basic relations of Allen's full Algebra $\mathcal{A}$ are disjoint, the intersection of any two relations $r_1, r_2 \in \mathcal{A}$, written $r_1 r_2$, consists of the set of *basic* relations in both r_1 and r_2. The composition $r_1 r_2$ is thus (as can be shown)

fully determined by the compositions of all basic relations in r_1 and r_2; table II in [67] can be used as a shortcut for the compositions of all pairs of basic relations.

Satisfiability for a fixed Allen interval subalgebra X is formally defined as follows.

$\mathcal{A}$-SAT(X)	
Input:	A set of variables V and a set of constraints I of the form xry where $x, y \in V$ and $r \in X$
Question:	Is there a function f from V to the set of all intervals such that $f(x)rf(y)$ holds for all $xry \in I$?

A given function f that satisfies all constraints in an instance I is called a *solution* for I (or a *model* of I, as in in [67]).

Example 3.4.1. ([67]) The instance $I = \{x(mo)y,\ y(df^{-1})z,\ x(m^{-1}s)z\}$ is satisfiable. The function given by $f(x) = [0, 2], f(y) = [1, 3]$ and $f(z) = [0, 4]$ is a solution for I.

If there is an algorithm running in polynomial time that solves all instances of $\mathcal{A}$-SAT(X), we say that such problem is tractable and, equivalently, that the subalgebra X is tractable. We now look at how results on the complexity of $\mathcal{A}$-SAT(X) relate to the temporal CSP dichotomy.

Recall Table 3.1. We use $\emptyset$ for the empty relation. Now for conciseness, by $\bar{r}^{+-}$ in a condition, for $\bar{r}$ an intersection of relations, we mean that both the whole condition with $\bar{r}$, and the whole condition with $\bar{r}^{-}$, should hold. So a condition written $(rs)^{+-} \subseteq \emptyset$ means that both $rs \subseteq \emptyset$ and $(rs)^{-} \subseteq \emptyset$ are required. It should be noted, though, that for any algebra $X \subseteq \mathcal{A}$, given a + and a − condition, either they are both simultaneously satisfied or both not satisfied. In Allen's framework, the subalgebra $\mathcal{H}$ can be defined as the set of all relations r such that the following conditions hold:

1. $r \cap (os)^{+-} \neq \emptyset$ & $r \cap (o^{-1}f)^{+-} \neq \emptyset \implies (d)^{+-} \subseteq r$;
2. $r \cap (ds)^{+-} \neq \emptyset$ & $r \cap (d^{-1}f^{-})^{+-} \neq \emptyset \implies (o)^{+-} \subseteq r$;
3. $r \cap (pm)^{+-} \neq \emptyset$ & $r \not\subseteq (pm)^{+-} \implies (o)^{+-} \subseteq r$.

Now, it turns out that $\mathcal{H}$ is the *largest* tractable fragment of $\mathcal{A}$ [85]; the result was obtained using an extensive computer-generated analysis. In fact Nebel and Bürckert had, previously to [67], defined a language which they call the *ORD-Horn subclass*. It is defined as follows. By an *interval formula* over an Allen algebra X we mean a formula of the form $x\bar{r}y$ where $\bar{r} = r_1 r_2 \dots r_n$ and $r_i \in X$. An ORD-clause is a disjunction of atoms of the form $x = y, x \leq y$ or $x \neq y$. If an interval formula φ is written as a conjunction of ORD-clauses then we say that φ is in ORD form. If every clause in φ contains at most one literal not of the form $x \neq y$, we say that φ is in *ORD-Horn form*. Then the ORD-Horn subclass is the subset of Allen's Interval Algebra all whose relations can be defined by interval formulae in ORD-Horn form. Now this subclass is equivalent to $\mathcal{H}$.

Remark 3.4.2. That $\mathcal{H}$ (see the conditions above) is equivalent to the ORD-Horn subclass is is not by any means obvious. An alternative equivalent formulation is the algebra of "pre-convex" relations defined by Ligozat (see Proposition 2 in [71]).

Similarly, by a conjunction of formulae of the form

$$(x_1 = y_1 \wedge \ldots \wedge x_{n-1} = y_{n-1}) \rightarrow x_n \circ y_n,$$

where $\circ \in \{=, \neq, \leq, <\}$, we denote an *ORD-Horn formula*. So $\mathcal{H}$ (equivalently, the ORD-Horn subclass) translates into a language $\Gamma_{\mathcal{H}} = (\mathbb{Q}, R_1, \ldots, R_n)$ where each R_i is definable by an ORD-Horn formula. Since satisfiability of propositional Horn clauses can be decided in polynomial time, by a simple reduction any temporal constraint language containing only ORD-Horn-definable relations is tractable (via Theorem 3.4 in [85]).

To show that maximal tractable interval subalgebras (when translated into the CSP framework) do not exhaust the tractable temporal cases one can use the following argument. Recall the operation *ll*, which by Theorem 3.3.11 ensures tractability of temporal constraint languages. It can be shown that all ORD-Horn languages are preserved under *ll* (Proposition 3.4 in [20]); thus, unsurprisingly, $\mathsf{CSP}(\Gamma_{\mathcal{H}})$ is tractable, for any "ORD-Horn language" $\Gamma_{\mathcal{H}}$. On the other hand, since $R_{min} := \{(a, b, c) \in \mathbb{Q}^3 \mid a > b \vee a > c\}$ is *ll*-closed [20] but not ORD-Horn-definable, that means there is a tractable temporal constraint language that strictly contains the ORD-Horn subclass (see the above-mentioned result by Nebel and Bürckert [85]). As can be seen, this is due to the more general way– in comparison to interval relations– in which relations are allowed to be defined in temporal constraint languages: namely, by means of first-order formulae over $(\mathbb{Q}, <)$ or $(\mathbb{R}, <)$. In addition, there is a temporal constraint language that strictly contains all AND-OR precedence constraints [82] (see Example 3.3.1); it suffices to note that they can be modelled using only "tractable relations" such as $R_{\vee} = R_{min}$ and $R_{\wedge}$ defined in Example 3.3.4.

4 Revisiting a result in temporal CSPs

4.1 Introduction

In this chapter we address an issue with the proof of Theorem 3.3.11 in [19], the main result in temporal CSPs that will be used in Chapter 6.

This theorem is proved through a series of technical lemmata. The full proof is beyond the scope of this thesis. Here, we focus on one of the technical lemmata, whose proof in [19] turns out to contain an error, and which we correct here. It makes use of the notion of "dominance by the i-th argument", which does not work as intended.

Our strategy is simply correcting the definition and then presenting reformulations and proofs when needed. Our main dichotomy result for OMQs over $(\mathbb{Q}, \leq)$ in Chapter 6 will then safely use the result.

We start by introducing required notions, and then we address the troubled definition by proposing a weaker one, fixing the lemmata affected by it. Finally, we show that one of the lemmata using the definition is not affected by it. We conclude this chapter with a remark on the proof of Theorem 3.3.11.

4.2 Required notions

We need some additional definitions and results for the discussion in this chapter.

Define $\mathbb{Q}^- := \{a \in \mathbb{Q} \mid a < 0\}$, $\mathbb{Q}_0^- := \{a \in \mathbb{Q} \mid a \leq 0\}$, $\mathbb{Q}^+ := \{a \in \mathbb{Q} \mid a \geq 0\}$. We say that an operation $f\colon D^k \to D$ is *interpolated* by a set of k-ary operations $\mathcal{F}$ if for every $A \subseteq D$, with A finite, there is some $g \in \mathcal{F}$ such that $f(a) = g(a)$ for all $a \in A^k$.

Definition 4.2.1. (Section 2.5 in [19])

- Let $\mathcal{F}$ be a set of k-ary operations on a set D. Then $\mathcal{F}$ *locally generates* an operation g on D if g is in the smallest function clone that is closed under interpolation and contains all operations in $\mathcal{F}$.
- Let $\mathcal{F}$ be a set of k-ary operations on $\mathbb{Q}$. Then $\mathcal{F}$ *generates* an operation g on $\mathbb{Q}$ if $\mathcal{F}$ together with all automorphisms $\alpha \in \mathit{Aut}(\mathbb{Q}, <)$ locally generates g. In particular, if $\mathcal{F}$ contains exactly one operation f, we say that f generates g.

Finally we state a lemma and define notions that are used for proving Theorem 4.5.2 and Lemma 4.5.3 below.

Lemma 4.2.2. *(Lemma 12 in [19]) An operation f generates an operation g if, and only if, every temporal relation preserved by f is also preserved by g.*

A set of the form $G = S_1 \times \ldots \times S_d$, for sets S_i, is called a *grid*. A subgrid $[k]^d$ of such a grid G is a subset of $S_1 \times \ldots \times S_d$ of the form $S'_1 \times \ldots \times S'_d$ where each S'_i is a k-element subset of S_i. We say that a k-ary operation f *behaves like a k-ary operation g* on a subgrid H of $\mathbb{Q}^k$ if for all tuples $t, t' \in H$ we have $f(t) \leq f(t')$ iff $g(t) \leq g(t')$.

Definition 4.2.3. (Definition 15 in [19]) Let f, g be binary operations on $\mathbb{Q}$. Then $[f|g]$ denotes some binary operation on $\mathbb{Q}$ such that for all $x, x', y, y' \in \mathbb{Q}$, the following holds:

- if $x \leq 0$ and $x' > 0$, then $[f|g](x, y) < [f|g](x', y')$;
- $[f|g]$ behaves like f on $\mathbb{Q}_0^- \times \mathbb{Q}$;
- $[f|g]$ behaves like g on $\mathbb{Q}^+ \times \mathbb{Q}$.

4.3 The notion and use of 'dominance by the i-th argument'

A certain notion of dominance is used in a crucial way in key lemmata used in the proof of the dychotomy theorem. We will go into such lemmata shortly. Before we do that, we introduce this notion and show why it is problematic. In Section 6.1, the authors first define a set of operations and then introduce the definition of 'dominance by the i-th argument' as follows:

Definition 4.3.1. (Section 6.1. [19]) Let $i \in \{1, \ldots, k\}$. A k-ary operation $f \colon \mathbb{Q}^k \to \mathbb{Q}$ is *dominated by the i-th argument* if for all $a_1, \ldots, a_k, b_1, \ldots, b_k \in \mathbb{Q}$, we have $f(a_1, \ldots, a_k) \leq f(b_1, \ldots, b_k)$ if, and only if, $a_i \leq b_i$.

Now $lex \colon \mathbb{Q}^2 \to \mathbb{Q}$ is an operation such that $lex(a, b) < lex(a', b')$ if either $a < a'$, or $a = a'$ and $b < b'$. The operations mentioned above are the following:

- $lex_{y,x}$ for the operation $(x, y) \mapsto lex(y, x)$,
- $lex_{y,-x}$ for the operation $(x, y) \mapsto lex(y, -x)$,
- $lex_{x,-y}$ for the operation $(x, y) \mapsto lex(x, -y)$,
- $lex_{x,y}$ for the operation $(x, y) \mapsto lex(x, y)$,
- p_x for the operation $(x, y) \mapsto x$,
- p_y for the operation $(x, y) \mapsto y$.

The authors give lex (i.e. $lex_{x,y}$) as an example of an operation dominated by the first argument. This is not true, as the following argument shows. Assume, for the sake of obtaining a contradiction, that lex is dominated by the first argument as per Definition 4.3.1. Now consider any two pairs (a, b), (a', b') of rational numbers such that $a = b = a'$ and $b > b'$. The definition of lex immediately yields $lex(a, b) > lex(a', b')$. Since $a \leq a'$ holds, by assumption that lex is dominated by its first argument, we have $lex(a, b) \leq lex(a', b')$. Contradiction.

Basically the same argument shows that assuming $lex_{y,x}$ is dominated by the second argument will lead to the same error. On the other hand, the notion works as intended for the operations p_x, p_y, which are dominated by the first and the second argument, respectively. Actually (here for $1 \leq i \leq 2$) we have that $f\colon \mathbb{Q}^2 \to \mathbb{Q}$ is dominated by the i-th argument if, and only if, f behaves like p_x if $i = 1$ (p_y if $i = 2$).

A close inspection of the proofs of certain lemmata in [19] reveal that the notion of dominance, that is, Definition 4.3.1, is used in a crucial way. Each of these lemmata are used to prove the main result, Theorem 3.3.11 above.

In order to fix those lemmata (which we will indicate in the next subsection) and their proofs we introduce the following notion:

Definition 4.3.2. Let $i \in \{1, \ldots, k\}$. A k-ary operation $f\colon \mathbb{Q}^k \to \mathbb{Q}$ is *weakly dominated* by the i-th argument if the following holds for all rational numbers $a_1, \ldots, a_k, b_1, \ldots, b_k$:

1. If $f(a_1, \ldots, a_k) \leq f(b_1, \ldots, b_k)$, then $a_i \leq b_i$.
2. If $a_i < b_i$, then $f(a_1, \ldots, a_k) < f(b_1, \ldots, b_k)$.

The new notion captures the desired distinction:

Proposition 4.3.3. *Let $\mathcal{F}_1 := \{lex_{x,y}, p_x, lex_{x,-y}\}$ and $\mathcal{F}_2 := \{lex_{y,x}, p_y, lex_{y,-x}\}$. Also let $i \in \{1, 2\}$. Then for all $f \in \mathcal{F}_i$, f is weakly dominated by the i-th argument.*

Proof. We need to show that Definition 4.3.2 applies.

We show the first part (1). By contraposition, assume it is not the case that $a \leq a'$, i.e., $a > a'$. First, let $f \in \mathcal{F}_1$. For lex_{xy}, by definition of this function we obtain that $lex_{xy}(a', b') < lex_{x,y}(a, b)$, since $lex(a', b') < lex(a, b)$ For lex_{x-y}, by definition of this function we get $lex_{x-y}(a', -b') < lex_{x-y}(a, -b)$, since $lex(a', b') < lex(a, b)$. Similarly for p_x. By exchanging a and b, a' and b' (that is, assuming $b > b'$) we verify the case where $f \in \mathcal{F}_2$.

Now for (2) of Definition 4.3.1. assume $a < a'$ and consider first $f \in \mathcal{F}_1$. For lex_{xy} by definition we get $lex_{xy}(a, b) < lex_{xy}(a', b')$. For lex_{x-y} by definition we have $lex(a, -b) < lex_{x-y}(a', -b')$. Now consider $f \in \mathcal{F}_2$ and assume accordingly that $b < b'$. This yields $lex_{yx}(a, b) < lex_{yx}(a', b')$ (by definition of lex_{yx}), since $lex(b, a) < lex(b', a')$; and $lex_{y-x}(-a, b) < lex_{y-x}(-a', b')$ (by definition of lex_{y-x}), since $lex(b, a) < lex(b', a')$, as desired. Similarly for p_y.

Since both parts of the Definition have been verified, it follows that all functions in $\mathcal{F}_1$ are weakly dominated by the first argument and all functions in $\mathcal{F}_2$ are weakly dominated by the second argument.

□

4.4 The affected lemmata

We now fix the lemmata that are affected by the troubled definition. Note that Theorem 3.3.11 uses the lemma below in the proof presented by [19]:

Lemma 4.4.1. *(Lemma 49 in [19]) Let f be a binary operation that preserves $<$ but not R_{Betw}. Then f generates ll, dual-ll, pp, or dual-pp.*

This lemma makes essential use of lemmata 4.4.2 and 4.4.3 below, both of which depend on a working definition of "dominance by the i-th argument". We now proceed to check and fix them when needed—by providing detailed proofs that employ, basically, the same strategy as the authors'.

We slightly rewrite the corresponding lemma in [19], replacing the notion introduced in Definition 4.3.1 by the one introduced in Definition 4.3.2, and provide a detailed proof thereof:

Lemma 4.4.2. *(Reformulation of Lemma 43 in [19]) Let*

$$f, g \in \{lex_{x,y}, lex_{x,-y}, lex_{y,x}, lex_{y,-x}, p_x, p_y\},$$

and let f' (g') be $lex_{x,y}$ if f (g) is weakly dominated by the first argument, and $lex_{y,x}$ otherwise. Then $\{lex, [f|g]\}$ generates $[f'|g']$.

Proof. By Lemma 4.2.2 it suffices to show that $[f'|g']$ preserves every relation that is preserved by $\{lex, [f|g]\}$. So let R be a k-ary relation preserved by $\{lex, [f|g]\}$, and let $t_1, t_2 \in R$. We show that $t_3 := [f'|g'](t_1, t_2) \in R$.

Using the same strategy as in [19], we pick $\alpha \in Aut(\mathbb{Q}, <)$ such that for all $t[i]$ and all $t'[j]$ with $t[i] \leq 0$, α maps $lex(t[i], t'[j])$ to a negative value; otherwise α maps $lex(t[i], t'[j])$ to a positive value. Let $s := [f|g](\alpha(lex(t_1, t_2)), lex(t_2, t_1))$. We show that there is an automorphism $\beta \in Aut(\mathbb{Q})$ such that $\beta(s) = t_3$, which shows $t_3 \in R$. By symmetry it is enough to show for $j_1, j_2 \in [k]$ with $t_1[j_1] \leq t_1[j_2]$ that ($\star$) $s[j_1] \leq s[j_2]$ iff $t_3[j_1] \leq t_3[j_2]$. We distinguish three cases:

- $t_1[j_1] \leq 0,\ t_1[j_2] > 0$. By construction, $\alpha(lex(t_1[j_1], t_2[j_1])) \leq 0$ given that $t_1[j_1] \leq 0$ and $\alpha(lex(t_1[j_2], t_2[j_2])) > 0$ since $t_1[j_2] > 0$. Thus by Definition 4.2.3

  $$s[j_1] = [f|g](\alpha(lex(t_1[j_1], t_2[j_1]))) < [f|g](\alpha(lex(t_1[j_2], t_2[j_2]))) = s[j_2].$$

  Also by assumption and again directly by Definition 4.2.3 we get

  $$t_3[j_1] = [f'|g'](t_1[j_1], t_2[j_1]) < [f'|g'](t_1[j_2], t_2[j_2]) = t_3[j_2].$$

  Thus ($\star$) holds.

- $t_1[j_2] \leq 0$. We first show that $f'(x, y)$ behaves like $f(lex(x, y), lex(y, x))$, i.e., for all $x, x', y, y' \in \mathbb{Q}$:

  $$\dagger\ f'(x, y) \leq f'(x', y') \text{ iff } f(lex(x, y), lex(y, x)) \leq f(lex(x', y'), lex(y', x')).$$

  So assume $f'(x, y) \leq f'(x', y')$. If $f' = lex_{x,y}$, by Definition 4.3.2 $x \leq x'$, so $f(lex(x, y), lex(y, x)) \leq f(lex(x', y'), lex(y', x'))$. If $f' = lex_{y,x}$, by Definition 4.3.2 $y \leq y'$, so $f(lex(x, y), lex(y, x)) \leq f(lex(x', y'), lex(y', x'))$. For the other direction,

assume $f(lex(x,y), lex(y,x)) \leq f(lex(x',y'), lex(y',x'))$. If $f = lex_{x,y}$, by Definition 4.3.2 $lex(x,y) \leq lex(x',y')$ so $x \leq x'$, thus $f'(x,y) \leq f'(x',y')$ by assumption. If $f = lex_{y,x}$, by Definition 4.3.2 $lex(y,x) \leq lex(y',x')$ so $y \leq y'$, which yields $f'(x,y) \leq f'(x',y')$ by assumption. Similarly for the remaining functions. So (†) holds. Now

$$\begin{aligned}
t_3[j_1] \leq t_3[j_2] &\Leftrightarrow [f'|g'](t_1[j_1], t_2[j_1]) \leq [f'|g'](t_1[j_2], t_2[j_2]) \\
&\Leftrightarrow f'(t_1[j_1], t_2[j_1]) \leq f'(t_1[j_2], t_2[j_2]) \\
&\text{(since } t_1[j_1] \leq t_1[j_2] \leq 0, [f'|g'] \text{ behaves like } f' \text{ on } \mathbb{Q}_0^- \times \mathbb{Q}) \\
&\Leftrightarrow f(lex(t_1[j_1], t_2[j_1]), lex(t_2[j_1], t_1[j_1])) \\
&\quad \leq f(lex(t_1[j_2], t_2[j_2]), lex(t_2[j_2], t_1[j_2])) \text{ (using †)} \\
&\Leftrightarrow f(\alpha(lex(t_1[j_1], t_2[j_1]), lex(t_2[j_1], t_1[j_1]))) \\
&\quad \leq f(\alpha(lex(t_1[j_2], t_2[j_2]), lex(t_2[j_2], t_1[j_2]))) \\
&\Leftrightarrow s[j_1] \leq s[j_2],
\end{aligned}$$

which shows (⋆).

- $t_1[j_1] > 0$. Analogous to the case above.

□

We now fix the proof of the following lemma, using Definition 4.3.2:

Lemma 4.4.3. *(Lemma 44 in [19]) For $f, g \in \{p_y, lex_{y,x}\}$, the operation $[f|g]$ generates $[lex_{x,y}|g]$.*

Proof. Again we use the same strategy as in [19]. To obtain that $[f|g]$ generates $[lex_{x,y}|g]$, by Lemma 4.2.2 it suffices to show that $[lex_{x,y}|g]$ preserves every relation that is preserved by $[f|g]$. So let R be a k-ary relation preserved by $[f|g]$, and let $t_1, t_2 \in R$. We show that $t_3 := [lex_{x,y}|g](t_1, t_2) \in R$.

Now let ℓ be the number of values of t_1 in $\mathbb{Q}_0^-$, and $a_1, a_2, \ldots, a_\ell, a_{\ell+1}, \ldots, a_m$ be an enumeration of all its k values in ascending order. We pick $\alpha_i \in Aut(\mathbb{Q}, <)$ such that each α_i, $1 \leq i \leq \ell$, maps all but the i-th smallest values of t_1 to positive non-zero values, as in the table below.

	a_1	a_2	a_3	$\ldots$	a_ℓ	$a_{\ell+1}$	$\ldots$	a_m
t_1	≤ 0	≤ 0	≤ 0	$\ldots$	≤ 0	> 0	$\ldots$	> 0
$\alpha_1(t_1)$	$\mathbf{\leq 0}$	> 0	> 0	$\ldots$	> 0	> 0	$\ldots$	> 0
$\alpha_2(t_1)$	≤ 0	$\mathbf{\leq 0}$	> 0	$\ldots$	> 0	> 0	$\ldots$	> 0
$\alpha_3(t_1)$	≤ 0	≤ 0	$\mathbf{\leq 0}$	$\ldots$	> 0	> 0	$\ldots$	> 0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\alpha_\ell(t_1)$	≤ 0	≤ 0	≤ 0	$\ldots$	$\mathbf{\leq 0}$	> 0	$\ldots$	> 0

Define the sequence of tuples $s_1, s_2, \ldots, s_\ell$ where $s_1 = t_2$ and $s_i := [f|g](\alpha_i(t_1), s_{i-1})$ for $i \geq 2$. Because R is preserved by $Aut(\mathbb{Q}, <)$ and by $[f|g]$, we have that $s_i \in R$ for all

$i \in [\ell]$. We claim that there is an automorphism $\alpha' \in Aut(\mathbb{Q}, <)$ such that $\alpha'(s_\ell) = t_3$, so that $t_3 \in R$. By symmetry it is enough to show for $j_1, j_2 \in [k]$ with $t_1[j_1] \leq t_1[j_2]$ that ($\star$) $s_\ell[j_1] \leq s_\ell[j_2]$ iff $t_3[j_1] \leq t_3[j_2]$. There are three cases to consider:

1. $t_1[j_1] = t_1[j_2] \leq 0$. Therefore $\alpha_i(t_1[j_1]) = \alpha_i(t_1[j_2])$ for all $i \in [\ell]$. Now since $t_1[j_1] = t_1[j_2]$ it is readily checked that $t_2[j_1] = t_2[j_2]$ iff $s_\ell[j_1] = s_\ell[j_2]$. On the other hand, since f is weakly dominated by the second argument, Definition 4.3.2 gives

$$\begin{aligned} t_2[j_1] < t_2[j_2] &\Leftrightarrow [f|g](\alpha_i(t_1[j_1]), s_{i-1}[j_1]) < [f|g](\alpha_i(t_1[j_2]), s_{i-1}[j_2]) \text{ for all } i \\ &\Leftrightarrow s_\ell[j_1] < s_\ell[j_2]. \end{aligned}$$

   Altogether (that is, for $t_2[j_1] = t_2[j_2]$ and $t_2[j_1] < t_2[j_2]$) we obtain equivalently $s_\ell[j_1] \leq s_\ell[j_2]$. Also (using the fact that $t_1[j_1] = t_1[j_2]$)

$$\begin{aligned} t_3[j_1] \leq t_3[j_2] &\Leftrightarrow [lex_{x,y}|g](t_1[j_1], t_2[j_1]) \leq [lex_{x,y}|g](t_1[j_2], t_2[j_2]) \\ &\Leftrightarrow lex_{x,y}(t_1[j_1], t_2[j_1]) \leq lex_{x,y}(t_1[j_2], t_2[j_2]) \\ &\Leftrightarrow t_2[j_1] \leq t_2[j_2] \\ &\Leftrightarrow s_1[j_1] \leq s_1[j_2] \text{ because } s_1 = t_2 \\ &\Leftrightarrow s_\ell[j_1] \leq s_\ell[j_2]; \end{aligned}$$

   hence ($\star$) holds.

2. $t_1[j_1] < t_1[j_2], t_1[j_1] \leq 0$. By the construction of the automorphisms (see the table above) we can pick $\alpha_i \in Aut(\mathbb{Q}, <)$, $1 \leq i \leq \ell$, such that $\alpha_i(t_1[j_1]) \leq 0$ and $\alpha_i(t_1[j_2]) > 0$. By Definition 4.2.3, we obtain $f(\alpha_i(t_1[j_1]), s_{i-1}[j_1]) < f(\alpha_i(t_1[j_2]), s_{i-1}[j_2])$, so $s_i[j_1] < s_i[j_2]$. Given that $[f|g]$ preserves $<$, a simple induction on $n \geq i$ shows that $s_\ell[j_1] < s_\ell[j_2]$. On the other hand from the assumption that $t_1[j_1] < t_1[j_2], t_1[j_1] \leq 0$ we obtain $t_3[j_1] = \mathfrak{h}(t_1[j_1], t_2[j_1]) < \mathfrak{h}(t_1[j_2], t_2[j_2]) = t_3[j_2]$ with $\mathfrak{h} \in \{lex_{x,y}, [lex_{x,y}|g]\}$, using the definition of $\mathfrak{h}$. Altogether we obtain ($\star$).

3. $t_1[j_1] > 0$. Then by construction $\alpha_i(t_1[j_1]) > 0$ for all $i \in [\ell]$. Let $\mathfrak{h} \in \{[f|g], [lex_{x,y}|g]\}$. Notice that by Definition 4.2.3, $\mathfrak{h}$ behaves like g on $\mathbb{Q}^+ \times \mathbb{Q}$. Thus

$$\begin{aligned} t_3[j_1] \leq t_3[j_2] &\Leftrightarrow \mathfrak{h}(t_1[j_1], t_2[j_1]) \leq \mathfrak{h}(t_1[j_2]), t_2[j_2]) \\ &\Leftrightarrow \mathfrak{h}(\alpha_2(t_1[j_1]), s_1[j_1]) \leq \mathfrak{h}(\alpha_2(t_1[j_2]), s_1[j_2]) \text{ because } s_1 = t_2 \\ &\Leftrightarrow \mathfrak{h}(\alpha_3(t_1[j_1]), s_2[j_1]) \leq \mathfrak{h}(\alpha_3(t_1[j_2]), s_2[j_2]) \text{ again by definition of } s_2 \\ &\vdots \\ &\Leftrightarrow \mathfrak{h}(\alpha_\ell(t_1[j_1]), s_{\ell-1}[j_1]) \leq \mathfrak{h}(\alpha_\ell(t_1[j_2]), s_{\ell-1}[j_2]) \\ &\Leftrightarrow s_\ell[j_1] \leq s_\ell[j_2], \end{aligned}$$

   which establishes ($\star$) as desired.

