Leveraging the Structure of Uncertain Data

Antoine Amarilli

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2016-ENST-0021

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TÉLÉCOM ParisTech

Spécialité « Informatique »

présentée et soutenue publiquement par

Antoine Amarilli

le 14 mars 2016

Tirer parti de la structure des données incertaines

Directeur de thèse: **Pierre Senellart**

Jury

M. Serge ABITEBOUL, Directeur de recherche, Inria Saclay	Examinateur
M. Michael BENEDIKT, Professor, University of Oxford	Examinateur
M. Pierre BOURHIS, Chargé de recherche, CNRS CRIStAL	Invité
M. Martin GROHE, Professor, RWTH Aachen University	Rapporteur
Mme Marie-Laure MUGNIER, Professeure, Université de Montpellier	Examinatrice
M. Jacques SAKAROVITCH, Directeur de recherche émérite, CNRS, Télécom ParisTech	Examinateur
M. Pierre SENELLART, Professeur, Télécom ParisTech	Directeur de thèse
Mme Cristina SIRANGELO, Professeure, Université Paris Diderot	Examinatrice
M. Dan SUCIU, Professor, University of Washington	Rapporteur

TÉLÉCOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

On ne sait aujourd'hui comment gérer des faits
Lorsqu'ils sont inexacts ou qu'ils sont incomplets.
Les probabilités font que rien n'est faisable,
Et, sous le monde ouvert, tout est indécidable.
L'objet de cette thèse est de montrer comment
Les probabilités et le raisonnement
Sont à notre portée et à notre mesure
Si règles comme faits sont munis de structure.

On recherche d'abord quelle condition
Sur les faits donnés rend l'évaluation
De requêtes facile et rend la provenance
Aisément calculable étant donné l'instance.
Nous montrons, dans plusieurs des cadres existants,
Que c'est le cas des faits quasi-arborescents ;
Mais, réciproquement, la tâche est infaisable
Si l'instance n'est pas ainsi décomposable.

On étudie ensuite un monde indéfini,
Partiellement connu mais supposé fini,
Et l'on cherche à savoir si notre règle entraîne
Que, sous les faits donnés, la requête est certaine.
On montre que l'on peut, même en haute arité,
Chaque règle bornant la cardinalité
Ou bien ne propageant qu'une unique variable,
Prouver que ce nouveau problème est décidable.

Abstract

The management of data uncertainty can lead to intractability, in the case of probabilistic databases, or even undecidability, in the case of open-world reasoning under logical rules. My thesis studies how to mitigate these problems by restricting the *structure* of uncertain data and rules.

My first contribution investigates conditions on probabilistic relational instances that ensure the tractability of query evaluation and lineage computation. I show that these tasks are tractable when we bound the *treewidth* of instances, for various probabilistic frameworks and provenance representations. Conversely, I show intractability under mild assumptions for *any* other condition on instances.

The second contribution concerns query evaluation on incomplete data under logical rules, and under the *finiteness* assumption usually made in database theory. I show that this task is decidable for unary inclusion dependencies and functional dependencies. This establishes the first positive result for finite open-world query answering on an arbitrary-arity language featuring both referential constraints and number restrictions.

La gestion des données incertaines peut devenir infaisable, dans le cas des bases de données probabilistes, ou même indécidable, dans le cas du raisonnement en monde ouvert sous des contraintes logiques. Cette thèse étudie comment pallier ces problèmes en limitant la *structure* des données incertaines et des règles.

La première contribution présentée s'intéresse aux conditions qui permettent d'assurer la faisabilité de l'évaluation de requêtes et du calcul de lignage sur les instances relationnelles probabilistes. Nous montrons que ces tâches sont faisables, pour diverses représentations de la provenance et des probabilités, quand la *largeur d'arbre* des instances est bornée. Réciproquement, sous des hypothèses faibles, nous pouvons montrer leur infaisabilité pour *toute* autre condition imposée sur les instances.

La seconde contribution concerne l'évaluation de requêtes sur des données incomplètes et sous des contraintes logiques, sous l'hypothèse de *finitude* généralement supposée en théorie des bases de données. Nous montrons la décidabilité de cette tâche pour les dépendances d'inclusion unaires et les dépendances fonctionnelles. Ceci constitue le premier résultat positif, sous l'hypothèse de la finitude, pour la réponse aux requêtes en monde ouvert avec un langage d'arité arbitraire qui propose à la fois des contraintes d'intégrité référentielle et des contraintes de cardinalité.

Remerciements

L'amorce de cette thèse a été mon stage de master avec Pierre Senellart ; à présent qu'elle se termine, c'est Pierre que je souhaite remercier en premier lieu. Ce stage avec lui, et la thèse ensuite, m'ont appris tant de choses que je ne sais pas par où commencer : recherche, rédaction scientifique, informatique pratique et administration système, typographie, cuisine, géographie parisienne... Je resterai toujours impressionné par son efficacité absolument sidérante, par son perfectionnisme et sa fiabilité irréprochables, et par ses capacités apparemment illimitées pour comprendre instantanément les idées souvent alambiquées que je proposais et les reformuler en des arguments compréhensibles. Je pense que Pierre a su m'encadrer d'une façon qui me correspondait parfaitement : il m'a guidé dans la bonne direction, m'a offert plus d'opportunités que je n'en méritais, et a toujours été disponible pour me conseiller, tout en me laissant la plus grande liberté pour travailler comme je l'entendais. Pierre, j'ai vraiment apprécié de travailler et de passer du temps avec toi. Comme tu le disais toi-même de ton propre directeur, et comme nombre de tes étudiants ont pu te le dire avant moi, mon plus grand regret en terminant cette thèse est de cesser d'être ton élève.

Mon stage avec Pierre m'ayant convaincu de poursuivre en thèse avec lui, c'est avec son aide que j'ai pu choisir mon sujet et obtenir un financement ; c'est lui aussi qui m'a mis en contact avec Michael Benedikt et Pierre Bourhis à Oxford, et Tova Milo à Tel Aviv, chez qui je suis parti en stage ensuite. Le premier stage, à Tel Aviv, a été ma première expérience de recherche théorique, dans un contexte fort dépaysant. Je remercie chaudement Tova de m'avoir accueilli, et d'avoir su mener cette recherche avec nous d'une façon si précise, efficace et bienveillante. Je suis heureux d'avoir pu rencontrer l'équipe de Tova en général, particulièrement ma collaboratrice Yael, et Benoît qui m'a contaminé là-bas avec un problème sur lequel nous travaillons encore. Je remercie aussi François pour son généreux dépannage qui m'a sauvé de problèmes de logement difficilement surmontables autrement.

Mon second stage à Oxford m'a fait prendre goût à des questions de logique qui n'ont jamais vraiment cessé de me travailler depuis. Je remercie Michael pour son accueil, et pour sa culture et son expérience, qui m'ont énormément appris, et m'ont permis de développer un pan important de ma thèse avant même qu'elle n'ait officiellement commencé. Je remercie aussi Andreas pour avoir lancé notre recherche dans une direction inattendue mais fructueuse. Je suis heureux d'avoir rencontré de nombreux collègues à Oxford, et de les avoir rejoints au pub plus d'une fois ; en particulier Pierre Bourhis, avec qui j'allais collaborer officiellement par la suite. En tout cas, je remercie Pierre pour son aide là-bas, pour m'avoir indiqué comment survivre à Oxford et quoi y faire, et pour m'avoir invité au *formal dinner*.

En commençant ma thèse proprement dite à Télécom ensuite, j'ai eu la joie de pouvoir y travailler avec de nouvelles personnes. D'abord Daniel (de Tel Aviv,

pendant une brève visite) et Lamine (de Télécom) sur les ordres incertains, puis Ruiming (de Singapour, en visite chez nous) et son encadrant Stéphane, et enfin Pierre Bourhis. La visite de ce dernier à Télécom m'a lancé, un peu par hasard, dans la direction de la tractabilité structurelle sur les bases de données probabilistes, qui a donné à ce manuscrit son titre et la moitié de son contenu. Travailler avec les deux Pierre sur ce sujet a été une expérience particulièrement enrichissante, la rigueur et l'esprit synthétique de l'un complétant parfaitement la créativité et l'imagination de l'autre. Je remercie en particulier Pierre Bourhis pour ses multiples invitations à Lille, pour des discussions aussi intenses que fructueuses, et plus généralement pour le soutien indéfectible qu'il m'a apporté tout au long de ma thèse. J'ai plus récemment commencé à travailler avec Fabian et Luis à Télécom, que j'avais rencontré au MPI avant ma thèse, grâce à la généreuse invitation de Fabian ; plus récemment encore avec Silviu. Je mesure la chance incroyable que j'ai eue de pouvoir travailler sur autant de sujets avec tant de collaborateurs géniaux ; c'est grâce à vous tous que ma thèse a été si prenante et stimulante, et je vous remercie pour tout le temps que nous avons passé à résoudre des problèmes ensemble.

Je suis également très chanceux d'avoir pu beaucoup voyager pendant cette thèse. Je remercie pour cela Michael pour m'avoir invité à Dagstuhl et plusieurs fois à Oxford, Tova pour m'avoir envoyé à Bali et à nouveau invité à Tel Aviv, Meghyn pour son invitation à Montpellier, et Pierre pour m'avoir invité à Eilat et m'avoir organisé deux séjours à Singapour, où je suis heureux d'avoir pu passer du temps avec Mikaël, Miyoung, Qing et Ruiming. Je remercie plus généralement Pierre, Talel, et les organismes qui nous financent, pour avoir permis de prendre en charge mes nombreux déplacements. Je voudrais aussi remercier les compagnons qui ont fait de ces voyages une expérience conviviale : Pierre au Cambodge et en Thaïlande, Andreas en Colombie, Roxana en Chine, Roxana et Benoît en Australie, Ted au Japon, et Oana, Danai et Luis à Marseille, Hyères et Porquerolles.

Quand je n'étais pas ailleurs, j'ai eu le bonheur quotidien de profiter à Télécom d'un environnement accueillant et chaleureux. Je voudrais pour cela remercier tous les membres de mon équipe (y compris les anciens et les membres d'adoption) pour de nombreux déjeuners agrémentés de discussions sur bien des sujets inattendus. Dans l'ordre alphabétique, merci à Albert, Danai, Fabian, Jacob, Jean-Benoît, Jean-Louis, Katerina, Lamine, Luis, Marie, Mauro, Maximilien, Mikaël, Miyoung, Oana, Pierre, Quentin, Roxana, Sébastien, Talel, Thomas et Ziad. Je suis ravi de la diversité du groupe que nous formons, que ce soit en termes de convictions, de goûts, de croyances, ou de nationalités ; et même si je suis souvent trop gêné pour vous le dire, je suis bien heureux que l'heure de vous quitter ne soit pas encore tout à fait venue.

En ce qui concerne la soutenance de thèse, je souhaite remercier Martin Grohe et Dan Suciu pour avoir été mes rapporteurs, et tous les membres de mon jury pour avoir accepté leurs fonctions. À présent que je candidate à des postes de chercheur, je souhaiterais également remercier, pour leur disponibilité et leur soutien : Cristina et Olivier, Joachim, Pierre et Sophie, Luc et Serge, Marie-Laure et Meghyn. En ce qui concerne ce manuscrit, merci à Pablo pour de nombreux conseils, et pour le squelette LaTeX que j'ai utilisé. Merci à Pierre d'avoir eu le courage de le relire en respectant des délais très serrés. Merci à Nina pour avoir identifié une maladresse dans l'épigraphe, merci à mon père pour avoir relu le résumé en français de ce manuscrit, et merci à olasd pour sa relecture de ces remerciements (y compris la présente phrase).

Beaucoup des personnes que j'ai mentionnées jusqu'ici sont bien plus que de

simples collègues, et la frontière entre vie personnelle et vie professionnelle est souvent perméable. Je souhaiterais toutefois conclure ces remerciements en m'adressant à d'autres personnes qui me sont chères et n'ont pas de lien avec mon travail. Ma famille, tout d'abord ; mes parents, mes grands-parents, Victor et Arthur, Étienne et Frédérique, et tous les autres dont je n'essaierai pas de dresser une liste exhaustive. Merci d'avoir toujours été là quand j'en avais besoin. J'aimerais pouvoir vous témoigner plus souvent à quel point vous comptez pour moi.

Enfin, mes amis. C'est principalement vous qui m'avez permis d'échapper de temps à autre à la thèse dont il est ici question, et c'est sans doute grâce à vous tous que je ne suis pas encore complètement fou. Je vous suis immensément reconnaissant pour les efforts que vous déployez inlassablement en ce sens, en m'empêchant de me jeter à corps perdu dans ma recherche, en me venant en aide même quand je suis trop fier ou trop timide pour vous le demander, et en me tirant de ma solitude même quand je n'ai pas l'initiative de vous donner signe de vie. J'espère parvenir à être un jour l'ami que vous mériteriez. Je voudrais vous remercier tous, mais il m'a toujours semblé indélicat d'établir des listes de noms pour ce faire ; permettez-moi donc de terminer par un florilège aléatoire d'allusions et d'expériences où vous pourrez essayer de vous reconnaître.

La recherche d'un punt à Cambridge en hiver
Au plus noir de la nuit à Valence et en mer
Aider l'académique à libérer son texte
Pour faire une soirée avons-nous un prétexte
Le chat alexandrin sur un certain canal
Partez au Canada pour voir si c'est légal
Des crêpes et des jeux occupent le dimanche
Par la cryptomonnaie un geek prend sa revanche
Un repas de promo sauvage à Montsouris
De Paris à la Suisse et de Suisse à Paris
Fêtes du Nouvel An d'un bout de France à l'autre
La balade delphique et l'herbe où l'on se vautre
Un matin du cinq août en chœur à quatre voix
Certains karaokés attirent les Chinois
Sous la plaque banale une grotte interdite
Ne mourrons pas de faim au jeu dans la marmite
Mariage slovaque escale à Budapest
Tour humide à vélo par l'Europe de l'Est
En randonnant là-haut se peut-il que je tombe
L'annuaire à présent est sorti de sa tombe
Un séjour en montagne arrosé de Cointreau
Tarot divinatoire au grand nord ou sous l'eau
La contrainte à l'écrit et sous tant d'autres formes
Des mails toujours plus lents mais toujours plus énormes
À Noël entre amis crêpes et bonne humeur
Savoir si devant vous je deviendrai docteur

Ces souvenirs me sont plus chers que vous ne le croyez peut-être. Merci à vous tous.

Table of Contents

General Introduction

The great success story of database research is the relational model [Codd 1970]. This model has spanned an entire field of theoretical research, and continues to have tremendous practical impact. Relational database management systems now provide a ubiquitous abstraction in computing, like compilers or filesystems: they offer a high-level declarative interface for applications to use, and rely on a rich body of clever but domain-agnostic optimizations implemented in database engines.

Yet, the relational model does not handle *uncertainty* over the data that it manages. In particular, it cannot adequately represent:

Missing values. For instance, in a database of users, it is possible that some users did not fill in some fields when registering, or that new fields were added since they registered.

Incomplete data. The data that you have may not represent all available information; for instance, in a database of information that you are extracting from the Web, you may want to answer a query from what has already been extracted.

Outdated data. For instance, phone numbers or addresses in customer records become incorrect over time.

Wrong data. Data may have been plain incorrect in the first place, for many reasons, e.g., it was incorrectly entered, or relied on false assumptions.

Managing uncertain data, however, has become more and more crucial. Indeed, data is nowadays extracted from natural-language text in random Web pages by automated and error-prone extraction programs [Carlson et al. 2010]; integrated from diverse sources through approximate mappings [Dong, Halevy, and Yu 2007]; contributed by untrustworthy users to collaboratively editable knowledge bases [Vrandečić and Krötzsch 2014]; deduced from the imprecise answers of random workers on crowdsourcing platforms [Amsterdamer, Grossman, Milo, and Senellart 2013; Parameswaran et al. 2012]; or more generally produced by data mining or machine learning techniques. Unlike hand-curated databases, it is no longer possible to assume that mistakes will not happen, or that you can fix all of them while you find them. The data is incomplete and contains errors, and you have to deal with that.

Yet, for now, the only uncertainty management issue widely addressed by mainstream relational database implementations is that of missing values, with the notion of `NULL`s in SQL. Unfortunately, as has been pointed out very early [Grant 1977] the semantics of `NULL`s as defined by the SQL standard [ISO 2008] has many drawbacks:

- `NULL`s can only represent missing values in a record, not missing records, or uncertainty about whether an entire record is correct;
- `NULL`s cannot indicate that several unknown values are the same, or are constrained in a certain way, e.g., chosen among multiple possible values;
- `NULL` evaluation is based on three-valued logics, with very counter-intuitive effects, e.g., the selection of values that are either *equal to 42* or *different from 42* will not return `NULL` values, even though those values should be selected no matter which actual value the `NULL` stands for.

Of course, research has investigated improved semantics for `NULL`s [Imieliński and Lipski 1984], and the debate continues to this day [Libkin 2014]. For now, however, applications have no generic solution that would allow them to perform general uncertain data management tasks. For this reason, they rely on naive solutions: e.g., set a confidence threshold, discard everything below the threshold, consider everything above the threshold as certain. The main exception are domains where uncertainty management is paramount, such as optical character recognition, automated speech recognition (ASR), computational biology, etc.: powerful solutions have been developed in such contexts, such as weighted finite-state transducers [Mohri, Pereira, and Riley 2002] for ASR, but they are *domain-specific*. The vision of *uncertain data management* is therefore that *principled* and *generic* tools should be available to manage noisy and incomplete data in all domains, in the same way that relational databases today are both highly optimized and domain-agnostic.

The database research community has been hard at work to realize this vision, and many theoretical models have been proposed, beyond `NULL`s, to manage uncertain data in a powerful and generic way. For instance, to deal with the issue of *incompleteness* in databases, the notion of *open-world query answering* has been put forward: information not present in the database is unknown rather than false (the so-called *open-world semantics*), and the answers that should be returned for the query are those that are implied by the data and by logical rules provided along with the data. To deal with tuples whose correctness is not certain, probabilistic formalisms [Suciu, Olteanu, Ré, and Koch 2011] such as *tuple-independent databases* (TID) have been proposed: TIDs are relational databases where tuples carry an independent probability of existence. To represent *correlations* between tuples, or to represent the result of queries evaluated over such databases, more expressive formalisms such as *pc-tables* have been put forward. Other formalisms have been developed for other kinds of uncertain data, such as *probabilistic XML* [Kimelfeld and Senellart 2013] for XML documents.

Yet, these efforts still face many challenges. It is hard to define principled frameworks to manage uncertainty and probabilities on unusual data structures, and even harder to represent such information concisely. It is challenging to reason efficiently in the presence of probabilities, as probabilistic query evaluation tends to be much harder than non-probabilistic query evaluation [Dalvi and Suciu 2007]. Further, in the case of open-world query answering under rules, it may be even undecidable to reason over the data, depending on the rule language.

My PhD research has attacked these challenges from a new angle. Rather than developing techniques that can manage *any* kind of data, and studying which tasks

can *always* be tractably performed (e.g., the *safe queries* [Dalvi and Suciu 2012] in the TID formalism), I focus on assumptions on the *structure* of the data. This is motivated by the notion that real-world data is not arbitrary, and may be tractable even in cases where arbitrary data may not be. More specifically, my research throughout my PhD has focused on three main axes:

1. I have investigated which structural conditions on instances ensure the tractability of uncertain data management and probabilistic evaluation, in particular, bounds on the instance *treewidth*.

2. I have studied new uncertainty representation frameworks for unusual data structures, namely, uncertainty frameworks for *ordered* data tuples and values.

3. I have worked on *open-world query answering* for incomplete data, and shown new settings where the structure of logical rules, and their precise language, could ensure decidability for this problem, both in the finite and unrestricted contexts.

The next three sections of this introduction propose a complete overview of my doctoral work on these topics. The thesis itself only focuses on two of these contributions, as is clarified throughout the introduction and summarized at the end. Part of this overview was presented at the SIGMOD/PODS PhD Symposium [Amarilli 2015b].

1 Provenance and Probability on Treelike Instances

The first axis of my PhD research concerns the tractability of uncertain data management, namely provenance computation and probability evaluation, in the setting of treelike instances. This work is presented in Part I of this manuscript; it appeared at ICALP'15 [Amarilli, Bourhis, and Senellart 2015] and was accepted at PODS'16 [Amarilli, Bourhis, and Senellart 2016].

It is surprising how probabilistic query evaluation is considerably harder than regular query evaluation, even for the simple language of *conjunctive queries* (CQs), and in the simple probabilistic model of *tuple-independent instances* (TIDs). Consider for instance a dating website that manages a table of users, indicating their city of residence, and a table indicating messages sent from one user to another. Consider a conjunctive query asking for pairs of users that messaged each other and live in the same city. The evaluation of this query is tractable w.r.t. the database (i.e., in *data complexity*), more specifically it is AC^0 [Abiteboul, Hull, and Vianu 1995], as is the case for any first-order query.

Imagine now that the dating website uses the TID formalism to represent uncertainty about whether a user is looking for a relationship or not, and whether messages express a positive sentiment or not. This could be estimated, for each user and for each message, by, e.g., machine learning or sentiment analysis, assuming independence across users and messages. The dating website now wishes to use this information to determine the *probability* that someone sent a positive message to someone in the same city, and that both users were looking for a relationship, according to this probabilistic data. As it turns out, this task is intractable, specifically, it is #P-hard. Intuitively, because of the correlations caused by the multiple

occurrences of facts, in the worst case it is unlikely that one can do much better than considering the exponential numbers of possible worlds.

In fact, a *dichotomy result* has been shown [Dalvi and Suciu 2012] that characterises the queries for which probabilistic query evaluation is tractable on all input TID instances. However, this result leaves open the question of whether more queries could not be tractable on *reasonable* input instances, in a sense that we must define. We think that this is an important direction because real datasets are not arbitrary, so it may be possible to work around the intractability of probabilistic query evaluation on arbitrary datasets. In fact, a result of this kind was already known: the equivalent of TID for probabilistic XML, namely, the $\text{PrXML}^{\text{mux,ind}}$ model, is known to enjoy tractable probabilistic evaluation [Cohen, Kimelfeld, and Sagiv 2009], intuitively thanks to the tree structure in which probabilistic choices only have a local effect.

The precise question that we ask is then: *on which instance families is probabilistic evaluation tractable, still in data complexity, for all queries in an expressive language?*

We give a complete answer to this question in Part I of this manuscript, by showing a new dichotomy result, centered on the structure of instances rather than queries. More specifically, we show that, on *bounded-treewidth* TID instance families, which are intuitively close to a tree, probabilistic query evaluation is tractable for all queries in the expressive language of *guarded second-order* logic (GSO). Conversely, we show that there are *first-order* queries such that, for *any* choice of instance family of unbounded treewidth, probabilistic query evaluation is intractable in data complexity, assuming arity-two signatures and a mild constructibility requirement.

The upper bound of our dichotomy is proven as a general efficient technique to compute *provenance* representations, or lineages, on bounded-treewidth instances, presented in Chapter 3 of this thesis. More specifically, we use the connection originally shown by Courcelle between queries on treelike instances, and tree automata on tree encodings [Courcelle 1990; Flum, Frick, and Grohe 2002]. We introduce a general representation of the runs of a tree automaton on an uncertain tree, as a *provenance circuit* [Deutch, Milo, Roy, and Tannen 2014], and we lift this to a provenance computation scheme with linear data complexity for GSO queries on treelike instances.

We then show how this provenance can be used to construct tractable lineage representations, namely *OBDDs* [Bryant 1992; Olteanu and Huang 2008] and *d-DNNFs* [Darwiche 2001]. We use this in Chapter 4 to show that probabilistic query evaluation for GSO queries on TID instances of bounded treewidth has ra-linear-time data complexity, i.e., linear time up to the cost of arithmetic operations. To extend these results to more expressive formalisms, we introduce *pcc-instances*, a circuit-based variation of pc-tables [Huang, Antova, Koch, and Olteanu 2009; Green and Tannen 2006] with a natural definition of treewidth covering both the instance and correlations. We use this to derive bounded-treewidth tractability results for pc-tables, BID tables [Barbará, Garcia-Molina, and Porter 1992; Ré and Suciu 2007], and probabilistic XML, capturing in particular the result of [Cohen, Kimelfeld, and Sagiv 2009]. We also show a connection to the results of [Dalvi and Suciu 2012], by explaining how the tractability of *inversion-free* unions of conjunctive queries can be explained by a lineage-preserving rewriting of their input instances to a bounded-pathwidth instance.

We additionally derive results in Chapter 5 about the tractable computation of provenance representations on treelike instances in expressive semirings, beyond

Boolean lineages, via a connection to semiring provenance [Green, Karvounarakis, and Tannen 2007]. Indeed, provenance representations have more uses than probabilistic query evaluation: they allow us to keep track of the link between query result and input instance, and can be used to solve many problems such as multiplicity counting, access right management, view maintenance, etc. We thus extend our definitions of provenance for trees and treelike instances to more general semirings that capture these problems. We then show, by a variation of our techniques, that provenance circuits for the universal semiring $\mathbb{N}[X]$ can be computed in linear time data complexity on treelike instances for unions of conjunctive queries, down from the polynomial-time complexity of computing them in the general case.

We show the lower bound of our dichotomy in Chapter 6, relying on recent polynomial bounds [Chekuri and Chuzhoy 2014a] on the extraction of planar graphs as minors of high-treewidth graphs, which we apply to arbitrary unbounded-treewidth graph families provided they are constructible in a certain sense. This allows us to show, on arity-two signatures, the #P-hardness under RP reductions of a specific first-order (FO) query on *any* such family of input instances. In other words, any condition on instances that ensures the tractability of probabilistic query evaluation for FO must imply a bound on treewidth, or non-constructibility of the instances. By contrast, in the context of non-probabilistic evaluation and match counting, we show similar results with a monadic second-order query, revisiting and improving earlier results [Kreutzer and Tazari 2010; Ganian, Hliněný, et al. 2014].

We extend this result to a stronger dichotomy, for the more stringent requirement of restricting instances to ensure that queries have *tractable OBDD lineages*. While we again know that bounded-treewidth ensures this for all GSO queries, we show that even a query in $\mathrm{UCQ}^{\neq}$, a union of conjunctive queries with disequalities, can have no tractable OBDD representations on *any* unbounded-treewidth, constructible, arity-two instance family. We conclude our study by a characterization of the *connected* $\mathrm{UCQ}^{\neq}$ queries that are thus intractable (in terms of OBDDs) on any such instance family, in a query-centric *meta-dichotomy* result.

We see our dichotomy results as a foundation for a different study of probabilistic query evaluation, that would be neither instance-centric, nor query-centric. Our hope is that our tractable constructions for bounded treewidth instances can be combined with techniques for safe query evaluation, and could thus achieve both theoretical and practical tractability on probabilistic evaluation tasks where the *interaction* between query and instance is tractable, even though neither of them would be tractable in isolation.

2 Uncertainty on Ordered Data

The second axis of my PhD research concerns uncertainty on data with an order on tuples or on numerical values. This work is not presented in this manuscript: the first subsection is available as a preprint [Amarilli, Ba, Deutch, and Senellart 2016]; the second is also available as a preprint [Amarilli, Amsterdamer, Milo, and Senellart 2016] and was sketched at the UnCrowd workshop [Amarilli, Amsterdamer, and Milo 2014b].

There are many data management contexts where we must take into account an *order* relation on data values (such as dates or numerical numbers), or on the tuples

themselves (e.g., a list of log entries, or a ranked list of results). This section present my work on uncertainty for order representation, first in a setting with uncertain order on relational tuples, and then in a setting with partially ordered numerical values.

2.1 Representing Order Uncertainty on Relational Tuples

To manage ordered tuples in the relational model, an interesting phenomenon is that uncertainty on the order arises even when we apply the standard relational algebra operators to input relations that are certain in terms of order. For a practical example, consider a travel website where you can search for hotels and which ranks them by quality. You are a party of four, and you want either two twin rooms, or one room with four beds; but the website does not allow you to search for both possibilities at once. So you perform one search, and then the other, and you obtain two ordered lists of hotels, of which you want to compute the union. However, the *order* on the union list is uncertain: it depends on the website's estimation of quality, which you do not control. It is not fully unspecified, however, because two hotels that occurred only in the first list, or only in the second list, would probably keep the same order in the union.

To address this phenomenon, our work [Amarilli, Ba, Deutch, and Senellart 2016] proposes operators for the relational algebra that apply to partially ordered relations, under the *bag* semantics, in the spirit of [Grumbach and Milo 1999]. This allows us to combine totally and partially ordered relations with the relational algebra, to integrate data and compute new ordered relations, representing all possible consistent ordering choices. This resembles *rank aggregation* techniques [Fagin, Lotem, and Naor 2001; Jacob, Kimelfeld, and Stoyanovich 2014; Dwork, Kumar, Naor, and Sivakumar 2001] to reconcile ordered lists of results, but these methods are *quantitative*, i.e., if something appears close to the top in most lists, then it should also do so in the result. We focus instead on ways to represent *all* consistent ordering choices. Of course, a straightforward approach for this would be to use existing relational uncertainty frameworks, such as c-tables, to represent the uncertainty on tuple positions, but this would not work well: if we know that a result must come *before* another one, the dependency on numerical ranks becomes very tedious to represent.

The semantics that we define for relational algebra thus allows us to combine ordered data in a principled way. However, order is maintained as implicit information, and cannot be *queried* by our operators. To address this, we propose a general order-dependent *accumulation* operator on relations with uncertain order, which we can use to compute what we want to find out about the order. Accumulation simply computes all possible concatenations of tuple values, for all possible orders, in a given monoid. We can use it for queries on the order, e.g., "is it certain that all hotels in this district are better than all hotels in that district?"

The main technical contribution of our work is to study the *complexity* of query evaluation in this scheme, more specifically, the complexity of computing *possible* and *certain* answers for *instance* possibility and certainty [Antova, Koch, and Olteanu 2007], and for aggregation. Surprisingly, it is already intractable, i.e., respectively NP-hard and coNP-hard, to determine whether a query result (i.e., a full totally ordered relation) is possible or certain, even for very simple queries, though certainty (unlike possibility) is tractable in the absence of accumulation, and more generally

when accumulating in a *cancellative* monoid.

We therefore show how hardness can be mitigated by restricting the *structure* of the input ordered relations, using order-theoretic measures. We show tractability of our problems, for some subset of the operators, when the input relations have constant poset *width* [Brandstädt, Le, and Spinrad 1987]: we show how such bounds are preserved on query results, and show the tractability of possible and certain answers in this context, through a dynamic algorithm. We show a similar result when the input relations are almost totally unordered, introducing the new poset measure of *ia-width* to define this. We also extend our results under an additional duplicate elimination operator, to go back from bag semantics to set semantics.

2.2 Completing Missing Numerical Values

In different work, I have focused [Amarilli, Amsterdamer, Milo, and Senellart 2016] on *top-k query evaluation* on numerical values which are unknown, but for which we know partial order constraints and some exact values.

This work is inspired by crowd data sourcing scenarios [Amsterdamer, Grossman, Milo, and Senellart 2013; Parameswaran et al. 2012], where we wish to extract information from a crowd of users by asking questions. It is often the case that we wish to extract many interdependent numerical values from the crowd. For instance, to consolidate the catalog of a Web store, we may wish to determine the compatibility of each item in the catalog with each category of a taxonomy of products, where we assume that compatibility scores are partially ordered following the taxonomy: the "shirt" category is more specific than the "clothing" category, so if an item fits in the "shirt" category, then it must also fit in the "clothing" category.

The simplest approach would be to query the crowd about each of the categories, but in general we cannot afford to do that. Indeed, every crowd query that you make introduces latency (you need to wait for the answers to arrive) and monetary costs (you have to pay the workers). For this reason, we studied principled ways to *interpolate* the values that we do not have, using the known partial order *structure* on them, from those values that we have already obtained from the crowd. The goal is to find the top-k items with the highest expected value (here, the top-k categories in which our product can be filed) given the limited information that we have.

While completing a *total* order of values is well-understood and can be done with linear interpolation, it appears that the general question of interpolating on a *partial* order of unknown values had not been studied before. We propose a definition for this problem as that of computing the expected value of each unknown variable in the convex polytope induced by the partial order constraints, taking the uniform distribution as our prior. We show an algorithm to solve the interpolation problem in $\mathrm{FP}^{\#\mathrm{P}}$, and show that the task is #P-hard, based on results on partial order theory [Brightwell and Winkler 1991]. We extend this to show that it is already hard to simply decide which item has the highest expected value, even without computing the value. We also show tractable approaches: we can design a fully polynomial randomized approximation scheme for the interpolation problem, using a scheme to sample convex polytopes [Kannan, Lovász, and Simonovits 1997], and we show that interpolation (for our principled definition) is tractable for tree-shaped taxonomies, using a dynamic algorithm.

3 Decidability for Open-World Query Answering

The third axis of my PhD research concerns open-world query answering on incomplete data whose structure is constrained by logical constraints, studying new logical fragments for which this task is decidable. The work in the first subsection was published at IJCAI'15 [Amarilli and Benedikt 2015a] but is not presented in this manuscript. The work in the second subsection is presented in Part II of this manuscript; it was published as an extended abstract at LICS'15 [Amarilli and Benedikt 2015b] and an full version is currently undergoing peer review [Amarilli and Benedikt 2016].

3.1 Bridging Approaches for Open-World Query Answering

The *open-world query answering problem* (OWQA) asks, given a database I, logical constraints Σ, and a query q, to compute the answers to q that are *certain* under Σ given I; in other words, it asks for the answers to q which are true on *all* possible completions of I that satisfy the constraints Σ. The OWQA problem thus allows us to reason about an *incomplete* database I, by constraining the *structure* of its completions, following logical rules that we know about the world. OWQA is of course undecidable if arbitrary logical rules are allowed in Σ, so there has been much research on finding expressive logical languages for which OWQA remains decidable, and hopefully has a reasonable complexity.

This research question has been studied in at least three different contexts:

Relational databases, originally under the equivalent rephrasing of *query containment under constraints* [Johnson and Klug 1984], and then in the context of incomplete databases [Calì, Lembo, and Rosati 2003a].

This approach uses as logical rules the classical constraints of database theory, e.g., tuple-generating and equality-generating dependencies [Abiteboul, Hull, and Vianu 1995], but the interaction between both kinds of dependencies was quickly seen to lead to undecidability [Mitchell 1983].

Description logics, or DLs, which focus on scalability in the instance I, and on studying the precise complexity tradeoff when allowing more or less expressive constraints in Σ.

DLs work on *arity-two data*, i.e., labeled graphs, which is less expressive than relational databases; however, in this context, OWQA remains decidable even with very expressive constraints, for instance disjunction, negation, and functionality assertions (a form of equality-generating dependencies).

Existential rules, which essentially amount to tuple-generating dependencies, but for which specific decidable classes have been identified, e.g., frontier-guarded [Baget, Leclère, and Mugnier 2010].

Existential rules are not as expressive as DLs, and can only say that a conjunction of facts implies another conjunction of facts. However, unlike DL rules, they can express constraints on arbitrary-arity facts.

My first contribution to the study of OWQA [Amarilli and Benedikt 2015a] is to draw bridges between the approaches of DLs and existential rules. Specifically, the goal is to design languages to reason about incomplete data where both existential

rules and DL rules are allowed, so as to have the best of both worlds: expressive DL rules on arity-two facts, and existential rules on higher-arity facts. In fact, as our main focus is on the *decidability* of OWQA, we do not restrict to specific DLs, but allow constraints expressed in GC^2, the guarded two-variable fragment of first order logic with counting quantifiers, an expressive arity-two logical formalism. Indeed, OWQA for GC^2 is known to be decidable [Pratt-Hartmann 2009], which covers the decidability of OWQA for many description logics (see, e.g., [Kazakov 2004] for the connection).

Our work pinpoints which features of these two language families are dangerous and give an undecidable language when mixed. The main problematic feature of DLs (or GC^2) are *functionality assertions* (or counting quantifiers): being able to say that, e.g., a person has *only one* place of birth. If we want to express such constraints, we must disallow two problematic features of existential rules. The first is *exporting two variables*, intuitively, e.g. "If *someone* was born in *some* country, then *that person* has lived in *that country*". Hence, we restrict to the *frontier-one* fragment of existential rules [Baget, Leclère, Mugnier, and Salvat 2009], intuitively requiring that each existential rule consequence should depend on only one variable of the hypothesis: "If *someone* won a literary prize, then *they* wrote some book." The second problematic feature is the possibility to assert *cyclic* patterns in the head of frontier-one rules, which causes undecidability; we introduce a fragment of *head-non-looping* frontier-one rules to disallow this.

We can then show the decidability of OWQA for the combined language of GC^2 constraints and head-non-looping frontier-one rules. The argument proceeds with a *treeification* construction to rewrite the rules, eliminating bad cycles in their body if there are any, and thus translating to the so-called *fully-non-looping* fragment. We then show that the resulting rules can be *shredded* to GC^2 on a fully binary signature. We extend our results to allow the higher-arity *functional dependencies* (FDs) from database theory, in addition to the GC^2 constraints and existential rules, restricting the interaction of FDs with the rules using the existing *non-conflicting condition* [Calì, Gottlob, and Pieris 2012].

Our results suggest that combining the DL and existential rule approaches could ensure the decidability (and maybe tractability) of OWQA for more expressive *hybrid* logical languages.

3.2 Finite Open-World Query Answering

My second contribution to the study of OWQA concerns the standard context of relational databases. In this setting, an important difference is that the underlying world is often assumed to be *finite*. Specifically, we perform *finite* OWQA: instead of considering *all* completions of the instance I that satisfy the constraints Σ, and finding the certain answers to the query q over them, we restrict our attention to the *finite* completions.

This slight difference in problem phrasing is motivated by the intuitive assumption that databases should be finite. As it turns out, this assumption can make a difference. Consider for instance the following information about an organization:

- The database I: "Jane advises John"
- A first constraint in Σ: "Each advisee is also the advisor of someone". In

database parlance, this is an *inclusion dependency*, a special kind of tuple-generating dependency.

- A second constraint in Σ: "No advisee has two different advisors". This is a *functional dependency*, a special kind of equality-generating dependency.

Consider the OWQA problem for I, Σ, and the Boolean query q that asks whether someone advises Jane. The first constraint allows us to deduce that John, as he is advised by Jane, advises someone else, say Janice; Janice herself advises someone else, say Jack. In general, this could go on indefinitely, and we cannot deduce that someone advises Jane. However, if we also assume *finiteness*, then the chain has to stop somewhere: someone along this chain (Jennifer, say) must advise someone that we already know about. Using the rule that no one is advised by two different people, we deduce that Jennifer must be advising Jane.

While this example is simple, the general impact of finiteness on OWQA is currently very poorly understood. Indeed, OWQA is often studied via *universal models*, obtained, e.g., by the *chase*; or by *unraveling* techniques. However, these models are generally infinite, so these tools do not apply to finite OWQA.

What we do know is that, for some logical languages, finite and unrestricted OWQA are equivalent; we then call these languages *finitely controllable*. Such a result was first shown for inclusion dependencies in [Rosati 2006; Rosati 2011], and later generalized [Bárány, Gottlob, and Otto 2010] to the guarded fragment. Another such result was more recently shown in [Gogacz and Marcinkowski 2013] for the *sticky Datalog* fragment of [Calì, Gottlob, and Pieris 2010]. All of these results have highly technical proofs. None of them, however, allows functional dependencies, or equality-generating dependencies of any kind, so they do not cover the previous example.

Finite OWQA is better understood on *arity-two* signatures, because we know that it is decidable even for languages where it is not finitely controllable, and even for languages allowing functional dependencies. This is the case for GC^2 by a specific argument [Pratt-Hartmann 2009], but it is also the case for some description logics [Rosati 2008; Ibáñez-García, Lutz, and Schneider 2014], following an interesting general method: complete the constraints by a *finite closure* procedure, that deduces all consequences of the constraints that hold in the finite, and show finite controllability of the resulting constraints. Thus, this context covers our previous example, but only in the *arity-two* setting: if the advisor–advisee relation also included more information (e.g., a rating), then we could no longer express it in this context.

Our work [Amarilli and Benedikt 2015b; Amarilli and Benedikt 2016] reuses this finite closure approach, but applies it to arbitrary-arity signatures, for database constraints allowing both inclusion dependencies (IDs) and functional dependencies (FDs). Hence, it captures the previous example no matter the arity of the predicates. However, as OWQA is generally undecidable under IDs and FDs [Calì, Lembo, and Rosati 2003a], we must make another restriction: we restrict to *unary* inclusion dependencies (UIDs), i.e., those which are *frontier-one* rules: they export only a single variable from the body to the head, as does the inclusion dependency in the example. This avoids undecidability, and in fact a finite closure procedure for this language is already known [Cosmadakis, Kanellakis, and Vardi 1990]. We thus show

that UIDs and FDs are finitely controllable once this finite closure procedure has been applied, which proves the first decidability result for finite OWQA on arbitrary-arity signatures under a natural language featuring both IDs and FDs.

The proof of this result is highly technical, and adapts various techniques from previous work:

- the use of *k-bounded simulations* to preserve small acyclic queries [Ibáñez-García, Lutz, and Schneider 2014],
- a partition of UIDs into *connected components* that have limited interaction, which we satisfy component-by-component [Cosmadakis, Kanellakis, and Vardi 1990; Ibáñez-García, Lutz, and Schneider 2014],
- a *finite chase* procedure that reuses sufficiently similar elements [Rosati 2011],
- a product construction using *groups of large girth* to blow up cycles [Otto 2002].

The proof is presented in Part II of the manuscript.

Structure of the Manuscript

I chose to limit the scope of this manuscript, and to focus on a subset of the contributions surveyed above. Specifically:

- Part I of the manuscript corresponds to Section 1 above, published as [Amarilli, Bourhis, and Senellart 2015] and [Amarilli, Bourhis, and Senellart 2016].
- Part II of the manuscript corresponds to Section 3.2 above, published as [Amarilli and Benedikt 2015b] and submitted as [Amarilli and Benedikt 2016].

Both parts can be read independently. Further:

- Sections 2 and 3.1 above, while directly relevant to the topic of my thesis, are not presented in the manuscript: they correspond to publications [Amarilli, Amsterdamer, and Milo 2014b; Amarilli and Benedikt 2015a] and preprints [Amarilli, Ba, Deutch, and Senellart 2016; Amarilli, Amsterdamer, Milo, and Senellart 2016]
- Some additional work performed during my PhD, but which is not directly related to my thesis topic, is surveyed in Appendix A.

Part I

Provenance and Probability on Treelike Instances

Chapter 1

Introduction

This first part of my thesis studies probabilistic query evaluation and provenance computation under the instance-based tractability requirement of bounded treewidth. It presents my work with Pierre Bourhis and Pierre Senellart, published at ICALP'15 [Amarilli, Bourhis, and Senellart 2015] and accepted for publication at PODS'16 [Amarilli, Bourhis, and Senellart 2016]. The latter covers Sections 3.5, 3.6, and 4.5, as well as Chapter 6; the former presents the other results.

To represent uncertain information in a relational database, a common need is to insert tuples which have a probability of being wrong. The simplest possible model to express this are *tuple-independent databases* (TID), where each tuple in the database is annotated with an independent probability of being present. TID instances are thus a concise representation of a *probability distribution* on possible instances, namely, possible subsets of the tuples, with probability given according to the product distribution.

However, in this simple model, and even for simple queries, we cannot hope to have tractable query evaluation techniques. Indeed, for many simple Boolean conjunctive queries, it is intractable to compute the probability that the query holds on the possible worlds of the TID instance, even from the angle of *data complexity*, i.e., complexity in the input TID instance, with the query being fixed. The investigation of this issue has culminated with the dichotomy result of [Dalvi and Suciu 2012]: there is a tractable algorithm to evaluate some queries in PTIME, intuitively by applying some simplification rules, and *all* queries for which this algorithm fails are intractable over arbitrary TID instances.

During my thesis, I revisited this problem by studying it from the point of view of *instances*. Indeed, the query-based result of [Dalvi and Suciu 2012] shows that, when *all* instances can be given as input, some queries are easy and all others are hard. This is motivated by the fact that the *queries* posed by users in real life are not arbitrary, so it is interesting to find simple cases that are tractable. However, real-life *data* is not arbitrary either, and it could be the case that even hard queries become tractable when restricted to simple datasets. One hint in this direction was shown in [Cohen, Kimelfeld, and Sagiv 2009]: on the XML analogue of TID instances, we can tractably evaluate any query that we can represent as a tree automaton. Can this be covered by a general result on tractable instances of a certain shape?

This part of my thesis presents our answer to this question: an *instance-based* dichotomy result, formulated using the criterion of instance *treewidth*. Intuitively, the treewidth of a relational instance measures how close it is to a tree: XML documents

can be seen as a relational instance of treewidth 1. The main result on probability evaluation given in this thesis (Theorem 6.1.2) is thus the following dichotomy:

- On families of input instances with treewidth bounded by a constant, the probability evaluation problem is *always* tractable in data complexity. In fact, this tractability extends beyond the simple TID formalism, and beyond the unions of conjunctive queries studied in [Dalvi and Suciu 2012], all the way to *guarded second-order* queries on relational and XML models with expressive (but bounded-treewidth) correlations.

- On instances of unbounded treewidth, probability evaluation is intractable. This is no surprise if arbitrary input instances are allowed, but we can in fact show something much stronger: on arity-two signatures, and under mild constructibility assumptions, there is a first-order query for which probabilistic evaluation is intractable on *any* input instance family of unbounded treewidth, no matter what other conditions we impose on the instances.

Instance treewidth thus appears to be *the* correct instance-based measure of the tractability of probabilistic query evaluation. Bounding it ensures tractability (in data complexity) for expressive query languages and probabilistic frameworks. Conversely, on arity-two signatures, *any* condition on the underlying instances that ensures tractability *must* imply a bound on treewidth, or inconstructibility in some sense.

Both directions of this dichotomy result are proven in this part of the thesis, and are extended in several directions. The tractability results are proven in Chapters 3–5, and the intractability results are proven in Chapter 6. Both kinds of results are proven using very different techniques, so these two directions are mostly independent.

Tractability results. Our tractability results on bounded-treewidth instances are shown using the connection to tree automata pioneered by the work of Courcelle [Courcelle 1990]. We extend this connection from query evaluation to probability evaluation, and in fact more generally to *provenance computation*, following the general technique of computing query probability via a Boolean lineage of the query. Our basic Boolean provenance construction for treelike instances is shown in Chapter 3; we then present its consequences for probability evaluation in Chapter 4, and its extension to *semiring provenance* in Chapter 5. These two chapters both depend on Chapter 3 but are otherwise independent.

The general Boolean provenance construction in Chapter 3 is based on a new notion of provenance for tree automata on trees, for which we show in Section 3.1 a linear time construction of *provenance circuits*. We then review in Section 3.2 the general translation from treelike instances to trees, and from guarded second-order (GSO) queries to tree automata, identifying a property of *subinstance-compatibility* that allows us to extend the translation to provenance computation. Section 3.3 relies on this property to lift the previous results on trees and show that provenance circuits can be tractably computed (in data complexity) for queries on treelike instances.

We conclude Chapter 3 by investigating which *other* kinds of provenance representations can be computed using our methods. We first show in Section 3.4 that *monotone* provenance circuits can be computed for monotone queries, which we will reuse for more expressive provenance representations in Chapter 5. We then show in Section 3.5 that we can compute polynomial-size OBDDs and linear-size d-DNNFs

representations of the lineage, that we will use for probability evaluation in Chapter 4. Last, in Section 3.6, we justify our focus on *circuit* representations of provenance, showing that representations as *Boolean formulae* are necessarily less concise.

We use our provenance representations for two distinct purposes. First, in Chapter 4, we apply them to our original focus of *probabilistic query evaluation*.

We first show in Section 4.1 that our d-DNNF representations of lineage imply that probability evaluation for GSO queries is tractable in data complexity on bounded-treewidth TID instances: in fact, it is *linear-time*, up to the cost of arithmetic operations.

We then extend this tractability result to other probabilistic models that have been studied in the literature [Suciu, Olteanu, Ré, and Koch 2011; Kimelfeld and Senellart 2013], in particular those that allow *correlations* between probabilistic events, which TID cannot capture. We do this by introducing our own model of *pcc-instances* in Section 4.2, where correlations between tuples are represented as a circuit annotation. We show the tractability of GSO query evaluation on *treelike* pcc-instances, when the treewidth of the tuples and annotations are simultaneously bounded: we do so by rewriting pcc-instances to TIDs, rewriting the query to evaluate the annotation circuit, and applying our previous results.

We then leverage this model to show bounded-treewidth tractability results for existing relational probabilistic models (Section 4.3), namely, pc-instances and block-independent-disjoint instances. We also apply it to probabilistic XML (Section 4.4), re-proving the tractability of $\mathsf{PrXML}^{\mathsf{mux,ind}}$ (as already shown in [Cohen, Kimelfeld, and Sagiv 2009]) and showing that the more general $\mathsf{PrXML}^{\mathsf{fie}}$ model is tractable if we impose a *bounded event scopes* condition.

We conclude Chapter 4 with two other applications of our results. The first one, in Section 4.5, connects our approach to the query-based result of [Dalvi and Suciu 2012]. More specifically, we re-prove that probability evaluation for *inversion-free UCQs* is tractable on all input instances [Jha and Suciu 2013]. We do so from an instance-based perspective: given a query in this language, we can rewrite the input instance to make it bounded-treewidth, without changing the lineage of the query, and we can then apply our results. Our second application (Section 4.6) studies the problem of counting *query matches*: we connect this task to the probabilistic evaluation problem, and re-prove with our methods the bounded-treewidth tractability of GSO match counting that had been shown in [Arnborg, Lagergren, and Seese 1991].

The second use of our provenance representations is to extend them to the context of *semiring provenance* [Green, Karvounarakis, and Tannen 2007]. Indeed, it is natural to ask whether our Boolean provenance constructions can be generalized to more general semiring provenance representations, which capture many tasks beyond probability evaluation [Cheney, Chiticariu, and Tan 2009; Karvounarakis and Green 2012]. In particular, our Boolean provenance circuits are reminiscent of the recently introduced semiring provenance circuits [Deutch, Milo, Roy, and Tannen 2014].

We accordingly show in Chapter 5 how our constructions can be extended to general semirings. We first present the basics of provenance semirings in Section 5.1, explaining why our monotone provenance constructions in Section 3.4 already capture provenance in the semiring of monotone Boolean functions. We then make the choice of restricting the query language from GSO to unions of conjunctive queries with disequalities ($\mathrm{UCQ}^{\neq}$), a choice that we justify in Section 5.4.

Under this restriction, we revisit in Section 5.2 our constructions of provenance for tree automata in Section 3.1, extending them to the universal $\mathbb{N}[X]$-*provenance*. To do this, we must capture the *multiplicity* of node uses, which we do by extending the alphabet of trees beyond Boolean annotations, and the *multiplicity* of derivations, which we count as the number of automaton runs. We then extend these results to treelike instances in Section 5.3, generalizing Section 3.3, which allows us to show a linear-time $\mathbb{N}[X]$-provenance construction for $\mathrm{UCQ}^{\neq}$ on treelike instances.

To summarize, when instance treewidth is bounded, our results show that the Courcelle translation from treelike instances to trees, and from MSO and GSO to tree automata, can be extended to support provenance computation, even in expressive semirings (for $\mathrm{UCQ}^{\neq}$), and implies tractable query evaluation on probabilistic models, even under expressive bounded-treewidth correlations.

Intractability results. Our intractability results in Chapter 6 show a converse to the above results: query evaluation for some first-order (FO) query is hard on any family of probabilistic TID instances if their treewidth is unbounded. To prove this, however, we must assume that the family is *constructible*, namely, that high-treewidth instances in the family can be efficiently built. Further, we rely on technical tools that apply only to graphs, not arbitrary arity instances, so we must restrict to *arity-two* signatures. More specifically, we use a recent polynomial bound [Chekuri and Chuzhoy 2014a] on the result of [Robertson and Seymour 1986] about extracting arbitrary planar graphs as minors of high-treewidth families.

From our main lower bound on probabilistic query evaluation in Section 6.1, we derive lower bounds using the same methods in Section 6.2 for non-probabilistic query evaluation on subinstance-closed families. Thanks to our use of [Chekuri and Chuzhoy 2014a], our results improve existing bounds for the same problem [Kreutzer and Tazari 2010; Ganian, Hliněný, et al. 2014]. We also study the problem of counting query matches in Section 6.3.

Last, we show lower bounds for a more stringent notion of tractability, namely, the existence of polynomial-size OBDD representations of query lineage. In Section 6.4, we accordingly construct a $\mathrm{UCQ}^{\neq}$ query q_{p} which we show to have no polynomial-size OBDDs on *any* constructible arity-two unbounded-treewidth instance family. Of course, this does not imply that probabilistic query evaluation for q_{p} could not be tractable for different reasons (see examples in [Jha and Suciu 2013]). However, it gives us a dichotomy on the existence of polynomial-size OBDDs, even for the restricted language of $\mathrm{UCQ}^{\neq}$: while the $\mathrm{UCQ}^{\neq}$ q_{p} has no polynomial OBDDs (under our assumptions) unless instance treewidth is bounded, we can always construct polynomial OBDDs for queries on bounded-treewidth instances (see Section 3.5). Further, for *connected* $\mathrm{UCQ}^{\neq}$ queries, we are able to show a query-based *meta-dichotomy* (in Section 6.5) on the existence of polynomial OBDD representations: the meta-dichotomy classifies connected $\mathrm{UCQ}^{\neq}$ in *intricate queries*, that *never* have polynomial-size OBDDs on unbounded-treewidth families under our assumptions, and the other queries which are shown to have *constant-width* OBDDs on some well-chosen unbounded-treewidth instance family.

We give preliminary notions in the next chapter, and then start with the tractability results by showing our bounded-treewidth Boolean provenance construction in Chapter 3.

Chapter 2

Preliminaries

2.1 Trees and Tree Automata

Trees. Given a fixed *alphabet* Γ, we define a Γ-*tree* $T = (V, L, R, \lambda)$ as a finite set of *nodes* V, two partial mappings $L, R : V \to V$ that associate an internal node with its left and right child, and a *labeling function* $\lambda : V \to \Gamma$. Unless stated otherwise, the trees that we consider are *rooted*, *ordered* (i.e., there is a total order on the children on nodes), *binary*, and *full* (i.e., each node has either zero or two children). We write $n \in T$ to mean $n \in V$.

We say that two trees T_1 and T_2 are *isomorphic* if there is a bijection between their node sets which preserves children and labels (we simply write it $T_1 = T_2$). We say that two trees T_1 and T_2 have *same skeleton* if they are isomorphic except for labels.

Tree automata. A *bottom-up nondeterministic tree automaton* on Γ-trees, or Γ-*bNTA*, is a tuple $A = (Q, F, \iota, \delta)$ of a finite set Q of *states*, a subset $F \subseteq Q$ of *accepting states*, an *initial relation* $\iota : \Gamma \to 2^Q$ giving possible states for leaves from their label, and a *transition relation* $\delta : Q^2 \times \Gamma \to 2^Q$ determining possible states for internal nodes from their label and the states of their children.

A *run* of A on a Γ-tree $T = (V, L, R, \lambda)$ is a function $\rho : V \to Q$ such that for each leaf n we have $\rho(n) \in \iota(\lambda(n))$, and for every internal node n we have $\rho(n) \in \delta(\rho(L(n)), \rho(R(n)), \lambda(n))$. A run is *accepting* if, for the root n_r of T, we have $\rho(n_\mathrm{r}) \in F$. We say that A *accepts* T (written $T \models A$) if A has an accepting run on T. Tree automata capture usual query languages on trees, such as MSO [Thatcher and Wright 1968] and tree-pattern queries [Neven 2002].

A *bottom-up deterministic tree automaton* on Γ-trees, or Γ-*bDTA*, is a Γ-bNTA $A = (Q, F, \iota, \delta)$ such that $\iota(\gamma)$ and $\delta(q_1, q_2, \gamma)$ are singletons for all $\gamma \in \Gamma$ and $q_1, q_2 \in Q$. In this case, we abuse notation and see ι as an *initial function* $\iota : \Gamma \to Q$ and δ as a *transition function* $\delta : Q^2 \times \Gamma \to Q$. It is clear that, on any Γ-tree T, a Γ-bDTA A has exactly one run, which may or may not be accepting.

It is well known that, given a Γ-bNTA A, we can construct a Γ-bDTA A' that accepts the same language, namely, $T \models A$ iff $T \models A'$ for any Γ-tree T: see, for instance, [Comon et al. 2007].

2.2 Boolean Functions, Formulae, and Circuits

Boolean functions. A (Boolean) *valuation* of a set of *variables* X is a function $\nu : X \to \{0,1\}$. A *Boolean function* is a function from the set of valuations of X to $\{0,1\}$. We write $\nu \leqslant \nu'$ whenever, for all $x \in X$, $\nu(x) = 1$ implies $\nu(x') = 1$, and we call a Boolean function φ *monotone* if for any two valuations $\nu \leqslant \nu'$, if $\varphi(\nu) = 1$ then $\varphi(\nu') = 1$.

Boolean circuits. A *Boolean circuit* is a directed acyclic graph $C = (G, W, g_0, \mu)$ where G is a finite set of *gates*, $W \subseteq G \times G$ is a set of *wires* (edges), $g_0 \in G$ is a distinguished *output gate*, and μ associates each *gate* $g \in G$ with a *type* $\mu(g)$ that can be inp (*input gate*, with no incoming wire in W), $\neg$ (NOT-gate, with exactly one incoming wire in W), $\wedge$ (AND-gate) or $\vee$ (OR-gate).

Each valuation ν of the *input gates* C_{inp} of C can inductively be extended to an *evaluation* $\nu' : C \to \{0,1\}$ as follows: $\nu'(g)$ is $\nu(g)$ if $g \in C_{\mathsf{inp}}$ (i.e., $\mu(g) = \mathsf{inp}$); it is $\neg\nu'(g')$ if $\mu(g) = \neg$ (with $(g', g) \in W$); otherwise it is $\bigodot_{(g',g)\in W} \nu'(g')$ where $\odot$ is $\mu(g)$ (hence, $\wedge$ or $\vee$). We often abuse notation and identify valuations and evaluations, and write $\nu(C)$ to mean $\nu(g_0)$. We use *0-gates* and *1-gates* as syntactic sugar for OR-gates and AND-gates with no inputs; they evaluate to 0 and 1 respectively.

The Boolean function *captured* by C is the one that maps any valuation ν of C_{inp} to $\nu(C)$. It is clear that any Boolean function can be captured by a Boolean circuit.

A *monotone* circuit is one that has no NOT-gate. It is clear that the Boolean function captured by a monotone circuit is a monotone Boolean function, and conversely any monotone Boolean function can be captured by such a circuit.

Boolean formulae. A *Boolean formula* on variables X is an expression built using the elements of X, the constants 0 and 1, the unary operator $\neg$, and the associative operators $\vee$ and $\wedge$. A formula is *monotone* if it has no $\neg$. We define in the standard way the Boolean function *expressed* by a Boolean formula. We can equivalently see formulae as circuits whose reverse DAG is a tree except that the input nodes may have multiple parents; unlike circuits, formulae cannot share common subexpressions.

2.3 Instances and Graphs

We work in a *signature* σ, i.e., a finite set of *relation names* (e.g., R) with associated *arity* $|R| \geqslant 1$. The *arity* of σ, written $|\sigma|$, is the maximum of $|R|$ over $R \in \sigma$; we call σ *arity-k* if $|\sigma| = k$.

Relational instances Fixing a countable domain $\mathcal{D} = \{a_k \mid k \geqslant 0\}$, a *relational instance* I over σ (or σ-instance) is a finite set of *ground facts* of the form $R(\mathbf{a})$ with $R \in \sigma$, where $\mathbf{a}$ is a tuple of $|R|$ elements of $\mathcal{D}$. We follow the *active domain semantics*, where the *domain* $\mathrm{dom}(I) \subseteq \mathcal{D}$ of I is the finite set of elements of $\mathcal{D}$ used in I. The *size* of I, denoted $|I|$, is its number of facts.

A *class* of instances, $\mathcal{I}$, is just a (potentially infinite) set of instances on the signature σ.

A *homomorphism* from a σ-instance I to a σ-instance I' is a function $h : \mathrm{dom}(I) \to \mathrm{dom}(I')$ such that for all $R(a_1, \ldots, a_k) \in I$ we have $R(h(a_1), \ldots, h(a_k)) \in I'$. A homomorphism is an *isomorphism* if it is bijective and its inverse is also a homomorphism.

We say that an instance I' is a *subinstance* of I, written $I' \subseteq I$, if I' is a subset of the facts of I, which implies $\mathrm{dom}(I') \subseteq \mathrm{dom}(I)$.

Graphs. Unless otherwise specified, a *graph* is undirected, simple, and unlabeled. Formally, we can equivalently see a graph G as an instance of the *graph signature* formed of a single predicate E of arity 2 and such that:

1. $\forall x\, E(x,x) \notin G$; and
2. $\forall xy\, E(x,y) \in G \Rightarrow E(y,x) \in G$.

As we follow the active domain semantics, this implies that we disallow *isolated vertices* in graphs. The facts of G are called *edges*. The set of *vertices* (or *nodes*) of a graph G, denoted $V(G)$, is its domain. Two vertices x and y of a graph G are *adjacent* if $E(x,y) \in G$, and x and y are then called the *endpoints* of the edge, and the edge is *incident* to them. Two edges are *incident* if they share a vertex.

The *degree* of a vertex x is the number of its adjacent vertices. For $k \in \mathbb{N}$, a graph is *k-regular* if all vertices have degree k. More generally, it is *K-regular*, where K is a finite set of integers, if every vertex has degree k for some $k \in K$. Finally, a graph is *degree-k* if k is the maximum of the degree of all its vertices, i.e., if it is $\{1, \ldots, k\}$-regular. A graph is *planar* if it can be drawn on the plane without edge crossings, in the standard sense [Diestel 2005].

A *path* of length $n \in \mathbb{N}_{>0}$ in a graph G is a set of edges $\{E(x_0,x_1), E(x_1,x_2), \ldots, E(x_{n-1},x_n)\}$ that are all in G; the path is *simple* if all x_i's are distinct. A *cycle* is a path of length $n \geqslant 3$ where all vertices are distinct except that $x_0 = x_n$; a graph is *cyclic* if it has a cycle. A graph is *connected* if there is a path from every vertex to every other vertex.

The *Gaifman graph* of an instance I is the graph on $\mathrm{dom}(I)$ where two elements are connected if they co-occur in some fact, so that each fact of I induces a clique in the Gaifman graph.