□

4.5 A safe use of dominance. Conclusion

Only Lemma 4.4.1 (Lemma 49 in [19]) remains to be checked. As it turns out, it uses dominance in a safe way, as we point out briefly in this section. The following results are used in our remark:

Theorem 4.5.1. *(Theorem 20 in [19]) Let Γ be a temporal constraint language. Then it satisfies at least one of the following properties:*

a. *There is a PP-definition of R_{Betw}, R_{Cyc} or R_{Sep} in Γ.*

b. *$Pol(\Gamma)$ contains a constant operation.*

c. *$Aut(\Gamma)$ contains all permutations of $(\mathbb{Q}, <)$.*

d. *There is a PP-definition of $<$ in Γ. Moreover, Γ contains a binary operation that does not preserve R_{Betw}.*

Theorem 4.5.2. *(Product Ramsey Theorem; slightly rephrased from [19]) For all positive integers d, r, m and $k \geq m$, there is a positive integer $L = \mathbf{R}(d, r, m, k)$ such that for every mapping $f\colon S \to [r]$ of the $[m]^d$ subgrids of $[L]^d$, there exists a $[k]^d$ subgrid G of $[L]^d$ such that f is constant on G.*

Lemma 4.5.3. *(Lemma 47 in [19]) Let f be a binary operation that preserves $<$, and let $S_1, T_1 \subseteq \mathbb{Q}$ be sets of cardinality at least $\mathbf{R}(k)$. Then there exist sets $S_2 \subset S_1, T_2 \subset T_1$ of cardinality k such that f behaves on $S_2 \times T_2$ like one out of the following operations:*

- *p_x or p_y;*
- *$lex_{x,y}$ or $lex_{y,x}$;*
- *$lex_{x,-y}$ or $lex_{y,-x}$.*

Lemma 4.5.4. *(Lemma 48 in [19]) Let f be a binary operation that does not preserve R_{Betw} and preserves $<$. Then there are $t_1, t_2 \in R_{Betw}$ such that (t_1, t_2) has three distinct entries and $f(t_1, t_2) \notin R_{Betw}$.*

The authors' strategy for proving Lemma 4.4.1 is to start with the fact (from Lemma 4.5.4) that there are tuples $t_1, t_2 \in R_{Betw}$ such that $t = f(t_1, t_2) \notin R_{Betw}$ and f is injective and assuming w.l.o.g. that $t_1[1] < t_1[2] < t_1[3]$ and $t_2[1] > t_2[2] > t_2[3]$. Then grids can be defined from intervals between certain positions in t_1 and t_2 so that, by the Product Ramsey Theorem (Theorem 4.5.2) via Lemma 4.5.3, f behaves on those grids like one of the versions of *lex* and projections. It can be shown then that f generates the functions lex, ll, pp or their duals, depending on whether it is weakly dominated by the first or by the second argument, by lemmata 4.4.2 and 4.4.3. It can be checked that nothing need to be changed in the proof presented in [19].

Lemma 4.4.1 guarantees that operations that do not preserve R_{Betw} but preserve $<$ generate "tractable operations". Using Theorem 3.2.20, it is shown that R_{Betw} captures

NP-hardness via PP-definability of temporal relations in $\Gamma_{Betw} := (\mathbb{Q}, R_{Betw})$. Finally, Theorem 4.5.1 is invoked in Theorem 3.3.11 for classifying the complexity of Γ depending on its polymorphism clone. Its third case, namely, that $<$ has a PP-definition in Γ and Γ contains a binary operation that does not preserve R_{Betw}, follows from Lemma 4.4.1. Theorem 3.3.11 thus follows from the corrected lemmata.

5 Query answering and CSPs

5.1 Introduction

In this chapter, we introduce the machinery for attacking the complexity of the query answering problem for a very broad class of OMQs. First we show negative results: that query answering becomes undecidable for many important datatypes, including $(\mathbb{Q}, <)$, $(\mathbb{Z}, <)$ and $(\mathbb{Z}, \leq)$. In order to ensure decidability we require that OMQs satisfy the bounded match depth property (BMDP), which ensures that answers can be determined based on a bounded subset of the chase of a Horn-$\mathcal{L}(\mathcal{D})$ KB. We then introduce the main result of this chapter, which is a general framework for transfering complexity classification results from CSPs to OMQ answering.

5.2 Undecidability results

We show that answering Boolean OMQs over simple and relevant datatypes $\mathcal{D}$, due to the (possibly) very complicated structure of universal pre-models is undecidable already for a subset of Horn-$\mathcal{L}(\mathcal{D})$, i.e., $\mathit{DL\text{-}Lite}_{\mathcal{R}}^{attrib}(\mathcal{D})$.

The strategy is first proving that the problem for (1) $\mathcal{D} \in \{(\mathbb{Z}, \neq), (\mathbb{Q}, \neq)\}$ is undecidable, and then reduce it to query evaluation over (2) $\mathcal{D}' \in \{(\mathbb{Z}, <), (\mathbb{Z}, \leq), (\mathbb{Q}, <)\}$. It is an open problem whether OMQ answering over $(\mathbb{Q}, \leq)$ is decidable or not. These items will be discussed in different sections.

5.2.1 Numerical datatypes with inequality

Our proof of undecidability for OMQ answering over $\mathcal{D} \in \{(\mathbb{Z}, \neq), (\mathbb{Q}, \neq)\}$ is based on a reduction from the problem of unrestricted tiling of the discrete plane. We present all the details of the construction as it provides insight into the very intrincate structure of completions of canonical pre-models. Intuitively the problem is as follows. A tile type is a unit square divided into four triangles by means of two diagnonals. Each triangle is coloured using exactly one colour out of a set of m colours. Tiles cannot be rotated or deflected. We say that two tile types are horizontally (vertically) compatible if they can be put one beside (above) the other with matching colours. Tiling the plane means correctly placing tiles on the discrete plane, that is, generating a suitable mapping from the infinite $\mathbb{N} \times \mathbb{N}$ grid to tile types. More concretely, a solution is a mapping where each two adjacent tiles are both horizontally and vertically compatible. The problem asks whether there exists a tiling of $\mathbb{N} \times \mathbb{N}$ given a set of n m-coloured tile types. This problem is undecidable [11]. We introduce the formal definition with which we will work.

Unrestricted $\mathbb{N} \times \mathbb{N}$ tiling	
Input:	A set of tile types $\mathfrak{T} := \{T_1, \dots, T_n\}$ with colours $\mathsf{up}(T_i)$, $\mathsf{down}(T_i)$, $\mathsf{right}(T_i)$, $\mathsf{left}(T_i)$, for $1 \leq i \leq n$.
Question:	Does there exist a function $f\colon \mathbb{N} \times \mathbb{N} \to \mathfrak{T}$ such that $\mathsf{right}(f(i,j)) = \mathsf{left}(f(i{+}1, j))$ and $\mathsf{up}(f(i,j)) = \mathsf{down}(f(i, j{+}1))$, for all $i, j \geq 0$?

The reduction is done so that given a tiling problem $\mathfrak{T}$, we can construct a Boolean OMQ $Q_\mathfrak{T} = (\mathcal{T}, q_\mathfrak{T})$, over datatype $\mathcal{D} \in \{(\mathbb{Z}, \neq), (\mathbb{Q}, \neq)\}$, where $\mathcal{T}$ is a TBox in a proper subset of Horn-$\mathcal{ALCHI}^{attrib}$ and $q_\mathfrak{T}$ is a UCQ, and a $\mathcal{D}$-ABox $\mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}) \models q_\mathfrak{T}$ iff $\mathfrak{T}$ does not tile $\mathbb{N} \times \mathbb{N}$.

Obtaining binary tree-shaped pre-models Let $\mathcal{D} \in \{(\mathbb{Z}, \neq), (\mathbb{Q}, \neq)\}$, v, h be role names, X be a concept name and U, U_k^i, for $1 \leq i \leq 2$ and $1 \leq k \leq n$, be attribute names. Define $\mathcal{T} = \{C \sqsubseteq \exists h, C \sqsubseteq \exists v, C \sqsubseteq U_i^k\}$, where $C \in \{X, \exists v^-, \exists h^-\}$. Also let $\mathcal{A} = \{X(a)\}$. Now consider the canonical pre-model of $(\mathcal{T}, \mathcal{A})$. The result is an infinite binary tree with "horizontal" edges h and "vertical" edges v, having a as its root. Each node c in the tree contains assertions $U(c, u)$ and $U_k^i(c, u_k^i)$ for u, u_k^i data nulls.

Defining a quotient structure for each completion f of $\mathrm{can}(\mathcal{T}, \mathcal{A})$ We introduce a few definitions from abstract algebra. Let $\mathcal{R}$ be an equivalence relation on a set A. For all $a \in A$ let $\|a\|_\mathcal{R}$ be the equivalence class of a. Then the *quotient of A modulo $\mathcal{R}$*, written $A/\mathcal{R}$, is the set $\{\|a\|_\mathcal{R} \mid a \in A\}$. Now let $\mathcal{B} = (A, R_1, R_2, \dots)$ be a relational structure, with R_i a k_i-ary relation, and $\mathcal{R}$ an equivalence relation on $\mathcal{B}$. Also let $R^{A/\mathcal{R}}$ be the relation induced on $A/\mathcal{R}$ by R, that is

$$R^{A/\mathcal{R}} := \{(\|a_1\|_\mathcal{R}, \dots, \|a_k\|_\mathcal{R}) \in (A/\mathcal{R})^k \mid (b_1, \dots, b_k) \in R \text{ with } b_i \in \|a_i\|_\mathcal{R}\}.$$

Then $\mathcal{B}/\mathcal{R} = (A/\mathcal{R}, R^{A/\mathcal{R}})$ is called the *quotient structure of $\mathcal{B}$ under $\mathcal{R}$*.

Let f be a completion function for $\mathrm{can}(\mathcal{T}, \mathcal{A})$. We now set

$$\sim_f := \{(c, d) \mid c, d \in \mathrm{dom}(f(\mathrm{can}(\mathcal{T}, \mathcal{A}))) \text{ with } \{(c, u), (d, u)\} \subseteq U^{f(\mathrm{can}(\mathcal{T}, \mathcal{A}))}\}.$$

It can be checked that $(x \sim y)$ defines a natural equivalence relation on any completion $f(\mathrm{can}(\mathcal{T}, \mathcal{A}))$. Then $f(\mathrm{can}(\mathcal{T}, \mathcal{A}))/\sim_f$ is the quotient structure of $f(\mathrm{can}(\mathcal{T}, \mathcal{A}))$ under $\sim_f$. We write $\not\sim_f$ for the complementary relation. Equivalence classes $||d||_f$ in a quotient structure $f(\mathrm{can}(\mathcal{T}, \mathcal{A}))/\sim_f$ are here called *nodes*.

The same symbols (without the subscript) are used as syntactic abbreviations in queries:

- $(x \sim y) = U(x, z_{x,y}) \wedge U(y, z_{x,y})$;
- $(x \not\sim y) = U(x, z_x) \wedge U(y, z_y) \wedge z_x \neq z_y$.

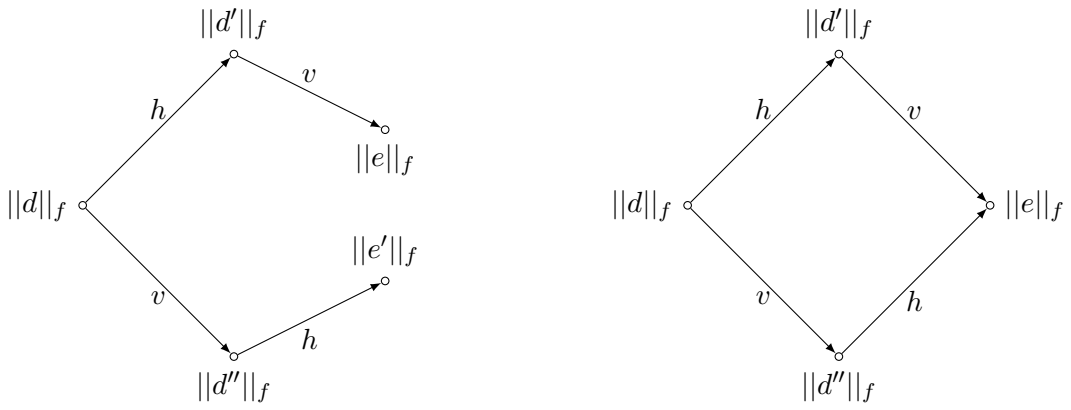

Figure 5.1: Open (left) and closed (right) cells on a grid

Construction of $q_{\mathfrak{T}}$ as a set of constraints simulating $\mathfrak{T}$ We now construct the Boolean UCQ $q_{\mathfrak{T}}$ in such a way as to simulate, using $\mathcal{T}$ and $\mathcal{A}$, the original constraints of the problem. The disjuncts of $q_{\mathfrak{T}}$ are the CQs defined as follows. For each CQ we show that some property hold. Together these properties show the desired reduction.

Enforcing a grid structure First we enforce a grid structure with vertical lines v and horizontal lines h between nodes. We write the line $v(||d||_f, ||d'||_f)$ (resp. $(h(||d||_f, ||d'||_f))$ whenever there exists $d \in ||d||_f$, $d' \in ||d'||_f$ with $(d, d') \in v^{\text{can}(\mathcal{T},\mathcal{A})}$ (resp. $(d, d') \in h^{\text{can}(\mathcal{T},\mathcal{A})}$). Define a *cell* in a quotient structure $f(\text{can}(\mathcal{T}, \mathcal{A}))/\sim_f$ as a collection of nodes $||d||_f, ||d'||_f, ||d''||_f$ with $h(||d||_f, ||d'||_f)$ and $v(||d||_f, ||d''||_f)$ such that there exist nodes $||e||_f, ||e'||_f$ with $v(||d'||_f, ||e||_f)$ and $h(||d''||_f, ||e'||_f)$. A cell is said to be *closed* if $||e||_f = ||e'||_f$. Otherwise we call it an *open cell*. Figure 5.1 illustrates an open and a closed cell. A quotient structure $f(\text{can}(\mathcal{T}, \mathcal{A}))/\sim_f$ is a *grid* if it does not contain an open cell. Define the CQ

$$q_0 \leftarrow (x_1 \sim x_2) \wedge h(x_1, y_1) \wedge (y_1 \sim y_1') \wedge v(y_1', z_1) \wedge v(x_2, y_2) \wedge (y_2 \sim y_2') \wedge h(y_2', z_2) \wedge (z_1 \not\sim z_2).$$

We claim that

$$(\mathcal{T}, \mathcal{A}) \not\models q_0 \iff \exists f \text{ such that } f(\text{can}(\mathcal{T}, \mathcal{A}))/\sim_f \text{ is a grid.}$$

The proposition is equivalent to the following statement: q_0 has a match in $f(\text{can}(\mathcal{T}\mathcal{A}))$ if, and only if $f(\text{can}(\mathcal{T}, \mathcal{A}))/\sim_f$ has an open cell.

To show this, first suppose that q_0 has a match in $f(\text{can}(\mathcal{T}\mathcal{A}))$. Thus there exist $d, d', d'' \in \Delta^{\text{can}(\mathcal{T},\mathcal{A})}$ with $(d, d') \in v^{\text{can}(\mathcal{T},\mathcal{A})}$, $(d, d'') \in h^{\text{can}(\mathcal{T},\mathcal{A})}$, $(d, u) \in U^{f(\text{can}(\mathcal{T},\mathcal{A}))}$, $(d', u) \in U^{f(\text{can}(\mathcal{T},\mathcal{A}))}$, $(d, u') \in U^{f(\text{can}(\mathcal{T},\mathcal{A}))}$, $(d'', u') \in U^{f(\text{can}(\mathcal{T},\mathcal{A}))}$. Also, there exist $e, e' \in \Delta^{\text{can}(\mathcal{T},\mathcal{A})}$ with $(d', e) \in v^{\text{can}(\mathcal{T},\mathcal{A})}$, $(d'', e') \in h^{\text{can}(\mathcal{T},\mathcal{A})}$, $(d', u) \in U^{f(\text{can}(\mathcal{T},\mathcal{A}))}$, $(e, u) \in U^{f(\text{can}(\mathcal{T},\mathcal{A}))}$, $(d'', u') \in U^{f(\text{can}(\mathcal{T},\mathcal{A}))}$, $(e', u') \in U^{f(\text{can}(\mathcal{T},\mathcal{A}))}$, $(e, u_1) \in U^{f(\text{can}(\mathcal{T},\mathcal{A}))}$, $(e'', u_2) \in U^{f(\text{can}(\mathcal{T},\mathcal{A}))}$, with $u_1 \neq u_2$. Consider $f(\text{can}(\mathcal{T}, \mathcal{A}))/\sim_f$ and the definitions of

quotient structure, cell, and open cell. From the facts above, it follows directly that $f(\text{can}(\mathcal{T},\mathcal{A}))/\sim_f$ has an open cell (see Figure 5.1, left-hand side). Assuming, on the contrary, for the other direction, that $f(\text{can}(\mathcal{T},\mathcal{A}))/\sim_f$ does not have an open cell (see Figure 5.1, right-hand side), it is clear that q_0 does not have a match in $f(\text{can}(\mathcal{T}\mathcal{A}))$.

Placing one and exactly one tile type at each node Intuitively a grid $f(\text{can}(\mathcal{T},\mathcal{A}))/\sim_f$ represents the discrete plane where tile types can be placed. We say that the tile type T_k is true at node $||d||_f$ in $f(\text{can}(\mathcal{T},\mathcal{A}))/\sim_f$ if $(d,u) \in (U_k^1)^{f(\text{can}(\mathcal{T},\mathcal{A}))}$ and $(d,u) \in (U_k^2)^{f(\text{can}(\mathcal{T},\mathcal{A}))}$ where $d \in ||d||_f$, for some data value u. In queries we use the abbreviation $T_k(x) = U_k^1(x,v) \wedge U_k^2(x,v)$. Define for each $k_i \neq k_j$ the CQ

$$q_{k_i,k_j} \leftarrow (x \sim y) \wedge T_{k_i}(x) \wedge T_{k_j}(y).$$

Also define the single CQ

$$q_1 \leftarrow \bigwedge_{1\leq k\leq n} (U_k^1(x,v_{1,k}) \wedge U_k^2(x,v_{2,k}) \wedge (v_{1,k} \neq v_{2,k})).$$

It can be checked that

- $(\mathcal{T},\mathcal{A}) \not\models q_{k_i,k_j} \iff \exists f$ such that for all nodes $||d||_f$ in $f(\text{can}(\mathcal{T},\mathcal{A}))/\sim_f$ and for all distinct tile types T_{k_i}, T_{k_j} it is not the case that both T_{k_i} and T_{k_j} are true at $||d||_f$; and
- $(\mathcal{T},\mathcal{A}) \not\models q_1 \iff \exists f$ such that for nodes $||d||_f$ there exists a tile type T_k that is true at $||d||_f$.

The properties above together entail that there exists a completion function f for $\text{can}(\mathcal{T},\mathcal{A}))$ such that for each node $||d||_f$ in $f(\text{can}(\mathcal{T},\mathcal{A}))/\sim_f$ there exists one, and exactly one, tile type T_k that is true at it. (Intuitively, one tile type is placed at each point in the discrete plane.)

Enforcing vertical and horizontal compatibility conditions Now we need to enforce the vertical and horizontal compatibility conditions among the tile types. So for all tile types $T_{k_i},\ T_{k_j}$,

- if $\mathsf{right}(T_{k_i}) \neq \mathsf{left}(T_{k_j})$, then introduce the CQ $q_{h,k_i,k_j} \leftarrow h(x,y) \wedge T_{k_i}(x) \wedge T_{k_j}(y)$;
- if $\mathsf{up}(T_{k_i}) \neq \mathsf{down}(T_{k_j})$, then introduce the CQ $q_{v,k_i,k_j} \leftarrow v(x,y) \wedge T_{k_i}(x) \wedge T_{k_j}(y)$.

Let $\bar{H}$ denote the set of all pairs of horizontally, and $\bar{V}$ the set of all pairs of vertically, incompatible tile types from $T_1,\ldots,T_n$. We use $[\bar{H}]$ and $[\bar{V}]$ to denote their respective sets of pairs of indices. It can be checked that $(\mathcal{T},\mathcal{A}) \not\models (\bigvee_{(k_i,k_j)\in[\bar{H}]} q_{h,k_i,k_j}) \vee (\bigvee_{(k_i,k_j)\in[\bar{V}]} q_{v,k_i,k_j}) \iff$ there exists a completion function f for $\text{can}(\mathcal{T},\mathcal{A})$ such that there does not exist nodes $||d||_f, ||d'||_f$ in $f(\text{can}(\mathcal{T},\mathcal{A}))/\sim_f$ with $v(||d||_f, ||d'||_f)$ (resp., $h(||d||_f, ||d'||_f)$) where $(T_{k_i},\ T_{k_j}) \in \bar{V}$ (resp., $(T_{k_i},\ T_{k_j}) \in \bar{H}$) and both T_{k_i} is true at $||d||_f$ and T_{k_j} is true at $||d'||_f$.

Now let

$$q_{\mathfrak{T}} := q_0 \vee q_1 \vee \Big(\bigvee_{i,j\in[n]} q_{k_i,k_j}\Big) \vee \Big(\bigvee_{(k_i,k_j)\in[\bar{H}]} q_{h,k_i,k_j}\Big) \vee \Big(\bigvee_{(k_i,k_j)\in[\bar{V}]} q_{v,k_i,k_j}\Big).$$

It can be checked that $(\mathcal{T},\mathcal{A}) \models q_{\mathfrak{T}}$ iff $\mathfrak{T}$ does not tile $\mathbb{N}\times\mathbb{N}$. This concludes the reduction.

Example 5.2.1. Let $\mathfrak{T} = \{T_1, T_2, T_3, T_4\}$ be a set of tile types with colours

$$\begin{aligned}
&\mathsf{up}(T_1) = 0,\ \mathsf{down}(T_1) = 1,\ \mathsf{left}(T_1) = 0,\ \mathsf{right}(T_1) = 1,\\
&\mathsf{up}(T_2) = 0,\ \mathsf{down}(T_2) = 1,\ \mathsf{left}(T_2) = 1,\ \mathsf{right}(T_2) = 0,\\
&\mathsf{up}(T_3) = 1,\ \mathsf{down}(T_3) = 0,\ \mathsf{left}(T_3) = 0,\ \mathsf{right}(T_3) = 1,\\
&\mathsf{up}(T_4) = 1,\ \mathsf{down}(T_4) = 0,\ \mathsf{left}(T_4) = 1,\ \mathsf{right}(T_4) = 0.
\end{aligned}$$

Figure 5.2.1 illustrates them with 0 = white and 1 = black. It is easy to see that we can tile $\mathbb{N}\times\mathbb{N}$ by forming a repeating pattern of black losanges as shown in Figure 5.2.1. This is accomplished by the function $g\colon \mathbb{N}\times\mathbb{N} \to \mathfrak{T}$ given by

$$\begin{aligned}
&g(i,j) = T_1, \text{ for } i \in \{0,2,4,\ldots\},\ j \in \{1,3,5,\ldots\},\\
&g(i,j) = T_2, \text{ for } i,j \in \{1,3,5,\ldots\},\\
&g(i,j) = T_3, \text{ for } i,j \in \{0,2,4,\ldots\},\\
&g(i,j) = T_4, \text{ for } i \in \{1,3,5,\ldots\},\ j \in \{0,2,4,\ldots\}.
\end{aligned}$$

Let $\mathcal{T},\mathcal{A}$ be as mandated by our construction above. We define a Boolean UCQ $q_{\mathfrak{T}}$ that encodes the tiling problem by means of negative constraints:

$$\begin{aligned}
q_{\mathfrak{T}} := {} & q_0 \vee q_1 \vee q_{1,2} \vee \vee q_{1,3} \vee q_{1,4} \vee q_{2,3} \vee q_{2,4} \vee q_{3,4}\\
& \vee q_{h,2,4} \vee q_{h,4,2} \vee q_{h,1,3} \vee q_{h,3,1} \vee q_{v,2,1} \vee q_{v,1,2} \vee q_{v,4,3} \vee q_{v,3,4}.
\end{aligned}$$

where each CQ is as defined above. Any completion will give us a grid. We know that $(\mathcal{T},\mathcal{A}) \not\models q_{\mathfrak{T}}$ iff for some completion f of $\mathrm{can}(\mathcal{T},\mathcal{A})$, $f(\mathrm{can}(\mathcal{T},\mathcal{A})) \not\models q_{\mathfrak{T}}$. Such a completion can be found by using as guidance the function g defined above.

By the reduction above we immediately obtain the result:

Theorem 5.2.2. *Let $\mathcal{D} \in \{(\mathbb{Z},\neq),(\mathbb{Q},\neq)\}$. Then the query evaluation problem for OMQs with DL-Lite$_{\mathcal{R}}^{attrib}(\mathcal{D})$ TBoxes is undecidable in combined complexity.*

5.2.2 Numerical datatypes with linear order

We now address the decidability of query evaluation with linear order in the datatype. We start with strict linear order. Again we obtain a negative result:

Theorem 5.2.3. *Let $\mathcal{D} \in \{(\mathbb{Z},<),(\mathbb{Q},<)\}$. Then the query evaluation problem for OMQs with DL-Lite$_{\mathcal{R}}^{attrib}(\mathcal{D})$ TBoxes is undecidable in combined complexity.*

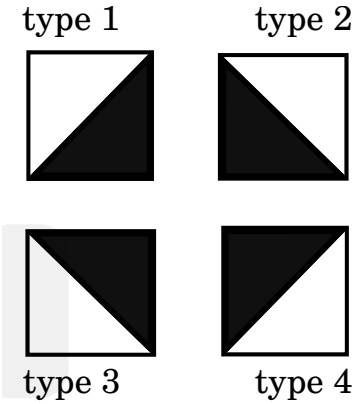

Figure 5.2: Tile types for Example 5.2.1

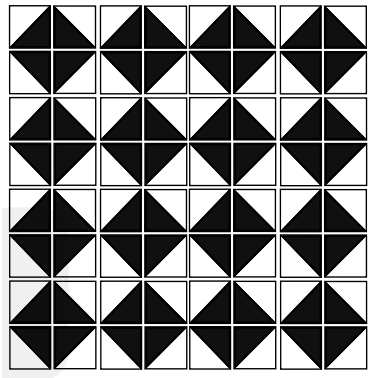

Figure 5.3: Tiling of $\mathbb{N} \times \mathbb{N}$ using tile types 1-4 (infinite pattern) for Example 5.2.1

Proof. Let $\mathcal{D} \in \{(\mathbb{Z}, \neq), (\mathbb{Q}, \neq)\}$ and $\mathcal{D}' \in \{(\mathbb{Z}, <), (\mathbb{Q}, <)\}$. By Theorem 5.2.2, query evaluation is undecidable for $\mathcal{D}$ already when only a single datatype atom is allowed in each CQ in the input UCQ and TBoxes do not contain qualified attribute restrictions. Notice that $x < y \vee x > y$ if, and only if, $x \neq y$. So to obtain a reduction one only needs to replace each CQ q in the input UCQ that contains an atom $x \neq y$ by two CQs q_1, q_2 where q_1 is exactly like q except that it contains contains $x < y$ in place of $x \neq y$, and q_2 is exactly like q except that it contains $x > y$ in place of $x \neq y$. Thus undecidability of query answering over $\mathcal{D}$ yields undecidability of query answering over $\mathcal{D}'$. The theorem now follows. □

This result can be used to show that query answering over *discrete* numerical datatypes with *non-strict linear order* again suffer from undecidability. This is done by simulating OMQ answering over $(\mathbb{Z}, <)$ with *DL-Lite*$_{\mathcal{R}}^{attrib}(\mathbb{Z}, <)$ TBoxes, which we proved undecidable, using OMQs over $(\mathbb{Z}, \leq)$.

Theorem 5.2.4. *The query evaluation problem for OMQs with DL-Lite*$_{\mathcal{R}}^{attrib}(\mathbb{Z}, \leq)$ *TBoxes is undecidable in combined complexity.*

Proof. Let $(\mathcal{T}, \mathcal{A})$ be a *DL-Lite*$_{\mathcal{R}}^{attrib}(\mathbb{Z}, <)$ KB and let q be an UCQ over $(\mathbb{Z}, <)$. We reduce $Q = (\mathcal{T}, q)$ and $\mathcal{A}$ into an OMQ $Q' = (\mathcal{T}', q')$ over $(\mathbb{Z}, \leq)$ where $\mathcal{T}'$ is a *DL-Lite*$_{\mathcal{R}}^{attrib}(\mathbb{Z}, \leq)$-TBox, with a $(\mathbb{Z}, \leq)$-ABox $\mathcal{A}'$. The reduction uses an auxiliary *DL-Lite*$_{\mathcal{R}}^{attrib}(\mathbb{Z}, \leq)$ TBox $\mathcal{T}_{\text{ord}}$ and an auxiliary UCQ q_{ord} defined next, which allow us to simulate $<$ and $=$ on the attributes in $\mathcal{T}$. We will show that $\mathcal{A} \models Q$ if, and only if, $\mathcal{A}' \models (\mathcal{T} \cup \mathcal{T}_{\text{ord}}, q')$.

Let

$$\mathcal{T}_{\text{ord}} = \{\exists r^- \sqsubseteq \exists r, \exists r \sqsubseteq \exists r^-, \exists r^- \sqsubseteq \exists V\},$$

where V is an attribute name and r is a role name, both not occurring in $\mathcal{T}, q$ or $\mathcal{A}$. Let q_{ord} be a Boolean UCQ consisting of the following disjuncts:

$$q_{\text{ord}}^{<} \leftarrow r(x, y) \wedge V(x, z_x) \wedge V(y, z_y) \wedge z_y \leq z_y,$$
$$q_{\text{ord}}^{U} \leftarrow r(x, y) \wedge V(y, z) \wedge U(s, z),$$

for each attribute name U occuring in $\mathcal{T}$. We will define a linearly ordered sequence $\ldots, I_{-2}, I_{-1}, I_0, I_1, I_2, \ldots$ of disjoint intervals over $\mathbb{Z}$, which we write $(I_i)_{i \in \mathbb{Z}}$. This sequence will be shown to satisfy the property that each value of an attribute in $\mathcal{T}$ is contained in some I_i. Also, given data values v_1, v_2 of an attribute in $\mathcal{T}$ we will set $v_1 \approx v_2$ if v_1, v_2 belong to the same interval and $v_1 \prec v_2$ if the interval where v_1 occurs precedes the interval where v_2 occurs.

To define each interval I_i we use $\mathcal{T}_{\text{ord}}$ and q_{ord} as follows. Take a completion $\mathcal{I}$ of $\text{can}(\mathcal{T} \cup \mathcal{T}_{\text{ord}}, \mathcal{A} \cup r(a_0, a_1))$ where a_0, a_1 are distinct individual names. Then

$$r^{\mathcal{I}} = \{\ldots, (a_{-2}, a_{-1}), (a_{-1}, a_0), (a_0, a_1), (a_1, a_2), \ldots\},$$

that is, it is an infinite path (Figure 5.4).

Also, for each $i \in \mathbb{Z}$, there is exactly one element $v_i \in \mathbb{Z}$ such that $(a_i, v_i) \in V^{\mathcal{I}}$; see Figure 5.5.

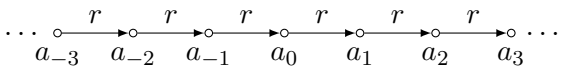

Figure 5.4: Infinite path in $r^{\mathcal{I}}$

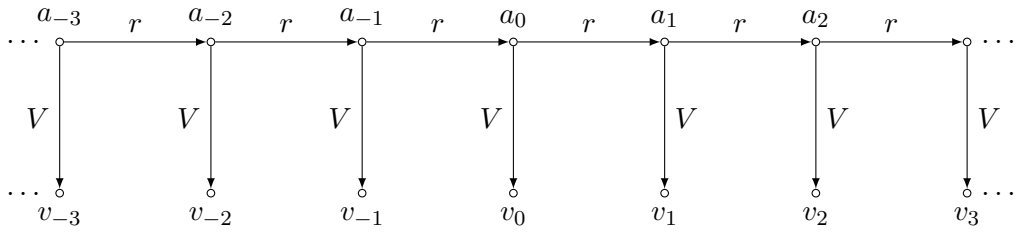

Figure 5.5: Infinite path of values of $\mathbb{Z}$ from the pairs in $V^{\mathcal{I}}$

Now we are ready to define the intervals $(I_i)_{i\in\mathbb{Z}}$. Let, for all $i \in \mathbb{Z}$,

$$I_i := \{v \in \mathbb{Z} \mid v_i < v < v_{i+1}\}.$$

Assume $\mathcal{I} \not\models q_{\text{ord}}$. Then,

1. For all $i \in \mathbb{Z}$, we have $v_i < v_{i+1}$, as shown in Figure 5.6;

2. For all U, for all $(a, u) \in U^{\mathcal{I}}$ there exists a unique $i \in \mathbb{Z}$ such that $u \in I_i$.

As for (2), notice that since $r^{\mathcal{I}}$ is an infinite path and $(\mathbb{Z}, \leq)$ is discrete, there exist j_1, j_2 with $j_1 < j_2$ such that $v_{j_1} \leq u \leq v_{j_2}$. But the construction is such that, since we assume $\mathcal{I} \not\models q^U_{\text{ord}}$, it is also the case that $u \neq v_i$ for all $i \in \mathbb{Z}$. Thus, together with the fact that the intervals $(I_i)_{i\in\mathbb{Z}}$ are disjoint, we have $u \in I_i$ for exactly one $i \in \mathbb{Z}$. Therefore $(I_i)_{i\in\mathbb{Z}}$ is as desired (Figure 5.7).

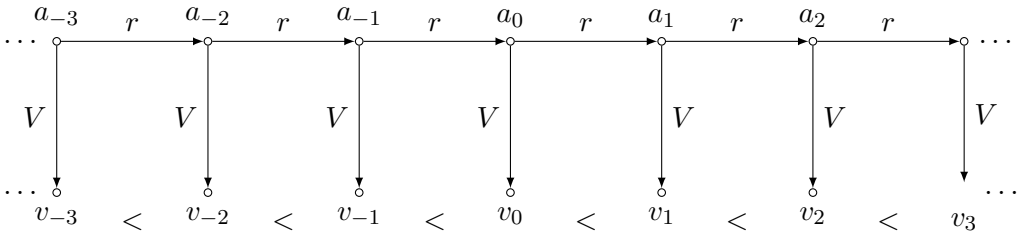

Figure 5.6: Strict linear order on $\mathbb{Z}$ from the pairs in $V^{\mathcal{I}}$

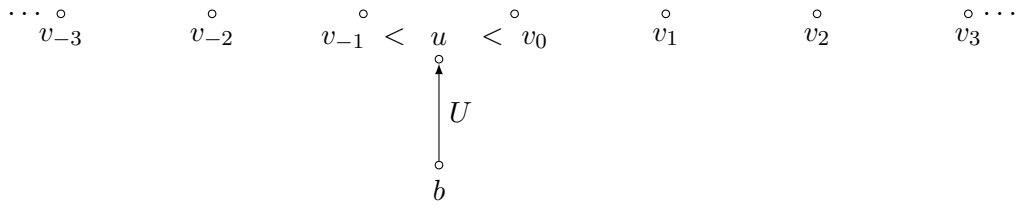

Figure 5.7: Any value u in $U^{\mathcal{I}}$ falls into a "bucket" or interval delimited by values in $V^{\mathcal{I}}$

We introduce the following abbreviations:

$$\begin{aligned} z \in (x,y) :\iff\ & r(x,y) \land V(x,v_x) \land V(y,v_y) \land \\ & v_x \leq z \land z \leq v_y, \\ (x,y) < (x',y') :\iff\ & r(x,y) \land r(x',y') \land V(y,v_y) \land \\ & V(x',v_{x'}) \land v_y \leq v_{x'}. \end{aligned}$$

In a model $\mathcal{I}$ as above, "$z \in (x,y)$" means that $(x,y) = (a_i, a_{i+1})$ for some $i \in \mathbb{Z}$ and $z \in I_i$, and "$(x,y) < (x',y')$" means that there are $i < j$ such that $(x,y) = (a_i, a_{i+1})$ and $(x',y') = (a_j, a_{j+1})$.

We the following abbreviations to define the desired orderings $\prec$ and $\approx$:

$$\begin{aligned} z_1 \approx z_2 :\iff\ & z_1 \in (x,y) \land z_2 \in (x,y), \\ z_1 \prec z_2 :\iff\ & z_1 \in (x_1,y_1) \land z_2 \in (x_2,y_2) \land \\ & (x_1,y_1) < (x_2,y_2), \end{aligned}$$

where "$z_1 \approx z_2$" means that z_1 and z_2 belong to the same interval I_i, and "$z_1 \prec z_2$" means that z_1 belongs to an interval that precedes the interval of z_2.

To conclude, recall that $\mathcal{T}' := \mathcal{T} \cup \mathcal{T}_{\text{ord}}$ and let q' be the UCQ given by q'', q_{ord} where q'' is obtained from q by expressing $<$ and equality on data variables in terms of $\prec$ and $\approx$. We also let $\mathcal{A}' := \mathcal{A} \cup \{r(a_0, a_1)\}$. It is straightforward to show that $\mathcal{A} \models (\mathcal{T}, q)$ iff $\mathcal{A}' \models (\mathcal{T}', q')$. □

Remark 5.2.5. Decidability of query evaluation over dense, (non-strictly) linearly ordered datatypes is open. It is not difficult to see why the construction above fails when we go from $(\mathbb{Z}, \leq)$ to $(\mathbb{Q}, \leq)$: it relies on the fact that the former is discrete and allows us to construct "buckets" of finite cardinality delimited by values from $\mathbb{Z}$ that cover the whole domain, as shown in Figure 5.7. When we move to $(\mathbb{Q}, \leq)$ we cannot do so. We consider a simple counterexample. Let $\mathcal{I}$ be the completion where $v_i \in [0,1]$ for all i and $u \in \mathbb{Q} \setminus [0,1]$ for some attribute value u occuring in $\text{can}(\mathcal{T}, \mathcal{A})$. Notice that assuming $\mathcal{I} \not\models q_{\text{ord}}$, we do obtain an infinite path in $r^{\mathcal{I}}$, $\ldots, a_{-2}, a_{-1}, a_0, a_1, a_2, \ldots$, where for each (a_i, v_i), v_i is a rational value and for all v_i, v_{i+1}, we have $v_i < v_{i+1}$; and also, for all attribute values v occuring in $\text{can}(\mathcal{T}, \mathcal{A})$, $v \neq v_i$ for all i. The (now dense) disjoint intervals I_i over $\mathbb{Q}$ are constructed in the same way. However, clearly there is no i such that $u \in I_i$.

5.3 Restoring decidability: the Bounded-Match Depth Property of OMQs

We now introduce a property of OMQs based on the notion of completion defined in Chapter 2 which allows us to restore decidability for a great portion of UCQs used in practice.

5.3.1 Safe and rooted queries

The constructions used to prove Theorems 5.2.2, 5.2.4, and 5.2.3 rely crucially on TBoxes whose chase does not terminate and on UCQs that are neither *safe* nor *rooted*. These properties of UCQs are a starting point in our discussion on decidability conditions and will be defined next.

A *path* from a variable x to a variable y in a CQ q over a datatype $\mathcal{D}$ is a sequence $x, z_1, \dots, z_n, y$, where all of $z_1, \dots, z_n$ are non-data variables and each two subsequent variables in $x, z_1, \dots, z_n, y$ occur together in an atom of q. Then we say that x and y are *connected in* q if there exists a path from x to y in q.

Example 5.3.1. Let $q_1() \leftarrow r(x,y), r(y,z), U(y,u)$ be a CQ over a datatype $\mathcal{D}$. Then x and z are connected in q_1. Now we look at a negative example. Let

$$q_2(x) \leftarrow r(x,y), U(x,u), U(y,u), U(z,u)$$

be a CQ over $\mathcal{D}$. Then x and z are not connected in q_2 because there is no path from x to z that avoids u, which is a data variable.

We define:

Definition 5.3.2. Let q be a CQ over a datatype $\mathcal{D}$. Then

1. q is *rooted* if any non-data variable in q is connected to a non-data answer variable of q.

2. q is *safe* if for all non-data variables x, y in q, the following holds:
   - x is connected to y in q; or
   - x is connected to a non-data answer variable x' and y is connected to a non-data answer variable y'.

Remark 5.3.3. Rooted CQs were studied in [12] in the context of query containment and query optimisation in *DL-Lite*. This class of queries is of course only a subset of queries used in practical applications; nonetheless, the restriction is quite reasonable, as seen in the context of CQ rewritability [87] (for this topic, see Chapter 7).[1] Notice that given a CQ q over $\mathcal{D}$, q is rooted only if q is safe. The converse, however, does not hold.

[1] "[Rooted queries form] a very general class, which arguably captures most practical OBDA queries" [87].

Example 5.3.4. The CQ q_1 in Example 5.3.1 is safe, but not rooted; q_2 is not safe, and therefore not rooted. The CQ

$$q_3(x) \leftarrow r(x,y), r(y,z)$$

is rooted and thus safe.

We say that a UCQ q is *rooted (safe)* if all its disjuncts are rooted (safe, respectively).

5.3.2 Introducing the BMDP

Recall the notions introduced in Chapter 2. Here $\text{can}^d(\mathcal{T},\mathcal{A})$ denotes the subinterpretation of $\text{can}(\mathcal{T},\mathcal{A})$ induced by the set of domain elements reachable from elements of $\mathcal{A}$ in the Gaifman graph of $\text{can}(\mathcal{T},\mathcal{A})$ in at most d steps. We define the desired property of OMQs using a notion from the database literature([30]; see following Remark 5.3.6):

Definition 5.3.5. Let $\mathcal{D}$ be a datatype and $Q = (\mathcal{T}, q)$ be an OMQ over $\mathcal{D}$. Then Q has the *bounded match depth property* (BMDP) if there exists $d \geq 0$ such that for all $\mathcal{D}$-ABoxes $\mathcal{A}$ and all tuples $\bar{c}$,

$$\mathcal{T},\mathcal{A} \models q(\bar{c}) \iff f(\text{can}^d(\mathcal{T},\mathcal{A})) \models q(\bar{c})$$

for all completion functions f.

Remark 5.3.6. This property is well known in the database literature, having been introduced with different notation and within a similar framework, for dealing with non-terminating chases. In [30], the authors define the Bounded Guard-Depth Property (BGDP) (see Definition 1 in that paper) and the strictly stronger notion of the Bounded Derivation Depth Property (BDDP, Definition 2). We adapt and use the latter in our framework, employing the notion of the match.

We now consider three important classes of OMQs which have the BMDP: OMQs with *DL-Lite* TBoxes for which the chase terminates; OMQs with rooted queries; and OMQs with safe queries over homogeneous datatypes.

OMQs with *DL-Lite* TBoxes and terminating chase. Let $Q = (\mathcal{T}, q)$ be an OMQ over a datatype $\mathcal{D}$ where $\mathcal{T}$ is a *DL-Lite* TBox such that for all ABoxes $\mathcal{A}$, $\text{chase}(\mathcal{T},\mathcal{A})$ terminates in finitely many steps. This is sometimes the case in practice [56]. Note that this implies that there exists an uniform bound n such that for all ABoxes $\mathcal{A}$, the chase terminates in at most n steps. Therefore there exists a d such that for all $\mathcal{A}$, $\text{can}^d(\mathcal{T},\mathcal{A}) = \text{can}(\mathcal{T},\mathcal{A})$. It can be readily checked that by Definition 5.3.5 Q has the BMDP.

Rooted OMQs. Now let $Q = (\mathcal{T}, q)$ be rooted and $\mathcal{A}$ be a $\mathcal{D}$-ABox; we relax the requirement that chase$(\mathcal{T}, \mathcal{A})$ terminates. Yet since Q is rooted, it can be answered by looking at a finite portion of the chase. Consider all matches of q in $f(\text{can}(\mathcal{T}, \mathcal{A}))$ for a completion function f, and let $\bar{x}$ denote the answer variables in q. Note that, for each match μ, one only needs to look at elements that are reachable from $\mu(\bar{x})$ in at most d steps in $f(\text{can}(\mathcal{T}, \mathcal{A}))$, where d is the maximum number of atoms in a CQ in q. (Recall that by definition only ABox individuals and datatype values can be answers to q.) We can therefore construct $\text{can}^d(\mathcal{T}, \mathcal{A})$ and answer q on the interpretations $f(\text{can}^d(\mathcal{T}, \mathcal{A}))$ where f ranges over the completion functions for $\text{can}^d(\mathcal{T}, \mathcal{A})$. That is, Q has the BMDP.

Safe queries over homogeneous datatypes. We recall the notion of an *homogeneous* structure [39], already introduced in Chapter 3. Let $\mathcal{B} = (B, R_1^{\mathcal{B}}, R_2^{\mathcal{B}}, \dots)$ be a relational structure over a signature Σ.[2] Then the Σ-structure $\mathcal{D} = (D, R_1^{\mathcal{D}}, R_2^{\mathcal{D}}, \dots)$ is an *induced substructure of* $\mathcal{B}$ if $D \subseteq B$ and $R_i^{\mathcal{D}} = R_i^{\mathcal{B}} \cap D^n$ for every n-ary relation symbol R_i in Σ.

Definition 5.3.7. (Restated) A datatype $\mathcal{B}$ is *homogeneous* if for all pairs of induced substructures $\mathcal{B}, \mathcal{B}'$ of $\mathcal{D}$, an isomorphism between $\mathcal{B}$ and $\mathcal{B}'$ can be extended to an automorphism of $\mathcal{D}$.

Example 5.3.8. Examples of homogeneous datatypes are:

The structure $(\mathbb{Q}, <)$, or the same structure with reverse order. Let h be a partial isomorphism from a substructure $\mathcal{A}$ to a substructure $\mathcal{B}$ of $(\mathbb{Q}, <)$. Take any rational number s not in $\mathcal{A}$. Then it can be checked that we can choose (by using the right interval) a rational number t not in $\mathcal{B}$ such that h can be extended into a partial isomorphism by taking s to t. For instance, if $\mathcal{A} = (\{1, 2\}, <)$ and $\mathcal{B} = (\{3, 4\}, <)$, and $h(1) = 3,\ h(2) = 4$, if we pick 1.5 we can extend h into a partial isomorphism by mapping it to 3.5. By repeating this procedure, in the limit one obtains an automorphism of $\mathbb{Q}$.

Structures $(\mathbb{Q}, R)$ R is taken to be the ternary linear *betweenness* relation; when $\mathbb{Q}$ is bended to form a circle, that is, where $R(x, y, z) \iff (x < y < z) \vee (y < z < x) \vee (z < x < y)$; and when R is the *separation* relation. That is, where $R \in \{R_{Betw},\ R_{\text{Cyc}},\ R_{\text{Sep}}\}$ are as defined in temporal languages in Chapter 3. The homogeneity of such structures has been shown [36].

Certain structures $(D, \leq)$ such that for all $a, b \in D$ there exists a $c \in D$ such that $c < a \ \wedge \ c < b$; and the set of all predecessors of an element a, written $\ell(a)$, is a chain, that is, $x, y \in \ell(a)$ implies $x \leq y$ or $y \leq x$. Such structures are called *pseudo-trees* and sometimes simply *trees* and are studied in [86]. Homogeneous cases have been classified in [45]. The typical example is $(\mathbb{Q}, \leq)$.

ω-categorical relational structures $\mathcal{D}$ (see Chapter 3) satisfying the condition that Th$(\mathcal{D})$ has quantifier elimination [39]. A proof that such structures are homogeneous can be found in [79].

[2] For the purpose of this definition we restore the distinction between a relation $R^{\mathcal{D}}$ and the relation symbol R from Σ, given a Σ-structure $\mathcal{D}$ containing R.

A comprehensive survey on countable homogeneous structures can be found in [79].

Example 5.3.9. In contrast, the datatype $\mathcal{D} = (\mathbb{N}, \leq)$ is not homogeneous. To see this, consider the substructures $\mathcal{D}' = (\{1, 3\}, R_{13})$ and $\mathcal{D}'' = (\{0, 1\}, R_{01})$ where $R_{ij} = \{(i, j)\}$. Then there is bijection α between $\mathcal{D}'$ and $\mathcal{D}''$, to wit $\alpha(1) = 0$ and $\alpha(3) = 1$. But clearly α cannot be extended to an automorphism of $(\mathbb{N}, \leq)$; for then we cannot map e.g. 2 to any natural number in order to obtain a partial isomorphism, since the interval $(0, 1)$ over $\mathbb{N}$ is empty.

Before proving that safe OMQs over homogeneous datatypes have the BMDP, we illustrate this fact with an example. So let $(\mathcal{T}, \mathcal{A})$ be a Horn-$\mathcal{L}(\mathbb{Q}, \leq)$-KB with

$$\mathcal{T} = \{\exists r^- \sqsubseteq \exists r, \exists r \sqsubseteq \exists r^-, \exists r^- \sqsubseteq U\}$$

and $\mathcal{A} = \{r(a, b)\}$. An identical KB was used in the construction in the proof of Theorem 5.2.4. Now consider the CQs $q() \leftarrow r(x, y), U(x, u_1), U(y, u_2), u_1 \leq u_2$ (safe) and $q'() \leftarrow U(x, u_1), U(y, u_2), u_1 \leq u_2$ (not safe). We compare the answers to such CQs against $\text{can}(\mathcal{T}, \mathcal{A})$. It is easy to see that to refute q we only need to compare the attributes of individuals in $\text{can}(\mathcal{T}, \mathcal{A})$ that are connected through r. In contrast, checking whether $(\mathcal{T}, \mathcal{A}) \not\models q'$ holds requires comparing (via $<$) attribute values of individuals in $\text{can}(\mathcal{T}, \mathcal{A})$ that can be arbitrarily far apart.

Example 5.3.9, the proofs of lemmas used in Corollary 5.3.17 as well as Proposition 5.3.10 below provide a hint as to how homogenity relates to the BMDP of non-safe queries.

Proposition 5.3.10. *There is a safe OMQ $Q = (\mathcal{T}, \mathcal{A})$ over a non-homogeneous datatype $\mathcal{D}$ such that Q does not have the BMDP.*

Proof. Let $Q = (\mathcal{T}, q)$ be an OMQ over $(\mathbb{N}, \leq)$, which is not homogeneous, where

$$\begin{aligned}\mathcal{T} =& \{\exists r^- \sqsubseteq \exists r, \exists r \sqsubseteq \exists U, \exists r^- \sqsubseteq \exists U\},\\ & q() \leftarrow r(x, y), U(x, v_1), U(y, v_2), v_1 \leq v_2.\end{aligned}$$

Also let $\mathcal{A} = \{r(a_0, a_1)\}$. We show that (1) $\mathcal{T}, \mathcal{A} \models q$ and (2) for all d one can find a completion function f such that $f(\text{can}^d(\mathcal{T}, \mathcal{A})) \not\models q$.

To show (1) suppose q is not entailed by $(\mathcal{T}, \mathcal{A})$. Then there is a completion function f with $f(\text{can}(\mathcal{T}, \mathcal{A})) \not\models q$. Note that $\text{can}(\mathcal{T}, \mathcal{A})$ can be represented by the set containing exactly $r^{\text{can}(\mathcal{T}, \mathcal{A})} = \{(a_i, a_{i+1}) \mid i \geq 0\}$ and $U^{\text{can}(\mathcal{T}, \mathcal{A})} = \{(a_i, u_i) \mid i \geq 0\}$, where the nulls u_i, a_i are indexed in the order in which they are introduced by the chase. Then $a_0, a_1, \ldots$ forms an infinite sequence. Given that q is false in $\mathcal{I}$, we have $f(u_i) = n_i > n_{i+1} = f(u_{i+1})$ for all u_i. Since $f(d_j) = 0$ for some $j \geq 0$ we obtain that $a_0, a_1, \ldots, a_j$ is a finite sequence. Contradiction.

To show (2), note that $\text{can}^d(\mathcal{T}, \mathcal{A})$ can be represented by the set containing exactly $r^{\text{can}(\mathcal{T}, \mathcal{A})} = \{(a_i, a_{i+1}) \mid 0 \leq i \leq d\}$ and $U^{\text{can}(\mathcal{T}, \mathcal{A})} = \{(a_i, u_i) \mid 0 \leq i \leq d\}$. We obtain thus a sequence $u_0, u_1, \ldots, u_n$ where $n = d$. We begin by setting $f(u_0) = d + 1$ and then assign values to the u_i in such a way that $f(u_0) > f(u_1) > \ldots > f(u_n)$ is a (finite) strictly

descending chain. Thus there is no match of q in $f(\text{can}^d(\mathcal{T}, \mathcal{A}))$. Since d was arbitrary, we obtain that for all d there is a completion function f such that $f(\text{can}^d(\mathcal{T}, \mathcal{A})) \not\models q$.

Altogether we obtain that Q does not have the BMPD (Definition 5.3.5).

□

We now prove the fact stated above, that is, that safe OMQs over homogeneous datatypes have the BMDP. Notice that *non-Boolean safe queries* are rooted, so they have the BMDP by Definition 5.3.2. For this reason we focus here, without loss of generality, on *Boolean safe queries*.

We heavily use the following forest representation of pre-models $\text{can}(\mathcal{T}, \mathcal{A})$ of Horn-$\mathcal{L}(\mathcal{D})$-KBs $(\mathcal{T}, \mathcal{A})$.

Forest representation of canonical pre-models

Definition 5.3.11. Let $(\mathcal{T}, \mathcal{A})$ be a Horn-$\mathcal{L}(\mathcal{D})$-KB for a datatype $\mathcal{D}$. The *chase forest* of $\mathcal{T}$ and $\mathcal{A}$, denoted by $\text{CF}(\mathcal{T}, \mathcal{A})$, is defined inductively as follows.

We start with an empty forest and add, for each assertion $\alpha \in \mathcal{A}$, a new root node v_α labelled with the set $B(v_\alpha) := \{\alpha\}$.

For the induction step, consider a node v of $\text{CF}(\mathcal{T}, \mathcal{A})$ and let $B(v)$ be its label. Suppose some rule of the chase procedure can be applied to the pre-ABox $B(v)$ which generates a new atom β.[3] Let v' be the lowest ancestor of v such that $B(v')$ contains all elements that occur both in $B(v)$ and as arguments of β. If all arguments of β occur in $B(v')$, then we add β to $B(v')$. Otherwise, we create a new child v'' of v' and let $B(v'') := \{\beta\}$ be its label.

When we apply chase rules that introduce constraints for data nulls, we also add these constraints to the pre-ABox $B(v)$ of the corresponding node v.

For each node v of $\text{CF}(\mathcal{T}, \mathcal{A})$, we call $B(v)$ the *bag* of v and denote the *depth* of v by $\text{depth}(v)$. Given a set V of nodes of $\text{CF}(\mathcal{T}, \mathcal{A})$, we define

$$B(V) := \bigcup_{v \in V} B(v).$$

The forest representation coincides with the canonical pre-model. It is easy to see that the set of all assertions that occur in the bags of $\text{CF}(\mathcal{T}, \mathcal{A})$ coincides with $\text{can}(\mathcal{T}, \mathcal{A})$, up to renaming of individuals and data nulls that do not occur in $\mathcal{A}$. In the following, we may therefore assume without loss of generality that the set of all assertions that occur in the bags of $\text{CF}(\mathcal{T}, \mathcal{A})$ coincides *exactly* with $\text{can}(\mathcal{T}, \mathcal{A})$. The following lemma is immediate from the construction of $\text{CF}(\mathcal{T}, \mathcal{A})$.

Lemma 5.3.12. *Let $(\mathcal{T}, \mathcal{A})$ be a Horn-$\mathcal{L}(\mathcal{D})$-KB. If v is a node of $\text{CF}(\mathcal{T}, \mathcal{A})$, then there are at most two elements that occur as arguments of assertions in $B(v)$, and v has at most $2 \cdot |\mathcal{T}|$ children.*

[3] Here we assume that fresh constants or data nulls that are generated by chase rules of the type 3, 6, or 7 do not occur already in $\text{CF}(\mathcal{T}, \mathcal{A})$.

Additional notions. We will also require the following concepts of an *ℓ-neighborhood* and an *ℓ-type*. Informally, the ℓ-neighborhood of a node v of $\mathrm{CF}(\mathcal{T},\mathcal{A})$ contains all assertions in bags of nodes at distance less than ℓ from v. The ℓ-type of v w.r.t. a completion function f enriches the ℓ-neighborhood $\mathcal{N}$ of v with information about the relationships between the data values assigned by f to the data nulls in $\mathcal{N}$.

Definition 5.3.13. Let $(\mathcal{T},\mathcal{A})$ be a Horn-$\mathcal{L}(\mathcal{D})$-KB, let v be a node of $\mathrm{CF}(\mathcal{T},\mathcal{A})$, let $\ell \geq 1$, and let f be a completion function for $\mathrm{can}(\mathcal{T},\mathcal{A})$.

- The *ℓ-neighborhood* of v in $\mathrm{CF}(\mathcal{T},\mathcal{A})$ is defined as $\mathcal{N}_\ell(v) := B(N_\ell(v))$, where $N_\ell(v)$ is the set of all nodes at distance less than ℓ from v in $\mathrm{CF}(\mathcal{T},\mathcal{A})$.
- The *ℓ-type* of v in $\mathrm{CF}(\mathcal{T},\mathcal{A})$ with respect to f, denoted by $\mathrm{tp}_\ell(v)$, is the union of $\mathcal{N}_\ell(v)$ and the set of all atoms $R(u_1,\ldots,u_r)$ over $\mathcal{D}$ with each u_i a data value or a data null in $\mathcal{N}_\ell(v)$ and $(f(u_1),\ldots,f(u_r)) \in R$.