2.4 Tree Decompositions

A *tree decomposition* of an instance I is a $\mathcal{T}$-tree $T = (B, L, R, \mathrm{dom})$ where $\mathcal{T}$ is the set of subsets of $\mathrm{dom}(I)$. The nodes of T are called *bags* and their label is written $\mathrm{dom}(b)$. We require:

1. for every fact $R(\mathbf{a})$ of I, there exists a bag $b_{\mathbf{a}} \in B$ such that $\mathbf{a} \subseteq \mathrm{dom}(b_{\mathbf{a}})$;
2. for every $a \in \mathrm{dom}(I)$, letting $B_a := \{b \in B \mid a \in \mathrm{dom}(b)\}$, for every two bags $b_1, b_2 \in B_a$, all bags on the (unique) undirected path from b_1 to b_2 are also in B_a.

The *width* of T is $\mathrm{tw}(T) := \max_{b \in T} |\mathrm{dom}(b)| - 1$. The *treewidth* (or *width*) of an instance I, written $\mathrm{tw}(I)$, is the minimal width $\mathrm{tw}(T)$ of a tree decomposition T of I. For instance, the treewidth of a tree is 1, that of a cycle is 2, and that of a k-clique or $(k-1)$-grid is $k-1$. It is immediate that we have $\mathrm{tw}(I') \leqslant \mathrm{tw}(I)$ for any $I' \subseteq I$.

It is NP-hard, given an instance I, to determine $\mathrm{tw}(I)$. However, given a fixed width k, one can compute in linear time in I a tree decomposition of width $\leqslant k$ of an input instance I if one exists [Bodlaender 1996].

A *tree decomposition* of a graph $G = (V, E)$, is defined as for instances, where we replace $\mathrm{dom}(I)$ by the set of vertices V of G, and the requirement on every fact $R(\mathbf{a})$ of I is replaced by the analogous requirement on every *edge* $\{e, e'\} \in E$ of G. A *tree decomposition* of a circuit $C = (G, W, g_0, \mu)$ is a tree decomposition of (G, W), seen as an undirected graph.

A *path decomposition* is a tree decomposition which is also a path. Formally, following our requirement that tree decompositions are full binary trees, a path decomposition is a tree decomposition where the right child of any internal node is a leaf mapped by the labeling function dom to the empty set.

2.5 Queries

A *query* q is a logical formula in (function-free) first-order logic (FO) or second-order logic (SO) on σ, without free second-order variables; a σ-instance I can *satisfy* it ($I \models q$) or *violate* it ($I \not\models q$), with the standard definition. Remember that we follow the *active domain semantics*, so that quantifiers are taken to range over the *active domain* of instances. For simplicity, unless stated otherwise, we restrict to queries which are *Boolean*, that is, that have no free variables, and which are *constant-free*.

MSO and GSO. We will not work with the entire second-order logic, but with restrictions thereof:

- *Monadic second-order logic* (MSO), where second-order quantification is only allowed over sets.
- *Guarded second-order* (GSO), where second-order quantification is allowed on arbitrary-arity relations, but such quantification is semantically restricted to be only about *guarded tuples* (i.e., tuples that already co-occur in some instance fact); see [Grädel, Hirsch, and Otto 2002] for the formal definition.

CQ, UCQ, CQ$^{\neq}$, UCQ$^{\neq}$. As for FO, we will also be interested in specific fragments which are frequently used in database theory:

- The language CQ of *conjunctive queries*, which are existentially quantified conjunctions of atoms over σ;
- The language CQ$^{\neq}$ of conjunctive queries where additional atoms of the form $x \neq y$ (called *disequality* atoms) are allowed, where x and y are variables appearing in some regular atom;
- The language UCQ of *union of conjunctive queries*, i.e., disjunctions of CQs;
- The language UCQ$^{\neq}$ of disjunctions of CQ$^{\neq}$ queries.

We do not allow *equality atoms* in any of these languages (but of course we allow *disequality atoms* in CQ$^{\neq}$ and UCQ$^{\neq}$). The size $|q|$ of a UCQ$^{\neq}$ query q is its total number of atoms, i.e., the sum of the number of atoms in each CQ$^{\neq}$.

A *homomorphism* from a CQ q to an instance I is a mapping h from the variables of q to $\mathrm{dom}(I)$ such that for each atom $R(x_1, \ldots, x_k)$ of q we have $R(h(x_1), \ldots, h(x_k)) \in I$. For CQ$^{\neq}$ queries, we require that $h(x) \neq h(y)$ whenever q contains the disequality

atom $x \neq y$. A *homomorphism* from a UCQ$^{\neq}$ q to I is a homomorphism from some disjunct of q to I. A homomorphism is also called a *match* of q in I, as it witnesses that $I \models q$; by a slight abuse, we also call *match* of q the facts of I in the image of the homomorphism when no confusion may ensue.

Datalog. A *Datalog query* P over the signature σ consists of a signature σ_{int} of *intensional predicates* with a special 0-ary relation Goal, and a finite set of *rules* of the form $R(\mathbf{x}) \leftarrow R_1(\mathbf{y_1}), \ldots, R_k(\mathbf{y_k})$ where we have $R \in \sigma_{\text{int}}$ and $R_i \in \sigma \sqcup \sigma_{\text{int}}$ for $1 \leqslant i \leqslant k$ (where $\sqcup$ denotes disjoint union); further, we require that each variable in the tuple $\mathbf{x}$ also occurs in some tuple $\mathbf{y_i}$. The left-hand (resp., right-hand) side of a Datalog rule is called the *head* (resp., *body*) of the rule.

The query P is *guarded* [Grädel 2000; Gottlob, Grädel, and Veith 2002] if, for each rule $R(\mathbf{x}) \leftarrow R_1(\mathbf{y_1}), \ldots, R_k(\mathbf{y_k})$, there is some atom A in the body of the rule where all the variables $\mathbf{y_1}, \ldots, \mathbf{y_k}$ of the body of the rule appear. The query P is *frontier-guarded* [Baget, Leclère, and Mugnier 2010] if the same holds except that A is only required to contain all the variables $\mathbf{x}$ that appear in the head of the rule.

A *proof tree* T of a Datalog query P over an instance I is a non-binary rooted tree with nodes annotated by facts over $\sigma \cup \sigma_{\text{int}}$ on elements of $\text{dom}(I)$, with internal nodes further annotated by rules. We require that the fact of the root of T is Goal, the facts of the leaf nodes are all facts of I, and, for every internal node n in T with children $n_1, \ldots, n_m$, the indicated rule $R(\mathbf{x}) \leftarrow \Psi(\mathbf{y})$ on n in P is such that there is a homomorphism h mapping $R(\mathbf{x})$ to the fact of n and mapping $\Psi(\mathbf{y})$ to the facts of the n_i, each child n_i having exactly one preimage atom in $\Psi(\mathbf{y})$ for its fact. Note that this definition implies that internal nodes are necessarily annotated by a fact of σ_{int}. We write $I \models P$ if P has a proof tree on I.

Properties. A query is *monotone* if $I \models q$ and $I \subseteq I'$ imply $I' \models q$ for any two instances I, I'. A query is *closed under homomorphisms* if we have $I' \models q$ whenever $I \models q$ and I has a homomorphism to I', for any I and I'. UCQ is an example of query class that is both monotone and closed under homomorphisms, while UCQ$^{\neq}$ is monotone but not closed under homomorphisms.

2.6 Query Evaluation and Probabilities

Query evaluation problem. The *query evaluation problem* for a class $\mathcal{Q}$ of queries and $\mathcal{I}$ of instances is the problem, given a query $q \in \mathcal{Q}$ and an instance $I \in \mathcal{I}$, of deciding whether $I \models q$. Its *combined complexity* is the complexity as a function of I and q, and its *data complexity* is the complexity when q is fixed and only I is given as input.

Match counting problem. The *match counting problem* for an MSO formula with free second-order variables $q(\mathbf{X})$ on an instance family $\mathcal{I}$ is the problem, given an instance $I \in \mathcal{I}$, of counting how many vectors $\mathbf{A}$ of domain subsets are such that I satisfies $q(\mathbf{A})$.

The restriction to free second-order variables is without loss of generality: free first-order variables can be rewritten to free second-order ones which are asserted to be interpreted as singletons.

Probabilities. All probabilities, indeed all numbers which are not integers, are represented as rational numbers, given as the ratio of their integer numerator and integer denominator. We say that an algorithm runs in *ra-linear time* if it runs in linear time assuming that arithmetic operations over rational numbers take constant time and rationals are stored in constant space, and runs in polynomial time without this assumption.

A *probability distribution* is a pair $(\mathcal{U}, \Pr)$ of a finite *universe* $\mathcal{U}$ (whose elements are called *possible worlds*) and a *probability measure* $\Pr : \mathcal{U} \to [0,1]$ such that $\sum_{I \in \mathcal{U}} \Pr(I) = 1$.

An *uncertainty framework* is a language CL of objects J, called *uncertain instances*, and a semantics $[\![\cdot]\!]$ that maps any $J \in \mathsf{CL}$ to a universe $[\![J]\!]$. Likewise, a *probabilistic framework* gives to any $J \in \mathsf{CL}$, called a *probabilistic instance*, a semantics $[\![J]\!]$ which is a probability distribution. A *relational uncertainty framework* is an uncertainty framework whose semantics maps to universes of relational instances, and a *relational probabilistic framework* is defined analogously.

The most important relational probabilistic framework used in the sequel is that of *tuple-independent databases* (TID). Having fixed a signature σ, a probabilistic instance in this model (called a *TID instance*) is a pair $J = (I, \pi)$ of a σ-instance I, and a probability annotation function that maps each fact $F = R(\mathbf{a})$ of I to a probability $\pi(F) \in [0,1]$. The probability distribution $[\![J]\!]$ is the one obtained by seeing each fact as kept or discarded with the indicated probability, assuming independence between facts. Formally, we have $[\![J]\!] = (\mathcal{U}_J, \Pr_J)$, where $\mathcal{U}_J := \{I' \mid I' \subseteq I\}$, and $\Pr_J$ is defined as follows for $I' \in \mathcal{U}_J$:

$$\Pr_J(I') := \prod_{F \in I \cap I'} \pi(F) \prod_{F \in I \setminus I'} (1 - \pi(F))$$

Probability evaluation problem. Given a fixed query q in SO and a relational probabilistic framework, the *probability evaluation problem* is to determine, for an input probabilistic instance J, the probability that q holds in a possible world of J, namely, writing $(\mathcal{U}_J, \Pr_J) = [\![J]\!]$, we wish to compute:

$$\Pr_J(q) := \sum_{\substack{I \in \mathcal{U}_J \\ I \models q}} \Pr_J(I)$$

It is known that for many queries, e.g., the fixed CQ $\exists x\ R(x) \wedge S(x,y) \wedge T(y)$, the probability evaluation problem for TID instances is #P-hard [Dalvi and Suciu 2007]. More specifically, the following dichotomy result is known: in the TID framework, for any UCQ q, the probability evaluation problem for q is either in PTIME or it is #P-hard [Dalvi and Suciu 2012]. In these results, #P denotes the class of counting problems whose result can be computed as the number of accepting paths of some polynomial-time nondeterministic Turing machine. We will also work with the class $\mathrm{FP}^{\#\mathrm{P}}$ of computation problems (whose output is not necessarily an integer) which can be solved by a polynomial-time deterministic Turing machine with a #P-oracle.

When working in the framework of TID instances, we define, for any fixed query q and instance class $\mathcal{I}$, the *probability evaluation problem* for q and $\mathcal{I}$, as the probability evaluation problem for q where the input TID instances $J = (I, \pi)$ are restricted by requiring that $I \in \mathcal{I}$. Note that $\mathcal{I}$ imposes no constraint on π.

Chapter 3

Provenance for Treelike Instances

This chapter introduces our formalism for provenance constructions on trees and treelike instances. We will use this formalism, or refinements thereof, as our basis to obtain all upper bounds in the next two chapters.

We start with the case of trees in Section 3.1, and formalize the notion of a *provenance circuit* of a tree automaton A on a tree, namely, a circuit that captures how the query expressed by A depends on the labels of the input tree. The main result is Theorem 3.1.4, stating that such circuits can be efficiently constructed in the automaton and in the tree, and that they have bounded treewidth:

Theorem 3.1.4. *For any alphabet Γ, letting $\overline{\Gamma} := \Gamma \times \{0,1\}$, given a $\overline{\Gamma}$-bNTA A and a Γ-tree T, we can construct in time $O(|A| \cdot |T|)$ a provenance circuit C of A on T that has treewidth $O(|A|)$.*

We then move on to *treelike instances* (i.e., bounded-treewidth instances). First, in Section 3.2, we review the usual technique [Courcelle 1990; Flum, Frick, and Grohe 2002] of rewriting such instances to trees on a certain finite alphabet, and rewriting queries to tree automata, in such a way that evaluating the query on the instance amounts to evaluating the automaton on the tree. Then, in Section 3.3, we use this to extend Theorem 3.1.4 to treelike instances rather than trees, and show:

Theorem 3.3.2. *For any fixed $k \in \mathbb{N}$ and GSO query q, for any σ-instance I such that $\mathrm{tw}(I) \leqslant k$, we can construct a provenance circuit C of q on I in time $O(|I|)$. The treewidth of C only depends on k and q (not on I).*

Unlike Theorem 3.1.4, this result only claims tractability in *data complexity*, i.e., the query is fixed and only the treelike instance is given as input.

In Section 3.4, we specialize these results to the setting of *monotone* queries and *monotone* circuits, and show the analogue of Theorem 3.1.4 and Theorem 3.3.2, via a notion of *monotone* tree automata. This will relate to our study of the connections with *semiring provenance* in Chapter 5.

Theorem 3.4.2. *For any fixed $k \in \mathbb{N}$ and* monotone *GSO query q, for any σ-instance I such that $\mathrm{tw}(I) \leqslant k$, one can construct in time $O(|I|)$ a* monotone *provenance circuit of q on I whose treewidth only depends on k and q (not on I).*

In Section 3.5, we show how our construction of bounded-treewidth provenance circuits implies the existence of other standard representations of provenance (or *lineage*) which are often used in knowledge compilation [Jha and Suciu 2013], namely

OBDDs and *d-DNNFs*. Last, we justify in Section 3.6 our choice of representing the provenance as a *circuit* rather than as a Boolean formula, by showing queries for which provenance representations on treelike instances are more concise as circuits than as formulae.

3.1 Provenance Circuits for Trees

Let us thus start by studying a notion of provenance for tree automata, defined in an uncertain tree framework. Fixing a finite alphabet Γ throughout this section, we view a Γ-tree T as an *uncertain tree*, where each node carries an unknown Boolean annotation in $\{0, 1\}$. We then consider all possible *valuations* that choose an annotation for each node of T, calling $\overline{\Gamma}$ the alphabet of annotated trees:

Definition 3.1.1. We write $\overline{\Gamma} := \Gamma \times \{0, 1\}$. For any Γ-tree $T = (V, L, R, \lambda)$ and valuation $\nu : V \to \{0, 1\}$, we define $\nu(T)$ to be the $\overline{\Gamma}$-tree with same skeleton as T, where each node n is given the label $(\lambda(n), \nu(n))$. $\lhd$

We consider automata on *annotated* trees, namely, $\overline{\Gamma}$-bNTAs, and define their *provenance* on a Γ-tree T as a Boolean function that describes which valuations of T are accepted by the automaton. Intuitively, provenance keeps track of the dependence between Boolean annotations and acceptance or rejection of the tree.

Definition 3.1.2. The *provenance* of a $\overline{\Gamma}$-bNTA A on a Γ-tree $T = (V, L, R, \lambda)$ is the Boolean function $\mathrm{Prov}(A, T)$ mapping any valuation $\nu : V \to \{0, 1\}$ to 1 or 0 depending on whether $\nu(T) \models A$ or not. $\lhd$

We now introduce the notion of a *provenance circuit* of A on a Γ-tree T, which is a Boolean circuit that captures the provenance of A on T, i.e., $\mathrm{Prov}(A, T)$. Formally:

Definition 3.1.3. Let A be a $\overline{\Gamma}$-bNTA and $T = (V, L, R, \lambda)$ be a Γ-tree. A *provenance circuit* of A on T is a Boolean circuit C with $C_{\mathsf{inp}} = V$ that captures the Boolean function $\mathrm{Prov}(A, T)$. $\lhd$

This section shows that we can construct provenance circuits for Γ-bNTAs on Γ-trees in linear time, and that their *treewidth* only depends on the automaton, not on the tree:

Theorem 3.1.4. *For any alphabet Γ, letting $\overline{\Gamma} := \Gamma \times \{0, 1\}$, given a $\overline{\Gamma}$-bNTA A and a Γ-tree T, we can construct in time $O(|A| \cdot |T|)$ a provenance circuit C of A on T that has treewidth $O(|A|)$.*

The intuition is that we create one gate in C per state of A per node of T, and we write out in C all possible transitions of A at each node n of T, depending on the input gate that indicates the annotation of n. The rest of this section gives a formal proof of this result.

Proof of Theorem 3.1.4. Fix the Γ-tree $T = (V, L, R, \lambda)$, the $\overline{\Gamma}$-bNTA $A = (Q, F, \iota, \delta)$, and construct the provenance circuit $C = (G, W, g_0, \mu)$. For each node n of T, create the following gates:

- one input gate g_n^{i} in C (which we identify to n, so that we have $C_{\mathsf{inp}} = V$);
- one NOT-gate $g_n^{\neg\mathrm{i}}$ which is a NOT-gate of g_n^{i};
- one gate g_n^q for every $q \in Q$, which we will explain how to define.

If n is a leaf node, for $q \in Q$, set g_n^q to be an OR-gate of:

- the gate $g_n^{\neg\mathrm{i}}$ if $q \in \iota(\lambda(n), 0)$, and a 0-gate otherwise;
- the gate g_n^{i} if $q \in \iota(\lambda(n), 1)$, and a 0-gate otherwise;

If n is an internal node, for every pair $q_{\mathrm{L}}, q_{\mathrm{R}} \in Q$ (that appears as input states of a transition of δ), create the following three gates:

- $g_n^{q_{\mathrm{L}},q_{\mathrm{R}}}$ which is an AND-gate of $g_{L(n)}^{q_{\mathrm{L}}}$ and $g_{R(n)}^{q_{\mathrm{R}}}$;
- $g_n^{q_{\mathrm{L}},q_{\mathrm{R}},\mathrm{i}}$ which is an AND-gate of $g_n^{q_{\mathrm{L}},q_{\mathrm{R}}}$ and of g_n^{i};
- $g_n^{q_{\mathrm{L}},q_{\mathrm{R}},\neg\mathrm{i}}$ which is an AND-gate of $g_n^{q_{\mathrm{L}},q_{\mathrm{R}}}$ and of $g_n^{\neg\mathrm{i}}$.

Now, still assuming that n is an internal node, for $q \in Q$, set g_n^q to be an OR-gate of:

- all the $g_n^{q_{\mathrm{L}},q_{\mathrm{R}},\neg\mathrm{i}}$ such that $q \in \delta(q_{\mathrm{L}}, q_{\mathrm{R}}, (\lambda(n), 0))$;
- all the $g_n^{q_{\mathrm{L}},q_{\mathrm{R}},\mathrm{i}}$ such that $q \in \delta(q_{\mathrm{L}}, q_{\mathrm{R}}, (\lambda(n), 1))$.

Last, add gate g_0 which is an OR-gate of all the g_r^q such that $q \in F$, where r is the root node of T.

This construction is in time $O(|A| \cdot |T|)$: more precisely, for every node of the tree T, we create a number of gates that is linear in the number of states in Q and in the number of transitions of δ.

Now we show that C is indeed a provenance circuit of A on T. Let $\nu : V \to \{0, 1\}$ be a valuation that we extend to an evaluation of C. We show by induction on $n \in T$ that for any $q \in Q$, we have $\nu(g_n^q) = 1$ iff, letting T_n be the subtree of T rooted at n, there is a run ρ of A on T_n such that $\rho(n) = q$.

For a leaf node n, choosing $q \in Q$, for any $b \in \{0, 1\}$, if $\nu(n) = b$ then $\nu(g_n^q) = 1$ iff $q \in \iota(\lambda(n), b)$, so we can define a run ρ as $\rho(n) := q$. Conversely, the existence of a run clearly ensures that $\nu(g_n^q) = 1$.

For an internal node n, choosing $q \in Q$, for any $b \in \{0, 1\}$, if $\nu(n) = b$ then, $\nu(g_n^q) = 1$ iff there are some $q_{\mathrm{L}}, q_{\mathrm{R}} \in Q$ such that $q \in \delta(q_{\mathrm{L}}, q_{\mathrm{R}}, (\lambda(n), b))$ and $\nu(g_{\mathrm{L}(n)}^{q_{\mathrm{L}}}) = \nu(g_{\mathrm{R}(n)}^{q_{\mathrm{R}}}) = 1$. By induction hypothesis, this implies the existence of a run ρ_{L} of A on $T_{L(n)}$ such that $\rho_{\mathrm{L}}(\mathrm{L}(n)) = q_{\mathrm{L}}$ and a run ρ_{R} of A on $T_{R(n)}$ such that $\rho_{\mathrm{R}}(\mathrm{R}(n)) = q_{\mathrm{R}}$, from which we construct a run ρ of A on T_n such that $\rho(n) = q$, by setting $\rho(n) := q$ and setting $\rho(n')$ either to $\rho_{\mathrm{L}}(n')$ or to $\rho_{\mathrm{R}}(n')$ depending on whether $n' \in T_{L(n)}$ or $n' \in T_{R(n)}$. Conversely, the existence of such a run ρ implies the existence of two such runs ρ_{L} and ρ_{R}, from which we deduce that $\nu(g_{\mathrm{L}(n)}^{q_{\mathrm{L}}}) = \nu(g_{\mathrm{R}(n)}^{q_{\mathrm{R}}}) = 1$, and we conclude that $\nu(g_n^q) = 1$ following the transition of A used in the run ρ at n.

The claim proven by induction clearly justifies that C is a provenance circuit, as, applying it to the root of T, we deduce from the definition of g_0 that, for any valuation ν, we have $\nu(C) = 1$ iff there is an accepting run of A on $\nu(T)$.

We last show that C has a tree decomposition of width $3 \cdot |\delta| + 3 \cdot |Q| + 2$. Consider a tree T' that has same skeleton as T, and define a tree decomposition $(T', \mathrm{L}, \mathrm{R}, \mathrm{dom})$ as follows: for each bag $b \in T'$, letting n be the corresponding node in T, we set $\mathrm{dom}(b)$ to contain:

- g_n^{i}, $g_n^{\neg \mathrm{i}}$, and g_n^q for $q \in Q$;
- If n is an internal node, $g_{L(n)}^{q_\mathrm{L}}$, $g_{R(n)}^{q_\mathrm{R}}$, $g_n^{q_\mathrm{L}, q_\mathrm{R}}$, $g_n^{q_\mathrm{L}, q_\mathrm{R}, \mathrm{i}}$, and $g_n^{q_\mathrm{L}, q_\mathrm{R}, \neg \mathrm{i}}$, for $q_\mathrm{L}, q_\mathrm{R} \in Q$;
- If n is the root node, the output gate g_0.

This clearly ensures that $|\mathrm{dom}(b)|$ is at most $2 + |Q| + 2 \cdot |Q| + 3 \cdot |\delta| + 1$, so the only thing left to show is that we have indeed defined a tree decomposition of C.

For the first condition, we must show that whenever $C = (G, W, g_0, \mu)$ contains an edge $(g_1, g_2) \in W$, then there is a bag b of T' such that $g_1, g_2 \in \mathrm{dom}(b)$. If g_2 is the output gate g_0, then, by construction of C, g_1 must be a gate of the form g_r^q where r is the root of T, so the root bag of T' witnesses that the condition is satisfied. Otherwise, g_2 is of the form $g_n^\bullet$ for some $n \in T$. Then, by construction of C, the gate g_1 must be either of the form $g_n^\bullet$, or of the form $g_{L(n)}^{q_\mathrm{L}}$ for some $q_\mathrm{L} \in Q$, or of the form $g_{R(n)}^{q_\mathrm{R}}$ for some $q_\mathrm{R} \in Q$. In either case, we can pick the bag b corresponding to n in T to conclude.

For the second condition, we must show that for any gate $g \in G$, the subset T_g of bags of T' such that $g \in \mathrm{dom}(b)$ form a connected subtree of T'. For $g = g_0$, the claim is immediate as g_0 only occurs in the root bag of T'. Otherwise, for g of the form $g_n^\bullet$ for some $n \in T$, letting b be the corresponding bag in T', it is clear by construction of T' that g only occurs in $\mathrm{dom}(b)$ and possibly in $\mathrm{dom}(L(b))$ and $\mathrm{dom}(R(b))$, so T_g is indeed a connected subtree.

Hence, we have shown that T' is a tree decomposition of C of width $O(|A|)$, so C has treewidth $O(|A|)$. This concludes the proof. □

3.2 Rewriting Queries on Treelike Instances to Tree Automata

Our goal is now to extend Theorem 3.1.4 from trees to treelike instances. We do this using the standard connection between query evaluation on bounded-treewidth instances, and tree automaton evaluation on trees. The original result is by [Courcelle 1990], with an extension to relational instances (rather than graphs) given in [Flum, Frick, and Grohe 2002].

We first introduce an abstraction of the mechanism that we use to encode queries on treelike instances to automata on trees, called a *tree interpretation scheme*, and claim the existence of such a scheme. The construction is standard but crucially ensures a property, called *subinstance-compatibility*, which is what we rely on to adapt the results of the previous section. This property is not hard to ensure, but implies that we cannot directly use existing schemes such as the one of [Flum, Frick, and Grohe 2002], and we must give the details of a suitable scheme.

We then give our formal construction of the tree interpretation scheme, and prove its correctness. The proof of the correctness of the scheme is inspired by the proof in [Flum, Frick, and Grohe 2002], but our scheme is different and inspired by an idea of [Chaudhuri and Vardi 1992]. We give these details to show that our property of

subinstance-compatibility is respected. However, it is possible to skip these proofs on first reading, because the subinstance-compatible tree interpretation scheme thus defined will be used as a black box until Chapter 5.

Definition 3.2.1. Let $k \in \mathbb{N}$ and let σ be a relational signature. Write $\mathcal{I}^\sigma_{\leqslant k}$ for the class of σ-instances of treewidth $\leqslant k$. A *tree interpretation scheme* for k and σ and for a class $\mathcal{Q}$ of (Boolean constant-free) queries consists of:

- A *finite* alphabet Γ^σ_k, computable from k and σ;
- A computable *translation function* $\mathcal{A}$ that maps each query q in $\mathcal{Q}$ to a Γ^σ_k-bNTA $\mathcal{A}(q)$;
- An *encoding function* $\mathcal{E}$ that maps *in linear time* each instance I of $\mathcal{I}^\sigma_{\leqslant k}$ to a Γ^σ_k-tree $\mathcal{E}(I)$;
- A *decoding function* $\langle\cdot\rangle$ that maps each Γ^σ_k-tree E to an instance $\langle E\rangle$ in $\mathcal{I}^\sigma_{\leqslant k}$.

We require the following properties:

- For any $I \in \mathcal{I}^\sigma_{\leqslant k}$, the instance $\langle\mathcal{E}(I)\rangle$ is isomorphic to I.
- For any $q \in \mathcal{Q}$, the bNTA $\mathcal{A}(q)$ *tests* q, namely: for any Γ^σ_k-tree E, we have $E \models \mathcal{A}(q)$ iff $\langle E\rangle \models q$. ◁

The intuition of the scheme that we will define is that the alphabet Γ^σ_k describes all possible facts, and the encoding function proceeds by constructing in linear time a tree decomposition of the instance (for the fixed treewidth k), then converting it to an Γ^σ_k-tree where each fact of I is coded in some node.

It is a standard result that, for any $k \in \mathbb{N}$ and signature σ, there is a tree interpretation scheme for MSO queries, although the translation function may have non-elementary complexity [Meyer 1975]. The result can in fact be extended from MSO to guarded second-order (GSO), which captures more database languages, e.g., guarded Datalog; indeed, GSO collapses to MSO on treelike instances [Grädel, Hirsch, and Otto 2002].

The existence of a tree interpretation scheme clearly implies that the data complexity of GSO query evaluation on bounded treewidth instances is linear-time: we can translate the query (in a data-independent way) to a tree automaton A, encode the instance in linear time to an encoding E, and evaluate A on E in linear time in E.

However, our construction in the next section will rely on an additional property of the scheme, of which we give an abstract definition:

Definition 3.2.2. Using the notations of Definition 3.2.1, a tree interpretation scheme is *subinstance-compatible* if there is:

- a *neutering* mapping $\cdot \mapsto \underline{\cdot}$ from Γ^σ_k to itself;
- for any instance $I \in \mathcal{I}^\sigma_{\leqslant k}$, an injective mapping φ_I from I to the nodes of the tree $\mathcal{E}(I)$ which can be computed in linear time.

We require the following additional conditions:

- For any Γ_k^σ-tree E and Γ_k^σ-tree E' with same skeleton, assume that, for any node n of E and corresponding n' of E', we have $\lambda(n') = \lambda(n)$ or $\lambda(n') = \underline{\lambda(n)}$. We then require for any such E and E' that $\langle E' \rangle \subseteq \langle E \rangle$.

- Specifically, for any subinstance $I' \subseteq I$, let $E'(I')$ be the Γ_k^σ-tree with same skeleton as $\mathcal{E}(I)$, where each node n' corresponding to n in $\mathcal{E}(I)$ has label $\lambda(n)$ if $n \in \varphi_I(I')$ and $\underline{\lambda(n)}$ otherwise. We then require, for any $I' \subseteq I$, that $\langle E'(I') \rangle$ is isomorphic to I'. ◁

Intuitively, a tree interpretation scheme is subinstance-compatible if each fact of the instance is encoded in one node of the encoding (pointed to by φ_I), and if we can obtain a valid encoding of any subinstance by relabeling the nodes of the missing facts according to the fixed *neutering* mapping, and more generally if any relabeling of this kind decodes to a subinstance of the original decoding.

So the result that we will be using in the sequel is:

Theorem 3.2.3. *For any $k \in \mathbb{N}$ and relational signature σ, there is a subinstance-compatible tree interpretation scheme for GSO queries.*

We will always use the notations Γ_k^σ, $\mathcal{A}$, $\mathcal{E}$, $\langle\cdot\rangle$, $\underline{\cdot}$, and φ_I, from Definitions 3.2.1 and 3.2.2, to refer to the tree interpretation scheme whose existence is claimed by the theorem.

We prove the theorem in the rest of this section, following the standard ideas of [Courcelle 1990; Flum, Frick, and Grohe 2002], though our actual construction of the alphabet follows [Chaudhuri and Vardi 1992]. We start in Section 3.2.1 by defining the alphabet, the encoding function $\mathcal{E}$, and the decoding function $\langle\cdot\rangle$. We then define the injective mapping φ_I and the neutering mapping to show that the scheme is subinstance-compatible. We then construct the translation function $\mathcal{A}$ in Section 3.2.2.

3.2.1 Encoding Treelike Instances and Decoding Trees

Let us start by defining the alphabet that we will use to encode treelike instances.

Alphabet. Our finite alphabet Γ_k^σ is intuitively the set of possible facts on an instance of domain size fixed to $2k + 2$. The point is that we will use element co-occurrence between a child and parent node in tree encodings to encode identities between elements, following the definition of proof trees in [Chaudhuri and Vardi 1992].

Formally, we take Γ_k^σ to be the set defined as follows:

Definition 3.2.4. The set of *k-facts* of the signature σ, written Γ_k^σ, is the set of pairs $\tau = (d, s)$ where:

- the *domain* d is a subset of size at most $k + 1$ of the first $2k + 2$ elements of the countable domain $\mathcal{D}$, written $a_1, \ldots, a_{2k+2}$;

- the *structure* s is a zero- or single-fact structure over σ for which we require $\mathrm{dom}(s) \subseteq d$. ◁

It is clear that the alphabet Γ_k^σ is computable from σ and k. In fact, for n_σ the number of relations of σ and $|\sigma|$ the arity of σ, we have:

$$|\Gamma_k^\sigma| = \sum_{i=0}^{k+1} \binom{2k+2}{i} \left(1 + \sum_{R \in \sigma} i^{|R|}\right) = O\left(2^{2k} n_\sigma k^{|\sigma|}\right)$$

We can now define the neutering mapping $\cdot \mapsto \underline{\cdot}$ on the alphabet Γ_k^σ, which we will use to show subinstance-compatibility:

Definition 3.2.5. For any k-fact $\tau = (d, s) \in \Gamma_k^\sigma$, we define the *neutered k-fact* $\underline{\tau}$ as $(d, \emptyset)$. In particular, if $s = \emptyset$ then $\underline{\tau} = \tau$. ◁

Decoding. We call *tree encoding* (of width k) a Γ_k^σ-tree. We first explain how a tree encoding E can be *decoded* to a structure $I = \langle E \rangle$ (defined up to isomorphism), and to a tree decomposition T of width $\leqslant k$ of I which has same skeleton as E, justifying that I has indeed treewidth k.

Process E top-down. At each non-root node n of E with label $\lambda(n) = (d, s)$, whose parent node n' has label $\lambda(n') = (d', s')$, pick fresh elements in $\mathcal{D}$ for the elements of $d \backslash d'$. Now, if s contains a fact F, let F' be F where each element in $d \backslash d'$ is replaced by the corresponding fresh element of $\mathcal{D}$, and each element a in $d \cap d'$ is replaced by the element of $\mathcal{D}$ used for a when processing n'. Add the fact F' to I, and add a bag to the tree decomposition T containing the elements of I matching those in d. At the root node, do the same process but picking fresh elements in $\mathcal{D}$ for all elements of d rather than considering the parent node.

Note that we may attempt to create the same fact multiple times when we decode: creating any copy beyond the first copy of the fact has no effect.

The result of this process is an instance I along with a tree T with same skeleton as E. Let us check that T is a tree decomposition of I. Indeed:

1. For each fact F of I, all its elements occur in the bag which we created in T at the same time that we created F in I.

2. As E and T have same skeleton, observe that the occurrences of any element $a \in \mathrm{dom}(I)$ in T clearly match the connected subtree of *contiguous* occurrences in E of the element of $\{a_1, \ldots, a_{2k+2}\}$ for which it was created.

This shows that T is indeed a tree decomposition of I, and it is immediate that T has width at most k. This concludes the definition of the decoding operator.

Encoding and subinstance-compatibility. We now justify that one can compute in linear time a tree encoding $\mathcal{E}(I)$ of width k of any instance I of treewidth $\leqslant k$. First, as k is constant, we can compute in linear time [Bodlaender 1996] in I a tree decomposition T of I of width $\leqslant k$.

We now show that we can compute in linear time from T the tree encoding $\mathcal{E}(I)$ of the instance I, defining the operator $\mathcal{E}$ (this result is implicit in [Chaudhuri and Vardi 1992]). This is also how we compute the injective mapping φ_I required to show subinstance-compatibility.

Lemma 3.2.6. *From a tree decomposition T of width k of a σ-structure I, one can compute in linear time in $|I| + |T|$:*

- *a tree encoding $E := \mathcal{E}(I)$ of width k of I (which is a Γ_k^σ-tree) such that $\langle E \rangle$ is isomorphic to I;*
- *an injective mapping φ_I from I to E such that the neutering function $\cdot \mapsto \underline{\cdot}$ and the mapping φ_I satisfy the conditions of subinstance-compatibility.*

Proof. Fix the σ-structure I and its tree decomposition T of width k. We start by precomputing a mapping that indicates, for every tuple $\mathbf{a}$ of I such that some fact $R(\mathbf{a})$ holds in I, the topmost bag $\mathsf{node}(\mathbf{a})$ of T such that $\mathbf{a} \subseteq \mathrm{dom}(\mathsf{node}(\mathbf{a}))$. This can be performed in linear time by Lemma 3.1 of [Flum, Frick, and Grohe 2002]. Then, we label the tree decomposition T with the facts of I as follows: for each fact $F = R(\mathbf{a})$ of I, we add F to the label of $\mathsf{node}(\mathbf{a})$.

Informally, we now build E by walking through the decomposition T in a top-down way, and encoding each node n of T as a chain of nodes in E, one for each fact assigned to n, and define φ_I to map each fact to the node created in E for that fact. We pick the labels of nodes in Γ_k^σ so that the elements shared between a bag and its parent in T are retained, and the new elements are chosen so as not to overlap with the parent node.

Formally, at any state of the top-down processing of the tree decomposition T, we process a bag b of T; if it is not the root node, then we write b' for its parent. We write n' for the node in E under which we are encoding b and the descendants of b, and write the label $\lambda(n')$ as (d', s'); at the root, n' is undefined. When b is not the root node, we denote by $f_{b'}$ a bijection from $\mathrm{dom}(b')$ to d' that we will have defined inductively.

To encode the bag b of T under the node n' of E, if b is not the root bag, partition $\mathrm{dom}(b) = d_\mathrm{o} \sqcup d_\mathrm{n}$ where $d_\mathrm{o} := \mathrm{dom}(b) \cap \mathrm{dom}(b')$ are the *old elements* already present in $\mathrm{dom}(b')$, and $d_\mathrm{n} := \mathrm{dom}(b') \setminus \mathrm{dom}(b)$ are the *new elements* that did not appear in $\mathrm{dom}(b')$. If b is the root bag, then we set $d_\mathrm{o} := \emptyset$ and $d_\mathrm{n} := \mathrm{dom}(b)$. Now, if b is not the root bag, letting (d', s') be the label of the node n' in E under which we are encoding, consider the bijection $f_{b'}$ from $\mathrm{dom}(b')$ to d' that we have defined inductively, restrict its domain to $\mathrm{dom}(b) \cap \mathrm{dom}(b')$, and extend it to an injective function f_b from $\mathrm{dom}(b)$ to $\{a_1, \ldots, a_{2k+2}\}$ such that the newly defined $f_b(d_\mathrm{n})$ is disjoint from $f_{b'}(\mathrm{dom}(b'))$; this is possible, as there are $2k+2$ elements to choose from and $|\mathrm{dom}(b')|$ and $|\mathrm{dom}(b)|$ are $\leqslant k+1$. At the root bag b, simply choose an arbitrary injective function f_b from $d_\mathrm{n} = \mathrm{dom}(b)$ to $\{a_1, \ldots, a_{k+2}\}$. We set $d := f_b(\mathrm{dom}(b))$.

We encode b as a chain of nodes in E under n' (or, at the root bag, as a chain whose first node is the root of E), and set the label of each node to be (d, s_i) where each s_i encodes one of the facts in the label of b. Define the mapping φ_I to map each fact F in the label of b to the node $\varphi_I(F)$ in the chain that we create for F: this is well-defined because each fact of I is assigned to exactly one bag in T, and it is clearly injective because each node contains a single fact. If there are zero facts in the label of b, create a $(d, \emptyset)$ zero-fact node instead, rather than creating no node.

Now, recursively encode the children of b (if any) in T, under the lowest node n of this chain of nodes in E, using the bijection f_b when encoding them. Note that f_b is indeed a bijection from $\mathrm{dom}(b)$ to the first component d of the label of n, as required for the induction.

We obtain the final result of the process by completing with zero-fact $(\emptyset, \emptyset)$ child nodes so that each non-leaf node has exactly two children, and E is a full binary tree. We assume that all arbitrary choices are done in a consistent manner so that

the process is deterministic and defines $\mathcal{E}(I)$. As the process traverses T and the number of steps for each bag $b \in T$ is proportional in the number of facts assigned to $b \in T$, and each fact of I is assigned to precisely one $b \in T$, the overall running time is linear in $|I| + |T|$.

We first verify that $\langle E \rangle$ is isomorphic to I. A straightforward induction shows that, at any bag b of T, letting n be the highest node in E created for b, the result of decoding the subtree of E rooted at n is isomorphic to the subinstance of I consisting of the facts of I whose label assigned following the node mapping is in the subtree rooted at b in T.

We now show that subinstance-compatibility is respected. We first show the second condition. Observe that, for any subinstance $I' \subseteq I$, let E' be the result of modifying the encoding $E := \mathcal{E}(I)$ to neuter everything but the facts that are present in I'. As E and E' have same skeleton and the first components of the labels in E are the same as in E', it is clear that the decoding process will proceed in E' exactly as in E, except that it will only create the facts that are in I'. Hence, the tree interpretation scheme that we defined is indeed subinstance-compatible.

The first condition is shown in the same manner: neutering some nodes, i.e., removing the fact in this nodes, just implies that these facts are not created, but the same facts are created for the other nodes. □

This concludes our presentation of the alphabet Γ_k^σ of our tree interpretation scheme, its encoding and decoding functions $\mathcal{E}$ and $\langle\cdot\rangle$, and its functions $\cdot \mapsto \dot{\cdot}$ and φ_I for subinstance-compatibility.

The point of encoding treelike instances to Γ_k^σ-trees in this way is that GSO queries on such instances can be translated to Γ_k^σ-bNTA on the encodings. The next section explains how this translation operator is defined.

3.2.2 Translating Queries to Automata

The only thing left to do is to define the translation operator $\mathcal{A}$, which we do in a way inspired by [Flum, Frick, and Grohe 2002]. We start by defining $\mathcal{A}$ over Boolean MSO queries. We will show at the end of the section how to extend the construction from MSO to GSO following [Grädel, Hirsch, and Otto 2002].

We first take care of a subtlety regarding domain semantics. Remember that we follow the *active domain semantics*. To avoid any problem with semantics when performing the translation, we first rewrite the MSO query q to *relativize* all quantifiers. Specifically, we define a unary predicate $\mathrm{Occ}(x) := \bigvee_{R \in \sigma, 1 \leqslant i \leqslant |R|} \mathrm{Occ}^{R,i}(x)$, where $\mathrm{Occ}^{R,i}(x) := \exists \mathbf{y}\ R(y_1, \ldots, y_{i-1}, x, y_{i+1}, \ldots, y_n)$. We then replace any first-order universal quantification $\forall x\ \varphi(\mathbf{X}, x, \mathbf{y})$ in q by $\forall x\ \mathrm{Occ}(x) \Rightarrow \varphi(\mathbf{X}, x, \mathbf{y})$, and likewise replace any first-order existential quantification $\exists x\ \varphi(\mathbf{X}, x, \mathbf{y})$ by $\exists x\ \mathrm{Occ}(x) \wedge \varphi(\mathbf{X}, x, \mathbf{y})$. For second-order quantification, we do the same but using the $\mathrm{Occ2}(X)$ unary predicate instead, which is defined by $\mathrm{Occ2}(X) := \forall x\ (x \in X \Rightarrow \mathrm{Occ}(x))$. Letting q' be the result of this relativization process, it is then clear that, for any instance I, under our active domain semantics, we have $I \models q$ iff $I \models q'$; and further q' is *domain-independent*, so its interpretation under the *natural* semantics (where the domain may differ from the active domain) will always match its interpretation under the active domain semantics. See [Abiteboul, Hull, and Vianu 1995] for details.

Hence, we assume that we are working with a domain-independent Boolean MSO query q. We proceed by the standard technique [Flum, Frick, and Grohe 2002] of translating q to a Boolean query ψ in the language of *MSO queries over* Γ_k^σ*-trees*, whose signature features:

- binary predicates for the left child and right child relations;
- unary predicates that test the label of nodes.

Once this is done, we can translate ψ to a Γ_k^σ-bNTA using [Thatcher and Wright 1968].

To do this, we must introduce some terminology. Let us consider a Γ_k^σ-tree $E = (V, L, R, \lambda)$. We say that a node $n \in V$ *contains* a_i if, writing its label $\lambda(n)$ as (d, s), we have $a_i \in d$. For $1 \leqslant i \leqslant 2k+2$, we say that $n, n' \in V$ are a_i*-connected* if every node n'' on the undirected path in E from n to n' (including n and n') contains a_i. We say that $n \in V$ is an a_i*-intro-node* if it contains a_i and either it is the root of E or its parent in E does not contain a_i. We further say that it is a *proper* a_i-intro-node if it is additionally a_i-connected to some node n' (possibly itself) whose label $(d', s') = \lambda(n')$ is such that $s' \neq \emptyset$ and a_i occurs in s'.

We now introduce vocabulary to establish a correspondence between elements and sets of elements on instances, and sets of nodes of a certain kind in tree encodings; this is again inspired by [Flum, Frick, and Grohe 2002]. We write $\tilde{V}$ for the set of $(2k+2)$-tuples of subsets of V, writing $\tilde{N} \in \tilde{V}$ as $\tilde{N} = N^1, \ldots, N^{2k+2}$; we will also write $\tilde{X} = X^1, \ldots, X^{2k+2}$ for a $(2k+2)$-tuple of second-order variables. We call *pseudo-element* an element $\tilde{N} \in \tilde{V}$ such that exactly one of the N^i is not empty, that N^i is a singleton, and the one node n that it contains is a proper a_i-intro-node. We write $\tilde{V}_\mathrm{e}$ for the subset of $\tilde{V}$ consisting of the pseudo-elements (note that $\tilde{V}_\mathrm{e}$ depends on E, even though this is not reflected in the notation). The intuition is that each pseudo-element $\tilde{N}$ of $\tilde{V}_\mathrm{e}$ points to a node of E, namely the one node n that it contains; and to a specific element in n, namely, the element a_i such that n was put in N^i. By our requirement that n is a_i-connected to a node where a_i occurs in a fact, we ensure that a_i will indeed yield an actual element when decoding; and we choose to pick the *root node* n for this copy of a_i to avoid any ambiguity.

We call a *pseudo-set* an element $\tilde{N} \in \tilde{V}$ such that, for all $1 \leqslant i \leqslant 2k+2$, for all $n \in N^i$, considering $\tilde{N}_{i,n} \in \tilde{V}$ defined by $N^i_{i,n} = \{n\}$ and $N^j_{i,n} = \emptyset$ for $j \neq i$, then $\tilde{N}_{i,n}$ is a pseudo-element. We write $\tilde{V}_\mathrm{s}$ for the subset of $\tilde{V}$ consisting of the pseudo-sets. Note that $\tilde{V}_\mathrm{e} \subseteq \tilde{V}_\mathrm{s}$ as a pseudo-element is identical to the singleton pseudo-set containing exactly that element.

We continue with our definitions on E. Let $I := \langle E \rangle$. We define a mapping η from $\mathrm{dom}(I)$ to $\tilde{V}$ as follows. For any $c \in \mathrm{dom}(I)$, consider, among the facts of I that contain c, the first one F that was created while decoding E. Let n be the node in E that was being decoded and a_i be the element in $\lambda(n)$ that was mapped to c. Consider the connected rooted subtree of a_i-connected nodes in E that contains n, and let n' be its root (possibly $n = n'$). We set $\eta(c) := \tilde{N}$ where $N^i = \{n'\}$ and $N^j = \emptyset$ for $j \neq i$. It is easily seen that $\eta(c)$ is then a pseudo-element, and that the mapping η is actually a bijection from $\mathrm{dom}(I)$ to $\tilde{V}_\mathrm{e}$. Let us extend η to a mapping θ from the powerset of $\mathrm{dom}(I)$ to $\tilde{V}$ by setting $\theta(A)$ for $A \subseteq \mathrm{dom}(I)$ to be $\tilde{N}$ defined by $N^i := \bigcup_{c \in A} (\eta(c))^i$; it is easily seen that this is actually a disjoint union and that θ is a bijection from the powerset of $\mathrm{dom}(I)$ to $\tilde{V}_\mathrm{s}$.

We will now show that for for any *non-Boolean* MSO formula $\varphi(X_1, \ldots, X_n, x_1, \ldots, x_m)$ with free first-order and second-order variables, there exists a non-Boolean MSO formula on Γ_k^σ-trees $\psi(\tilde{Y}_1, \ldots, \tilde{Y}_n, \tilde{Z}_1, \ldots, \tilde{Z}_m)$ with free second-order variables such that for any Γ_k^σ-tree $E = (V, L, R, \lambda)$, writing $I := \langle E \rangle$, defining the bijections η and θ as above, we have:

Forward: For any $A_1, \ldots, A_n \subseteq \mathrm{dom}(I)$ and $c_1, \ldots, c_m \in \mathrm{dom}(I)$, if we have $I \models \varphi(A_1, \ldots, A_n, c_1, \ldots, c_m)$, then $E \models \psi(\theta(A_1), \ldots, \theta(A_n), \eta(c_1), \ldots, \eta(c_m))$.

Backward: For any $V_1, \ldots, V_n, W_1, \ldots, W_n \in \tilde{V}$, if $E \models \psi(V_1, \ldots, V_n, W_1, \ldots, W_n)$, then $V_i \in \tilde{V}_\mathrm{s}$ for all $1 \leqslant i \leqslant n$ and $W_j \in \tilde{V}_\mathrm{e}$ for all $1 \leqslant j \leqslant m$, and, letting $A_i := \theta^{-1}(V_i)$ for all $1 \leqslant i \leqslant n$ and $c_j := \eta^{-1}(W_j)$ for all $1 \leqslant j \leqslant m$, we have $I \models \varphi(A_1, \ldots, A_n, c_1, \ldots, c_n)$.

Once this claim is shown, it is clear that, in particular, for any *Boolean* MSO formula φ, letting ψ be its translation according to the above, it is the case that, for any Γ_k^σ-tree E, we have $E \models \psi$ iff $\langle E \rangle \models \varphi$. We can then use [Thatcher and Wright 1968] to translate ψ to a Γ_k^σ-bNTA A that tests ψ: for any Γ_k^σ-tree E, we have $E \models \psi$ iff $E \models A$. We can thus set $\mathcal{A}(q) := A$, and we have ensured that the property required by a tree interpretation scheme is satisfied. Hence, all that remains is to prove the claim above.

To do so, it suffices to show it for the atoms, as we can then simply build ψ for non-atomic φ by substituting the atoms by their translations, and arguing by a straightforward induction that the claim holds.

It is easy to see that we can express in MSO over Γ_k^σ-trees a formula $\mathrm{PE}(\tilde{Z})$ that tests whether $\tilde{Z} \in \tilde{V}_\mathrm{e}$. Indeed, make PE the disjunction of the PE_i, each of which tests the following: Z^j is empty for all $j \neq i$, Z^i is a singleton, and the only element n of Z^i is a proper a_i-intro-node. This can clearly be expressed in MSO over trees. From there, it is straightforward to define $\mathrm{PS}(\tilde{Y})$ that tests whether $\tilde{Y} \in \tilde{V}_\mathrm{s}$.

We now have three kinds of atomic formulae to translate:

- If $\varphi(x_1, x_2)$ is an equality atom $x_1 = x_2$, we define $\psi(\tilde{Z}_1, \tilde{Z}_2)$ to be $\mathrm{PE}(\tilde{Z}_1) \wedge \mathrm{PE}(\tilde{Z}_2) \wedge \bigwedge_{1 \leqslant i \leqslant 2k+2} (\forall x\ (x \in Z_1^i) \Leftrightarrow (x \in Z_2^i))$, and the properties are immediate.

- If $\varphi(X, x)$ is a membership atom $x \in X$, we define $\psi(\tilde{Y}, \tilde{Z})$ to be $\mathrm{PS}(\tilde{Y}) \wedge \mathrm{PE}(\tilde{Z}) \wedge \bigvee_{1 \leqslant i \leqslant 2k+2} (\forall x\ x \in Z_i \Rightarrow x \in Y_i)$, and it is easily seen that the required properties are true.

- If $\varphi(x_1, \ldots, x_m)$ is an actual σ-atom $A = R(x_{i_1}, \ldots, x_{i_p})$ where R has arity p and where $1 \leqslant i_j \leqslant m$ for all $1 \leqslant j \leqslant p$, we set $\psi(\tilde{Z}_1, \ldots, \tilde{Z}_m)$ to be $(\bigwedge_{1 \leqslant i \leqslant m} \mathrm{PE}(Z_i)) \wedge \bigvee_{\mathbf{t} \in \{1, \ldots, 2k+2\}^p} \psi_\mathbf{t}(\tilde{Z}_1, \ldots, \tilde{Z}_m)$, where each $\psi_\mathbf{t}(\tilde{Z}_1, \ldots, \tilde{Z}_m)$ is $\exists n\ L_\mathbf{t}^R(n) \wedge \bigwedge_{1 \leqslant j \leqslant p} \chi^{t_j}(n, \tilde{Z}_{i_j})$, where $L_\mathbf{t}^R(n)$ tests whether the second element of the label of n is exactly $R(a_{t_1}, \ldots, a_{t_m})$, and $\chi^{t_j}(n, \tilde{Z}_{i_j})$ checks that $Z_{i_j}^{t_j}$ is non-empty and that its one element n' is such that n and n' are a_{t_j}-connected.

  To see why this is correct, for the forward direction, observe that when a fact $R(c_1, \ldots, c_p)$ is in $\langle E \rangle$, then it was created by decoding at a node n some fact $R(a_{t_1}, \ldots, a_{t_m})$ for some $\mathbf{t} \in \{1, \ldots, 2k+2\}^p$, and then, for all $1 \leqslant j \leqslant p$, the one node of $\eta(c_j)$ is t_j-connected to n. Conversely, clearly ψ is only satisfied by pseudo-elements, and when it is satisfied, decoding at a witnessing node n

will indeed create an R-fact that satisfies φ on the elements of $\mathrm{dom}(I)$ that the pseudo-elements represent.

This concludes the proof of the fact that we can translate Boolean MSO formulae on σ-instances to Boolean MSO formulae on Γ_k^σ-trees in a way that satisfies the constraints required of the tree interpretation scheme. Hence, we have finished the proof for queries in MSO.

From MSO to GSO. To extend the result from MSO to GSO, we use the fact that any Boolean GSO query q on signature σ can be rewritten to a Boolean MSO query q^* on the *incidence signature* [Grädel, Hirsch, and Otto 2002, Proposition 7.1]. Formally, for a signature σ, the *incidence signature* is an arity-two signature σ^* that contains, for each relation $R \in \sigma$ and for all $1 \leqslant i \leqslant |R|$, a binary predicate R_i^*; and an *incidence instance* for a σ-instance I is a σ^*-instance I^* where we created one fresh element a_F in $\mathrm{dom}(I^*)$ for each $F \in I$ and created the facts $R_i^*(a_F, a_i)$ for each fact $F = R(\mathbf{a})$ and $1 \leqslant i \leqslant |R|$.

Proposition 7.1 of [Grädel, Hirsch, and Otto 2002] then shows that for any Boolean GSO query q over σ, there is a Boolean MSO query q^* over σ^* such that, for any σ-instance I and incidence instance I^* of I, we have $I \models q$ iff $I^* \models q^*$.

Hence, let q be a Boolean GSO query over σ, and let us define $\mathcal{A}(q)$ for q and thus complete the description of our tree interpretation scheme. Let q^* be the query given by [Grädel, Hirsch, and Otto 2002]. Using our previous construction for MSO, translate q^* to a $\Gamma_{k+1}^{\sigma^*}$-bNTA A that tests q^*: for any $\Gamma_{k+1}^{\sigma^*}$-tree E', we have $E' \models A^*$ iff $\langle E' \rangle \models q^*$. Our goal is to define a Γ_k^σ-bNTA A for q. For this, it will prove useful to define a notion of incidence Γ_k^σ-trees. Letting E be a Γ_k^σ-tree, define the $\Gamma_{k+1}^{\sigma^*}$-tree E^* obtained by rewriting each node n of E as follows:

- if $\lambda(n) = (d, s)$ is such that $s = \emptyset$, leave n unchanged;
- if $\lambda(n) = (d, s)$ is such that $s = \{R(\mathbf{t})\}$, writing $p := |R|$, build a chain of nodes $n_0, \ldots, n_{p+1}$ to replace n, where:
  - the parent of n_{p+1} in E^* is the rewriting of the parent of n in E;
  - n_i is the left child of n_{i+1} for all $0 \leqslant i \leqslant p$;
  - the right child of n_i for $0 < i \leqslant p$ is a fresh dummy node labeled by $(\emptyset, \emptyset)$;
  - the left and right children of n_0 in E^* are the rewriting of the left and right children of n in E respectively;
  - we set $\lambda(n_0) = \lambda(n_{p+1}) = (d, \emptyset)$;
  - letting s^* be an incidence instance of s, taking $a_{2(k+1)+2}$ to be the one element of $\mathrm{dom}(s^*) \setminus \mathrm{dom}(s)$, enumerating the facts of s^* as $F_1^*, \ldots, F_p^*$, we set $\lambda(n_i) = (d \sqcup \{a_{2(k+1)+2}\}, \{F_i^*\})$.

It is clear by construction that $\langle E^* \rangle$ is (up to isomorphism) an incidence instance of $\langle E \rangle$. Note that this is not too surprising: if a relational instance I has treewidth ℓ, then it is known that any incidence instance of I has treewidth $\leqslant \ell + 1$ [Grädel, Hirsch, and Otto 2002].

We will now construct a Γ_k^σ-bNTA A so that, for any Γ_k^σ-tree E, we have $E \models A$ iff $E^* \models A^*$. If we can do this, it is clear that $\mathcal{A}(q) := A$ tests q, namely: for any

Γ_k^σ-tree E, we would then have $E \models \mathcal{A}(q)$ iff $E^* \models A^*$ iff $\langle E^* \rangle \models q^*$ iff $\langle E \rangle \models q$. If this is the case, then we are done defining our tree interpretation scheme.

It is in fact easy to define such a Γ_k^σ-bNTA A from A^*. Indeed, writing $A^* = (Q, F, \iota^*, \delta^*)$, let A be the Γ_k^σ-bNTA (Q, F, ι, δ) defined by:

- For all $(d, \emptyset) \in \Gamma_k^\sigma$ and $q, q' \in Q$, set $\iota((d, \emptyset)) := \iota^*((d, \emptyset))$ and $\delta(q, q', (d, \emptyset)) := \delta^*(q, q', (d, \emptyset))$.
- For all $(d, s) \in \Gamma_k^\sigma$ with $s \neq \emptyset$, for all $q, q' \in Q$, set $\iota((d, s))$ to be the result of reading the translation of such a node in E^*, and set $\delta(q, q', (d, s))$ to be the result of reading the translation from states q and q'.

  Formally, letting $s^* = \{F_1^*, \ldots, F_p^*\}$ be an incidence structure of s where the additional element of $\mathrm{dom}(s^*) \setminus \mathrm{dom}(d)$ is $a_{2(k+1)+2}$, letting $q_\perp = \iota((\emptyset, \emptyset))$ and $q_0 = \iota((d, \emptyset))$, define a sequence $q_i = \delta(q_{i-1}, q_\perp, (\mathrm{dom}(s^*), \{F_i^*\}))$ for $1 \leqslant i \leqslant n$, and $q_{n+1} = \delta(q_n, q_\perp, (d, \emptyset))$, and set $\iota((d, n)) := q_{n+1}$. Define likewise $\delta(q, q', (d, s))$ for $q, q' \in Q$ as above but setting $q_0 = \delta(q, q', (d, \emptyset))$ instead.

It is clear that, for any Γ_k^σ-tree E, A has an accepting run on E iff A^* has an accepting run on E^*, so that, by our previous reasoning, A tests q. Hence, we have defined $\mathcal{A}(q)$ for any Boolean GSO query q in a way that satisfies the conditions of a tree interpretation scheme. This concludes our definition of the tree interpretation scheme.

3.3 Provenance Circuits for Treelike Instances

We have shown in the previous section the existence of a *tree interpretation scheme* for GSO queries. We now leverage the construction of Section 3.1 through the tree interpretation scheme, to make it apply to GSO queries on *relational instances* of bounded treewidth, relying on subinstance-compatibility.

We fix the notations Γ_k^σ, $\mathcal{E}$, $\langle\cdot\rangle$, and $\mathcal{A}$, to refer to the subinstance-compatible tree interpretation scheme whose existence was shown in Theorem 3.2.3.

As before, we consider unknown Boolean annotations on the facts of an instance. However, rather than annotating the facts, it is more natural to say that a fact annotated by 1 is *kept*, and a fact annotated by 0 is *deleted*. Formally, given an instance I, a *valuation* ν is a function from the facts of I to $\{0, 1\}$, and we define $\nu(I)$ as the subinstance $\{F \in I \mid \nu(F) = 1\}$ of I.

We can then define the provenance of a query on an instance, without restricting yet to any specific query language:

Definition 3.3.1. The *provenance* of a query q on a σ-instance I is the function $\mathrm{Prov}(q, I)$ mapping any valuation $\nu : I \to \{0, 1\}$ to 0 or 1 depending on whether $\nu(I) \not\models q$ or $\nu(I) \models q$. A *provenance circuit* of q on I is a Boolean circuit C with $C_{\mathsf{inp}} = I$ that captures $\mathrm{Prov}(q, I)$. ◁

We study provenance for treelike instances (i.e., bounded-treewidth instances) and for GSO queries. Combining the results of the two previous sections, we claim that provenance for GSO queries on treelike instances can be computed in linear time data complexity, and that the resulting provenance circuit has treewidth independent of the instance.

Theorem 3.3.2. *For any fixed $k \in \mathbb{N}$ and GSO query q, for any σ-instance I such that $\mathrm{tw}(I) \leqslant k$, we can construct a provenance circuit C of q on I in time $O(|I|)$. The treewidth of C only depends on k and q (not on I).*

To use the results of Section 3.1 and prove the claim, the problem is that we need to work with $\overline{\Gamma_k^\sigma}$-bNTAs rather than Γ_k^σ-bNTAs. Intuitively, we need bNTAs that read *Boolean-annotated* tree encodings, and the Boolean annotation of a node should indicate whether the corresponding fact is *kept* or *deleted*. To work around this technicality, we show how to lift a Γ_k^σ-bNTA A to a $\overline{\Gamma_k^\sigma}$-bNTA such that 1-annotated nodes are read like the original label without annotation, and 0-annotated nodes are read like the *neutering* of the original label, relying on subinstance-compatibility:

Definition 3.3.3. For $\tau \in \Gamma_k^\sigma$ and $b \in \{0,1\}$, we write $\tau_{[b]}$ to be τ if b is 1 and $\underline{\tau}$ if b is 0.