Let $q \leftarrow \varphi$ be a Boolean CQ. The *size* of q, denoted by $|q|$, is the number of atoms that occur in φ. Given a pre-match μ of q in $\mathrm{can}(\mathcal{T},\mathcal{A})$, we define $\mu(q)$ to be the image of q under μ, i.e., the smallest sub-interpretation $\mathcal{I}$ of $\mathrm{can}(\mathcal{T},\mathcal{A})$ such that μ is a match of q in $\mathcal{I}$.

Crucial properties of safe OMQs

Lemma 5.3.14. *Let $(\mathcal{T},\mathcal{A})$ be a Horn-$\mathcal{L}(\mathcal{D})$-KB, and let f be a completion function for* $\mathrm{can}(\mathcal{T},\mathcal{A})$.

1. *Consider a safe CQ q and a pre-match μ of q in* $\mathrm{can}(\mathcal{T},\mathcal{A})$. *Suppose that there is a node v of* $\mathrm{CF}(\mathcal{T},\mathcal{A})$ *at depth at least $|q|$ whose bag contains some atom of $\mu(q)$. Then, $\mu(q)$ is contained in $\mathcal{N}_{|q|}(v) \subseteq \mathrm{tp}_{|q|}(v)$.*
2. *If the number of relations in $\mathcal{D}$ is finite, then for each $\ell \geq 1$ the number of distinct ℓ-types of nodes of* $\mathrm{CF}(\mathcal{T},\mathcal{A})$ *with respect to f is at most*

$$t_\ell := 2^{m\cdot|\mathcal{T}|^{\mathcal{O}(\ell)}},$$

where m is the total number of concept names, role names, and attribute names that occur in $\mathcal{T}$.

Proof. Ad 1: Easy consequence of the definition of $\mathrm{CF}(\mathcal{T},\mathcal{A})$ and that of safe CQs.

Ad 2: Let v be a node of $\mathrm{CF}(\mathcal{T},\mathcal{A})$. By Lemma 5.3.12, the set $N_\ell(v)$ of all nodes at distance less than ℓ from v has size at most

$$\sum_{i=0}^{\ell-1} (2\cdot|\mathcal{T}|+1)^i \leq (2\cdot|\mathcal{T}|+1)^\ell.$$

Let $X := \Delta^{\mathcal{N}_\ell(v)}$. Since the bag of each node in $\mathrm{CF}(\mathcal{T},\mathcal{A})$ contains at most two elements (Lemma 5.3.12), we have

$$|X| \leq 2\cdot(2\cdot|\mathcal{T}|+1)^\ell = \mathcal{O}(|\mathcal{T}|)^\ell.$$

Each type consists of atoms built from concept names, role names, attribute names, and relation names in $\mathcal{D}$, using the elements in X as arguments. The number of such atoms is at most $m_1 \cdot |X|^2 + m_2 \cdot |X|^r$, where m_1 is the number of concept names, role names, and attribute names in $\mathcal{T}$, m_2 is the number of relations in $\mathcal{D}$, and r is the maximum arity of a relation in $\mathcal{D}$. Consequently, the number of distinct ℓ-types of nodes of $\mathrm{CF}(\mathcal{T}, \mathcal{A})$ with respect to f is at most

$$t_\ell = 2^{m_1 \cdot |X|^2 + m_2 \cdot |X|^r} = 2^{m_1 \cdot |\mathcal{T}|^{\mathcal{O}(\ell)}}.$$

Here we use that m_2 and r are constant (because $\mathcal{D}$ is constant). □

The main result. We are now ready to show that safe OMQs over homogeneous datatypes enjoy the BMDP. The following lemma states this result for TBoxes without attribute restrictions. Remark 5.3.16 outlines how to adapt the proof to the case of TBoxes with attribute restrictions.

Lemma 5.3.15. *Let $Q = (\mathcal{T}, q)$ be a safe Boolean OMQ over a homogeneous datatype $\mathcal{D}$ with $\mathcal{T}$ a Horn-$\mathcal{ALCHI}^{attrib}(\mathcal{D})$ TBox. Set $d := 2s + t_s$, where s is the maximum size of a disjunct of q and t_s is defined as in Lemma 5.3.14.*[4]

For every $\mathcal{D}$-ABox $\mathcal{A}$ that is satisfiable relative to $\mathcal{T}$, the following are equivalent:

1. *$f(\mathrm{can}(\mathcal{T}, \mathcal{A})) \models q$ for all completion functions f.*
2. *$f(\mathrm{can}^d(\mathcal{T}, \mathcal{A})) \models q$ for all completion functions f.*

Proof. Note that since $\mathcal{T}$ and q are finite and therefore refer to only finitely many relations of $\mathcal{D}$, we can assume without loss of generality that $\mathcal{D}$ contains only finitely many relations.

Let $\mathcal{A}$ be a $\mathcal{D}$-ABox that is satisfiable relative to $\mathcal{T}$. Clearly, if q is true in all completions of $\mathrm{can}^d(\mathcal{T}, \mathcal{A})$, then q is true in all completions of $\mathrm{can}(\mathcal{T}, \mathcal{A})$. Thus, it remains to prove the other direction: if q is false in some completion of $\mathrm{can}^d(\mathcal{T}, \mathcal{A})$, then q is false in some completion of $\mathrm{can}(\mathcal{T}, \mathcal{A})$.

To this end, suppose that there is a completion function f for $\mathrm{can}^d(\mathcal{T}, \mathcal{A})$ with $f(\mathrm{can}^d(\mathcal{T}, \mathcal{A})) \not\models q$. We use f to construct a completion function $\hat{f}$ for $\mathrm{can}(\mathcal{T}, \mathcal{A})$ with $\hat{f}(\mathrm{can}(\mathcal{T}, \mathcal{A})) \not\models q$.

We construct $\hat{f}$ inductively. Let $v_1, v_2, v_3, \ldots$ be a repetition-free enumeration of all the nodes of $\mathrm{CF}(\mathcal{T}, \mathcal{A})$ at depth at least $d + 1$ in a breadth-first fashion. For each $i \geq 0$ define the pre-interpretation

$$\mathcal{I}_i := \mathrm{can}^d(\mathcal{T}, \mathcal{A}) \cup B(\{v_j \mid 1 \leq j \leq i\}).$$

For each $i \geq 0$ we construct a completion function f_i for $\mathcal{I}_i$ with the following two properties:

[4] Here we assume that $\mathcal{D}$ contains only relations that occur in $\mathcal{T}$ or q. In particular, $\mathcal{D}$ can be assumed to be finite.

1. $f_i(\mathcal{I}_i) \not\models q$.

2. If $i \geq 1$, then f_i coincides with f_{i-1} on all elements that occur in a bag of $\mathrm{CF}(\mathcal{T}, \mathcal{A})$ at depth less than $s + \mathrm{depth}(v_i) - d$.

Finally, we let $\hat{f}$ be the completion function of $\mathrm{can}(\mathcal{T}, \mathcal{A})$ that maps each data null u to $f_i(u)$, where i is the smallest integer with $f_i(u) = f_j(u)$ for all $j \geq i$ (this integer exists by the second property above). We then have $\hat{f}(\mathrm{can}(\mathcal{T}, \mathcal{A})) \not\models q$. It remains to construct the functions f_i.

We let $f_0 := f$. Then, f_0 is a completion function for $\mathcal{I}_0 = \mathrm{can}^d(\mathcal{T}, \mathcal{A})$ such that $f_0(\mathcal{I}_0) \not\models q$. The second property is trivially satisfied for f_0.

Next assume that f_i has been constructed. We wish to construct f_{i+1}. Let W be the set of the $t_s + 1$ deepest ancestors of v_{i+1} of depth at most $\mathrm{depth}(v_{i+1}) - s$. Then, for each node $w \in W$,

$$\mathrm{depth}(w) \leq \mathrm{depth}(v_{i+1}) - s, \tag{5.1}$$

$$\begin{aligned} \mathrm{depth}(w) &\geq \mathrm{depth}(v_{i+1}) - s - t_s \\ &= s + \mathrm{depth}(v_{i+1}) - d. \end{aligned} \tag{5.2}$$

Note that (5.1) implies $\mathcal{N}_s(w) \subseteq \mathcal{I}_i$, and therefore f_i is defined on all data nulls in $\mathcal{N}_s(w)$. Moreover, (5.2) and $\mathrm{depth}(v_{i+1}) \geq d + 1$ imply that each node in W has depth at least $s + 1$. Consequently, by Lemma 5.3.14(1), each match of some disjunct of q in $\mathrm{can}(\mathcal{T}, \mathcal{A})$ that contains an atom from the bag of a node $w \in W$ is contained in $\mathcal{N}_s(w)$.

Now, since W contains $t_s + 1$ many nodes, there must be two distinct nodes $w_1, w_2 \in W$ with $\mathrm{tp}_s(w_1) \cong \mathrm{tp}_s(w_2)$. Say, w_1 is an ancestor of w_2. Let g_0 be an isomorphism from $\mathrm{tp}_s(w_1)$ to $\mathrm{tp}_s(w_2)$. We can extend this isomorphism to an isomorphism g that includes the atoms that occur in the subtree rooted at w_1. For each data value or data null u in $\mathcal{N}_s(w_1)$, let $h_0(f_i(u)) := f_i(g(u))$. Then, h_0 is an isomorphism between finite induced substructures of $\mathcal{D}$. Since $\mathcal{D}$ is homogeneous, h_0 extends to an automorphism h of $\mathcal{D}$. Now, for each data null u in $\mathrm{can}(\mathcal{T}, \mathcal{A})$, let $f_{i+1}(u) := f_i(u)$ if u does not occur in $\mathcal{I}_{i+1}$ in the subtree below w_2, and $f_{i+1}(u) := h(f_i(g^{-1}(u)))$ otherwise.

It is easy to see that f_{i+1} coincides with f_i on all elements that occur in a bag of $\mathrm{CF}(\mathcal{T}, \mathcal{A})$ at depth less than $s + \mathrm{depth}(v_{i+1}) - d$. It remains to show that $f_{i+1}(\mathcal{I}_{i+1}) \not\models q$. Suppose that $f_{i+1}(\mathcal{I}_{i+1}) \models q$. Then there is a match μ of some disjunct q' of q in $f_{i+1}(\mathcal{I}_{i+1})$. Since μ is not a match of q' in $f_i(\mathcal{I}_i)$ (by the induction hypothesis), and f_{i+1} coincides with f_i on all atoms that occur outside the subtree of w_2 (by construction), it must be the case that $\mu(q')$ contains an atom from the subtree of w_2. This implies that $\mu(q') \subseteq \mathcal{N}_s(w_2)$ or $\mu(q')$ is contained in the subtree rooted at w_2. But then the mapping

$$\mu'(x) := \begin{cases} g^{-1}(\mu(x)), & \text{if } x \text{ is not a data variable,} \\ h^{-1}(\mu(x)), & \text{if } x \text{ is a data variable} \end{cases}$$

is a match of q' in $f_i(\mathcal{I}_i)$, a contradiction. □

Remark 5.3.16. If $Q = (\mathcal{T}, q)$ is a safe Boolean OMQ over a homogeneous datatype $\mathcal{D}$ with $\mathcal{T}$ a Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$ TBox, then we augment the proof of Lemma 5.3.15 as follows. Let C be the set of all elements of $\mathrm{dom}(\mathcal{D})$ that occur in attribute restrictions of $\mathcal{T}$. We use an extended notion of ℓ-neighborhood: the *extended ℓ-neighborhood* of a node v in $\mathrm{CF}(\mathcal{T}, \mathcal{A})$ is defined as $\mathcal{N}_\ell(v) \cup C$. Note that this also induces an extended notion of ℓ-type, which is based on the extended ℓ-neighborhood rather than the ℓ-neighborhood. The extended notion of ℓ-type captures the relationship of data values and data nulls in the ℓ-neighborhood of a node and the elements in C. Finally, we consider the extended datatype $\hat{\mathcal{D}} = (\mathcal{D}, c_1, \ldots, c_n)$, where $c_1, \ldots, c_n$ is an enumeration of all the elements in C. This restricts isomorphisms and automorphisms to be the identity on the elements on C. Consequently, when we translate the completion function of a subtree of $\mathrm{CF}(\mathcal{T}, \mathcal{A})$ by applying an automorphism of $\hat{\mathcal{D}}$, we automatically preserve all the constraints imposed on the data nulls. Note that $\hat{\mathcal{D}}$ is homogeneous if $\mathcal{D}$ is homogeneous. This means that all other details of the proof can be left unchanged.

Theorem 5.3.17. *The following OMQs enjoy the BMDP: safe OMQs over homogeneous datatypes, rooted OMQs, and OMQs with a DL-Lite TBox $\mathcal{T}$ such that* $\mathrm{can}(\mathcal{T}, \mathcal{A})$ *terminates in a finite number of steps for each ABox $\mathcal{A}$.*

Proof. From paragraphs "OMQs with *DL-Lite* TBoxes and terminating chase" and "rooted OMQs" above, as well as Lemma 5.3.15 directly using Definition 5.3.5 (for safe OMQs over homogenous datatypes).

□

5.4 A framework for transfering classification results from CSPs to query answering

The main purpose of this work, as stated in the Introduction, is to carry forth a detailed investigation of OMQ answering with datatypes aiming at finding tractable cases. As is known, CSPs (studied in Chapter 3) have been subject of intense investigation and is an on-going research area– it includes, as seen in Chapter 3, dichotomies in terms of algebraic properties of relational structures. In this regard, our approach consists of reframing the complexity of answering OMQs in terms of CSPs, and vice-versa.

The "back-and-forth" framework that, as we found out, achieves this, takes advantage of the BMDP introduced in the previous section and is developed in the following way. Informally, we

- show that every OMQ $Q = (\mathcal{T}, q)$ over $\mathcal{D}$ that enjoys the BMDP can be reduced to a constraint satisfaction problem $\mathsf{CSP}_c(\Gamma)$ where Γ is defined solely by the patterns of formulae over $\mathcal{D}$ that occur in Q.
- to sharpen the link between OMQ answering and CSP further, show a converse polynomial reduction of CSPs with constants to the complement of answering OMQs with the BMDP.

The first reduction is used to transfer results from CSPs to OMQ answering. This converse reduction, in contrast, can be used to transfer NP-hardness results from the CSP world to co-NP-hardness results for OMQ answering. Both are based on the datatype atoms that occur in OMQs.

5.4.1 Datatype patterns

We describe such datatype atoms using the notion of the *datatype pattern* of an OMQ Q. The *datatype pattern* of Q is defined as $\mathsf{dtype}(Q) = (\theta_{\mathcal{T}}, \theta_q)$, where $\theta_{\mathcal{T}}$ is the set of all PP formulae that occur in qualified attribute restrictions of $\mathcal{T}$, and θ_q contains, for each disjunct q' of q, the conjunction of all atoms of q' over $\mathcal{D}$. Note that $\theta_{\mathcal{T}}$ is a set of PP formulae with constants and a single free variable, whereas θ_q is a set of PP formulae without constants. We refer to datatype patterns of OMQs over $\mathcal{D}$ as *datatype patterns over* $\mathcal{D}$.

Example 5.4.1. Let $\mathcal{T} = \{A \sqsubseteq \exists U_1.1 \leq x \wedge x \leq 3\}$ be a $\mathcal{D}$-TBox, for $\mathcal{D} = (\mathbb{Q}, \leq)$, and let q be the UCQ over $\mathcal{D}$ given by $q_1(x), q_2(x)$, where

$$q_1(x) \leftarrow \textstyle\bigwedge_{i=1}^{3} U_i(x, u_i) \wedge u_1 \leq u_2 \wedge u_1 \leq u_3,$$
$$q_2(x) \leftarrow U_1(x, u_1') \wedge U_2(x, u_2') \wedge u_2' \leq u_1'.$$

Then $\mathsf{dtype}(\mathcal{T}, q) = (\theta_{\mathcal{T}}, \theta_q)$, where $\theta_{\mathcal{T}} = \{1 \leq x \wedge x \leq 3\}$ and

$$\theta_q = \{u_1 \leq u_2 \wedge u_1 \leq u_3, u_2' \leq u_1'\}.$$

Now, with each datatype pattern $\theta = (\theta_{\mathcal{T}}, \theta_q)$ over $\mathcal{D}$, where $\theta_{\mathcal{T}} = \{\varphi_1, \ldots, \varphi_m\}$ and $\theta_q = \{\psi_1, \ldots, \psi_n\}$, we associate the constraint language

$$\Gamma_\theta = (\mathrm{dom}(\mathcal{D}), R_{\varphi_1}, \ldots, R_{\varphi_m}, R_{\neg\psi_1}, \ldots, R_{\neg\psi_n}),$$

where for each formula $\varphi(x_1, \ldots, x_k)$ over $\mathcal{D}$, we let $R_\varphi = \{\bar{a} \in \mathrm{dom}(\mathcal{D})^k \mid \mathcal{D} \models \varphi(\bar{a})\}$.

5.4.2 Transfer result

We first show the fact that satisfiability of ABoxes w.r.t. TBoxes can be reduced to a CSP_c in polynomial time.

Lemma 5.4.2. *Let $\mathcal{T}$ be a Horn-$\mathcal{L}(\mathcal{D})$-TBox in normal form where $\theta_{\mathcal{T}} = \{\varphi_1, \ldots, \varphi_n\}$. Then satisfiability of $\mathcal{D}$-ABoxes $\mathcal{A}$ w.r.t. $\mathcal{T}$ is polynomially reducible to*

$$\mathsf{CSP}_c(\mathrm{dom}(\mathcal{D}), R_{\varphi_1}, \ldots, R_{\varphi_n}).$$

Proof. Let $(\mathcal{T}, \mathcal{A})$ be a Horn-$\mathcal{L}(\mathcal{D})$-KB with $\mathcal{T}$ in normal form where $\theta_{\mathcal{T}} = \{\varphi_1, \ldots, \varphi_n\}$. We show that one can reduce satisfiability of $\mathcal{A}$ w.r.t. $\mathcal{T}$ to $\mathsf{CSP}_c(\mathrm{dom}(\mathcal{D}), R_{\varphi_1}, \ldots, R_{\varphi_n})$ in polynomial time. This is done by modifying the chase procedure so as to obtain an algorithm which outputs unsatisfiable or constructs an instance Φ_{sat} of

$$\mathsf{CSP}_c(\mathrm{dom}(\mathcal{D}), R_{\varphi_1}, \ldots, R_{\varphi_n})$$

1. If $A \sqsubseteq \bot \in \mathcal{T}$ and $A(a) \in S$, then terminate and output unsatisfiable;
2. If $C \sqsubseteq A \in \mathcal{T}$ and $C(a) \in S$ and $A(a) \notin S$, then add $A(a)$ to S.
3. If $C \sqsubseteq \exists r.A \in \mathcal{T}$ and $C(a) \in S$ and there is no $r(a, c) \in S$ with $A(c) \in S$, then add $r(a, c_{\exists r})$ and $A(c_{\exists r})$ to S, with provisos $(\star)$ and $(\dagger)$;
4. If $C \sqsubseteq \forall r.A \in \mathcal{T}$ and $C(a) \in S$ and $r(a, b) \in S$ but $A(b) \notin S$, then add $A(b)$ to S.
5. If $C \sqsubseteq \exists U \in \mathcal{T}$ and $C(a) \in S$ and there is no data value or data null v with $U(c, v) \in S$ then add $U(a, u)$ to S and set $Z(u) = \emptyset$, where u is a fresh data null.
6. If $C \sqsubseteq \exists U.\varphi \in \mathcal{T}$ and $C(a) \in S$ and there is no data value or data null v with $U(c, v) \in S$ and $\varphi \in Z(v)$, then add $U(a, u)$ to S and set $Z(u) = \{\varphi(u)\}$, where u is a fresh data null.
7. If $C \sqsubseteq \forall U.\varphi \in \mathcal{T}$ and $C(a) \in S$ and $U(c, u) \in S$ for u a data value or data null such $\varphi \notin Z(u)$, then add φ to $Z(u)$.
8. If $r_1 \sqsubseteq r_2 \in \mathcal{T}$ and $r_1(a, u) \in S$ and $r_2(a, u) \notin S$, then add $r_2(a, b)$ to S.
9. If $U_1 \sqsubseteq U_2 \in \mathcal{T}$ and $U_1(a, u) \in S$ and $U_2(a, u) \notin S$, then add $U_2(a, u)$ to S.

Figure 5.8: Modified rules (terminating procedure) for Lemma 5.4.2

that is satisfiable iff $\mathcal{A}$ is satisfiable w.r.t. $\mathcal{T}$.

Recall the chase steps described in Figures 2.1 and 2.3. Notice that the chase does not terminate if, and only if, a chase step via Rule 3 (for a fixed axiom) occurs an infinite number of times. To see this, recall Example 2.6.18. Let $\mathcal{T} := \{A \sqsubseteq \exists r.A\}$ and $\mathcal{A} := \{A(a)\}$. We use nulls from a set $\{c_0, c_1, \ldots\}$. Thus

$$\text{chase}(\mathcal{T}, \mathcal{A}) = \{A(a), r(a, c_0)\} \cup \{A(c_i) \mid i \in \mathbb{N}\} \cup \{r(c_i, c_{i+1}) \mid i \in \mathbb{N}\}.$$

So any combination of TBox and ABox that contains axioms and assertions of the form above does not terminate. We now change the chase procedure, using the rules in Figure 5.8 instead.

The proviso $(\star)$ in Figure 5.8 is that when Rule 3 is triggered, we only add a *fresh* individual $c_{\exists r}$ if no axiom containing $\exists r$ on the right-hand side was applied earlier in a chase step. Otherwise $c_{\exists r}$ is reused. $(\dagger)$ Even though it does not affect termination, we also provide that the assertion $A(c_{\exists r})$ is not introduced if S already contains it: if $C \sqsubseteq \exists r.A,\ C' \sqsubseteq \exists r.A \in \mathcal{T}$ with $C(a),\ C'(a') \in S$, when Rule 3 is applied to $C \sqsubseteq \exists r.A$ and $C(a)$ we already have $A(c_{\exists r}) \in S$.

Now given the modified chase it is clear that we obtain a polynomial-time algorithm for computing S as needed. Termination can occur either because Rule 1 is applied, so

that the output is unsatisfiable; or because at some step no more rules apply. In the latter case we construct an instance Φ_{sat} of $\mathsf{CSP}_c(\text{dom}(\mathcal{D}), R_{\varphi_1}, \ldots, R_{\varphi_n})$ as

$$\Phi_{\text{sat}} = \exists \bar{u} \bigwedge_{i=1}^{k} \bigwedge_{\varphi \in Z^{\mathcal{I}}(u_i)} R_\varphi(u_i)$$

where $u_1, \ldots, u_k$ is a repetition-free enumeration of all data nulls that occur in the chase. It is readily checked that $(\mathcal{T}, \mathcal{A})$ is satisfiable iff $(\text{dom}(\mathcal{D}), R_{\varphi_1}, \ldots, R_{\varphi_n}) \models \Phi_{\text{sat}}$.

□

Theorem 5.4.3. *Let θ be a datatype pattern over $\mathcal{D}$.*

If $Q = (\mathcal{T}, q)$ is an OMQ over $\mathcal{D}$ with $\mathsf{dtype}(Q) = \theta$ and Q enjoys the BMDP, then evaluating Q is polynomially reducible to the complement of $\mathsf{CSP}_c(\Gamma_\theta)$.

Conversely, there is a rooted OMQ Q over $\mathcal{D}$ with $\mathsf{dtype}(Q) = \theta$ such that the complement of $\mathsf{CSP}_c(\Gamma_\theta)$ is polynomially reducible to evaluating Q.

Proof. Let $\theta = (\theta_{\mathcal{T}}, \theta_q)$, where $\theta_{\mathcal{T}} = \{\varphi_1, \ldots, \varphi_m\}$ and $\theta_q = \{\psi_1, \ldots, \psi_n\}$. Assume Q enjoys the BMDP, and that q is given as $q_1(\bar{x}), \ldots, q_n(\bar{x})$. Let $\mathcal{A}$ be a $\mathcal{D}$-ABox and let $\bar{c}$ be a tuple of individual names and data values of the same length as $\bar{x}$. By Lemma 5.4.2, satisfiability of $\mathcal{A}$ relative to a given Horn-$\mathcal{L}(\mathcal{D})$-TBox $\mathcal{T}$ is polynomially reducible to $\mathsf{CSP}_c(\text{dom}(\mathcal{D}), R_{\varphi_1}, \ldots, R_{\varphi_m})$. Thus, we can assume that $\mathcal{A}$ is satisfiable relative to $\mathcal{T}$. Consider the pre-model $\mathcal{I} := \text{can}^d(\mathcal{T}, \mathcal{A})$, where d is an integer that satisfies the BMPD but is independent of $\mathcal{A}$. A *pre-match* of q_i in $\mathcal{I}$ is a match of the abstract part of q_i (i.e. q_i stripped off of all atoms over $\mathcal{D}$) in $\mathcal{I}$. Let X_i be the set of all pre-matches μ of q_i in $\mathcal{I}$ with $\mu(\bar{x}) = \bar{c}$, and let $\bar{u} = u_1, \ldots, u_k$ be a repetition-free enumeration of all the data nulls that occur in the image of some pre-match in $X_1 \cup \cdots \cup X_n$. Then the following is an instance of $\mathsf{CSP}_c(\Gamma_\theta)$:

$$\Phi := \exists \bar{u} \left(\bigwedge_{i=1}^{k} \bigwedge_{\varphi \in Z^{\mathcal{I}}(u_i)} R_\varphi(u_i) \wedge \bigwedge_{i=1}^{n} \bigwedge_{\mu \in X_i} R_{\neg\psi_i}(\mu(\bar{z}_i)) \right),$$

where $u_1, \ldots, u_k$ are identified with individual variables in Φ. We check that $\mathcal{T}, \mathcal{A} \not\models q(\bar{c})$ iff $\Gamma_\theta \models \Phi$. For simplicity, assume q has a single disjunct and that X is not empty. Assume that $\Gamma_\theta \models \Phi$. It is clear, by the construction of $R_{\neg\psi}$ (which contains all values in $\text{dom}(\mathcal{D})$ that refute some atom in ψ), that for all pre-matches μ in X, there exists an assignment α of values in $\text{dom}(\mathcal{D})$ to the variables in the datatype pattern part ψ of q that will make some atom of ψ false in $\mathcal{D}$. We use α as a completion function for $\mathcal{I}$. Using the definition of the BMDP, which by assumption is a property of Q, it can be seen that we obtain $(\mathcal{T}, \mathcal{A}) \not\models q(\bar{c})$. The other direction is similar.

For the converse, we encode each instance Φ of $\mathsf{CSP}_c(\Gamma_\theta)$ by an ABox $\mathcal{A}_\Phi$ as follows. We use a distinguished individual a_Φ to denote the root of $\mathcal{A}_\Phi$. For each atom $\alpha = R_{\neg\psi_i}(x_1, \ldots, x_k)$ in Φ there is an individual b_α connected to a_Φ via an assertion $r_i(a_\Phi, b_\alpha)$. For each $j \in \{1, \ldots, k\}$, this individual is connected to individual c_{x_j} via an assertion $s_j(b_\alpha, c_{x_j})$. Finally, for each variable x that occurs in Φ we include the assertion $A(c_x)$;

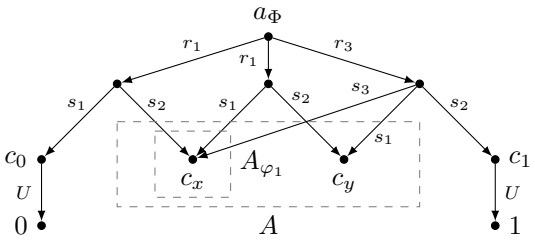

Figure 5.9: ABox $\mathcal{A}_\Phi$ from the proof of Theorem 5.4.3

if $R_{\varphi_i}(x)$ occurs in Φ we additionally include $A_{\varphi_i}(c_x)$. Furthermore, for each element $u \in \mathrm{dom}(\mathcal{D})$ that occurs in Φ we include the assertion $U(c_u, u)$. Figure 5.9 illustrates this construction for the case $m = 1$, $n = 3$, and Φ being

$$\exists x, y \, (R_{\varphi_1}(x) \wedge R_{\neg\psi_1}(0, x) \wedge R_{\neg\psi_1}(x, y) \wedge R_{\neg\psi_3}(y, 1, x)) \, .$$

Let $\mathcal{T} = \{A \sqsubseteq \exists U\} \cup \{A_{\varphi_i} \sqsubseteq \forall U.\varphi_i \mid 1 \leq i \leq m\}$ and let q be the UCQ given by $q_1(x), \ldots, q_n(x)$, where $q_i(x)$ is

$$r_i(x, y) \wedge \bigwedge_{1 \leq j \leq k} (s_j(y, z_j) \wedge U(z_j, u_j)) \wedge \psi_i(u_1, \ldots, u_k)$$

and k is the number of free variables of ψ_i. Clearly, $Q = (\mathcal{T}, q)$ is a rooted OMQ over $\mathcal{D}$ with $\mathsf{dtype}(Q) = \theta$. It is straightforward to verify that $\Gamma_\theta \models \Phi$ if and only if $\mathcal{T}, \mathcal{A}_\Phi \not\models q(a_\Phi)$. □

Note that the TBoxes constructed in the converse direction of the proof of Theorem 5.4.3 are very simple.

Corollary 5.4.4. *Let $\mathcal{D} \in \{(\mathbb{Z}, \neq), (\mathbb{Q}, \neq), (\mathbb{Z}, <), (\mathbb{Q}, <), (\mathbb{Z}, \leq)\}$. Then evaluating OMQs over $\mathcal{D}$ with the BMDP is in* co-NP*.*

Proof. Immediately from Theorem 5.4.3. □

In the next chapter we show how Theorem 5.4.3 can be used to understand the OMQs with the BMDP whose query evaluation problem is not only in co-NP but in PTIME.

6 Instantiations

6.1 Introduction

The previous section concludes with a general framework for transferring results from the CSP world to OMQ answering and back again.

Here we instantiate it for OMQ answering over all finite datatypes, as well as *numerical datatypes* typically used in applications: $(\mathbb{Q}, \leq), (\mathbb{R}, \leq)$ and a family of datatypes with a first order definition in $(\mathbb{Z}, succ)$.

We get a dichotomy result for all finite datatypes. We also obtain results for the numerical datatypes above based on the datatype patterns θ occuring in OMQs. In the case of $(\mathbb{Q}, \leq), (\mathbb{R}, \leq)$ we provide a straightforward syntactical criterion which enables one to decide, given θ, whether any OMQ with θ as datatype pattern can be answered in polynomial time; otherwise an intractable OMQ can be constructed which contains θ. As for the datatypes based on $(\mathbb{Z}, succ)$, we prove a basic dichotomy for the datatype $(\mathbb{Z}, \overline{succ})$, the complement of $(\mathbb{Z}, succ)$, as well as a basic dichotomy for a restricted form of datatype patterns over a large family of datatypes with a first order definition in $(\mathbb{Z}, succ)$.

Some of those results are quite involved and make essential use of the notions and theorems introduced in detail in Chapter 3.

6.2 All finite datatypes

We now formulate a general dichotomy result for query answering over Horn-$\mathcal{L}(\mathcal{D})$-KBs where $\mathcal{D}$ is finite, that is, $|\operatorname{dom}(\mathcal{D})| < \omega$.

Recall that for finite domain templates Γ, by Proposition 3.2.12 we have $\mathsf{CSP}(\Gamma) = \mathsf{CSP}(\Gamma')$ where Γ' is the core of Γ.

We show:

Theorem 6.2.1. *For all finite templates* Γ, $\mathsf{CSP}_c(\Gamma)$ *is either in* PTIME *or* NP*-hard.*

Proof. Assume Γ is given. Then, $\mathsf{CSP}_c(\Gamma)$ is polynomially equivalent to the problem $\mathsf{CSP}(\Gamma')$, where Γ' is the template obtained from Γ by adding predicates P_a, a in Γ, to the signature of Γ and $P_a(a)$, a in Γ, to Γ. (Thus, Γ' is a precoloured template.) By Theorem 3.2.15, $\mathsf{CSP}(\Gamma')$ is either in PTIME or NP-hard. □

Now immediately from Theorem 5.4.3 and Theorem 6.2.1 we obtain the dichotomy for query answering over finite datatypes:

Theorem 6.2.2. *Let* θ *be a datatype pattern over* $\mathcal{D}$*, where* $\operatorname{dom}(\mathcal{D})$ *has finite cardinality. Then either (a) or (b) holds, where*

(a) *for all OMQs* $Q = (\mathcal{T}, q)$ *with the BMDP with* $\mathsf{dtype}(Q) = \theta$*, evaluating* Q *is in* PTIME*;*

(b) *there is a rooted OMQ* $Q = (\mathcal{T}, q)$ *with* $\mathsf{dtype}(Q) = \theta$ *such that evaluating* Q *is* co-NP*-complete.*

6.3 Dense linear orders

Now let $\mathcal{D} = (\mathbb{Q}, \leq)$. We will use the classification result for $(\mathbb{Q}, <)$ by Bodirsky and Kára, Theorem 3.3.11, to obtain a classification for OMQ answering over $\mathcal{D}$. The results will hold all the same for $(\mathbb{R}, \leq)$ or any dense linear order. For that purpose we introduce various definitions and prove some preliminary results.

To simplify the presentation, we assume w.l.o.g. that for all datatype patterns $\theta = (\theta_{\mathcal{T}}, \theta_q)$ over $(\mathbb{Q}, \leq)$ the formulas in θ_q are *acyclic*, that is, they contain no subformulas of the form $x_1 \leq x_2 \wedge x_2 \leq x_3 \wedge \cdots \wedge \leq x_n \leq x_1$ for $n \geq 1$. Cycles as such occurring in a formula in θ_q can always be eliminated by removing all their atoms and replacing each x_i with x_1.

We use *min-pattern* and *max-pattern* to refer to formulas of the form $x_0 \leq x_1 \wedge x_0 \leq x_2 \wedge \cdots \wedge x_0 \leq x_k$ and $x_1 \leq x_0 \wedge x_2 \leq x_0 \wedge \cdots \wedge x_k \leq x_0$, for $k \geq 0$, respectively.

We now state the main result of this section.

Theorem 6.3.1. *Let* $\theta = (\theta_{\mathcal{T}}, \theta_q)$ *be a datatype pattern over* $(\mathbb{Q}, \leq)$*, where* $\theta_q = \{\psi_1, \ldots, \psi_n\}$*.*

If each ψ_i *is a min-pattern or each* ψ_i *is a max-pattern, then evaluating OMQs* Q *over* $(\mathbb{Q}, \leq)$ *with* $\mathsf{dtype}(Q) = \theta$ *and the BMDP is in* PTIME*.*

Otherwise, there is a rooted OMQ Q *over* $(\mathbb{Q}, \leq)$ *with* $\mathsf{dtype}(Q) = \theta$ *such that evaluating* Q *is* co-NP*-complete.*

This simple and purely syntactic characterization of the tractable cases makes make it very easy to verify whether evaluating a given OMQ over $(\mathbb{Q}, \leq)$ with the BMDP is either guaranteed to be tractable or possibly coNP-complete. For instance, Theorem 6.3.1 implies that the OMQ $(\mathcal{T}, q)$ in Example 5.4.1 and in general all OMQs over $(\mathbb{Q}, \leq)$ that have the same datatype pattern and enjoy the BMDP can be evaluated in PTIME. On the other hand, if we consider the datatype pattern that has $z_1 \leq z_2 \wedge z_2 \leq z_3$ in place of $z_1 \leq z_2 \wedge z_1 \leq z_3$, then there are rooted OMQs over $(\mathbb{Q}, \leq)$ with that datatype pattern for which evaluation is coNP-complete.

We now introduce definitions and show results needed to prove Theorem 6.3.1.

Polymorphisms of the weak linear order on the rational numbers From now on we will let $\mathcal{F}$ be the set containing the functions ll, min, mi and mx. Recall Definition 3.3.8 and let *dual-*$\mathcal{F}$ be the set containing all the duals of the functions in $\mathcal{F}$. We will later need the following easy result regarding preservation of the weak linear order $<$ on $\mathbb{Q}$ by functions in $\mathcal{F} \cup$ *dual-*$\mathcal{F}$. Recall that given a tuple $t = (a_1, \ldots, a_n)$ we use $t[i]$ to denote the i-th entry of t.

Proposition 6.3.2.

1. *$<$ is preserved by min, mi, mx, ll and their duals.*
2. *$<$ is not preserved by any constant function.*

Proof. We only consider preservation under *min*, *mi*, *mx*, *ll* and constant functions. The proofs for the duals of *min*, *mi*, *mx*, *ll* are similar.

Preservation under min: Let $a_1 < a_2$ and $b_1 < b_2$. We have to show that $c_1 < c_2$, where $c_i := min(a_i, b_i)$. If $c_1 = a_1$, then $c_1 = a_1 < a_2$ and $c_1 = a_1 \leq b_1 < b_2$, thus $c_1 < c_2$. Similarly, if $c_1 = b_1$, then $c_1 = b_1 \leq a_1 < a_2$ and $c_1 = b_1 < b_2$, thus $c_1 < c_2$. This shows that $<$ is preserved under *min*.

Preservation under mi: Let $a_1 < a_2$ and $b_1 < b_2$. We have to show that $c_1 < c_2$, where $c_i := mi(a_i, b_i)$. Since $<$ is preserved by *min*, we have $\min(a_1, b_1) < \min(a_2, b_2)$. Hence, $c_1 = mi(a_1, b_1) < mi(a_2, b_2) = c_2$. Altogether, this shows that $<$ is preserved under *mi*.

Preservation under mx: Let $a_1 < a_2$ and $b_1 < b_2$. We have to show that $c_1 < c_2$, where $c_i := mx(a_i, b_i)$. Since $<$ is preserved by *min*, we have $\min(a_1, b_1) < \min(a_2, b_2)$. This implies $c_1 = mx(a_1, b_1) < mx(a_2, b_2) = c_2$. Altogether, we have shown that $<$ is preserved under *mx*.

Preservation under ll: Let $a_1 < a_2$ and $b_1 < b_2$. We have to show that $ll(a_1, b_1) < ll(a_2, b_2)$. If $a_1 \leq 0$, then $a_1 < a_2$ immediately yields $ll(a_1, b_1) < ll(a_2, b_2)$. Now suppose that $a_1 > 0$. Since $a_1 < a_2$, we also have $a_2 > 0$. But then, $b_1 < b_2$ immediately yields $ll(a_1, b_1) < ll(a_2, b_2)$.

Non-preservation under constant functions: For a contradiction, suppose that $<$ is preserved under a constant function $f\colon \mathbb{Q}^k \to \{c\}$. Take any $a_1 < b_1, \dots, a_k < b_k$. Since f preserves $<$, we obtain $c = f(a_1, \dots, a_k) < f(b_1, \dots, b_k) = c$, which is impossible. □

A basic dichotomy We now combine Theorem 5.4.3 and Theorem 3.3.11 to obtain a basic dichotomy for evaluating OMQs over $(\mathbb{Q}, \leq)$ based on their datatype patterns. This is an intermediate step for the proof of the main theorem.

Theorem 6.3.3. *Let $\theta = (\theta_{\mathcal{T}}, \theta_q)$ be a datatype pattern over $(\mathbb{Q}, \leq)$, where $\theta_q = \{\psi_1, \dots, \psi_n\}$.*

1. *If some function $f \in \mathcal{F} \cup$ dual-$\mathcal{F}$ preserves each $R_{\neg\psi_i}$, then evaluating OMQs Q over $(\mathbb{Q}, \leq)$ with $\mathsf{dtype}(Q) = \theta$ and the BMDP is in* PTIME.
2. *Otherwise, there is a rooted OMQ Q over $(\mathbb{Q}, \leq)$ with $\mathsf{dtype}(Q) = \theta$ such that evaluating Q is coNP-complete.*

Proof. 1. Let Q be an OMQ over $(\mathbb{Q}, \leq)$ with $\mathsf{dtype}(Q) = \theta$ that enjoys the BMDP. By Theorem 5.4.3, evaluating Q is polynomially reducible to the complement of $\mathsf{CSP}_c(\Gamma_\theta)$. We show that $\mathsf{CSP}_c(\Gamma_\theta)$ is polynomially reducible to $\mathsf{CSP}(\Gamma)$, where

$$\Gamma = (\mathbb{Q}, <, \leq, R_{\neg\psi_1}, \dots, R_{\neg\psi_n}),$$

and that $\mathsf{CSP}(\Gamma)$ is in PTIME. This suffices to establish the first part of the theorem.

To show that $\mathsf{CSP}_c(\Gamma_\theta)$ is polynomially reducible to $\mathsf{CSP}(\Gamma)$, consider an instance Φ of $\mathsf{CSP}_c(\Gamma_\theta)$. Replacing in Φ each atom of the form $R_\varphi(x)$, $\varphi \in \theta_\mathcal{T}$, by $\varphi(x)$ yields an instance Φ' of $\mathsf{CSP}_c(\mathbb{Q}, \leq, R_{\neg\psi_1}, \dots, R_{\neg\psi_n})$ with

$$\Gamma_\theta \models \Phi \iff (\mathbb{Q}, \leq, R_{\neg\psi_1}, \dots, R_{\neg\psi_n}) \models \Phi'.$$

Next, we eliminate all constants from Φ'. Let $c_1 < \cdots < c_k$ be the sequence of all elements of $\mathbb{Q}$ that occur as constants in Φ'. We simulate these constants by making each c_i an existentially quantified variable and adding constraints $c_i < c_{i+1}$, for each $i \in \{1, \dots, k-1\}$, to ensure that any assignment preserves the relative order of these constants:

$$\Phi'' = \exists c_1 \cdots \exists c_k \left(\Phi' \wedge c_1 < c_2 \wedge \cdots \wedge c_{k-1} < c_k\right).$$

We thus obtain an instance Φ'' of $\mathsf{CSP}(\Gamma)$. We claim:

$$(\mathbb{Q}, \leq, R_{\neg\psi_1}, \dots, R_{\neg\psi_n}) \models \Phi' \iff \Gamma \models \Phi''.$$

The direction from left to right follows from the construction of Φ''. For the converse, assume $\Gamma \models \Phi''$. Let g be an assignment of rational numbers to the existential variables in Φ'' that satisfies the quantifier-free part of Φ'' in Γ. Pick any automorphism α of $(\mathbb{Q}, <)$ such that $\alpha(g(c_i)) = c_i$ for all $i \in \{1, \dots, k\}$. Then $\alpha \circ g$ also satisfies the quantifier-free part of Φ'' in Γ. Since $\alpha \circ g$ interprets each c_i by itself, this implies $(\mathbb{Q}, \leq, R_{\neg\psi_1}, \dots, R_{\neg\psi_n}) \models \Phi'$. The sentence Φ'' can clearly be computed in polynomial time on input Φ. Altogether, we have shown that $\mathsf{CSP}_c(\Gamma_\theta)$ is polynomially reducible to $\mathsf{CSP}(\Gamma)$.

It remains to show that $\mathsf{CSP}(\Gamma)$ is in PTIME. This is trivial if each of the ψ_i is empty. Otherwise, at least one of the ψ_i is non-empty, which implies that f is not a constant function. Since $<$ and $\leq$ are preserved under any non-constant function in $\mathcal{F} \cup \mathit{dual}\text{-}\mathcal{F}$ (Proposition 6.3.2), we know Γ is preserved under f and thus $\mathsf{CSP}(\Gamma)$ is in PTIME (by Theorem 3.3.11).

2. By the theorem's hypothesis,

$$\Gamma = (\mathbb{Q}, R_{\neg\psi_1}, \dots, R_{\neg\psi_n})$$

is not preserved by any function in $\mathcal{F} \cup \mathit{dual}\text{-}\mathcal{F}$, which implies that $\mathsf{CSP}(\Gamma)$ is NP-complete (by Theorem 3.3.11). Since Γ is a substructure of Γ_θ, $\mathsf{CSP}_c(\Gamma_\theta)$ is also NP-complete. By Theorem 5.4.3, there is a rooted OMQ Q over $(\mathbb{Q}, \leq)$ with $\mathsf{dtype}(Q) = \theta$ such that the complement of $\mathsf{CSP}_c(\Gamma_\theta)$ is polynomially reducible to evaluating Q. This concludes the proof of the second part of the theorem. □

Structure of 'tractable' datatype patterns Theorem 6.3.3 establishes a basic P/coNP-dichotomy for evaluating OMQs over $(\mathbb{Q}, \leq)$ that enjoy the BMDP. The tractable cases of this dichotomy are characterized in terms of preservation properties of the relations $R_{\neg\psi}$, where ψ is a formula in the UCQ part θ_q of the datatype pattern. To obtain a

purely syntactic characterization of these tractable cases, we here analyze the structure of formulae $\psi \in \theta_q$ such that $R_{\neg\psi}$ is preserved under one of the functions in $\mathcal{F} \cup \textit{dual-}\mathcal{F}$. This analysis is one of the main ingredients for our proof of Theorem 6.3.1.

Here we prove two auxilliary lemmas. The first lemma is straightforward but provides a useful tool for the proof of the second lemma, which is the core of the analysis. For a tuple $t = (t_1, \ldots, t_n) \in \mathbb{Q}^n$ and an integer $i \in \{1, \ldots, n\}$, we write $t[i] = t_i$.

Lemma 6.3.4. *Consider a function $f\colon \mathbb{Q}^2 \to \mathbb{Q}$ and elements $a_1, \ldots, a_4, b_1, \ldots, b_4 \in \mathbb{Q}$ such that*

$$f(a_1, b_1) \geq \cdots \geq f(a_4, b_4).$$

Let $1 \leq i_1 \leq i_2 \leq i_3 \leq i_4 \leq n$, and suppose that $(a_j, b_j) = (a_{j'}, b_{j'})$ if $i_j = i_{j'}$. Then, there are tuples $t_1, t_2 \in \mathbb{Q}^n$ such that $t_1[i_j] = a_j$ and $t_2[i_j] = b_j$ for all $j \in \{1, 2, 3, 4\}$, and

$$f(t_1[1], t_2[1]) \geq \cdots \geq f(t_1[n], t_2[n]).$$

Proof. Define $t_1, t_2 \in \mathbb{Q}^n$ such that for all $p \in \{1, \ldots, n\}$, we have that

$$t_1[p] := \begin{cases} a_1, & \text{if } p \leq i_1 \\ a_2, & \text{if } i_1 < p \leq i_2 \\ a_3, & \text{if } i_2 < p \leq i_3 \\ a_4, & \text{if } i_3 < p \end{cases}$$

and

$$t_2[p] := \begin{cases} b_1, & \text{if } p \leq i_1 \\ b_2, & \text{if } i_1 < p \leq i_2 \\ b_3, & \text{if } i_2 < p \leq i_3 \\ b_4, & \text{if } i_3 < p. \end{cases}$$

Clearly, $t_1[i_j] = a_j$ and $t_2[i_j] = b_j$ for all $j \in \{1, 2, 3, 4\}$. From the construction of t_1 and t_2 and the properties of $a_1, \ldots, a_4, b_1, \ldots, b_4$, it immediately follows that $f(t_1[1], t_2[1]) \geq \cdots \geq f(t_1[n], t_2[n])$. □

Consider a datatype pattern $\theta = (\theta_{\mathcal{T}}, \theta_q)$ over $(\mathbb{Q}, \leq)$. What is the structure of formulae $\psi \in \theta_q$ for which $R_{\neg\psi}$ is preserved by a function in $\mathcal{F} \cup \textit{dual-}\mathcal{F}$? Since ψ is acyclic by assumption, the negation of ψ that defines $R_{\neg\psi}$ is equivalent to a disjunction Ψ of atomic formulae $x < y$, with x and y variables, such that the directed graph with the variables of Ψ as its vertices, and edges (y, x) for each atomic formula $x < y$ of Ψ is acyclic. We call such formulae Ψ *acyclic disjunctive* formulae.

Lemma 6.3.5. *Let $R \subseteq \mathbb{Q}^n$ be defined by an acyclic disjunctive formula Ψ over $(\mathbb{Q}, <)$. Let $f \in \{min, mi, mx\}$.*

1. *If R is preserved under f, then for every two disjuncts $x_i < x_j$ and $x_{i'} < x_{j'}$ of Ψ we have $j = j'$.*

2. *If R is preserved under dual-f, then for every two disjuncts $x_i < x_j$ and $x_{i'} < x_{j'}$ of Ψ we have $i = i'$.*

Proof. We will only consider the case that R is preserved under f. The dual of f can be dealt with similarly.

Let R be preserved under f, and let $x_i < x_j$ and $x_{i'} < x_{j'}$ be disjuncts of Ψ. For the sake of contradiction, assume $j \neq j'$. Without loss of generality, we assume that $j < j'$. We are going to construct tuples $t_1, t_2 \in R$ such that $t_3 = f(t_1, t_2) \notin R$.

Since Ψ is acyclic, we can assume that the variables $x_1, \ldots, x_n$ are topologically sorted, i.e., if $x_p < x_q$ is an atom of Ψ, then $p < q$. In particular, $i < j$ and $i' < j'$. By the topological ordering, any tuple $t \in \mathbb{Q}^n$ with $t[i] < t[j]$ or $t[i'] < t[j']$ belongs to R, whereas no tuple $t \in \mathbb{Q}^n$ with $t[1] \geq \cdots \geq t[n]$ can belong to R. We will use these properties to obtain the desired tuples t_1 and t_2.

We distinguish the following three cases:

Case 1 ($i' \leq i$): In this case, we have $i' \leq i < j < j'$. Let $a_i, a_{i'}, a_j, a_{j'} \in \mathbb{Q}$ and $b_i, b_{i'}, b_j, b_{j'} \in \mathbb{Q}$ be defined by $a_i = a_{i'} = b_i = b_{i'} = 2$, $b_j = 1$, $a_{j'} = 0$, and $a_j = b_{j'} = 3$; see Figure 6.1 for an illustration. We then have $a_i < a_j$ and $b_{i'} < b_{j'}$. It is also

a_i	$a_{i'}$	a_j	$a_{j'}$
2	2	3	0
2	2	1	3
b_i	$b_{i'}$	b_j	$b_{j'}$

Figure 6.1: Choice of $a_i, a_{i'}, a_j, a_{j'} \in \mathbb{Q}$ and $b_i, b_{i'}, b_j, b_{j'} \in \mathbb{Q}$ in Case 1.

straightforward to verify that $f(a_i, b_i) = f(a_{i'}, b_{i'}) > f(a_j, b_j) > f(a_{j'}, b_{j'})$. Indeed, $min(a_i, b_i) = min(a_{i'}, b_{i'}) = 2$, $min(a_j, b_j) = 1$, and $min(a_{j'}, b_{j'}) = 0$, so the claim is true for $f = min$. For mi and mx, the claim is true, since $min(x, y) > min(x', y')$ implies $mi(x, y) > mi(x', y')$ and $mx(x, y) > mx(x', y')$. Now, Lemma 6.3.4 implies that there are tuples $t_1, t_2 \in \mathbb{Q}^n$ such that $t_1[i] < t_1[j]$, $t_2[i'] < t_2[j']$, and $f(t_1[1], t_2[1]) \geq \cdots \geq f(t_1[n], t_2[n])$. Hence, $t_1, t_2 \in R$ and $t_3 = f(t_1, t_2) \notin R$.

Case 2 ($i < i' < j$): In this case, we have $i < i' < j < j'$. Let $a_i, a_{i'}, a_j, a_{j'} \in \mathbb{Q}$ and $b_i, b_{i'}, b_j, b_{j'} \in \mathbb{Q}$ be defined by $b_i = 3$, $a_{i'} = 2$, $b_j = 1$, $a_{j'} = 0$, $a_i = b_{i'} = 4$, and $a_j = b_{j'} = 5$; see Figure 6.2 for an illustration. We then have $a_i < a_j$ and $b_{i'} < b_{j'}$. It

a_i	$a_{i'}$	a_j	$a_{j'}$
4	2	5	0
3	4	1	5
b_i	$b_{i'}$	b_j	$b_{j'}$

Figure 6.2: Choice of $a_i, a_{i'}, a_j, a_{j'} \in \mathbb{Q}$ and $b_i, b_{i'}, b_j, b_{j'} \in \mathbb{Q}$ in Case 2.

is straightforward to verify that $f(a_i, b_i) > f(a_{i'}, b_{i'}) > f(a_j, b_j) > f(a_{j'}, b_{j'})$. Indeed, $min(a_i, b_i) = 3$, $min(a_{i'}, b_{i'}) = 2$, $min(a_j, b_j) = 1$, and $min(a_{j'}, b_{j'}) = 0$, so the claim is true for $f = min$. For mi and mx, the claim is true, since $min(x, y) > min(x', y')$ implies $mi(x, y) > mi(x', y')$ and $mx(x, y) > mx(x', y')$. Now, Lemma 6.3.4 implies that there are tuples $t_1, t_2 \in \mathbb{Q}^n$ such that $t_1[i] < t_1[j]$, $t_2[i'] < t_2[j']$, and $f(t_1[1], t_2[1]) \geq \cdots \geq f(t_1[n], t_2[n])$. Hence, $t_1, t_2 \in R$ and $t_3 = f(t_1, t_2) \notin R$.

Case 3 ($j \leq i'$): In this case, we have $i < j \leq i' < j'$. Let $a_i, a_j, a_{i'}, a_{j'} \in \mathbb{Q}$ and $b_i, b_j, b_{i'}, b_{j'} \in \mathbb{Q}$ be defined by $a_i = 2$, $b_j = b_{i'} = 1$, $a_{j'} = 0$, and $b_i = a_j = a_{i'} = b_{j'} = 3$; see Figure 6.3 for an illustration. We then have $a_i < a_j$ and $b_{i'} < b_{j'}$. It is also

a_i	a_j	$a_{i'}$	$a_{j'}$
2	3	3	0
3	1	1	3
b_i	b_j	$b_{i'}$	$b_{j'}$

Figure 6.3: Choice of $a_i, a_j, a_{i'}, a_{j'} \in \mathbb{Q}$ and $b_i, b_j, b_{i'}, b_{j'} \in \mathbb{Q}$ in Case 3.

straightforward to verify that $f(a_i, b_i) > f(a_j, b_j) = f(a_{i'}, b_{i'}) > f(a_{j'}, b_{j'})$. Indeed, $min(a_i, b_i) = 2$, $min(a_j, b_j) = min(a_{i'}, b_{i'}) = 1$, and $min(a_{j'}, b_{j'}) = 0$, so the claim is true for $f = min$. For mi and mx, the claim is true, since $min(x, y) > min(x', y')$ implies $mi(x, y) > mi(x', y')$ and $mx(x, y) > mx(x', y')$. Now, Lemma 6.3.4 implies that there are tuples $t_1, t_2 \in \mathbb{Q}^n$ such that $t_1[i] < t_1[j]$, $t_2[i'] < t_2[j']$, and $f(t_1[1], t_2[1]) \geq \cdots \geq f(t_1[n], t_2[n])$. Hence, $t_1, t_2 \in R$ and $t_3 = f(t_1, t_2) \notin R$.

Altogether, this concludes the proof. □

The following lemma is the combinatorial core of our analysis.

Lemma 6.3.6. *Let $R \subseteq \mathbb{Q}^n$ be defined by an acyclic disjunctive formula Ψ over $(\mathbb{Q}, <)$. If R is preserved by a function in $\mathcal{F} \cup$ dual-$\mathcal{F}$, then Ψ has the form $\bigvee_{i=1}^k x_i < x_0$ if $f \in \mathcal{F}$, and $\bigvee_{i=1}^k x_0 < x_i$ if $f \in$ dual-$\mathcal{F}$.*

Proof. Let $\Psi = \bigvee_{1 \leq i \leq k} y_{s_i} < y_{t_i}$. Without loss of generality, we assume that $s_i \neq t_i$ for all $i \in \{1, \ldots, k\}$, and that any two pairs $(s_p, t_p), (s_q, t_q)$ with $p \neq q$ are distinct. If $k \leq 1$, then Ψ already has the required form. It remains to consider the case that $k \geq 2$. Note that in this case f cannot be a constant function. Furthermore, if f is *ll* or *dual-ll*, then the lemma follows from [20]. In what follows, we therefore assume that $k \geq 2$ and that f is one of *min*, *mi*, *mx*, and their duals.

We distinguish the following two cases:

Case 1: f is min, mi, or mx. In this case, Lemma 6.3.5 implies that for each $j \in \{2, \ldots, k\}$ we have $t_1 = t_j$. In particular, Ψ has the form $\bigvee_{i=1}^k x_i < x_0$.