Given a $\overline{\Gamma_k^\sigma}$-tree E, we define its *evaluation* $\epsilon(E)$ as the Γ_k^σ-tree that has same skeleton, where for every node $n \in E$ with corresponding node n' in $\epsilon(E)$, letting $\lambda(n) = (\tau, b) \in \Gamma_k^\sigma \times \{0,1\}$, we have $\lambda(n') = \tau_{[b]}$. ◁

Lemma 3.3.4. *For any Γ_k^σ-bNTA A, one can compute in linear time a $\overline{\Gamma_k^\sigma}$-bNTA A' such that, for any $\overline{\Gamma_k^\sigma}$-tree E, we have $E \models A'$ iff $\epsilon(E) \models A$.*

Proof. Let $A = (Q, F, \iota, \delta)$. We construct the bNTA $A' = (Q, F, \iota', \delta')$ according to the following definition: $\iota'((\tau, b)) := \iota(\tau_{[b]})$ and $\delta'((\tau, b), q_1, q_2) := \delta(\tau_{[b]}, q_1, q_2)$ for all $b \in \{0,1\}$, $\tau \in \Gamma_k^\sigma$, and $q_1, q_2 \in Q$. The process is clearly in linear time in A. Now, it is immediate that $E \models A'$ iff $\epsilon(E) \models A$, because a run of A' on E is a run of A on $\epsilon(E)$, and vice-versa. □

The key point is now that we can describe any subinstance $I' \subseteq I$ as a valuation $\nu : I \to \{0,1\}$, which we can apply to the tree encoding $\mathcal{E}(I)$ following the mapping φ_I, and this tree encoding decodes to I' when evaluated, so the provenance of the lifting of $\mathcal{A}(q)$ on $\mathcal{E}(I)$ matches the provenance of q on I. We now give the formal proof of the main result for this section:

Proof of Theorem 3.3.2. Fix $k \in \mathbb{N}$ and the GSO query q, and compute a Γ_k^σ-bNTA $A := \mathcal{A}(q)$ that tests q on instances of treewidth $\leqslant k$. Let A' be the $\overline{\Gamma_k^\sigma}$-bNTA obtained by lifting A using Lemma 3.3.4. All of this is performed in constant time in the instance.

Now, given the input instance I such that $\mathrm{tw}(I) \leqslant k$, we compute in linear time its encoding $\mathcal{E}(I)$. We now use Theorem 3.1.4 to construct a provenance circuit C of A' on I. Using subinstance-compatibility, consider now the function φ_I that maps the facts of I to the nodes of E where those facts are encoded, which we also compute in linear time in I. We modify C to replace the input gate g_n^{i} for any $n \in E$ not in the image of φ_I, setting it to be a 0-gate; and renaming the input gates g_n^{i} for any $n \in \varphi_I(I)$ to identify them with the fact F such that $\varphi_I(F) = n$. Let C' be the result of this process. C' is thus a Boolean circuit such that $C'_{\mathsf{inp}} = I$, it was computed in linear time from I, and we know that its treewidth is at most that of C, which only depends on A'. Hence, the width of C' only depends on A', which only depends on k and q, not on I.

The only thing left to do is to show that C' captures $\mathrm{Prov}(q, I)$. Let $\nu : I \to \{0,1\}$ be a valuation of I. We show that $\nu(C') = 1$ iff $\nu(I) \models q$. Let ν' be the valuation

of E defined by $\nu'(n) = \nu(\varphi_I^{-1}(n))$ if n is in the image of φ_I, and $\nu'(n) = 0$ otherwise. We then know by subinstance-compatibility that $\langle \epsilon(\nu'(E)) \rangle$ is isomorphic to the subinstance $\nu(I)$ of I. Having observed this, we know that, because A tests q, we have $\nu(I) \models q$ iff $\epsilon(\nu'(E)) \models A$. Now, by definition of Lemma 3.3.4, we have $\epsilon(\nu'(E)) \models A$ iff $\nu'(E) \models A'$, which by definition of the provenance circuit C is the case iff $\nu'(C) = 1$, which by definition of C' is the case iff $\nu(C') = 1$. Hence, C' is indeed a provenance circuit of q on I. □

We conclude the section by two remarks on Theorem 3.3.2:

- Our definition of provenance is intrinsic to the query and does not depend on the query's formulation. Hence, the provenance that we capture by this result does *not* depend on the choice of bNTA to translate the query, or on the choice of tree decomposition.
- In contrast to Section 3.1, tractability holds only in data complexity. For combined complexity, we incur the cost of translating the query to an automaton, which is nonelementary in general [Meyer 1975]. However, for some restricted query classes, the translation phase has lower cost, for instance it is in EXPTIME for UCQ queries [Amarilli, Bourhis, and Senellart 2015, Appendix B.4].

3.4 Monotone Provenance Circuits

We now specialize the previous results to the case of *monotone* GSO queries, showing that we can then construct *monotone* provenance circuits. This will be relevant for our extensions to the setting of *semiring provenance* [Green, Karvounarakis, and Tannen 2007], more specifically PosBool$[X]$*-provenance*, in Chapter 5.

It is easy to observe that if a query is monotone, then its provenance is monotone as well:

Lemma 3.4.1. *For any query q, if q is monotone, then* $\mathrm{Prov}(q, I)$ *is a monotone Boolean function for any instance I.*

Proof. Fix q and I and consider any two valuations $\nu \leqslant \nu'$ of I. By definition of $\leqslant$, this implies $\nu(I) \subseteq \nu'(I)$. Hence, as q is monotone, if $\nu(I) \models q$ then $\nu'(I) \models q$. Hence, letting $\varphi := \mathrm{Prov}(q, I)$, if $\varphi(\nu) = 1$ then $\varphi(\nu') = 1$. □

Hence, it is natural to wonder whether Theorem 3.1.4 and Theorem 3.3.2 can be used to create *monotone* provenance circuits for monotone queries, to capture this monotone provenance. Indeed, in this section, we will show:

Theorem 3.4.2. *For any fixed $k \in \mathbb{N}$ and* monotone *GSO query q, for any σ-instance I such that* $\mathrm{tw}(I) \leqslant k$*, one can construct in time $O(|I|)$ a* monotone *provenance circuit of q on I whose treewidth only depends on k and q (not on I).*

The proof of this result relies on a natural notion of *monotonicity* for tree automata on trees annotated by Boolean values. Intuitively, if a $\overline{\Gamma}$-tree is accepted by a monotone automaton, it will not be rejected when changing annotations from 0 to 1.

Definition 3.4.3. Fix any alphabet Γ, and consider the partial order $<$ on $\overline{\Gamma}$ defined by $(\tau, 0) < (\tau, 1)$ for all $\tau \in \Gamma$.

We say that a $\overline{\Gamma}$-bNTA $A = (Q, F, \iota, \delta)$ is *monotone* if for every $\tau \leqslant \tau'$ in $\overline{\Gamma}$, we have $\iota(\tau) \subseteq \iota(\tau')$ and $\delta(q_1, q_2, \tau) \subseteq \delta(q_1, q_2, \tau')$ for every $q_1, q_2 \in Q$. ◁

The definition of automaton monotonicity clearly ensures the following: for any *monotone* $\overline{\Gamma}$-bNTA A, for any Γ-tree T, for any valuations $\nu \leqslant \nu'$ of T (according to the order on Boolean functions defined in Section 2.2), if $\nu(T) \models A$, then $\nu'(T) \models A$. Indeed, monotonicity ensures that any run of A on $\nu(T)$ is also a run of A on $\nu'(T)$. This implies that the provenance $\mathrm{Prov}(A, T)$ of A on T (recall Definition 3.1.2) is then a *monotone* Boolean function: for any valuations $\nu \leqslant \nu'$ of T, letting $\varphi := \mathrm{Prov}(A, T)$), if $\varphi(\nu) = 1$ then $\varphi(\nu') = 1$.

As it turns out, adapting Theorem 3.1.4, we can show that the provenance of a monotone automaton can be tractably captured by a monotone circuit:

Theorem 3.4.4. *Given a* monotone *$\overline{\Gamma}$-bNTA A and a Γ-tree T, we can construct in time $O(|A| \cdot |T|)$ a* monotone *provenance circuit C of A on T that has treewidth $O(|A|)$.*

Proof. We do the same construction as in the proof of Theorem 3.1.4, but we replace all NOT-gates, namely the gates $g_n^{\neg i}$ for $n \in T$, by 1-gates. The running time and treewidth are clearly unaffected, so the only thing to adapt is the correctness proof. We do so for each valuation ν of T by proving by induction on $n \in T$ the same claim as before: for any $q \in Q$, we have $\nu(g_n^q) = 1$ iff there is a run ρ of A on the subtree T_n of T rooted at n such that $\rho(n) = q$.

For a leaf node n, for $q \in Q$, either $\nu(n) = 0$ or $\nu(n) = 1$. If $\nu(n) = 0$, we show the claim exactly as before. If $\nu(n) = 1$, it is no longer the case that $\nu(g_n^q) = 1$ iff $q \in \iota(\lambda(n), \nu(n))$; we have instead $\nu(g_n^q) = 1$ iff $q \in \iota(\lambda(n), 0) \cup \iota(\lambda(n), 1)$. Now, as A is monotone, we have $\iota(\lambda(n), 0) \subseteq \iota(\lambda(n), 1)$, so we can in fact conclude as before.

For an internal node n, for $q \in Q$, we adapt the proof in the same way. If $\nu(n) = 0$, the proof is exactly the same. If $\nu(n) = 1$, we have $\nu(g_n^q) = \nu(n)$ iff there are some $q_\mathrm{L}, q_\mathrm{R} \in Q$ such that $\nu(g_{\mathrm{L}(n)}^{q_\mathrm{L}}) = 1$, $\nu(g_{\mathrm{R}(n)}^{q_\mathrm{R}}) = 1$, and $q \in \delta(q_\mathrm{L}, q_\mathrm{R}, (\lambda(n), 0)) \cup \delta(q_\mathrm{L}, q_\mathrm{R}, (\lambda(n), 1))$. We again use the monotonicity of A to conclude in the same manner as before.

This concludes the adaptation of the correctness proof, so that our claim holds. □

We now adapt the results of Section 3.3 to show that we can translate monotone GSO queries to monotone bNTAs. We can do this without relying on the internal details of the translation: we simply show how to modify Lemma 3.3.4 to lift an arbitrary bNTA that tests the query to a monotone $\overline{\Gamma_k^\sigma}$-bNTAs.

Lemma 3.4.5. *For any $k \in \mathbb{N}$, for any Γ_k^σ-bNTA A that tests a monotone query q, one can compute in linear time a* monotone *$\overline{\Gamma_k^\sigma}$-bNTA A'' such that, for any $\overline{\Gamma_k^\sigma}$-tree E, we have $E \models A''$ iff $\epsilon(E) \models A$.*

Proof. Let A be the Γ_k^σ-bNTA that tests q, and let $A' = (Q', F', \iota', \delta')$ be the $\overline{\Gamma_k^\sigma}$-bNTA that we constructed in Lemma 3.3.4, so that we have $E \models A'$ iff $\epsilon(E) \models A$.

We build the bNTA $A'' = (Q', F', \iota'', \delta'')$ by setting, for all $(\tau, i) \in \overline{\Gamma_k^\sigma}$ and for all $q_1, q_2 \in Q$, $\iota''((\tau, i)) := \bigcup_{0 \leqslant j \leqslant i} \iota'((\tau, j))$ and $\delta''(q_1, q_2, (\tau, i)) := \bigcup_{0 \leqslant j \leqslant i} \delta'(q_1, q_2, (\tau, j))$.

Clearly A'' is monotone by construction for $\overline{\Gamma_k^\sigma}$; it suffices to show that, for any $\overline{\Gamma_k^\sigma}$-tree E, we have $E \models A'$ iff $E \models A''$. The forward implication is immediate, so it suffices to prove the converse implication.

Let E be a $\overline{\Gamma_k^\sigma}$-tree, and consider an accepting run ρ of A'' on E. We build a new tree E' whose skeleton is that of E and where for any leaf (resp. internal node) $n' \in E'$ with corresponding node $n \in E$ with $\lambda(n) = (\tau, j)$, we set $\lambda(n')$ in E' to be (τ, i) for some $i \in \{0, 1\}$ such that $\rho(n) \in \iota((\tau, i))$ (resp. $\rho(n) \in \delta(\rho(L(n)), \rho(R(n)), (\tau, i))$); the existence of such an i is guaranteed by the definition of ι' (resp. δ').

We now observe that, by construction, ρ is a run of A' on E', and it is still accepting, so that E' is accepted by A'. Hence, $\langle \epsilon(E') \rangle \models q$. But now we observe that, once again by construction, for every node n' of E' with label τ' and with corresponding node n in E with label τ, it holds that $\tau' \leqslant \tau$. From this we see by subinstance-compatibility that $\langle \epsilon(E') \rangle \subseteq \langle \epsilon(E) \rangle$, and thus, by monotonicity of q, we must have $\langle E \rangle \models q$. Thus, as A' tests q, we must have $E \models A'$, proving the desired result. □

We can then prove Theorem 3.4.2 similarly to our proof of Theorem 3.3.2, applying Lemma 3.4.5 instead of Lemma 3.3.4, and applying Theorem 3.4.4 instead of Theorem 3.1.4, relying on the fact that the bNTA created by Lemma 3.4.5 is monotone. This concludes the adaptation of our result to monotone queries and monotone circuits.

3.5 OBDD and d-DNNF Representations

We have shown that we can compute bounded-treewidth (monotone) provenance circuits for bounded-treewidth instances and (monotone) GSO queries, in linear time in the input instance. In this section and the next, we study what *other* forms of provenance representations can be tractably computed for such queries and instances. We specifically focus on the lineage representations such as OBDDs and d-DNNFs commonly used in the field of knowledge compilation [Jha and Suciu 2013]. Our results are summarized in the top part of Table 3.1.

To do this, we notice that the Boolean circuits that we compute are the same as the *expression DAGs* of [Jha and Suciu 2012]. Hence, by a result of [Jha and Suciu 2012], we deduce that GSO queries on bounded-treewidth instances have polynomial-size *OBDD representations* of their lineage [Bryant 1992; Olteanu and Huang 2008], as we show in Section 3.5.1. Further, it is easy to notice that our construction gives *bounded-pathwidth* provenance circuits for GSO queries on *bounded-pathwidth* instances, so that they have *constant-width* OBDDs in this context: we show this in Section 3.5.2. The results of [Jha and Suciu 2012] only claim the *existence* of these tractable lineage representations, so we extend these results to show that they can be computed efficiently.

We then show in Section 3.5.3, by a straightforward modification of our construction, that we can ensure that our linear-size provenance circuits are d-DNNFs [Darwiche 2001] if we compile queries to *deterministic* automata. As we will show in the next chapter, this implies the tractability of probability evaluation on bounded-treewidth TID instances (Theorem 4.1.1), and we will also use it to show tractability in different probabilistic frameworks.

Table 3.1: Bounds for lineage representations (including intractable ones)

Upper bounds (Section 3.5): computation bounds imply size bounds

Instance	Queries	Representation	Note	Time	Source	
bounded-pw	GSO	OBDD	$O(1)$ width	$O(n)$	**here**	Thm. 3.5.4
bounded-pw	(monotone) GSO	(monotone) circuit	bounded-pw	$O(n)$	**here**	Prop. 3.5.5
bounded-tw	GSO	OBDD		$O(\mathrm{Poly}(n))$	**here**	Thm. 3.5.2
bounded-tw	(monotone) GSO	(monotone) circuit	bounded-tw	$O(n)$	**here**	Thm. 3.3.2
bounded-tw	GSO	d-DNNF		$O(n)$	**here**	Thm. 3.5.8
any	inv.-free UCQ	OBDD	$O(1)$ width	$O(\mathrm{Poly}(n))$	[Jha and Suciu 2013]	Prop. 5
any	pos. rel. alg.	monotone formula		$O(\mathrm{Poly}(n))$	[Imieliński and Lipski 1984]	Thm. 7.1
any	Datalog	monotone circuit		$O(\mathrm{Poly}(n))$	[Deutch, Milo, Roy, and Tannen 2014]	Thm. 2

Lower bounds (Section 3.6)

Instance	Queries	Representation	Size	Source	
any	Datalog	monotone formula	$n^{\Omega(\log n)}$	[Deutch, Milo, Roy, and Tannen 2014]	Thm. 1
tree	tree automaton	formula	$\Omega(n^2)$	**here**	Prop. 3.6.1
tree	MSO	formula	$\Omega(n^2)$	**here**	Prop. 3.6.2
tree	$\mathrm{CQ}^{\neq}$	formula	$\Omega(n \log \log n)$	**here**	Prop. 3.6.3
tree	$\mathrm{CQ}^{\neq}$	monotone formula	$\Omega(n \log n)$	**here**	Prop. 3.6.4

As for *formula-based* representations of lineage, e.g., read-once formulae [Jha and Suciu 2013], we will show in the next section that they cannot be as concise as our linear-size Boolean circuits or d-DNNF representations, which justifies our decision to use circuit-based representations of lineage.

3.5.1 OBDD

We start by defining *OBDDs*, a common tractable representation of Boolean functions [Bryant 1992; Olteanu and Huang 2008]:

Definition 3.5.1. An *ordered binary decision diagram* (*OBDD*) is a rooted directed acyclic graph (DAG) whose leaves are labeled 0 or 1, and whose non-leaf nodes are labeled with a variable and have two outgoing edges labeled 0 and 1. We require that there exists a total order Π on the variables such that, for every path from the root to a leaf, no variable occurs in two different internal nodes on the path, and the order in which the variables occur is compatible with Π.

An OBDD defines a Boolean function on its variables: each valuation is mapped to the value of the leaf reached from the root by following the path given by the valuation.

The *size* of an OBDD is its number of nodes, and its *width* is the maximum number of nodes at every *level*, where a level is the set of nodes reachable by enumerating all possible values of variables in a prefix of Π. ◁

Our result is that we can compute polynomial-size OBDDs for GSO queries on bounded-treewidth instances in PTIME:

Theorem 3.5.2. *For any fixed GSO query q and $k \in \mathbb{N}$, there is $c \in \mathbb{N}$ such that, given an input instance I of treewidth $\leqslant k$, one can compute in time $O(|I|^c)$ an OBDD (of size $O(|I|^c)$) capturing* $\mathrm{Prov}(q, I)$.

We show this using [Jha and Suciu 2012, Corollary 2.14]: any bounded-treewidth Boolean circuit can be represented by an equivalent OBDD of polynomial width. We complete this result and show that the OBDD can also be computed in polynomial time:

Lemma 3.5.3. *For any $k \in \mathbb{N}$, there is $c \in \mathbb{N}$ such that, given a Boolean circuit C of treewidth k, we can compute an equivalent OBDD in time $O(|C|^c)$.*

Combined with our Theorem 3.3.2, this lemma clearly implies Theorem 3.5.2, so it suffices to prove the lemma. The overall idea is to construct in PTIME the variable order used in [Jha and Suciu 2012]. We then build the corresponding tractable OBDD level by level, testing the equivalence of partial valuations in PTIME thanks to the fact that the circuit for C has bounded treewidth. Here is the formal proof:

Proof of Lemma 3.5.3. We rely on [Jha and Suciu 2012, Corollary 2.14]: there is a doubly exponential function f such that, for any $k \in \mathbb{N}$, there is $c' := f(k) \in \mathbb{N}$ such that, for any tree decomposition T of width $\leqslant k$ of C, the OBDD O obtained for a certain variable order Π^R has width c'.

The order Π^R on variables is defined following an in-order traversal of T where children are ordered by the number of variables in this subtree; clearly this quantity can be computed over the entire tree in PTIME, so the order Π^R can be computed

in PTIME. We show that we can construct the OBDD O in PTIME as well, in a level-wise manner inspired by [Jha and Suciu 2013].

Write $\Pi^R = X_1, \ldots, X_n$, and construct O level-by-level in the following way. Assuming that we have constructed O up to level $l-1$, create two children for each node at level $l-1$ (depending on the value of variable X_l), and then merge all such children n and n' that are *equivalent*. To define this, call *equivalent* two partial valuations ν and ν' of variables $X_1, \ldots, X_l$ if the Boolean function represented by C on the other variables $X_{l+1}, \ldots, X_n$ under ν is the same as under ν'. Now, call n and n' *equivalent* if, for any partial valuation ν leading to n (represented by a path from the root of O to n) and any partial valuation ν' leading to n', these two partial valuations of $X_1, \ldots, X_l$ are equivalent. As we will always ensure in the construction, any paths leading to the parent node of n are equivalent partial valuations of $X_1, \ldots, X_{l-1}$, so n and n' are equivalent iff, picking any two valuations ν for n and ν' for n' by following a path from the root to n and to n' respectively, ν and ν' are equivalent.

Hence, it suffices to show that there is a function g such that we can test in time $O(|C|^{g(k)})$ whether two partial valuations are equivalent. Indeed, we can then build O in the indicated time, because the maximal number of node pairs to test at any level of the OBDD is $\leqslant (2 \cdot |C|^{f(k)})^2$: we had at most $|C|^{f(k)}$ at the previous level, and each of them creates two children, before we merge the equivalent children. Hence, if we can test equivalence in the indicated time, then clearly we can construct O in time $O(|C|^c)$ for $c := 1 + 1 + 2 \cdot f(k) + g(k)$ (the first term accounts for the linear number of levels, and the second term accounts for the linear time required to find a partial valuation for a node).

We thus show that the equivalence of partial valuations can be tested in time $O(|C|^{g(k)})$ for some function g. Considering two partial valuations ν and ν' of the same set of variables $\mathcal{X}$, let C_ν and $C_{\nu'}$ be the two circuits obtained from C by substituting the input gates for $\mathcal{X}$ with constant gates according to ν and ν' respectively. Note that C_ν and $C_{\nu'}$ have the same set of input gates $\mathcal{X}'$, formed precisely of the variables not in $\mathcal{X}$. We rename the internal gates of $C_{\nu'}$ so that the only gates shared between C_ν and $C_{\nu'}$ are the input gates $\mathcal{X}'$. Now, C' be the circuit obtained by taking the union of C_ν and $C_{\nu'}$ (on the same set of variables), and adding an output gate and a constant number of gates such that the output gate is true iff the output gates of C_ν and $C_{\nu'}$ carry different values (this can be done with 5 additional gates in total). It is easy to see that there is a valuation of $\mathcal{X}'$ that makes the circuit C' evaluate to true iff the partial valuations ν and ν' are *not* equivalent. Now, observe that we can immediately construct from T a tree decomposition T'' of width $\leqslant 2k+5$ of C'. Indeed, it is obvious that T is a tree decomposition of C_ν, and we can rename gates to obtain from T a tree decomposition T' of $C_{\nu'}$, such that T and T'' both have the same width k and the same skeleton. Now, construct T'' that has same skeleton as T and T', where each bag is the union of the corresponding bags of T and T', adding the 5 intermediate gates to each bag. The result T'' clearly has width $\leqslant 2k+5$ and it is immediate that it is a tree decomposition of C'.

We can then use message-passing techniques [Lauritzen and Spiegelhalter 1988; Huang and Darwiche 1996] to determine in time exponential in $2k+5$ and polynomial in C' whether the bounded-treewidth circuit C' has a satisfying assignment, from which we deduce whether ν and ν' are equivalent. For details, see, e.g., [Amarilli, Bourhis, and Senellart 2015, Theorem D.2]. □

3.5.2 Bounded-Pathwidth

We now refine the previous result: in the case of *bounded pathwidth* instances, we can compute *constant-width* OBDDs:

Theorem 3.5.4. *For any fixed GSO query q and constant $k \in \mathbb{N}$, given an input instance I of pathwidth $\leqslant k$, one can compute in polynomial time an OBDD of constant width capturing* $\mathrm{Prov}(q, I)$.

To prove the result, we first observe that our results construct *bounded-pathwidth* provenance circuits in linear time on bounded-pathwidth instances:

Proposition 3.5.5. *For any fixed $k \in \mathbb{N}$ and (monotone) GSO query q, for any σ-instance I of* pathwidth $\leqslant k$, *we can construct a (monotone) provenance circuit C of q on I in time $O(|I|)$. The* pathwidth *of C only depends on k and q (not on I).*

Proof. Given a path decomposition of an instance I, which is a tree decomposition with a linear tree, the resulting tree encoding E of I is clearly also a linear tree. Hence, as the provenance circuit C constructed in Theorems 3.1.4 and Theorem 3.4.4 has a tree decomposition with same skeleton as E. As the result C' of Theorems 3.3.2 and 3.4.2 is the same circuit as C up to replacing some input gates by constant gates and performing a bijective renaming of input gates, we conclude that C' is the desired bounded-pathwidth circuit. □

By [Jha and Suciu 2012, Corollary 2.13], this implies the existence of a constant-width OBDD representation, which we again show to be computable, proving Theorem 3.5.4.

Lemma 3.5.6. *For any $k \in \mathbb{N}$, for any Boolean circuit C of pathwidth $\leqslant k$, we can compute in polynomial time in C an OBDD equivalent to C whose width depends only on k.*

Proof. As in the proof of Lemma 3.5.3, we can compute in PTIME the order Π^R on variables, and we can compute the OBDD under this order in the same way. This uses the fact that a path decomposition of circuit C is in particular a tree decomposition of C. □

3.5.3 d-DNNF

We now turn to the more expressive tractable lineage formalism of *d-DNNFs*, introduced in [Darwiche 2001]; we follow the definitions of [Jha and Suciu 2013]:

Definition 3.5.7. A *deterministic, decomposable negation normal form* (*d-DNNF*) is a Boolean circuit $C = (G, W, g_0, \mu)$ that satisfies the following conditions:

1. Negation is only on input gates. Formally, for any gate $g \in G$ with $\mu(g) = \neg$, the one gate $g' \in G$ such that $(g', g) \in W$ must be such that $g' \in C_{\mathsf{inp}}$.

2. The inputs of AND-gates depend on disjoint sets of input gates. Formally, for any g with $\mu(g) = \wedge$, for any two gates $g_1 \neq g_2$ in G such that $(g_1, g) \in W$ and $(g_2, g) \in W$, there is no input gate $g' \in C_{\mathsf{inp}}$ such that both g_1 and g_2 are reachable from g' in the DAG (G, W);

3. The inputs of OR-gates are mutually exclusive. Formally, for any g with $\mu(g) = \vee$, for any two gates $g_1 \neq g_2$ in G such that $(g_1, g) \in W$ and $(g_2, g) \in W$, there is no valuation ν of C such that g_1 and g_2 both evaluate to 1 under ν. $\lhd$

We show a variant of Theorem 3.3.2 where we construct d-DNNFs instead of general circuits.

Theorem 3.5.8. *For any fixed GSO query q and constant $k \in \mathbb{N}$, given an input instance I of treewidth $\leqslant k$, one can compute in time $O(|I|)$ a d-DNNF capturing* $\mathrm{Prov}(q, I)$.

We will use this result in the next chapter to derive the tractability of probability evaluation, as this problem is tractable for d-DNNFs.

The idea of the proof is that we automatically obtain a d-DNNF if we choose to translate queries to a *deterministic* automaton, that is, a *bDTA* rather than a *bNTA*. Let us show this:

Proof of Theorem 3.5.8. We adapt Theorem 3.1.4 to show that a provenance d-DNNF of a *deterministic* $\overline{\Gamma}$-bDTA A on a $\overline{\Gamma}$-tree E can be constructed in time $O(|A| \cdot |E|)$. We construct the circuit exactly as in our previous proof of Theorem 3.1.4, and we now show that it is a d-DNNF.

1. First, observe that the only NOT gates that we use are the $g_n^{\neg \mathrm{i}}$, which are NOT gates of the g_n^{i}, which are input gates; so we only apply negation to leaf nodes.

2. Second, we show that the sets of leaves reachable from the children of any AND gate are pairwise disjoint. The AND gates that we create and that have multiple inputs are the following:

   - The $g_n^{q_L, q_R}$, which are the AND of $g_{\mathrm{L}(n)}^{q_L}$ and $g_{\mathrm{R}(n)}^{q_L}$; now, $g_{\mathrm{L}(n)}^{q_L}$ only depends on the input gates $g_{n'}^{\mathrm{i}}$ for nodes n' of the subtree of E rooted at $\mathrm{L}(n)$, and likewise $g_{\mathrm{R}(n)}^{q_L}$ only depends on input gates in the right subtree;
   - The $g_n^{q_L, q_R, \mathrm{i}}$, which are the AND of $g_n^{q_L, q_R}$ and g_n^{i}; now, the $g_n^{q_L, q_R}$ do not depend on g_n^{i}, only on input gates $g_{n'}^{\mathrm{i}}$ for n' a strict descendant of n in E;
   - The $g_n^{q_L, q_R, \neg \mathrm{i}}$, which are the AND of $g_n^{q_L, q_R}$ and $g_n^{\neg \mathrm{i}}$; now, the $g_n^{q_L, q_R}$ do not depend on the sole input gate under $g_n^{\neg \mathrm{i}}$, i.e., g_n^{i}, but only on input gates $g_{n'}^{i}$ for n' a strict descendant of n in E.

3. Third, we show that the children of any OR gate are mutually exclusive. The OR gates that we create and that have multiple inputs are the following:

   - The g_n^q when n is a leaf node of E, for which the claim is immediate, as the only two possible children are g_n^{i} and $g_n^{\neg \mathrm{i}}$ which are clearly mutually exclusive.
   - The g_n^q when n is an internal node of E, which are the OR of gates of the form $g_n^{q_L, q_R, \mathrm{i}}$ or $g_n^{q_L, q_R, \neg \mathrm{i}}$ over several pairs q_L, q_R.
     To observe that these gates are mutually exclusive, remember that, for a valuation ν of the tree E, the gate $g_{n'}^q$ is true iff there is a run ρ of A on the subtree of $\nu(E)$ rooted at n' such that $\rho(n') = q$. However, as A is deterministic, for each n', there is at most one state q for which this

is possible. Hence, for any valuation ν' of the circuit C, for our node n, there is at most one q'_L such that $g^{q'_L}_{\mathrm{L}(n)}$ is true under valuation ν', and only at most one q'_R such that $g^{q'_R}_{\mathrm{R}(n)}$ is true under ν'. Hence, by definition of the g^{q_L,q_R}_n, there is at most one of them which can be true under valuation ν', namely, $g^{q'_L,q'_R}_n$, which also means that only the gate $g^{q'_L,q'_R,\mathrm{i}}_n$ and the gate $g^{q'_L,q'_R,\neg\mathrm{i}}_n$ can be true under ν'. But these two gates are clearly mutually exclusive (only one can evaluate to true, depending on the value of $\nu(n)$), which proves the claim.

- The output gate g_0 which is the OR of gates of the form g^q_r for r the root node of E. Again, as the automaton A is deterministic, for any valuation ν' of C, letting ν be the corresponding valuation of the $\overline{\Gamma}$-tree E, there is only one state q' such that A has a run ρ on E with $\rho(r) = q'$, so at most one state q' such that $g^{q'}_r$ is true under ν.

Having shown this variant of Theorem 3.1.4, we can clearly modify our tree interpretation scheme to translate queries, not to a Γ^σ_k-bNTA, but to a Γ^σ_k-bDTA: just construct the bDTA from the bNTA using standard techniques [Comon et al. 2007], which is still instance-independent so it does not affect the data complexity. Now we conclude the proof of Theorem 3.5.8 by constructing the d-DNNF as a provenance circuit, just like in the original proof of Theorem 3.3.2, but using instead the variant of Theorem 3.1.4 that we showed, noting that Lemma 3.3.4 preserves the fact that the automaton is deterministic. We can then perform the same replacement of input gates by constant gates and bijective renaming of the input gates, as in the original proof of Theorem 3.3.2, which does not affect the fact that the result is a d-DNNF (up to evaluating negations of constant gates as constant gates). This concludes the proof. □

3.6 Formula Representations

We have shown that GSO queries on bounded-treewidth instances have linear provenance representations as bounded-treewidth circuits and as d-DNNFs. In this section, we study whether we could obtain linear provenance representations as *Boolean formulae*, e.g., as *read-once formulae* [Jha and Suciu 2013]. This question is natural because most prior works on provenance (with the notable exception of [Deutch, Milo, Roy, and Tannen 2014]) focus on formula representations of provenance, and existing work on probabilistic data seldom represents query lineages as Boolean circuits (rather than Boolean formulae, or other representations such as OBDDs, FBDDs and d-DNNFs [Jha and Suciu 2013]).

Intuitively, an important difference between formula and circuit representations is that circuits can share common subformulae whereas formulae cannot. We show in this section that this difference matters: formula-based representations of provenance for GSO queries on bounded-treewidth instances *cannot* be linear in general, because of superlinear lower bounds. We derive these bounds from classical results [Wegener 1987] and summarize them in the lower part of Table 3.1.

We first show in Section 3.6.1 that, already in the setting of provenance for tree automata, Theorem 3.1.4 cannot extend if we ask for a representation of provenance as a Boolean formula, rather than as a Boolean circuit: we construct an automaton,

intuitively capturing the parity function, for which there is a quadratic lower bound on a representation of provenance as a Boolean formula. We translate this result in Section 3.6.2 to show the same for MSO queries on treelike instances. By contrast, we know that there are linear-size provenance representations as circuits in this context, and that they can even be computed in linear time (Theorems 3.1.4 and 3.3.2). While we do not know whether these bounds are tight, note that even an $\Omega(|I|^3)$ lower bound is probably out of reach without new developments in the study of Boolean formulae. Indeed, the best currently known lower bound on formula size, for *any* explicit function of n variables with linear circuits, is in $\Omega(n^{3-o(1)})$ [Håstad 1998].

We then show in Section 3.6.3 that a superlinear lower bound already holds for formula provenance representations of even $\mathrm{CQ}^{\neq}$ queries, and on trivial instances; more specifically, the bound is $\Omega(n \log \log n)$. We then show that this can be improved to $\Omega(n \log n)$ if we ask for a *monotone* provenance formula.

We note that formula-based and circuit-based representations of Boolean provenance were already investigated in the recent work [Deutch, Milo, Roy, and Tannen 2014], whose Theorem 1 establishes a superpolynomial lower bound on the size of formula provenance representations for Datalog queries. Their results, however, apply to *arbitrary* instances; our results hold even on bounded-treewidth instances.

3.6.1 Tree Automata Lower Bounds

We start by studying provenance representations for tree automata, as in Section 3.1, and show a quadratic lower bound on the size of Boolean formulae that capture the provenance of a certain tree automata.

Proposition 3.6.1. *There is a finite alphabet Γ and a $\overline{\Gamma}$-bNTA A, such that, for any Γ-tree T and Boolean formula φ capturing* $\mathrm{Prov}(A, T)$*, we have* $|\varphi| = \Omega(|T|^2)$.

Proof. Consider $\Gamma = \{\bullet\}$, so that all nodes have label $\bullet$. Consider the $\overline{\Gamma}$-bNTA $A = (Q, F, \iota, \delta)$ defined as follows:

- $Q := \{0, 1\}$;
- $F := \{1\}$;
- $\iota((\bullet, b)) = b$ for all $b \in \{0, 1\}$;
- $\delta(q_1, q_2, (\bullet, b)) := q_1 \oplus q_2 \oplus b$, where $\oplus$ denotes exclusive OR.

It is clear that, for any $\overline{\Gamma}$-tree T, we have $T \models A$ iff T has an odd number of nodes annotated 1. Hence, $\mathrm{Prov}(A, T_n)$ for any tree T_n with n nodes is the parity function over n variables.

Now, by [Wegener 1987, Chapter 8, Theorem 8.2], any Boolean formula using $\wedge$, $\vee$, $\neg$ that expresses the parity function over n variables has a number of variable occurrences that is at least n^2. This implies the claimed lower bound. □

3.6.2 MSO Lower Bounds on Treelike Instances

We can adapt the result of the previous section to show a quadratic lower bound for formula-based representations of the provenance of MSO queries on treelike instances, more specifically on trees:

Proposition 3.6.2. *There is an MSO query q, and a family $\mathcal{I}$ of relational instances with treewidth 1, such that, for any $I \in \mathcal{I}$, for any Boolean formula ψ capturing* $\mathrm{Prov}(q, I)$*, we have* $|\psi| = \Omega(|I|^2)$.

We adapt the previous proof by testing the parity of the number of facts in a unary predicate, using an auxiliary binary relation, and relying on the same lower bound.

Proof. Consider the signature σ with a unary predicate L and binary predicate E. We define the family $\mathcal{I} = (I_n)$ with I_n having domain $\{a_1, \ldots, a_n\}$ and facts $L(a_i)$ for each $1 \leqslant i \leqslant n$ and $E(a_i, a_{i+1})$ for each $1 \leqslant i < n$. Clearly, all instances in $\mathcal{I}$ have treewidth 1. We consider the MSO formula q that intuitively uses the E-facts to test whether the number of L-facts is odd. Formally, we define q as follows, inspired by the definition of the tree automaton in the proof of Proposition 3.6.1:

$$q := \forall X_0 X_1 \, \mathrm{Part}(X_0, X_1) \wedge \mathrm{Tr}(X_0, X_1) \wedge \mathrm{Init}(X_0, X_1) \Rightarrow \forall x \, (\neg \exists y \, E(y, x) \Rightarrow x \in X_1)$$

where $\mathrm{Part}(X_0, X_1)$ asserts that X_0 and X_1 partition the domain (where $\oplus$ denotes exclusive OR):

$$\mathrm{Part}(X_0, X_1) := \forall x \, (x \in X_0) \oplus (x \in X_1)$$

$\mathrm{Tr}(X_0, X_1)$ is the conjunction of the following transition rules, for each $b \neq b'$ in $\{0, 1\}$:

$$\forall xy \; E(x, y) \wedge y \in X_b \wedge L(x) \Rightarrow x \in X_{b'}$$
$$\forall xy \; E(x, y) \wedge y \in X_b \wedge \neg L(x) \Rightarrow x \in X_b$$

and $\mathrm{Init}(X_0, X_1)$ asserts the initial states:

$$\forall x \, (\neg \exists y \; E(x, y)) \wedge \neg L(x) \Rightarrow x \in X_0$$
$$\forall x \, (\neg \exists y \; E(x, y)) \wedge L(x) \Rightarrow x \in X_1$$

Intuitively, on any possible world of I_n where all E-facts are present, q is true whenever the number of L-facts is odd. Indeed, it is clear that there is a unique choice of X_0 and X_1 in such worlds, defined by putting a_n in X_1 or X_0 depending on whether $L(a_n)$ holds, and, for $1 \leqslant i < n$, letting b such that $a_{i+1} \in X_b$ and b' be 1 or 0 depending on whether $L(a_i)$ holds or not, putting a_i in $X_{b \oplus b'}$. Hence, in worlds containing all E-facts, there is a unique choice of X_0 and X_1 where we have $a_i \in X_1$ iff the number of facts $L(a_j)$ with $i \leqslant j \leqslant n$ is odd. Hence, q is satisfied iff, in this unique assignment, a_1 (the only node with no incoming E-edge) is in X_1, that is, if the overall number of L-facts is odd.

Hence, pick $I \in \mathcal{I}$ and let ψ be a formula representation of $\mathrm{Prov}(q, I)$. Replacing the input gates for the E-facts by constant 1-gates, we obtain a formula of the same size that computes the parity function of the inputs corresponding to the L-gates, the number of which is $\lceil |I| / 2 \rceil$. We conclude as in the proof of Proposition 3.6.1. □

3.6.3 CQ$^{\neq}$ Lower Bounds on Treelike Instances

We now show that superlinear lower bounds still hold on formula-based representations of provenance when we restrict the query language to CQ$^{\neq}$.

Proposition 3.6.3. *There is a CQ$^{\neq}$ query q, and a family $\mathcal{I}$ of relational instances with treewidth 0, such that, for any $I \in \mathcal{I}$, for any Boolean formula ψ capturing* $\mathrm{Prov}(q, I)$*, we have* $|\psi| = \Omega(|I| \log \log |I|)$.

The idea is to express the threshold function over n variables, for which there is a superlinear lower bound on the formula size [Wegener 1987].

Proof. Consider the signature with a single unary predicate R, and consider the CQ$^{\neq}$ $q : \exists xy\, R(x) \wedge R(y) \wedge x \neq y$. Consider the family of instances $\mathcal{I}$ defined as $\{R(a_1), \ldots, R(a_n)\}$ for all $n \in \mathbb{N}$. Clearly, for any $I \in \mathcal{I}$, $\mathrm{Prov}(q, I)$ is the threshold function checking whether at least two of its inputs are true.

By [Wegener 1987, Chapter 8, Theorem 5.2], any formula using $\wedge$, $\vee$, $\neg$ expressing the threshold function over n variables has size $\Omega(n \log \log n)$, the desired bound. □

As CQ$^{\neq}$ queries are monotone, we can also ask for *monotone* provenance representations. From the previous section, we still have linear representations of the provenance as a *monotone* circuit. By contrast, if we restrict to *monotone* Boolean formulae, we obtain an improved lower bound:

Proposition 3.6.4. *There is a CQ$^{\neq}$ query q, and a family $\mathcal{I}$ of relational instances with treewidth 0, such that, for any $I \in \mathcal{I}$, for any* monotone *Boolean formula ψ capturing* $\mathrm{Prov}(q, I)$*, we have* $|\psi| = \Omega(|I| \log |I|)$.

Proof. We use the same proof as for Proposition 3.6.3 but relying on [Hansel 1964] (also [Wegener 1987, Chapter 8, Theorem 1.2]), which shows that, for ψ built over the monotone basis $\wedge$ and $\vee$, we have $|\psi| = \Omega(n \log n)$. □

Hence, our results in this section justify why we focus on *circuit* representations of provenance.

This section concludes our study of Boolean provenance representations for treelike instances, which we will use in the next chapter for probability evaluation in various frameworks.

Chapter 4

Applications to Probabilistic Query Evaluation

In Chapter 3, we have seen how the provenance of queries on treelike instances could be tractably computed, and how this implied the existence of other forms of lineages, such as OBDDs and d-DNNFs, for which probability evaluation is tractable.

This section focuses on the problem of probabilistic evaluation, and studies the consequences of our results for several probabilistic frameworks. We start with the simplest such framework, the *tuple-independent* (TID) instances, and study *probability evaluation* as defined in Section 2.6. We can then show, in Section 4.1:

Theorem 4.1.1. *The probabilistic evaluation problem for GSO queries and treelike TID instances is in ra-linear time: for any fixed GSO query q and constant $k \in \mathbb{N}$, given an input TID instance $J = (I, \pi)$ with $\mathrm{tw}(I) \leqslant k$, we can determine $\Pr_J(q)$ in ra-linear time in J.*

This result contrasts sharply with the picture on *arbitrary* TID instances, where many queries have #P-hard data complexity, even in the class CQ, and where a dichotomy was shown by [Dalvi and Suciu 2012] between such *unsafe* queries, and *safe* queries whose data complexity is PTIME. We show our result using the d-DNNF representations of the lineage that we can compute in linear time (Theorem 3.5.8) and the ra-tractability of probability evaluation for d-DNNF [Darwiche 2001]. As we will show in Chapter 6, there is little hope that such a result extends to larger instance classes than bounded-treewidth instances.

The rest of the chapter studies the tractability of other probabilistic frameworks, beyond TID. To do so, we introduce our own framework of uncertain and probabilistic instances in Section 4.2, namely, *cc-instances* and *pcc-instances*. They are inspired by c-instances and pc-instances [Huang, Antova, Koch, and Olteanu 2009; Green and Tannen 2006; Suciu, Olteanu, Ré, and Koch 2011], which allow complex correlations between facts; the difference is that the annotations of facts in (p)cc-instances are represented as a circuit, which makes it easier to define a notion of *treewidth* for them. We then show that GSO queries have tractable lineages on bounded-treewidth cc-instances, from which we deduce the tractability of probability evaluation.

We then use pcc-instances in Section 4.3 to deduce tractability results for the existing relational probabilistic frameworks of *pc-instances* [Huang, Antova, Koch, and Olteanu 2009; Green and Tannen 2006; Suciu, Olteanu, Ré, and Koch 2011] and *block-independent disjoint* (BID) instances [Barbará, Garcia-Molina, and Porter

1992; Ré and Suciu 2007; Suciu, Olteanu, Ré, and Koch 2011]. We first show the tractability in data complexity of GSO query evaluation on bounded-treewidth pc-instances introducing our own notion of treewidth for them:

Theorem 4.3.3. *For any fixed GSO query q and constant $k \in \mathbb{N}$, given an input pc-instance J of treewidth $\leqslant k$, we can compute $\Pr_J(q)$ in ra-linear time in J.*

We then use this result to show that the same holds for bounded-treewidth block-independent disjoint instances. The treewidth is simply defined as that of the underlying relational instance, and we impose a mild requirement that the instances are *block-sorted*.

Theorem 4.3.5. *For any fixed GSO query q and constant $k \in \mathbb{N}$, given an input block-sorted BID instance J of treewidth $\leqslant k$, we can compute $\Pr_J(q)$ in ra-linear time in J.*

Section 4.4 then moves to a non-relational probabilistic framework, namely, *probabilistic XML*, where we show the tractability of MSO query evaluation on $\mathsf{PrXML}^{\mathsf{mux,ind}}$ (reproving a result of [Cohen, Kimelfeld, and Sagiv 2009]) and on $\mathsf{PrXML}^{\mathsf{fie}}$ assuming a condition on *bounded event scopes*.

We last show two other applications of our results in the last two sections. We first re-prove in Section 4.5 that query evaluation is tractable for *inversion-free* UCQ queries on *any* input instances, and that there are constant-width OBDD representations of their lineage, a result known from [Jha and Suciu 2013]. We do this by showing that the input instances to such queries can always be *rewritten* to a bounded-rankwidth instance, in the sense of a general notion of *unfolding* which we introduce, which preserves the lineage.

Second, in Section 4.6, we adapt our results from probability evaluation to the *match counting* problem defined in Section 2.6. This allows us to give a new proof of an existing result:

Theorem 4.6.1. *[Arnborg, Lagergren, and Seese 1991] For any fixed GSO query $q(\mathbf{X}, \mathbf{x})$ with free first- and second-order variables, for any constant $k \in \mathbb{N}$, given an input instance I of width $\leqslant k$, the match counting problem for q on I can be solved in ra-linear data complexity.*

4.1 Probability Evaluation on Treelike TID Instances

We start by showing the tractability of probability evaluation on bounded-treewidth TID instances:

Theorem 4.1.1. *The probabilistic evaluation problem for GSO queries and treelike TID instances is in ra-linear time: for any fixed GSO query q and constant $k \in \mathbb{N}$, given an input TID instance $J = (I, \pi)$ with $\mathrm{tw}(I) \leqslant k$, we can determine $\Pr_J(q)$ in ra-linear time in J.*

The result follows directly from the fact that we can compute a d-DNNF representation of the lineage in linear time, using Theorem 3.5.8. It then suffices to use the known fact that probability evaluation for d-DNNFs is tractable [Darwiche 2001]:

Definition 4.1.2. Given a circuit C and a probability valuation π mapping each input in C_{inp} to a value in $[0,1]$, the *circuit probability evaluation problem* for C_{inp} and π is intuitively to determine the probability that C evaluates to 1 under π. Formally, we wish to determine, for $\mathrm{Val}(C_{\mathsf{inp}})$ the set of valuations of C_{inp}, the following probability:

$$\pi(C) := \sum_{\substack{\nu \in \mathrm{Val}(C_{\mathsf{inp}}) \\ \nu(C)=1}} \mathrm{Pr}_{C,\pi}(\nu)$$

where we define:

$$\mathrm{Pr}_{C,\pi}(\nu) := \prod_{\substack{g \in C_{\mathsf{inp}} \\ \nu(g)=1}} \pi(g) \prod_{\substack{g \in C_{\mathsf{inp}} \\ \nu(g)=0}} (1-\pi(g)). \qquad \triangleleft$$

Theorem 4.1.3 ([Darwiche 2001])**.** *The circuit probability evaluation problem for d-DNNFs is ra-linear: given a d-DNNF C and a probability valuation π of C, we can determine the probability $\pi(C)$ in ra-linear time in C and π.*

This directly implies Theorem 4.1.1. Indeed:

Proof of Theorem 4.1.1. Given a GSO query q and TID instance $J = (I, \pi)$, compute a d-DNNF representation C of $\mathrm{Prov}(q, I)$ in linear time using Theorem 3.5.8, and use Theorem 4.1.3 to compute the probability $\pi(C)$.

It is immediate to see that this indeed computes the correct probability, namely, that $\mathrm{Pr}_J(q) = \pi(C)$, because there is a bijection between valuations ν of $I = C_{\mathsf{inp}}$ and subinstances of I, which maps ν to $\nu(I)$, such that $\nu(C) = 1$ iff $\nu(I) \models q$ by definition of C capturing $\mathrm{Prov}(q, I)$. Further, this bijection preserves probability, as $\mathrm{Pr}_{C,\pi}(\nu) = \mathrm{Pr}_J(\nu(I))$ for any valuation ν of I. □

An alternative way to prove Theorem 4.1.1, which does not involve d-DNNFs, is to use message-passing techniques [Lauritzen and Spiegelhalter 1988; Huang and Darwiche 1996] to compute directly $\pi(C)$ for the bounded-treewidth circuit C. We do not give details here, and refer instead to [Amarilli, Bourhis, and Senellart 2015, Theorem D.2].

4.2 CC-Instances

TID instances are the simplest probabilistic relational framework, but they do not support *correlations* between facts. This section accordingly introduces our model of *cc-instances* and *pcc-instances*, a more expressive model in which we can adapt our bounded-treewidth construction. We will use pcc-instances to prove tractability results for *pc-instances* and *BID instances* in the next section.

4.2.1 Defining the Frameworks

The notion of cc-instances and pcc-instances is inspired by the definition of *c-instances* and *pc-instances*, which we now review:

Definition 4.2.1 ([Imieliński and Lipski 1984; Green and Tannen 2006; Huang, Antova, Koch, and Olteanu 2009])**.** A *c-instance* $J = (I, X, \psi)$ consists of:

- a relational instance I;

- a set X of Boolean variables (or *events*);
- a *labeling function* ψ mapping each fact F of I to a Boolean formula $\psi(F)$ over X called the *condition* of F.

For a valuation ν of X mapping each variable to $\{0,1\}$, the possible world $\nu(J)$ is obtained by retaining exactly the tuples whose annotation evaluates to 1 under ν, and $[\![J]\!]$ is the universe of all these possible worlds over all valuations ν. Note that different valuations may yield the same possible world.

A *pc-instance* $J = (I, X, \psi, \pi)$ is a 4-tuple where $J' = (I, X, \psi)$ is a c-instance and π is a *probability valuation* from X to $[0,1]$. The probability distribution $[\![J]\!]$ defined by J has universe $[\![J']\!]$ and probability measure $\Pr_J(I) := \sum_{\nu|\nu(J')=I} \Pr_J(\nu)$ with the product distribution on valuations:

$$\Pr_J(\nu) := \prod_{\substack{x\in X\\ \nu(x)=1}} \pi(x) \prod_{\substack{x\in X\\ \nu(x)=0}} (1-\pi(x)). \qquad \lhd$$

Our goal is to modify this definition to obtain a formalism where it is more natural to define the *treewidth* of an instance. We do so using Boolean circuits instead of Boolean formulae for the annotation. This is how we define *cc-instances* and *pcc-instances*:

Definition 4.2.2. A *cc-instance* $J = (I, C, \psi)$ consists of:

- a relational instance I;
- a Boolean circuit C, whose output gate is not relevant, and whose inputs C_{inp} are also called the *inputs* (or *events*) of J and written J_{inp};
- a *labeling function* ψ mapping each fact F of I to a gate $\psi(F)$ of J_{inp}.

For every valuation ν of J_{inp}, the possible world $\nu(J)$ is the subinstance of I that contains the facts F of I such that, seeing ν as an evaluation of C, we have $\nu(\psi(F)) = 1$. Like for c-instances, the universe $[\![J]\!]$ is the set of possible worlds of J.

A *pcc-instance* is a 4-tuple $J = (I, C, \psi, \pi)$ where $J' = (I, C, \psi)$ is a cc-instance (and $J_{\mathsf{inp}} := J'_{\mathsf{inp}}$) and $\pi : J_{\mathsf{inp}} \to [0,1]$ gives a probability to each input. As for pc-instances, the probability distribution $[\![J]\!]$ has universe $[\![J']\!]$ and probability measure $\Pr_J(I') = \sum_{\nu|\nu(J)=I'} \Pr_J(\nu)$ with the product distribution on valuations:

$$\Pr_J(\nu) = \prod_{\substack{g\in J_{\mathsf{inp}}\\ \nu(g)=1}} \pi(g) \prod_{\substack{g\in J_{\mathsf{inp}}\\ \nu(g)=0}} (1-\pi(g)). \qquad \lhd$$

It is straightforward that (p)cc-instances capture (p)c-instances:

Proposition 4.2.3. *For any (p)c-instance J, one can compute in linear time a (p)cc-instance J' whose inputs are the variables X of J, such that for any valuation ν of X, $\nu(J) = \nu(J')$ (and, for the probabilistic version, $\Pr_J(\nu) = \Pr_{J'}(\nu)$).*

Proof. We first show the result for c-instances. Create one input gate g_x per variable x, and create for each tuple a Boolean circuit that represents the formula that annotates that tuple (in linear time in the annotation of the tuple). Map the tuple to the

distinguished function gate of this circuit, and let C be the union of the circuits for each tuple. It is clear that, for any valuation ν, the possible world of J and J' is the same. The same argument applies to pc-instances, taking $\pi'_J(g_x) := \pi_J(x)$ for every variable x. □

Of course, it is also the case that (p)c-instances capture (p)cc-instances, as (p)c-instances can clearly represent any universe or probability distribution. However, the naïve translation from cc-instances to c-instances leads to a blowup, intuitively because c-instances cannot share common subexpressions between tuple annotations. We leave open the question of whether (p)cc-instances can be shown to be more compact than (p)c-instances.

Example 4.2.4. Fix $n \in \mathbb{N}$ and a universe $\mathcal{U}_n$ whose possible worlds are $\emptyset$, $\{R(a_1, a_2)\}$, $\{R(a_1, a_2), R(a_2, a_3)\}$, …, $\{R(a_1, a_2), \ldots, R(a_{n-1}, a_n)\}$. For instance, this could be a representation of the possible worlds of a chain in an uncertain XML document (as in Section 4.4), where the removal of an edge implies that all descendant edges are also removed.

It is easy to represent $\mathcal{U}_n$ as a cc-instance J with underlying instance $I := \{R(a_1, a_2), \ldots, R(a_{n-1}, a_n)\}$, where C is defined by $g_1 = g_1^{\mathsf{i}}$, and $g_j := g_{j-1} \wedge g_j^{\mathsf{i}}$ for $1 < j \leqslant n-1$, with the g_j^{i} being inputs, and $\psi(R(a_j, a_{j+1})) = g_j$. This J has size linear in n.

By contrast, it is harder to represent $\mathcal{U}_n$ as a c-instance, as the annotation of every fact "depends" on that of the previous fact in the chain. If we consider the natural encoding J' where each fact $R(a_j, a_{j+1})$ is annotated by the conjunction $e_1 \wedge \cdots \wedge e_j$, we see that J' has size quadratic in n.

We will study *treelike* (p)cc-instances, where the *treewidth* of a (p)cc-instance is defined through a straightforward *relational encoding*: we code (p)cc-instances to ordinary relational instances on an augmented signature so that the wires of the circuit and the mapping ψ are also represented.

Definition 4.2.5. Let σ be a relational signature. Let σ^+ be the signature with one relation R^+ of arity $|R| + 1$ for every relation R of σ, and with a fresh binary relation R^{W}.

The *relational encoding* I_J of a cc-instance $J = (I, C, \psi)$ over signature σ, is the σ^+ instance with domain $\mathrm{dom}(I_J) := \mathrm{dom}(I) \sqcup G$ (we assume without loss of generality that G and $\mathrm{dom}(I)$ are disjoint) that contains:

- one fact $R^+(\psi(F), \mathbf{a}))$ for every fact $F = R(\mathbf{a})$;
- one fact $R^{\mathrm{W}}(g, g')$ for every wire $(g, g') \in W$.

A *tree decomposition* of a cc-instance J is a tree decomposition of its relational encoding I_J, and the *width* $\mathrm{tw}(J)$ of J is simply $\mathrm{tw}(I_J)$. We extend these definitions from cc-instances to pcc-instances in the expected way, defining the *relational encoding* of a pcc-instance as that of its underlying cc-instance. ◁

4.2.2 Provenance Circuits

We will show, for GSO queries, that we can tractably compute provenance circuits on treelike cc-instances, so that probability evaluation for such queries on treelike

pcc-instances is ra-linear. To do this, let us first define the *provenance* of a query q on a cc-instance J. This provenance $\mathrm{Prov}(q, J)$ simply maps any valuation ν of the *inputs* of J to a Boolean value indicating whether $\nu(J) \models q$:

Definition 4.2.6. The *provenance* of a query q on a cc-instance J is the function $\mathrm{Prov}(q, J)$ mapping any valuation $\nu : J_{\mathsf{inp}} \to \{0, 1\}$ to 0 or 1 depending on whether $\nu(J) \not\models q$ or $\nu(J) \models q$.

A *provenance circuit* of q on J is a Boolean circuit C with $C_{\mathsf{inp}} = J_{\mathsf{inp}}$ that captures $\mathrm{Prov}(q, J)$. ◁

We now show that treelike cc-instances, like treelike instances (by Theorem 3.3.2), have linear-size provenance circuits:

Theorem 4.2.7. *For any fixed $k \in \mathbb{N}$ and GSO query q, for any cc-instance J such that $\mathrm{tw}(J) \leqslant k$, we can construct a provenance circuit C of q on J in time $O(|J|)$. The treewidth of C only depends on k and q (not on J).*

We prove this claim in the rest of this subsection in the following way: we rewrite any GSO query q on a cc-instance J to a GSO query q' on the *full* relational encoding I'_J of J, defined by augmenting the relational encoding as follows:

Definition 4.2.8. Let σ be a signature. Let σ^{++} be the signature obtained by completing σ^+ with fresh unary relations R^{i}, $R^{\neg}$, $R^{\wedge}$ and $R^{\vee}$ to indicate the type of a gate, and R^1 to indicate that a gate evaluates to 1.

The *full relational encoding* of a cc-instance $J = (I, C, \psi)$, where we write $C = (G, W, g_0, \mu)$, is the σ^{++}-instance I'_J obtained by adding to the relational encoding I_J the following facts:

- the fact $R^{\mu(g)}(g)$ for each $g \in G$;
- a fact $R^1(g)$ for each $g \in C_{\mathsf{inp}}$. ◁

Intuitively, we will design the query q' on I'_J to be equivalent to q except that q' additionally performs the evaluation of the Boolean annotation circuit to know which facts it can see, from the (tuple-independent) valuation of the input gates. We then see a valuation of the cc-instance J as a valuation of I'_J which indicates whether the input gates are assigned to 0 (represented by the absence of their R^1-fact) or 1 (represented by the presence of that fact). Thus, a provenance circuit for q on J can just be obtained as a provenance circuit of q' on I'_J obtained by Theorem 3.3.2, where we set all inputs to 1 except those that stand for the R^1-facts.

Hence, let us show:

Proposition 4.2.9. *For any GSO query q, there is a GSO query q' such that, for any cc-instance J with full relational encoding I'_J, for any valuation ν of J_{inp}, letting ν' be the valuation of I'_J that maps each fact $R^1(g)$ to $\nu(g)$ for each $g \in J_{\mathsf{inp}}$, and maps the other facts to 1, we have $\nu(J) \models q$ iff $\nu'(I'_J) \models q'$.*

Proof. Fix the (Boolean) GSO query q. Define the following predicate:

$$R^{\mathrm{gate}}(g) := R^{\mathsf{i}}(x) \vee R^{\wedge}(x) \vee R^{\vee}(x) \vee R^{\neg}(x)$$

Let T be a fresh second-order variable. We define a GSO query q'' by doing the following:

- Replace each σ-atom $R(\mathbf{x})$ of q by $\exists g\ R^+(g, \mathbf{x}) \wedge T(g)$, where g is a fresh variable;
- Relativize all other quantifications to apply to elements x such that $R^{\text{gate}}(x)$ does *not* hold.

  In other words, we replace $\exists x\ \varphi(\mathbf{X}, x, \mathbf{y})$ (except for the quantifications $\exists g$ that we introduced in the previous step) by $\exists x\ \neg R^{\text{gate}}(x) \wedge \varphi(x, \mathbf{x})$, replace $\exists X\ \varphi(X, \mathbf{Y}, \mathbf{x})$ by $\exists X\ (\forall x\ (x \in X) \Rightarrow \neg R^{\text{gate}}(x)) \wedge \varphi(X, \mathbf{Y}, \mathbf{x})$, and translate universal quantification in the analogous way.

We now define q' as:

$$q' : \exists T\ q''(T) \wedge q_{\text{wf}}(T)$$

where we define $q_{\text{wf}}(T)$ as follows. Intuitively, $q_{\text{wf}}(T)$ imposes that the set T is the set of the elements standing for the gates of C in I'_J that evaluate to 1, following the evaluation of C described by setting the inputs to 1 or 0 depending on whether their R^1-fact is present or absent. Formally, the definition is at follows; remember that R^{W} is the binary relation of σ^+ that codes the wires of the circuit.

$$\begin{aligned}
q_{\text{wf}}(T) &: q^{\text{i}}(T) \wedge q^{\wedge}(T) \wedge q^{\vee}(T) \wedge q^{\neg}(T) \wedge (\forall x\ T(x) \Rightarrow R^{\text{gate}}(x)) \\
q^{\text{i}}(T) &: \forall x\ R^{\text{i}}(x) \Rightarrow (T(x) \Leftrightarrow R^1(x)) \\
q^{\wedge}(T) &: \forall x\ R^{\wedge}(x) \Rightarrow (T(x) \Leftrightarrow (\forall y\ R^{\text{W}}(y, x) \Rightarrow T(y))) \\
q^{\vee}(T) &: \forall x\ R^{\vee}(x) \Rightarrow (T(x) \Leftrightarrow (\exists y\ R^{\text{W}}(y, x) \Rightarrow T(y))) \\
q^{\neg}(T) &: \forall x\ R^{\neg}(x) \Rightarrow (T(x) \Leftrightarrow (\forall y\ R^{\text{W}}(y, x) \Rightarrow \neg T(y)))
\end{aligned}$$

Consider now a cc-instance J, let C be its annotation circuit, and let ν and ν' be two valuations defined as in the proposition statement. We show that we have $\nu(J) \models q$ iff $\nu'(I'_J) \models q'$, concluding the proof. Assuming that $\nu(J) \models q$, it is clear that a satisfying assignment of T is the set of gates of C which evaluate to true under ν: this clearly satisfies $q_{\text{wf}}(T)$, and it is clear that it satisfies $q''(T)$ because $\nu(J) \models q$, so indeed $\nu'(I'_J) \models q'$. Conversely, assuming that $\nu'(I'_J) \models q'$, it is clear that the *only* possible witness for T is defined in the same manner, whence we deduce that indeed $\nu(J) \models q$. This concludes the proof. □

We are now ready to show that we can compute provenance circuits for GSO queries on bounded-treewidth cc-instances:

Proof of Theorem 4.2.7. Fix the GSO query q and treewidth bound $k \in \mathbb{N}$. Compute the GSO query q' given by Proposition 4.2.9 (this is instance-independent). Now, given a cc-instance J, letting C be its annotation circuit, compute in linear time its full relational encoding I'_J; as I'_J is just the relational encoding I_J with additional unary facts, we clearly have $\text{tw}(I'_J) = \text{tw}(I_J)$, which is by definition $\text{tw}(J)$, which is $\leqslant k$. Hence, we can use Theorem 3.3.2 to compute in linear time a provenance circuit C' of q' on I'_J. Let C'' be obtained by identifying the input gates of C' corresponding to the facts $R^1(g)$ to the gate g itself for each $g \in C$, and replacing by constant 1-gates all other inputs of C'. By the property ensured by Proposition 4.2.9, it is then clear that C'' is a provenance circuit for q on J. It is then clear that C'' captures $\text{Prov}(q, J)$, as, for any valuation ν of J and corresponding valuation ν' of I'_J, we have $\nu(J) \models q$ iff $\nu'(I'_J) \models q'$ iff $\nu'(C') = 1$ iff $\nu(C'') = 1$. □

4.2.3 Probability Evaluation

As in Section 4.1, we now now show that probability evaluation for GSO queries on treelike pcc-instances is in ra-linear time. This is simply by adapting the proof of Theorem 4.2.7, as in Theorem 3.5.8, to produce a d-DNNF representation of the provenance. As circuit probability evaluation is tractable for d-DNNFs by Theorem 4.1.3, we deduce the following:

Theorem 4.2.10. *The probability evaluation problem for GSO queries on bounded-treewidth pcc-instances has ra-linear data complexity: for any GSO query q and $k \in \mathbb{N}$, given an input pcc-instance J with $\mathrm{tw}(J) \leqslant k$, we can compute in ra-linear time in J the probability $\Pr_J(q)$ that J satisfies q.*

A similar result on probabilistic networks was shown in [Bodlaender 2012]: probability evaluation for MSO queries evaluated over bounded-treewidth probabilistic networks is tractable.