Case 2: f is the dual of min, mi, or mx. In this case, Lemma 6.3.5 implies that for each $j \in \{2, \ldots, k\}$ we have $s_1 = s_j$. This implies that Ψ has the form $\bigvee_{i=1}^k x_0 < x_i$.

Altogether, this concludes the proof of the lemma. □

The main dichotomy We are now ready to show Theorem 6.3.1.

Let $\theta = (\theta_{\mathcal{T}}, \theta_q)$ be a datatype pattern over $(\mathbb{Q}, \leq)$, where $\theta_q = \{\psi_1, \ldots, \psi_n\}$. By Theorem 6.3.3, it suffices to show that the following are equivalent:

1. Each $\psi \in \theta_q$ is a min-pattern, or each $\psi \in \theta_q$ is a max-pattern.
2. There is a function $f \in \mathcal{F} \cup \textit{dual-}\mathcal{F}$ such that each $R_{\neg\psi}$, $\psi \in \theta_q$, is preserved under f.

If each $\psi \in \theta_q$ is a min-pattern, then for each $\psi \in \theta_q$ the relation $R_{\neg\psi}$ is defined by a formula of the form $\bigvee_{i=1}^{n} x_0 > x_i$. Similarly, if each $\psi \in \theta_q$ is a max-pattern, then for each $\psi \in \theta_q$ the relation $R_{\neg\psi}$ is defined by a formula of the form $\bigvee_{i=1}^{n} x_i > x_0$. It is known [20, Proposition 3.5] that relations defined by such formulas are preserved under a function in $\mathcal{F} \cup \textit{dual-}\mathcal{F}$.

Conversely, let f be a function in $\mathcal{F} \cup \textit{dual-}\mathcal{F}$ such that each $R_{\neg\psi}$ with $\psi \in \theta_q$ is preserved under f. Let $\psi \in \theta_q$. Then, $R_{\neg\psi}$ is defined by an acyclic disjunctive formula Ψ. By Lemma 6.3.6, Ψ has the form $\bigvee_{i=1}^{n} x_i < x_0$, if $f \in \mathcal{F}$, and $\bigvee_{i=1}^{n} x_0 < x_i$, if $f \in \textit{dual-}\mathcal{F}$. This implies that each $\psi \in \theta_q$ is a min-pattern, or each $\psi \in \theta_q$ is a max-pattern.

Refinement of the complexity analysis We now refine the analysis of the datatype patterns that lead to OMQs with an evaluation problem in PTIME further by presenting a dichotomy between those that can be used in PTIME-hard OMQs and those that always lead to OMQs in NLogSpace. It turns out that the NLogSpace upper bound holds for all OMQs Q whose datatype pattern contains atomic formulas only.

Theorem 6.3.7. *Evaluating OMQs Q over $(\mathbb{Q}, \leq)$ with the BMDP and $\mathsf{dtype}(Q) = (\theta_{\mathcal{T}}, \theta_q)$ such that each formula in θ_q is of the form $x_0 \leq x_1$ is in* NLogSpace.

Proof. (Sketch) Straightforward application of Part 1 of Theorem 5.4.3, which allows us to reduce evaluating OMQs Q over $(\mathbb{Q}, \leq)$ with the BMDP where $\mathsf{dtype}(Q) = (\theta_{\mathcal{T}}, \theta_q)$, such that each formula in θ_q is of the form $x_0 \leq x_1$, to the complement of $\mathsf{CSP}_c(\mathbb{Q}, <, \leq)$. It is not difficult to see that this reduction can be carried out in logarithmic space. The fact that $\mathsf{CSP}_c(\mathbb{Q}, <, \leq)$ is in NLogSpace can be verified by noticing that it can be solved by doing a simple reachability test. □

The following result entails that the NLogSpace upper bound cannot be generalised to further tractable datatype patterns.

Theorem 6.3.8. *There is a rooted OMQ Q over $(\mathbb{Q}, \leq)$ with $\mathsf{dtype}(Q) = (\emptyset, \{x \leq y \wedge x \leq z\})$ such that evaluating Q is* PTIME*-complete.*

Proof. By Part 2 of Theorem 5.4.3 it suffices to show that $\mathsf{CSP}_c(\mathbb{Q}, R_{\neg\psi})$ is PTIME-hard, where $\psi = \{x \leq y \wedge x \leq z\}$. To this end we show that the alternating reachability problem [98, 63] is polynomially reducible to $\mathsf{CSP}_c(\mathbb{Q}, R_{\neg\psi})$. An *alternating graph* is a directed graph $G = (V, E)$ where V is the disjoint union of the set $V_\exists$ of *existential* vertices and the set $V_\forall$ of *universal* vertices. An *alternating path* from vertex x to vertex y in G exists, in short, $\mathrm{apath}_G(x, y)$ holds, if

1. $x = y$, or
2. $x \in V_\exists$ and there is a $z \in V$ with $(x, z) \in E$ and $\text{apath}_G(z, y)$ holds, or
3. $x \in V_\forall$ and for all $z \in V$ with $(x, z) \in E$, $\text{apath}_G(z, y)$ holds.

Alternating reachability is the problem to decide, given an alternating graph G and designated vertices s, t, whether $\text{apath}_G(s, t)$ holds. Alternating reachability is still PTIME-hard if we assume that G is acyclic, that all vertices have out-degree either 0 or 2, that no universal vertex has outdegree 0, that s is existential and has no incoming edge, and that t is universal and has no outgoing edge. Assume G and vertices s, t with these properties are given. We regard the set V of vertices of G as variables, take the constants $0, 1 \in \mathbb{Q}$ and construct a PP sentence $\varphi_{G,s,t}$ with constants over $(\mathbb{Q}, R_{\neg\psi})$ as the conjunction of the following formulae:

- all $R_{\neg\psi}(v, w, w')$ such that $v \in V_\exists$ and w, w' are both successors of v;
- all $R_{\neg\psi}(v, w, w)$ such that $v \in V_\forall$ and w is a successor of v,
- all $R_{\neg\psi}(v, 0, 0)$ such that $v \neq t$ has outdegree 0, and
- $R_{\neg\psi}(0, s, s)$.

Recall that $R_{\neg\psi} = \{(a, b, c) \in \mathbb{Q}^3 \mid a < b \vee a < c\}$. Then one can easily show that $\text{apath}_G(s, t)$ holds iff $(\mathbb{Q}, R_{\neg\psi}) \models \varphi_{G,s,t}$, as required. □

6.4 Integers with distance relations

Let $succ := \{(a, b) \in \mathbb{Z}^2 \mid b = a + 1\}$. Also for each $k \geq 1$ we define the relation $Dist_k = \{(a, b) \in \mathbb{Z}^2 \mid |a - b| = k\}$. Such relations are here called *distance relations*. In this section we study the complexity of OMQ answering over certain families of datatypes whose domain is $\mathbb{Z}$ and all relations are first-order definable over $(\mathbb{Z}, succ)$. First (1) we focus on the general case where all datatype patterns are allowed, and the datatype is fixed to be $(\mathbb{Z}, \overline{succ})$, the complement of $(\mathbb{Z}, succ)$. Then (2) we study the case of single atom datatype patterns (a single datatype atom in each CQ) over a large family of datatypes called "distance datatypes", whose relations satisfy a certain finiteness constraint. In this section, we assume all integers are encoded in unary.

Main classification result in CSPs All templates Γ with finite signature and a first order definition over $(\mathbb{Z}, succ)$ exhibit a complexity dichotomy. That is, $\mathsf{CSP}(\Gamma)$ is either in PTIME or NP-complete [21]. The following formulation of the dichotomy will be used throughout this section.

Theorem 6.4.1. *[From Theorem 2 in [21]] Let Γ be a template with finite signature, and a first-order definition over $(\mathbb{Z}, succ)$. Then $\mathsf{CSP}(\Gamma) = \mathsf{CSP}(\Gamma')$ where Γ' is one of the following:*

1. *a finite template. In this case,* $\mathsf{CSP}(\Gamma)$ *is in* PTIME *or* NP*-complete.*

2. *a template with a first-order definition in* $(\mathbb{Z}, =)$*, in which case* $\mathsf{CSP}(\Gamma)$ *is known to be in* PTIME *or* NP*-complete.*

3. *a template with a first-order definition in* $(\mathbb{Z}, succ)$ *in which the relation* $Dist_k$ *is PP-definable for all* $k > 0$*, in which case* $\mathsf{CSP}(\Gamma)$ *is* NP*-complete.*

4. *a template with a first-order definition in* $(\mathbb{Z}, succ)$ *containing succ. In this case,* $\mathsf{CSP}(\Gamma)$ *is either in* PTIME *or* NP*-complete.*

This theorem will be used for proving a basic dichotomy for OMQ answering over (A) the complementary datatype $(\mathbb{Z}, \overline{succ})$ and (B) a large family of datatypes we will define later, called distance datatypes. The case distinction 1-4 will be instrumental in showing such results. Before going into that we present the instantiation and the finiteness property of templates used in the second part of this section.

Transfer theorem We instantiate our main transfer theorem for datatypes with a first-order definition over $(\mathbb{Z}, succ)$.

Theorem 6.4.2. *Let* $\mathcal{D}$ *be a datatype with a first order definition in* $(\mathbb{Z}, succ)$ *and* θ *be a datatype pattern over* $\mathcal{D}$*.*

If $Q = (\mathcal{T}, q)$ *is an OMQ over* $\mathcal{D}$ *with* $\mathsf{dtype}(Q) = \theta$ *and* Q *has the BMPD, then evaluating* Q *is polynomially reducible to the complement of* $\mathsf{CSP}_c(\Gamma_\theta)$*.*

Conversely, there is a rooted OMQ Q *over* $\mathcal{D}$ *with* $\mathsf{dtype}(Q) = \theta$ *such that the complement of* $\mathsf{CSP}_c(\Gamma_\theta)$ *is polynomially reducible to evaluating* Q*.*

Locally finite templates We introduce a local finiteness property of templates over $(\mathbb{Z}, succ)$ which has a link to results on the associated CSP. The *distance degree* of a relation R is defined as follows. If there exists a greatest integer d such that there exist $x, y \in \mathbb{Z}$ occuring together in a tuple in R with $|x - y| = d$, we say that the distance degree of R is d; in this case we say that R is *bounded*. Otherwise, R is called *unbounded*. A template is termed *locally finite* if all its relations are bounded.

6.4.1 The datatype $(\mathbb{Z}, \overline{succ})$

We start by classifying the complexity of OMQ answering over $\mathcal{D} = (\mathbb{Z}, \overline{succ})$. In qualified attribute restrictions, that is, axioms of the form $C \sqsubseteq \exists U.\varphi$ and $C \sqsubseteq \forall U.\varphi$, φ is here required to be a PP formula over $\bar{\mathcal{D}}$ with one free variable x.

Remark 6.4.3. Given this particular datatype, any such PP formula is of the form $\exists \bar{z}(succ(x_1, x_2) \wedge \ldots \wedge succ(x_{n-1}, x_n))$ where $x = x_i$ for some $i \in [n]$ and $x \notin \bar{z}$. It is easy to see that x can be constrained to have at most one integer value. As a consequence we can restrict ourselves to equivalent formulas of the form $(x = d)$ with $d \in \mathbb{Z}$.

Theorem 6.4.4. *Let θ be a datatype pattern over $\mathcal{D} = (\mathbb{Z}, \overline{succ})$. Then the problem of evaluating OMQs $Q = (\mathcal{T}, q)$ with the BMDP over $\mathcal{D}$, with $\mathsf{dtype}(Q) = \theta$, is either in* PTIME *or possibly co-*NP*-hard.*

Proof. Let $Q = (\mathcal{T}, q)$ be an OMQ with the BMDP over the datatype $\mathcal{D} = (\mathbb{Z}, \overline{succ})$ with $\mathsf{dtype}(Q) = \theta = (\theta_{\mathcal{T}}, \theta_q)$, where $\theta_{\mathcal{T}} = \{\varphi_1, \ldots, \varphi_m\}$ and $\theta_q = \{\psi_1, \ldots, \psi_n\}$. By Theorem 6.4.2, we get a polynomial time reduction from evaluating Q to the complement of $\mathsf{CSP}_c(\Gamma_\theta)$ where

$$\Gamma_\theta = (\mathbb{Z}, R_{\varphi_1}, \ldots, R_{\varphi_m}, R_{\neg\psi_1}, \ldots, R_{\neg\psi_n}).$$

Fix an input Ψ to $\mathsf{CSP}_c(\Gamma_\theta)$. Now observe that as per Remark 6.4.3 each R_{φ_i} is interpreted as a singleton $\{d\}$ with $d \in \mathbb{Z}$. We call it $R_{=d}$. Thus all atoms $R_{=d}(x)$ in Ψ can be dealt with in the following way: if x does not occur anywhere else, then clearly $R_{=d}(x)$ is trivially satisfiable in Γ_θ; otherwise $R_{=d}(x)$ can be deleted provided that x is replaced with d everywhere in the input to the CSP (recall that $\mathsf{CSP}_c(\Gamma_\theta)$ allows inputs containing constants). For that reason, w.l.o.g. and for the sake of simplicity, we assume $\mathcal{T}$ does not contain qualified attribute restrictions. Thus we can assume Γ_θ is of form $(\mathbb{Z}, R_{\neg\psi_1}, \ldots, R_{\neg\psi_n})$.

First we show that we can obtain a polynomial time reduction to $\mathsf{CSP}(\Gamma_\theta)$ when *succ* is PP-definable in Γ_θ. Then we do a case analysis directly by first eliminating a degenerate case and then using the complexity classification of the associated CSPs (Theorem 6.4.1).

Claim: If *succ* is PP-definable in Γ_θ, then $\mathsf{CSP}_c(\Gamma_\theta) \leq_p \mathsf{CSP}(\Gamma_\theta)$.
Proof: Let Φ_c be an instance of $\mathsf{CSP}_c(\Gamma_\theta)$, that is, a PP_c formula over Γ_θ. (Recall that constants are assumed to be encoded in unary.) Assume that the relation *succ* can be defined by a PP formula over Γ_θ. We construct a PP formula Φ from Φ_c in such a way that

$$(\star) \quad \Gamma_\theta \models \Phi_c \iff \Gamma_\theta \models \Phi.$$

To construct Φ we first replace all constants in Φ_c by existentially quantified variables; then, using the relation *succ*, we enforce among variables in Φ the same (uni-directional) distance separating the constants in Φ_c they replace.

Let $\bar{c} = c_1, \ldots, c_k$ denote all constants in Φ_c, in ascending order. We "fill the gap" in the successor-line of the $\bar{c}$ by introducing the missing constants, obtaining a gapless sequence $\bar{d}$. For example: if $\bar{c} = 3, 6, 7$ we complete the sequence by letting $\bar{d}$ be $3, 4, 5, 6, 7$. We introduce all needed variables by letting $\bar{x}_d$ be a sequence of fresh variables corresponding to $\bar{d}$, whose indexes equal the value of the constants they replace. So replace the constants c in Φ_c by the corresponding variables x_d. We take care of variables y already present in Φ_c as follows. When an atom of the form $succ(d) = y$ ($succ(y) = d$) occurs in Φ_c we accomodate y in the corresponding $d+1$-th ($d-1$-th) position in the sequence $\bar{d}$, accordingly, replacing y with x_{d+1} (x_{d-1}) everywhere. Similarly with atoms of the form $(y = d)$ where d is an integer. The resulting Φ is a PP sentence over Γ_θ.

We now show that $(\star)$ holds. We assume w.l.o.g. that the Gaifman graph of Φ_c is connected. The "only if" direction is straightforward: take a satisfying assignment to Φ_c and extend it by using the indices of the variables in Φ (i.e. assign a to variable

x_a). The extended assignment will clearly satisfy Φ. For the converse direction, suppose $\Gamma_\theta \models \Phi$. Take a satisfying assignment α_0 of integers to the variables $\bar{x}_d$ in Φ. By a straighforward result in [16], we know that for any tuples $\bar{a}, \bar{b} \in \mathbb{Z}^k$, there is an automorphism α of $(\mathbb{Z}, succ)$ with $\alpha(a_i) = b_i$ for all $i \in [k]$ if, and only if, $a_i - a_j = b_i - b_j$ for all $1 \leq i, j \leq k$. Thus using α_0 we can use the "one-direction distance preserving" automorphisms of $(\mathbb{Z}, succ)$, which are guaranteed to exist, to find a satisfying assignment to Φ_c. Constructively, we define an injective function δ from the image of α_0 to $\mathbb{Z}$ such that $\delta(x) = x + \kappa$, where κ is a constant integer. Indeed, to determine κ, take the minimum a of the set $\{x \mid x \text{ is a constant occurring in } \Phi_c\} \cup \{x \mid x \text{ is in the image of } \alpha_0\}$; and take the first variable x_j using the sequence $\bar{d}$. Then

$$\kappa = \begin{cases} |j - a| \text{ if } j \leq a \\ -|j - a| \text{ otherwise.} \end{cases}$$

Then the restriction of $\delta \circ \alpha_0$ to the corresponding variables $\bar{x}$ in Φ gives us a satisfying assignment for Φ_c so that $\Gamma_\theta \models \Phi_c$ as desired. The claim now follows. ■
We are now ready to do a case distinction using Theorem 6.4.1.

First we eliminate the one degenerate case which corresponds to item (2) of the theorem.

Claim: Assume $\mathsf{CSP}(\Gamma_\theta)$ is not equivalent to a CSP of a finite template. Then $\mathsf{CSP}(\Gamma_\theta)$ is not equal to $\mathsf{CSP}(\Gamma)$ where Γ has a first order definition over $(\mathbb{Z}, =)$.
Proof: By Theorem 4 in [21] there exists a template Γ' with a first order definition in $(\mathbb{Z}, =)$ such that $\mathsf{CSP}(\Gamma_\theta)$ equals $\mathsf{CSP}(\Gamma')$ if, and only if, all binary relations which are PP-definable over Γ_θ are either the equality relation or an unbounded distance relation. It then suffices to prove that there exists a binary relation with a PP-definition in Γ_θ which is neither the equality relation nor an unbounded distance relation. So set

$$R_\psi = \{\bar{a}, \bar{b} \in \mathbb{Z}^{|2m|} \mid \bigwedge_{i=1}^{m} \overline{succ}(a_i, b_i)\}.$$

Then

$$R^{=}_{\neg\psi} = \{\bar{a}, \bar{b} \in R_{\neg\psi} \mid \bar{a} = \bar{b}\} = \emptyset$$

has a PP-definition in Γ_θ. Now $R^{=}_{\neg\psi}$ is neither the equality relation, nor an unbounded distance relation. The claim now follows.

■

Now for the cases:

1. Γ_θ has a finite core. Since $\mathsf{CSP}(\Gamma_\theta)$ has a dichotomy by the Feder-Vardi conjecture, in this case we use Theorem 6.2.1 to get a dichotomy for $\mathsf{CSP}_c(\Gamma_\theta)$.

2. Γ_θ contains *succ*, or *succ* can be defined by a PP formula using only the relations $R_{\neg\psi_1}, \ldots, R_{\neg\psi_n}$. We showed above that $\mathsf{CSP}_c(\Gamma_\theta) \leq_p \mathsf{CSP}(\Gamma_\theta)$. By Theorem 6.4.1, (4), $\mathsf{CSP}(\Gamma_\theta)$ is in PTIME or NP-hard.

3. Γ_θ does not contain *succ* or *succ* is not PP-definable over Γ_θ, and also $\mathsf{CSP}(\Gamma_\theta)$ is not equivalent to any CSP with a finite template. Then by Theorem 6.4.1, (2) and (3), $\mathsf{CSP}(\Gamma_\theta)$ equals $\mathsf{CSP}(\Gamma)$ where Γ either has a first order definition over $(\mathbb{Z}, =)$ or the relation $Dist_k$ is PP-definable in Γ for all $k \in \mathbb{Z}$. The former cannot be the case, as shown above. Therefore the relation $Dist_k$ is PP-definable in Γ for all $k \in \mathbb{Z}$ and so by Theorem 6.4.1, (3), $\mathsf{CSP}(\Gamma_\theta)$ is NP-hard. Now note that $\mathsf{CSP}(\Gamma_\theta) \subseteq \mathsf{CSP}_c(\Gamma_\theta)$, that is, any instance of $\mathsf{CSP}(\Gamma_\theta)$ is an instance of $\mathsf{CSP}_c(\Gamma_\theta)$. Therefore $\mathsf{CSP}_c(\Gamma_\theta)$ is NP-hard.

When $\mathsf{CSP}_c(\Gamma_\theta)$ is in PTIME we are done. When $\mathsf{CSP}_c(\Gamma_\theta)$ is NP-hard we use Theorem 6.4.2 to obtain that there is a rooted OMQ Q' over $\mathcal{D}$ with $\mathsf{dtype}(Q') = \theta$ such that the complement of $\mathsf{CSP}_c(\Gamma_\theta)$ is polynomially reducible to evaluating Q'. Altogether evaluating Q' is co-NP-hard. Therefore evaluating Q is either in PTIME or possibly co-NP-hard. The theorem now follows. □

Complexity bounds We prove two complexity results for $(\mathbb{Z}, \overline{succ})$-KBs based on the datatype pattern used. In more detail, we show that the problem of answering OMQs Q with $\mathsf{dtype}(Q) = \{\overline{succ}(x, y)\}$ has polynomial time complexity. Then we prove that already given a datatype pattern $\theta = \{\overline{succ}(x, y) \wedge \overline{succ}(z, w)\}$ we can construct a co-NP-hard OMQ Q with $\mathsf{dtype}(Q) = \theta$.

Theorem 6.4.5. *Let $(\mathcal{T}, \mathcal{A})$ be a $(\mathbb{Z}, \overline{succ})$-KB and $Q = (\mathcal{T}, q)$ a rooted OMQ where $\mathsf{dtype}(Q)$ contains a single atom $\overline{succ}(x, y)$. Then evaluating Q is in* PTIME.

Proof. Let $Q = (\mathcal{T}, q)$ with $\mathsf{dtype}(Q) = \{\overline{succ}(x, y)\}$. For simplicity and w.l.o.g. assume $\mathcal{T}$ does not contain qualified attribute restrictions; on that, see the first paragraph of the proof of Theorem 6.4.4. Thus $\Gamma_\theta = (\mathbb{Z}, succ)$. Now using the first claim proved in Theorem 6.4.4, since Γ_θ contains *succ*, we have $\mathsf{CSP}_c(\Gamma_\theta) \leq_p \mathsf{CSP}(\Gamma_\theta)$. We show that $\mathsf{CSP}(\mathbb{Z}, succ)$ is solvable in polynomial time. Let

$$\varphi = \exists \bar{x}(succ(x_1, x_2) \wedge \ldots \wedge succ(x_{n-1}, x_n))$$

be an input to $\mathsf{CSP}(\mathbb{Z}, succ)$. We assume that φ is acyclic (as defined in the previous section for $(\mathbb{Q}, \leq)$) and has no conjunct of the form $succ(x, x)$, since otherwise φ is trivially unsatisfiable in $(\mathbb{Z}, succ)$. For simplicity, assume the Gaifman graph of φ is connected. We attempt to define, in a stepwise manner, a satisfying (w.r.t. $(\mathbb{Z}, succ)$) assignment α of integers to the variables of φ, written $Var(\varphi)$, using the following algorithm:

```
select x ∈ Var(φ);
α(x) = 0;
while ∃z, y ∈ Var(φ) with z ∉ dom(α) and y ∈ dom(α) do
    select z, y;
    if succ(y, z) ∈ φ then
        α(z) = α(y) + 1;
    end
    if succ(z, y) ∈ φ then
        α(z) = α(y) − 1;
    end
end
if α is a function with (ℤ, succ) ⊨_α φ then
    return Yes;
end
return No;
```

It is easy to check that $\varphi \in \mathsf{CSP}(\mathbb{Z}, succ)$ if, and only if, the algorithm outputs "Yes". It is readily checked that it has polynomial time complexity w.r.t. the number of conjuncts in φ.

□

We now prove that if $\mathsf{dtype}(q) = \{\overline{succ}(x, y) \wedge \overline{succ}(z, w)\}$, then we can construct a co-NP-hard OMQ $Q = (\mathcal{T}, q)$.

Theorem 6.4.6. *Let $\theta = \{\overline{succ}(x, y) \wedge \overline{succ}(z, w)\}$. Then there is a rooted OMQ $Q = (\mathcal{T}, q)$ with $\mathsf{dtype}(q) = \theta$ such that evaluating Q is* co-NP*-hard.*

Proof. We will define $R_{dist\ k} = \{(a, b) \in \mathbb{Z}^2 \mid |a - b| = k\}$ for all $k \in \mathbb{Z}$ using solely the relation $R = \{(x, y, z, w) \in \mathbb{Z}^4 \mid succ(x) = y \vee succ(z) = w\}$. It will then immediately follow from Theorem 6.4.2 and Theorem 6.4.1, (3), that there is a rooted OMQ $Q = (\mathcal{T}, q)$ with $\mathsf{dtype}(q) = \{\overline{succ}(x, y) \wedge \overline{succ}(z, w)\}$ such that evaluating Q is co-NP-hard. For all $k > 0$, $R_{dist_k} = \{(x_0, x_k) \in \mathbb{Z}^2 \mid \varphi(x_0, x_k)\}$, where $\varphi(x_0, x_k)$ can be written as

$$(\star)\ \exists x_1 \dots x_{k-1}(succ(x_0, x_1) \wedge \dots \wedge succ(x_{k-1}, x_k)) \vee (succ(x_k, x_{k-1}) \wedge \dots \wedge succ(x_1, x_0)).$$

Indeed, by the distributive law, $(\star)$ is equivalent to

$$\exists x_1 \dots x_{k-1} \bigwedge_{0 \leq i < k, 0 \leq j < k} (succ(x_i, x_{i+1}) \vee succ(x_{j+1}, x_j))$$

and, thus, R_{dist_k} can be expressed as a PP formula over R. □

6.4.2 Distance datatypes

We now study the complexity of OMQ answering over all datatypes of the form $\mathcal{D} := (\mathbb{Z}, R_1, \dots, R_n)$ where each R_i is first order definable over $(\mathbb{Z}, succ)$ and $\overline{R_i}$ (the complement of R_i) is locally finite. We call such structures *distance datatypes*. We disallow qualified

attribute restrictions in the language, that is, here we set $\mathcal{L} := \mathcal{ALCHI}^{attrib}$. Also, for the results of this section we restrict ourselves to *atomic datatype patterns*, that is, ones of the form $\{S_1(\bar{z}_1), \dots, S_m(\bar{z}_m)\}$ where S_i is a $|\bar{z}_i|$-ary relation from $\mathcal{D}$.

Example 6.4.7. Let $\mathcal{D} = (\mathbb{Z}, \overline{succ}, \overline{R_{\text{dist1}}})$ where $\overline{R_{\text{dist1}}}$ is the complement of $R_{\text{dist1}} = \{(a,b) \in \mathbb{Z}^2 \mid succ(a,b) \vee succ(b,a)\}$. It can be checked that both *succ* and R_{dist1} are locally finite. On the other hand, $\overline{succ}$ and $\overline{R_{\text{dist1}}}$ themselves are not locally finite. To see this, fix an integer k. Then for all $k' \in \mathbb{Z}$ satisfying $k' \neq k+1$, the tuple (k, k') is in $\overline{succ}$. Thus the relation $\overline{succ}$ is unbounded. The argument can be adapted for $\overline{R_{\text{dist1}}}$.

Example 6.4.8. An infinite class of useful distance datatypes is defined by using as a basis the relations R_{dist_k} for arbitrary $k > 0$. First, to define R_{dist_k} we use the construction in Theorem 6.4.6. We then define $R_{\text{dist}_{\leq k}}$ as the relation containing all tuples (a,b) such that the distance between a and b is (zero or) at most k. That is,

$$R_{\text{dist}_{\leq k}} := \{(a,b) \in \mathbb{Z}^2 \mid a = b \vee (\bigvee_{i=1}^{k} R_{\text{dist}_i}(a,b))\}.$$

It is straightforward to check that such relations are all locally finite. We write $R_{\text{dist}_{>k}}$ for $\overline{R_{\text{dist}_{\leq k}}}$. Then for all $m \geq 0$, the datatype $\mathcal{D} := (\mathbb{Z}, R_1, \dots, R_m)$ where each R_i, $1 \leq i \leq m$, is $R_{\text{dist}_{>k}}$, for some $k \in \mathbb{Z}$, is a distance datatype.

The reason for disallowing conjunctions of datatype atoms is that simple "conjunctive" datatype patterns over $(\mathbb{Z}, \overline{succ})$ (studied in the previous section) already give rise to templates which are not locally finite.

Example 6.4.9. Let $\theta := \{\overline{succ}(x,y) \wedge \overline{succ}(x,z)\}$. We obtain the template $\Gamma_\theta = (\mathbb{Z}, R_{\neg\psi})$ with $R_{\neg\psi} = \{(a,b,c) \in \mathbb{Z}^3 \mid succ(a,b) \vee succ(a,c)\}$. It is easy to see that tuples $(1,2,k)$ for all $k \in \mathbb{Z}$ are in $R_{\neg\psi}$. Thus $R_{\neg\psi}$ is unbounded, which yields Γ_θ is not locally finite.

Basic dichotomy for distance datatypes We use the result below from the CSP literature, which refines the general result from [21] for the special case of locally finite templates.

Theorem 6.4.10. *([16]) Let $\Gamma = (\mathbb{Z}, R_1, R_2, \dots)$ be a locally finite template without a finite core which is first-order definable over $(\mathbb{Z}, succ)$. Then,*

1. *If succ is not PP-definable in Γ, then $\mathsf{CSP}(\Gamma)$ is* NP*-hard.*
2. *Otherwise $\mathsf{CSP}(\Gamma)$ is either in* PTIME *or* NP*-hard.*

Remark 6.4.11. Trivially, all templates $\Gamma_\theta = (\mathbb{Z}, R_{\neg\psi_1}, \dots, R_{\neg\psi_n})$ obtained for the instantiation (Theorem 6.4.2) are locally finite, given the assumption that $\mathcal{D}$ is a distance datatype.

Given an atomic datatype pattern θ over a distance datatype we obtain a dichotomy for answering OMQs Q with $\mathsf{dtype}(Q) = \theta$.

Theorem 6.4.12. *Let θ be an atomic datatype pattern over a distance datatype $\mathcal{D}$. Then the problem of evaluating an OMQ Q with the BMDP over $\mathcal{D}$ with $\mathsf{dtype}(Q) = \theta$ is either in* PTIME*, or there is a rooted OMQ Q' with $\mathsf{dtype}(Q') = \theta$ such that evaluating Q' is co-*NP*-hard.*

Proof. Let $Q = (\mathcal{T}, q)$ be an OMQ with the BMDP over a distance datatype $\mathcal{D} = (\mathbb{Z}, R_1, \dots, R_n)$ with $\mathsf{dtype}(Q) = \theta$. By Theorem 6.4.2, we get a polynomial time reduction to the complement of $\mathsf{CSP}_c(\Gamma_\theta)$ where $\Gamma_\theta = (\mathbb{Z}, S_1, \dots, S_m)$ with each S_j being the complement of some R_i. If Γ_θ has a finite core, we are done; so we assume Γ_θ is an infinite template.

First assume Γ_θ contains *succ*, or *succ* can be defined by a PP formula using only the relations $S_1, \dots, S_m$. Using a slight modification of the constant-elimination method used in Theorem 6.4.4 we obtain that $\mathsf{CSP}_c(\Gamma_\theta) \leq_p \mathsf{CSP}(\Gamma_\theta)$. By Theorem 6.4.10, (1), $\mathsf{CSP}(\Gamma_\theta)$ is in PTIME or NP-hard. Therefore evaluating Q is either in PTIME or possibly co-NP-hard. Otherwise Γ_θ does not contain *succ*; again by Theorem 6.4.10, (2), $\mathsf{CSP}(\Gamma_\theta)$ is NP-hard. Now note that $\mathsf{CSP}(\Gamma_\theta) \subseteq \mathsf{CSP}_c(\Gamma_\theta)$, that is, any instance of $\mathsf{CSP}(\Gamma_\theta)$ is an instance of $\mathsf{CSP}_c(\Gamma_\theta)$. Therefore $\mathsf{CSP}_c(\Gamma_\theta)$ is NP-hard. Then by Theorem 6.4.2 there is a rooted OMQ Q' over $\mathcal{D}$ with $\mathsf{dtype}(Q') = \theta$ such that the complement of $\mathsf{CSP}_c(\Gamma_\theta)$ is polynomially reducible to evaluating Q'. Altogether evaluating Q' is co-NP-hard. □

Tractability characterisation For locally finite templates Γ which do not have a finite core the complexity of $\mathsf{CSP}(\Gamma)$ is characterised by certain "min" and "max" polymorphisms defined next. For templates which are not necessarily locally finite, still such polymorphisms constitute a suficient condition for tractability.

Definition 6.4.13. Let $d \in \mathbb{N}^*$.

The *d-modular max operation* is the operation $max_d \colon \mathbb{Z}^2 \to \mathbb{Z}$ defined by

$$max_d(x, y) = \begin{cases} max(x, y), \text{ if } x = (y \mod d) \\ x, \text{ otherwise.} \end{cases}$$

The *d-modular min operation* is the operation $min_d \colon \mathbb{Z}^2 \to \mathbb{Z}$ defined by

$$min_d(x, y) = \begin{cases} min(x, y), \text{ if } x = (y \mod d) \\ x, \text{ otherwise.} \end{cases}$$

Let Γ be a template. Then Γ has a modular (or the d-modular) polymorphism if Γ is preserved by the d-modular *max* or the d-modular *min* operation for some $d \in \mathbb{N}^*$.

Theorem 6.4.14. *([16, 21]) Let Γ be a template with finite signature, without a finite core, and a first order definition in $(\mathbb{Z}, succ)$. If Γ is locally finite, then Γ has a d-modular polymorphism and $\mathsf{CSP}(\Gamma)$ is in* PTIME*, or $\mathsf{CSP}(\Gamma)$ is* NP*-hard. Otherwise Γ has a d-modular polymorphism only if $\mathsf{CSP}(\Gamma)$ is in* PTIME*.*

Remark 6.4.15. An easy "use case" of such polymorphisms is showing that $\mathsf{CSP}(\mathbb{Z}, \mathit{succ})$ is solvable in PTIME. (We gave a direct proof of this in Theorem 6.4.5.) One only needs to check that *succ* is preserved by modular $\mathit{min}_1 = \mathit{min}$, by Theorem 6.4.14. Take any two tuples $t_1 = (a, b), t_2 = (c, d) \in \mathit{succ}$. Given that $b = a + 1$ and $d = c + 1$, we have that $(a, a + 1),\ (c, c + 1) \in \mathit{succ}$. Clearly, $\mathit{succ}(\min(a, c)) = \mathit{min}(a + 1, c + 1)$. The claim now follows from Theorem 6.4.14, *in fine*. The relationship between non-local finiteness of templates and intractability of the associated CSP is, to our knowledge, not well-understood. Nevertheless, it can be verified that one of the simplest non-locally finite templates, $(\mathbb{Z}, \overline{\mathit{succ}})$, does not have a d-modular polymorphism, so that poly-time complexity of $\mathsf{CSP}(\mathbb{Z}, \overline{\mathit{succ}})$ is not guaranteed. The property needed for showing this is that for all d there exist tuples $t_1,\ t_2 \in \overline{\mathit{succ}}$ such that $\mathit{min}_d(t_1, t_2) \notin \overline{\mathit{succ}}$ and there exist tuples $t'_1,\ t'_2 \in \overline{\mathit{succ}}$ such that $\mathit{max}_d(t_1, t_2) \notin \overline{\mathit{succ}}$. It can indeed be verified that for all d, the tuples $(-d - 1, 0),\ (0, -d)$, with the condition that both are in $\overline{\mathit{succ}}$, are counterexamples to preservation of $(\mathbb{Z}, \overline{\mathit{succ}})$ under min_d; also, for all d, the tuples $(d - 1, 0),\ (0, d)$ do the job of ruling out max_d. Moreover, if $(\mathbb{Z}, \overline{\mathit{succ}})$ is shown not to satisfy the property under Theorem 2, n. 3 in [21], then $\mathsf{CSP}(\mathbb{Z}, \overline{\mathit{succ}})$ will be shown to be NP-hard.

From Theorem 6.4.14 we also obtain a dichotomy in terms of polymorphisms.

Corollary 6.4.16. *Let $\theta = (\theta_{\mathcal{T}}, \theta_q)$ be a datatype pattern over a distance datatype, where $\theta_q = \{\psi_1, \ldots, \psi_n\}$. Then (1) if each $R_{\neg\psi_i}$ has a d-modular polymorphism, then evaluating OMQs Q with $\mathsf{dtype}(Q) = \theta$ and the BMDP is in* PTIME*; (2) otherwise, there is a rooted OMQ Q with $\mathsf{dtype}(Q) = \theta$ such that evaluating Q is* co-NP*-complete.*

7 Related work

7.1 Introduction

We review the literature on datatypes and query answering over lightweight DLs with datatypes. Our aim is identifying both research directions and results which are related to ours, focusing mainly on tractability results.

In Section 7.2 we revisit the early introduction of expressive languages with datatypes, formulating decidability and the first complexity results in the literature. In Section 7.3 we introduce and discuss DLs with multiple datatypes. We reformulate old results and languages when possible by using our own notation and approaches.

The complexity of query answering over languages with datatypes depends, obviously, on the various parameters of the problem. We classified complexity of query answering in a non-uniform way: from a problem that is co-NP-hard in general we obtained complexity dichotomies first by fixing a datatype or a family of datatypes. For instance, if the datatype is finite, then we obtain a basic dichotomy. Second, in some cases, by looking at the structure of the datatype patterns, we presented a more fine-grained analysis of the tractable cases. The tractable cases we identified in this way are not amenable to typical query rewriting techniques. In contrast, in Section 7.4 we review results on query answering over lightweight DLs with datatypes which are uniform; that is, they hold in general for the very lightweight languages considered and can be extended with unary datatypes without loss of the rewritability property (assuming that such datatypes satisfy some sort of convexity property). Similarly, we will also look into results that hold for all queries over n-ary datatypes and allow for rewritability, but at the cost of severely restricting the classes of datatypes allowed.

We will use Table 7.1 below as a reference for the various DLs discussed throughout this chapter.

7.2 The origins: concrete domains

The capability of referring to concrete objects such as numbers and strings is a desirable feature of formalisms given existing applications, from encoding ER diagrams to engineering [83]. Very early formalisms for Knowledge Representation already dealt with attributes in an *ad hoc* way e.g. [46] and the CLASSIC system [26]. In the latter system, for example, besides concepts representing "real world individuals of a domain", the representation language also allowed the description of "individuals in the implementation language" (at the time, Common Lisp), such as strings and integers. Nevertheless, DLs with datatypes would only be introduced in 1991 in [9]. They were then investigated

Name	Syntax	Semantics
atomic concept	A	$A^{\mathcal{I}}$
top concept	$\top$	$\Delta_{\mathsf{ind}}^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
concept negation	$\neg C$	$\Delta_{\mathsf{ind}}^{\mathcal{I}} \setminus C^{\mathcal{I}}$
role negation	$\neg r$	$\Delta_{\mathsf{ind}}^{\mathcal{I}} \times \Delta_{\mathsf{ind}}^{\mathcal{I}} \setminus r^{\mathcal{I}}$
attribute negation	$\neg U$	$\Delta_{\mathsf{ind}}^{\mathcal{I}} \times \mathrm{dom}(\mathcal{D}) \setminus U^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
attribute range	$\mathrm{ran}(U)$	$\{v \in \mathrm{dom}(\mathcal{D}) \mid (a, v) \in U^{\mathcal{I}})\}$
existential restriction	$\exists r.C$	$\{a \in \Delta_{\mathsf{ind}}^{\mathcal{I}} \mid \exists b \text{ with } (a, b) \in r^{\mathcal{I}} \ \& \ b \in C^{\mathcal{I}}\}$
universal restriction	$\forall r.C$	$\{a \in \Delta_{\mathsf{ind}}^{\mathcal{I}} \mid \forall b \ (a, b) \in r^{\mathcal{I}} \implies b \in C^{\mathcal{I}}\}$
attribute restriction	$\exists U$	$\{a \in \Delta_{\mathsf{ind}}^{\mathcal{I}} \mid \exists v((a, v) \in U^{\mathcal{I}})\}$
attribute range restriction	$\forall U.D_i$	$(\mathrm{ran}(U))^{\mathcal{I}} \subseteq \mathrm{dom}(\mathcal{D}_i)$
$\exists$-qual. att. restriction	$\exists U.\varphi$	$\{a \in \Delta_{\mathsf{ind}}^{\mathcal{I}} \mid \exists v \ ((a, v) \in U^{\mathcal{I}} \ \& \ \mathcal{D} \models \varphi(v))\}$
$\forall$-qual. att. restriction	$\forall U.\varphi$	$\{a \in \Delta_{\mathsf{ind}}^{\mathcal{I}} \mid \forall v \ ((a, v) \in U^{\mathcal{I}} \implies \mathcal{D} \models \varphi(v))\}$
n-ary attr. restriction	$\exists U_1 \ldots U_n.R$	$\{a \in \Delta_{\mathsf{ind}}^{\mathcal{I}} \mid \exists (v_1, \ldots, v_n) \in R^{\mathcal{D}}$ with $(a, v_i) \in (U_i)^{\mathcal{I}}\}$
n-ary attr. range restriction	$\forall U_1 \ldots U_n.R$	$\{a \in \Delta_{\mathsf{ind}}^{\mathcal{I}} \mid \forall v_1, \ldots, v_n \in \mathrm{dom}(\mathcal{D})$ with $(a, v_i) \in (U_i)^{\mathcal{I}}$ we have $(v_1, \ldots, v_n) \in R^{\mathcal{D}}\}$
number restriction	$\leq q.r$	$\{a \in \Delta_{\mathsf{ind}}^{\mathcal{I}} \mid \lvert\{(a, b) \in r^{\mathcal{I}}\}\rvert \leq q\}$
number restriction	$\geq q.r$	$\{a \in \Delta_{\mathsf{ind}}^{\mathcal{I}} \mid \lvert\{(a, b) \in r^{\mathcal{I}}\}\rvert \geq q\}$
concept inclusion	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
role inclusion	$r_1 \sqsubseteq r_2$	$r_1^{\mathcal{I}} \subseteq r_2^{\mathcal{I}}$
attribute inclusion	$U_1 \sqsubseteq U_2$	$U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}}$
role functionality	(funct r)	$\forall a, b_1, b_2 \in \Delta_{\mathsf{ind}}^{\mathcal{I}}$, $(a, b_1) \in r^{\mathcal{I}} \wedge (a, b_2) \in r^{\mathcal{I}} \implies b_1 = b_2$
attribute functionality	(funct U)	$\forall a, v_1, v_2 \in \Delta^{\mathcal{I}}$, $(a, v_1) \in U^{\mathcal{I}} \wedge (a, v_2) \in U^{\mathcal{I}} \implies v_1 = v_2$
disjointness constraint	$\mathsf{disj}(X_1, X_2)$	$X_1^{\mathcal{I}} \cap X_2^{\mathcal{I}} = \emptyset$

Table 7.1: Syntax and semantics of DLs

intensively [74]. Datatypes were called *concrete domains* (with concrete predicates). The ones explored in early work were typically quite expressive, as was the basic language extended with them, $\mathcal{ALC}$, a superset of which was introduced in Chapter 2.

The original extension $\mathcal{ALC}(\mathcal{D})$ with $\mathcal{D}$ a datatype [9] augments $\mathcal{ALC}$ with concepts of the form $\exists u_1 \dots u_n.R$ where each u_i is a composition of *functional attributes*. In this way, the semantics is given in terms of a composition of partial functions $f_1^i \dots f_{n_i}^i g^i$, where the f^i are defined from the domain of individuals to the domain of individuals and g^i from the domain of individuals to $\mathrm{dom}(\mathcal{D})$; each u_i is called a *path*. See [74], Definitions 2 and 3 for the formal definitions and discussion. Such concepts allow one to use relations R from $\mathcal{D}$ for representing constraints on data values. The authors of [9] introduced a sufficient condition for decidability of checking satisfiability of concepts (equivalently subsumption between concepts [8]) under $\mathcal{ALC}(\mathcal{D})$ with empty TBox. Call a datatype $\mathcal{D} = (\mathrm{dom}(\mathcal{D}), R_1, R_2, \dots)$ *admissible* if (1) $\mathcal{D}$ is closed under negation, that is, if $R \in \{R_1, R_2, \dots\}$, then there exists $P \in \{R_1, R_2, \dots\}$ such that $P = \bar{R}$; (2) there exists a unary $R \in \{R_1, R_2, \dots\}$ denoting $\mathrm{dom}(\mathcal{D})$; and (3) $\mathsf{CSP}(\mathcal{D})$ is decidable. Clearly (1) is desirable given that any concept in $\mathcal{ALC}$ can be negated. Examples of admissible datatypes can be easily found. Consider e.g. the real numbers with all predicates that can be defined by using first order formulas over the reals. Such datatype is admissible from Tarski's famous result, namely the decidability of real arithmetic [97]. Another example is the datatype having the rational numbers as the domain, the unary predicates $\top_{\mathbb{Q}}$, $\bot_{\mathbb{Q}}$ (extension: all rational numbers and the empty set, resp.), the binary predicates $\leq, <, =, >, \geq$ and one unary predicate per comparison per rational number, as well as ternary predicates for addition and its complement. On the other hand, we provided an example of a simple datatype with integers that is not admissible in Example 3.2.13; given that the CSP is undecidable, extending the datatype with predicates in order to satisfy items (1) and (2) of the definition is not enough. Modulo admissible datatypes, the authors designed a tableau algorithm that computes concept satisfiability and subsumption (again, for the case where the TBox is empty). The decidability result can be formulated as follows:

Theorem 7.2.1. *([9]) Concept satisfiability and subsumptions under $\mathcal{ALC}(\mathcal{D})$ with empty TBox, where $\mathcal{D}$ is admissible, is decidable.*

Notice that the tableaux generated by the algorithm introduced in [9] can be exponentially large in the size of the input assertion,[1] in the worst case. Motivated by this, further work [72, 75] analysed the complexity of the problem by refining the algorithm so that the tableaux generated are tree-shaped, with polynomial-sized paths.

Theorem 7.2.2. *([75]) Concept satisfiability and subsumption under $\mathcal{ALC}(\mathcal{D})$ with empty TBox is* PSPACE*-complete, provided that $\mathcal{D}$ is admissible and $\mathsf{CSP}(\mathcal{D})$ is in* PSPACE*.*

Notice that PSPACE-hardness comes from $\mathcal{ALC}(\mathcal{D})$ alone. Thus even for admissible datatypes for which the CSP is in PTIME, concept satisfiability and subsumption without TBoxes is PSPACE-complete. An example is the admissible datatype on the rational

[1] For checking concept satisfiability of C, the input assertion for the tableau algorithm is simply $\{C(a)\}$.

numbers defined above. Polynomiality of the CSP for this datatype can be shown by a reduction from linear programming (a slight modification of the reduction from mixed integer programming in [75]).

Recall that $\mathcal{ALC}$ with general TBoxes is already ExpTime-complete [8]. As for $\mathcal{ALC}(\mathcal{D})$, adding general TBoxes makes concept satisfiability and subsumption undecidable (see Theorem 8 in [75]). One way of regaining decidability is to allow only for less expressive attribute constructors. This is done by disallowing paths, i.e., allowing only n-ary attribute assertions of the form $\exists U_1 \dots U_n.R$ where each U is an attribute (see Table 7.1). The language can then be augmented with number restrictions and role inclusions without loss of decidability. Conventionally this language has been denoted $\mathcal{SHN}(\mathcal{D})$, for which a tableau was designed to compute concept satisfiability and subsumption. [57]

Theorem 7.2.3. *([57]) Concept satisfiability and subsumption under path-free $\mathcal{SHN}(\mathcal{D})$ with general TBoxes, where $\mathcal{D}$ is admissible, is decidable.*

Another way of regaining decidability is by restricting the datatype itself. For full $\mathcal{ALC}(\mathcal{D})$ where $\mathcal{D} = (\mathbb{Q}, <, \leq, =, \neq, \geq, >)$, for instance, concept satisfiability and subsumption with general TBoxes was shown to be ExpTime-complete. [73].

7.3 Multiple datatypes

In our work all results are formulated with a single datatype, for simplicity sake. Many among the efforts for adding datatypes to DLs, nonetheless, opt for formulating results with an arbitrary number of datatypes [3, 4, 5, 6, 94]. We now provide a cursory formulation of that feature for the language studied in this work followed by an exposition of the literature.

Here we use $\mathbf{D}$ to denote a set of datatypes $\{\mathcal{D}_1, \dots, \mathcal{D}_n\}$. Assume that the domains of the datatypes are mutually disjoint. Let $Q = (\mathcal{T}, q)$ be an OMQ where $\mathcal{T}$ is a Horn-$\mathcal{ALCHI}^{attrib}(\mathbf{D})$-TBox and q is a CQ with the BMDP whose datatype pattern contains relations from any datatype $\mathcal{D}_i$, and each attribute U is defined over a single datatype $\mathcal{D}_i$. Now let θ_i be the part of $\mathsf{dtype}(Q)$ that only refers to $\mathcal{D}_i$, and q_i the component of q containing θ_i. Using a slight reformulation of Theorem 5.4.3, we reduce answering q_i to solving the complement of the CSP_c for the constraint language Γ_{θ_i}. Notice that, given the mutual disjointness of domains of datatypes in $\mathbf{D}$, there is no interaction between the different datatype patterns in components of q. Then it can be checked that the data complexity of answering Q is bounded by the complexity of the complement of $\mathsf{CSP}_c(\Gamma_{\theta_i})$ for some i; namely the hardest CSP obtained. The argument can be easily adapted for UCQs.

Multiple datatypes can be combined (dropping the assumption of mutual disjointness of domains) into what has been called a *datatype hierarchy* in the literature [4, 94], by including both the "top datatype" with all values in the collection, and the "empty datatype".

Example 7.3.1. Using the OWL 2 Datatype Map[2] we can construct the multiple (unary) datatype $\mathbf{D} = \{\bot_{\mathbf{D}}, \top_{\mathbf{D}}, \mathbf{xsd} : \mathbf{integer},\ \mathbf{xsd} : \mathbf{string},\ \mathbf{rdfs} : \mathbf{Literal}\}$. Notice that per the specification, $\mathbf{rdfs} : \mathbf{Literal}$ contains all values of all datatypes, so it has the same domain as $\top_{\mathbf{D}}$.

In the literature, the primary concern has been identifying closure properties of combined multiple datatypes, having decidability or, more recently, tractability in mind. I.e., a certain property of all datatypes in the collection which is preserved when assembling them together in some way. For instance, let $\mathbf{D}$ be a set of datatypes. Call the *disjoint union* of all datatypes in $\mathbf{D}$ the single datatype consisting of (1) the disjoint union of the $\mathrm{dom}(\mathcal{D}_i)$ together with a set W of countably infinitely many untyped values; (2) all relations in each of the datatypes, plus the unary predicates $\top_{\mathcal{D}}$ and $\bot_{\mathcal{D}}$ with the obvious extensions, a unary predicate $=_w$ for each $w \in W$, as well as the binary predicate $=$. Take for instance the property of admissibility presented in the previous section. The decidability results will then hold for such combinations; the same for *products* of all datatypes in $\mathbf{D}$, as defined in a similar way in [6]. A stronger closure property of combinations of multiple datatypes, which requires (among other conditions) polynomiality of each $\mathsf{CSP}(\mathcal{D}_i)$, can also be shown to hold; see Lemma 2.5 and Lemma 2.6 in [6].

Theorem 7.3.2. *(from results in [6]) Let $\mathbf{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a multiple datatype where each $\mathcal{D}_i$ is admissible. Then the disjoint union, and the product, of all such datatypes is also admissible.*

For multiple datatypes all of which are unary (most of the studies focus on this case), tractability properties were investigated in a series of papers [5, 94, 3, 4]. They will be used in the next section. We will refer to Table 7.2 (where we use the notation D for $\mathrm{dom}(\mathcal{D})$ to avoid clutter). It can be checked that (inf-compl) $\Longrightarrow$ (inf) $\Longrightarrow$ (inf-s), and (inf-compl) $\Longrightarrow$ (inf-compl-s) $\Longrightarrow$ (inf-s).

7.4 Query answering over lightweight Description Logics

Ontology languages for which typical reasoning tasks, including query answering, are tractable in general have attracted special interest over the latest years. Such DLs have been called *lightweight languages* for that reason. Here we focus on two families of DLs that go under that guise, to wit, $\mathcal{EL}$ and *DL-Lite*. In the following we make the unique names assumption.

The *DL-Lite* family The *DL-Lite* family is especially significant as it forms the basis of OWL 2 QL,[3] one of the OWL 2 profiles,[4] that is, fragments of the OWL 2 language[5]

[2] `https://www.w3.org/TR/owl2-syntax/#Datatype_Maps`
[3] `https://www.w3.org/TR/owl2-profiles/#OWL_2_QL_2`
[4] `https://www.w3.org/TR/owl2-profiles/`
[5] `https://www.w3.org/TR/owl2-syntax/`

Name	Condition
(inf)	$\lvert\bigcap_{k=1}^{m} D_{i_k}\rvert = 0$ or $\lvert\bigcap_{k=1}^{m} D_{i_k}\rvert \geq \omega$ for all finite sets of domains $D_{i_1}, \ldots, D_{i_k}$ in $\mathbf{D}$
(inf-s)	$\lvert\bigcap_{k=1}^{m} D_{i_k}\rvert \leq 1$ or $\lvert\bigcap_{k=1}^{m} D_{i_k}\rvert \geq \omega$ for all finite sets of domains $D_{i_1}, \ldots, D_{i_k}$ in $\mathbf{D}$
(inf-compl)	$\lvert\bigcap_{k=1}^{m} D_{i_k} \setminus \bigcup_{k=1}^{m} D_{j_k})\rvert = 0$ or $\lvert\bigcap_{k=1}^{m} D_{i_k} \setminus \bigcup_{k=1}^{m} D_{j_k}\rvert \geq \omega$ for all finite sets of domains $D_{i_1}, \ldots, D_{i_m}$ and $D_{j_1}, \ldots, D_{j_n}$ in $\mathbf{D}$
(inf-compl-s)	$\lvert\bigcap_{k=1}^{m} D_{i_k}\rvert \leq 1$ or $\lvert\bigcap_{k=1}^{m} D_{i_k} \setminus \bigcup_{k=1}^{m} D_{j_k}\rvert = 0$ or $\lvert\bigcap_{k=1}^{m} D_{i_k} \setminus \bigcup_{k=1}^{m} D_{j_k}\rvert \geq \omega$ for all finite sets of domains $D_{i_1}, \ldots, D_{i_m}$ and $D^{\mathcal{D}}_{j_1}, \ldots, D_{j_n}$ in $\mathbf{D}$
(convex)	if $\bigcap_{k=1}^{m} D_{i_k} \subseteq \bigcup_{k=1}^{m} D_{j_k}$, then there exists $\ell,\ 1 \leq \ell \leq n$, such that $\bigcap_{k=1}^{m} D_{i_k} \subseteq D_{j_\ell}$ for all finite sets of domains $D_{i_1}, \ldots, D_{i_m}$ and $D_{j_1}, \ldots, D_{j_n}$ of datatypes in $\mathbf{D}$

Table 7.2: Properties of unary multiple datatypes $\mathbf{D}$ [4]

designed with specific application requirements in mind. OWL QL in particular is aimed at applications which use large volumes of instance data whilst allowing users to query such data in an effective way using relational database systems; this is done by rewriting queries into a standard relational query language, as we will see later. *DL-Lite* was introduced in [32] and further studied in [33] and [34] by the same authors; recently a systematic study of combined and data complexity of reasoning in extensions of *DL-Lite* was undertaken in [2].

The *DL-Lite* family comprises basically *DL-Lite*$_{core}$, *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$, where *DL-Lite*$_{core}$ is *DL-Lite*$_{\mathcal{R}}$ (whose extensions with attributes were described in Chapter 2) without role inclusions and *DL-Lite*$_{\mathcal{F}}$ is *DL-Lite*$_{core}$ added with the ability of specifying functionality on roles and their inverses. Checking satisfiability of a *DL-Lite*-TBox w.r.t. an ABox $\mathcal{A}$ is in PTIME in both combined complexity and data complexity [33].