We conclude the section by noticing that Theorem 4.2.10 would *not* hold if we had defined the treewidth of (p)cc-instances as that of the underlying instance and that of the annotation circuit *taken separately*. This justifies that the treewidth of cc-instances should be defined by looking at the *joint* treewidth of the instance and circuit, as we did. Indeed, we can show:

Proposition 4.2.11. *There is a fixed CQ q such that probability evaluation for q is #P-hard on pcc-instances, even when input pcc-instances $J = (I, C, \psi, \pi)$ are restricted by imposing $\mathrm{tw}(I) = 1$ and $\mathrm{tw}(C) = 0$.*

Proof. We show a reduction from the problem of determining the probability $\pi'(F)$ of a Boolean formula F given as a monotone 2-DNF (disjunctive normal form with only positive literals, and two variables per disjunct) under a probability valuation π' mapping each variable of F to a probability in $[0, 1]$. This problem itself is indeed #P-hard by an immediate reduction from #MONOTONE-2SAT, the same problem for monotone 2-CNF formulae, which is #P-hard [Provan and Ball 1983]. Indeed, to compute the probability of a monotone 2-CNF χ, build its 2-DNF negation using de Morgan's rule, replace each literal $\neg x$ by x and change the probability of each variable from p to $1 - p$. The resulting formula χ' has probability $1 - p'$, where p' is the probability of the original formula χ, and χ' is a 2-DNF of positive literals.

Hence, consider a monotone 2-DNF formula $\chi = \bigvee_{1 \leqslant i \leqslant n}(c_i^1 \wedge c_i^2)$ and a probability valuation π' mapping each variable of χ to a probability in $[0, 1]$. We let q be the fixed CQ $\exists xyz\ A(x, y) \wedge B(y, z)$, and encode χ to a pcc-instance $J = (I, C, \psi, \pi)$:

- We set $\mathrm{dom}(I) := \bigcup_{1 \leqslant i \leqslant n}\{a_i, b_i, c_i\}$, and define $I := \bigcup_{1 \leqslant i \leqslant n}\{A(a_i, b_i), B(b_i, c_i)\}$
- The circuit C consists only of input gates x for each variable x in χ
- We define π by setting $\pi(x) := \pi'(x)$ for each variable x in χ
- We define ψ by setting $\psi(A(a_i, b_i)) := c_i^1$ and $\psi(B(b_i, c_i)) := c_i^2$.

It is clear that C has treewidth 0, as it does not contain any wire, just input gates. Further, I has treewidth 1 as it is a forest.

We now show that the reduction is correct. It is immediate that to each valuation ν of the variables of χ corresponds a valuation ν' of J_{inp} with the same probability, and

we have $\nu(\chi) = 1$ iff $c_i^1 \wedge c_i^2$ is true for some i, which happens iff both $A(a_i, b_i)$ and $B(b_i, c_i)$ are in $\nu'(I_\chi)$ for some $1 \leqslant i \leqslant n$, i.e., iff $\nu'(I) \models q$. Hence, the probability that q holds in J is exactly $\pi'(\chi)$. As the reduction is clearly in PTIME, this concludes the hardness proof. □

4.3 Applications to Probabilistic Relational Frameworks

We have shown the tractability of probability evaluation on bounded-treewidth TID instances (Theorem 4.1.1) in Section 4.1. We have introduced the richer model of *pcc-instances* in the previous section, and shown that probability evaluation is also tractable on then when we assume bounded-treewidth (Theorem 4.2.10).

In this section, we use this last result to show the tractability of probability evaluation for GSO queries on existing probabilistic database formalisms, again under bounded-treewidth assumptions. We first study *pc-instances* (Definition 4.2.1) in Section 4.3.1, and then study *BID instances* in Section 4.3.2.

4.3.1 PC-Instances

To state the bounded-treewidth tractability of pc-instances, we first need to define a notion of *treewidth* for them. Clearly it would not suffice to bound the treewidth of the underlying instance, because probabilistic evaluation is clearly already hard with a single fact with an arbitrarily complex Boolean formula as annotation. Conversely, it does not suffice to impose that the Boolean annotations are simple, as probability evaluation on TID instances is already #P-hard.

We thus introduce a notion of *treewidth* which allows us to restrict the complexity of the underlying instance, of the annotations, and of the interaction between the two, in the spirit of cc-instances:

Definition 4.3.1. Let $\sigma^{\mathrm{o}} = \sigma \cup \{\mathrm{Occ}, \mathrm{Cooc}\}$, where Occ and Cooc have arity two. From a c-instance $J = (I, X, \psi)$, assuming without loss of generality that X and $\mathrm{dom}(I)$ are disjoint, we define the *relational encoding* I_J of J as the σ^{o}-instance I' with domain $\mathrm{dom}(I') := \mathrm{dom}(I) \sqcup X$ where I' contains:

- all facts of I';
- a fact $\mathrm{Occ}(a, x)$ in I_J whenever $a \in \mathrm{dom}(I)$ occurs in a fact $F \in I$ such that $\psi(F)$ involves $x \in X$;
- a fact $\mathrm{Cooc}(x, y)$ for $x, y \in X$ whenever there is a fact F in I such that x and y both occur in $\psi(F)$.

The *treewidth* $\mathrm{tw}(J)$ of a c-instance J is the treewidth $\mathrm{tw}(I_J)$ of its relational encoding I_J. The treewidth of a pc-instance is defined analogously as that of its underlying c-instance. ◁

This notion of treewidth, through event occurrences and co-occurrences, can be connected to treewidth for (p)cc-instances, as we now show:

Proposition 4.3.2. *For any fixed k, given a (p)c-instance J of width $\leqslant k$, we can compute in linear time a (p)cc-instance J which is equivalent (has the same possible worlds, with the same probabilities if applicable) and has treewidth depending only on k and σ.*

The only technicality in this proof is that we need to first rewrite the annotations of facts in J so they have constant size. We now give the formal proof:

Proof. We first show the claim for a c-instance $J = (I, X, \psi)$.

Let us first justify that we can compute in linear time from J a cc-instance J' with the same events such that for any valuation ν, we have $\nu(J) = \nu(J')$, and the annotations of J' have size depending only on k.

Indeed, we observe that, by our assumption that $\mathrm{tw}(J) \leqslant k$, for any formula $\psi(F)$, the number of distinct events occurring in $\psi(F)$ is at most $k+1$. Indeed, there is a Cooc-clique between such events in I_J; now, as $\mathrm{tw}(I_J) \leqslant k$, by Lemma 1 of [Gavril 1974], there is no clique of $> k+1$ elements in I_J.

Now, we observe that any Boolean formula $\chi = \psi(F)$ can be rewritten, in linear time in χ, to an equivalent formula χ' whose size depends only on k. Indeed, for every valuation ν of the input events, which means at most 2^k valuations by the above, we can evaluate $\nu(\chi)$ in linear time; then we can rewrite χ to the disjunction of all valuations that satisfy it, each valuation being tested as the conjunction of the corresponding events and negation of events. So we thus produce in linear time a c-instance (I, X, ψ') which is equivalent to J and where the size of Boolean formulae in the image of ψ' depends only on k. Let us assume that J has been preprocessed in this way, so that the size of the annotations of J is bounded by a constant.

Let now $J' = (I, C, \psi')$ be the cc-instance which is the translation of the c-instance J given in the proof of Proposition 4.2.3: remember that the inputs of C are g_x for all $x \in X$, and that, for each fact $F \in I$, we create in C a circuit representation C_F of the Boolean function $\psi(F)$ on X, whose inputs are some subset of C_{inp} and whose internal gates are fresh, and we set $\psi'(F)$ to be the output gate of C_F. By the Proposition, J and J' are equivalent and J' is constructed in linear time from J'.

Consider now the relational encoding I_J of the c-instance J and a tree decomposition T of I_J of width k. We construct a tree decomposition T' of the relational encoding $I_{J'}$ of the cc-instance J' whose width depends only on k. Create T' to have the same skeleton as T, and, for each bag b in T, letting b' be the corresponding bag in T', add to $\mathrm{dom}(b')$ all elements of $\mathrm{dom}(b) \cap \mathrm{dom}(I)$, and add g_x for all $x \in \mathrm{dom}(b) \cap X$.

Now, consider each fact $F = R(\mathbf{a})$ of I. Let $\mathbf{x}$ be the set of events used in the annotation $\psi(F)$ of F in J. Note that every pair of $S = \mathbf{a} \sqcup \mathbf{x}$ co-occurs in some fact of I_J: the elements of $\mathbf{a}$ co-occur in F, the elements of $\mathbf{x}$ co-occur in a Cooc fact, and any pair of elements from $\mathbf{a}$ and $\mathbf{x}$ co-occur in some Occ fact. Hence, by Lemma 1 of [Gavril 1974] again, there is a *representative* bag $b_F \in T$ such that $S \subseteq \mathrm{dom}(b_F)$. Add $G_F \backslash C_{\mathsf{inp}}$ to the domain of the corresponding bag b'_F in T', where G_F are the gates of circuit C_F; by our assumption on J, the size $|G_F|$ depends only on k.

We check that T' thus defined is indeed a tree decomposition of the relational encoding $I_{J'}$ of J'. For the first condition, observe that any fact $R(g, \mathbf{a})$ of $I_{J'}$ that stands for a fact $F = R(\mathbf{a})$ of I is covered in the bag b'_F in T' that corresponds to the representative bag b_F in T chosen for F. Further, any fact $R^{\mathrm{W}}(g, g')$ in $I_{J'}$ that stands

for a wire of C, letting C_F be the part of C where the wire occurs, is again covered in the bag b'_F. For the second condition, we check it for elements of $\mathrm{dom}(I)$ by relying on the fact that their occurrences in T' exactly correspond to their occurrences of T, where they are a connected subtree because T is a tree decomposition. Likewise, for elements of C_{inp}, we rely on the fact that the occurrences of the corresponding element of X in T is also a connected subtree. The other elements of $\mathrm{dom}(I_{J'})$ are precisely the G_F for $F \in I$, which only occur in a single bag, so their occurrences are clearly a connected subtree.

It is now clear that the width of T' only depends on k and σ. Indeed, for the elements of $\mathrm{dom}(I)$ and C_{inp} added initially, there are at most k per bag. Now, for the elements of C added subsequently, for each fact F of I, their number depends only on k. Now, for each bag b of T, the number of distinct facts F of I such that we choose b as the representative bag b_F for F is bounded by a function that depends only on k and σ, i.e., it is bounded by $n_\sigma k^{|\sigma|}$ where n_σ is the number of relations of σ. Hence, indeed, the width of T' only depends on k and σ. This concludes the proof for c-instances.

Clearly, the same proof adapts to any pc-instance J by translating it to an equivalent pcc-instance J' using Proposition 4.2.3. Hence, the same result holds for pc-instances. □

We can now combine the above with Theorem 4.2.10, and deduce the tractability of query evaluation on bounded-treewidth pc-instances.

Theorem 4.3.3. *For any fixed GSO query q and constant $k \in \mathbb{N}$, given an input pc-instance J of treewidth $\leqslant k$, we can compute $\Pr_J(q)$ in ra-linear time in J.*

Proof. Use Proposition 4.3.2 to compute in linear time a pcc-instance J' of bounded-treewidth which is equivalent to the input pc-instance J. Now, use Theorem 4.2.10 to solve in ra-linear time the probability evaluation problem of q on J'; its answer is the same as that of the probability evaluation problem for q on J. □

4.3.2 BID instances

We now study *block-independent disjoint* (BID) instances. Following [Barbará, Garcia-Molina, and Porter 1992; Ré and Suciu 2007], we define:

Definition 4.3.4. A *BID instance* $J = (I, \pi)$ is a relational instance with each relation partitioned into *key* and *value* positions. A *key* of J is a pair $(R, \mathbf{a})$ formed of a relation R and a valuation of the key positions of R; the *matching facts* $F_1, \ldots, F_n$ of $(R, \mathbf{a})$ are the R-facts whose values on the key positions match $\mathbf{a}$: they form a *block*. We require that $\sum_i \pi(F_i) \leqslant 1$ in each block.

The semantics $[\![J]\!]$ of J is defined as follows: instances are drawn at random by choosing one or zero fact per block, independently across blocks. Within each block, all facts of the block are mutually exclusive, and each fact is kept with the probability indicated by π; so we have a non-zero probability of choosing no fact for the block iff the probabilities of the facts of the block sum up to < 1.

We define the *treewidth* $\mathrm{tw}(J)$ of a BID instance J as that of the underlying relational instance, forgetting about the probabilities.

To ensure ra-linear time complexity, we assume that BID instances are *block-sorted*, namely, they are given with facts regrouped per blocks; otherwise our bounds are PTIME rather than ra-linear as we first need to sort the facts. ◁

We can then show that GSO query evaluation is tractable on bounded-treewidth BID instances:

Theorem 4.3.5. *For any fixed GSO query q and constant $k \in \mathbb{N}$, given an input block-sorted BID instance J of treewidth $\leqslant k$, we can compute* $\Pr_J(q)$ *in ra-linear time in J.*

We again translate to a bounded-treewidth pcc-instance and use Theorem 4.2.10, which is formally stated as follows:

Lemma 4.3.6. *For any fixed $k \in \mathbb{N}$, given a BID instance J with* $\mathrm{tw}(J) \leqslant k$*, we can compute in ra-linear time an equivalent pcc-instance J' where* $\mathrm{tw}(J')$ *depends only on k and σ.*

However, the proof of this result is not so trivial: we design J' by considering a tree decomposition of J and designing a circuit to choose, for each block, a fact for the block, in the subtree of occurrences of elements for that block.

Proof of Lemma 4.3.6. Fix k and $J = (I, \pi)$. First, compute in linear time a tree decomposition T of J of width $\mathrm{tw}(J) \leqslant k$.

Without loss of generality, we can assume that the probabilities of facts within each block of J are rationals with the same denominator (if this is not the case, we normalize these probabilities in ra-linear time).

As in the proof of Lemma 3.2.6, we can assume that every fact F of J has been assigned in linear time to a bag $\beta(F)$ of T where it is covered (i.e., writing $F = R(\mathbf{a})$, we have $\mathbf{a} \subseteq \mathrm{dom}(\beta(F))$). Actually, still in the spirit of the proof of Lemma 3.2.6, we can modify the decomposition T by copying nodes to create chains, so that we can assume that at most one fact is assigned to each bag: in other words, β is an injective mapping. This preprocessing can be performed in linear time.

We construct the pcc-instance $J' = (I, C, \psi, \pi')$ by building C, ψ, and π', and build a tree decomposition T' of the relational encoding $I_{J'}$ of J'. We initialize T' as a copy of T, so it is a tree decomposition of I. We will add the gates of C to T' to turn it into a tree decomposition of J'.

Let $\mathcal{K}$ be the set of the keys of J, namely, the set of pairs $(R, \mathbf{a})$ of a relation symbol R and a tuple $\mathbf{a}$ that is a key in J for that relation. We write $I_{R,\mathbf{a}}$ to refer to the block of the facts of J that match the key $(R, \mathbf{a})$, and we write $|J_{R,\mathbf{a}}|$ to denote the size of $I_{R,\mathbf{a}}$ *plus* the size of $\pi_{|I_{R,\mathbf{a}}}$: i.e., the size of representing both the facts of $(R, \mathbf{a})$ and the associated probabilities. Hence, we have $\sum_{(R,\mathbf{a})\in\mathcal{K}} |J_{R,\mathbf{a}}| = |J|$, the size of the original instance.

Now, for every $(R, \mathbf{a}) \in \mathcal{K}$, consider the subset of the bags $T_\mathbf{a}$ of T that cover $\mathbf{a}$. It is clear that $T_\mathbf{a}$ is a connected subtree of T, as it is the intersection for every element $a \in \mathbf{a}$ of the occurrence subtree T_a of a, each T_a being a connected subtree because T is a tree decomposition; further, $T_\mathbf{a}$ is not empty because the elements of $\mathbf{a}$ must occur together in some fact of J, so they also do in some bag of T. What is more, we can precompute in linear time the roots of all the $T_\mathbf{a}$ (by the same precomputation as in the proof of Lemma 3.2.6). It is also clear that $\sum_{(R,\mathbf{a})\in\mathcal{K}} |T_\mathbf{a}|$ is of size linear in $|J|$, as, for fixed σ and k, each bag of T can only occur in a constant number of $T_\mathbf{a}$.

So we prove the result in the following way: for each $(R, \mathbf{a}) \in \mathcal{K}$, we compute in time $O(|J_{R,\mathbf{a}}| + |T_\mathbf{a}|)$ a circuit $C_{R,\mathbf{a}}$ to annotate the facts of $I_{R,\mathbf{a}}$ in J', and we add the gates of $C_{R,\mathbf{a}}$ to T' to obtain a tree decomposition of J' as constructed so

far, making sure that we add only a constant number of gates to each bag, and only to bags in T' that correspond to bags in $T_{\mathbf{a}}$. If we can manage this for every $(R, \mathbf{a}) \in \mathcal{K}$, then the result follows, as we can process the blocks in J in order (as they are provided); our final pcc-instance will have width that is still constant (for each bag of T can only occur in a constant number of $T_{\mathbf{a}}$, so the width increase in T' relative to T only depends on k and σ); and by the arguments about the sizes of the sums, the overall running time of the algorithm is linear in J.

So in what follows we fix $(R, \mathbf{a}) \in \mathcal{K}$ and describe the construction of $C_{R,\mathbf{a}}$ and how we complete T' for $C_{R,\mathbf{a}}$.

Using our preprocessed table to find the root of $T_{\mathbf{a}}$, we can identify its nodes by going over $T_{\mathbf{a}}$ top-down, in time linear in $T_{\mathbf{a}}$. We now notice that for every fact $F = R(\mathbf{a}, \mathbf{v})$ of $I_{R,\mathbf{a}}$, the bag $\beta(F)$ covers F so it must be in $T_{\mathbf{a}}$. We write $\beta_{R,\mathbf{a}}$ for the restriction of the function β to the facts of $J_{R,\mathbf{a}}$.

We now say that a bag $b \in T_{\mathbf{a}}$ is an *interesting bag* when it is either in the image of $\beta_{R,\mathbf{a}}$ or when it is a lowest common ancestor of some subset of bags that are in the image of $\beta_{R,\mathbf{a}}$. We now observe that the number of interesting bags of $T_{\mathbf{a}}$ is linear in the number of facts of $J_{R,\mathbf{a}}$; indeed, the interesting bags form the internal nodes and leaves of a binary tree whose leaves must all be in the image of $\beta_{R,\mathbf{a}}$, so the number of leaves is at most the number of facts of $J_{R,\mathbf{a}}$, so the total number of nodes in the tree is linear in the number of leaves.

We now define a weight function w on $T_{\mathbf{a}}$ by setting $\mathrm{w}(b) := 0$ if b is not in the image of $\beta_{R,\mathbf{a}}$ and otherwise, letting F be the preimage of b in $\beta_{R,\mathbf{a}}$ (remember that β is injective), setting $\mathrm{w}(b) := \pi(F)$. We then define bottom-up a cumulative weight function w′ on all $b \in T_{\mathbf{a}}$ by setting $\mathrm{w}'(b)$ to be $\mathrm{w}(b)$, plus $\mathrm{w}'(L(b))$ if $L(b) \in T_{\mathbf{a}}$, plus $\mathrm{w}'(R(b))$ if $R(b) \in T_{\mathbf{a}}$. For notational convenience we also extend w′ by $\mathrm{w}'(b) := 0$ for any b such that $b \notin T_{\mathbf{a}}$ or b does not exist. Clearly, by definition, for b_{r} the root of $T_{\mathbf{a}}$ $\mathrm{w}'(b_{\mathrm{r}})$ is the total probability of the facts in the block $I_{R,\mathbf{a}}$.

Observe now that for any non-interesting bag b, we can represent $\mathrm{w}(b)$ and $\mathrm{w}'(b)$ either as 0 or as a pointer to some $\mathrm{w}(b')$ or $\mathrm{w}'(b')$ for an interesting bag b'. Indeed, if b is non-interesting then we must have $\mathrm{w}(b) = 0$. Now we show that if b has a topmost interesting descendant b' then it is unique: indeed, the lowest common ancestor of two interesting descendants of b is a descendant of b and it is also interesting, so there is a unique topmost one. Now this means that either there is no interesting descendant b', and $\mathrm{w}'(b) = 0$, or a topmost interesting descendent b' does exist, and all descendants of b that are in the image of $\beta_{R,\mathbf{a}}$ are descendants of b', so that $\mathrm{w}'(b) = \mathrm{w}'(b')$ and we can just make $\mathrm{w}'(b)$ a pointer to $\mathrm{w}'(b')$.

Now this justifies that we can compute $\mathrm{w}(b)$ and $\mathrm{w}'(b)$ for every $b \in T_{\mathbf{a}}$, in a bottom-up fashion, in time $O(|T_{\mathbf{a}}| + |J_{R,\mathbf{a}}|)$: observe that we are working on rationals with the same denominator, so the sums that we perform are sums of integers, whose size always remains less than the common denominator. As the number of interesting bags is linear in the number of facts of $I_{R,\mathbf{a}}$, and are the interesting bags are the only bags for which a value needs to be written, which is a value whose size is that of the probability of the corresponding fact in $I_{R,\mathbf{a}}$, then the computation is performed in time $O(|T_{\mathbf{a}}| + |J_{R,\mathbf{a}}|)$ overall.

We now explain how to create the circuit $C_{R,\mathbf{a}}$ with the correct probabilities, using the w and w′ functions. For each bag $b \in T_{\mathbf{a}}$, we create a gate g^{i}_b; for the root bag b_{r} it is an input gate with probability $\pi'(g^{\mathsf{i}}_{b_{\mathrm{r}}}) := \mathrm{w}'(b_{\mathrm{r}})$; for other bags it is a gate

whose value is defined by the parent bag. Intuitively, g_b^{i} describes whether to choose a fact from $J_{R,\mathbf{a}}$ within the subtree rooted at b.

For every interesting bag b, writing $\mathrm{w}'(b) = k'/d$ and $\mathrm{w}(b) = k/d$ with d the common denominator for the block $I_{R,\mathbf{a}}$, create one input gate g_b^{h} with probability $\pi'(g_b^{\mathrm{h}}) := \frac{\mathrm{w}(b)}{\mathrm{w}'(b)} = k/k'$, and one gate $g_b^{\mathrm{h}\wedge}$ which is the AND of g_b^{h} and g_b^{i}. Intuitively, the gate $g_b^{\mathrm{h}\wedge}$ describes whether to generate the fact F with $\beta(F) = b$, if any. If there is such a fact F, set $\psi(F) := g_b^{\mathrm{h}\wedge}$. Now if $\mathrm{w}'(b) > \mathrm{w}(b)$ (intuitively: there is still the possibility to generate fact at child nodes), we create one input gate $g_b^{\leftrightarrow}$ which has probability $\pi'(g_b^{\leftrightarrow}) := \frac{\mathrm{w}'(L(b))}{\mathrm{w}'(b)-\mathrm{w}(b)}$. Once again, this probability simplifies to a rational whose numerator and denominator are $< d$. We create a gate g_b^{L} to be $g_b^{\mathrm{i}} \wedge \neg g_b^{\mathrm{h}} \wedge g_b^{\leftrightarrow}$ (creating a constant number of intermediate gates as necessary), and g_b^{R} to be $g_b^{\mathrm{i}} \wedge \neg g_b^{\mathrm{h}} \wedge \neg g_b^{\leftrightarrow}$, and we set $g_{L(b)}^{\mathrm{i}}$ to be a gate equal to g_b^{L} (e.g., an AND-gate whose sole input is g_b^{L}) if $L(b)$ exists and is in $T_{\mathbf{a}}$, and set likewise $g_{R(b)}^{\mathrm{i}}$ to be equal to g_b^{R} if $R(b)$ exists and is in $T_{\mathbf{a}}$.

For every non-interesting bag b, if $L(b)$ is interesting or has an interesting descendent we just set $g_{L(b)}^{\mathrm{i}}$ to be equal to g_b^{i} and set $g_{R(b)}^{\mathrm{i}}$ to be 0. Otherwise, we set $g_{R(b)}^{\mathrm{i}}$ to be equal to g_b^{i} and set $g_{L(b)}^{\mathrm{i}}$ to be 0. Remember that only one of $L(b)$ and $R(b)$ is interesting or has an interesting descendent, as otherwise b is the lowest common ancestors of interesting bags and thus must be interesting itself.

We now observe that by construction the resulting circuit has a tree decomposition that is compatible with $T_{\mathbf{a}}$, so that we can add its gates to the nodes of T' corresponding to $T_{\mathbf{a}}$, adding only constantly many nodes per bag, as required. It is also easy to see that the circuit gives the correct distribution on the facts of $J_{R,\mathbf{a}}$, with the following invariant: for any bag $b \in T_{\mathbf{a}}$, the probability that g_b^{i} is 1 is $\mathrm{w}'(b)$, and $g_b^{\mathrm{h}\wedge}$, g_b^{L} and g_b^{R} are either all 0 if g_b^{i} is 0 or, if g_b^{i} is 1, exactly one is true and they respectively have marginal probabilities $\mathrm{w}(b)$, $\mathrm{w}'(L(b))$, and $\mathrm{w}'(R(b))$. Now the circuit construction is once again in time $O(|J_{R,\mathbf{a}}| + |T_{\mathbf{a}}|)$, noting that interesting nodes are the only nodes where numbers need to be computed and written; and we have performed the entire computation in time $O(|J_{R,\mathbf{a}}| + |T_{\mathbf{a}}|)$, so the overall result is proven. □

Hence, we have shown Lemma 4.3.6, which, combined with Theorem 4.2.10, allows us to prove Theorem 4.3.5. This concludes our study of relational probabilistic frameworks.

4.4 Applications to Probabilistic XML

We now turn to the framework of *probabilistic XML*, a non-relational probabilistic framework to represent probability distributions on tree-shaped documents. Specifically, we use the PrXML framework [Kimelfeld and Senellart 2013], which allows various *probabilistic* nodes in XML documents, and thus contains diverse fragments of varying expressiveness.

We study how our tractability results for pcc-instances imply tractability results on PrXML documents. Our first result concerns the $\mathsf{PrXML}^{\mathsf{mux,ind}}$ fragment, for which probabilistic choices are *local*, i.e., they are performed independently at various nodes of the document. In this context, as the underlying structure is a tree and thus has bounded-treewidth, we can show that probability evaluation for MSO over trees is

always ra-linear. This re-proves a result that had already been shown in [Cohen, Kimelfeld, and Sagiv 2009] using methods specific to probabilistic XML.

Theorem 4.4.21. *[Cohen, Kimelfeld, and Sagiv 2009] MSO query evaluation on* $\mathsf{PrXML}^{\mathsf{mux,ind}}$ *has ra-linear data complexity.*

Our next result applies to the more expressive $\mathsf{PrXML}^{\mathsf{fie}}$ fragment, which allows probabilistic nodes labeled with arbitrary *formulae of independent events* (hence the name of fie). In this language, probability evaluation is not always tractable, even though the XML structure is a tree: the probabilistic choices are no longer *local* because the same events can be reused at arbitrary places in the document. What we show is that query evaluation is tractable assuming that the number of events to *remember* at any point in the tree is bounded by a constant.

Intuitively, we define the *scope* of event x in a document D as the smallest subtree in the left-child-right-sibling encoding of D which covers the nodes whose parent edge mentions x: intuitively, these are the places where we must remember x. The *scope size* of a node n of D is then the number of events with n in their scope: intuitively, this is the number of events to remember at n. We can then show:

Theorem 4.4.16. *For any fixed* $k \in \mathbb{N}$*, MSO query evaluation on* $\mathsf{PrXML}^{\mathsf{fie}}$ *documents with scopes assumed to have size* $\leqslant k$ *has ra-linear data complexity.*

We will first prove the result on scopes (Theorem 4.4.16) and then use it to prove the result on local models (Theorem 4.4.21). As the results are proven by translating PrXML documents to pcc-instances (via pc-instances), we must first give some prerequisites on relational translations of XML documents (Section 4.4.1), and extend them to translate PrXML documents to pc-instances (Section 4.4.2). We then show Theorem 4.4.16 in Section 4.4.3, and Theorem 4.4.21 in Section 4.4.4.

4.4.1 Relational Encodings of XML Documents

We first describe XML documents and their connections to relational models.

Definition 4.4.1. An *XML document* with *label set* Λ (or Λ-*document*) is an *unranked*, rooted and ordered Λ-tree. ◁

We always assume that the label set Λ is fixed (not provided as input). As XML documents are unranked, it is often more convenient to manipulate their binary left-child-right-sibling representation:

Definition 4.4.2. The *left-child-right-sibling* (LCRS) representation of a Λ-document T is the *non-full* Λ-tree T' defined as follows: each node n of T whose children are the ordered sequence of siblings $n_1, \ldots, n_k$ is encoded as the node n' with $L(n') = n'_1$, $R(n'_1) = n'_2$, ..., $R(n'_{k-1}) = n'_k$, where each n'_i is the encoding of n_i. ◁

We now define how XML documents can be encoded to relational instances.

Definition 4.4.3. Given a Λ-document D, let σ_Λ be the relational signature with two binary predicates FC and NS (for "first child" and "next sibling"), and unary predicates P_λ for every $\lambda \in \Lambda$. The *relational encoding* I_D of D is the σ_Λ-instance whose domain $\mathrm{dom}(I_D)$ is exactly the set of nodes of D, such that:

- for any consecutive siblings (n, n'), $NS(n, n')$ holds;
- for every pair (n, n') of a node $n \in D$ and its first child $n' \in D$ following sibling order, $FC(n, n')$ holds;
- for every node $n \in D$, the fact $P_{\lambda(n)}(n)$ holds. ◁

Lemma 4.4.4. *The relational encoding I_D of an XML document D has treewidth* 1 *and can be computed in linear time.*

Proof. Immediate: the relational encoding is clearly computable in linear time and there is a width-1 tree decomposition of the relational encoding that has same skeleton as the LCRS representation of the XML document. □

We now define formally the language of MSO queries on XML documents [Neven and Schwentick 2002], and show that it can be easily translated to MSO queries on the relational encoding:

Definition 4.4.5. An *MSO query* on XML documents is a MSO formula where first-order variables refer to nodes and where atoms are:

- $\lambda(x)$, meaning that x has label λ;
- $x \to y$, meaning that x is the parent of y;
- $x < y$, meaning that x and y are siblings and x comes before y. ◁

Lemma 4.4.6. *For any MSO query q on Λ-documents, one can compute in linear time an MSO query q' on σ_Λ such that for any Λ-document D, we have $D \models q$ iff $I_D \models q'$.*

Proof. We add a constant overhead to q by defining the predicates $\lambda(x)$ for $\lambda \in \Lambda$ as $P_\lambda(x)$, the predicate $x < y$ to be the transitive closure of NS, namely:

$$x < y := \neg(x = y) \wedge \forall S\ (x \in S \wedge (\forall z z'\ (z \in S \wedge NS(z, z')) \Rightarrow z' \in S) \Rightarrow y \in S)$$

and the predicate $x \to y$ to be defined as follows, where $z < y$ is encoded as above:

$$x \to y := \exists z\ FC(x, z) \wedge (z = y \vee z < y)$$

It is clear that the semantics of those atoms on I_D match that of the corresponding atoms on D, so that a straightforward structural induction on the formula shows that the resulting query q' satisfies the desired properties. □

As we will be working with probabilistic XML documents, where we can choose to discard nodes, it is helpful to define a looser notion of relational encoding:

Definition 4.4.7. Given label set Λ and letting $\Lambda' := \Lambda \sqcup \{\bot, \mathrm{det}\}$, we say that a Λ'-document D is a *sparse representation* of a Λ-document D' if the root of D is labeled with an element of Λ, and the XML document obtained from D by removing every $\bot$ node and their descendants, and replacing every det node by the collection of its children, in order, is exactly D'.

We say that a $\sigma_{\Lambda \sqcup \{\mathrm{det}, \bot\}}$ instance I is a *weak relational encoding* of an XML document D with label set Λ if there exists a sparse representation D' of D such that I is isomorphic to the relational encoding $I_{D'}$ of D'. ◁

We show that we can still rewrite queries following this looser notion of encoding, and strengthen Lemma 4.4.6:

Proposition 4.4.8. *For any MSO query q on Λ-documents, one can compute in linear time an MSO query q' on $\sigma_{\Lambda \sqcup \{\mathrm{det}, \bot\}}$ such that for any Λ-document D:*

Forward: *If $D \models q$ then $I \models q'$ for any weak relational encoding I of D.*

Backward: *If $D \not\models q$ then $I \not\models q'$ for any weak relational encoding I of D.*

Proof. We show that any MSO query q on Λ-documents can be translated in linear time to a MSO query q' on Λ'-documents, where $\Lambda' := \Lambda \sqcup \{\mathrm{det}, \bot\}$, such that for any Λ-document D:

Forward: If $D \models q$ then $D' \models q'$ for any sparse representation D' of D.

Backward: If $D \not\models q$ then $D' \not\models q'$ for any sparse representation D' of D.

The claimed result then follows by Lemma 4.4.6.

To explain how to design q', consider a document D and a sparse representation D' of D. Define a mapping f from D to D' witnessing this: f maps each node n of D to the node $f(n)$ of D' that will be n when we evaluate D' following Definition 4.4.7. Let us consider a node $n \in D$ with children $n_1, \ldots, n_k$ in order, and determine what is the relationship between $f(n)$ and the $f(n_i)$ in D'.

We say that a node in a Λ'-tree is *regular* if its label is in Λ. It is straightforward to observe that $f(n)$ is regular and the $f(n_i)$ are topmost regular descendants of $f(n)$ in D'. Further, for $i < j$, there is some node n' in D' (intuitively, the lowest common ancestor of n_i and n_j, which is a descendant of $f(n)$, possibly $f(n)$ itself) such that n' is both an ancestor of $f(n_i)$ and $f(n_j)$, n' is a descendant of $f(n)$, and n' has two children n'_1 and n'_2 such that $f(n_i)$ is a descendant of n'_1 (maybe $f(n_i) = n'_1$), $f(n_j)$ is a descendant of n'_2 (maybe $f(n_j) = n'_2$), and $n'_1 < n'_2$ in D'. Note that n', n'_1 and n'_2 are not necessarily regular nodes of D but can be det nodes. In addition, no $\bot$ node can be traversed in any of the ancestor–descendant chains discussed in this paragraph.

It is clear that we can define MSO predicates $\to'$ and $<'$ in D' following these informal definitions (and not depending on D or D'), from the predicates $\to$, $<$ and $\lambda(\cdot)$ on D', such that for every D and sparse encoding D' of D, for every nodes $n, n' \in D$, we have $n \to n'$ in D iff n and n' are regular nodes of D' and $f(n) \to f(n')$ in D', and likewise for $<$. Last, it is clear that the predicates $\lambda(\cdot)$ of D can be encoded directly to the same predicates in D'. □

This will prove useful when dealing with the possible worlds of probabilistic XML documents (or relational representations thereof).

4.4.2 Probabilistic XML

We now introduce probabilistic XML documents. We start by $\mathsf{PrXML}^{\mathsf{fie}}$, i.e., PrXML with formulae of independent events.

Definition 4.4.9. A $\mathsf{PrXML}^{\mathsf{fie}}$ probabilistic XML document $D = (D', X, \pi)$ over alphabet Λ is a triple of:

- a set X of Boolean events;
- a $(\Lambda \sqcup \{\mathsf{fie}\})$-document D' where edges from fie nodes to their children are labeled with a propositional formula over X;
- a probability valuation π mapping each $x \in X$ used in D to an independent probability $\pi(x) \in [0, 1]$ of being true.

We require that the root node of D has a label in Λ.

The semantics $[\![D]\!]$ of D is obtained by extending π to a probability distribution on valuations ν of X as usual, and defining $\nu(D)$ for ν to be D' where all fie nodes are replaced by the collection of their children with edge annotation Φ such that $\nu(\Phi) = 1$. The other children of fie nodes, as well as their descendants, are discarded. ◁

We now define the probability evaluation problem on $\mathsf{PrXML}^{\mathsf{fie}}$ documents:

Definition 4.4.10. The *probability evaluation problem* for a MSO query q over Λ-trees and a $\mathsf{PrXML}^{\mathsf{fie}}$ document over Λ is to determine the total probability in $[\![D]\!]$ of the possible worlds of D that satisfy q; we study the *data complexity* of this problem, i.e., its complexity as a function of D. ◁

We reduce probability evaluation for $\mathsf{PrXML}^{\mathsf{fie}}$ documents to probability evaluation in the relational setting, by encoding $\mathsf{PrXML}^{\mathsf{fie}}$ documents to pc-instances:

Definition 4.4.11. The *pc-encoding* of a $\mathsf{PrXML}^{\mathsf{fie}}$ document $D = (D', X, \pi)$ over alphabet Λ is the pc-instance $J_D = (I'_{D'}, X, \psi, \pi')$ with same events, $\pi' = \pi$, and:

- $I'_{D'}$ is the relational encoding $I_{D'}$ of D' seen as a Λ'-document for $\Lambda' := \Lambda \sqcup \{\mathsf{fie}\}$, except that P_{fie}-facts are replaced by P_{det}-facts, and one fact $P_\perp(n)$ is added for all $n \in D'$; so $I'_{D'}$ is a $\sigma_{\Lambda \sqcup \{\perp, \mathrm{det}\}}$-instance.
- for any *NS*- or *FC*-fact F in $I'_{D'}$, we set $\psi(F) := 1$.
- for any $P_\lambda(n)$-fact F in $I'_{D'}$ with $\lambda \neq \perp$, we set $\psi(F)$ to be the annotation χ of the edge from the parent of n to n, if χ exists, and set $\psi(F) := 1$ otherwise.
- for any $P_\perp(n)$-fact F in $I'_{D'}$, we set $\psi(F) := \neg\psi(F')$, where F' is the fact $P_\lambda(n)$ with $\lambda \neq \perp$ that must exist in $I'_{D'}$. ◁

Observe that in this definition of pc-encoding, the possible worlds of J_D are *not* relational encodings of the possible worlds of D; intuitively, they are *weak* relational encodings. For instance, the fie nodes are retained (as det nodes), and *FC*- and *NS*-facts are always retained even if the corresponding nodes are dropped (i.e., annotated by $\perp$). Intuitively again, having exactly the right facts in all possible worlds would lead to a quadratic number of possible *NS*-facts in long sequences of siblings, which is why we restrict to weak encodings (and rely on the query to perform the required simplifications).

We now show how MSO queries can be translated from the setting of Λ-documents to the relational setting in a way that preserves satisfaction of queries between valuations of $\mathsf{PrXML}^{\mathsf{fie}}$ documents and their pc-encodings:

Proposition 4.4.12. *For any MSO query q on Λ-documents, letting $\Lambda' := \Lambda \sqcup \{\mathrm{det}, \perp\}$, one can compute in linear time an MSO query q' on $\sigma_{\Lambda'}$-instances such that for any $\mathsf{PrXML}^{\mathsf{fie}}$ XML document D, for any valuation ν of D, we have $\nu(D) \models q$ iff $\nu(J_D) \models q'$.*

Proof. We prove that for any valuation ν of D, we have that $\nu(J_D)$ is a weak encoding of $\nu(D)$. The result then follows by Proposition 4.4.8, as we can just use q' as defined by that proposition.

Let ν be a valuation of D. Let D' be the $(\Lambda \sqcup \{\text{det}, \bot\})$-document obtained from D by replacing all nodes not kept in $\nu(D)$ by $\bot$-nodes, relabeling all fie-nodes to be det-nodes, and removing all edge annotations. It is then clear that D' is a sparse representation of $\nu(D)$ and that the relational encoding $I_{D'}$ of D' is isomorphic to $\nu(J_D)$. Hence, $\nu(J_D)$ is a weak encoding of $\nu(D)$, which concludes the proof. □

We have now introduced all the required preliminaries to study our tractability conditions on PrXML documents.

4.4.3 Tractability for PrXML$^{\text{fie}}$ Assuming Bounded Scopes

We cannot hope that the pc-encoding of a PrXML$^{\text{fie}}$ document always has constant treewidth, for it is known that the probability evaluation problem for MSO queries on arbitrary PrXML$^{\text{fie}}$ documents is #P-hard [Kimelfeld, Kosharovsky, and Sagiv 2008, Theorem 5.2].

However, a clear case where probability evaluation is tractable on PrXML$^{\text{fie}}$ documents D is when the pc-encoding of D has bounded treewidth (as defined in Definition 4.3.1). Indeed, Proposition 4.4.12 and Theorem 4.3.3 clearly imply the following:

Corollary 4.4.13. *For* PrXML$^{\text{fie}}$ *documents with bounded-treewidth pc-encoding, the MSO probability evaluation problem can be solved in ra-linear time data complexity.*

The notion of *bounded scopes* that we will define is a sufficient condition to ensure this. We give its formal definition:

Definition 4.4.14. Consider a PrXML$^{\text{fie}}$ document (D, X, π). We say that an event $x \in X$ *occurs* at a node n of D if x occurs in the annotation of the edge from the parent of n to n. Let D' be the LCRS representation of D, and define for every $x \in X$ the smallest connected subtree D'_x of D' that covers all nodes n of D' such that x occurs at n in D. The *event scope* $S(n)$ of a node $n \in D'$ is $\{x \in X \mid n \in D'_x\}$. The *event scope width* of D is $w_{\text{s}}(D) := \max_{n \in D'} |S(n)|$. ◁

We are now ready to prove the result on XML element scopes. The key point is that the pc-encoding J_D of bounded-scope PrXML$^{\text{fie}}$ documents has bounded treewidth, as is witnessed by the tree decomposition of J_D that follows the structure of D' but adds the additional events as described by the scopes.

Proposition 4.4.15. *For any* PrXML$^{\text{fie}}$ *document D, we have* $\text{tw}(J_D) \leqslant w_{\text{s}}(D) + 1$.

Proof. Write $J_D = (I, X, \psi, \pi)$, and let us define a tree decomposition T of J_D. Start by the tree decomposition T of I that is isomorphic to the LCRS encoding D' of D: the root node n_{r} of D' is coded to a bag $b_{n_{\text{r}}}$ containing $\{n_{\text{r}}\}$, and each node n of the LCRS encoding with parent n' is coded to a bag b_n containing $\{n', n\}$. It is clear that T so far is such that, for any node n of D, the occurrences of n in T are connected subtrees: each n occurs precisely in b_n and $b_{n'}$ for all children n' of n in D.

We now add to T, for each bag b_n corresponding to a node n, the events of $S(n)$. It is clear that T is of the prescribed width, and that the occurrences of all events are

connected subtrees, because scopes are defined as connected subtrees of D' containing the events.

We now argue that T is a tree decomposition of the relational encoding of the pc-instance J_D, but this is easily seen: T covers all *NS*- and *FC*- facts represented in J_D, and covers all occurrences and co-occurrences by construction of the scopes. □

Thanks to Corollary 4.4.13, we conclude our tractability result on $\mathsf{PrXML}^{\mathsf{fie}}$ documents:

Theorem 4.4.16. *For any fixed $k \in \mathbb{N}$, MSO query evaluation on $\mathsf{PrXML}^{\mathsf{fie}}$ documents with scopes assumed to have size $\leqslant k$ has ra-linear data complexity.*

4.4.4 Tractability of $\mathsf{PrXML}^{\mathsf{mux,ind}}$

We now introduce the definitions and proofs for the local model, $\mathsf{PrXML}^{\mathsf{mux,ind}}$.

Definition 4.4.17. A $\mathsf{PrXML}^{\mathsf{mux,ind}}$ probabilistic document is an XML document D over $\Lambda \sqcup \{\mathsf{ind}, \mathsf{mux}\}$, where edges from ind and mux nodes to their children are labeled with a probability in $[0, 1]$, the annotations of outgoing edges of every mux node summing to $\leqslant 1$. We require the root of D to have label in Λ.

The semantics $[\![D]\!]$ of D is defined as follows:

- for every ind node, decide to keep or discard each child according to the indicated probability, and replace the node by the (possibly empty) collection of its kept children;
- for every mux node, choose one child node to keep according to the indicated probabilities (possibly keep no node if they sum to < 1), and replace the mux node by the chosen child (or discard it if no child was chosen).

All probabilistic choices are performed independently. When a node is discarded, so are its descendants.

We define the *probability evaluation problem* for MSO queries on $\mathsf{PrXML}^{\mathsf{mux,ind}}$ documents analogously to Definition 4.4.10. ◁

Our goal is to show the tractability of probability evaluation on arbitrary $\mathsf{PrXML}^{\mathsf{mux,ind}}$-documents, thanks to the fact that probabilistic choices are "local" and follow the tree structure. To do this, we first show that we can rewrite $\mathsf{PrXML}^{\mathsf{mux,ind}}$ documents to a simpler form:

Definition 4.4.18. We say that a $\mathsf{PrXML}^{\mathsf{mux,ind}}$ is in *binary form* if it is a full binary tree, and the sum of the outgoing probabilities of every mux node is equal to 1.

Two $\mathsf{PrXML}^{\mathsf{mux,ind}}$ documents D_1 and D_2 are *equivalent* if for every XML document D, we have $\Pr_{D_1}(D) = \Pr_{D_2}(D)$. ◁

Lemma 4.4.19. *From any $\mathsf{PrXML}^{\mathsf{mux,ind}}$ document D, we can compute in ra-linear time in D an equivalent $\mathsf{PrXML}^{\mathsf{mux,ind}}$ document D' which is in binary form.*

Proof. In this proof, for brevity, we use det nodes to refer to ind nodes whose child edges are all annotated with probability 1.

We rewrite each mux node n of D whose outgoing probabilities sum up to < 1 by adding a single det child to n whose parent edge carries the remaining probability for n. This operation is in ra-linear time.

Next, add det nodes to rewrite the children of regular and ind nodes to a chain so that all regular and ind nodes have at most 2 children. This only causes a constant-factor blowup of the document, so it is in linear time.

Next, rewrite mux nodes with more than two children to a hierarchy of mux nodes in the obvious way: considering a mux node n with k children $n_1, \ldots, n_k$ and probabilities $p_1, \ldots, p_k$ summing to 1, we replace n by a hierarchy $n'_1, \ldots, n'_{k-1}$ of mux nodes: the children of each n'_i is n_i with probability $\frac{p_i}{\sum_{j<i} p_j}$ and n'_{i+1} with probability $1 - \frac{p_i}{\sum_{j<i} p_j}$; except for n'_{k-1} whose children are n_{k-1} and n_k (with these same probabilities). This operation can be performed in ra-linear time.

Now, replace mux nodes with < 2 children by ind nodes (the probabilities are unchanged).

Last, add det children to nodes so that the degree of every node is either 2 or 0.

It is thus clear that the process that we described can be performed in ra-linear time overall, and that the resulting document is in binary form; equivalence has been maintained through all steps. □

Now, we can show:

Proposition 4.4.20. *For any* $\mathsf{PrXML}^{\mathsf{mux,ind}}$ *document* D *in binary form, one can compute in linear time an equivalent* $\mathsf{PrXML}^{\mathsf{fie}}$ *document whose scopes have size* $\leqslant 1$.

Proof. For every ind node n with two children n_1 and n_2 having probabilities p_1 and p_2, introduce two fresh events $x_n^{\mathsf{ind},1}$ and $x_n^{\mathsf{ind},2}$ with probabilities p_1 and p_2, and replace n by a fie node so that its first and second outgoing edges are annotated with $x_n^{\mathsf{ind},1}$ and $x_n^{\mathsf{ind},2}$.

Likewise, for every mux node n with two children n_1 and n_2 with probabilities p and $1-p$, introduce a fresh event x_n^{mux} with probability p and replace n by a fie node so that its first and second outgoing edges are annotated with x_n^{mux} and $\neg x_n^{\mathsf{mux}}$.

It is immediate that the resulting document D' is equivalent to D. Now, consider the scope of any node of this document. Only one event occurs at this node, and the only events that occur more than one time in the document occur exactly twice, on the edges of two direct sibling nodes, so they never occur in the scope of any other node. Hence all scopes in D have size $\leqslant 1$. □

From this, we deduce the tractability of probability evaluation for MSO queries on $\mathsf{PrXML}^{\mathsf{mux,ind}}$, as was already shown in [Cohen, Kimelfeld, and Sagiv 2009]:

Theorem 4.4.21. *[Cohen, Kimelfeld, and Sagiv 2009] MSO query evaluation on* $\mathsf{PrXML}^{\mathsf{mux,ind}}$ *has ra-linear data complexity.*

4.5 Connection to Safe Queries

We now leave the context of existing relational and XML probabilistic frameworks, and show another consequence of our results, by connecting them to *query-based* tractability conditions.

More specifically, we focus on UCQs that are tractable in the sense of having polynomial OBDD representations of their lineage: by the results of [Jha and Suciu 2013], those are the *inversion-free UCQs*.

As we will show, the tractability of such queries can also be explained by our *data-based* tractability conditions: for any inversion-free UCQ, we can rewrite the input instances to constant pathwidth instances in a provenance-preserving way.

To do this, we introduce a general notion of *unfoldings* in Section 4.5.1. We then apply this notion to inversion-free UCQs in Section 4.5.2, and re-prove that they have polynomial OBDDs on any input instance, intuitively because input instances can be rewritten in PTIME to a bounded-pathwidth instance.

4.5.1 Unfoldings

Our results are based on instance rewritings of a general kind. We let I denote an arbitrary instance in this subsection, and let q denote a query closed under homomorphisms.

Definition 4.5.1. An *unfolding* of instance I is an instance I' with a homomorphism h to I which is *bijective on facts*: for any fact $F(\mathbf{a})$ of I, there is exactly one fact $F(\mathbf{a}')$ in I such that $h(a'_i) = a_i$ for all i. ◁

The bijection defined by the homomorphism allows us to see the provenance $\mathrm{Prov}(q, I')$ of q on an unfolding I' of I as a Boolean function on the same variables as the provenance $\mathrm{Prov}(q, I)$ of q on I.

We use unfoldings as a tool to show lineage-preserving instance rewritings. Indeed, we can immediately see from the homomorphism h from I' to I that any match of q in I' is preserved in I through h. In other words:

Lemma 4.5.2. *If I' is an unfolding of I, then for any valuation ν of the facts of I, if $\nu(\mathrm{Prov}(q, I')) = 1$ then $\nu(\mathrm{Prov}(q, I)) = 1$.*

The converse generally fails, but a sufficient condition is:

Definition 4.5.3. An unfolding I' of I *respects* q if, for any match $M \subseteq I$ of q on I, letting M' be its preimage in I', we have $M' \models q$. ◁

Intuitively, the unfolding does not "break" the matches of q. This ensures that the lineage is preserved exactly:

Lemma 4.5.4. *If I' is an unfolding of I that respects q, then $\mathrm{Prov}(q, I)$ and $\mathrm{Prov}(q, I')$ are the same Boolean function.*

Proof. By Lemma 4.5.2, it suffices to show that for any match M of q in I, the preimage M' of M by the bijection on facts is also a match of q; but this is precisely what is guaranteed by the fact that I respects q. □

4.5.2 Inversion-Free UCQs

We use unfoldings to study Boolean constant-free *inversion-free* UCQ queries. I will not restate their formal definition here, and refer the reader to [Jha and Suciu 2013, Section 2]. The following is known:

Theorem 4.5.5. *[Jha and Suciu 2013, Proposition 5] For any inversion-free UCQ q, for any input instance I, the lineage of q on I has an OBDD of constant width (i.e., the width only depends on q).*

When studying inversion-free UCQs, it is convenient to assume that the *ranking* transformation was applied to the query and instance [Dalvi, Schnaitter, and Suciu 2010; Dalvi and Suciu 2012]. A UCQ is *ranked* if, defining a binary relation on its variables by setting $x < y$ when x occurs before y in some atom, then $<$ has no cycle. In particular, in a ranked query, no variable occurs twice in an atom. An instance is *ranked* if there is a total order $<$ on its domain such that for any fact $R(\mathbf{a})$ and $1 \leqslant i < j \leqslant |R|$, we have $a_i < a_j$. In particular, no element occurs twice in a fact. Up to changing the signature, we can always rewrite a UCQ q to a ranked UCQ q', and rewrite separately any instance I to a ranked instance I', so that the lineage of q on I is the same as that of q' on I'; see [Dalvi, Schnaitter, and Suciu 2010; Dalvi and Suciu 2012] for details.

We will thus assume that the ranking transformation has been applied to the query, and to the instance. Note that this can be performed in linear time in the instance, and does not change its treewidth or pathwidth (the Gaifman graph is unchanged by this operation).

Once this ranking transformation has been performed, we can show the following:

Theorem 4.5.6. *For any ranked inversion-free UCQ q, for any ranked instance I, there is an unfolding I' of I that respects q and has pathwidth $\leqslant |\sigma|$.*

Hence, in particular, $\mathrm{Prov}(q, I) = \mathrm{Prov}(q, I')$, as shown by Lemma 4.5.4. By Theorem 3.5.4, this implies the result of Theorem 4.5.5, and (via Proposition 3.5.5) generalizes it slightly: it shows that $\mathrm{Prov}(q, I)$ can even be represented by a bounded-pathwidth circuit.

Theorem 4.5.6 thus suggests that the tractability of probability evaluation for inversion-free UCQs can be understood in terms of bounded-pathwidth tractability: what inversion-free UCQs "see" in an instance is a bounded pathwidth structure. The rest of this section proves Theorem 4.5.6.

The roadmap of the proof is as follows. We use an *inversion-free expression* [Jha and Suciu 2013] for q to define an order on relation attributes which is compatible across relations. We then unfold each relation by distinguishing each element depending on the tuple of elements on the preceding positions; this is inspired by [Jha and Suciu 2013, Proposition 5]. The result preserves the inversion-free expression and has a path decomposition that enumerates the facts lexicographically.

To follow this roadmap, we start by defining inversion-free expressions as in [Jha and Suciu 2013]:

Definition 4.5.7. A *hierarchical expression* [Jha and Suciu 2013] is a logical sentence built out of atoms, conjunction, disjunction, and existential quantification, where each variable is a *root variable*, i.e., occurs in all atoms in the scope of its existential quantifier.

An *inversion-free expression* is a hierarchical expression such that, for each relation symbol R, we can define a total order $<_R$ on its positions $\{R^1, \ldots, R^{|R|}\}$, such that, in every R-atom $R(\mathbf{x})$, if $R^i <_R R^j$ then the quantifier $\exists x_j$ in the query is in the scope of the quantifier $\exists x_i$. ◁

By Proposition 2 of [Jha and Suciu 2013], a ranked UCQ is inversion-free iff it can be written as an inversion-free expression, so it suffices to show Theorem 4.5.6 for inversion-free expressions.

We first define our unfolding I' of an input instance I. For each fact $R(\mathbf{a})$ of I, we create the fact $R(\mathbf{b})$ defined as follows. Writing $R^{i_1} <_R \cdots <_R R^{i_n}$ the positions of R according to the total order $<_R$, we define b_{i_1} as the tuple (a_{i_1}), and define b_{i_j} as the tuple formed by concatenating $b_{i_{j-1}}$ and (a_{i_j}). We call f_R the operation thus defined, with $\mathbf{b} = f_R(\mathbf{a})$. Clearly the operation h mapping each tuple to its last element is a homomorphism from I' to I, and it is bijective on facts because it is the inverse of the operation that we described. Hence, I' is an unfolding of I.

We must show that I' has bounded pathwidth. To do this, consider the sequence $\mathcal{A}$ of the $|\sigma|$-tuples of elements of $\mathrm{dom}(I)$, ordered in lexicographic order, following the ranking order on the domain of I. Consider a path decomposition whose sequence of bags are obtained from those tuples (in this order) by taking all possible non-empty truncations (as prefixes). We claim that this is indeed a path decomposition. Indeed:

- We show that any fact F' of I' is covered. Indeed, by definition of I', all its facts contain an element such that all other elements of the facts are truncations of that element. Letting $\mathbf{a}$ be this element for F, it is then clear that F is covered at any node created for a tuple in $\mathcal{A}$ of which $\mathbf{a}$ is a prefix.

- We show that the occurrences of any element are consecutive. This is immediate: the occurrences of any tuple $\mathbf{a}$ are precisely the subsequence of tuples of $\mathcal{A}$ of which $\mathbf{a}$ is a prefix.

Further, this path decomposition clearly has width at most $|\sigma|$. Note that this definition is similar to the construction used in the proof of Proposition 5 in [Jha and Suciu 2013].

The only thing left to show is that I' respects q. For this, let us consider the inversion-free expression Q of q. For any subexpression φ of Q with free variables $\mathbf{x}$, let us define the *ordered free variables* of φ, denoted $\mathrm{ofv}(\varphi)$, as follows. If φ contains no atoms (i.e., it is the constant formula "true" or "false"), then $\mathbf{x}$ is empty and so is $\mathrm{ofv}(\varphi)$. Otherwise, as Q is inversion-free, it is in particular hierarchical, so all free variables of φ must occur in all atoms of φ: this is by definition, for any free variable x_i of φ, of the subexpression of Q that includes φ whose outermost operator is $\exists x_i$. Hence, consider any atom $A = R(\mathbf{x})$, and, remembering that no variable occurs twice in A (as Q is ranked), define $\mathrm{ofv}(\varphi)$ as the total order on $\mathbf{x}$ given by $x_i < x_j$ iff $R^i <_R R^j$.

It is clear that $\mathrm{ofv}(\varphi)$ is well-defined, i.e., that does not depend on our choice of atom in φ: this is because Q is an inversion-free expression, so the order of variables in atoms must reflect the order in which the variables are quantified.

We now show the claim that I' respects Q:

Lemma 4.5.8. *If I has a match M of Q, then, defining M' by mapping each fact $R(\mathbf{a})$ of M to the fact $R(f_R(\mathbf{a}))$ of I', M' is a match of Q in I'.*

Proof. Let f be defined on tuples of $\mathrm{dom}(I)$ by $f(\mathbf{a}) := (a_1, (a_1, a_2), \ldots, \mathbf{a})$. For any subformula φ with n free variables and for any n-tuple $\mathbf{a}$ of $\mathrm{dom}(I)$, we write $I \models \varphi[\mathrm{ofv}(\varphi) := \mathbf{a}]$ to mean the Boolean formula with constants obtained by substituting each variable in $\mathrm{ofv}(\varphi)$ by the corresponding element in $\mathbf{a}$ following the order of $\mathbf{a}$ and $\mathrm{ofv}(\varphi)$.

We proceed by induction on the subformulae of Q, showing that if a subformula φ and tuple $\mathbf{a} \in \mathrm{dom}(M)$ is such that $M \models \varphi[\mathrm{ofv}(\varphi) := \mathbf{a}]$, then $M' \models \varphi[\mathrm{ofv}(\varphi) := f(\mathbf{a})]$.

- For atoms, this is by definition of ofv and by definition of M'.
- For $\varphi \wedge \psi$, we observe that we have $\mathrm{ofv}(\varphi) = \mathrm{ofv}(\varphi \wedge \psi) = \mathrm{ofv}(\psi)$: write $\mathbf{x}$ to refer to these ordered free variables. If $M \models (\varphi \wedge \psi)[\mathbf{x} := \mathbf{a}]$, then $M \models \varphi[\mathbf{x} := \mathbf{a}]$ and $M \models \psi[\mathbf{x} := \mathbf{a}]$, as by induction $M' \models \varphi[\mathbf{x} := f(\mathbf{a})]$ and $M' \models \psi[\mathbf{x} := f(\mathbf{a})]$, we deduce $M' \models (\varphi \wedge \psi)[\mathbf{x} := f(\mathbf{a})]$
- For $\varphi \vee \psi$, the reasoning is the same.
- For $\varphi : \exists y\ \psi$, writing $\mathbf{x} := \mathrm{ofv}(\varphi)$, by definition of ofv, y is the last variable of $\mathbf{x}' := \mathrm{ofv}(\psi)$. As $M \models \varphi[\mathbf{x} := \mathbf{a}]$, by definition there is $c \in \mathrm{dom}(M)$ such that, letting $\mathbf{a}'$ be the concatenation of $\mathbf{a}$ and c, $M \models \psi[\mathbf{x}' := \mathbf{a}']$. By induction hypothesis we have $M' \models \psi[\mathbf{x}' := f(\mathbf{a}')]$, and as removing the last element of $f(\mathbf{a}')$ yields $f(\mathbf{a})$, we deduce that $M' \models (\exists y\ \psi)[\mathbf{x} := f(\mathbf{a})]$.

The outcome of this induction is that $M \models Q$ implies $M' \models Q$, the desired claim. □

This concludes the proof of Theorem 4.5.6, and concludes our study of inversion-free queries. We now move to another application of our results.

4.6 Application to Match Counting

We conclude this chapter with a last application of our results, for the *match counting* problem, as defined in Section 2.6. Note that match counting should not be confused with *model counting* (counting how many subinstances satisfy a Boolean formula) which is closely related[1] to probability evaluation.

We show that we can solve in ra-linear time the match counting problem for GSO queries with free variables. This proceeds via a reduction to the probability evaluation problem, which we solve using our methods. This result was shown for MSO in [Arnborg, Lagergren, and Seese 1991].

Theorem 4.6.1. *[Arnborg, Lagergren, and Seese 1991] For any fixed GSO query $q(\mathbf{X}, \mathbf{x})$ with free first- and second-order variables, for any constant $k \in \mathbb{N}$, given an input instance I of width $\leqslant k$, the match counting problem for q on I can be solved in ra-linear data complexity.*

Proof. We will show the result for GSO queries $q(\mathbf{X})$ where all free variables are second-order, as we can clearly rewrite any GSO query to replace each free first-order variable x to a free monadic second-order variable X, adding a conjunct that asserts that X contains exactly one element, and using quantification variable x in the query to bind it to the one element of X.

Consider the input treelike instance I over signature σ, and augment the signature to σ' by adding an n_i-ary relation A_i for each free second-order variable X_i of q of arity n_i. We then let I' be the instance obtained from I by adding, for each variable X_i of arity n_i, the fact $A_i(a_1, \ldots, a_{n_i})$ to I, for each *guarded* tuple $\mathbf{a}$ of I. As each fact of I guards a number of tuples whose number only depends on σ, the number of guarded tuples of I is linear in I, and we can construct this instance I' in linear time

[1] The number of models of a query q in an instance of size n is 2^n multiplied by the probability of q under the valuation that gives probability $1/2$ to each fact.

in I. Let q' be the Boolean GSO query obtained from q by replacing each occurrence of each free variable X_i in an atom $X_i(\mathbf{x})$ by the atom $A_i(\mathbf{x})$.

Now construct in linear time in I a d-DNNF representation C of $\mathrm{Prov}(q, I')$ using Theorem 3.5.8. Change C to C' by replacing by a 1-gate all input gates of C that stand for a fact of I, so the inputs of C correspond exactly to the facts of $I' \backslash I$. It is immediate that there is a bijection between assignments of $\mathbf{X}$ on $\mathrm{dom}(I)$ and subinstances I'' of I' such that $I \subseteq I''$, from which we deduce that there is a bijection from such assignments to the valuations of C': the bijection maps any assignment $\mathbf{B}$ of $\mathbf{X}$ to the valuation $\nu_{\mathbf{B}}$ of C'_{inp} obtained by setting the gate for each fact $A_i(\mathbf{a})$ to 1 iff B_i contains the tuple $\mathbf{a}$. Further, it is immediate that $I \models q(\mathbf{B_i})$ iff $\nu(I') \models q'$, that is, iff $\nu(C')$ evaluates to 1. Hence, we have shown that the number N of matches of q on I is exactly the number of valuations that satisfy C', that is, letting $N' := 2^{|I' \backslash I|}$, we have $N := N' \cdot \pi(C')$ where π maps each $g \in C'_{\mathsf{inp}}$ to $1/2$.

Hence, we compute in ra-linear time $p := \pi(C')$ using Theorem 4.1.3, and, as N' is singly exponential in $|I|$, we can compute in ra-linear time from p the number N of matches of q on I by the previous formula, concluding the proof. □

Chapter 5

Extending to General Semirings

We have shown in the previous chapter how our general construction of provenance on treelike instances, described in Chapter 3, could be used to derive the tractability of probability evaluation for many existing relational and XML probabilistic frameworks, assuming that treewidth is bounded.

In this chapter, we extend the results of Chapter 3 in a different direction: we show how they connect to the existing definitions of *semiring provenance* on arbitrary relational instances [Green, Karvounarakis, and Tannen 2007].

We start in Section 5.1 by recalling the fundamental definitions of semiring provenance [Green, Karvounarakis, and Tannen 2007]. We notice that our definitions of Boolean provenance in the *monotone* case (Section 3.4) already capture PosBool$[X]$-*provenance*, i.e., the provenance of queries in the semiring of positive Boolean functions. We accordingly focus on provenance in the *universal* semiring $\mathbb{N}[X]$, for which we limit our study to UCQ$^{\neq}$ queries; the reason for this is presented at the end of the chapter, in Section 5.4. Under this restriction, we can show:

Theorem 5.3.10. *For any fixed $k \in \mathbb{N}$ and UCQ$^{\neq}$ query q, for any σ-instance I such that $\mathrm{tw}(I) \leqslant k$, one can construct a $\mathbb{N}[X]$-circuit that captures $\mathrm{Prov}_{\mathbb{N}[X]}(q, I)$ in time $O(|I|)$. The treewidth of C only depends on k and q (not on I).*

To prove this theorem, we start by extending our provenance constructions for trees (Theorem 3.1.4) in Section 5.2, to support what is needed to compute $\mathbb{N}[X]$-provenance. We then generalize this result to treelike instances and UCQ queries in Section 5.3, for which we must revisit our tree interpretation scheme, and conclude the proof of the result.

5.1 Defining Semiring Provenance

We now review standard definitions of semiring provenance. We start by defining and exemplifying semirings in Section 5.1.1. We then present in Section 5.1.2 the semiring provenance of Datalog queries as defined in [Green, Karvounarakis, and Tannen 2007]; we focus on $\mathbb{N}[X]$-provenance as it captures all other semirings. In particular, we show that the result of specializing $\mathbb{N}[X]$-provenance for Datalog programs matches the notion of provenance that we studied so far.

We then restrict to UCQ and UCQ$^{\neq}$ queries in Section 5.1.3, for which we give a simplified $\mathbb{N}[X]$-provenance definition: for UCQs, which can be expressed in Datalog, the definition matches that of [Green, Karvounarakis, and Tannen 2007]. We will use

these definitions in the rest of this chapter to show how to extend our constructions to capture $\mathbb{N}[X]$-provenance for $\text{UCQ}^{\neq}$ queries.

5.1.1 General Semiring Definitions

We start by recalling the definition of *commutative semirings* (all semirings that we consider will be commutative):

Definition 5.1.1. A *commutative semiring* $(K, \oplus, \otimes, 0_K, 1_K)$ is a set K with binary operations $\oplus$ and $\otimes$ and distinguished elements 0_K and 1_K, such that $(K, \oplus)$ and $(K, \otimes)$ are commutative monoids with identity element 0_K and 1_K, the operation $\otimes$ distributes over $\oplus$, and $0_K \otimes a = 0_K$ for all $a \in K$. ◁

The provenance in a semiring K for positive relational algebra queries according to [Green, Karvounarakis, and Tannen 2007] is defined on instances where each fact is annotated with an element of K. The provenance of such a query on such an instance is an element of K obtained by combining fact annotations when applying each operator of the query, intuitively describing how the query output depends on the annotations.

We will soon formalize this, but for now, let us consider a few examples of semirings, along with the intuition of the provenance that we obtain if we use them:

Example 5.1.2. For any variable set X, we call $\text{PosBool}[X]$ the *monotone Boolean functions* over X. We can define a *semiring* structure $(\text{PosBool}[X], \vee, \wedge, 0, 1)$, where $\vee$ and $\wedge$ denote the Boolean OR and AND operations on $\text{PosBool}[X]$ (i.e., $f \vee g$ is the Boolean function defined by $\nu(f \vee g) := \nu(f) \vee \nu(g)$, and likewise for $\wedge$), and 0 and 1 denote the constant Boolean functions. Observe that $\vee$ is an associative operation with identity element 0, that $\wedge$ is an associative operation with identity 1, that $\wedge$ distributes over $\vee$, and that 0 is *absorptive* for $\wedge$, i.e., $0 \wedge \varphi = 0$ for all $\varphi \in \text{PosBool}[X]$.

The *natural numbers* $\mathbb{N}$, with the usual sum $+$ and product $\times$, and with $0, 1 \in \mathbb{N}$, form a semiring. On instances where facts are all annotated with $1 \in \mathbb{N}$, the $\mathbb{N}$-provenance of a query describes its number of matches under the bag semantics.

The *security semiring* $\mathbb{S}$ [Foster, Green, and Tannen 2008] is defined on the ordered set $1_{\mathbb{S}} < C < S < T < 0_{\mathbb{S}}$ (respectively: *always available*, *confidential*, *secret*, *top secret*, *never available*) as $(\{1, C, S, T, 0\}, \min, \max, 0, 1)$. The $\mathbb{S}$-provenance of a query, on an instance where facts are labeled by their security clearance, denotes the minimal clearance level required to see that the query holds.

The *fuzzy semiring* [Amsterdamer, Deutch, and Tannen 2011] is the semiring $([0, 1], \max, \min, 0, 1)$. On instances where facts are annotated with a fuzziness level in $[0, 1]$, the provenance of a query for this semiring is the minimal fuzziness value that has to be tolerated for the facts used to witness that the query is satisfied.

The *tropical semiring* [Deutch, Milo, Roy, and Tannen 2014] is the semiring $(\mathbb{N} \sqcup \{\infty\}, \min, +, \infty, 0)$. On instances where each fact is annotated by its cost, the tropical provenance of a query is the minimal cost of the facts required to satisfy it, with multiple uses of a fact being charged multiple times.

For any set of variables X, the *polynomial semiring* $\mathbb{N}[X]$ is the semiring of polynomials with variables in X and coefficients in $\mathbb{N}$, with the usual sum and

product over polynomials, and with $0, 1 \in \mathbb{N}$. The $\mathbb{N}[X]$-provenance of queries, as we will see, captures the provenance for any other semiring.

5.1.2 $\mathbb{N}[X]$-Provenance and PosBool$[X]$-Provenance

We define provenance for the *Datalog* query language (as defined in Section 2.5), because it is the most expressive query language studied in [Green, Karvounarakis, and Tannen 2007]. Here is the formal definition of semiring provenance for Datalog, according to [Green, Karvounarakis, and Tannen 2007]:

Definition 5.1.3. [Green, Karvounarakis, and Tannen 2007] Given a semiring K and an instance I where each fact F carries an annotation $\alpha(F) \in K$, the *K-provenance* of a Datalog query P on I is:

$$\mathrm{Prov}_K(P, I) := \bigoplus_{\substack{T \text{ proof} \\ \text{tree of } P}} \bigotimes_{\substack{n \text{ leaf} \\ \text{of } T}} \alpha(n).$$

Note that this expression may not always be defined depending on the query, instance, and semiring. In particular, the number of terms in the sum may be infinite, so that the result cannot necessarily be represented in the semiring. ◁

$\mathbb{N}[X]$-provenance. Instead of working in an arbitrary semiring K and with arbitrary instance annotations, it will be simpler to fix K to be the semiring $\mathbb{N}[X]$, and fix instance annotations to give each fact its own variable.

Indeed, as shown by [Green, Karvounarakis, and Tannen 2007], the provenance of any Datalog query P, for *any* semiring K, can be computed from the $\mathbb{N}[X]$-provenance of P on instances where each fact is annotated by its own variable in X. Indeed, the provenance can then be *specialized* to K, by interpreting the resulting expression as an expression in K, and the actual annotations in K of the instance facts, once known, can be used to replace the variables in the expression, thanks to a *commutation with homomorphisms* property.

Hence, except for the next paragraph about PosBool$[X]$, the rest of this chapter focuses on $\mathbb{N}[X]$-provenance, and assumes each instance fact to be labeled by its own element in X. This is without loss of generality, and covers all of the previous examples.

PosBool$[X]$-provenance. We note, however, that specializing $\mathbb{N}[X]$-provenance to the semiring PosBool$[X]$ of positive Boolean functions yields back the definition of provenance that we have defined in Chapter 3.

Proposition 5.1.4. *For any Datalog program P and instance I, the* PosBool$[X]$-provenance *of P on I, i.e., the result of specializing* $\mathrm{Prov}_{\mathbb{N}[X]}(P, I)$ *to* PosBool$[X]$, *is exactly* $\mathrm{Prov}(q, I)$.

Proof. Let P be a Datalog program and I be an instance where each fact F of I is labeled by its own variable $\alpha(F)$ in X. Specializing the definition of Definition 5.1.3, we obtain:

$$\mathrm{Prov}_{\mathrm{PosBool}[X]}(P, I) := \bigvee_{\substack{T \text{ proof} \\ \text{tree of } P}} \bigwedge_{\substack{n \text{ leaf} \\ \text{of } T}} \alpha(n).$$

Observe now that for any $I' \subseteq I$, which we can write $I' = \nu(I)$ for some valuation ν, we have $\nu(\mathrm{Prov}_{\mathrm{PosBool}[X]}(P, I)) = 1$ iff one of the disjuncts is satisfied, namely, iff I' contains all leaves of some derivation tree P, which is clearly equivalent to saying that $I' \models P$. Hence, we have $\nu(\mathrm{Prov}_{\mathrm{PosBool}[X]}(P, I)) = 1$ iff $\nu(I) \models P$, which is precisely the definition of $\mathrm{Prov}(P, I)$.

Therefore, the two definitions indeed match, which concludes the proof. □

This implies that the results of Section 3.4 already apply to PosBool$[X]$-provenance, for those Datalog queries that can be expressed in GSO. In particular, by Theorem 3.4.2, we can compute PosBool$[X]$-provenance circuits, i.e., monotone Boolean circuits, representing the PosBool$[X]$-provenance of such queries on treelike instances, in linear time data complexity.

5.1.3 Restricting to UCQ$^{\neq}$

We have explained why we can understand our previous sections as *semiring provenance* constructions for *monotone* queries, in the specific semiring PosBool$[X]$. It is then natural to try to extend them to $\mathbb{N}[X]$-provenance constructions, which we do in the rest of this chapter.

To do so, we will need to restrict the query language; we will revisit this restriction in Section 5.4. Specifically, we will restrict to *UCQ$^{\neq}$ queries*, which are in particular monotone. Importantly, throughout this chapter, *we allow UCQ$^{\neq}$ queries to contain multiple occurrences of the same atom*, e.g., we do not identify the queries $\exists x\ R(x)$ and $\exists x\ R(x) \wedge R(x)$, intuitively because they will have different $\mathbb{N}[X]$-provenances, even though they are logically equivalent (and have the same PosBool$[X]$-provenance).

We first define the provenance of UCQ queries via the definition of provenance for Datalog queries given above. We later give an equivalent restatement of this definition, which we use in the sequel, which is also generalized to UCQ$^{\neq}$. Here is our definition of the $\mathbb{N}[X]$-provenance of UCQ queries through provenance for Datalog:

Definition 5.1.5. We assume that CQs and UCQs contain no equality atoms. The Datalog query P_q *associated* to a CQ q has only one rule, Goal $\leftarrow q$. The Datalog query P_q associated to a UCQ $q = \bigvee_i q_i$ has rules Goal $\leftarrow q_1$, ..., Goal $\leftarrow q_n$. The $\mathbb{N}[X]$-*provenance* of a CQ or UCQ query q, on an instance where each fact is labeled by its own variable in X, is $\mathrm{Prov}_{\mathbb{N}[X]}(P_q, I)$ defined as in Definition 5.1.3.

Observe that in this case the provenance of P_q in the sense of Definition 5.1.3 is always defined, no matter the semiring, as the number of possible derivation trees is clearly finite. ◁

It will be simpler, however, to work with the following equivalent definition of $\mathbb{N}[X]$-provenance for UCQ, which we extend to UCQ$^{\neq}$:

Definition 5.1.6. Let $q = \bigvee_{i=1}^{n} \exists \mathbf{x}_i\ q_i(\mathbf{x}_i)$ be a UCQ$^{\neq}$ query, where each q_i is a CQ$^{\neq}$. The $\mathbb{N}[X]$-provenance of q on an instance I is defined as:

$$\mathrm{Prov}_{\mathbb{N}[X]}(q, I) \quad := \quad \bigoplus_{i=1}^{n} \quad \bigoplus_{\substack{f:\mathbf{x}_i \to \mathrm{dom}(I) \\ \text{such that} \\ I \models q_i(f(\mathbf{x}_i))}} \quad \bigotimes_{A(\mathbf{x}_i) \in q_i} A(f(\mathbf{x}_i))$$

In other words, we sum over each disjunct, and over each match of the disjunct (which must in particular respect the disequality atoms); for each match, we take

the product, over the atoms of the disjunct, of their image fact in I. Note that we identify each fact to the one variable in X that annotates it. ◁

We show that the above definition, for the case of UCQ queries, is equivalent to the Datalog-based one, so that we can indeed use it in the rest of this chapter:

Proposition 5.1.7. *For any UCQ q, $\mathrm{Prov}_{\mathbb{N}[X]}(q, I)$ (Definition 5.1.6) is exactly $\mathrm{Prov}_{\mathbb{N}[X]}(P_q, I)$ (Definition 5.1.3).*

Proof. We first show the claim for any CQ q. The proof trees for P_q have a fixed structure, the only unspecified part being the assignment of variables. It is then clear that each variable assignment gives a proof tree, and this mapping is injective because all variables in the assignment occur in the proof tree, so the sum in both definitions of provenance is over the same set. Further, each term of the sum in $\mathrm{Prov}_{\mathbb{N}[X]}(P_q, I)$ is the leaves of the proof tree, which is how we defined $\mathrm{Prov}_{\mathbb{N}[X]}(q, I)$. So the claim is proven for any CQ.

To extend the claim to any UCQ q, notice that the set of proof trees of P_q is the disjoint union of the set of proof trees of each CQ, which follows our definition of $\mathrm{Prov}_{\mathbb{N}[X]}(q, I)$. □

We conclude with an example to illustrate $\mathbb{N}[X]$-provenance and compare it to PosBool$[X]$-provenance. This illustrates some of the challenges that we will have to face in the rest of this chapter.

Example 5.1.8. Consider the instance $I = \{F_1 := R(a, a), F_2 := R(b, c), F_3 := R(c, b)\}$ and the CQ $q : \exists xy\, R(x, y) \wedge R(y, x)$. We have $\mathrm{Prov}_{\mathbb{N}[X]}(q, I) = F_1^2 + 2F_2F_3$ and $\mathrm{Prov}(q, I) = F_1 \vee (F_2 \wedge F_3)$.

Unlike PosBool$[X]$-provenance, $\mathbb{N}[X]$-provenance can describe that multiple atoms of the query map to the same fact (here, F_1), and that the same subinstance is obtained with two different query matches (here, F_2F_3).

Evaluating the $\mathbb{N}[X]$-provenance in the semiring $\mathbb{N}$ with facts annotated by 1, we deduce that q has $1^2 + 2 \times 1 \times 1 = 3$ matches.

5.2 Semiring Provenance Circuits for Trees

We now start to investigate how we can extend our results in Chapter 3 to capture the $\mathbb{N}[X]$-provenance $\mathrm{Prov}_{\mathbb{N}[X]}(q, I)$ of a UCQ$^{\neq}$ q on a treelike instance I, using the definition of the previous section.

We start in this section by extending the results of Section 3.1, i.e., we work with tree automata on uncertain trees. We will build on this in the next section to generalize Section 3.3, i.e., our results on treelike instances. Our goal in this section is thus to give a definition of $\mathbb{N}[X]$-provenance for tree automata, and show how we can compute provenance circuits for this definition.

First, our definition should capture the *number of uses* of each fact. To do this, instead of considering $\overline{\Gamma}$-trees, we consider $\overline{\Gamma}^p$*-trees* for $p \in \mathbb{N}$, whose label set is $\Gamma \times \{0, \ldots, p\}$ rather than $\Gamma \times \{0, 1\}$. Intuitively, rather than uncertainty about whether facts are present or missing, we represent uncertainty about the *number of available copies* of facts, as UCQ matches may include the same fact multiple times. We write $\mathrm{Val}^p(T)$ for the set of all *p-valuations* $\nu : V \to \{0, \ldots, p\}$ of a Γ-tree T.

Second, we must capture the *number of matches*, which we do using the nondeterminism of our tree automata. We write $|\mathrm{aruns}(A, T)|$ for a $\overline{\Gamma}^p$-tree T and $\overline{\Gamma}^p$-bNTA A to denote the number of accepting runs of A on T.

We can now define:

Definition 5.2.1. The $\mathbb{N}[X]$-*provenance* of a $\overline{\Gamma}^p$-bNTA A on a Γ-tree T for $p \in \mathbb{N}$ is

$$\mathrm{Prov}_{\mathbb{N}[X]}(A, T) := \bigoplus_{\nu \in \mathrm{Val}^p(T)} |\mathrm{aruns}(A, \nu(T))| \otimes \bigotimes_{n \in T} n^{\nu(n)}$$

where each node $n \in T$ is identified with its own variable in X. Intuitively, we sum over all valuations ν of T to $\{0, \ldots, p\}$, and take the product of the tree nodes to the power of their valuation in ν, with the number of accepting runs of A on $\nu(T)$ as coefficient; in particular, the term for ν is 0 if A rejects $\nu(T)$. ◁

While this definition would specialize in PosBool$[X]$ to our earlier definition of $\mathrm{Prov}(A, T)$, it extends that definition with the two features of $\mathbb{N}[X]$: multiple copies of the same nodes (represented as $n^{\nu(n)}$) and multiple derivations (represented as $|\mathrm{aruns}(A, \nu(T))|$).

A subtlety is that the provenance of a bNTA, according to this definition, is obtained by summing over *all* valuations where the automaton accepts, going from 0 to the maximal multiplicity p. Our eventual goal, however, is to compute provenance for UCQ$^{\neq}$ queries, for which provenance is defined (Definition 5.1.6) by looking at matches of the query, not superinstances of query matches with useless additional facts. The reason why this design alternative issue did not arise for PosBool$[X]$-provenance is because PosBool$[X]$ is an *absorptive* semiring [Deutch, Milo, Roy, and Tannen 2014]. For this reason, we must introduce a variant of the previous definition, that we will use in Section 5.3:

Definition 5.2.2. For a Γ-tree T and $p \in \mathbb{N}$, we introduce for $l \in \mathbb{N}$ the set of p-valuations that sum to l:

$$\mathrm{Val}^p_l(T) := \{\nu \in \mathrm{Val}^p(T) \mid \sum_{n \in T} \nu(n) = l\}$$

The $\mathbb{N}[X]$-l-*provenance* of a $\overline{\Gamma}^p$-bNTA A on a Γ-tree T is defined like $\mathrm{Prov}_{\mathbb{N}[X]}(A, T)$ but with $\mathrm{Val}^p_l(T)$ instead of $\mathrm{Val}^p(T)$, namely:

$$\mathrm{Prov}_{\mathbb{N}[X]}(A, T, l) := \bigoplus_{\nu \in \mathrm{Val}^p_l(T)} |\mathrm{aruns}(A, \nu(T))| \otimes \bigotimes_{n \in T} n^{\nu(n)}$$

To make this definition capture Definition 5.2.1, and simplify further notation, we also allow $l = \mathrm{all}$, define $\mathrm{Val}^p_{\mathrm{all}}(T) := \mathrm{Val}^p(T)$, so that $\mathrm{Prov}_{\mathbb{N}[X]}(A, T, \mathrm{all}) = \mathrm{Prov}_{\mathbb{N}[X]}(A, T)$. ◁

Intuitively, $\mathbb{N}[X]$-l-provenance does not look at all valuations that make the bNTA accept, but only those where the total number of node multiplicities sums up to a fixed value. This will be useful for UCQ$^{\neq}$ queries later.

We now generalize Theorem 3.4.4 to $\mathbb{N}[X]$-l-provenance. To do this, we need *arithmetic circuits*, which capture values in arbitrary semirings (here, $\mathbb{N}[X]$) rather than PosBool$[X]$:

Definition 5.2.3. A *K-circuit* for semiring $(K, \oplus, \otimes, 0_K, 1_K)$ is a circuit with $\oplus$- and $\otimes$-gates instead of OR- and AND-gates (and no analogue of NOT-gates), whose input gates stand for elements of K. As before, the constants 0_K and 1_K can be written as $\oplus$- and $\otimes$-gates with no inputs. The element of K *captured* by a K-circuit is the element captured by its distinguished gate, under the recursive definition that $\oplus$- and $\otimes$-gates capture the sum and product of the elements captured by their operands, and input gates capture their own value. ◁

The main claim of this section is that we can efficiently construct provenance circuits for $\mathbb{N}[X]$-l-provenance of bNTAs, and hence for their $\mathbb{N}[X]$-provenance:

Theorem 5.2.4. *For any fixed $p \in \mathbb{N}$ and $l \in \mathbb{N} \sqcup \{\mathrm{all}\}$, given a $\overline{\Gamma}^p$-bNTA A and a Γ-tree T, we can construct a $\mathbb{N}[X]$-l-*provenance circuit *C (i.e., a $\mathbb{N}[X]$-circuit capturing $\mathrm{Prov}_{\mathbb{N}[X]}(A, T, l)$) in time $O(|A| \cdot |T|)$. Further, C, has treewidth $O(|A|)$.*

We prove the theorem in the rest of this section. The correctness proof for the $\mathbb{N}[X]$-circuit that we construct is again inductive, and will rely on the following identity that we prove separately:

Lemma 5.2.5. *For any $l, p \in \mathbb{N}$ with $l \leqslant p$, for any non-singleton Γ-tree T written as $T = (V, L, R, \lambda)$, letting T_L and T_R be its left and right subtrees and n_r be its root node, for any $\overline{\Gamma}^p$-bNTA $A = (Q, F, \iota, \delta)$, calling A_q the bNTA obtained from A by making $q \in Q$ the only final state, we have:*

$$\mathrm{Prov}_{\mathbb{N}[X]}(A_q, T, l) = \bigoplus_{\substack{l_L + l_R + l' = l \\ q_L, q_R \in Q \text{ such that} \\ q \in \delta(q_L, q_R, (\lambda(n_\mathrm{r}), l'))}} \mathrm{Prov}_{\mathbb{N}[X]}(A_{q_L}, T_L, l_L) \otimes \mathrm{Prov}_{\mathbb{N}[X]}(A_{q_R}, T_R, l_R) \otimes n_\mathrm{r}^{l'}.$$

Proof. We first observe the following identity, for any $\nu \in \mathrm{Val}^p_l(T)$ and any $q \in Q$, by definition of automaton runs:

$$|\mathrm{aruns}(A_q, \nu(T))| = \sum_{\substack{q_\mathrm{L}, q_\mathrm{R} \in Q \text{ such that} \\ q \in \delta(q_\mathrm{L}, q_\mathrm{R}, (\lambda(n_\mathrm{r}), \nu(n_\mathrm{r})))}} |\mathrm{aruns}(A_{q_\mathrm{L}}, \nu(T_\mathrm{L}))| \cdot |\mathrm{aruns}(A_{q_\mathrm{R}}, \nu(T_\mathrm{R}))|$$

We then observe that we can write:

$$\mathrm{Val}^p_l(T) = \bigsqcup_{l_\mathrm{L} + l_\mathrm{R} + l' = l} \mathrm{Val}^p_{l_\mathrm{L}}(T_\mathrm{L}) \times \mathrm{Val}^p_{l_\mathrm{R}}(T_\mathrm{R}) \times \{n_\mathrm{r} \mapsto l'\}$$

as a valuation of T summing to l can be chosen as a valuation of its left and right subtree and of n_r by assigning the weights in all possible ways.

We thus rewrite $\mathrm{Prov}_{\mathbb{N}[X]}(A_q, T, l)$ by splitting the product over $n \in T$ in its definition in a product on n_r, on $n \in T_\mathrm{L}$ and on $n \in T_\mathrm{R}$, and using the above equalities:

$$\mathrm{Prov}_{\mathbb{N}[X]}(A_q, T, l) = \bigoplus_{\substack{\nu_\mathrm{L} \in \mathrm{Val}^p_{l_\mathrm{L}}(T_\mathrm{L}) \\ \nu_\mathrm{R} \in \mathrm{Val}^p_{l_\mathrm{R}}(T_\mathrm{R}) \\ l_\mathrm{L} + l_\mathrm{R} + l' = l}} \bigoplus_{\substack{q_\mathrm{L}, q_\mathrm{R} \in Q \\ q \in \Delta}} m_\mathrm{L} \otimes m_\mathrm{R} \otimes \left(\bigotimes_{n \in T_\mathrm{L}} n^{\nu_\mathrm{L}(n)} \right) \otimes \left(\bigotimes_{n \in T_\mathrm{R}} n^{\nu_\mathrm{R}(n)} \right) \otimes n_\mathrm{r}^{l'}$$

where we abbreviated $m_\mathrm{L} := |\mathrm{aruns}(A_{q_\mathrm{L}}, \nu_\mathrm{L}(T_\mathrm{L}))|$, $m_\mathrm{R} := |\mathrm{aruns}(A_{q_\mathrm{R}}, \nu_\mathrm{R}(T_\mathrm{R}))|$, and $\Delta := \delta(q_\mathrm{L}, q_\mathrm{R}, (\lambda(n_\mathrm{r}), l'))$.

Reordering sums and performing factorizations, we obtain the following expression of $\mathrm{Prov}_{\mathbb{N}[X]}(A, T, l)$:

$$\bigoplus_{\substack{q_{\mathrm{L}}, q_{\mathrm{R}} \in Q \\ l_{\mathrm{L}} + l_{\mathrm{R}} + l' = l \\ q \in \Delta}} \left(\bigoplus_{\nu_{\mathrm{L}} \in \mathrm{Val}^p_{l_{\mathrm{L}}}(T_{\mathrm{L}})} m_{\mathrm{L}} \otimes \bigotimes_{n \in T_{\mathrm{L}}} n^{\nu_{\mathrm{L}}(n)} \right) \otimes \left(\bigoplus_{\nu_{\mathrm{R}} \in \mathrm{Val}^p_{l_{\mathrm{R}}}(T_{\mathrm{R}})} m_{\mathrm{R}} \otimes \bigotimes_{n \in T_{\mathrm{R}}} n^{\nu_{\mathrm{R}}(n)} \right) \otimes n_{\mathrm{r}}^{l'}.$$

Plugging back the definition of provenance yields the desired claim. □

We are now ready to prove Theorem 5.2.4, which concludes the section. Intuitively, we adapt the proof of Theorem 3.4.4; the main differences are that we replace AND- and OR-gates by $\otimes$- and $\oplus$-gates, and that we must consider possible annotations in $\{0, \ldots, p\}$ instead of $\{0, 1\}$.

Proof of Theorem 5.2.4. We modify the proof of Theorem 3.4.4.

We fix l_0 to be the l provided as input. We will first assume $l_0 \in \mathbb{N}$, we explain at the end of the proof how to handle the (simpler) case $l_0 = \text{all}$.

For every node n of the tree T, we create one input gate g_n^{i} in C (identified to n), and for $0 \leqslant j \leqslant p$, we create a gate $g_n^{\mathrm{i},j}$ which is a $\otimes$-gate of j copies of the input gate g_n^{i}. (By "copies" we mean $\otimes$- or $\oplus$-gates whose sole input is g_n^{i}, this being a technical necessity as K-circuits are defined as graphs and not multigraphs.) In particular, $g_n^{\mathrm{i},0}$ is always a 1-gate.

We create one gate $g_n^{q,l}$ for $n \in T$, $q \in Q$, and $0 \leqslant l \leqslant l_0$.

For leaf nodes n, for $q \in Q$ and $0 \leqslant l \leqslant l_0$, we set $g_n^{q,l}$ to be $g_n^{\mathrm{i},l}$ if $q \in \iota(\lambda(n), l)$ and a 0-gate otherwise.

For internal nodes n, for every pair $q_{\mathrm{L}}, q_{\mathrm{R}} \in Q$ that appears as input states of a transition of δ and $0 \leqslant l_{\mathrm{L}}, l_{\mathrm{R}} \leqslant l_0$ such that $l_{\mathrm{L}} + l_{\mathrm{R}} \leqslant l_0$, we create the gate $g_n^{q_{\mathrm{L}}, l_{\mathrm{L}}, q_{\mathrm{R}}, l_{\mathrm{R}}}$ as a $\otimes$-gate of $g_{L(n)}^{q_{\mathrm{L}}, l_{\mathrm{L}}}$ and $g_{R(n)}^{q_{\mathrm{R}}, l_{\mathrm{R}}}$, and, for $0 \leqslant l' \leqslant l_0$ such that $l_{\mathrm{L}} + l_{\mathrm{R}} + l' \leqslant l_0$, we create one gate $g_n^{q_{\mathrm{L}}, l_{\mathrm{L}}, q_{\mathrm{R}}, l_{\mathrm{R}}, l'}$ as the $\otimes$-gate of $g_n^{q_{\mathrm{L}}, l_{\mathrm{L}}, q_{\mathrm{R}}, l_{\mathrm{R}}}$ and $g^{\mathrm{i}, l'}$. For $0 \leqslant l \leqslant l_0$, we set $g_n^{q,l}$ to be a $\oplus$-gate of all the $g_n^{q_{\mathrm{L}}, l_{\mathrm{L}}, q_{\mathrm{R}}, l_{\mathrm{R}}, l'}$ such that $q \in \delta(q_{\mathrm{L}}, q_{\mathrm{R}}, (\lambda(n), l'))$ and $l_{\mathrm{L}} + l_{\mathrm{R}} + l' = l$.

We define the output gate g_0 as a $\otimes$-gate of the $g_{n_{\mathrm{r}}}^{q, l_0}$ where n_{r} is the root of T. The construction is again in $O(|A| \cdot |T|)$ for fixed l and p.

To prove correctness, we show by induction that the element captured by $g_n^{q,l}$ is $\mathrm{Prov}_{\mathbb{N}[X]}(A_q, T_n, l)$ where A_q is A with q as the only final state, and T_n is the subtree of T rooted at n.

As a general property, note that for any node n, the value captured by $g_n^{\mathrm{i},j}$ for $0 \leqslant j \leqslant p$ is n^j.

For a leaf node n, we have $\mathrm{Prov}_{\mathbb{N}[X]}(A_q, T_n, l) = n^l$ if $q \in \iota(\lambda(n), l)$ and 0 otherwise, which is the value captured by $g_n^{q,l}$.

For an internal node n, the claim follows immediately by Lemma 5.2.5, applying the induction hypothesis to $g_{L(n)}^{q_L, l_{\mathrm{L}}}$ and $g_{R(n)}^{q_R, l_{\mathrm{R}}}$.

We conclude because clearly we have $\mathrm{Prov}_{\mathbb{N}[X]}(A, T, l_0) = \bigoplus_{q \in F} \mathrm{Prov}_{\mathbb{N}[X]}(A_q, T, l_0)$, so the value captured by g_0 is indeed correct.

Now, if $l_0 = \text{all}$, we do the same construction, but we only need a single node g_n^q for $n \in T$ and $q \in Q$ instead of $l_0 + 1$ nodes $g_n^{q,l}$. For leaf nodes, g_n^q is the $\oplus$-node of the $g_n^{\mathrm{i},l}$; for internal nodes, g_n^q is simply the $\oplus$-gate of all $g_n^{q_L, q_R, l}$ gates with $q \in \delta(q_L, q_R, (\lambda(n), l))$, each of them being a $\otimes$-gate of $g_n^{q_L, q_R}$ and $g^{\mathrm{i},l}$, where $g_n^{q_L, q_R}$ is

a $\otimes$-gate of $g^{q_L}_{L(n)}$ and $g^{q_R}_{R(n)}$. Correctness is shown using a variant of Lemma 5.2.5 on $\mathrm{Prov}_{\mathbb{N}[X]}(A, T, \mathrm{all})$ which replaces $l_\mathrm{L} + l_\mathrm{R} + l' = l$ in the sum subscript by $0 \leqslant l' \leqslant p$.

The reason why the resulting circuit has bounded treewidth is the same as in Theorem 3.1.4. □

5.3 Semiring Provenance Circuits for Instances

In this section, we move from trees to treelike instances, as we did in Section 3.3, but relying on the results of the previous section. Remember that we restrict to $\mathrm{UCQ}^{\neq}$ queries, which are in particular monotone.

We first translate in Section 5.3.1 our $\mathrm{UCQ}^{\neq}$ queries to *bag queries* on *bag instances*, i.e., queries that can test the multiplicity of facts. We then show in Section 5.3.2 that $\mathrm{UCQ}^{\neq}$ queries, when seen as bag-queries, can be translated to a bNTA that tests fact multiplicities as in the previous section. This takes care of the issue of multiple fact uses, but not of multiple derivations. We conclude by showing the main result for this chapter, Theorem 5.2.4, which takes care of the issue of multiple derivations by ensuring that the number of accepting runs of the automaton is correct, and produces the final provenance circuits using the results of the previous section.

We fix in this section the tree interpretation scheme discussed in Section 3.2, namely, the encoding operation $\mathcal{E}$, the decoding $\langle\cdot\rangle$, the translation $\mathcal{A}$ used in Theorem 3.2.3, and the alphabet Γ^σ_k. This time, we do not abstract as *subinstance-compatibility* the property that we require about the coding scheme, but we must rely on the details of Definition 3.2.4.

5.3.1 Bag-instances and bag-queries

To study the provenance of $\mathrm{UCQ}^{\neq}$ queries (Definition 5.1.6), and describe the *number of times* a fact is used, we introduce *bag-instances*:

Definition 5.3.1. A *multiset* is a function M from a finite *support* $\mathrm{supp}(M)$ to $\mathbb{N}$. We define the relation $M \subseteq M'$ if $\mathrm{supp}(M) \subseteq \mathrm{supp}(M')$ and for all $s \in \mathrm{supp}(M)$ we have $M(s) \leqslant M'(s)$. We write $x \in M$ to mean that $M(x) > 0$.

A *bag-instance* J is a multiset of facts on $\mathrm{dom}(J)$. Where necessary to avoid confusion, we call the ordinary instances *set-instances*. The *truncation to* p of a bag-instance J is $J^{\leqslant p}(F) := \min(J(F), p)$ for all $F \in \mathrm{supp}(J)$. We call J *p-bounded* if $J = J^{\leqslant p}$, i.e., equivalently, if the multiplicity of all facts in J is $\leqslant p$.

A *bag-homomorphism* h from a bag-instance J to a bag-instance J' is a mapping from $\mathrm{supp}(J)$ to $\mathrm{supp}(J')$ with the following condition: for each $F \in \mathrm{supp}(J')$, letting $F_1, \ldots, F_n$ be the facts of $\mathrm{supp}(J)$ such that $h(F_i) = F$ for $1 \leqslant i \leqslant n$, we have $\sum_{i=1}^n J(F_i) \leqslant J'(F)$. ◁

We accordingly define *bag-queries* as queries on such bag-instances. Intuitively, bag-queries are like regular Boolean queries on instances, except that they can "see" the multiplicity of facts. This is crucial to talk about the required multiplicity of facts in matches, which we need to talk about the $\mathbb{N}[X]$-provenance of $\mathrm{UCQ}^{\neq}$s. However, note that it ignores the issue of the *number of matches*, which we will address directly in the proof of the main result at the very end.

Definition 5.3.2. A *bag-query* q is a query on bag-instances, i.e., simply a function from bag-instances to $\{0, 1\}$. The bag-query q' *associated* to a CQ$^{\neq}$ $\exists \mathbf{x}\ q(\mathbf{x})$ is defined as follows. A *match* of q in a bag-instance J is a bag-homomorphism from q to J, seeing q as a bag-instance of facts over $\mathbf{x}$ (remember that q can contain multiple times the same atom). We say that $J \models q'$ if q has a match in J.

The bag-query q' associated to a UCQ$^{\neq}$ $q = \bigvee_{i=1}^{n} \exists \mathbf{x}_i\ q_i(\mathbf{x}_i)$ is such that $J \models q'$ iff some q_i has a match in J. ◁

An equivalent definition of $J \models q'$, for q' the bag-query associated to a UCQ$^{\neq}$ q and J a bag-instance, is that J contains a bag of facts that can be used as the leaves of a derivation tree for the Datalog query P_q associated to q.

We notice that the bag-query associated to a UCQ$^{\neq}$ q is *bounded*, namely, to see whether it holds or not cannot, it suffices to look at the multiplicity of facts up to a certain maximal value, namely, the maximal number of atoms in a disjunct of the UCQ$^{\neq}$. This is crucial for the construction of the previous section to be usable. Formally, *boundedness* of a bag-query is defined as follows:

Definition 5.3.3. A bag-query q is *p-bounded* for $p \in \mathbb{N}$ if, for any bag-instance J, if $J \models q$, then the truncation $J^{\leqslant p}$ of J is such that $J^{\leqslant p} \models q$. A bag-query is *bounded* if it is p-bounded for some $p \in \mathbb{N}$. ◁

5.3.2 Translating Bag-Queries to Tree Automata

We now show in this subsection that UCQ$^{\neq}$ queries can be translated to tree automata, by showing that the associated bag-query can be translated to a $\overline{\Gamma_k^{\sigma}}^p$-bNTA.

5.3.2.1 Preliminary Definitions

First, we generalize tree encodings to bag-instances as tree encodings annotated with the multiplicities of facts, that is, $\overline{\Gamma_k^{\sigma}}^p$-trees:

Definition 5.3.4. Let $k, p \in \mathbb{N}$ and let J be a p-bounded bag-instance. Let $I := \mathrm{supp}(J)$ be the underlying instance of J, and let $E := \mathcal{E}(I)$ be its tree encoding (a Γ_k^{σ}-tree). We define the *(k, p)-tree-encoding* $\mathcal{E}(J)$ of J as the $\overline{\Gamma_k^{\sigma}}^p$-tree with same skeleton as E where any node n encoding a fact F of I (formally, any node in the image of φ_I) is given the label $(\lambda(n), J(F))$ and other nodes are given the label $(\lambda(n), 0)$.

We accordingly define $\langle E \rangle$ on any $\overline{\Gamma_k^{\sigma}}^p$-tree E to yield a p-bounded bag-instance in the expected way: the only subtlety is that, when two nodes of E cause the same fact to be created, the multiplicity of that fact is set to the *largest* multiplicity for this fact obtained over all *individual* nodes of E that code this fact – *not* to the sum of its multiplicity across nodes of E that code that fact. ◁

We then define what it means for a $\overline{\Gamma_k^{\sigma}}^p$-bNTA to *test* a bag-query q, like a Γ_k^{σ}-bNTA could *test* a query q according to Definition 3.2.1. Note that the definition implies that the automaton cannot "see" multiplicities beyond p, so we require that the query be p-bounded so that this limitation does not matter. Notice that, again, we do not require anything yet about the number of runs of the bNTA.

Definition 5.3.5. For q a bag-query and $k, p \in \mathbb{N}$, a $\overline{\Gamma_k^{\sigma}}^p$-bNTA A *tests* q for treewidth k if q is p-bounded and for every $\overline{\Gamma_k^{\sigma}}^p$-tree E, we have $E \models A$ iff $\langle E \rangle \models q$. ◁

5.3.2.2 Translating Forced CQ$^{\neq}$ to Automata

Our goal is then to show that the *bag-query* associated to any UCQ$^{\neq}$ can be translated to a tree automaton. We will first show this claim for CQ$^{\neq}$ queries of a certain kind: the *forced* queries.

Definition 5.3.6. A CQ$^{\neq}$ query q is *forced* if, for any distinct variables x and y used in q, the disequality atom $x \neq y$ is in q. ◁

We translate forced CQ$^{\neq}$ queries to bNTAs reusing the translation from queries to tree automata defined by our tree interpretation scheme (see Definition 3.2.1), using a trick that allows us to handle the multiplicity of facts:

Proposition 5.3.7. *Let q be a forced CQ$^{\neq}$ and let q' be its associated bag-query. There is $p \in \mathbb{N}$ such that, for any $k \in \mathbb{N}$, we can compute a $\overline{\Gamma_k^{\sigma}}^p$-bNTA A that tests q' for treewidth k.*

Proof. Fix the forced CQ$^{\neq}$ q and let q' be the associated bag-query. Choose p to be the number of atoms in q. Let σ_p be the signature obtained from σ by creating a relation R^i for $1 \leqslant i \leqslant p$, with arity $|R|$, for every relation R of σ. Now, let q'' be the rewriting of q obtained by merging together duplicate atoms: we replace every atom $R(\mathbf{x})$ that occurs m times in q, with $m \geqslant 1$, by the disjunction $\bigvee_{m \leqslant j \leqslant p} R^j(\mathbf{a})$; the disequality atoms are left unchanged. We see the result of this process as a UCQ$^{\neq}$ over σ_p, where there are no longer any duplicate atoms, but which is still *forced* in the sense that there are disequalities between any pair of variables.

We now claim that for any p-bounded bag-instance J on σ, letting I be the set-instance obtained by replacing every fact $F = R(\mathbf{a})$ of J with multiplicity $m = J(F)$ by the fact $R^m(\mathbf{a})$, we have $J \models q'$ iff $I \models q''$. To see why, observe that, as q is a forced CQ$^{\neq}$, if q has a match h then every *distinct* atom A of q (regrouping copies of the same atoms) must be mapped by h to a fact of J (written $h(A)$) and this mapping must be injective (because h is, because q is forced), so that the necessary and sufficient condition is that, for every atom A of q, we have $J(h(A)) \geqslant p_A$ where p_A is the multiplicity of A in q; and this is equivalent to $I \models q''$ because $p_A \leqslant p$.

Now, q'' is a UCQ$^{\neq}$ on σ_p, so it is expressible in GSO. Hence, fixing $k \in \mathbb{N}$, using our tree interpretation scheme in Definition 3.2.1, we can compute a $\Gamma_k^{\sigma_p}$-bNTA $A := \mathcal{A}(q'')$ that tests q'' for width k on σ_p-instances. We write $A = (Q, F, \iota, \delta)$

We now build a $\overline{\Gamma_k^{\sigma}}^p$-bNTA $A' = (Q, F, \iota', \delta')$ by relabeling A in the following way. Recall the definition of Γ_k^{σ} (Definition 3.2.4). We define, for every $\tau = ((d, s), i)$ in $\overline{\Gamma_k^{\sigma}}^p$ and for every $q_{\mathrm{L}}, q_{\mathrm{R}} \in Q$:

- $\iota'(((d, s), i))$ to be $\iota((d, s'))$;
- $\delta'(q_{\mathrm{L}}, q_{\mathrm{R}}, ((d, s), i))$ to be $\delta(q_{\mathrm{L}} q_{\mathrm{R}}, (d, s'))$;

where we define s' as:

- $s' := s$ if $s = \emptyset$;
- $s' := \{R^i(\mathbf{a})\}$, with i as above, if $s = \{R(\mathbf{a})\}$.

We now claim that A' tests q' for treewidth k on p-bounded bag-instances. To see why, it suffices to observe that for any $\overline{\Gamma_k^{\sigma}}^p$-tree E, letting E' be the $\Gamma_k^{\sigma_p}$-tree

obtained in the straightforward manner, then A' accepts E iff A accepts E', which is immediate by construction. Now indeed, as we know that A accepts E' iff $\langle E' \rangle \models q''$ (as A tests q''), and we have $\langle E' \rangle \models q''$ iff $\langle E \rangle \models q'$ (as immediately $\langle E' \rangle$ is the σ_p-instance corresponding to $\langle E \rangle$ as I corresponds to J above), so we deduce that, for any $\overline{\Gamma_k^{\sigma^p}}$-tree E, we have $E \models A'$ iff $\langle E \rangle \models q'$, proving the desired equivalence.

The only thing left is to observe that A' does not only correctly test q on p-bounded instances, but on all bag-instances. But this is straightforward: as q matches at most p fact occurrences in any bag-instance J, we have $J \models q'$ iff $J^{\leqslant p} \models q'$. This concludes the proof. □

5.3.2.3 Translating UCQ$^{\neq}$ to Automata

We now extend our claim to arbitrary UCQ$^{\neq}$ queries. For this, we will use a standard union construction on bNTAs:

Lemma 5.3.8. *[Comon et al. 2007, Section 1.3] For any alphabet Γ and Γ-bNTAs $A_1, \ldots, A_n$, we can construct a Γ-bNTA $A_\cup$ such that for any Γ-tree T, we have $T \models A_\cup$ iff $T \models A_i$ for some A_i.*

We then show our claim: every bag-query corresponding to a UCQ$^{\neq}$ can be translated to an automaton that tests it. We do so by enumerating each self-homomorphism of the disjuncts to reduce them to forced CQ$^{\neq}$s.

Proposition 5.3.9. *Let q be a UCQ$^{\neq}$ and q' be the associated bag-query. There is $p \in \mathbb{N}$ such that, for any $k \in \mathbb{N}$, we can compute a $\overline{\Gamma_k^{\sigma^p}}$-bNTA A that tests q' for treewidth k.*

Proof. We first observe that, writing the UCQ$^{\neq}$ q as the disjunction of CQ$^{\neq}$s q_i, if we can show the claim for each q_i with some $p_i \in \mathbb{N}$, then the result clearly follows from q by computing one bNTA A_i for each q_i that tests q_i for treewidth k and uses $p := \max_i p_i$, and constructing the union bNTA $A_\cup$ of these bNTAs with Lemma 5.3.8. So it suffices to consider CQ$^{\neq}$ queries.

Remember that we see a CQ$^{\neq}$ q as an existentially quantified multiset of atoms with some disequalities, where the same atom, i.e., the same relation name applied to the same variables in the same order, can occur multiple times. Let $\mathrm{Vars}(q)$ be the set of the variables of q (which are all existentially quantified, as q is Boolean). We call $\mathcal{E}_q$ the set of all equivalence classes on $\mathrm{Vars}(q)$ such that, for each disequality $x \neq y$, the variables x and y must be in different classes; $\mathcal{E}_q$ is of course finite. For $\sim \in \mathcal{E}_q$ we let $q/\!\sim$ be the query in CQ$^{\neq}$ obtained by choosing one representative variable in $\mathrm{Vars}(q)$ for each equivalence class of $\sim$ and mapping every $x \in \mathrm{Vars}(q)$ to the representative variable for the class of x (dropping in the result the useless existential quantifications on variables that do not occur anymore), and adding disequalities $x \neq y$ between each pair of the remaining variables.

We rewrite a CQ$^{\neq}$ q to the UCQ$^{\neq}$ $q' := \bigvee_{\sim \in \mathcal{E}_q} q/\!\sim$. In other words, we are considering all possible self-homomorphisms of the query, and for each of them we impose disequalities between all variables. We claim that for every bag-instance I, we have that $I \models q$ iff $I \models q'$. For the forward implication, assuming that $I \models q$, letting m be the witnessing match, we consider the $\sim_m$ relation defined by $x \sim_m y$ iff $m(x) = m(y)$, and it is easily seen that $I \models q/\!\sim_m$. For the backward implication, if $I \models q/\!\sim$ for some $\sim \in \mathcal{E}_q$, it is immediate that $I \models q$ with the straightforward

match. Hence, using again Lemma 5.3.8, it suffices to show the result for queries in $\mathrm{CQ}^{\neq}$ which include disequality axioms between all their variables. But those are precisely the forced $\mathrm{CQ}^{\neq}$, so we can conclude using Proposition 5.3.7. □

5.3.3 Putting it together: $\mathbb{N}[X]$-provenance circuits for UCQs

We have shown in the previous subsection that $\mathrm{UCQ}^{\neq}$ queries could be translated to $\overline{\Gamma_k^{\sigma}}^p$-bNTAs that test them on bag-instances (Proposition 5.3.9), taking care of the issue of fact multiplicities (but not yet of the number of derivations). Further, we have shown in the previous section that we could compute provenance circuits for $\overline{\Gamma_k^{\sigma}}^p$-bNTAs (Theorem 5.2.4). We can now take care of the remaining difficulty and combine these results to prove the main result of this chapter:

Theorem 5.3.10. *For any fixed $k \in \mathbb{N}$ and $\mathrm{UCQ}^{\neq}$ query q, for any σ-instance I such that $\mathrm{tw}(I) \leqslant k$, one can construct a $\mathbb{N}[X]$-circuit that captures $\mathrm{Prov}_{\mathbb{N}[X]}(q, I)$ in time $O(|I|)$. The treewidth of C only depends on k and q (not on I).*

Remember that an $\mathbb{N}[X]$-circuit can then be specialized to a circuit for an arbitrary semiring (in particular, if the semiring has no variable, the circuit can be used to evaluate directly the final annotation from the annotations of the instance facts); thus, this provides provenance circuits for UCQs on treelike instances for any semiring.

We now prove this claim and conclude the section:

Proof of Theorem 5.3.10. We first prove the claim for $\mathrm{CQ}^{\neq}$, and then extend to $\mathrm{UCQ}^{\neq}$.

Let $k \in \mathbb{N}$ and let $q : \exists \mathbf{x}\ q'(\mathbf{x})$ be the $\mathrm{CQ}^{\neq}$. Let σ_{add} be the signature where we add a fresh unary predicate P_x for each variable x in $\mathbf{x}$. We rewrite q to the $\mathrm{CQ}^{\neq}$ q_{add} on σ_{add} defined as $\exists \mathbf{x}\ q'(\mathbf{x}) \wedge \bigwedge_{x \in \mathbf{x}} P_x(x)$. Let p be the number of atoms of q_{add}. We apply Proposition 5.3.9 to compute a $\overline{\Gamma_k^{\sigma_{\mathrm{add}}}}^p$-bNTA that tests the bag-query associated to q_{add} for treewidth k, and we then *determinize* this automaton [Comon et al. 2007], letting A be the resulting $\overline{\Gamma_k^{\sigma_{\mathrm{add}}}}^p$-bDTA that tests the bag query associated to q_{add}.

Let I be the input instance, let I^-_{add} be the σ_{add}-instance that contains one fact $P_x(a)$ for all $x \in \mathbf{x}$ and $a \in \mathrm{dom}(I)$, and let $I_{\mathrm{add}} := I \sqcup I^-_{\mathrm{add}}$. We can clearly compute I_{add} from I in linear time, and the treewidth of I_{add} is clearly that of I. Let $E_{\mathrm{add}} := \mathcal{E}(I_{\mathrm{add}})$ be the tree encoding of I_{add} as defined in Section 3.2, that is, a $\Gamma_k^{\sigma_{\mathrm{add}}}$-tree, and let φ be the function that maps the facts of I_{add} to the nodes of E_{add}.

We now apply Theorem 5.2.4, taking l to be the number of atoms in q_{add}, and obtain a $\mathbb{N}[X]$-circuit C' that captures the $\mathbb{N}[X]$-l-provenance of A on E_{add}, namely:

$$\mathrm{Prov}_{\mathbb{N}[X]}(A, E_{\mathrm{add}}, l) := \bigoplus_{\nu \in \mathrm{Val}_l^p(E_{\mathrm{add}})} |\mathrm{aruns}(A, \nu(E_{\mathrm{add}}))| \otimes \bigotimes_{n \in E_{\mathrm{add}}} n^{\nu(n)}$$

We modify C' to fix to 1 all inputs not in the image of $\varphi(I)$, and to rename the remaining inputs to match the facts of I. The resulting circuit C has been computed in linear time overall, and its treewidth only depends on k and q. So the only thing left to show is that C indeed captures $\mathrm{Prov}_{\mathbb{N}[X]}(q, I)$.

From the value captured by C', the circuit C captures by definition:

$$\bigoplus_{\nu \in \mathrm{Val}_l^p(E_{\mathrm{add}})} |\mathrm{aruns}(A, \nu(E_{\mathrm{add}}))| \otimes \bigotimes_{n \in \varphi(I)} (\varphi^{-1}(n))^{\nu(n)}$$

We now notice that, as A tests q_{add}, the only possible way for A to accept $\nu(E_{\text{add}})$ is if $\langle \nu(E_{\text{add}}) \rangle$ satisfies the bag-query associated to q_{add}. Writing l' for the number of atoms of q and l'' for the number of variables of q, notice that the total multiplicity of ν is $l = l' + l''$, so, by definition of q_{add}, we must have $\sum_{n \in \varphi(I)} \nu(n) = l'$ and $\sum_{n \in \varphi(I^-_{\text{add}})} \nu(n) = l''$. Hence, $\sum_{n \in E_{\text{add}} \setminus \varphi(I)} \nu(n) = 0$. We therefore split the sum over $\nu \in \text{Val}^p_l(E_{\text{add}})$ above, so that C captures:

$$\bigoplus_{\nu_{\text{add}} \in \text{Val}^p_{l''}(\varphi(I^-_{\text{add}}))} \bigoplus_{\nu \in \text{Val}^p_{l'}(\varphi(I))} |\text{aruns}(A, (\nu \sqcup \nu_{\text{add}})(E_{\text{add}}))| \otimes \bigotimes_{n \in \varphi(I)} (\varphi^{-1}(n))^{\nu(n)}$$

where $\nu \sqcup \nu_{\text{add}}$ is the valuation of E_{add} defined following ν on $\varphi(I)$, following ν_{add} on $\varphi(I^-_{\text{add}})$, and mapping other nodes to 0.

At this point, let us write $\text{Val}^p_{l'}(I)$ to denote the set of p-valuations of I that sum to l', and likewise for $\text{Val}^p_{l''}(I^-_{\text{add}})$, and write $\nu \sqcup \nu_{\text{add}}$ for $\nu \in \text{Val}^p_{l'}(I)$ and $\nu_{\text{add}} \in \text{Val}^p_{l''}(I^-_{\text{add}})$ to mean the valuation of E_{add} defined through φ in the expected way. We can thus rewrite the value captured by C above as follows:

$$\bigoplus_{\nu_{\text{add}} \in \text{Val}^p_{l''}(I^-_{\text{add}})} \bigoplus_{\nu \in \text{Val}^p_{l'}(I)} |\text{aruns}(A, (\nu \sqcup \nu_{\text{add}})(E_{\text{add}}))| \otimes \bigotimes_{F \in I} F^{\nu(\varphi(F))}$$

More specifically, again because A tests q_{add}, for A to accept $(\nu \sqcup \nu_{\text{add}})(E_{\text{add}})$, the valuation ν_{add} that sums to l'' must give valuation 1 to exactly one node corresponding to a P_x-fact, for each variable x. By definition of I^-_{add}, this allows us to rewrite the value captured by C as:

$$\bigoplus_{f : \mathbf{x} \to \text{dom}(I)} \bigoplus_{\nu \in \text{Val}^p_{l'}(I)} |\text{aruns}(A, (\nu \sqcup \nu_f)(E_{\text{add}}))| \otimes \bigotimes_{F \in I} F^{\nu(\varphi(F))}$$

where we define from a function $f : \mathbf{x} \to \text{dom}(I)$ the valuation ν_f of $\varphi(I^-_{\text{add}})$ that maps, for each fact $F = P_x(a)$ of I^-_{add}, the node $\varphi(F)$ of E_{add} to 1 if $f(x) = a$ and to 0 otherwise.

More specifically still, once we have chosen a function $f : \mathbf{x} \to \text{dom}(I)$, for A to accept $(\nu \sqcup \nu_f)(E_{\text{add}})$, it must be the case that its decoding $I' := \langle (\nu \sqcup \nu_f)(E_{\text{add}}) \rangle$ satisfies the bag-query associated to q_{add}. By definition of q_{add}, this means that I' satisfies the bag-query associated to q with a match that maps each variable x in $\mathbf{x}$ to $f(x)$. Clearly, such a match can only exist if $I \models q'(f(\mathbf{x}))$ (remember that I is a set-instance and q' is a $\text{CQ}^{\neq}$ in the usual sense). So we can rewrite what C captures as:

$$\bigoplus_{\substack{f : \mathbf{x}_i \to \text{dom}(I) \\ \text{such that} \\ I \models q'(f(\mathbf{x}_i))}} \bigoplus_{\nu \in \text{Val}^p_{l'}(I)} |\text{aruns}(A, (\nu \sqcup \nu_f)(E_{\text{add}}))| \otimes \bigotimes_{F \in I} F^{\nu(\varphi(F))}$$

Now, when $I \models q'(f(\mathbf{x}))$, from our observations on what the valuation ν must satisfy, and from the fact that it must sum to l, it is clear that there is exactly one possible valuation for each choice of f, namely, the valuation ν such that $I' := \langle (\nu \sqcup \nu_f)(E_{\text{add}}) \rangle$ is *exactly* a match of the bag-query associated to q' that maps each variable x to $f(x)$. In this case, for each fact $F \in I$, the value $\nu(\varphi(F))$ is exactly the number of atoms of q' that map to F in this match. Further, in this case, as A is deterministic, we have $|\text{aruns}(A, (\nu \sqcup \nu_f)(E_{\text{add}}))| = 1$. So C actually captures:

$$\bigoplus_{\substack{f : \mathbf{x}_i \to \text{dom}(I) \\ \text{such that} \\ I \models q'(f(\mathbf{x}_i))}} \bigotimes_{A(\mathbf{x}) \in q'} A(f(\mathbf{x}))$$

and this is exactly the provenance $\mathrm{Prov}_{\mathbb{N}[X]}(q, I)$.

For UCQ$^{\neq}$, observe that the provenance we need to compute (Definition 5.1.6) is simply the sum of the provenance for each CQ$^{\neq}$ disjunct. So we can just independently build a circuit for each disjunct using the construction above, and combine the circuits into one (merging the input gates), adding to the result an output gate which is the $\oplus$-sum of each output gate of the disjuncts. A tree decomposition for the resulting circuit can be built from that of each circuit, because their tree decompositions have same skeleton, i.e., the skeleton of the tree encoding of the instance; hence, treewidth increases only by a constant factor. □

This finishes the proof of the main result of this chapter. We conclude by additional remarks about whether our results can extend beyond UCQ$^{\neq}$.

5.4 Going Beyond UCQ$^{\neq}$

One could hope that our results would extend, beyond UCQ$^{\neq}$, to more expressive languages that are Datalog-expressible and GSO-expressible, to which both our constructions (Theorem 3.3.2) and semiring provenance (Definition 5.1.3) apply. The main issue that prevents this is *fact multiplicity*: multiple uses of facts are easy to describe for UCQ$^{\neq}$ (Definition 5.1.6), but for more expressive languages we do not know how to define them and connect them to automata.

In fact, we can build a query P, in *guarded Datalog* (recall the definition in Section 2.5), such that the smallest number of occurrences of a fact in a derivation tree for P cannot be bounded independently from the instance, in the sense of Definition 5.3.3:

Proposition 5.4.1. *There is a guarded monadic Datalog query P whose associated bag-query q_P is not bounded.*

Proof. Consider the Datalog query P consisting of the rules:

$$S(y) \leftarrow S(x), R(x, w, y), A(w)$$
$$\mathrm{Goal} \leftarrow S(x), T(x)$$

For all $n \in \mathbb{N}$, consider the instance $I_n = \{R(a_1, a, a_2), R(a_2, a, a_3), \ldots, R(a_{n-1}, a, a_n), S(a_1), T(a_n), A(a)\}$. It is easily verified that the only proof tree of P on I_n has $n-1$ leaves with the fact $A(a)$. Hence, assuming that the bag-query q_P captured by P is bounded by p, considering the bag-instance J formed of the leaves of the sole proof tree of P on I_{p+2}, it is not the case that $J^{\leqslant p} \models q_P$, contradicting boundedness. □

Thus, we cannot rewrite P to a fixed finite bNTA testing multiplicities on all input instances using our techniques. However, as guarded monadic Datalog is monotone and MSO-expressible, note that we can still compute the PosBool$[X]$-provenance of P with Theorem 3.4.2.