Rewriting Since that is a requirement for understanding some of the results in the literature, we now briefly sketch a well-known technique called first-order (FO) rewriting, or simply rewriting. The technique consists (informally) in showing for a certain language $\mathcal{L}$ that it is possible to combine any CQ q and $\mathcal{L}$-TBox $\mathcal{T}$ into a new FO query $q_{\mathcal{T}}$ (see [13]) whose answers coincide exactly with the answers to q w.r.t. $\mathcal{T}$, for all ABoxes $\mathcal{A}$ (considered as finite interpretations i.e. closed-world databases). Thus $q_{\mathcal{T}}$ can be translated into a SQL statement and evaluated using a RDBMS [1]. The problem of evaluating such (fixed) queries is in AC^0; see [1, 63].[6] FO-rewritability has been extensively discussed in [35, 13]. We now provide a simple example of rewriting of a CQ over a *DL-Lite*$_{\mathcal{R}}$-TBox (recall Chapter 2) into a new UCQ.

Example 7.4.1. Let $(\mathcal{T}, \mathcal{A})$ be a *DL-Lite*$_{\mathcal{R}}$-KB where $\mathcal{A}$ is any ABox and

$$\mathcal{T} = \{\exists r \sqsubseteq A_1,\ A_2 \sqsubseteq A_3,\ r \sqsubseteq s\}$$

[6] The well-known complexity class AC^i contains all decision procedures definable using circuits of polynomial size with AND, OR and NOT gates, depth $O((\log(n))^i)$, and unbounded fan-in. Thus AC^0 includes all problems definable by such circuits having constant depth.

and let $q(x,y) \leftarrow A_1(x), s(x,y), A_3(y)$ be a CQ. Now we need to construct a query $q_{\mathcal{T}}$ with the property that $\mathcal{A} \models q_{\mathcal{T}}$ iff $\mathcal{T}, \mathcal{A} \models q$ for all $\mathcal{A}$. This is done by taking into consideration all axioms that are relevant to q. Let $\mathcal{M}$ be a model of $\mathcal{T}, \mathcal{A}$. For instance, if a pair of individuals a, b is such that $a \in (A_1)^{\mathcal{M}}$, $b \in (A_2)^{\mathcal{M}}$ and $(a,b) \in r^{\mathcal{M}}$, then $a \in (A_1)^{\mathcal{M}}$ (trivially), $a \in (A_3)^{\mathcal{M}}$ and $(a,b) \in s^{\mathcal{M}}$. Therefore the mapping $x \mapsto a$, $y \mapsto b$ is match of q in $\mathcal{M}$. So in order to account for that intensional information in $\mathcal{T}$, we introduce the query

$$q_1(x,y) \leftarrow r(x,y), A_2(y).$$

And so on. It can be seen that the UCQ $q_{\mathcal{T}} := q_1 \vee q_2 \vee \ldots \vee q_5$, where

$$\begin{aligned}
q_1(x,y) &\leftarrow r(x,y), A_2(y),\\
q_2(x,y) &\leftarrow r(x,z), s(x,y), A_3(y),\\
q_3(x,y) &\leftarrow r(x,z), s(x,y), A_2(y),\\
q_4(x,y) &\leftarrow s(x,y), A_1(x), A_2(y),\\
q_5(x,y) &\leftarrow A_3(y), r(x,y).
\end{aligned}$$

has the desired property. In the literature $q_{\mathcal{T}}$ is called a rewriting of q w.r.t. $\mathcal{T}$.

Query answering under *DL-Lite* with datatypes Now notice that any algorithm for rewriting CQs over *DL-Lite*$_{\mathcal{R}}$ (the classical result of [33]), similarly for the language called *DL-Lite*$_{\mathcal{A}}$ in [90], can be adapted for *DL-Lite*$_{\mathcal{R}}^{attrib}(\mathbf{D})$ where $\mathbf{D}$ is a multiple unary datatype satisfying (inf). In the query, in addition to atoms of the form $A(x)$ and $r(x,y)$, where A is a concept name, r is a role and x, y individual variables, we allow for atoms of the form $U(x,v)$ and $D(v)$ where v is a data variable, U is an attribute name, and D is a unary relation from $\mathbf{D}$. It can be checked that attributes under current assumptions behave exactly as roles. Also, it has been shown that this generalises to UCQ answering.

Theorem 7.4.2. *(from [90]) UCQ answering under DL-Lite*$_{\mathcal{R}}^{attrib}(\mathbf{D})$*, where* $\mathbf{D}$ *is a multiple unary datatype and satisfies (inf), is in* AC^0.

Extending these languages with either constructors of the form $\exists r.A$ on the left-hand side, or of the form $\forall r.A$ on the right-hand side of axioms already makes UCQ answering NLOGSPACE-complete in data complexity (and thus typical rewritability is lost; see [35], Theorem 4.1).

Table 7.3 presents the main results for for (U)CQ answering, depending on which conditions are satisfied or violated by $\mathbf{D}$. For some languages of the *DL-Lite* family, it is not difficult to see that if conditions (inf) or (convex) are violated, then the proof techniques in [94] can be used to reduce CQ answering to a co-NP-hard problem such as 2+2-CNF (if the cardinality of the intersection of some collection of predicates is 2) or k-Colourability (if the cardinality of the intersection of some collection of predicates is ≥ 3). On the other hand, under certain reasonable assumptions which depend on the language used, if both conditions are satisfied we can use query rewriting techniques– basically the same ones used for *DL-Lite* without datatypes– to obtain tractable query answering.

Language	Complexity
UCQ ans.: *DL-Lite*$_{\mathcal{R}}^{attrib}$(**D**) where **D** violates (inf)	co-NP-hard [94]
CQ ans.: *DL-Lite*$_{\mathcal{R}}^{attrib}$(**D**) where **D** violates (convex)	co-NP-hard [94]
CQ ans.: *DL-Lite*$_{\mathcal{R}}^{attrib}$(**D**) where **D** violates (inf-compl-s)	co-NP-hard [94]
UCQ ans.: *DL-Lite*$_{\mathcal{R}}^{attrib}$(**D**) where **D** satisfies (inf-compl-s) & (convex)	AC^0 [94, 4]

Table 7.3: Data complexity of CQ answering over multiple unary datatypes **D**

Generalisation to n-ary datatypes To conclude our section on languages of the *DL-Lite* family we now review work on the generalisation to n-ary datatypes. The recent data complexity results we now present were obtained via rewriting; they are uniform, as they hold for all (fixed) input UCQs, as long as the datatype satisfies certain properties. In earlier work on datatypes three essential properties had been considered: convexity, admissibility and p-admissibility. We introduced the first two. The definition of the latter, which we introduce below, uses a generalisation of (convex) for n-ary datatypes:

Definition 7.4.3. A datatype $\mathcal{D}$ is called *convex* if, whenever a conjunction Φ of atoms over $\mathcal{D}$ implies a non-empty disjunction Ψ of atoms over $\mathcal{D}$, then Φ also implies some disjunct ψ of Ψ.

Definition 7.4.4. Let $\mathcal{D} = (\mathrm{dom}(\mathcal{D}), R_1, R_2, \dots)$ be a convex datatype and Φ be the set containing all conjunctions, as well as all implications between conjunctions, of atoms $R_i(\bar{x}_i)$ over $\mathcal{D}$. Then $\mathcal{D}$ is said to be *p-admissible* if the problem of deciding whether a formula $\varphi \in \Phi$ is satisfiable over $\mathcal{D}$ is in PTIME.

In addition to those, in [52] a more restrictive property called *cr-admissibility* has been defined. To introduce it we need to define two additional properties of datatypes $\mathcal{D}$ which are mutually independent w.r.t. p-admissibility. We say that $\mathcal{D}$ is *functional* if for all k-ary relations R in $\mathcal{D}$, $d \in \mathrm{dom}(\mathcal{D})$ and $1 \leq i \leq k$, the formula $R(v_1, \dots, v_k) \wedge v_i = d$ has at most one satisfying assignment in $\mathcal{D}$. When Φ is a first order formula over $\mathcal{D}$, we let S_Φ denote the set of variable assignments that satisfy Φ in $\mathcal{D}$. Then we say that $\mathcal{D}$ is *constructive* if, for all conjunctions Φ, and all disjunctions Ψ, of atoms over $\mathcal{D}$, such that $S_\Phi \setminus S_\Psi \neq \emptyset$, we can compute an element of this set in polynomial time.

Definition 7.4.5. Let $\mathcal{D}$ be a datatype. Then $\mathcal{D}$ is called *cr-admissible* if $\mathcal{D}$ is p-admissible, functional, constructive, and contains all unary relations $=_v$ with $v \in \mathrm{dom}(\mathcal{D})$ where $=_v^{\mathcal{D}}$ is the singleton $\{v\}$, as well as the binary relation $=$, where $=^{\mathcal{D}}$ is the set $\{(v, v) \mid v \in \mathrm{dom}(\mathcal{D})\}$.

Remark 7.4.6. The main datatype we consider in this work, $(\mathbb{Q}, \leq)$, is not convex, so it is not even p-admissible by Definition 7.4.4. To see this, consider the simple implication $(x \leq y) \implies (x' \leq y) \vee (y \leq x')$. While $(x \leq y)$ implies $(x' \leq y) \vee (y \leq x')$, it is however not the case that $(x \leq y) \implies (x' \leq y)$ or $(x \leq y) \implies (y \leq x')$.

The main idea behind cr-admissibility is ensuring combined rewritability of CQs.[7] It should be noted that for unary datatypes neither p-admissibility nor constructivity are required (see Section 6 in [6]); however, convexity is needed for ensuring amenability to rewriting techniques.

The logic used in [52] is an extension of *DL-Lite* with attributes over a cr-admissible datatype $\mathcal{D}$. In addition to other usual constructors, it allows for role functionality, disjointness constraints, n-ary attribute restrictions of the form $\exists U_1, \ldots, U_n.R$ and n-ary attribute range restrictions of the form $\forall U_1, \ldots, U_n.R$, where R is an n-ary relation from $\mathcal{D}$. Notice that, in contrast, in qualified attribute restrictions in the language we defined (Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$), PP formulas with one free variable are used instead of a single atom. See [52] for a thorough description of the language which here we just denote by $\mathcal{L}(\mathcal{D})$ for $\mathcal{D}$ a cr-admissible datatype.

In [52] CQs over datatypes $\mathcal{D}$ are defined exactly as in our work: in addition to atoms of the form $A(x), r(x, y)$ and $U(x, z)$, ones of the form $R(\bar{z})$ are also allowed, where R is from $\mathcal{D}$ and each $z \in \bar{z}$ is a data variable. A safety assumption for CQs q over a cr-admissible datatype $\mathcal{D}$, in order to ensure rewritability in general, is that each data variable z in q occurs either in an atom of the form $U(x, z)$ or in an atom of the form $=_v (z)$, with $v \in \mathrm{dom}(\mathcal{D})$. Such CQs are called *safe*. (Recall Definition 7.4.5 above.) For the case of unary datatypes this assumption is not needed. Let $(\mathcal{T}, \mathcal{A})$ be a $\mathcal{L}(\mathcal{D})$-KB. We say that a CQ q is *combined rewritable w.r.t.* $\mathcal{T}$ if there exists a UCQ q' not depending on $\mathcal{A}$ such that $(\mathcal{T}, \mathcal{A}) \models q$ iff $\mathcal{I}_{\mathcal{T},\mathcal{A}} \models q'$, where $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is a finite interpretation of $\mathcal{T} \cup \mathcal{A}$. In the authors' construction, $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is defined via an "abstract" canonical model of $(\mathcal{T}, \mathcal{A})$. This canonical model is, similarly to our construction, defined in such a way that the data nulls are associated to the relevant TBox constraints; the fact that computing an appropriate assignment to such variables can be done in polynomial time is warranted by $\mathcal{D}$ being p-admissible and constructive. By instantiating this model the authors obtain a database-like interpretation over which UCQs can be evaluated.

An important assumption is that each individual in the domain of the abstract canonical model has at most $n_{\mathcal{T}}$ possible attribute values (each an assignment to a data null), where $n_{\mathcal{T}}$ is the maximum number of occurrences of attribute names in n-ary attribute restrictions $\exists U_1 \ldots U_m.R$, where R is from $\mathcal{D}$, occurring on the right-hand side of axioms in $\mathcal{T}$. This number is denoted $n_{\mathcal{T}}$. A CQ that satisfies this property is called *bounded*.

The rewriting algorithm produces a set Q which preserves the answers to the original query q; in particular, every CQ introduced is also safe and bounded provided that q is safe and bounded. This implies that Q is finite. Q can then be evaluated over the finite database obtained by instantiating the canonical model. The following theorem is then used to show that CQ answering over $\mathcal{L}(\mathcal{D})$ is in PTIME in data complexity.

Theorem 7.4.7. *([52]) For $\mathcal{D}$ cr-admissible, safe and bound CQs are combined rewritable w.r.t. $\mathcal{L}(\mathcal{D})$-TBoxes, and the rewritings are computable.*

[7] The so-called combined rewriting approach [76] generalises FO-rewriting. Roughly this is done by rewriting the ABox $\mathcal{A}$ and the TBox $\mathcal{T}$ into a first-order structure $\mathcal{B}$ (equivalent to a relational database) that is independent of the CQ q; and then rewriting q and $\mathcal{T}$ into a first-order query q' independent of $\mathcal{A}$. Then q' has the property that all answers to q' over $\mathcal{B}$ coincides with the answers to q over $(\mathcal{T}, \mathcal{A})$.

The lightweight language $\mathcal{EL}$ The popular ontology language $\mathcal{EL}$ originated in the quest for the design of languages for which concept subsumption and concept satisfiability are tractable. It first appears in the late 1990s [10] and was systematically studied in [7]. It is at the basis of the widely used medical ontology SNOMED CT [41]. $\mathcal{EL}$-TBoxes contain inclusions of the form $C \sqsubseteq D$ where $C, D ::= A \mid \exists r.C \mid C \sqcap D \mid \top$ with the usual semantics.

Theorem 7.4.8. *([35, 1, 92]) UCQ answering in $\mathcal{EL}$ is* PTIME*-complete in data complexity and* NP*-complete in combined complexity.*

The extension of $\mathcal{EL}$ called $\mathcal{EL}^{++}$, which underpins the OWL2 EL profile,[8] supports datatypes in addition to allowing role composition and role inclusions of the form $r_1 \circ \ldots \circ r_n \sqsubseteq r_{n+1}$, as well as nominals. In order to guarantee tractability of subsumption under $\mathcal{EL}^{++}(\mathcal{D})$, in [7] it is required that $\mathcal{D}$ satisfies p-*admissibility*, defined in the previous subsection.

Unfortunatelly under $\mathcal{EL}^{++}$ CQ answering is undecidable, as proved independently in [66] and [92]; both proofs hinge crucially on the use of role composition and role inclusions. For standard reasoning tasks the more basic $\mathcal{EL}$ behaves well in the face of numerical datatypes, under reasonable restrictions. That has been shown for the (here normalised as in [7]) language $\mathcal{EL}^{\bot}(\mathcal{D})$, where $\mathcal{D} = (\mathrm{dom}(\mathcal{D}), R_1, R_2, \ldots)$ with $\mathrm{dom}(\mathcal{D}) \subseteq \mathbb{R}$ and $R_i^{\mathcal{D}} \subseteq \mathrm{dom}(\mathcal{D}) \times \mathrm{dom}(\mathcal{D})$. A normalised $\mathcal{EL}^{\bot}$-TBox allows axioms of the form

$$A_1' \sqsubseteq A_2',\ A_1 \sqcap A_2 \sqsubseteq A,\ A \sqsubseteq \exists r.A', \exists r.A \sqsubseteq A',\ A \sqsubseteq \exists U.\varphi, \text{ and } \exists U.\varphi \sqsubseteq A \tag{7.1}$$

where A, A', A_1, A_2 are concept names; A_1' is either a concept name or $\top$; A_2' is either a concept name, $\bot$ or $\top$; r is a role name, U is an attribute name, and φ is (more restrictively than in our work) an *atomic* PP_c formula of the form $R(x, c)$ or $R(c, x)$ with R from $\mathcal{D}$ and $c \in \mathrm{dom}(\mathcal{D})$. In OWL 2 EL, $\mathcal{D}$ can contain only the equality relation. In [80] a detailed non-uniform complexity analysis of the task of classification (computing all subsumption relations) in $\mathcal{EL}^{\bot}(\mathcal{D})$ ontologies was carried out. The analysis is based on how datatypes occur in $\mathcal{EL}^{\bot}$-TBoxes– in particular, in the polarity of such occurences. Nonetheless, query answering was not considered and, to our knowledge, no such investigation exists in the literature. However, a hardness result for UCQ answering over $\mathcal{EL}^{\bot}(\mathcal{D})$-TBoxes, where $\mathcal{D}$ is a numerical datatype with predicates $<$ and $=$, can be obtained by a simple reduction from 3-Colourability (using proof techniques similar to those in [94]) already when datatype atoms are disallowed in the query. The reduction relies on the use of negative occurences of attribute restrictions $\exists U.\varphi$ (see 7.1 above) in axioms. Recall that this is disallowed in Horn-$\mathcal{ALCHI}^{qattrib}(\mathcal{D})$-TBoxes.

[8] `https://www.w3.org/TR/owl2-profiles/#OWL_2_EL_2`

Bibliography

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. Artif. Intell. Res. (JAIR)*, 36:1–69, 2009.

[3] A. Artale, Y. A. Ibáñez-García, R. Kontchakov, and V. Ryzhikov. Dl-lite with attributes and sub-roles (extended abstract). In *Proceedings of the 24th International Workshop on Description Logics (DL 2011), Barcelona, Spain, July 13-16, 2011*, 2011.

[4] A. Artale, R. Kontchakov, V. Ryzhikov, and O. Savkovic. Datatypes in DL-Lite (manuscript).

[5] A. Artale, V. Ryzhikov, and R. Kontchakov. *DL-Lite* with attributes and datatypes. In *ECAI*, pages 61–66, 2012.

[6] F. Baader, S. Borgwardt, and M. Lippmann. Query rewriting for *DL-Lite* with n-ary concrete domains (extended version). LTCS-Report 17-04, Chair for Automata Theory, Technische Universität Dresden, Germany, 2017.

[7] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI-05*, pages 364–369, 2005.

[8] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[9] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *IJCAI 1991*, pages 452–457, 1991.

[10] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proceedings of the Sixteenth*

International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages, pages 96–103, 1999.

[11] R. Berger. The undecidability of the domino problem. *Mem. Amer. Math. Soc. No.*, 66:72, 1966.

[12] M. Bienvenu, C. Lutz, and F. Wolter. Query containment in description logics reconsidered. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*, 2012.

[13] M. Bienvenu and M. Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web 2015*, pages 218–307, 2015.

[14] M. Bodirsky. *Constraint Satisfaction Problems with Infinite Templates*, pages 196–228. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[15] M. Bodirsky. The complexity of constraint satisfaction problems (invited talk). In *STACS 2015*, pages 2–9, 2015.

[16] M. Bodirsky, V. Dalmau, B. Martin, A. Mottet, and M. Pinsker. Distance constraint satisfaction problems. *Information and Computation*, 247:87 – 105, 2016.

[17] M. Bodirsky, M. Hermann, and F. Richoux. Complexity of existential positive first-order logic. In *Proceedings of the 5th Conference on Computability in Europe: Mathematical Theory and Computational Practice*, CiE '09, pages 31–36, Berlin, Heidelberg, 2009. Springer-Verlag.

[18] M. Bodirsky, M. Hils, and B. Martin. On the scope of the universal-algebraic approach to constraint satisfaction. In *2010 25th Annual IEEE Symposium on Logic in Computer Science*, pages 90–99, 2010.

[19] M. Bodirsky and J. Kára. The complexity of temporal constraint satisfaction problems. *J. ACM*, 57(2):9:1–9:41, 2010.

[20] M. Bodirsky and J. Kára. A fast algorithm and datalog inexpressibility for temporal reasoning. *ACM Trans. Comput. Log.*, 11(3):15:1–15:21, 2010.

[21] M. Bodirsky, B. Martin, and A. Mottet. Constraint satisfaction problems over the integers with successor. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 256–267, 2015.

[22] M. Bodirsky and J. Nesetril. Constraint satisfaction with countable homogeneous templates. *J. Log. Comput.*, 16(3):359–373, 2006.

[23] M. Bodirsky and M. Pinsker. Reducts of Ramsey structures. *CoRR*, abs/1105.6073, 2011.

[24] V. G. Bodnarchuk, L. A. Kaluzhnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras. *Kibernetika*, 5(3):1–10, May 1969.

[25] E. Botoeva, R. Kontchakov, V. Ryzhikov, F. Wolter, and M. Zakharyaschev. Query inseparability for description logic knowledge bases. In *KR 2014*, 2014.

[26] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, L. A. Resnick, and A. Borgida. Living with classic: When and how to use a kl-one-like language. In *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, 1991.

[27] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, Mar. 2005.

[28] A. A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, Jan. 2006.

[29] A. A. Bulatov. A dichotomy theorem for nonuniform csps. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330, Oct 2017.

[30] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, pages 77–86, 2009.

[31] A. Calì, G. Gottlob, and A. Pieris. Tractable reasoning in description logics with functionality constraints. In *In Search of Elegance in the Theory and Practice of Computation - Essays Dedicated to Peter Buneman*, pages 174–192, 2013.

[32] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Dl-lite: Tractable description logics for ontologies. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 602–607, 2005.

[33] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[34] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Inconsistency tolerance in P2P data integration: An epistemic logic approach. *Inf. Syst.*, 33(4-5):360–384, 2008.

[35] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. *Artif. Intell.*, 195:335–360, 2013.

[36] P. J. Cameron. Transitivity of permutation groups on unordered sets. *Mathematische Zeitschrift*, 148(2):127–139, Jun 1976.

[37] G. Cantor. Über unendliche, lineare Punktmannigfaltigkeiten. *Math. Anna.*, 23:453–488, 1884.

[38] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. on Knowl. and Data Eng.*, 1(1):146–166, Mar. 1989.

[39] C. Chang and H. Keisler. *Model Theory*. Elsevier, 1998.

[40] H. Chen. A rendezvous of logic, complexity, and algebra. *ACM Comput. Surv.*, 42(1):2:1–2:32, Dec. 2009.

[41] R. Côté, C. of American Pathologists, and A. V. M. Association. *The Systematized Nomenclature of Human and Veterinary Medicine: Numeric*. College of American Pathologists, 1993.

[42] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[43] R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[44] A. Deutsch, A. Nash, and J. B. Remmel. The chase revisited. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 149–158, 2008.

[45] M. Droste. Partially ordered sets with transitive automorphism groups. *Proceedings of the London Mathematical Society*, s3-54(3):517–543, 1987.

[46] J. Edelmann and B. Owsnicki-Klewe. Data models in knowledge representation system: A case study. In *GWAI-86 und 2. Österreichische Artificial-Intelligence-Tagung, Ottstein/Niederösterreich, September 22-26, 1986, Proceedings*, pages 69–74, 1986.

[47] P. Erdős and A. Rényi. Asymmetric graphs. *Acta Mathematica Academiae Scientiarum Hungarica*, 14(3):295–315, 1963.

[48] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[49] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, May 2005.

[50] T. Feder and M. Y. Vardi. Monotone monadic SNP and constraint satisfaction. In *Proc. of the ACM Symposium on Theory of Computing*, pages 612–622, 1993.

[51] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

[52] M. L. Franz Baader, Stefan Borgwardt. Query rewriting for dl-lite with n-ary concrete domains. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 786–792, 2017.

[53] Z. Galil and N. Megiddo. Cyclic ordering is NP-complete. *Theoretical Computer Science*, 5(2):179 – 182, 1977.

[54] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[55] D. Geiger. Closed systems of functions and predicates. *Pacific J. Math.*, 27(1):95–100, 1968.

[56] B. C. Grau, I. Horrocks, M. Krötzsch, C. Kupke, D. Magka, B. Motik, and Z. Wang. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Artif. Intell. Res. (JAIR)*, 47:741–808, 2013.

[57] V. Haarslev, R. Möller, and M. Wessel. The description logic $\text{alcnh}_{\text{r}+}$ extended with concrete domains: A practically motivated approach. In *Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, June 18-23, 2001, Proceedings*, pages 29–44, 2001.

[58] P. Hell and J. Nesetril. *Graphs and Homomorphisms*. Oxford Lecture Series in Mathematics and Its Applications. OUP Oxford, 2004.

[59] P. Hell and J. Nešetřil. On the complexity of h-coloring. *Journal of Combinatorial Theory, Series B*, 48(1):92 – 110, 1990.

[60] W. Hodges. *A Shorter Model Theory*. Cambridge University Press, New York, NY, USA, 1997.

[61] A. Horn. On sentences which are true of direct unions of algebras. *J. Symb. Log.*, 16(1):14–21, 1951.

[62] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 466–471, 2005.

[63] N. Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[64] R. M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pages 85–103, 1972.

[65] Y. Kazakov. Consequence-driven reasoning for horn shiq ontologies. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 2040–2045, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[66] A. Krisnadhi and C. Lutz. Data complexity in the EL family of dls. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, 8-10 June, 2007*, 2007.

[67] A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations: The tractable subalgebras of allen's interval algebra. *Journal of the ACM*, 50:2003, 2001.

[68] M. Krötzsch, S. Rudolph, and P. Hitzler. Complexities of horn description logics. *ACM Trans. Comput. Log.*, 14(1):2:1–2:36, 2013.

[69] R. E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, Jan. 1975.

[70] B. Larose and P. Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410(18):1629–1647, 2009.

[71] G. Ligozat. Corner relations in Allen's algebra. *Constraints*, 3(2/3):165–177, June 1998.

[72] C. Lutz. Reasoning with concrete domains. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 90–95, 1999.

[73] C. Lutz. Interval-based temporal reasoning with general tboxes. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 89–96, 2001.

[74] C. Lutz. Description logics with concrete domains-a survey. In *Advances in Modal Logic 4*, pages 265–296, 2002.

[75] C. Lutz. Pspace reasoning with the description logic ALCF(D). *Logic Journal of the IGPL*, 10(5):535–568, 2002.

[76] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic EL using a relational database system. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 2070–2075, 2009.

[77] C. Lutz and F. Wolter. Non-uniform data complexity of query answering in description logics. In *KR 2012*, 2012.

[78] C. Lutz and F. Wolter. The data complexity of description logic ontologies. *Logical Methods in Computer Science*, 13(4), 2017.

[79] D. Macpherson. A survey of homogeneous structures. *Discrete Mathematics*, 311(15):1599–1634, 2011.

[80] D. Magka, Y. Kazakov, and I. Horrocks. Tractable extensions of the description logic $\mathcal{EL}$ with numerical datatypes. *J. Autom. Reasoning*, 47(4):427–450, 2011.

[81] Y. Matiyasevitch. The Diophantineness of enumerable sets. *Soviet Math. Dokl.*, 11:354–358, 1970.

[82] R. H. Moehring, M. Skutella, and F. Stork. Scheduling with and/or precedence constraints. *SIAM Journal on Computing*, 33(2):393–415, February 2004.

[83] R. Molitor. *Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken.* PhD thesis, RWTH Aachen University, Germany, 2000.

[84] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.*, 7:95–132, 1974.

[85] B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of allen's interval algebra. *J. ACM*, 42(1):43–66, Jan. 1995.

[86] J. Nikiel. *Topologies on Pseudo-Trees and Applications.* Number no. 416 in American Mathematical Society: Memoirs of the. American Mathematical Society, 1989.

[87] C. Nikolaou, E. V. Kostylev, G. Konstantinidis, M. Kaminski, B. C. Grau, and I. Horrocks. The bag semantics of ontology-based data access. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1224–1230, 2017.

[88] J. Opatrny. Total ordering problem. *SIAM J. Comput.*, 8(1):111–114, 1979.

[89] M. Pinsker. Algebraic and model theoretic methods in constraint satisfaction. *CoRR*, abs/1507.00931, 2015.

[90] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[91] W. Rautenberg. *Einführung in die mathematische Logik - ein Lehrbuch mit Berücksichtigung der Logikprogrammierung.* Vieweg, 1996.

[92] R. Rosati. On conjunctive query answering in EL. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, 8-10 June, 2007*, 2007.

[93] F. Rossi, P. v. Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier, 2006.

[94] O. Savkovic and D. Calvanese. Introducing datatypes in *DL-Lite*. In *ECAI 2012*, pages 720–725, 2012.

[95] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 216–226, New York, NY, USA, 1978. ACM.

[96] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, 1991.

[97] A. Tarski. A decision method for elementary algebra and geometry. In B. F. Caviness and J. R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 24–84, Vienna, 1998. Springer Vienna.

[98] W. T. Tutte. The method of alternating paths. *Combinatorica*, 2(3):325–332, 1982.

[99] M. Vilain, H. Kautz, and P. van Beek. Readings in qualitative reasoning about physical systems. chapter Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report, pages 373–381. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[100] D. Zhuk. A proof of csp dichotomy conjecture. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 331–342, Oct 2017.