Chapter 6

Lower Bounds

This chapter concludes Part I of this manuscript by studying whether the previous results can be extended beyond bounded-treewidth instances. Indeed, more stringent conditions (e.g., bounded *clique-width* [Courcelle, Engelfriet, and Rozenberg 1993]) sometimes suffice to ensure the tractability of some tasks (e.g., non-probabilistic MSO query evaluation). For reasons that we will explain later, *we assume throughout this chapter that the signature is arity-two.*

We first focus on *tractable probability evaluation*, namely, our Theorem 4.1.1, according to which GSO probability evaluation is in ra-linear time on bounded-treewidth TID instances. We show that this result *cannot* be generalized to a less stringent requirement. Namely, there are even FO queries which are intractable to evaluate on *any* class of input instances that does not have bounded-treewidth, assuming a mild constructibility requirement, and under our assumption that the signature is arity-two.

This result should not be mistaken for the well-known fact that probability evaluation is known to be hard for some CQ queries when *arbitrary* input instances are allowed: we are really showing that *any other condition* on instances that would ensure the tractability of probability evaluation must imply that treewidth is bounded (or that the constructibility requirement is violated): if it does not, then, by our result, probability evaluation is intractable on any unbounded-treewidth subclass of instances that satisfy the condition. We present this result as a *dichotomy theorem*, together with our Theorem 4.1.1: assuming the constructibility requirement, any instance family is either bounded-treewidth, and then probability evaluation is in ra-linear data complexity for GSO queries by Chapter 3; or it is unbounded-treewidth, and a fixed FO query is already hard for it. This dichotomy result is presented in Section 6.1.

The lower bound of the dichotomy is proven by reducing a hard problem to the evaluation of an FO query on the arbitrary instance family, by *extracting* the hard problem instance as a topological minor of the family. For this to work, the instances of the family must be graphs (hence our requirement on the signature); the problem from which we reduce must be hard on restricted input graphs (namely, degree-3 planar graphs); and the FO query must test the problem under arbitrary subdivision, a challenging task with FO. We use the problem of counting graph matchings, and devise an appropriate FO query.

The specific extraction result that we use (as a black box) is the recent theorem from [Chekuri and Chuzhoy 2014a], giving a polynomial bound on the extraction.

This leads us to a digression to explain how a variation on our approach implies dichotomies for non-probabilistic MSO query evaluation and MSO match counting on subinstance-closed instance families. For non-probabilistic evaluation, we show in Section 6.2.1 a result obtained by direct adaptation of our techniques, and we show in Section 6.2.2 a variant of this result that improves existing lower bounds [Kreutzer and Tazari 2010; Ganian, Hliněný, et al. 2014] established before [Chekuri and Chuzhoy 2014a]. Assuming similar complexity-theoretic assumptions as in [Kreutzer and Tazari 2010], our result thus answers a conjecture of Grohe [Grohe 2007, Conjecture 8.3] for MSO, which [Kreutzer and Tazari 2010] had answered for MSO_2, and [Ganian, Hliněný, et al. 2014] had also addressed but under the additional assumption of closure under vertex labeling. For the problem of counting query matches, for which we were not aware of analogous results, we show in Section 6.3 a similar dichotomy.

Our second focus in this section is on *tractable lineage computation*, specifically, OBDDs. We have shown (Theorem 3.5.2) that OBDDs for GSO queries on bounded-treewidth instances can be computed in PTIME. As PTIME computation of OBDDs implies that probability evaluation is tractable, asking for tractable OBDDs is more demanding than asking for tractable probability evaluation as we did at first. As we show in Section 6.4, a dichotomy on the existence of polynomial-width OBDDs also holds, for the weaker language of $\mathrm{UCQ}^{\neq}$ (rather than MSO): there is a $\mathrm{UCQ}^{\neq}$ that has no polynomial-width OBDDs on *any* unbounded-treewidth instance family, assuming an even weaker constructibility requirement. This is again proven with minor extraction techniques, and a lower bound on OBDD width based on extracting sufficiently many independent matches from so-called *skewed grid* minors.

In the context of OBDDs, as it turns out, we can even classify *connected* $\mathrm{UCQ}^{\neq}$ in two categories: those for which the previous dichotomy holds, so they have no tractable OBDDs on all constructible unbounded-treewidth instance families, and those which are tractable on some well-chosen instance family. We study this in Section 6.5. Using a characterization of the first kind of queries (the *intricate* queries), we show a *meta-dichotomy theorem* on connected $\mathrm{UCQ}^{\neq}$. We also exhibit a *disconnected* $\mathrm{CQ}^{\neq}$ for which this meta-dichotomy fails.

6.1 Dichotomy on Probability Evaluation

This section studies whether we can extend our tractability result for *probability evaluation* (Theorem 4.1.1). Specifically, can we lift the bounded-treewidth requirement? We answer in the negative by a *dichotomy result* on arity-two signatures: there are queries for which probabilistic evaluation is tractable on bounded-treewidth families but is intractable on *any* efficiently constructible unbounded-treewidth family. The notion of *efficiently constructible* that we use is formally the following:

Definition 6.1.1. Remember that a *class* of instances $\mathcal{I}$ is just a (possibly infinite) set of instances. We say that $\mathcal{I}$ is *treewidth-constructible* if for all $k \in \mathbb{N}$, if $\mathcal{I}$ contains instances of treewidth $\geqslant k$, we can construct one in polynomial time given k written in unary[1]. ◁

[1]The requirement that k be given in unary rather than in binary means that *more* instance families are treewidth-constructible, so treewidth-constructibility in this sense is a weaker assumption than if the input k could be written in binary.

In particular, this implies that $\mathcal{I}$ must contain a subfamily of unbounded-treewidth instances that are small, i.e., have size polynomial in their treewidth. We discuss the impact of this choice of definition, and alternate definitions of *efficiently* constructible instances, in Section 6.2.2.

Remember that the *probability evaluation* problem for a query q on an instance class $\mathcal{I}$ asks, given an instance $I \in \mathcal{I}$ and a probability valuation π of I, to compute the probability $\pi(q, I)$ that q holds on I; and remember that we study this problem in *data complexity*, namely, as a function of I and π. Recall that we denote by $\mathrm{tw}(I)$ the width of an instance I. Our *main result* on probability evaluation is as follows:

Theorem 6.1.2. *Let σ be an arbitrary arity-2 signature. Let $\mathcal{I}$ be a treewidth-constructible class of σ-instances. Then the following dichotomy holds:*

- *If there is $k \in \mathbb{N}$ such that $\mathrm{tw}(I) \leqslant k$ for every $I \in \mathcal{I}$, then for every GSO query q, the probability evaluation problem for q on instances of $\mathcal{I}$ is solvable in ra-linear time.*
- *Otherwise, there is an FO query q_{h} (depending on σ but not on $\mathcal{I}$) such that the probability evaluation problem for q_{h} on $\mathcal{I}$ is $\mathrm{FP}^{\#P}$-complete under randomized polynomial time (RP) reductions.*

The first part of this result is precisely Theorem 4.1.1, so what must now prove is the second part of the theorem. Pay close attention to the statement: while some FO queries (in particular, unsafe CQs [Dalvi and Suciu 2012]) may have $\mathrm{FP}^{\#\mathrm{P}}$-hard probability evaluation when *all* input instances are allowed, our goal here is to build a query that is hard even when input instances are restricted to *arbitrary families* satisfying our conditions, a much harder claim.

To prove the second part of the theorem, we first present in Section 6.1.1 the general technique to extract degree-3 planar graphs as topological minors of any treewidth-constructible unbounded-treewidth family in randomized polynomial time, using [Chekuri and Chuzhoy 2014a]. We then formally construct in Section 6.1.2 the FO query that we use to test our choice of hard problem, namely, counting matchings, which is hard on degree-3 planar graphs by [Xia, Zhang, and Zhao 2007]. Last, Section 6.1.3 shows the complete reduction. We conclude by discussing alternative problems in Section 6.1.4.

6.1.1 Extracting Degree-3 Planar Graphs

We will reduce from the #P-hard problem of counting graph *matchings*, as explained in the next section. This section explains how we will be able to extract any instance of that problem as a topological minor of any treewidth-constructible unbounded-treewidth family $\mathcal{I}$. We give the formal definition:

Definition 6.1.3. A graph H is a *minor* of a graph G if there is a sequence of edge removals, vertex removals, and edge contractions (merging both endpoints of an edge into a single vertex adjacent to all neighbors of the two endpoints, not creating self-loops) which, applied on G, yields H.

An *embedding* of a graph H in a graph G is an injective mapping f from the vertices of H to the vertices of G and a mapping g that maps the edges (u, v) of H to paths in G from $f(u)$ to $f(v)$, all paths being vertex-disjoint. A graph H is a *topological minor* of a graph G if there is an embedding of H in G. ◁

We use the following lemma, that rephrases the recent polynomial bound [Chekuri and Chuzhoy 2014a] on Robertson and Seymour's grid minor theorem [Robertson and Seymour 1986] to the realm of topological minors:

Lemma 6.1.4. *There is $c \in \mathbb{N}$ such that for any degree-3 planar graph H, for any graph G of treewidth $\geqslant |V(H)|^c$, H is a topological minor of G and an embedding of H in G can be computed in randomized polynomial time in $|G|$.*

Proof. Fix the graph H. By [Chekuri and Chuzhoy 2014a, Corollary 1.1], for some $c > 0$, every graph of treewidth $O(|V(H)|^c)$ contains H as minor. Further, by [Chekuri and Chuzhoy 2014a, Theorem 1.1], there is a randomized polynomial time algorithm that computes, given G, a *model φ of H in G* (see Definition just before Section 2.1 in [Chekuri and Chuzhoy 2014b]), in randomized polynomial time. We explain how to obtain from this an embedding (f, g) of H in G, witnessing the fact that H is a topological minor of G, in PTIME. In so doing, we will use the fact that H is degree-3, which is why H being a minor of G implies that it is a topological minor.

For each pair u, v of adjacent nodes in H, we pick one *connecting edge* that has one endpoint in $\varphi(u)$ and the other in $\varphi(v)$. This can clearly be performed in PTIME. Now, these edges are all distinct because the $\varphi(w)$ are pairwise disjoint. All that remains is to pick in PTIME, for each node v of H, its image by f in $\varphi(v)$, and construct disjoint paths from that image to the endpoints of the connecting edges in $\varphi(v)$: this allows us to define g and concludes the claim.

To show this, for every vertex v, consider $\varphi(v)$ and the endpoints of connecting edges in $\varphi(v)$ (there are up to three). If a vertex is an endpoint for more than one connecting edge, pick it as $f(v)$, and, using the fact that $\varphi(v)$ is connected, take a path from $f(v)$ to the third endpoint (if any). Otherwise, if there are no more than two endpoints, pick any of them as $f(v)$ and conclude in the same manner. If there are three endpoints, as $\varphi(v)$ is connected, pick a covering tree T of $\varphi(v)$, and root it at one endpoint. Take the lowest common ancestor of the two other endpoints as $f(v)$, with disjoint paths from $f(v)$ to the root, and to its two descendent endpoints.

Hence, the above process being in PTIME, one can compute an embedding of H in G in randomized polynomial time. This proves the claim. □

6.1.2 Constructing the Hard FO Query

Having shown how to extract any input 3-regular planar graph G in RP as a topological minor in any treewidth-constructible unbounded-treewidth family, we formally define the #P-hard problem from which we reduce, namely, the problem of *counting matchings in a 3-regular planar graph*, #3PM:

Definition 6.1.5. A *matching* of a graph is a subset of its edges such that no two edges of the subset are *incident*, i.e., share an endpoint. The #3PM problem asks, given an input 3-regular planar graph, how many matchings does it have. ◁

The following is shown in [Xia, Zhang, and Zhao 2007], Theorem 11 (#3PM is #Pl-Pos-$\{F_{\{0,1\}}\}$-SAT in their terminology):

Lemma 6.1.6 ([Xia, Zhang, and Zhao 2007])**.** *#3PM is #P-complete.*

We accordingly construct our hard FO query q_{h} that we will use on all families of instances. We will explain in the next section how the query that we define now relates to the #3PM problem.

We define the FO formula Adj with two free variables x and y, that tests whether x and y co-occur in some fact:

$$\text{Adj}(x, y) := \bigvee_{\substack{R \in \sigma \\ |R|=2}} R(x, y) \vee R(y, x)$$

We use this to define FO formulae to test the degree of an element with respect to the Adj relation:

$$\begin{aligned}
\text{Deg1}(x) &:= \exists y\ \text{Adj}(x, y) \wedge (\forall y'\ \text{Adj}(x, y') \Rightarrow y = y') \\
\text{Deg3}(x) &:= \exists y_1 y_2 y_3\ \text{Adj}(x, y_1) \wedge \text{Adj}(x, y_2) \wedge \text{Adj}(x, y_3) \\
&\quad \wedge y_1 \neq y_2 \wedge y_1 \neq y_3 \wedge y_2 \neq y_3 \\
&\quad \wedge \forall y\ \text{Adj}(x, y) \Rightarrow (y = y_1 \vee y = y_2 \vee y = y_3)
\end{aligned}$$

We write a sentence testing that some element has degree 1 and its adjacent element (there is exactly one) does not have degree 3:

$$1\text{not}3 := \exists x\ (\text{Deg1}(x) \wedge (\forall y\ \text{Adj}(x, y) \Rightarrow \neg\text{Deg3}(y)))$$

We finally write an FO sentence testing that some element has degree 3 and has two distinct neighbors that do not have degree 1:

$$3\text{twonot}1 := \exists x\ (\text{Deg3}(x) \wedge (\exists yz\ y \neq z \wedge \text{Adj}(x, y) \wedge \text{Adj}(x, z) \wedge \neg\text{Deg1}(y) \wedge \neg\text{Deg1}(z)))$$

We then define our FO query q_{h}:

$$q_{\text{h}} := \neg 1\text{not}3 \wedge \neg 3\text{twonot}1$$

The membership in $\text{FP}^{\#\text{P}}$ of probability evaluation for q_{h} is easy to establish, by considering a Turing machine which nondeterministically chooses a subinstance of the input instance, and deterministically tests in PTIME whether the query holds; the number of accepting paths needs to be weighed by the probability of the instance, which can be achieved by standard techniques (see Lemma 5.1 of [Abiteboul, Chan, et al. 2011]). So what we have to do is to show hardness, which we do in the next section.

6.1.3 Completing the Reduction

We will use the notion of *subdivision*, which we recall:

Definition 6.1.7. A *subdivision* of a graph G is a graph obtained by replacing each edge by an arbitrary non-empty simple path (every node on this path being fresh except the endpoints of the original edge).

The *k-subdivision* G^k for $k \in \mathbb{N}_{>0}$ is the subdivision where we take all paths replacing an edge to be of length exactly k. ◁

The statement that H is a topological minor of G can be equivalently stated as: there is a subgraph of G which is isomorphic to a subdivision of H. Accordingly, when extracting our input graph G to the hard problem as a topological minor, we will use G^3 instead of G to guarantee that each edge of G was subdivided to a path of length at least 3.

The crucial lemma that connects q_{h} to our hard problem is the following. We first observe that if a graph G is planar and degree-3, then so is G^3; further, the nodes of G^3 having degree 3 are exactly those corresponding to the nodes of G. We now claim and show:

Lemma 6.1.8. *Write q'_{h} from q_{h} by replacing* Adj *by the adjacency relation E in the signature of graphs. Let G be a 3-regular planar graph. There is an algorithm that, given a graph G' with G^3 as topological minor (witnessed by an embedding), computes in PTIME a probability valuation π of G' such that we can compute the number of matchings of G in PTIME from G' and $\pi(q'_{\mathrm{h}}, G')$.*

Proof. Let (f, g) be an embedding of G in G' computed straightforwardly from the embedding of G^3 in G' and the obvious embedding of G in G^3. (Intuitively, the only point of using G^3 rather than G is to ensure that G' subdivides each edge of G to a path of length at least 3.)

Let π be the probability valuation of G' defined in PTIME by assigning:

- probability 0 to all edges not part of a path in the image of g;
- probability 1 to all other edges that are adjacent to a node in the image of f;
- and probability $\frac{1}{2}$ to all remaining edges.

First note that the image of any edge of G by g is a path of length $\geqslant 3$, whose edges have probability $\frac{1}{2}$ except the first and last one, which have probability 1; further, these paths are disjoint when ranging over the edges of G. Consider a *possible world* G'' of G', i.e., a subgraph with non-zero probability in π. We call an edge (u, v) of G *kept* in G'' if, considering its image in G' by g, and removing the first and last edge, *all* these edges are in G''; we call (u, v) *discarded* in G'' if *none* of these edges are in G''; and it is *broken* if some of these edges, but not all, are in G''. We call G'' *broken* if some edge of G is broken in G''.

Let n be the number of edges of G and m be the number of edges of G' of non-zero probability under π. The possible worlds of G' under π are partitioned into broken and non-broken worlds. We claim that the probability that a world is non-broken is 2^{3n-m}. To see why, observe that, as the edges of G' corresponding to each edge of G are disjoint sets, the events "e is non-broken in G''" and "e' is non-broken in G''", for $e \neq e'$ two edges of G, are pairwise independent. Hence, the probability that a world of G' is non-broken is the product of the probabilities that each edge is non-broken. For an edge e of G mapped by g (including the first and last edge) to a path of length l in G' (note that $l \geqslant 3$ by our use of G^3), the probability that e is non-broken in G' is exactly $2/2^{l-2} = 2^{3-l}$, because the edges (excluding the first and last one) have 2^{l-2} possible worlds, only two of which are non-broken. So the probability that a world is non-broken is 2^{3n-m} (remember that each edge of G' is in the image of exactly one edge of G).

Now, observe that the conjunct 1not3 of q'_{h} holds in a world of G' iff it is broken. Indeed, if a world G'' is broken, there must be a path p of edges of G' (corresponding to an edge of G minus the first and last edges) where one edge was kept in G'' and one was discarded in G'', and we can take two such edges to be adjacent: if there are no two adjacent edges in p with one kept and one discarded, then either all edges in p are kept or all are discarded. Calling u the common endpoint of these two edges, u must have degree 2 in G' (as a node on the path p), and it has degree 1 in G''. Now clearly its one adjacent vertex (the other endpoint of the kept edge) does not have degree 3, as the only elements of degree 3 in G' (hence in G'') are those in the image of f, and u is within a path in the image of g with the first and last edges removed. Hence, 1not3 holds. Conversely, if 1not3 holds, then considering a witness vertex, it can neither be in the image of f (those have degree 3 in G' and in G'' because their adjacent edges are non-probabilistic), nor adjacent to such a vertex (because those are always adjacent to a degree-3 vertex), nor among nodes not in the image of g (as their incident edges have probability 0, such nodes have no incident edge). Therefore, it must be strictly within a path of the image of g minus the first and last edges; and it is clear that this can only happen in G'' if the preimage edge of G is broken in G''.

Now, we claim that the probability, given that a world is non-broken, that it satisfies q'_{h}, is exactly $M/2^n$, where M is the number of matchings of G. To see why, first note that q'_{h} holds in non-broken worlds iff 1not3 does not hold, by the previous paragraph. Now, observe that there is a clear bijection between subgraphs of G and non-broken possible worlds of G', and we claim that $G''' \subseteq G$ is a matching iff the corresponding possible world of G' does not satisfy 3twonot1. Indeed, if G''' is not a matching then some vertex u of G has two neighbors in G''', and, in the corresponding possible world G'' of G', $u' := f(u)$ has two neighbors with degree > 1 (they are adjacent to u' and to the second edge of the path coding the corresponding incident edge of u in G'''), hence u' witnesses that 3twonot1 holds in G''. Conversely, if a possible world G'' of G' satisfies 3twonot1, the witnessing vertex u' has two neighbors which are each adjacent to an edge, indicating that, letting $u := f^{-1}(u')$, the vertex u has two neighbors in the corresponding subgraph G''' of G, so that G''' is not a matching. Hence, among the 2^n non-broken possible worlds of G' (which all have same probability), there are exactly M that do not satisfy 3twonot1, and thus satisfy q'_{h}.

It follows that the probability $\pi(q'_{\mathrm{h}}, G')$ is the probability that a world of G' is non-broken *and* that, knowing that the world is non-broken, the corresponding subgraph of G is a matching. Hence, we have:

$$\pi(q'_{\mathrm{h}}, G') := 2^{3n-m} \times \frac{M}{2^n}$$

whence it follows that:

$$M = 2^{m-2n} \times \pi(q'_{\mathrm{h}}, G')$$

As the values under consideration are in polynomial size in $|G'|$ (i.e., the *values* themselves are singly exponential in $|G'|$), we can indeed compute M from $\pi(q'_{\mathrm{h}}, G')$ in PTIME, as claimed. This shows the desired result. □

We now fix the treewidth-constructible family $\mathcal{I}$ of unbounded treewidth, and call $\mathcal{A}$ the algorithm which, given a value $k \in \mathbb{N}$ in unary, computes in PTIME an instance of $\mathcal{I}$ with treewidth $\geqslant k$. We can now prove the hardness direction of the second part of Theorem 6.1.2:

Proof of Theorem 6.1.2. We reduce in randomized polynomial time the #3PM problem, which is hard by Lemma 6.1.6, to the probability evaluation problem for q_{h} on $\mathcal{I}$.

Consider an instance of #3PM, namely, a 3-regular planar graph G (clearly we can assume without loss of generality that it has no isolated vertices). Construct $H := G^3$ from G in PTIME.

Let c be as in Lemma 6.1.4; letting $n := |V(H)| = O(|V(G)|)$, the value n^c is polynomial in $|G|$ and we can write it in unary in polynomial time (as n can be written in unary in linear time in the input). Thanks to treewidth-constructibility, we can apply algorithm $\mathcal{A}$ to n^c to compute, in PTIME in $|G|$, an instance I in $\mathcal{I}$ with treewidth $\geqslant n^c$; its Gaifman graph G' has the same treewidth, so that, by Lemma 6.1.4, we can compute in randomized PTIME in I (hence in randomized PTIME in $|G|$) an embedding (f, g) of H in G'.

Using Lemma 6.1.8, compute in PTIME the valuation π of G' such that we can compute in PTIME from G' and $\pi(q'_{\mathrm{h}}, G')$ the number of matchings of G, which is the quantity we are interested in. We define from π' a probability valuation on I as follows:

- For all facts of I that correspond to no edge in G', i.e., facts of the form $R(a)$ and $S(a, a)$, give them probability 0.
- For each edge (u, v) of the Gaifman graph G', choose a *single* witness fact for this edge, and give it probability $\pi(u, v)$; give probability 0 to all other facts for this edge.

There is now a clear bijection between the possible worlds of $\pi'(I)$ with probability > 0 and those of $\pi(G)$, which preserves probability, and it is immediate by construction that the Gaifman graph of a possible world I' of $\pi'(I)$ is exactly the graph which is the corresponding possible world of $\pi(G)$. We conclude because it is immediate that an instance satisfies q_{h} iff its Gaifman graph satisfies q'_{h}, so that we can compute in PTIME our desired number of matchings from G' and $\pi'(q'_{\mathrm{h}}, I)$; and the evaluation of $\pi'(q'_{\mathrm{h}}, I)$ is precisely the problem we are reducing to. □

6.1.4 Discussion

We discuss our choice of hard queries and an alternate problem phrasing where instance families are given with their probabilities.

6.1.4.1 Choice of Hard Query

Not only is our query q_{h} independent from the class of instances $\mathcal{I}$, but it is also an FO query, so, in the *non-probabilistic* setting, its data complexity on any instance is in AC^0. In fact, our choice of q_{h} has also *linear-time* data complexity: one can determine in linear time in an input instance I whether $I \models q_{\mathrm{h}}$. This contrasts sharply with the $\mathrm{FP}^{\#\mathrm{P}}$-completeness (under RP reductions) of *probability evaluation* for q_{h} on *any* unbounded-treewidth instance class (if it is treewidth-constructible).

The query q_{h}, however, is not monotone. We will show in Section 6.5 a dichotomy on OBDD representations which applies to a monotone FO query (actually a query in $\mathrm{UCQ}^{\neq}$). For Theorem 6.1.2, we do not know whether we can do the same, but we know that we can use instead a monotone MSO query. More specifically, we

show that we can use instead a query in C2RPQ$^{\neq}$, the class of *conjunctive two-way regular path queries* [Calvanese, De Giacomo, Lenzerini, and Vardi 2000; Calvanese, De Giacomo, and Vardi 2005] where we additionally allow disequalities between variables.

Indeed, consider the query:

$$q_{\mathrm{RPQ}} := \exists \mathbf{xyz}\ W(x_0, x_1, x_2, x_3) \wedge W(y_0, y_1, y_2, y_3) \wedge W(z_0, z_1, z_2, z_3) \\ \wedge \mathrm{Adj}^*(x_0, y_0) \wedge \mathrm{Adj}^*(x_0, z_0) \wedge x_0 \neq y_0 \wedge x_0 \neq z_0 \wedge y_0 \neq z_0$$

where Adj is as before a shorthand for the disjunction of all binary relations in the signature and their inverses (definable in C2RPQ), and where W is:

$$W(x_0, x_1, x_2, x_3) := \mathrm{Adj}(x_0, x_1) \wedge \mathrm{Adj}(x_0, x_2) \wedge \mathrm{Adj}(x_0, x_3) \wedge \bigwedge_{i \neq j} x_i \neq x_j$$

Analogously to Lemma 6.1.8, we can show that for any 3-regular planar graph G and graph G' having G^3 has a topological minor, there is a valuation π'' such that the number of matchings of G can be computed in PTIME from $\pi''(q_{\mathrm{RPQ}}, G')$. Indeed, define the valuation π'' as π in the proof of that lemma, but assigning probability $1/2$ to *one* edge per path encoding an edge of G, and probability 1 to the others. We show that $\pi''(q_{\mathrm{RPQ}}, G')$ is $1 - M/2^n$, where M is the number of matchings of G and n its number of edges. Indeed, we show there is an obvious bijection between subgraphs of G and non-zero-probability valuations of G' under π''.

For a subgraph which is *not* a matching, q_{RPQ} holds, as witnessed by mapping x_0 to the node of G' representing the node of G with two adjacent edges, mapping y_0 and z_0 to the nodes of G' representing the two neighbors of x in G, mapping the other x_i, y_i and z_i to the neighbors of x_0, y_0, z_0 in G^3.

Conversely, whenever q_{RPQ} holds, x_0, y_0 and z_0 must be mapped to nodes representing nodes of G (they are the only nodes with degree 3, and the images of x_0, y_0 and z_0 have degree 3, as evidenced by the disequalities); further, by the disequalities, x_0, y_0 and z_0 must be different. Further, y_0 and z_0 are accessible from x_0, which means that in the subgraph of G, there is a path from of node of G to two different nodes of G; this means this subgraph is not a matching as, in a matching, connected components have size at most 2.

We conclude that we can show Theorem 6.1.2 using q_{RPQ} instead of q_{h}, proceeding exactly as in our original proof above.

6.1.4.2 Providing Valuations with the Instances

When we fix the instance family $\mathcal{I}$, the probability valuation is not prescribed as part of the family, but can be freely chosen. If the instances of $\mathcal{I}$ were provided with their probability valuations, or if probability valuations were forced to be $1/2$, then it is unlikely that an equivalent to Theorem 6.1.2 would hold.

Indeed, fix *any* query q such that, given any instance I, it is in #P to count how many subinstances of I satisfy q; e.g., let q be a CQ. Consider a family $\mathcal{I}$ of instances *with valuations* such that there is only one instance in $\mathcal{I}$ per encoding length: e.g., take the class of R-grids with probability $1/2$ on each edge, for some binary relation R. Consider the problem, given the *length* of the encoding of an instance I (written in unary), of computing how many subinstances of I satisfy q. This problem is in the class $\#\mathrm{P}_1$ Valiant 1979. Hence, the probability computation problem for q on $\mathcal{I}$ is in

$\mathrm{FP}^{\#\mathrm{P}_1}$: rewrite the encoding of the input instance I to a word of the same length in a unary alphabet, use the $\#\mathrm{P}_1$-oracle to compute the number of subinstances, and normalize the result by dividing by the number of possible worlds of I.

It thus seems unlikely that probabilistic evaluation of q on $\mathcal{I}$ with its valuations is #P-hard, so that our dichotomy result probably does not adapt if input instance families are provided with their valuations.

6.2 Dichotomy on Non-Probabilistic Evaluation

Our probability evaluation problem allows the valuation to set edges to have probability 0. We can thus restrict to any subinstance of an instance in the class $\mathcal{I}$. In other words, the freedom to choose valuations in probability evaluation gives us at least the possibility of choosing subinstances for non-probabilistic query evaluation.

For this reason, in this section, we study the (non-probabilistic) query evaluation problem (remember its definition in Section 2.6) on instance classes $\mathcal{I}$ which are *closed under taking subinstances* (or *subinstance-closed*), namely, for any $I \in \mathcal{I}$ and $I' \subseteq I$, we have $I' \in \mathcal{I}$.

We will show dichotomy results for this problem on unbounded-treewidth instance families as before, though we will use an MSO query rather than a FO query. We give two phrasings of our results. The first one still requires treewidth-constructibility, and shows hardness for every level of the polynomial hierarchy, again under RP reductions: we show it in Section 6.2.1.

The second phrasing is inspired by the results of [Ganian, Hliněný, et al. 2014]: it relies on complexity assumptions (namely, the non-uniform exponential time hypothesis) but works with a weaker notion of constructibility, namely, it requires treewidth to be strongly unbounded poly-logarithmically. We present it, introduced by a discussion of the connection between these phrasings, in Section 6.2.2.

6.2.1 Hardness Formulation

Our first dichotomy is as follows, phrased using treewidth-constructibility. In this result, Σ^{P}_i denotes the complexity class at the i-th existential level of the polynomial hierarchy.

Theorem 6.2.1. *Let σ be an arbitrary arity-2 signature. Let $\mathcal{I}$ be a class of σ-instances which is treewidth-constructible and subinstance-closed. The following dichotomy holds:*

- *If there exists $k \in \mathbb{N}$ such that $\mathrm{tw}(I) \leqslant k$ for every $I \in \mathcal{I}$, then for every GSO query q, the evaluation problem for q on $\mathcal{I}$ is solvable in linear time.*
- *Otherwise, for each $i \in \mathbb{N}$, there is an MSO query q^i_{h} (depending only on σ, not on $\mathcal{I}$) such that the evaluation problem for q^i_{h} on $\mathcal{I}$ is Σ^{P}_i-hard under RP reductions.*

The upper bound is by Courcelle's results [Courcelle 1990; Flum, Frick, and Grohe 2002], so our contribution is the hardness part, which we now prove. The only thing to change relative to the proof of Theorem 6.1.2 is the hard problems from which we reduce. We reduce from hard problems on planar $\{1, 3\}$-regular graphs,

which we obtain from the *alternating coloring problem* as [Ganian, Hliněný, et al. 2014; Ganian et al. 2010], restricted to such graphs using techniques shown there, plus an additional construction to remove vertex labellings. Here is our formal claim about the existence of such hard problems:

Lemma 6.2.2. *For any $i \in \mathbb{N}$, there exists an MSO formula ψ_i on the signature of graphs such that the evaluation of ψ_i on planar $\{1,3\}$-regular graphs is Σ_i^{P}-hard. Moreover, for any such graph G, we have $G \models \psi_i$ iff $G' \models \psi_i$ for any subdivision G' of G.*

To prove this lemma, we will need an additional lemma that allows us to move from labeled graphs (as in [Ganian, Hliněný, et al. 2014]) to unlabeled graphs. We focus on graphs labeled by a Boolean value, i.e., there is a set of nodes that are *labeled*, and the others are *unlabeled*. We stress that this lemma would also apply in the setting of [Ganian, Hliněný, et al. 2014], and it is *not* thanks to this result that we can do away with the vertex labels used in [Ganian, Hliněný, et al. 2014]: their use of vertex labels is required for deeper reasons, and it is our use of the polynomial bounds of [Chekuri and Chuzhoy 2014a] that allows us to avoid them.

Lemma 6.2.3. *For any MSO formula φ on* labeled *graphs (i.e., φ has a symmetric binary predicate E to test edges and a unary predicate L to test for labeled vertices), we can construct in PTIME an MSO formula ψ on unlabeled graphs, such that for any labeled graph G, we can construct in PTIME a graph G' such that $G \models \varphi$ iff $G' \models \psi$.*

Proof. We first define the PTIME translation τ of the labeled graph $G = (V, E, L)$ (where $L \subseteq V$ is the set of labeled vertices) into a graph $\tau(G) = (V', E')$. We set $V' = V \sqcup \{d_1, a_1, a_2, b_1, b_2, b'_1, b'_2, l\}$, where the additional vertices are fresh and pairwise distinct. We set E' to be E plus the following edges:

- An edge between a_1 and every $v \in V$, and an edge between a_2 and every $v \in V$.
- An edge between a_1 and a_2.
- An edge between a_i and b_i, and between a_i and b'_i, for $i \in \{1, 2\}$.
- An edge between b_1 and b_2, and an edge between b'_1 and b'_2.
- An edge between d_1 and l.
- An edge between l and every labeled vertex $v \in L$.

Observe that the following claims hold about G':

1. *d_1 and l are the only nodes of G' that may have degree* 1. Indeed, all nodes in V are adjacent to a_1 and a_2, and the claim is clear for the a_i, b_i and b'_i (considering that each is connected to at least two of the new nodes)
2. *The nodes of L are exactly those with a neighbor that has another neighbor of degree* 1. Indeed, all labeled nodes are adjacent to l, which is adjacent to d_1, which clearly always has degree 1. For the converse, let us distinguish the case where $L = \emptyset$, in which case only d_1 and l have degree 1, and they form a separate connected component so no node has a neighbor with another

neighbor of degree 1. Assuming $L \neq \emptyset$, l is adjacent to some labeled node and to d_1, so it has degree $\geqslant 2$. Hence, by point 1, the only node with degree 1 is d_1, its only neighbor is l, and its other neighbors are the nodes of L, so indeed only nodes of L satisfy the condition.

3. *a_1 and a_2 are adjacent to all nodes of G' except exactly four of them.* This is immediate: for a_i, the missing nodes are d_1, l, and b_{3-i} and b'_{3-i}.

4. *All vertices of G' except the a_i have at least five nodes to which they are not adjacent.* Indeed, every $v \in V$ is non-adjacent to d_1 and b_i and b'_i for $i \in \{1, 2\}$, hence, 5 nodes. d_1 and l are non-adjacent to the a_i and b'_i, so at least 6 nodes. The b_i are non-adjacent to d_1, l, the two b'_i, and a_{3-i}, so at least 5 nodes. The same argument works for the b'_i.

5. *a_1 and a_2 are exactly the nodes of G' which are non-adjacent to exactly four nodes.* This follows immediately from the two previous points.

6. *The b_i and b'_i are the only nodes adjacent to exactly one of a_1, a_2.* It is immediate by construction that each b_i is adjacent to a_i but not to a_{3-i}, and likewise for b'_i. Now, d_1 and l are adjacent to no a_i, and the vertices in V are adjacent to both a_1 and a_2.

We now write auxiliary MSO predicates. For any *constant* $c \in \mathbb{N}$, it is clear that we can write in MSO (actually, in FO) a formula $\deg_c(x)$ that tests whether x has degree exactly c, and a formula $\overline{\deg}_c(x)$ that tests whether the number of elements that are *not* adjacent to x is exactly c. We write:

$$
\begin{aligned}
L(w) &:= \exists xy\ \deg_1(x) \wedge E(x,y) \wedge E(y,w) \wedge x \neq w \\
G_1(x) &:= \overline{\deg}_4(x) \\
G_2(x) &:= \exists y\ G_1(y) \wedge E(x,y) \wedge (\forall y'\ G_1(y') \wedge E(x,y') \Rightarrow y = y') \\
G_3(x) &:= \deg_1(x) \vee (\exists y\ E(x,y) \wedge \deg_1(y)) \\
V(x) &:= \neg G_1(x) \wedge \neg G_2(x) \wedge \neg G_3(x)
\end{aligned}
$$

We now claim that for any labeled graph $G = (V, E, L)$, writing $\tau(G) = (V', E')$:

- We have $\tau(G) \models L(w)$ iff $w \in L$. Indeed, we have $\tau(G) \models L(w)$ iff w has a neighbor that has another neighbor of degree 1, so point 2 of the list of claims concludes.

- We have $\tau(G) \models V(v)$ iff $v \in V$. Indeed, we have $\tau(G) \models G_1(v)$ iff v has exactly 4 non-adjacent elements in G', so that point 5 of the list of claims ensures that this is precisely the case for the a_i. This implies that $\tau(G) \models G_2(v)$ iff v is adjacent to exactly one of the a_i, i.e., by point 6 of the list of claims, iff v is one of the b_i or b'_i. Now, $\tau(G) \models G_3(v)$ iff v has degree 1 or is adjacent to a degree 1 node, and by point 1 of the list of claims (noticing also that whenever l has degree 1 then its one neighbor is d_1), this is the case iff v is l or d_1. Hence, $\tau(G) \models G(v)$ iff $v \in V$.

We conclude that we can construct ψ in PTIME from φ by relativizing every quantified formula of the form $\forall x\ \chi(x, \mathbf{y})$ into a formula $\forall x\ V(x) \to \chi(x, \mathbf{y})$, every quantified formula of the form $\exists x\ \chi(x, \mathbf{y})$ into a formula $\exists x\ V(x) \wedge \chi(x, \mathbf{y})$, by replacing each

occurrence of the unary predicate L by the subformula defined above, and by keeping E unchanged. By construction, for any labeled graph G, we have $G \models \varphi$ iff $\tau(G) \models \psi$, as we have shown that the interpretation of the subformulae in ψ on $\tau(G)$ is exactly that of the predicates in φ on G. This concludes the proof. □

We can now prove our lemma that states the existence of hard problems:

Proof of Lemma 6.2.2. We fix $i \in \mathbb{N}$. Without loss of generality, we assume i to be odd (if not, we will show $\Sigma^{\mathrm{P}}_{i+1}$-hardness, which implies Σ^{P}_i-hardness). We reduce from the *alternating coloring problem* or Σ_i3COL problem defined in [Ganian, Hliněný, et al. 2014], which is complete for Σ^{P}_i under polynomial-time reductions by [Ganian, Hliněný, et al. 2014, Theorem 5.2].

By [Ganian, Hliněný, et al. 2014, Lemma 5.3], an instance of Σ_i3COL can be given as a graph with $i+3$ label predicates and such that, on such a labeled graph structure, there exists a formula φ_i that expresses Σ_i3COL.

By [Ganian, Hliněný, et al. 2014, Lemma 5.4], we can construct in polynomial-time from φ_i an MSO formula φ'_i on labeled graphs with a *single* label predicate such that any graph with $i+3$ label predicates G can be rewritten in polynomial time to a graph G' with a single label predicate such that $G \models \varphi_i$ iff $G' \models \varphi'_i$.

By our Lemma 6.2.3, we can construct in polynomial-time from φ'_i an MSO formula φ''_i on unlabeled graphs such that any labeled graph G' with a single label predicate can be rewritten in polynomial time to an unlabeled graph G'' such that $G \models \varphi_i$ iff $G'' \models \varphi''_i$.

Finally, by [Ganian et al. 2016, Theorem 5.1], we can construct in polynomial-time from φ''_i an MSO formula ψ_i such that any graph G'' can be rewritten in polynomial time to a $\{1,3\}$-regular planar graph H such that $H \models \psi_i$ iff $G'' \models \varphi''_i$, and such that ψ_i is invariant under subdivisions: for any graph H', we have $H' \models \psi_i$ iff any subdivision H'' of H' is such that $H'' \models \psi$. This means that the evaluation of ψ_i is Σ^{P}_i-hard on planar $\{1,3\}$-regular graphs and that the additional properties that we claimed are verified, which concludes the proof. □

Thanks to this lemma, we can now prove the hardness part of our dichotomy result:

Proof of Theorem 6.2.1. Once again, the first claim of the theorem is known (it is a consequence of Courcelle's results, [Courcelle 1990]), so we only prove the second claim.

Fix $i \in \mathbb{N}$. Using Lemma 6.2.2, we reduce from the evaluation of ψ_i on planar $\{1,3\}$-regular graphs. We take q^i_{h} to be identical to ψ_i except that the E predicates (which are symmetric) are replaced with the Adj subformula asserting adjacency in the Gaifman graph, as in Section 6.1.

Fix the class $\mathcal{I}$. Let G be a planar $\{1,3\}$-regular graph. As in Section 6.1, compute in randomized PTIME an embedding (f,g) of G in the Gaifman graph G' of an instance I' of $\mathcal{I}$, and let $I \in \mathcal{I}$ be the subinstance of I' obtained by keeping exactly the facts that correspond to edges of G' in the image of g, so the Gaifman graph of I is a subdivision of G, i.e., each edge of G corresponds to a path in I. It is clear that I satisfies q^i_{h} iff $G \models \psi_i$, i.e., iff $G \models \psi_i$, by the property on ψ_i. This completes the reduction. □

As a closing remark, note that Theorem 6.2.1 relies crucially on the class $\mathcal{I}$ being *subinstance-closed*. Otherwise, considering the class $\mathcal{I}$ of cliques of a single binary relation E, this class is clearly unbounded-treewidth and treewidth-constructible, yet it has bounded clique-width so MSO query evaluation has linear data complexity on this class [Courcelle, Makowsky, and Rotics 2000].

Further, the hypothesis of *treewidth-constructibility* is also crucial. Without this assumption, [Makowsky and Marino 2003, Proposition 32] shows the existence of graph families of unbounded treewidth which are subinstance-closed yet for which MSO query evaluation is in PTIME.

6.2.2 Alternate Formulation

We now give an alternative phrasing of Theorem 6.2.1 which connects it to the existing results of [Kreutzer and Tazari 2010; Ganian, Hliněný, et al. 2014]. Table 6.1 tersely summarizes their results in comparison to our own results and other related results. As [Kreutzer and Tazari 2010; Ganian, Hliněný, et al. 2014] are phrased in terms of graphs, and not arbitrary arity-2 relational instances, we do so as well in this section.

[Kreutzer and Tazari 2010; Ganian, Hliněný, et al. 2014] show the intractability of MSO on any subgraph-closed unbounded-treewidth families of graphs, under finer notions than our *treewidth-constructibility*. [Kreutzer and Tazari 2010] proposed the notion of families of graphs with treewidth *strongly unbounded poly-logarithmically* and showed that MSO_2 (MSO with quantifications over both vertex- and edge-sets) over any such graph families is not fixed-parameter tractable in a strong sense (it is not in XP), unless the exponential-time hypothesis (ETH) fails. [Ganian, Hliněný, et al. 2014] proved a related result, introducing the weaker notion of *densely unbounded poly-logarithmically* but requiring graph families to be closed under *vertex relabeling*; in such a setting, [Ganian, Hliněný, et al. 2014, Theorem 4.1] shows that MSO (with vertex labels) cannot be fixed-parameter quasi-polynomial unless the *non-uniform* exponential-time hypothesis fails.

These two results of [Kreutzer and Tazari 2010] and [Ganian, Hliněný, et al. 2014] are incomparable: [Kreutzer and Tazari 2010] requires a stronger unboundedness notion (strongly unbounded vs densely unbounded) and a stronger query language (MSO_2 vs MSO), but it does not require vertex relabeling, and makes a weaker complexity theory assumption (ETH vs non-uniform ETH). See the Introduction of [Ganian, Hliněný, et al. 2014] for a detailed comparison.

Our Theorem 6.2.1 uses MSO and no vertex labeling, but it requires *treewidth-constructibility*, which is stronger than densely/strongly poly-logarithmic unboundedness: strongly unboundedness only requires constructibility in $o(2^n)$ and densely unboundedness does not require constructibility at all. The advantage of treewidth-constructibility is that we were able to show *hardness* of our problem (under RP reductions), without making *any* complexity assumptions. However, if we make the same complexity-theoretic hypotheses as [Ganian, Hliněný, et al. 2014], we now show that we can phrase our results in a similar way to theirs, and thus strengthen them.

We accordingly recall the definition of densely poly-logarithmic unboundedness:

Definition 6.2.4. [Ganian, Hliněný, et al. 2014, Definition 3.3] A graph class $\mathcal{G}$ has treewidth *densely unbounded poly-logarithmically* if for all $c > 1$, for all $m \in \mathbb{N}$, there exists a graph $G \in \mathcal{G}$ such that $\mathrm{tw}(G) \geqslant m$ and $|V(G)| < O(2^{m^{1/c}})$. ◁

Table 6.1: Summary of results for non-probabilistic query evaluation: if a graph class has unbounded treewidth in *some* sense, is closed under *some* operations, and has (in data complexity) tractable model checking in *some* sense for *some* logic, then *some* complexity assumption is violated

Logic	**Unboundedness**	**Closure**	**Tractability**	**Consequence**	**Source**	
MSO	unbounded	subgraph	PTIME	no violation: holds for some classes	[Makowsky and Marino 2003]	Prop. 32
MSO_2	unbounded	subgraph	PTIME	no violation: holds for some classes	[Kreutzer and Tazari 2010]	(remark)
∃MSO	unbounded	topolog. minors	PTIME	P = NP	[Makowsky and Marino 2003]	Thm. 11
MSO_2	strongly unb. polylog.	subgraph	PTIME	PH ⊆ DTIME($2^{o(n)}$)	[Kreutzer and Tazari 2010]	Thm. 1.2
MSO	densely unb. polylog.	subgr., vert. lab.	quasi-poly	PH ⊆ DTIME($2^{o(n)}$)/SUBEXP	[Ganian, Hliněný, et al. 2014]	Thm. 5.5
MSO	unb., treewidth-constr.	subgraph	PTIME	PH ⊆ RP	**here**	Thm. 6.2.1
MSO	densely unb. polylog.	subgraph	quasi-poly	PH ⊆ DTIME($2^{o(n)}$)/SUBEXP	**here**	Thm. 6.2.5

We now state our intractability result on densely unbounded poly-logarithmically graph classes. It is identical to [Ganian, Hliněný, et al. 2014, Theorem 5.5] but applies to arbitrary MSO formulae, without a need for vertex relabeling: in the result, PH denotes the polynomial hierarchy. This result answers [Grohe 2007, Conjecture 8.3], as we claimed in the introduction to this chapter.

Theorem 6.2.5. *Unless* PH $\subseteq$ DTIME$(2^{o(n)})$/SubEXP*, there is no graph class* $\mathcal{G}$ *satisfying all three properties:*

a) $\mathcal{G}$ *is closed under taking subgraphs;*

b) the treewidth of $\mathcal{G}$ *is densely unbounded poly-logarithmically;*

c) the evaluation problem for any MSO query q *on* $\mathcal{G}$ *is quasi-polynomial, i.e., in time* $O(n^{\log^d n \times f(|q|)})$ *for* $n = |V(G)|$*, an arbitrary constant* $d \geqslant 1$*, and some computable function* f*.*

The proof technique is essentially the same as in [Ganian, Hliněný, et al. 2014] up to using the newer results of [Chekuri and Chuzhoy 2014a]. It is immediate that an analogous result holds for probability query evaluation, as standard query evaluation obviously reduces to it (take the valuation giving probability 1 to each fact).

Proof of Theorem 6.2.5. We start with a problem $\mathcal{P}$ in the polynomial hierarchy PH, say in Σ_i^{P}, and show that it can be solved in sub-exponential time with sub-exponential advice if such a graph class $\mathcal{G}$ exists. The structure of the proof follows that of Theorems 4.1 and 5.5 of [Ganian, Hliněný, et al. 2014], except that we use the polynomial bound from [Chekuri and Chuzhoy 2014a] (restated here as Lemma 6.1.4) on the required treewidth as a function of the size of the desired topological minor.

Let F be a polynomial-time reduction from $\mathcal{P}$ to the problem ψ_i of Lemma 6.2.2, which is hard for Σ_i^{P}. Let x be an instance of $\mathcal{P}$, and $H := F(x)$; as F is polynomial-time, we know that $|H| \leqslant |x|^l$ and $|V(H)| \leqslant |x|^l$ for some l. Let α be the constant c of Lemma 6.1.4. Pose $m := |x|$.

Since $\mathcal{G}$ has treewidth densely unbounded poly-logarithmically, letting d be given by our assumption on $\mathcal{G}$, choose $c := l \cdot \alpha \cdot (d+2)$, and take $G_m \in \mathcal{G}$ such that $\mathrm{tw}(G_m) \geqslant m^{l \cdot \alpha}$ and $|V(G_m)| < O(2^{m^\beta})$, where $\beta < 1$ is defined as $\beta := \frac{1}{d+2}$.

Since $\mathrm{tw}(G_m) \geqslant m^{l\alpha} \geqslant |V(H)|^\alpha$, by Lemma 6.1.4, H is a topological minor of G_m. Let (f_m, g_m) be an embedding of H in G_m.

We consider the advice function $A(m) := (G_m, f_m, g_m)$. As $|G_m| \leqslant |V(G_m)|^2 < O(2^{2m^\beta})$ and the size of a function is smaller than the product of the size of the domain and range of that function, the size of the advice $A(m)$ is $< O(2^{4m^\beta})$, which is asymptotically sub-exponential in m because $\beta < 1$.

The embedding (f_m, g_m) maps H to a subgraph G' of G such that H is isomorphic to a subdivision of G'. As $\mathcal{G}$ is closed under taking subgraphs, and as ψ_i is not sensitive to taking subdivisions by Lemma 6.2.2, we can decide ψ_i on H by evaluating ψ_i on G' in quasi-polynomial time, i.e., in time $O(2^{m^\beta m^{d\beta} f(|\psi_i|)})$ for d, f given by the assumptions on $\mathcal{G}$.

As $\beta = \frac{1}{d+2}$, this gives us an algorithm to decide $\mathcal{P}$ on input x in time $O(2^{o(|x|)})$ with advice of size asymptotically sub-exponential in m. This concludes the proof. □

6.3 Dichotomy on Match Counting

We now study a second non-probabilistic problem on subgraph-closed instances, namely, the problem of *match counting*, i.e., counting how many assignments satisfy a *non-Boolean MSO formula* (recall its definition in Section 2.6, and recall that it should not be confused with the model counting problem). To our knowledge, no dichotomy-like result on match counting for MSO queries was known before; we show the following one in this section:

Theorem 6.3.1. *Let σ be an arbitrary arity-2 signature. Let $\mathcal{I}$ be a subinstance-closed and treewidth-constructible class of relational instances over σ. The following dichotomy holds:*

- *If there is $k \in \mathbb{N}$ such that $\mathrm{tw}(I) \leqslant k$ for every $I \in \mathcal{I}$, then for every MSO query $q(\mathbf{X})$ with free second-order variables, the counting problem for q on $\mathcal{I}$ is solvable in ra-linear time.*
- *Otherwise, there is an MSO query $q'_{\mathrm{h}}(X)$ (depending only on σ, not on $\mathcal{I}$) with one free second order variable such that the counting problem for q'_{h} on $\mathcal{I}$ is $FP^{\#P}$-complete under RP reductions.*

The first claim is shown in [Arnborg, Lagergren, and Seese 1991]; we gave a proof of it using our methods as Theorem 4.6.1. Hence, we focus again on the second part. We reduce from the problem #3PCH of counting *Hamiltonian cycles* in an input 3-regular planar graph.

Definition 6.3.2. A *Hamiltonian cycle* in a graph G is a cycle $E(x_0, x_1), \ldots, E(x_{n-1}, x_n)$, with $x_0 = x_n$, such that $x_1, \ldots, x_n$ are precisely the vertices of G.

The #3PCH problem asks, given a planar 3-regular graph G, its number of Hamiltonian cycles. ◁

Lemma 6.3.3. *[Liskiewicz, Ogihara, and Toda 2003, Corollary 6] #3PCH is $FP^{\#P}$-complete.*

Let us define a query $q'_{\mathrm{h}}(S)$ on graphs. It will apply to the result H of subdividing each edge of an input 3-regular planar graph G exactly once, namely, $H := G^2$, the *incidence graph* of G. Note that the vertices of H have either degree 3 (in which case they correspond to vertices of the 3-regular graph G) or degree 2 (in which case they correspond to edges of G).

The query $q'_{\mathrm{h}}(S)$ therefore asks that S is a subset of edges (intuitively, of H) which is *connected*, which contains all degree-3 elements of the graph (so all original vertices of G), and which ensures that each element of S is adjacent to precisely two other elements of S (so it stands for a path of G).

Formally, we define:

$$\begin{aligned}
\mathrm{Deg3}(x) &:= \exists y_1 y_2 y_3 \; E(x,y_1) \wedge E(x,y_2) \wedge E(x,y_3) \wedge y_1 \neq y_2 \wedge y_1 \neq y_3 \wedge y_2 \neq y_3 \\
&\quad \wedge \forall y \; E(x,y) \Rightarrow (y = y_1 \vee y = y_2 \vee y = y_3) \\
\mathrm{TwoN}(x,S) &:= \exists y_1 y_2 \; E(x,y_1) \wedge E(x,y_2) \wedge y_1 \neq y_2 \wedge y_1 \in S \wedge y_2 \in S \\
&\quad \wedge \forall y \; (E(x,y) \wedge y \in S) \Rightarrow (y = y_1 \vee y = y_2) \\
\mathrm{Conn}(S) &:= \forall xy \; x \in S \wedge y \in S \Rightarrow \forall S' \; ((x \in S' \\
&\quad \wedge (\forall zw \; z \in S' \wedge E(z,w) \wedge w \in S \Rightarrow w \in S')) \Rightarrow y \in S') \\
q'_{\mathrm{h}}(S) &:= \mathrm{Conn}(S) \wedge (\forall x \; \mathrm{Deg3}(x) \Rightarrow x \in S) \wedge (\forall x \; x \in S \Rightarrow \mathrm{TwoN}(x,S))
\end{aligned}$$

We first show that q'_{h} correctly tests Hamiltonian cycles:

Lemma 6.3.4. *For any graph G, letting $H := G^2$, for any subset S of vertices of H, we have $H \models q'_{\mathrm{h}}(S)$ iff S denotes a Hamiltonian cycle of G, namely, all vertices originally from G are in S, and the additional vertices of S stand for edges of G such that there is a Hamiltonian cycle that uses exactly these edges.*

Proof. Fix G and $H := G^2$. For the backward direction, let S denote a Hamiltonian cycle of G. It is clear that S must be connected, that S must contain all degree-3 vertices of H (which stand precisely for the original vertices in the 3-regular G), and that every vertex of S has precisely two neighbors in S (as each edge of the cycle in G contains two edges, and each vertex is incident to precisely two edges). Hence, we have $H \models q'_{\mathrm{h}}(S)$.

For the forward direction, consider a subset S of vertices of H such that $H \models q'_{\mathrm{h}}(S)$. Choose any element v_1 of S, any adjacent element v_2 of S, and follow the sequence of elements of S by taking, after each element v_i, the adjacent element v_{i+1} in H which is not v_{i-1}. As all elements in S have exactly two adjacent elements in S, this process is well-defined and it must loop back on v_1, and as S is connected this enumerates all elements of S, i.e., as S contains all degree-3 elements of H, it enumerates all vertices of H that stand for vertices of G. The result of this process is thus a Hamiltonian cycle in G that uses precisely the edges corresponding to the additional vertices in S. □

We now prove the lower bound of our dichotomy result on match counting:

Proof of Theorem 6.3.1. We use the query $q''_{\mathrm{h}}(S)$ obtained from $q'_{\mathrm{h}}(S)$ by replacing the relation E with the relation Adj defined in Section 6.1, that tests adjacency in the Gaifman graph for the signature σ. Clearly the counting problem for q''_{h} is in $\mathrm{FP}^{\#\mathrm{P}}$ for any input instance, as can be seen by a PTIME nondeterministic Turing machine that chooses a valuation for the free variable S in nondeterministic PTIME, and then checks in deterministic PTIME whether the chosen S is satisfies $q''_{\mathrm{h}}(S)$. So we only have to show the hardness claim.

Fix a 3-regular planar graph $G = (V, E)$, let $n := |V|$ be its number of vertices, and let us reduce the problem of counting Hamiltonian cycles in G to the problem of match counting of q''_{h} on $H := G^2$.

Using the fact that $\mathcal{I}$ is subinstance-closed, unbounded-treewidth, and treewidth-constructible, compute in randomized PTIME as in the previous sections an instance I of $\mathcal{I}$ whose Gaifman graph is a subdivision of H. By Lemma 6.3.4, it is clear that $q''_{\mathrm{h}}(S)$ holds iff S is the image in I of a Hamiltonian cycle of G. Further, it is easily seen that each such image corresponds to exactly $|V|$ Hamiltonian cycles of G, namely, cycles that differ only by the choice of the starting vertex. Hence, letting N' be the number of matches of q''_{h} on I, the number N of Hamiltonian cycles of G, which is what we wish to compute, is such that $N' = n \cdot N$, so we can clearly compute N in PTIME from N': remember that the number of Hamiltonian cycles is a *singly exponential* value so its size is *polynomial* and arithmetic operations on it are polynomial as well. Thus, the #3PCH instance reduces in randomized PTIME to the query counting problem for q''_{h}. This shows $\mathrm{FP}^{\#\mathrm{P}}$-hardness and concludes the proof. □

Unlike in Theorem 6.1.2, the query q'_{h} does not have tractable model checking (as opposed to probability evaluation). We do not know whether our dichotomy for match counting could be shown with a tractable query.

6.4 Dichotomy for OBDD Representations

We now turn to our second main dichotomy result for this chapter: our Theorem 3.5.2 on the existence of tractable lineage representations as OBDDs is unlikely to extend to milder conditions than bounded-treewidth, because there are even $\mathrm{UCQ}^{\neq}$ queries that have no polynomial-width OBDDs on any unbounded-treewidth input instance with treewidth densely unbounded poly-logarithmically, again on arity-two signatures.

This result can be compared to our first main dichotomy result (Theorem 6.1.2). Our first result applied to the more general task of probability evaluation: if OBDDs for the query on input instances of the family can be computed in PTIME, then probability evaluation is in PTIME as well, but the converse does not hold: see [Jha and Suciu 2013] for counterexamples. However, our first result applied to a query in non-monotone FO, whereas our second result applies to a query in the more restricted class $\mathrm{UCQ}^{\neq}$. Further, as our second result is based on different techniques, it makes a weaker constructibility requirement (but still relies on the minor extraction tools of [Chekuri and Chuzhoy 2014a]).

Here is our dichotomy result on tractable OBDDs:

Theorem 6.4.1. *There exists a constant $d \in \mathbb{N}$ such that the following holds. Let σ be an arbitrary arity-2 signature and $\mathcal{I}$ be a class of σ-instances. Assume there is a function $f(k) = O(2^{k^{1/d}})$ such that, for all $k \in \mathbb{N}$, if $\mathcal{I}$ contains instances of treewidth $\geqslant k$, one of them has size $\leqslant f(k)$. We have the following dichotomy:*

- *If there is $k \in \mathbb{N}$ such that $\mathrm{tw}(I) \leqslant k$ for every $I \in \mathcal{I}$, then for every GSO query q, an OBDD of q on I can be computed in time polynomial in $|I|$.*
- *Otherwise, there is a UCQ$^{\neq}$ query q_{p} (depending on σ but not on $\mathcal{I}$) such that the width of any OBDD of q_{p} on $I \in \mathcal{I}$ (i.e., an OBDD representation of $\mathrm{Prov}(q_{\mathrm{p}}, I)$) cannot be bounded by any polynomial in $|I|$.*

This does *not* require treewidth-constructibility, and imposes instead a slight weakening[2] of densely unbounded poly-logarithmic treewidth. It does not require $\mathcal{I}$ to be subinstance-closed either, unlike in Sections 6.2 and 6.3.

The first part of the theorem is by Theorem 3.5.2, so we will only show the second part. Our choice of $\mathrm{UCQ}^{\neq}$ q_{p}, which will be discussed in more depth in Section 6.5, intuitively tests the existence of a path of length 2 in the Gaifman graph of the instance, i.e., a violation of the fact that the possible world is a matching of the original instance. Formally, we define:

$$q_{\mathrm{p}} := \exists xyz\ \mathrm{Adj}(x, y) \wedge \mathrm{Adj}(y, z) \wedge x \neq z$$

where we use the Adj predicate as in Section 6.1 to test the existence of an edge in the Gaifman graph. It is important to observe that q_{p} is a *connected* $\mathrm{UCQ}^{\neq}$, for which we give a formal definition:

[2] The condition is weaker because we require the subexponentiality to work for some fixed d, not an arbitrary c.

Definition 6.4.2. A $\mathrm{CQ}^{\neq}$ is *connected* if, building the graph G on its atoms that connects those that share a variable (ignoring $\neq$-atoms), G is connected (in particular it has no isolated vertices, unless it consists of a single isolated vertex). A $\mathrm{UCQ}^{\neq}$ is *connected* if all its $\mathrm{CQ}^{\neq}$ disjuncts are *connected*. ◁

Again, remember that, while we know that probability evaluation for q_{p} is $\mathrm{FP}^{\#\mathrm{P}}$-hard if we allow *arbitrary* input instances (as counting matchings reduces to it), our task is to show that q_{p} has no polynomial-width OBDDs when restricting to *any* instance family, a much harder task.

To show this, we draw a link between treewidth and OBDD width for q_{p} on *individual* instances, with the following result (which is specific to q_{p}):

Theorem 6.4.3. *Let σ be an arity-2 signature. There exist constants $d', k_0 \in \mathbb{N}$ such that for any instance I on σ of treewidth $\geqslant k_0$, the width of an OBDD for q_{p} on I is $\geqslant 2^{(\mathrm{tw}(I))^{1/d'}}$.*

Indeed, once this is shown, then the lower bound of Theorem 6.4.1 can be derived as follows:

Proof of Theorem 6.4.1. Fix $d := d' + 1$. Fix the signature σ, the family $\mathcal{I}$ and the function f giving the size bound. Let β be such that $f(k) \leqslant \beta \cdot 2^{k^{1/d}}$ for all $k \in \mathbb{N}$.

Assume by way of contradiction that we have a polynomial bound on the size of OBDDs for q_{p} on $\mathcal{I}$: there are α and $c \in \mathbb{N}$ such that, for all $I \in \mathcal{I}$, there is an OBDD of width $\leqslant \alpha \cdot |I|^c$ for q_{p} on I. For any $k \geqslant k_0$ (with k_0 given by Theorem 6.4.3), let us build $I_k \in \mathcal{I}$ such that:

1. $k \leqslant \mathrm{tw}(I)$, using the fact that $\mathcal{I}$ has unbounded-treewidth;
2. $|I| \leqslant f(\mathrm{tw}(I))$, using the hypothesis in Theorem 6.4.1.

Let w_k the smallest width of an OBDD for q_{p} on I_k: by our assumption, we have $w_k \leqslant \alpha \cdot |I_k|^c$. By Theorem 6.4.3, we have $w_k \geqslant 2^{(\mathrm{tw}(I_k))^{1/d'}}$; and we have $|I_k| \leqslant \beta \cdot 2^{(\mathrm{tw}(I_k))^{1/d}}$ by the bound on f, so $|I_k|^c \leqslant \beta^c \cdot 2^{c \cdot (\mathrm{tw}(I_k))^{1/d}}$, which means $w_k \leqslant \alpha \cdot \beta^c \cdot 2^{c \cdot (\mathrm{tw}(I_k))^{1/d}}$.

We conclude that $2^{(\mathrm{tw}(I_k))^{1/d'}} \leqslant \alpha \cdot \beta^c \cdot 2^{c \cdot (\mathrm{tw}(I_k))^{1/d}}$. Recall that we have $\mathrm{tw}(I_k) \geqslant k$ and that this holds for arbitrarily large $k \geqslant k_0$. As $d > d'$, we reach a contradiction by picking a sufficiently large $k \in \mathbb{N}$. This proves the lower bound of Theorem 6.4.1. □

Hence, in the rest of this section, we prove Theorem 6.4.3. The first step is to show in Section 6.4.1 a lower bound on OBDD width based on a certain structure on the prime implicants of a monotone Boolean function, called *dncpi-sets*. This allows us to rephrase the proof of Theorem 6.4.3 as the proof of the existence of such a pattern in the lineage of q_{p} on arbitrary instance families. In Section 6.4.2, we define a structure of *matches* of q_{p}, called a *dncm-set*, which we show implies the existence of a dncpi-set, so we can rephrase the proof of Theorem 6.4.3 again to the proof of the existence of a dncm-set. In Section 6.4.3, we finally explain how such a dncm-set can be obtained, using graph extraction techniques with a family of high-treewidth degree-3 graphs, the *skewed grids*, whose structure allows us to guarantee the existence of large dncm-sets.

6.4.1 Lower Bounds on OBDD Width from DNCPI-Sets

This section shows a lower bound on OBDD width for general monotone Boolean functions from the structure of the prime implicants of the function. We first give the necessary definitions and notations:

Definition 6.4.4. Given a total order Π on a set X of variables, for any $0 \leqslant p \leqslant |X|$, we write $\Pi_{\leqslant p}$ for the prefix of the first p variables of Π, and $\Pi_{>p}$ for the suffix that enumerates the other variables, so that Π is the concatenation of $\Pi_{\leqslant p}$ and of $\Pi_{>p}$.

Given a *monotone* Boolean function φ, a *prime implicant* P of φ is a minimal subset of the variable set X of φ that makes φ evaluate to 1: in other words, defining ν_P for all $x \in X$ by $\nu_P(x) := 1$ iff $x \in P$, we have $\nu_P(\varphi) = 1$, but for any valuation $\nu < \nu_P$ (with the pointwise order on valuations), we have $\nu(\varphi) = 0$. ◁

The following is immediate:

Lemma 6.4.5. *For any monotone Boolean function φ on variables X, for any valuation ν of X such that $\nu(\varphi) = 1$, there is a prime implicant P of φ such that $P \subseteq \{x \in X \mid \nu(x) = 1\}$.*

Our lower bound will be derived by considering sets of disjoint prime implicants with an additional non-covering condition, which we call *dncpi-sets*. Here is the formal definition:

Definition 6.4.6. Given a monotone Boolean function φ on variables X, a *disjoint non-covering prime implicant set* (dncpi-set) of φ is a set S of prime implicants which:

- are pairwise disjoint: for any $P_1 \neq P_2$ in S, we have $P_1 \cap P_2 = \emptyset$.
- are *non-covering* in the following sense: for any prime implicant P of φ, if $P \subseteq \bigsqcup S$, then $P \in S$.

The *size* of S is the number of prime implicants that it contains.

Given a total order Π of X, we say that Π *shatters* a dncpi-set S if there exists $1 \leqslant p \leqslant |X|$ such that, for all $P \in S$, we have $P \cap \Pi_{\leqslant p} \neq \emptyset$ and $P \cap \Pi_{>p} \neq \emptyset$. ◁

We now show the following bound: informally, the width of an OBDD is exponential in the size n that guarantees that every total order of variables shatters some dncpi-set of that size.

Lemma 6.4.7. *Let φ be a monotone Boolean function on variables X and $n \in \mathbb{N}$. Assume that, for every total order Π on X, there is some dncpi-set S of φ with $|S| \geqslant n$, such that Π shatters S. Then any OBDD for φ has width $\geqslant 2^n$.*

Proof. Fix φ and $n \in \mathbb{N}$. Consider the total order Π followed by an ODD of minimal width for φ. Fix the dncpi-set S of φ of size $\geqslant n$ such that Π shatters S; we write $S = \{P_1, \ldots, P_n\}$. Let p be the prefix length that witnesses that Π shatters S.

As Π shatters S, write each P_i as $X_i \sqcup Y_i$, where $X_i := P_i \cap \Pi_{\leqslant p}$ and $Y_i := P_i \cap \Pi_{>p}$. As each X_i is non-empty, choose an arbitrary $x_i \in X_i$ for all $1 \leqslant i \leqslant n$. Let $X_0 = \{x_1, \ldots, x_n\}$.

To each valuation ν of X_0, we consider a valuation ν' of the variables of $\Pi_{\leqslant p}$ defined as follows:

- each x_i is mapped to $\nu(x_i)$,
- the other variables of $\bigsqcup_i X_i$ are mapped to 1,
- other variables of $\Pi_{\leqslant p}$ are mapped to 0.

For each of the 2^n valuations ν of X_0, we consider the node $v_{\nu'}$ at level p of the OBDD that we reach by following a path from the root according to ν'. We will now show that the mapping $\nu \mapsto v_{\nu'}$ is injective, which implies that the number of nodes at level p of the OBDD is $\geqslant 2^n$, proving the claim.

To show this, consider two valuations $\nu_1 \neq \nu_2$ of X_0, and the corresponding ν'_1 and ν'_2, and let us prove that $v_{\nu'_1} \neq v_{\nu'_2}$. We build a valuation ν'' of the variables of $\Pi_{>p}$ such that, letting ν''_1 be the valuation of the entire X obtained by combining ν'' and ν'_1, and defining analogously ν''_2 from ν'' and ν'_2, we have $\nu''_1(\varphi) \neq \nu''_2(\varphi)$. If we show this, then it indeed implies that $v_{\nu'_1} \neq v_{\nu'_2}$, as otherwise, if they were equal, then following the path labeled by ν'' from them, we would reach a leaf labeled either 0 or 1, and for one of the ν''_i this would be an inaccurate answer.

Let us thus define ν''. As $\nu'_1 \neq \nu'_2$, there is $x \in X$ such that $\nu'_1(x) \neq \nu'_2(x)$. Without loss of generality, say that $\nu_1(x) = 0$ and $\nu_2(x) = 1$. As ν'_1 and ν'_2 match except on X_0, it must be the case that $x \in X_0$, so we have $x = x_j$ for some j. Consider the valuation ν'' of $\Pi_{>p}$ defined as follows:

- $\nu''(y) = 1$ for $y \in Y_j$,
- $\nu''(y) = 0$ otherwise.

We show that ν''_1 and ν''_2, defined as before, are such that $\nu''_1(\varphi) = 0$ but $\nu''_2(\varphi) = 1$, which proves the claim.

First observe that we have $\nu''_2(\varphi) = 1$. Indeed, ν''_2 sets all elements of S_j to 1: x_j is set to 1 by ν_2 as we observed, $X_j \backslash \{x_j\}$ is set to 1 by definition of ν'_2, and Y_j is set to 1 by definition of ν''.

Now observe that we have $\nu''_1(\varphi) = 0$. Indeed, first observe that for any $x \in X$, if $\nu''_1(x) = 1$, then by definition we have $x \in \bigsqcup S$. Hence, assuming to the contrary that $\nu''_1(\varphi) = 1$, by Lemma 6.4.5 there must be a prime implicant P of φ such that ν''_1 sets all variables of P to 1, and as we observed we must have $P \subseteq \bigsqcup S$. Now, as S is non-covering, this means that P is one of the S_i, say S_k. But as Y_k is non-empty, pick $y_k \in Y_k$. We must have $y_k \in \Pi_{>p}$ by definition of Y_k, and we know that ν''_1 sets y_k to 1, but observe that the only variables set to true by ν'' are those of Y_j. Hence, as S_j and S_k are disjoint if $j \neq k$, we must have $j = k$. But this means we have a contradiction, because S_j contains x_j and we know that $\nu_1(x_j) = 0$, contradicting our claim that ν''_1 sets all variables of S_j to 1. Hence indeed we have $\nu''_1(\varphi) = 0$. This concludes the proof. □

6.4.2 Rephrasing to Disjoint Non-Covering Match Sets

We will prove Theorem 6.4.3 using Lemma 6.4.7, and we explain in this section how this reduces to a search for matches of a certain structure.

Let I be any instance and q be a query. Let us define the following notion:

Definition 6.4.8. A *minimal match* of q in I is a subinstance $I' \subseteq I$ such that $I' \models q$ but for any $I'' \subsetneq I'$, we have $I'' \not\models q$. ◁

If φ is a monotone Boolean function, the valuation *induced by* a prime implicant P of φ is the valuation of the variables of φ setting variables in P to 1 and other variables to 0. If I is a relational instance, the valuation corresponding to a subinstance I' of I is the valuation setting all facts of I' to 1 and all other facts of I to 0. As any query q in $\mathrm{UCQ}^{\neq}$ is monotone, the lineage $\mathrm{Prov}(q, I)$ of q on I (recall Definition 3.3.1) is a monotone Boolean function (from Lemma 3.4.1), and the following is easily observed:

Lemma 6.4.9. *Let q be a query in $\mathrm{UCQ}^{\neq}$, I an instance, and $\varphi := \mathrm{Prov}(q, I)$ the lineage of q on I. The valuations induced by the prime implicants of φ are exactly those corresponding to the minimal matches of q in I.*

As we will use Lemma 6.4.7, we want to find dncpi-sets in φ. Let us see what this means in terms of the instance I and query q.

Definition 6.4.10. A *disjoint non-covering match set* (or *dncm-set*) for q in I is a set S of minimal matches of q such that the domains of the matches are pairwise disjoint (no two facts from different matches share any common element). Its *size* is its number of matches. ◁

We now consider a dncm-set S of a *connected* $\mathrm{UCQ}^{\neq}$ query q in an instance I. It is easy to observe that to S corresponds a dncpi-set S' of φ of the same cardinality. Indeed, the variables corresponding to each match are a prime implicant of φ by Lemma 6.4.9, so we must show that S' satisfies the conditions of dncpi-sets. The disjointness condition on S' is respected because the facts in each match must be different as their domains are disjoint. To see why the non-covering condition is respected, consider a prime implicant P covered by S' and assume it is not in S'. In terms of the instance, this means that there is a minimal match M_0 of q whose facts are all in $\bigsqcup S$ and that contains facts from two different minimal matches in S, say a fact F in a match M and F' in a match M' with $M \neq M'$. Now, as q is connected, its minimal matches are *connected*, which implies that there must be a path of facts $F = F_1, \ldots, F_n = F'$ in M_0, with F_i and F_{i+1} sharing some element of $\mathrm{dom}(I)$ for all i. Remembering that $M_0 \subseteq \bigsqcup S$, this implies that there must be some F_i in a match of S and F_{i+1} in a different match of S, but this is impossible because we imposed disjointness of the domains.

This correspondence between dncm-sets and dncpi-sets encourages us to define the following:

Definition 6.4.11. Letting Π be a total order on the facts of I and $\Pi_{\leqslant p}$ a prefix, letting S be a dncm-set, we say Π *shatters* S if there exists $1 \leqslant p \leqslant |I|$ such that each match of S has a non-empty intersection both with $\Pi_{\leqslant p}$ and with $\Pi_{>p}$.

It is immediate that this implies that the dncpi-set corresponding to S is shattered by Π. ◁

Hence, using Lemma 6.4.7, the above implies that we can rephrase our goal, Theorem 6.4.3, as the following result about dncm-sets for our connected $\mathrm{UCQ}^{\neq}$ q_{p}:

Lemma 6.4.12. *For any arity-two signature σ, there exist constants $d', k_0 \in \mathbb{N}$ such that, for any σ-instance I with $\mathrm{tw}(I) \geqslant k_0$, for every total order Π on the facts of I, there exists a dncm-set for q_{p} in I of size $\geqslant (\mathrm{tw}(I))^{1/d'}$ that is shattered by Π.*

So it suffices to prove Lemma 6.4.12 to conclude the proof of Theorem 6.4.3, and hence of Theorem 6.4.1. We do this in the next section.

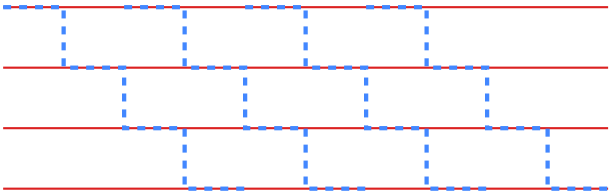


Figure 6.1: The 4×4 skewed grid, with horizontal (red) and vertical paths (thick dashed blue) in the proof of Lemma 6.4.14

6.4.3 Finding DNCM-Sets in Skewed Grids

This section proves Lemma 6.4.12. Let us introduce the relevant concepts. Inspired by the standard class of *wall graphs* [Dragan, Fomin, and Golovach 2011], we will be interested in graphs that we call *skewed grids* (see Figure 6.1 for an illustration, disregarding colors and line style for now):

Definition 6.4.13. The $n \times n$ *skewed grid* G_n is the graph on $\{(r, c) \mid 1 \leqslant r \leqslant n, 1 \leqslant c \leqslant 3n - 1\}$ defined by the edges:

- for all $1 \leqslant r \leqslant n$ and $1 \leqslant c < 3n - 1$, vertices (r, c) and $(r, c + 1)$ are adjacent;
- for all $1 \leqslant r < n$ and $1 \leqslant j \leqslant n$, letting $c = 2j + r - 1$, vertices (r, c) and $(r + 1, c)$ are adjacent. ◁

The point of skewed grids is the following lemma, which we will use to create dncm-sets from a skewed grid extracted as a topological minor:

Lemma 6.4.14. *Let G_n be the $n \times n$ skewed grid (with n a positive even integer), and let G be an arbitrary subdivision of G_n. Let Π be a total order on the edges of G. There exists $1 \leqslant p \leqslant |\Pi|$ such that there is a set V of $n/2$ vertices with each $v \in V$ having one incident edge in $\Pi_{\leqslant p}$ and one incident edge in $\Pi_{>p}$.*

Proof. Fix G and Π. For $1 \leqslant p \leqslant |\Pi|$ and π a simple path in G, we will call π:

- *complete* at p, if $\pi \subseteq \Pi_{\leqslant p}$;
- *empty* at p, if $\pi \subseteq \Pi_{>p}$;
- *shattered* at p, if π intersects both $\Pi_{\leqslant p}$ and $\Pi_{>p}$.

We call a vertex of G *shattered at* p if it has an incident edge in $\Pi_{\leqslant p}$ and an incident edge in $\Pi_{>p}$. Hence, our goal is to show the existence of $1 \leqslant p \leqslant |\Pi|$ such that there are $\geqslant n/2$ vertices that are shattered at p.

We make a simple preliminary observation: if a simple path π is shattered at p, then some vertex of π is shattered at p. Indeed, π being shattered means it has edges in either set, so there must be some vertex adjacent to one edge of each set.

Consider the n *horizontal paths* $\pi_1, \ldots, \pi_n$ defined as the simple paths of G corresponding to the paths, for $1 \leqslant r \leqslant n$, defined by $(r, 1), (r, 2) \ldots, (r, 3n - 2)$, $(r, 3n - 1)$, in G_n (see Figure 6.1 for an illustration). Define p to be the smallest value such that there are at least $n/2$ of the π_i which are complete at p. Indeed, the existence of such a p follows from the fact that none of the π_i are complete at $p = 0$, all are complete at $p = |\Pi|$, and the number which are complete is a non-decreasing

function of p. Further, this function varies continuously, because each additional edge completes at most one of the π_i (as they are edge-disjoint); so for our choice of p, there are *exactly* $n/2$ of the π_i which are complete.

Consider the *vertical paths* $\pi'_1, \ldots, \pi'_n$ which are the simple paths of G corresponding, for $1 \leqslant j \leqslant n$, letting $j' := 2j - 1$, to the paths defined by $(1, j'), (1, j' + 1)$, $(2, j' + 1), (2, j' + 2), (3, j' + 2), \ldots, (n, j' + (n - 1)), (n, j' + n)$ in G_n. Either no π'_i is complete at p, or some π'_i is complete at p. In the first case, observe that each of the π'_i shares some edge with all of the π_i, hence, as there is at least one π_i which is complete at p (indeed there are $n/2 > 0$ of them), none of the π'_i can be empty at p. Hence, each of the π'_i is neither complete nor empty at p, so it must be shattered; and we deduce from the preliminary observation that each of these n shattered π'_i has a vertex that is shattered at p, so as the π'_i are simple, we can conclude the proof.

In the second case, let i_0 be such that π'_{i_0} is complete at p. Remember that exactly $n/2$ of the π_i are complete at p, and let us investigate the status of the $n/2$ other π_i. They cannot be complete at p, by our choice of p. They cannot be empty, because each of them shares an edge with π'_{i_0}. Hence, those $n/2$ remaining π_i are shattered at p. By the preliminary observation again, we conclude that there are $n/2$ vertices that are shattered at p. This concludes the proof. □

We can finally prove Lemma 6.4.12. We will fix d' and k_0 later. Observe first that skewed grids are degree-3 planar graphs, and $|V(G_n)| = n(3n - 1)$. Fix the instance I, let k be its treewidth, and let G be its Gaifman graph, whose treewidth is also k. Letting c' be the exponent of Lemma 6.1.4, letting $n := 2\lfloor \frac{1}{6} \times k^{1/(2c')} \rfloor$, letting $H := G_n$, we have $|V(H)|^{c'} \leqslant (3n)^{2c'} \leqslant k$. Hence, H is a topological minor of G. We fix k_0 to be large enough so that $n \geqslant 2$.

Let Π be any order on the facts of I. We define from Π a total order Π' on the edges of the skewed grid minor H' of H embedded in G in the following way: we go through Π in order and add an edge to Π' precisely when *all* witnessing facts for this edge have been enumerated in Π (remember that G is the Gaifman graph of I and that, therefore, several facts of I may correspond to the same edge in G). By Lemma 6.4.14 applied to Π' and H' (which is a subdivision of $H = G_n$), as n is even, we deduce the existence of $1 \leqslant p \leqslant |\Pi|$ such that there is a set A of $n/2$ elements of the domain of I, with each $a \in A$ occurring in a fact $F_a^{\leqslant}$ of $\Pi_{\leqslant p}$ and occurring in a fact $F_a^{>}$ of $\Pi_{>p}$, the other element occurring in $F_a^{>}$ being different from the one occurring in $F_a^{\leqslant}$ (because the witnessing edges in H' were not the same), and both facts $F_a^{\leqslant}$ and $F_a^{>}$ being reflected as edges in H'.

We will construct from these facts our desired dncm-set, which will conclude the proof of Lemma 6.4.12. We construct a *candidate* dncm-set S' by taking, for each $a \in A$, the match M_a of q_{p} formed of $\{F_a^{\leqslant}, F_a^{>}\}$. M_a is indeed a match of q_{p}, as it consists of two facts that connect one common element to two distinct elements. Observe that by construction of A, each match has a different domain (of size 3) since it corresponds to a different pair of edges in the Gaifman graph of I; however, we have to ensure that the domains of the matches are *disjoint*, which is not true on S' in general. But since the domain of each match is by construction a subset of vertices of the minor H', whose degree is bounded by 3, there is a constant α such that each vertex in some domain in S' occurs in at most α matches (α can be bounded by $3 + 3 \times 2$ where 3 is the maximum number of matches with the vertex in the middle, and 3×2 the maximum number of matches with the vertex

at an extreme point). In particular, each vertex occurs in at most α matches of S'. Hence, let us do the following: we pick one match M_1 in S' and simply eliminate all other matches where an element of $\mathrm{dom}(M_1)$ co-occurs, i.e., we eliminate at most $|\mathrm{dom}(M_1)| \cdot \alpha - 1$ matches, so at most $3\alpha - 1$ matches.

Hence, we can repeat the process and obtain a dncm-set S of size at least $\frac{1}{3\alpha} \cdot \frac{n}{2}$. Recalling that n is defined to be $\Theta(k^{1/(2c')})$, this means that, for sufficiently large k, the dncm-set S has size $\geqslant k^{1/(2c'+1)}$. Hence, we can fix d' to be $2c'+1$ and pick a sufficiently large k_0 (which can be independent from $\mathcal{I}$ or Π, in particular we always have $\alpha \leqslant 9$), and we have proven Lemma 6.4.12.

This concludes the proof of Theorem 6.4.3, and hence of Theorem 6.4.1, which concludes the section.

6.5 Meta-Dichotomy for OBDD Representations

We conclude this chapter by studying whether our dichotomy on tractable OBDD representations in the previous chapter (Theorem 6.4.1) could not apply to different UCQ$^{\neq}$ queries, or to more restricted query languages.

Our main result in this section is a complete classification of *connected* UCQ$^{\neq}$ queries relative to Theorem 6.4.1, in a *meta-dichotomy* (Theorem 6.5.1). We characterize the UCQ$^{\neq}$ queries for which we can prove the analogue of Theorem 6.4.1, such as q_{p}; they are the *intricate* queries. For the others, we show that there is a family of treewidth-constructible unbounded-treewidth instances where they have tractable OBDDs; in fact, *constant-width* OBDDs. Hence, if a connected UCQ$^{\neq}$ has polynomial OBDDs on some unbounded-treewidth instance family, then there must be such a family on which it has constant-width OBDDs.

We also classify other query classes. In Section 6.5.3, we classify connected CQ$^{\neq}$ queries, showing that neither of them are intricate, so that Theorem 6.4.1 cannot be proven with a CQ$^{\neq}$ query. In Section 6.5.4, we classify UCQ queries, and in fact all queries closed under homomorphisms, showing that Theorem 6.4.1 cannot apply to them either. Last, we study *disconnected* CQ$^{\neq}$ queries. For them, we show in Section 6.5.5 that our meta-dichotomy theorem does not hold: there is a disconnected CQ$^{\neq}$ query q_{d} that has no constant-width OBDDs on any unbounded-treewidth treewidth-constructible family, but there is one such family where the OBDDs have width *linear* in the treewidth, not exponential, so that neither of the cases of the meta-dichotomy applies to q_{d}.

6.5.1 Meta-Dichotomy Theorem

Our meta-dichotomy applies to *connected* UCQ$^{\neq}$ (recall Definition 6.4.2). We first give the meta-dichotomy result, without explaining yet what is our definition of an *intricate* query (we will define this just afterwards):

Theorem 6.5.1. *For any connected UCQ$^{\neq}$ q on an arity-2 signature:*

- *If q is not intricate, there is a treewidth-constructible and unbounded-treewidth family $\mathcal{I}$ of instances such that q has* constant-width *OBDDs on $\mathcal{I}$; the OBDDs can be computed in PTIME from the input instance.*

- *If q is intricate, then Theorem 6.4.1 applies to q: in particular, for any unbounded-treewidth family $\mathcal{I}$ of instances satisfying the hypotheses, q does not have polynomial-width OBDDs on $\mathcal{I}$.*

In other words, the analogue of the dichotomy of Theorem 6.4.1 holds for a connected UCQ$^{\neq}$ q if and only if it is intricate. Further, *non-intricate* queries, which do not have a lower bound on OBDD width as a function of treewidth, must actually have *constant-width* OBDD on some counterexample unbounded-treewidth family.

We now give our definition of *intricate* queries. We characterize them by looking at *line instances*:

Definition 6.5.2. A *line instance* is an instance I of the following form: a domain $a_1, \ldots, a_n$, and, for $1 \leqslant i < n$, one single binary fact between a_i and a_{i+1}: either $R(a_i, a_{i+1})$ for some $R \in \sigma$ or $R(a_{i+1}, a_i)$ for some $R \in \sigma$. ◁

The intuition is that a query is intricate if, on any sufficiently long line instance, it must have a minimal match that contains the two middle facts (i.e., the ones that are incident to the middle element). Formally:

Definition 6.5.3. A UCQ$^{\neq}$ q is *n-intricate* for $n \in \mathbb{N}$ if, for every line instance I with $|I| = 2n + 2$, letting F and F' be the two facts of I incident to the middle element a_{n+2}, there is a *minimal* match of q on I that includes both F and F'.

We call q *intricate* if it is $|q|$-intricate. ◁

Observe that queries q with $|q| < 2$ clearly cannot be intricate. Further, if a query has no matches that include only binary facts, then it cannot be intricate; in other words, any disjunct that contains an atom for a unary relation can be ignored when determining whether a query is intricate. By contrast, our query q_{p} of Theorem 6.4.1 was designed to be intricate, in fact q_{p} is 0-intricate. Note that, clearly, an n-intricate query is always m-intricate for any $m > n$, by considering the restriction of any line instance of size $2m + 2$ to a line instance of size $2n + 2$, and finding a match in the restriction.

We note that we can decide whether UCQ$^{\neq}$ queries are intricate or not, by enumerating line instances. We do not know the precise complexity of this task:

Lemma 6.5.4. *Given a connected UCQ$^{\neq}$ q, we can decide in PSPACE whether q is intricate.*

Proof. Enumerate all possible line instance of size $3 \cdot |q|$, and, for each such instance I, compute all matches of q on I and check the condition. □

What remains to do is to prove Theorem 6.5.1, which we do in the next section. We then study the language of connected CQ$^{\neq}$ in Section 6.5.3, queries closed under homomorphism (including UCQ) in Section 6.5.4, and non-connected CQ$^{\neq}$ in Section 6.5.5.

6.5.2 Proof of Theorem 6.5.1

We prove separately the two claims of the result. The first claim is about non-intricate queries, and the second claim is about intricate queries.

6.5.2.1 First Claim: Non-Intricate Queries

We first define the *concatenation* of line instances:

Definition 6.5.5. The *concatenation* $I \circ I'$ of two line instances I and I' is defined in the expected way by identifying the right endpoint of I with the left endpoint of I'; it is a line instance of size $|I| + |I'|$. The *boundary element* of $I \circ I'$ is the element which is incident both to the last fact of I and the first fact of I'. $\lhd$

We will construct the tractable instance family for the query as a grid formed of line instances that are counterexamples to intricacy. For this, we need to know how to construct grids from a line instance:

Definition 6.5.6. Let I be a line instance. We define the $n \times n$ *I-grid*, written $I^{n\times n}$, as the grid whose edges are copies of I (in the top-to-bottom, left-to-right direction).

If $I = I_1 \circ I_2$, the *unfolded $n \times n$ I-grid* is obtained from $I^{n\times n}$ by replacing every copy in $I^{n\times n}$ of a boundary element I_1 and I_2 with two fresh domain elements, each of which being used in one of the two facts where the boundary element occurs. $\lhd$

Note that the operation that creates the unfolded grid is a particular kind of *unfolding*, as defined in Section 4.5.

Now, let q be a non-intricate query. By definition there is a witness line instance I decomposable as $I = I_1 \circ I_2$ with $|I_1|, |I_2| \geqslant |q|$ such that no minimal match of q on I spans both I_1 and I_2. The class of I-grids is clearly unbounded-treewidth (its Gaifman graphs are subdivisions of the grid graphs) and it is treewidth-constructible because the size of I is constant. Now, we claim that the lineage of q on $I^{n\times n}$ is exactly its lineage on the unfolded $n \times n$ I-grid, which we call I'. Indeed, if this were not the case, as $|I_1|, |I_2| \geqslant |q|$, and q is connected, a counterexample match on $I^{n\times n}$ that would be broken in I' would entirely be within some copy of I in $I^{n\times n}$; but no match within some copy of I can span both I_1 and I_2.

Conversely, as q is connected, its matches are connected, so there can be no matches on I' involving both copies of boundary element of $I^{n\times n}$ (the copies not connected in the Gaifman graph of I'), so all matches in I' must be matches in $I^{n\times n}$. Now, we conclude because the unfolded grid I' has bounded pathwidth (its connected components have constant size), so we can apply Theorem 3.5.4.

6.5.2.2 Second Claim: Intricate Queries

We now attack the second claim. Fix an intricate query q. We proceed as in the proof of the lower bound of Theorem 6.4.1, so, as q is a connected $\mathrm{UCQ}^{\neq}$, it suffices to show the analogue of Lemma 6.4.12 but with our choice of intricate query q instead of q_{p}.

We fix d' as in the proof of Lemma 6.4.12 in Section 6.4.3, and, as before, we will fix k_0 later. As before, fix the instance I, let k be its treewidth, let c' be the exponent of Lemma 6.1.4, and, letting n be defined as before, let $n' := n + 2 \cdot |q|$. Up to fixing k_0 to be sufficiently large, we can extract the skewed grid $H = G_{n'}$ as a topological minor H' of the Gaifman graph G of I. Let H'' be the $n \times n$ skewed grid topological minor of G by restricting H' following the embedding of the $n \times n$ skewed grid as a topological minor of the $n' \times n'$ skewed grid, such that the horizontal paths $1, \ldots, n$ of H'' correspond to the horizontal paths $|q| + 1, \ldots, |q| + n$ of H', and likewise for the vertical paths.

Let us now fix the order Π on the facts of I, and define an order Π' on the edges of H'' exactly as before. By Lemma 6.4.14, as n' is even, we deduce the existence of $1 \leqslant p \leqslant |\Pi'|$ such that there are $n/2$ elements of I that occur in an edge of $\Pi'_{\leqslant p}$ and occur in an edge of $\Pi'_{>p}$.

We will construct from this our desired dncm-set, concluding the proof. We construct a candidate set by considering, for each element a in the $n/2$ elements, some simple path of H' of length $2\,|q| + 2$ (not necessarily a horizontal or vertical path) that includes the two edges incident to a; this is possible, by definition of H'' from H', because a is in H''. We then build from this path a line instance $I'_a \subseteq I$ of size $2 \cdot |q| + 2$ of I where a is the middle element, and with one of the two incident facts $F_a^{\leqslant}$ occurring in $\Pi_{\leqslant p}$ and the other one $F_a^{>}$ occurring in $\Pi_{>p}$. Now, as q is intricate, there must be a minimal match $I''_a \subseteq I'_a$ of q' in I' where $F_a^{\leqslant}$ and $F_a^{>}$ occur, and we take I''_a as a candidate match for a. Thus, we obtain a candidate set S' of minimal matches I''_a for each of the $n/2$ elements a, which satisfies everything but the disjointness condition, and where all facts are reflected by an edge of H'.

Now, as before, we can ensure the disjointness condition by observing that, having picked a match in S', we can simply eliminate everything at distance $\leqslant 2\,|q|$ in H', so a constant number of vertices (as the degree of H' is, again, bounded by 3). This uses the fact that the matches in S' are connected and that their edges are reflected in H'. Hence, we conclude as before.

This concludes the proof of Theorem 6.5.1.

6.5.3 Connected CQ$^{\neq}$ Queries

Having proven Theorem 6.5.1, we give an example of what can be proven using the notion of intricate queries, by showing that a CQ$^{\neq}$ can never be intricate.

Proposition 6.5.7. *A connected CQ$^{\neq}$ is never intricate.*

By Theorem 6.5.1, this implies that any CQ$^{\neq}$ query q has an unbounded-treewidth, treewidth-constructible family of instances $\mathcal{I}$ such that q has constant-width OBDDs on $\mathcal{I}$ (that can be computed in PTIME); and it also implies that we could not have proven Theorem 6.4.1 with a connected CQ$^{\neq}$ query.

To prove Proposition 6.5.7, we first exclude the case of signatures with more than one binary relation. The argument for this is straightforward, and in fact does not rely on connectedness of the query:

Proposition 6.5.8. *On arity-2 signatures that contain more than one binary relation, for any CQ$^{\neq}$ query q, there is a treewidth-constructible instance family $\mathcal{I}$ which has unbounded treewidth such that $I \not\models q$ for all $I \in \mathcal{I}$. This implies that the lineage of q on $\mathcal{I}$ has trivial constant-width OBDD representations.*

Proof. Fix such a query q. Pick a relation R in σ. If some other relation than R occurs in q, then take for $\mathcal{I}$ the unbounded-treewidth, treewidth-constructible family of grids with R-edges, and q is never satisfied on them. Hence, we can assume that q only contains R-atoms. But as σ contains more than one binary relation, pick $S \neq R$, and take for $\mathcal{I}$ the class of grids with S-edges. We conclude in the same way. □

Hence, we can assume that the signature σ contains exactly one binary relation. We can now show Proposition 6.5.7:

Proof of Proposition 6.5.7. We can assume that σ contains a single binary relation R, and it suffices to restrict our attention to $\mathrm{CQ}^{\neq}$ queries q with $|q| \geqslant 2$, as otherwise q cannot be intricate.

If q contains two atoms $A = R(x, y)$ and $A' = R(y, z)$ where x and z are two different variables, then consider the line instance I defined by $R(a_1, a_2)$, $R(a_3, a_2)$, $R(a_3, a_4)$, $R(a_5, a_4)$, $R(a_5, a_6)$, $R(a_7, a_6)$, ..., until we reach $a_{2\cdot|q|+2}$. It is clear that q has no match on I at all, by considering just A and A', so in particular there is no minimal match involving the two facts incident to the middle fact $a_{|q|+2}$. Thus, I witnesses that q is not intricate.

Now, if q does not contain two such atoms, consider the line instance $I' = \{R(a_i, a_{i+1}) \mid 1 \leqslant i < 2 \cdot |q| + 2\}$. Assume by contradiction that it is not a counterexample to intricacy, so there is a minimal match on q on I that includes the two facts $F = R(a_{|q|+1}, a_{|q|+2})$ and $F' = R(a_{|q|+2}, a_{|q|+3})$. Thus, let x be a variable of q mapped to $a_{|q|+1}$ in this match, and y be a variable of q mapped to $a_{|q|+3}$.

As q is connected, let $x = z_1, \ldots, z_l = y$ be a simple path of pairwise distinct variables from x to y such that z_i and z_{i+1} co-occur in some atom of q for all i. Now, consider the sequence $a_{i_1}, \ldots, a_{i_l}$ of the images of the z_i in I'. Clearly we must have $i_j \neq i_{j+1}$ for all $1 \leqslant j < l$, as otherwise q contains an atom A with two distinct variables matched to the same element in I', but the result cannot be a match of A in I', hence of q, by definition of I'. Hence, the sequence $i_1, \ldots, i_l$ is a sequence of integers with $i_1 = |q| + 1$, $i_l = |q| + 3$, any two consecutive values are different, and any two consecutive values i_j, i_{j+1} must have a difference with absolute value 1, as is seen by considering the match in I' of the witness atom in q for z_j and z_{j+1}. It is then obvious that there must be $1 < j_0 < l$ such that $i_{j_0-1} = |q| + 1$, $i_{j_0} = |q| + 2$, and $i_{j_0+1} = |q| + 3$.

Thus, consider the witness atoms $A(z_{j_0-1}, z_{j_0})$ and $A'(z_{j_0}, z_{j_0+1})$. Their match in I' must be F and F'. Hence, we deduce that A and A' are of the form that we initially excluded (remember that the path $z_1, \ldots, z_l$ is a simple path of pairwise distinct variables, so z_{j_0-1} and z_{j_0+1} must be different). We have reached a contradiction. Thus, we conclude that I' is a counterexample to the intricacy of q. Hence, q is not intricate, which concludes the proof. □

6.5.4 Homomorphism-Closed Queries

Second, we investigate the status of UCQ in our meta-dichotomy. We can in fact show a result for all queries that are *closed under homomorphisms*, no matter whether they are connected or not. Further, we can even choose a single class of instances which is easy for *all* query closed under homomorphisms. (Remember that our queries are always constant-free.)

Proposition 6.5.9. *For any arity-2 signature, there is a treewidth-constructible instance family $\mathcal{I}$ with unbounded treewidth and $w \in \mathbb{N}$ such that any query q closed under homomorphisms has OBDDs of width w on $\mathcal{I}$ that can be computed in PTIME in the input instance.*

This implies by Theorem 6.5.1 the following:

Corollary 6.5.10. *A connected UCQ is never intricate.*

It also implies that we could not have shown Theorem 6.4.1 with a UCQ query rather than a UCQ$^{\neq}$ query.

Again, this result should not be confused with those of [Jha and Suciu 2012; Jha and Suciu 2013]. Of course, not all homomorphism-closed queries, or even UCQs, have constant-width OBDDs on arbitrary instances. We are merely claiming the *existence* of high-treewidth instance classes for which we have constant-width OBDDs whatever the query.

Proof of Proposition 6.5.9. Pick any binary relation $R \in \sigma$. For any $n \in \mathbb{N}$, let I_n be the bipartite directed R-graph on $\{1,2\} \times \{1,\ldots,n\}$ formed of all facts of the form $\{ R((1,i),(2,j)) \mid 1 \leqslant i,j \leqslant n\}$. Let $\mathcal{I} = (I_n)_{n\in\mathbb{N}}$. The family $\mathcal{I}$ is unbounded-treewidth and treewidth-constructible; we show that any homomorphism-closed q has a constant-width OBDD on any instance of $\mathcal{I}$ that can be computed in PTIME.

Fix $I_n \in \mathcal{I}$. Either q has no match on I_n, or it has some match on I_n, which we can decide in PTIME in I_n. In the first case, there is a trivial constant-width OBDD of q on I_n that can be computed in PTIME. In the second case, consider a match $I' \subseteq I_n$ of q, and let I'' be its image under the homomorphism from I_n to I_1 that maps $(1,i)$ to $(1,1)$ and $(2,j)$ to $(2,1)$ for all $1 \leqslant i,j \leqslant n$. As q is closed under homomorphisms, q must hold on I'', and as I' is non-empty, we have $I'' = \{R((1,1),(2,1))\}$, so we conclude that q holds on any subinstance of I_n iff the subinstance is non-empty. Thus, we can construct in PTIME a constant-width OBDD for q on I_n just by enumerating the facts of I_n in an arbitrary order, and testing whether one of them is present.

Hence in either case q has constant-width OBDDs on $\mathcal{I}$ that can be computed in PTIME, concluding the proof. □

6.5.5 Beyond Connected Queries

We consider last whether our dichotomy in Theorem 6.4.1 could extend to *disconnected* CQ$^{\neq}$, which are not covered by Proposition 6.5.7 or by the meta-dichotomy of Theorem 6.5.1.

If the signature has more than one binary relation, this is hopeless: by the easy argument of Proposition 6.5.8, any CQ$^{\neq}$ then has constant-width OBDDs on the family of R-grids for some binary $R \in \sigma$.

However, quite surprisingly, on signatures with a *single* binary relation (and arbitrarily many unary ones) we can show a weakening of Theorem 6.4.1 for a disconnected CQ$^{\neq}$. The first part adapts (it holds for all MSO), so only the lower bound is interesting, which we can rephrase as before to a lower bound on OBDD width on individual input instances. We thus show the following variant of Theorem 6.4.3:

Proposition 6.5.11. *Let σ be an arity-2 signature with only one binary relation. There exists a disconnected CQ$^{\neq}$ query q_{d}, a constant $d' > 1$ and integer $n_0 \in \mathbb{N}$ such that: for any instance I on σ of size $\geqslant n_0$, letting k be the treewidth of I, the width of any OBDD for q_{d} is $\Omega(k^{1/d'})$.*

Observe that the bound on the OBDD width is no longer exponential in the instance treewidth; however, it is not constant, so this suffices to rule out that q_{d} could have constant-width OBDDs on any unbounded-treewidth, treewidth-constructible instance family.

To prove the result, let R be the only binary relation of the signature, and consider the following query q_{d}, which intuitively tests whether there are two R-facts that do not share an element:

$$q_{\mathrm{d}} := \exists xyzw\, (x \neq y) \wedge (x \neq z) \wedge (x \neq w) \wedge (y \neq z) \wedge (y \neq w) \wedge (z \neq w) \wedge R(x,y) \wedge R(z,w)$$

We show our claim using q_{d}:

Proof of Proposition 6.5.11. We proceed as in the proof of Lemma 6.4.12: we take an arbitrary order Π on facts, we extract pairs of adjacent facts from a skewed grid minor of the instance I where one fact is enumerated by an order prefix $\Pi_{\leqslant p}$ and one is non-enumerated, for some prefix length p. As in that proof, we can obtain $\Omega(k^{1/d'})$ such pairs of facts, with d' defined identically. We can also impose that each pair of facts has domain disjoint from every other pair of facts, in the same way as in the original proof. We call S be the resulting set of pairs of facts; by our hypothesis on σ, all these facts must be R-facts. To summarize, for n and α defined as in the proof of Lemma 6.4.12, S is a set of $\frac{1}{3\alpha} \cdot \frac{n}{2}$ pairs of R-facts, the pairs have pairwise disjoint domains, but any pair contains an element shared by the two facts of the pair; further, one fact of each pair is in $\Pi_{\leqslant p}$ and the other is in $\Pi_{>p}$. Hence, S is *not* a dncm-set for q_{d}, because it precisely consist of fact pairs that are not matches of q_{d}: intuitively, we will instead consider the matches of q_{d} on I formed of pairs of facts of S that are *not* in the same pair.

We use S to prove a variant of Lemma 6.4.7. For each pair $\{F_{\leqslant}, F_{>}\} \in S$, with $F_{\leqslant} \in \Pi_{\leqslant p}$ and $F_{>} \in \Pi_{>p}$, we consider the valuation $\nu_{F_{\leqslant}}$ that sets $F_{\leqslant}$ to 1, and sets all other facts of $\Pi_{\leqslant P}$ to 0, Every $\nu_{F_{\leqslant}}$ must yield a different node in an OBDD for q_{d}: indeed, the partial evaluation of $\mathrm{Prov}(q_{\mathrm{d}}, I)$ with $\nu_{F_{\leqslant}}$ does not evaluate to 1 when we set the corresponding $F_{>}$ to 1 and all other facts to 0, whereas the same evaluation for any other partial valuation $\nu_{F'_{\leqslant}}$ would yield 1. From our definition of n and d', we conclude that an OBDD for q on I must have width $\Omega(k^{1/d'})$ at level p. □

This implies that q_{d} does not satisfy the first part of the meta-dichotomy of Theorem 6.5.1. Surprisingly, however, we can show that the query q_{d} has OBDDs of width $O(k)$ on some unbounded-treewidth and treewidth-constructible instance class. Hence, q_{d} does not satisfy the second part of the meta-dichotomy either, so q_{d} witnesses that there are *disconnected* $\mathrm{CQ}^{\neq}$ which do not follow our meta-dichotomy at all!

Lemma 6.5.12. *There is a treewidth-constructible, unbounded treewidth class $\mathcal{I}$ of instances on σ such that, for any $I \in \mathcal{I}$, letting k be its treewidth, q_{d} has OBDDs of width $O(k)$ on I.*

Proof. Consider, for $n \in \mathbb{N}$ the $n \times n$ R-grid I_n: the vertices are $\{\, (i,j) \mid 1 \leqslant i,j \leqslant n \,\}$, the *horizontal edges* are $R((i,j),(i,j+1))$ for $1 \leqslant i \leqslant n$, $1 \leqslant j < n$, and the *vertical edges* are $R((i,j),(i+1,j))$ for $1 \leqslant i < n$, $1 \leqslant j \leqslant n$. This defines a family of instances $\mathcal{I} = (I_n)_{n \in \mathbb{N}}$ that has unbounded treewidth (the treewidth of I_n is n), and is treewidth-constructible.

For any $I_n \in \mathcal{I}$, we choose the following enumeration order Π of the facts of I_n:

- First, enumerate all horizontal edges $R((1,j),(1,j+1))$ in ascending order of j; we call this *horizontal row* 1;

- Then, enumerate all vertical edges $R((1,j),(2,j))$ in ascending order of j; we call this *vertical row* 1;
- Then, enumerate horizontal row 2, i.e., all horizontal edges $R((2,j),(2,j+1))$ in ascending order of j;
- Then, enumerate vertical row 2, i.e., all vertical edges $R((2,j),(3,j))$ in ascending order of j;
- Continue this process, enumerating horizontal row i followed by vertical row i for $3 \leqslant i < n$, finishing by horizontal row n.

An OBDD that follows Π has to determine, while enumerating horizontal row 1, whether there are three edges or two non-adjacent edges in the row, in which case the query must be true. Otherwise, once we have processed horizontal row 1, we must further remember:

- If there was a single edge in the previous horizontal row, which one it was;
- If there were two adjacent edges in the previous horizontal row, which ones they were.

This can be performed in width linear in n.

Now, we process vertical row 1. If there are two edges in vertical row 1, the query is true. Otherwise, if there is a single one, the query is true unless that edge is incident to the common vertex of the two adjacent edges of horizontal row i, or to one of the two vertices of the only edge at horizontal row i, or there was no edge in horizontal row i. Once we have processed the vertical row, we remember:

- Whether there was some edge in some *strictly preceding* horizontal or vertical row (here, in horizontal row 1);
- If there was a single edge in the previous vertical row, and the query is not yet true, which edge it is.

We then process horizontal row 2. If we have remembered that there was some edge in a strictly preceding horizontal or vertical row, then our job is much easier, as we can just succeed as soon as we see a single edge. If this was not the case, then if there was no edge in the previous vertical row, we proceed as for horizontal row 1. Otherwise, if we see an edge e in horizontal row 2 which is not adjacent to the memorized vertical edge e' in the previous vertical row, we succeed. Otherwise, if e is adjacent to e', we ignore e (it is easier to build a match with e'). At the end of the horizontal row, we simply need to memorize if there was an edge in a strictly preceding horizontal or vertical row, and, if not, the same information that we needed to remember for horizontal row 1. All of this requires $O(n)$ width.

We construct our OBDD by repeating this process, with the width remaining linear as we process the following horizontal and vertical rows. □

Conclusion. *This finishes our investigation of the limits of our meta-dichotomy result, which concludes our study of the intractability of OBDD computation, and of probability evaluation, on unbounded-treewidth instance families. Thus, we have presented our results on instance-based tractability conditions for probabilistic query evaluation and provenance computation.*

The general conclusions and perspectives are given at the end of this manuscript, after Part II; see in particular Section 1 of the Conclusion.

Part II

Finite Open-World Query Answering

Chapter 7

Introduction

In this second part of my thesis, we move on to the study of incomplete *instances, and query answering under logical rules that constrain the structure of instance completions. This part presents the full version of my work on finite query answering with Michael Benedikt: it was published as an extended abstract at LICS'15 [Amarilli and Benedikt 2015b], and the long version is currently under review [Amarilli and Benedikt 2016].*

The fundamental problem in the setting of incomplete instances with logical constraints is the *open-world query answering* problem, which asks: given a query q, a conjunction of constraints Σ, and a finite instance I_0, determine which answers to q are certain to hold over any instance I that extends I_0 and satisfies Σ. This problem can be rephrased to query containment under constraints, querying in the presence of ontologies, or reformulating queries under constraints. It has thus been the subject of intense study within several communities for decades; see, e.g., [Johnson and Klug 1984; Calì, Lembo, and Rosati 2003a; Pratt-Hartmann 2009; Bárány, Gottlob, and Otto 2010; Ibáñez-García, Lutz, and Schneider 2014].

In many settings, for instance in database theory, the instances I of interest are the finite ones. We can thus define *finite open-world query answering* (denoted here as FQA), which restricts the quantification to *finite* extensions I of I_0. In contrast, by *unrestricted open-world query answering* (UQA) we refer to the problem where I can be either finite or infinite. Generally the class of queries is taken to be the conjunctive queries (CQs), which are built up from relational atoms via existential quantification and conjunction. This part of my thesis always uses CQs, and thus we omit explicit mention of the query language.

The decidability of FQA and UQA clearly depends on the language of logical constraints that we allow; a natural challenge is thus to design languages for which FQA and UQA are decidable, but which are as expressive as possible, so that we can model the constraints that we want to impose on instances. The results presented here focus on the FQA problem, where decidability results have been traditionally quite hard to obtain. Hence, let us start by reviewing known results about FQA.

A first constraint class known to have tractable open-world query answering problems are *inclusion dependencies* (IDs) from database theory [Abiteboul, Hull, and Vianu 1995], i.e., constraints of the form, e.g., $\forall xyz\ R(x, y, z) \rightarrow \exists vw\ S(z, v, w, y)$. The fundamental results of [Johnson and Klug 1984] and [Rosati 2011] show that both FQA and UQA are decidable for ID and that, in fact, they coincide. We call *finitely controllable* such constraints for which FQA and UQA always coincide. These

results have been generalized in [Bárány, Gottlob, and Otto 2010] to a much richer class of constraints, the guarded fragment of first-order logic.

However, those results do not cover an important kind of constraints, namely *number restrictions*, which we can use to express, e.g., uniqueness: "for all x, there is at most one y such that $R(x, y)$ holds". The number restrictions that we consider here are *functional dependencies* (FDs), of the form $\forall \boldsymbol{xy}\ (R(x_1, \ldots, x_n) \wedge R(y_1, \ldots, y_n) \wedge \bigwedge_{i \in L} x_i = y_i) \rightarrow x_r = y_r$, which have also been commonly studied in database theory [Abiteboul, Hull, and Vianu 1995]. The FQA and UQA problems for FDs alone are trivial (it suffices to check if the initial instance satisfies them), so they are decidable, and coincide. What is interesting is to allow FDs in addition to other constraints, for instance IDs. This allows us to assert the existence of missing information in completions, e.g., "Every advisee is also the advisor of someone", while restricting what we can add, e.g., "No advisor advises two different people".

Sadly, the interaction between IDs and FDs is severe enough that UQA and FQA are undecidable in general when both are allowed [Calì, Lembo, and Rosati 2003a]. Some decidable cases have nevertheless been identified for UQA. For instance, UQA is known to be decidable when the FDs and the IDs are *non-conflicting* [Calì, Lembo, and Rosati 2003a; Calì, Gottlob, and Pieris 2012]. Intuitively, this condition guarantees that the FDs can be ignored, as long as they hold on the initial instance I_0, and one can then solve the query answering problem by considering the IDs alone. However, the non-conflicting condition only applies to UQA and not to FQA. In fact it is known that even for very simple classes of IDs and FDs, including non-conflicting classes, FQA and UQA do not coincide. Rosati [Rosati 2011] further showed that FQA is generally undecidable for non-conflicting IDs and FDs.

Thus, we do not know to what what extent these classes, FDs and IDs, can be combined while retaining decidable FQA. The only known decidable cases impose severe requirements. For example, the constraint class of *key-based* dependencies, generalized in [Rosati 2006] to *single key dependencies (KDs) and foreign keys (FKs)*, has decidable FQA, but such constraints cannot model, e.g., FDs which are not keys. Further, in contrast with the general case of FDs and IDs, these languages are always finitely controllable, which limits their expressiveness.

A second decidable case is where all relation symbols and all subformulae of the constraints have arity at most two. In this context, the results of [Pratt-Hartmann 2009] imply the decidability of both FQA and UQA for a very rich non-finitely-controllable sublogic of first-order logic. The complexity of FQA has recently been isolated for some fragments of this arity-two logic [Ibáñez-García, Lutz, and Schneider 2014]. Yet these results do not apply to arbitrary arity signatures. In a nutshell, *we know of no tools to solve FQA for non-finitely-controllable constraints on signatures of arbitrary arity.*

This part of my thesis presents the *first decidability result for finite query answering under non-finitely-controllable IDs and FDs over relations of arbitrary arity.* As the problem is undecidable in general, we must naturally make some restriction. Our choice is to limit to *Unary IDs* (UIDs), which export only one variable: for instance, the rule $\forall xyz\ R(x, y, z) \rightarrow \exists w\ S(w, x)$ is a UID. We study UIDs and FDs because they are not finitely controllable (see, e.g., [Rosati 2006, Theorem 6]), and can be used to model, e.g., single-attribute foreign keys, a common use case in database systems. The decidability of UQA for UIDs and FDs is known because they are

always non-conflicting, and [Johnson and Klug 1984] showed that UIDs in isolation are finitely controllable. This part of my thesis shows that *the FQA problem is decidable for arbitrary UIDs and FDs*, from which we deduce tight bounds on its complexity.

The idea is to *reduce the finite case to the unrestricted case*, but in a more complex way than by finite controllability. We make use of a technique originating in [Cosmadakis, Kanellakis, and Vardi 1990] to study finite implication on UIDs and FDs: the *finite closure* operation, which takes a conjunction of UIDs and FDs and determines exactly which additional UIDs and FDs are implied over finite instances. Indeed, other works [Rosati 2008; Ibáñez-García, Lutz, and Schneider 2014] have used the finite closure operation in their study of constraint classes over schemas of arity two. Such works showed that finite query answering for a query q, instance I_0, and constraints Σ reduces to unrestricted query answering for I_0, q, and the finite closure of Σ. In other words, the closure construction which is sound for implication is also sound for query answering.

We show that the same approach applies to arbitrary arity signatures, for the language of UIDs and FDs. Our main result thus reduces finite query answering to unrestricted query answering, for UIDs and FDs in arbitrary arity:

Theorem 9.3 (Main theorem)**.** *Conjunctions of FDs and UIDs are finitely controllable up to finite closure: for any finite instance I_0, conjunctive query q, and constraints Σ consisting of UIDs and FDs, the finite open-world query answering problem for I_0 and q under Σ has the same answer as the unrestricted open-world query answering problem for I_0 and q under the finite closure of Σ.*

Using the known results about the complexity of UQA for UIDs, we isolate the precise complexity of FQA with respect to UIDs and FDs, showing that it matches that of UQA:

Theorem 9.2. *The combined complexity of the finite open-world query answering problem for UIDs and FDs is* NP*-complete, and it is* AC^0 *in data complexity (that is, when the constraints and query are fixed).*

The remaining chapters of the manuscript give the complete proof of Theorem 9.3. The proof borrows and adapts a variety of techniques from prior work:

- using k-bounded simulations to preserve small acyclic CQs [Ibáñez-García, Lutz, and Schneider 2014],
- partitioning UIDs into components that have limited interaction, and satisfying the UIDs component-by-component [Cosmadakis, Kanellakis, and Vardi 1990; Ibáñez-García, Lutz, and Schneider 2014],
- performing a chase that reuses sufficiently similar elements [Rosati 2011],
- taking the product with groups of large girth to blow up cycles [Otto 2002].

However, the proof must also face many new difficulties that do not arise in the arity-two case (or in the case of IDs in isolation), to deal with number restrictions in our arbitrary arity setting. For instance, our chase can only reuse elements at positions (the *non-dangerous positions*) where this is permitted by the functional dependencies; at other positions, it connects together elements that can be matched in a *piecewise*

way, relying on the finite closure process to ensure that cardinalities match. These connections must be done while preserving a k-bounded simulation, for which we must perform preliminary chasing up to a sufficient distance before we can connect elements, relying on a structural observation on the chase by UIDs (Theorem 12.4.3), and on the component-wise decomposition of the constraints. Further, even when element reuses are possible when chasing, we must make sure that we do not violate the higher-arity FDs, which the finite closure process essentially does not restrict: we do so by shuffling the reuses following a combinatorial construction of "dense" models of FDs (Theorem 14.1.9). The group product operation to blow up cycles must also be adjusted to preserve fact overlaps and avoid violating the FDs.

The next chapter presents the required preliminaries for this part. Chapter 9 then gives more details about how to rephrase the result to a claim about the existence of finite universal models, and gives a roadmap of the successive steps of the proof. Each step of the proof is then presented in a separate chapter that builds on the previous ones.

Chapter 8

Preliminaries

8.1 Instances and Constraints

Instances. We assume an infinite countable set of *elements* (or *values*) $a, b, c, \ldots$ and *variable names* $x, y, z, \ldots$. A *schema* σ consists of *relation names* (e.g., R) with an *arity* (e.g., $|R|$) which we assume is $\geqslant 1$: we write $|\sigma| := \max_{R \in \sigma} |R|$. Following the *unnamed perspective*, the set of *positions* of R is $\mathsf{Pos}(R) := \{R^i \mid 1 \leqslant i \leqslant |R|\}$, and we define $\mathsf{Pos}(\sigma) := \bigsqcup_{R \in \sigma} \mathsf{Pos}(R)$, where we use $\sqcup$ to denote disjoint union. We identify R^i and i when no confusion can result.

A relational *instance* I of σ is a set of *ground facts* of the form $R(\boldsymbol{a})$ where R is a relation name and $\boldsymbol{a}$ an $|R|$-tuple of values. The *size* $|I|$ of a finite instance I is its number of facts. The *active domain* $\mathrm{dom}(I)$ of I is the set of the elements which appear in I. For any position $R^i \in \mathsf{Pos}(\sigma)$, we define the *projection* $\pi_{R^i}(I)$ of I to R^i as the set of the elements of $\mathrm{dom}(I)$ that occur at position R^i in I. For $L \subseteq \mathsf{Pos}(R)$, the projection $\pi_L(I)$ is a set of $|L|$-tuples defined analogously; for convenience, departing from the unnamed perspective, we index those tuples by the positions of L rather than by $\{1, \ldots, |L|\}$. A *superinstance* of I is a (not necessarily finite) instance I' such that $I \subseteq I'$.

A *homomorphism* from an instance I to an instance I' is a mapping $h : \mathrm{dom}(I) \to \mathrm{dom}(I')$ such that, for every fact $F = R(\boldsymbol{a})$ of I, the fact $h(F) := R(h(a_1), \ldots, h(a_{|R|}))$ is in I'.

Constraints. We consider *integrity constraints* (or *dependencies*) which are special sentences of first-order logic without function symbols. We write $I \models \Sigma$ when instance I satisfies constraints Σ, and we then call I a *model* of Σ.

An *inclusion dependency* ID is a sentence of the form $\tau : \forall \boldsymbol{x}\, R(x_1, \ldots, x_n) \to \exists \boldsymbol{y}\, S(z_1, \ldots, z_m)$, where $\boldsymbol{z} \subseteq \boldsymbol{x} \cup \boldsymbol{y}$ and no variable occurs at two different positions of the same fact. The *exported variables* are the variables of $\boldsymbol{x}$ that occur in $\boldsymbol{z}$. This work only studies *unary inclusion dependencies* (UIDs) which are the IDs with exactly one exported variable. We write a UID τ as $R^p \subseteq S^q$, where R^p and S^q are the positions of $R(\boldsymbol{x})$ and $S(\boldsymbol{z})$ where the exported variable occurs. For instance, the UID $\forall xy\, R(x, y) \to \exists z\, S(y, z)$ is written $R^2 \subseteq S^1$. We assume without loss of generality that there are no *trivial* UIDs of the form $R^p \subseteq R^p$.

A *functional dependency* FD is a sentence of the form $\varphi : \forall \boldsymbol{xy}\, (R(x_1, \ldots, x_n) \wedge R(y_1, \ldots, y_n) \wedge \bigwedge_{R^l \in L} x_l = y_l) \to x_r = y_r$, where $L \subseteq \mathsf{Pos}(R)$ and $R^r \in \mathsf{Pos}(R)$. For brevity, we write φ as $R^L \to R^r$. We call φ a *unary functional dependency* UFD if

$|L| = 1$; otherwise it is *higher-arity*. For instance, $\forall xx'yy'\, R(x, x') \wedge R(y, y') \wedge x' = y' \to x = y$ is a UFD, and we write it $R^2 \to R^1$. We assume that $|L| > 0$, i.e., we do not allow nonstandard or degenerate FDs. We call φ *trivial* if $R^r \in R^L$, in which case φ always holds; again we disallow trivial FDs. Two facts $R(\boldsymbol{a})$ and $R(\boldsymbol{b})$ *violate* a non-trivial FD φ if $\pi_L(\boldsymbol{a}) = \pi_L(\boldsymbol{b})$ but $a_r \neq b_r$.

For $L, L' \subseteq \mathsf{Pos}(R)$, we write $R^L \to R^{L'}$ the conjunction of FDs $R^L \to R^l$ for $R^l \in L'$. In particular, conjunctions of the form $\kappa : R^L \to R$ (i.e., $L' = \mathsf{Pos}(R)$) are called *key dependencies*. The key κ is *unary* if $|L| = 1$. If κ holds on a relation R, we call L a *key* (or *unary key*) of R.

8.2 Implication and Finite Implication

We say that a conjunction of constraints Σ in a class CL *finitely implies* a constraint φ if any *finite* instance that satisfies Σ also satisfies φ. We say that Σ *implies* φ if the same holds even for infinite instances. The *closure* Σ^* of Σ is the set of constraints of CL which are implied by Σ, and the *finite closure* $\Sigma^{\mathrm{f}*}$ is the set of those which are finitely implied.

An *axiomatization* of implication for CL is a set of deduction rules (which, given dependencies in CL, deduce new dependencies in CL), with the following property: for any conjunction Σ of dependencies in CL, letting Σ' be the result of defining $\Sigma' := \Sigma$ and applying iteratively the deduction rules while possible to inflate Σ', then Σ' is *exactly* Σ^*. An *axiomatization of finite implication* is defined similarly but for $\Sigma^{\mathrm{f}*}$.

Implication for IDs. Given a set Σ of IDs, it is known [Casanova, Fagin, and Papadimitriou 1984] that an ID τ is implied by Σ iff it is finitely implied. Further, when Σ are UIDs, we can easily compute in PTIME the set of implied UIDs (from which we exclude the trivial ones), by closing Σ under the *transitivity rule* [Casanova, Fagin, and Papadimitriou 1984]: if $R^p \subseteq S^q$ and $S^q \subseteq T^r$ hold in Σ, then so is $R^p \subseteq T^r$ unless it is trivial. We call Σ *transitively closed* if it is thus closed.

Implication for FDs. Again, a set Σ_{FD} of FDs implies an FD φ iff it finitely implies it: see, e.g., [Cosmadakis, Kanellakis, and Vardi 1990]. The standard axiomatization of FD implication is given in [Armstrong 1974], and includes the *transitivity rule*: for any $R \in \sigma$ and $L, L', L'' \subseteq \mathsf{Pos}(R)$, if $R^L \to R^{L'}$ and $R^{L'} \to R^{L''}$ hold in Σ_{FD}, then so does $R^L \to R^{L''}$.

Implication for UIDs and FDs. It was shown in [Cosmadakis, Kanellakis, and Vardi 1990] that implication for conjunctions Σ_{UID} of UIDs and Σ_{FD} of FDs can be axiomatized by the above UID and FD rules in isolation. However, for *finite* implication, we must add a *cycle rule*, which we now define.

Let Σ be a conjunction of dependencies formed of UIDs Σ_{UID} and FDs Σ_{FD}. Define the *reverse* of an UFD $\varphi : R^p \to R^q$ as $\varphi^{-1} := R^q \to R^p$, and the *reverse* of a UID $\tau : R^p \subseteq S^q$ as $\tau^{-1} := S^q \subseteq R^p$. A *cycle* in Σ is a sequence of UIDs and UFDs of Σ_{UID} and Σ_{FD} of the following form: $R_1^{p_1} \subseteq R_2^{q_2}$, $R_2^{p_2} \to R_2^{q_2}$, $R_2^{p_2} \subseteq R_3^{q_3}$, $R_3^{p_3} \to R_3^{q_3}$, …, $R_{n-1}^{p_{n-1}} \subseteq R_n^{q_n}$, $R_n^{p_n} \to R_n^{q_n}$, $R_n^{p_n} \subseteq R_1^{q_1}$, $R_1^{p_1} \to R_1^{q_1}$. The *cycle rule*, out of such a cycle, deduces the reverse of each UID and of each UFD in the cycle. We then have:

Theorem 8.2.1 ([Cosmadakis, Kanellakis, and Vardi 1990], Theorem 4.1)**.** *The* UID *and* FD *deduction rules and the cycle rule are an axiomatization of finite implication for* UIDs *and* FDs*.*

In terms of complexity, this implies:

Corollary 8.2.2 ([Cosmadakis, Kanellakis, and Vardi 1990], Corollary 4.4)**.** *Given* UIDs Σ_{UID} *and* FDs Σ_{FD}*, and a* UID *or* FD τ*, we can check in PTIME whether* τ *is finitely implied by* Σ_{UID} *and* Σ_{FD}*.*

8.3 Queries and QA

Queries. An *atom* $A = R(\boldsymbol{t})$ consists of a relation name R and an $|R|$-tuple $\boldsymbol{t}$ of variables or constants. This work studies the *conjunctive queries* CQ, which are existentially quantified conjunctions of atoms, such that each variable in the quantification occurs in some atom. The *size* $|q|$ of a CQ q is its number of atoms. A CQ is *Boolean* if it has no free variables.

A Boolean CQ q *holds* in an instance I exactly when there is a homomorphism h from the atoms of q to I such that h is the identity on the constants of q (we call this a *homomorphism from* q *to* I). We call such an h a *match* of q in I, and by a slight abuse of terminology we also call the image of h a *match* of q in I.

QA problems. We define the *unrestricted open-world query answering* problem (UQA) as follows: given a finite instance I, a conjunction of constraints Σ, and a Boolean CQ q, decide whether there is a superinstance of I that satisfies Σ and violates q. If there is none, we say that I and Σ *entail* q and write $(I, \Sigma) \models_{\text{unr}} q$. In other words, UQA asks whether the first-order formula $I \wedge \Sigma \wedge \neg q$ has some (possibly infinite) model.

This work focuses on the *finite query answering problem* (FQA), which is the variant of open-world query answering where we require the counterexample superinstance to be *finite*; if no such counterexample exists, we write $(I, \Sigma) \models_{\text{fin}} q$. Of course $(I, \Sigma) \models_{\text{unr}} q$ implies $(I, \Sigma) \models_{\text{fin}} q$.

The *combined complexity* of the UQA and FQA problems, for a fixed class CL of constraints, is the complexity of deciding it when all of I, Σ (in CL) and q are given as input. The *data complexity* is defined by assuming that Σ and q are fixed, and only I is given as input.

Assumptions on queries. Throughout this work, we will make three assumptions about CQs, without loss of generality for UQA and FQA. First, *we assume that CQs are constant-free*. Indeed, for each constant $c \in \text{dom}(I_0)$, we could otherwise do the following: add a fresh relation P_c to the signature, add a fact $P_c(c)$ to I_0, replace c in q by an existentially quantified variable x_c, and add the atom $P_c(x_c)$ to q. It is then clear that UQA with the rewritten instance and query is equivalent to UQA with the original instance and query under any constraints (remember that our constraints are constant-free); the same is true for FQA.

Second, we *assume all CQs to be Boolean*, unless otherwise specified. Indeed, to perform UQA for non-Boolean queries (where the domain of the free variables is that of the base instance I_0), we can always enumerate all possible assignments, and

solve our problem by solving polynomially many instances of the UQA problem with Boolean queries. Again, the same is true of FQA.

Third, *we assume all CQs to be connected*. A CQ q is *disconnected* if there is a partition of its atoms in two non-empty sets $\mathcal{A}$ and $\mathcal{A}'$, such that no variable occurs both in an atom of $\mathcal{A}$ and in one atom of $\mathcal{A}'$. In this case, the query $q : \exists \boldsymbol{x}\boldsymbol{y}\, \mathcal{A}(\boldsymbol{x}) \wedge \mathcal{A}(\boldsymbol{y})$ can be rewritten to $q_2 \wedge q_2'$, for two CQs q_2 and q_2' of strictly smaller size. In this part of my thesis, we will show that, on finitely closed dependencies, FQA and UQA coincide for *connected* queries. This clearly implies the same for disconnected queries, by considering all their connected subqueries. Hence, we can assume that queries are connected.

Chase. We say that a superinstance I' of an instance I is *universal* for constraints Σ if $I' \models \Sigma$ and if for any CQ q, $I' \models q$ iff $(I, \Sigma) \models_{\text{unr}} q$. We now recall the definition of the *chase* [Abiteboul, Hull, and Vianu 1995; Onet 2013], a standard construction of (generally infinite) universal superinstances. We assume that we have fixed an infinite set $\mathcal{N}$ of *nulls* which is disjoint from $\text{dom}(I)$. We only define the chase for transitively closed UIDs, which we call the *UID chase*.

We say that a fact $F_{\text{a}} = R(\boldsymbol{a})$ of an instance I is an *active fact* for a UID $\tau : R^p \subseteq S^q$ if, writing $\tau : \forall \boldsymbol{x}\, R(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y}\, S(\boldsymbol{z})$, there is a homomorphism from $R(\boldsymbol{x})$ to F_{a} but no such homomorphism can be extended to a homomorphism from $\{R(\boldsymbol{x}), S(\boldsymbol{z})\}$ to I. In this case we say that we *want* to apply the UID τ to a_p, written $a_p \in \mathsf{Wants}(I, \tau)$. Note that $\mathsf{Wants}(I, \tau) = \pi_{R^p}(I) \backslash \pi_{S^q}(I)$. For a conjunction Σ_{UID} of UIDs, we may also write $a \in \mathsf{Wants}_{\Sigma_{\text{UID}}}(I, S^q)$ if there is $\tau \in \Sigma_{\text{UID}}$ of the form $\tau : U^v \subseteq S^q$ such that $a \in \mathsf{Wants}(I, \tau)$; we drop the subscript when there is no ambiguity.

The result of a *chase step* on the active fact $F_{\text{a}} = R(\boldsymbol{a})$ for $\tau : R^p \subseteq S^q$ in I (we call this *applying* τ to F_{a}) is the superinstance I' of I obtained by adding a new fact $F_{\text{n}} = S(\boldsymbol{b})$ defined as follows: we set $b_q := a_p$, which we call the *exported element* (and S^q the *exported position* of F_{n}), and use fresh nulls from $\mathcal{N}$ to instantiate the existentially quantified variables of τ and complete F_{n}, using a different null at each position; we say the corresponding elements are *introduced* at F_{n}. This ensures that F_{a} is no longer an active fact in I' for τ.

A *chase round* of a conjunction Σ_{UID} of UIDs on I is the result of applying simultaneous chase steps on all active facts for all UIDs of Σ_{UID}, using distinct fresh elements. The *UID chase* $\mathsf{Chase}(I, \Sigma_{\text{UID}})$ of I by Σ_{UID} is the (generally infinite) fixpoint of applying chase rounds. It is a universal superinstance for Σ_{UID} [Fagin, Kolaitis, Miller, and Popa 2003].

As we are chasing by transitively closed UIDs, if we perform the *core chase* [Deutsch, Nash, and Remmel 2008; Onet 2013] rather than the UID chase that we just defined, we can ensure the following *Unique Witness Property*: for any element $a \in \text{dom}(\mathsf{Chase}(I, \Sigma_{\text{UID}}))$ and position R^p of σ, if two different facts of $\mathsf{Chase}(I, \Sigma_{\text{UID}})$ contain a at position R^p, then they are both facts of I. In our context, however, the core chase matches the UID chase defined above, except at the first round. Thus, modulo the first round, by $\mathsf{Chase}(I, \Sigma_{\text{UID}})$ we refer to the UID chase, which has the Unique Witness Property. See Section 8.4 for details.

Finite controllability. We say a conjunction of constraints Σ is *finitely controllable* for CQ if FQA and UQA coincide: for every finite instance I and every Boolean

CQ q, $(I, \Sigma) \models_{\text{unr}} q$ iff $(I, \Sigma) \models_{\text{fin}} q$.

It was shown in [Rosati 2006; Rosati 2011] that, while conjunctions of IDs are finitely controllable, even conjunctions of UIDs and FDs may not be. It was later shown in [Rosati 2008] that the finite closure process could be used to reduce UQA to FQA for some constraints on relations of arity at most two. Following the same idea, we say that a conjunction of constraints Σ is *finitely controllable up to finite closure* if for every finite instance I, and Boolean CQ q, $(I, \Sigma) \models_{\text{fin}} q$ iff $(I, \Sigma^{\text{f}*}) \models_{\text{unr}} q$, where $\Sigma^{\text{f}*}$ is the finite closure defined by Theorem 8.2.1. If Σ is finitely controllable up to finite closure, then we can reduce FQA to UQA, even if finite controllability does not hold, by computing the finite closure $\Sigma^{\text{f}*}$ of Σ and solving UQA on $\Sigma^{\text{f}*}$.

8.4 Details about the UID Chase

Recall the *Unique Witness Property*:

For any element $a \in \text{dom}(\mathsf{Chase}(I, \Sigma_{\text{UID}}))$ and position R^p of σ, if two facts of $\mathsf{Chase}(I, \Sigma_{\text{UID}})$ contain a at position R^p, then they are both facts of I.

We first give an example showing why this may not be guaranteed by the first round of the UID chase. Consider the instance $I = \{R(a), S(a)\}$ and the UIDs $\tau_1 : R^1 \subseteq T^1$ and $\tau_2 : S^1 \subseteq T^1$, where T is a binary relation. Applying a round of the UID chase creates the instance $\{R(a), S(a), T(a, b_1), T(a, b_2)\}$, with $T(a, b_1)$ being created by applying τ_1 to the active fact $R(a)$, and $T(a, b_2)$ being created by applying τ_2 to the active fact $S(a)$.

By contrast, the core chase would create only one of these two facts, because it would consider that two new facts are *equivalent*: they have the same exported element occurring at the same position. In general, the core chase keeps only one fact within each class of facts that are equivalent in this sense.

However, after one chase round by the core chase, there is no longer any distinction between the UID chase and the core chase, because the following property holds on the result I' of a chase round (be it by the core chase or by the UID chase) on any instance I'': (*) for any $\tau \in \Sigma_{\text{UID}}$ and element $a \in \mathsf{Wants}(I', \tau)$, a occurs in only one fact of I'. This is true because Σ_{UID} is transitively closed, so we know that no UID of Σ_{UID} is applicable to an element of $\text{dom}(I'')$ in I'; hence the only elements that witness violations occur in the one fact where they were introduced in I'.

We now claim that (*) implies that the Unique Witness Property holds when we chase by the core chase for the first round and the UID chase for subsequent rounds. Indeed, assume to the contrary that $a \in \text{dom}(\mathsf{Chase}(I, \Sigma_{\text{UID}}))$ violates the Property.

If $a \in \text{dom}(I)$, because Σ_{UID} is transitively closed, after the first chase round on I, we no longer create any fact that involves a. Hence, each one of F_1 and F_2 is either a fact of I or a fact created in the first round of the chase (which is a chase round by the core chase). However, if one of F_1 and F_2 is in I, then it witnesses that we could not have $a \in \mathsf{Wants}(I, R^p)$, so it is not possible that the other fact was created in the first chase round. It cannot be the case either that F_1 and F_2 were both created in the first chase round, by definition of the core chase. Hence, F_1 and F_2 are necessarily both facts of I.

If $a \in \text{dom}(\mathsf{Chase}(I, \Sigma_{\text{UID}})) \setminus \text{dom}(I)$, assume that a occurs at position R^p in two facts F_1, F_2. As $a \notin \text{dom}(I)$, none of them is a fact of I. We then show a contradiction. It is not possible that one of those facts was created in a chase

round before the other, as otherwise the second created fact could not have been created because of the first created fact. Hence, both facts must have been created in the same chase round. So there was a chase round from I'' to I' where we had $a \in \mathsf{Wants}(I'', R^p)$ and both F_1 and F_2 were created respectively from active facts F_1' and F_2' of I'' by UIDs $\tau_1 : S^q \subseteq R^p$ and $\tau_2 : T^r \subseteq R^p$. But then, by property (*), a occurs in only one fact, so as it occurs in F_1' and F_2' we have $F_1' = F_2'$. Further, as $a \notin \mathrm{dom}(I)$, F_1' and F_2' are not facts of I either, so by definition of the UID chase and of the core chase, it is easy to see a occurs at only one position in $F_1' = F_2'$. This implies that $\tau_1 = \tau_2$. Hence, we must have $F_1 = F_2$, a contradiction. This concludes the proof of the Unique Witness Property.

Chapter 9

Main Result and Overall Approach

We study open-world query answering for FDs and UIDs. For UQA, the following is already known:

Proposition 9.1. *UQA for FDs and UIDs has* AC^0 *data complexity and* NP*-complete combined complexity.*

Proof. UQA for UIDs in isolation is NP-complete in combined complexity. The lower bound is immediate from query evaluation [Abiteboul, Hull, and Vianu 1995], and [Johnson and Klug 1984] showed an NP upper bound for IDs with any fixed bound on the number of exported variables (which they call "width": in their terminology, UIDs are IDs of width 1). For data complexity, the upper bound is from the first-order rewritability of certain answers for arbitrary IDs, from [Calì, Lembo, and Rosati 2003b].

For UIDs and FDs, clearly the lower bound on combined complexity also applies. The upper bounds are proved by observing that UIDs and FDs are *separable*, namely, for any FDs Σ_{FD} and UIDs Σ_{UID}, for any instance I_0 and CQ q, if $I_0 \models \Sigma_{\mathrm{FD}}$ then we have $(I_0, \Sigma_{\mathrm{FD}} \wedge \Sigma_{\mathrm{UID}}) \models_{\mathrm{unr}} q$ iff $(I_0, \Sigma_{\mathrm{UID}}) \models_{\mathrm{unr}} q$. Assuming separability, to decide UQA for Σ_{UID} and Σ_{FD}, we first check whether $I_0 \models \Sigma_{\mathrm{FD}}$, in PTIME combined complexity, and AC^0 data complexity as Σ_{FD} is expressible in first-order logic. If $I_0 \not\models \Sigma_{\mathrm{FD}}$, UQA is vacuously true. Otherwise, we then determine whether $(I_0, \Sigma_{\mathrm{UID}}) \models_{\mathrm{unr}} q$, using the upper bound for UQA for UIDs. By separability, the answer to UQA under Σ_{UID} is the same as the answer to UQA under $\Sigma_{\mathrm{FD}} \wedge \Sigma_{\mathrm{UID}}$.

Hence, all that remains to show is that UIDs and FDs are always separable. This follows from the *non-conflicting condition* of [Calì, Lembo, and Rosati 2003a; Calì, Gottlob, and Pieris 2012] but we give a simpler self-contained argument. Assume that I_0 satisfies Σ_{FD}. It is obvious that $(I_0, \Sigma_{\mathrm{UID}}) \models_{\mathrm{unr}} q$ implies $(I_0, \Sigma_{\mathrm{FD}} \wedge \Sigma_{\mathrm{UID}}) \models_{\mathrm{unr}} q$, so let us prove the converse implication. We do it by noticing that the chase $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$ satisfies Σ_{FD}. Indeed, assuming to the contrary the existence of F and F' in $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$ violating an FD of Σ_{FD}, there must exist a position $R^p \in \mathsf{Pos}(\sigma)$ such that $\pi_{R^p}(F) = \pi_{R^p}(F')$. Yet, by the Unique Witness Property, this implies that F and F' are facts of I_0, but we assumed that $I_0 \models \Sigma_{\mathrm{FD}}$, a contradiction.

Hence, $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$ satisfies Σ_{FD}, so it is a superinstance of I_0 that satisfies $\Sigma_{\mathrm{FD}} \wedge \Sigma_{\mathrm{UID}}$. Hence, $(I_0, \Sigma_{\mathrm{FD}} \wedge \Sigma_{\mathrm{UID}}) \models_{\mathrm{unr}} q$ implies that we must have $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}}) \models q$. By universality of the chase, this implies $(I_0, \Sigma_{\mathrm{UID}}) \models_{\mathrm{unr}} q$.

Hence, the converse implication is proven, so the two UQA problems are equivalent, which implies that Σ_{UID} and Σ_{FD} are separable. □

In the *finite case*, however, even the decidability of FQA for FDs and UIDs was not known. This part of my thesis shows that it is decidable, and that the complexity matches that of UQA:

Theorem 9.2. *The combined complexity of the finite open-world query answering problem for UIDs and FDs is* NP*-complete, and it is* AC^0 *in data complexity (that is, when the constraints and query are fixed).*

This result follows from our Main Theorem, which is proven in the rest of this part of my thesis:

Theorem 9.3 (Main theorem)**.** *Conjunctions of FDs and UIDs are finitely controllable up to finite closure: for any finite instance* I_0*, conjunctive query* q*, and constraints* Σ *consisting of UIDs and FDs, the finite open-world query answering problem for* I_0 *and* q *under* Σ *has the same answer as the unrestricted open-world query answering problem for* I_0 *and* q *under the finite closure of* Σ*.*

From the Main Theorem, we can prove Theorem 9.2, using the closure process of [Cosmadakis, Kanellakis, and Vardi 1990]:

Proof of Theorem 9.2. Again, the NP-hardness lower bound is immediate from query evaluation [Abiteboul, Hull, and Vianu 1995], so we only show the upper bounds. Consider an instance of FQA for FDs and UIDs, consisting of an instance I_0, a conjunction Σ of IDs Σ_{UID} and FDs Σ_{FD}, and a CQ q. Let Σ^*_{FD} be the FDs and Σ^*_{UID} the UIDs of the finite closure $\Sigma^{\text{f}*}$. By our Main Theorem, we have $(I_0, \Sigma) \models_{\text{fin}} q$ iff $(I_0, \Sigma^{\text{f}*}) \models_{\text{unr}} q$. As the computation of $\Sigma^{\text{f}*}$ from Σ is data-independent, the data complexity upper bounds follow from Proposition 9.1, so we need only show the combined complexity upper bound.

Materializing $\Sigma^{\text{f}*}$ from the input may take exponential time, which we cannot afford, so we need a more clever approach. Remember from the proof of Proposition 9.1 that, as $\Sigma^{\text{f}*}$ consists of UIDs and FDs, it is separable. Hence, to solve UQA for I_0, $\Sigma^{\text{f}*}$ and q, as $\Sigma^{\text{f}*}$ is separable, we need to perform two steps:

1. Check whether $I_0 \models \Sigma^*_{\text{FD}}$;
2. If yes, solve UQA for I_0, Σ^*_{UID} and q.

To perform step 1, compute in PTIME the set Σ^*_{UFD} of the UFDs of $\Sigma^{\text{f}*}$, using Corollary 8.2.2. By [Cosmadakis, Kanellakis, and Vardi 1990] (remark above Corollary 4.4), all non-unary FDs in Σ^*_{FD} are implied by $\Sigma^*_{\text{UFD}} \wedge \Sigma_{\text{FD}}$ under the axiomatization of FD implication; hence, to check whether $I_0 \models \Sigma^*_{\text{FD}}$, it suffices to check whether $I_0 \models \Sigma^*_{\text{UFD}}$ and $I_0 \models \Sigma_{\text{FD}}$, which we do in PTIME.

To perform step 2, compute Σ^*_{UID} in PTIME by considering each possible UID (there are polynomially many) and determining in PTIME from Σ whether it is in $\Sigma^{\text{f}*}$, using Corollary 8.2.2. Then, solve UQA in NP combined complexity by Proposition 9.1. The entire process takes NP combined complexity, and the answer matches that of FQA by our Main Theorem, which proves the NP upper bound. □

In this chapter, we first explain how we can prove Theorem 9.3 from a different statement, namely: we can construct *finite universal models* for finitely closed UIDs and FDs. We conclude this chapter with the outline of the proof of this result (Theorem 9.1.3) which will be developed in the rest of this part of my thesis.

9.1 Finite Universal Superinstances

Our Main Theorem claims that a certain class of constraints, namely finitely closed UIDs and FDs, are finitely controllable for the class of conjunctive queries (CQ). To prove this, it will be easier to work with a notion of *k-sound* and *k-universal instances*.

Definition 9.1.1. For $k \in \mathbb{N}$, we say that a superinstance I of an instance I_0 is *k-sound* for constraints Σ and CQs (and I_0) if, for every CQ q of size $\leqslant k$ such that $I \models q$, we have $(I_0, \Sigma) \models_{\text{unr}} q$. We say it is *k-universal* if the converse also holds: $I \models q$ whenever $(I_0, \Sigma) \models_{\text{unr}} q$. For a subclass Q of CQs, we call I *k-sound* or *k-universal* for Σ and Q if the same holds for all queries q of size $\leqslant k$ that are in Q.

We say that a class CL of constraints *has finite universal superinstances* for a class Q of CQs, if for any constraints Σ of CL, for any $k \in \mathbb{N}$, for any instance I_0, if I_0 has some superinstance that satisfies Σ, then it has a *finite* superinstance that satisfies Σ and is k-sound for Σ and Q (and hence is also k-universal for Σ and Q). ◁

We will thus show that the class of finitely closed UIDs and FDs have finite universal superinstances for CQs. We explain why this implies our Main Theorem:

Proposition 9.1.2. *If constraint class CL has finite universal superinstances for query class Q, then CL is finitely controllable for Q.*

Proof. Let Σ be constraints in CL, I_0 be a finite instance and q be a query in Q. We show that $(I_0, \Sigma) \models_{\text{unr}} q$ iff $(I_0, \Sigma) \models_{\text{fin}} q$. The forward implication is immediate: if all superinstances of I_0 that satisfy Σ must satisfy q, then so do the finite ones.

For the converse implication, assume that $(I_0, \Sigma) \not\models_{\text{unr}} q$. In particular, this implies that I_0 has some superinstance that satisfies Σ, as otherwise the entailment would be vacuously true. As CL has finite universal superinstances for Q, let I be a finite k-sound superinstance of I_0 that satisfies Σ, where $k := |q|$. As I is k-sound, we have $I \not\models q$, and as $I \models \Sigma$, I witnesses that $(I_0, \Sigma) \not\models_{\text{fin}} q$. This proves the converse direction, so we have established finite controllability. □

So, the rest of part of my thesis actually shows the following restatement of the Main Theorem:

Theorem 9.1.3 (Universal models)**.** *The class of finitely closed UIDs and FDs has finite universal models for CQ: for every conjunction Σ of FDs Σ_{FD} and UIDs Σ_{UID} closed under finite implication, for any $k \in \mathbb{N}$, for every finite instance I_0 that satisfies Σ_{FD}, there exists a finite k-sound superinstance I of I_0 that satisfies Σ.*

Indeed, once we have shown this, we can easily deduce the Main Theorem, namely, that any conjunction Σ of FDs and UIDs is finitely controllable up to finite closure. Indeed, for any such Σ, for any instance I_0 and CQ q, we have $(I_0, \Sigma) \models_{\text{fin}} q$ iff $(I_0, \Sigma^{\text{f}*}) \models_{\text{fin}} q$: the forward statement is because any finite model of Σ is a model

Table 9.1: Roadmap of intermediate results.

	Signature	Universality	Constraints	Query
Chapter 10:	binary	weakly-sound	reversible UIDs, UFDs	ACQ
Chapter 11:	**arbitrary**	weakly-sound	reversible UIDs, UFDs	ACQ
Chapter 12:	arbitrary	***k*-sound**	reversible UIDs, UFDs	ACQ
Chapter 13:	arbitrary	k-sound	**finitely closed** UIDs, UFDs	ACQ
Chapter 14:	arbitrary	k-sound	finitely closed UIDs, **FDs**	ACQ
Chapter 15:	arbitrary	k-sound	finitely closed UIDs, FDs	**CQ**

of $\Sigma^{\mathrm{f}*}$, and the backward statement is tautological. Now, from the Universal Models Theorem and Proposition 9.1.2, we know that $\Sigma^{\mathrm{f}*}$ is finitely controllable, so that $(I_0, \Sigma^{\mathrm{f}*}) \models_{\mathrm{fin}} q$ iff $(I_0, \Sigma^{\mathrm{f}*}) \models_{\mathrm{unr}} q$. We have thus shown that $(I_0, \Sigma) \models_{\mathrm{fin}} q$ iff $(I_0, \Sigma^{\mathrm{f}*}) \models_{\mathrm{unr}} q$, which concludes the proof of the Main Theorem.

Hence, we will show the Universal Models Theorem in the rest of this part of my thesis. We proceed in incremental steps, following the plan that we outline next.

9.2 Proof Structure

We first make a simplifying assumption on the signature, without loss of generality, to remove *useless* relations. Given an instance I_0, UIDs Σ_{UID} and FDs Σ_{FD}, it may the be case that the signature σ contains a relation R that does not occur in $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$, namely, it does not occur in I_0 and the existence of an R-fact is not implied by Σ_{UID}. In this case, relation R is *useless*: a CQ q involving R will never be entailed under Σ, neither on unrestricted nor on finite models, unless I_0 has no completion at all satisfying the constraints. In any case, the query q can be replaced by the trivial CQ False, which is only (vacuously) entailed if there are no completions.

Hence, we can always remove useless relations from the signature, up to rewriting the query to the false query. Thus, without loss of generality, *we always assume that the signature contains no useless relations* in this sense: all relations of the signature occur in the chase.

We now present several assumptions that we use to prove weakenings of the Universal Models Theorem. The first one is on queries, which we require to be *acyclic*. The second is on FDs, which we require to be *unary*, i.e., UFDs. The third one is to replace k-soundness by the simpler notion of *weak-soundness*. Then we present two additional assumptions: the first one, reversible, is on the constraints, and requires that they have a certain special form; the second one, binary, is on the constraints and signature, which we require to be binary. In the next chapter, we show the Universal Models Theorem under all these assumptions, and then we lift the assumptions one by one, in each chapter. See Table 9.1 for a synopsis.

Hence, let us present the assumptions that we will make (and later lift).

Acyclic queries. It will be helpful to focus first on the subset of *acyclic* CQs, denoted ACQ, which are the queries that contain no *Berge cycle*. Formally:

Definition 9.2.1. A *Berge cycle* in a CQ q is a sequence $A_1, x_1, A_2, x_2, \ldots, A_n, x_n$ with $n \geqslant 2$, where the A_i are pairwise distinct atoms of q, the x_i are pairwise distinct variables of q, and x_i occurs in A_i and A_{i+1} for $1 \leqslant i \leqslant n$ (with addition modulo n, so x_n occurs in A_n and A_1). A query q is in ACQ if q has no Berge cycle and if no variable of q occurs more than once in the same atom.

Equivalently, consider the *incidence multigraph* of q, namely, the bipartite undirected multigraph on variables and atoms obtained by putting one edge between variable x and atom A for every time where x occurs in A (possibly multiple times). Then q is in ACQ iff its incidence multigraph is acyclic in the standard sense. ◁

Example 9.2.2. The queries $\exists x\ R(x,x)$, $\exists xy\ R(x,y) \wedge S(x,y)$, and $\exists xyz\ R(x,y) \wedge R(y,z) \wedge R(z,x)$ are not in ACQ: the first one has an atom with two occurrences of the same variable, the other two have a Berge cycle. The following query is in ACQ: $\exists xyzw\ R(x,y,z) \wedge S(x) \wedge T(y,w) \wedge U(w)$.

Intuitively, in the chase, all query matches are acyclic unless they involve some cycle in the initial instance I_0. Hence, only acyclic CQs have matches, except those that match on I_0 or those whose cycles have self-homomorphic matches, so, in a k-sound model, the CQs of size $\leqslant k$ which hold are usually acyclic. For this reason, we focus only on ACQ queries first. We will ensure in Chapter 15 that cyclic queries of size $\leqslant k$ have no matches.

Unary FDs. We will first show our result for *unary* FDs (UFDs); recall from Chapter 8 that they are the FDs with exactly one determining attribute. We do this because the finite closure construction of [Cosmadakis, Kanellakis, and Vardi 1990] is not concerned with higher-arity FDs, except for the UFDs that they imply. Hence, while the UFDs of the finite closure have a special structure that we can rely on, the higher-arity FDs are essentially arbitrary. This is why we deal with them only in Chapter 14, using a different approach.

***k*-soundness and weak-soundness.** Rather than proving that UIDs and UFDs have finite universal models for ACQ, it will be easier to prove first that they have *1-universal models*. More specifically, we will construct *weakly-sound superinstances*, which satisfy a *sufficient* condition for them to be *1-sound*:

Definition 9.2.3. A superinstance I of an instance I_0 is *weakly-sound* for a set of UIDs Σ_{UID} and for I_0 if the following holds:

- For any $a \in \text{dom}(I_0)$ and $R^p \in \mathsf{Pos}(\sigma)$, if $a \in \pi_{R^p}(I)$, then either $a \in \pi_{R^p}(I_0)$ or $a \in \mathsf{Wants}(I_0, R^p)$;
- For any $a \in \text{dom}(I) \setminus \text{dom}(I_0)$ and $R^p, S^q \in \mathsf{Pos}(\sigma)$, if a $\in \pi_{R^p}(I)$ and $a \in \pi_{S^q}(I)$ then either we have $R^p = S^q$ or $R^p \subseteq S^q$ and $S^q \subseteq R^p$ are in Σ_{UID}. ◁

Thus, we first show that UFDs and UIDs have *finite weakly-universal superinstances* for ACQ, defined analogously to Definition 9.1.1: for any constraints Σ_{U} of UFDs Σ_{UFD} and UIDs Σ_{UID}, for any query q in ACQ, for any instance I_0, if I_0 has a superinstance that satisfies Σ_{U}, then it has a *finite* superinstance that does and is weakly-sound for Σ_{UID} and I_0. This restriction is lifted in Chapter 12.

Assumption reversible. We will initially make a simplifying assumption on the structure of the UIDs and UFDs, which we call reversible. This assumption is motivated by the finite closure rules of Theorem 8.2.1; intuitively, it amounts to assuming that a certain *constraint graph* defined from the dependencies has a single connected component:

Definition 9.2.4. Let $\Sigma^{\text{rev}}_{\text{UID}}$ be a set of UIDs and Σ_{UFD} be a set of UFDs. We call $\Sigma^{\text{rev}}_{\text{UID}}$ and Σ_{UFD} *reversible* if:

- $\Sigma^{\text{rev}}_{\text{UID}}$ is closed under implication, and so is Σ_{UFD};
- All UIDs in $\Sigma^{\text{rev}}_{\text{UID}}$ are reversible (i.e., their reverses are also in $\Sigma^{\text{rev}}_{\text{UID}}$);
- for any UFD $\varphi : R^p \to R^q$ in Σ_{UFD}, if R^p occurs in some UID of $\Sigma^{\text{rev}}_{\text{UID}}$ and R^q also occurs in some UID of $\Sigma^{\text{rev}}_{\text{UID}}$, then φ is reversible, i.e., φ^{-1} is also in Σ_{UFD}. ◁

Assumption reversible: The UIDs Σ_{UID} and UFDs Σ_{UFD} are reversible.

When assumption reversible is made, we will write the UIDs $\Sigma^{\text{rev}}_{\text{UID}}$ rather than Σ_{UID}. Observe that $\Sigma^{\text{rev}}_{\text{UID}}$ and Σ_{UFD} are then finitely closed: they are closed under UID and UFD implication, and the UIDs and UFDs of any cycle must be reversible. To lift reversible and generalize to the general case, we will follow an SCC decomposition of the constraint graph to manage each SCC separately. See Chapter 13 for details.

Second assumption. We will start our proof in Chapter 10 by introducing important notions in the much simpler case of a binary signature. For this, we will initially make the following assumption binary on the signature and on Σ:

Assumption binary: Each relation R has arity 2 and the UFDs $R^1 \to R^2$ and $R^2 \to R^1$ hold in Σ.

We will lift this assumption in Chapter 11.

Roadmap. Each of the next chapters will prove that a certain constraint class has finite universal models for a certain query class in a certain sense, under certain assumptions. Table 9.1 summarizes the results that are proved in each chapter.

The rest of Part II of this manuscript follows this roadmap: each chapter starts by stating the result that it proves.

Chapter 10

Weak Soundness on Binary Signatures

Theorem 10.1. *Reversible UIDs and UFDs have finite weakly-universal superinstances for ACQs under assumption binary.*

We prove this result in this chapter. Fix an instance I_0 and reversible constraints $\Sigma_{\mathrm{U}}^{\mathrm{rev}}$ formed of UIDs $\Sigma_{\mathrm{UID}}^{\mathrm{rev}}$ and UFDs Σ_{UFD}. Assume that $I_0 \models \Sigma_{\mathrm{UFD}}$ as the question is vacuous otherwise, and make assumption binary.

Our goal is to construct a weakly-sound superinstance I of I_0 that satisfies $\Sigma_{\mathrm{U}}^{\mathrm{rev}}$. We do so by a *completion process* that adds new (binary) facts to connect elements together. As all possible UFDs hold, if we extend I_0 to I by adding a new fact $R(a_1, a_2)$, we must have $a_i \notin \pi_{R^i}(I_0)$ for $i \in \{1, 2\}$. Hence, by weak soundness, if $a_i \in \mathrm{dom}(I_0)$ then we must have $a_i \in \mathsf{Wants}(I_0, R^i)$. Our task is this section is thus to complete I_0 to I by adding R-facts, for each relation R, that connect together elements of $\mathsf{Wants}(I_0, R^1)$ and $\mathsf{Wants}(I_0, R^2)$.

10.1 Completing Balanced Instances

One easy situation to do this is when the instance I_0 is *balanced*: for every relation R, we can construct a bijection between the elements that want to be in R^1 and those that want to be in R^2:

Definition 10.1.1. We call I_0 *balanced* (for UIDs $\Sigma_{\mathrm{UID}}^{\mathrm{rev}}$) if, for every two positions R^p and R^q such that $R^p \to R^q$ and $R^q \to R^p$ are in Σ_{UFD}, we have $\left|\mathsf{Wants}_{\Sigma_{\mathrm{UID}}^{\mathrm{rev}}}(I_0, R^p)\right| = \left|\mathsf{Wants}_{\Sigma_{\mathrm{UID}}^{\mathrm{rev}}}(I_0, R^q)\right|$. ◁

If I_0 is balanced, we can show Theorem 10.1 by constructing I with $\mathrm{dom}(I) = \mathrm{dom}(I_0)$, adding new facts that pair together the existing elements:

Proposition 10.1.2. *Assuming binary and reversible, any balanced finite instance I_0 satisfying Σ_{UFD} has a finite weakly-sound superinstance I that satisfies $\Sigma_{\mathrm{U}}^{\mathrm{rev}}$, with $\mathrm{dom}(I) = \mathrm{dom}(I_0)$.*

We first exemplify this process:

Example 10.1.3. Consider four binary relations R, S, T, and U, with the UIDs $R^2 \subseteq S^1$, $S^2 \subseteq T^1$, $T^2 \subseteq R^1$ and their reverses, and the FDs prescribed by assumption

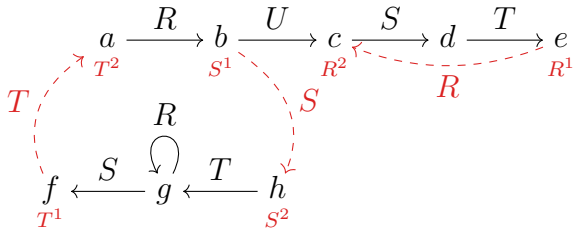


Figure 10.1: Balanced instances (see Example 10.1.3)

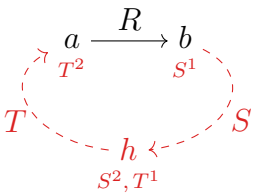


Figure 10.2: Balancing (see Example 10.2.1)

binary. Consider $I_0 := \{R(a,b), U(b,c), S(c,d), T(d,e), S(g,f), R(g,g), T(h,g)\}$, as depicted by the black elements and solid black arrows in Figure 10.1.

We compute, for each element, the set of positions where it wants to be, and write it in red under each element in Figure 10.1 (in this example, it is a singleton set for each element). For instance, we have $\mathsf{Wants}(I_0, T^1) = \{f\}$. We observe that the instance is balanced: we have $|\mathsf{Wants}(I_0, R^1)| = |\mathsf{Wants}(I_0, R^2)|$, and likewise for S, T, and U.

We can construct a weakly-sound superinstance I of I_0 as $I := I_0 \sqcup \{R(e,c), S(b,h), T(f,a)\}$: the additional facts are represented as dashed red arrows in Figure 10.1. Intuitively, we just create new facts that connect together elements to want to occur at the right positions.

We now give the formal proof of the result:

Proof of Proposition 10.1.2. Let us define a bijection f_R for every relation R of σ from $\mathsf{Wants}(I_0, R^1)$ to $\mathsf{Wants}(I_0, R^2)$; this is possible because I_0 is balanced.

Consider the superinstance I of I_0, with $\mathrm{dom}(I) = \mathrm{dom}(I_0)$, obtained by adding, for every R of σ, the fact $R(a, f_R(a))$ for every $a \in \mathsf{Wants}(I_0, R^1)$. I is clearly a finite weakly-sound superinstance of I_0, because for every $a \in \mathrm{dom}(I)$, if a occurs at some position R^p in some fact F of I, then either F is a fact of I_0 and $a \in \pi_{R^p}(I_0)$, or F is a new fact in $I \backslash I_0$ and by definition $a \in \mathsf{Wants}(I_0, R^p)$.

Let us show that $I \models \Sigma_{\mathrm{UFD}}$. Assume to the contrary that two facts $F = R(a_1, a_2)$ and $F' = R(a_1', a_2')$ in I witness a violation of a UFD $\varphi : R^1 \to R^2$ of Σ_{UFD}. As $I_0 \models \Sigma_{\mathrm{UFD}}$, one of F and F', say F, must be a new fact. By definition of the new facts, we have $a_1 \in \mathsf{Wants}(I_0, R^p)$, so that $a_1 \notin \pi_{R^1}(I_0)$. Now, as $\{F, F'\}$ is a violation, we must have $\pi_{R^1}(F) = \pi_{R^1}(F')$, so as $a_1 \notin \pi_{R^1}(I_0)$, F' must also be a new fact. Hence, by definition of the new facts, we have $a_2 = a_2' = f_R(a_1)$, so $F = F'$, which contradicts the fact that F and F' violate φ. For UFDs φ of the form $R^2 \to R^1$, the proof is similar, but we have $a_1 = a_1' = f_R^{-1}(a_2)$.

Let us now show that $I \models \Sigma^{\mathrm{rev}}_{\mathrm{UID}}$. Assume to the contrary that there is an active fact $F = R(a_1, a_2)$ that witnesses the violation of a UID $\tau : R^p \subseteq S^q$. If F is a fact of I_0, we had $a_p \in \mathsf{Wants}(I_0, S^q)$, so F cannot be an active fact in I as this violation was solved in I. So we must have $F \in I \backslash I_0$. Hence, by definition of the new facts, we had $a_p \in \mathsf{Wants}(I_0, R^p)$; so there must be $\tau' : T^r \subseteq R^p$ in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ such that $a_p \in \pi_{T^r}(I_0)$. Hence, because $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ is transitively closed, either $T^r = S^q$ or the UID $T^r \subseteq S^q$ is in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$. In the first case, as $a_p \in \pi_{T^r}(I_0)$, F cannot be an active fact for τ, a contradiction. In the second case, we had $a_p \in \mathsf{Wants}(I_0, S^q)$, so $a_p \in \pi_{S^q}(I)$ by definition of I, so again F cannot be an active fact for τ.

Hence, I is a finite weakly-sound superinstance of I_0 that satisfies $\Sigma^{\mathrm{rev}}_{\mathrm{U}}$ and with $\mathrm{dom}(I) = \mathrm{dom}(I_0)$, the desired claim. □

10.2 Adding Helper Elements

If our instance I_0 is not balanced, we cannot use the construction that we just presented. The idea is then to *make* I_0 *balanced*, which we do by adding "helper" elements that we assign to positions. The following example illustrates this:

Example 10.2.1. We use the same signature and dependencies as in Example 10.1.3. Consider $I_0 := \{R(a, b)\}$, as depicted in Figure 10.2. We have $a \in \mathsf{Wants}(I_0, T^2)$ and $b \in \mathsf{Wants}(I_0, S^1)$; however $\mathsf{Wants}(I_0, S^2) = \mathsf{Wants}(I_0, T^1) = \emptyset$, so I_0 is not balanced.

Still, we can construct the weakly-sound superinstance $I := I_0 \sqcup \{S(b, h), T(h, a)\}$ that satisfies the constraints. Intuitively, we have added a "helper" element h and "assigned" it to the positions $\{S^2, T^1\}$, so we could connect b to h with S and h to a with T.

We will formalize this idea of augmenting the domain with *helper elements*, as a *partially-specified superinstance*, namely, an instance that is augmented with helpers assigned to positions. However, we first need to understand at which positions the helpers can appear, without violating weak-soundness:

Definition 10.2.2. For any two positions R^p and S^q, we write $R^p \sim_{\mathsf{ID}} S^q$ when $R^p = S^q$ or when $R^p \subseteq S^q$ is in $\Sigma^{\text{rev}}_{\text{UID}}$ (and hence $S^q \subseteq R^p$ is in $\Sigma^{\text{rev}}_{\text{UID}}$ by assumption reversible). We write $[R^p]_{\mathsf{ID}}$ the $\sim_{\mathsf{ID}}$-class of R^p. ◁

As $\Sigma^{\text{rev}}_{\text{UID}}$ is transitively closed, $\sim_{\mathsf{ID}}$ is indeed an equivalence relation. Our choice of where to assign the helper elements will be represented as a mapping to an $\sim_{\mathsf{ID}}$-class. We call the result a *partially-specified superinstance*, or *pssinstance*:

Definition 10.2.3. A *pssinstance* of an instance I is a triple $P = (I, \mathcal{H}, \lambda)$ where $\mathcal{H}$ is a finite set of *helpers* and λ maps each $h \in \mathcal{H}$ to an $\sim_{\mathsf{ID}}$-class $\lambda(h)$.

We define $\mathsf{Wants}(P, R^p) := \mathsf{Wants}(I, R^p) \sqcup \{h \in \mathcal{H} \mid R^p \in \lambda(h)\}$. ◁

In other words, in the pssinstance, elements of I want to appear at the same positions as before, and helper elements want to occur at their $\sim_{\mathsf{ID}}$-class according to λ. A *realization* of a pssinstance P is then a superinstance of its underlying instance I which adds the helper elements, and whose additional facts respect $\mathsf{Wants}(P, R^p)$:

Definition 10.2.4. A *realization* of $P = (I, \mathcal{H}, \lambda)$ is a superinstance I' of I such that $\mathrm{dom}(I') = \mathrm{dom}(I) \sqcup \mathcal{H}$, and, for any fact $R(\boldsymbol{a})$ of $I' \setminus I$ and $R^p \in \mathsf{Pos}(R)$, we have $a_p \in \mathsf{Wants}(P, R^p)$. ◁

Example 10.2.5. In Example 10.2.1, a pssinstance of I_0 is $P := (I_0, \{h\}, \lambda)$ where $\lambda(h) := \{S^2, T^1\}$. Further, it is balanced. For instance, $\mathsf{Wants}(P, S^1) = \{b\}$ and $\mathsf{Wants}(P, S^2) = \{h\}$. The instance I in Example 10.2.1 is a realization of P.

It is easy to see that realizations of pssinstances are weakly-sound:

Lemma 10.2.6 (Binary realizations are completions)**.** *If I' is a realization of a pssinstance of I_0 then it is a weakly-sound superinstance of I_0.*

Proof. Consider $a \in \mathrm{dom}(I')$ and $R^p \in \mathsf{Pos}(\sigma)$ such that $a \in \pi_{R^p}(I')$. As I' is a realization, we know that either $a \in \pi_{R^p}(I)$ or $a \in \mathsf{Wants}(P, R^p)$. By definition of $\mathsf{Wants}(P, R^p)$, and because $\mathcal{H} = \mathrm{dom}(I') \setminus \mathrm{dom}(I)$, this means that either $a \in \mathrm{dom}(I)$ and $a \in \pi_{R^p}(I) \sqcup \mathsf{Wants}(I, R^p)$, or $a \in \mathrm{dom}(I') \setminus \mathrm{dom}(I)$ and $R^p \in \lambda(a)$. Hence, let us check from the definition that I' is weakly-sound, which concludes:

- For any $a \in \mathrm{dom}(I)$ and $R^p \in \mathsf{Pos}(\sigma)$, we have established that $a \in \pi_{R^p}(I')$ implied that either $a \in \pi_{R^p}(I)$ or $a \in \mathsf{Wants}(I, R^p)$.
- For any $a \in \mathrm{dom}(I') \setminus \mathrm{dom}(I)$ and for any $R^p, S^q \in \mathsf{Pos}(\sigma)$, we have established that $a \in \pi_{R^p}(I')$ and $a \in \pi_{S^q}(I')$ implies that $R^p, S^q \in \lambda(a)$, so that $R^p \sim_{\mathsf{ID}} S^q$, hence $R^p = S^q$ or $R^p \subseteq S^q$ is in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$. □

10.3 Putting it Together

What remains to show to conclude the proof of Theorem 10.1 is that we can construct a *balanced* pssinstance of I_0, even when I_0 itself is not balanced. By a *balanced* pssinstance, we mean the exact analogue of Definition 10.1.1 for pssinstances:

Definition 10.3.1. A pssinstance $P = (I, \mathcal{H}, \lambda)$ is *balanced* if for every two positions R^p and R^q such that $R^p \to R^q$ and $R^q \to R^p$ are in Σ_{UFD}, we have $|\mathsf{Wants}(P, R^p)| = |\mathsf{Wants}(P, R^q)|$. ◁

If I_0 is balanced, the empty pssinstance $(I, \emptyset, \lambda)$, with λ the empty function, is a balanced pssinstance of I_0, and we could just complete I_0 as we presented before. We now show that, even if I_0 is not balanced, we can always construct a balanced pssinstance, thanks to the helpers:

Lemma 10.3.2 (Balancing)**.** *Any finite instance I satisfying Σ_{UFD} has a balanced pssinstance.*

In fact, this lemma does not use assumption **binary**. We will accordingly reuse it in the next chapter.

Proof. Let I be a finite instance. For any position R^p, define $o(R^p) := \mathsf{Wants}(I, R^p) \sqcup \pi_{R^p}(I)$, i.e., the elements that either appear at R^p or want to appear there. We show that $o(R^p) = o(S^q)$ whenever $R^p \sim_{\mathsf{ID}} S^q$, which is obvious if $R^p = S^q$, so assume $R^p \neq S^q$. First, we have $\pi_{R^p}(I) \subseteq o(S^q)$: elements in $\pi_{R^p}(I)$ want to appear at S^q unless they already do, and in both cases they are in $o(S^q)$. Second, elements of $\mathsf{Wants}(I, R^p)$ either occur at S^q, or at some other position T^r such that $T^r \subseteq R^p$ is a UID of $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$, so that by transitivity $T^r = S^q$ or $T^r \subseteq S^q$ also is, and so they want to be at S^q or they already are. Hence $o(R^p) \subseteq o(S^q)$; and symmetrically $o(S^q) \subseteq o(R^p)$. Thus, the set $o(R^p)$ only depends on the $\sim_{\mathsf{ID}}$-class of R^p.

Let $N := \max_{R^p \in \mathsf{Pos}(\sigma)} |o(R^p)|$, which is finite. We define for each $\sim_{\mathsf{ID}}$-class $[R^p]_{\mathsf{ID}}$ a set $p([R^p]_{\mathsf{ID}})$ of $N - |o(R^p)|$ fresh helpers. We let $\mathcal{H}$ be the disjoint union of the $p([R^p]_{\mathsf{ID}})$ for all classes $[R^p]_{\mathsf{ID}}$, and set λ to map the elements of $p([R^p]_{\mathsf{ID}})$ to $[R^p]_{\mathsf{ID}}$. We have thus defined a pssinstance $P = (I, \mathcal{H}, \lambda)$.

Let us now show that P is balanced. Consider now two positions R^p and R^q such that $\varphi : R^p \to R^q$ and $\varphi^{-1} : R^q \to R^p$ are in Σ_{UFD}, and show that $|\mathsf{Wants}(P, R^p)| = |\mathsf{Wants}(P, R^q)|$. We have $|\mathsf{Wants}(P, R^p)| = |\mathsf{Wants}(I, R^p)| + |p([R^p]_{\mathsf{ID}})| = |o(R^p)| - |\pi_{R^p}(I)| + N - |o(R^p)|$, which simplifies to $N - |\pi_{R^p}(I)|$. Similarly $|\mathsf{Wants}(P, R^q)| = N - |\pi_{R^q}(I)|$. Since $I \models \Sigma_{\mathrm{UFD}}$ and φ and φ^{-1} are in Σ_{UFD} we know that $|\pi_{R^p}(I)| = |\pi_{R^q}(I)|$. Hence, P is balanced, as we claimed. □

We had seen in Proposition 10.1.2 that we could construct a weakly-sound superinstance of a balanced I_0 by pairing together elements. We now generalize this claim to the balanced pssinstances that we constructed, showing that we can build realizations of balanced pssinstances that satisfy $\Sigma^{\mathrm{rev}}_{\mathrm{U}}$, using a similar technique:

Lemma 10.3.3 (Binary realizations)**.** *For any balanced pssinstance P of an instance I which satisfies Σ_{UFD}, we can construct a realization of P that satisfies $\Sigma_{\mathrm{U}}^{\mathrm{rev}}$.*

Proof. As in Proposition 10.1.2, for every relation R, construct a bijection f_R between $\mathsf{Wants}(P, R^1)$ and $\mathsf{Wants}(P, R^2)$: this is possible, as P is balanced. We then construct our realization I' as in Proposition 10.1.2: we add to I the fact $R(a, f_R(a))$ for every R of σ and every $a \in \mathsf{Wants}(P, R^1)$.

We prove that I' is a realization as in Proposition 10.1.2 by observing that whenever we create a fact $R(a, f_R(a))$, then we have $a \in \mathsf{Wants}(P, R^1)$ and $f_R(a) \in \mathsf{Wants}(P, R^2)$. Similarly, we show that $I' \models \Sigma_{\mathrm{UFD}}$ as in Proposition 10.1.2.

We now show that I' satisfies $\Sigma_{\mathrm{UID}}^{\mathrm{rev}}$. Assume to the contrary that there is an active fact $F = R(a_1, a_2)$ that witnesses the violation of a UID $\tau : R^p \subseteq S^q$, so that $a_p \in \mathsf{Wants}(I', R^p)$. If $a_p \in \mathrm{dom}(I)$, then the proof is exactly as for Proposition 10.1.2. Otherwise, if $a_p \in \mathcal{H}$, clearly by construction of f_R and I' we have $a_p \in \pi_{T^r}(I')$ iff $T^r \in \lambda(a_p)$. Hence, as $a_p \in \pi_{R^p}(I')$ and as τ witnesses by assumption $\mathsf{reversible}$ that $R^p \sim_{\mathsf{ID}} S^q$, we have $a_p \in \pi_{S^q}(I')$, contradicting the fact that $a_p \in \mathsf{Wants}(I', S^q)$. □

We now conclude the proof of Theorem 10.1. Given the instance I_0, construct a balanced pssinstance P with the Balancing Lemma, construct a realization I' of P that satisfies $\Sigma_{\mathrm{U}}^{\mathrm{rev}}$ with the Binary Realizations Lemma, and conclude by the "Binary Realizations are Completions" Lemma that I' is a weakly-sound superinstance of I_0.

Chapter 11

Weak Soundness on Arbitrary Arity Signatures

We now lift assumption binary and extend the results to arbitrary arity signatures:

Theorem 11.1. *Reversible UIDs and UFDs have finite weakly-universal models for ACQs.*

A first complication when lifting assumption binary is that realizations cannot be created just by pairing two elements. To satisfy the UIDs we may have to create facts that connect elements on more than two positions, so we may need more than the bijections between two positions that we used before. A much more serious problem is that the positions where we connect together elements may still be only a subset of the positions of the relation, which means that the other positions must be filled somehow.

We address these difficulties by defining first *piecewise realizations*, which create partial facts on positions connected by UFDs, similarly to the previous section. We show that we can get piecewise realizations by generalizing the Binary Realizations Lemma. Second, to find elements to reuse at other positions, we define a notion of *saturation*. We show that, by an initial *saturation process*, we can ensure that there are existing elements that we can reuse at positions where this will not violate UFDs (the *non-dangerous positions*). Third, we define a notion of *thrifty chase step* to solve UID violations one by one. We last explain how to use thrifty chase steps to solve all UID violations on saturated instances, using a piecewise realization as a template; this is how we construct our weakly-sound completion.

11.1 Piecewise Realizations

Without assumption binary, we must define a new equivalence relation to reflect the UFDs, in addition to $\sim_{\mathsf{ID}}$ which reflects the UIDs:

Definition 11.1.1. For any two positions R^p and R^q, we write $R^p \leftrightarrow_{\mathrm{FUN}} R^q$ whenever $R^p = R^q$ or $R^p \to R^q$ and $R^q \to R^p$ are both in Σ_{UFD}. $\lhd$

By transitivity of Σ_{UFD}, $\leftrightarrow_{\mathrm{FUN}}$ is indeed an equivalence relation.

The definition of *balanced instances* (Definition 10.1.1) generalizes as-is to arbitrary arity. We do not change the definition of *pssinstance* (Definition 10.2.3), and talk of

them being balanced in the same way. Further, we know that the Balancing Lemma (Lemma 10.3.2) holds even without assumption binary.

Our general scheme is the same: construct a balanced pssinstance of I_0, and use it to construct the completion I. What we need is to change the notion of *realization*. We replace it by *piecewise realizations*, which are defined on $\leftrightarrow_{\text{FUN}}$-classes. We number the $\leftrightarrow_{\text{FUN}}$-classes of $\mathsf{Pos}(\sigma)$ as $\Pi_1, \ldots, \Pi_n$ and define *piecewise instances* by their projections to the Π_i:

Definition 11.1.2. A *piecewise instance* is an n-tuple $PI = (K_1, \ldots, K_n)$, where each K_i is a set of $|\Pi_i|$-tuples, indexed by Π_i for convenience. The *domain* of PI is $\mathrm{dom}(PI) := \bigcup_i \mathrm{dom}(K_i)$. For $1 \leqslant i \leqslant n$ and $R^p \in \Pi_i$, we define $\pi_{R^p}(PI) := \pi_{R^p}(K_i)$. ◁

We will realize a pssinstance P, not as an instance as in the previous chapter, but as a piecewise instance. The tuples in each K_i will be defined from P, and will connect elements that want to occur at the corresponding position in Π_i, generalizing the ordered pairs constructed with bijections in the proof of the Binary Realizations Lemma. Let us define accordingly the notion of a *piecewise realization* of a pssinstance as a piecewise instance:

Definition 11.1.3. A piecewise instance $PI = (K_1, \ldots, K_n)$ is a *piecewise realization* of the pssinstance $P = (I, \mathcal{H}, \lambda)$ if:

- $\pi_{\Pi_i}(I) \subseteq K_i$ for all $1 \leqslant i \leqslant n$,
- $\mathrm{dom}(PI) = \mathrm{dom}(I) \sqcup \mathcal{H}$,
- for all $1 \leqslant i \leqslant n$, for all $R^p \in \Pi_i$, for every tuple $\boldsymbol{a} \in K_i \backslash \pi_{\Pi_i}(I)$, we have $a_p \in \mathsf{Wants}(P, R^p)$. ◁

Notice that the definition is similar to the conditions imposed on realizations (Definition 10.2.4), although piecewise realizations are piecewise instances, not actual instances; so we will need one extra step to make real instances out of them: this is done in Section 11.4.

We must now generalize the Binary Realizations Lemma (Lemma 10.3.3) to construct these piecewise realizations out of balanced pssinstances. For this, we need to define what it means for a piecewise instance PI to "satisfy" $\Sigma_{\text{U}}^{\text{rev}}$. For Σ_{UFD}, we require that PI respects the UFDs within each $\leftrightarrow_{\text{FUN}}$-class. For $\Sigma_{\text{UID}}^{\text{rev}}$, we define it directly from the projections of PI.

Definition 11.1.4. A piecewise instance PI is Σ_{UFD}-*compliant* if, for all $1 \leqslant i \leqslant n$, there are no two tuples $\boldsymbol{a} \neq \boldsymbol{b}$ in K_i such that $a_p = b_p$ for some $R^p \in \Pi_i$.

PI is $\Sigma_{\text{UID}}^{\text{rev}}$-*compliant* if $\mathsf{Wants}(PI, \tau) := \pi_{R^p}(PI) \backslash \pi_{S^q}(PI)$ is empty for all $\tau : R^p \subseteq S^q$ in $\Sigma_{\text{UID}}^{\text{rev}}$.

PI is $\Sigma_{\text{U}}^{\text{rev}}$-*compliant* if it is Σ_{UFD}- and $\Sigma_{\text{UID}}^{\text{rev}}$-compliant. ◁

We can then state and prove the generalization of the Binary Realizations Lemma:

Lemma 11.1.5 (Realizations)**.** *For any balanced pssinstance P of an instance I that satisfies Σ_{UFD}, we can construct a piecewise realization of P which is $\Sigma_{\text{U}}^{\text{rev}}$-compliant.*

Before we prove the Realizations Lemma, we show a simple example:

Example 11.1.6. Consider a 4-ary relation R and the UIDs $\tau : R^1 \subseteq R^2$, $\tau' : R^3 \subseteq R^4$ and their reverses, and the UFDs $\varphi : R^1 \to R^2$, $\varphi' : R^3 \to R^4$ and their reverses. We have $\Pi_1 = \{R^1, R^2\}$ and $\Pi_2 = \{R^3, R^4\}$. Consider $I_0 := \{R(a, b, c, d)\}$, which is balanced, and the trivial balanced pssinstance $P := (I_0, \emptyset, \lambda)$, where λ is the empty function. A $\Sigma_{\mathrm{U}}^{\mathrm{rev}}$-compliant piecewise realization of P is $PI := (\{(a, b), (b, a)\}, \{(c, d), (d, c)\})$.

We conclude the section with the proof of the Realizations Lemma:

Proof of Lemma 11.1.5. Let $P = (I, \mathcal{H}, \lambda)$ be the balanced pssinstance. Recall that the $\leftrightarrow_{\mathrm{FUN}}$-classes of σ are numbered $\Pi_1, \ldots, \Pi_n$. By definition of P being balanced (Definition 10.1.1 applied to arbitrary arity), for any $\leftrightarrow_{\mathrm{FUN}}$-class Π_i, for any two positions $R^p, R^q \in \Pi_i$, we have $|\mathsf{Wants}(P, R^p)| = |\mathsf{Wants}(P, R^q)|$. Hence, for all $1 \leqslant i \leqslant n$, we can define s_i as the value of $|\mathsf{Wants}(P, R^p)|$ for any $R^p \in \Pi_i$.

For $1 \leqslant i \leqslant n$, we let m_i be $|\Pi_i|$, and number the positions of Π_i as $R^{p^i_1}, \ldots, R^{p^i_{m_i}}$. We choose for each $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m_i$ an arbitrary bijection φ^i_j from $\{1, \ldots, s_i\}$ to $\mathsf{Wants}(P, R^{p^i_j})$. We construct the piecewise realization $PI = (K_1, \ldots, K_n)$ by setting each K_i for $1 \leqslant i \leqslant n$ to be $\pi_{\Pi_i}(I)$ plus the tuples $(\varphi^i_1(l), \ldots, \varphi^i_{m_i}(l))$ for $1 \leqslant l \leqslant s_i$.

It is clear that PI is a piecewise realization. Indeed, the first two conditions are immediate. Further, whenever we create a tuple $\boldsymbol{a} \in \Pi_i$ for any $1 \leqslant i \leqslant n$, then, for any $R^p \in \Pi_i$, we have $a_p \in \mathsf{Wants}(P, R^p)$.

Let us then show that PI is Σ_{UFD}-compliant. Assume by contradiction that there is $1 \leqslant i \leqslant n$ and $\boldsymbol{a}, \boldsymbol{b} \in K_i$ such that $a_l = b_l$ but $a_r \neq b_r$ for some $R^l, R^r \in \Pi_i$. As I satisfies Σ_{UFD}, we assume without loss of generality that $\boldsymbol{a} \in K_i \backslash \pi_{\Pi_i}(I)$. Now either $\boldsymbol{b} \in \pi_{\Pi_i}(I)$ or $\boldsymbol{b} \in K_i \backslash \pi_{\Pi_i}(I)$.

- If $\boldsymbol{b} \in \pi_{\Pi_i}(I)$, then $b_l \in \pi_{R^l}(I)$. Yet, we know by construction that, as $\boldsymbol{a} \in K_i \backslash \pi_{\Pi_i}(I)$, we have $a_l \in \mathsf{Wants}(P, R^l)$, so that by definition of $\mathsf{Wants}(P, R^l)$ we have $a_l \in \mathsf{Wants}(I, R^l)$. But we have $a_l = b_l$, so we have a contradiction.

- If $\boldsymbol{b} \in K_i \backslash \pi_{\Pi_i}(I)$, then, writing $R^l = R^{p^i_j}$ and $R^r = R^{p^i_{j'}}$, the fact that $a_l = b_l$ but $a_r \neq b_r$ contradicts the fact that $\varphi^i_j \circ (\varphi^i_{j'})^{-1}$ is injective.

Hence, PI is Σ_{UFD}-compliant.

Let us now show that PI is $\Sigma_{\mathrm{UID}}^{\mathrm{rev}}$-compliant. We must show that, for every UID $\tau : R^p \subseteq S^q$ of $\Sigma_{\mathrm{UID}}^{\mathrm{rev}}$, we have $\mathsf{Wants}(PI, \tau) = \emptyset$, which means that we have $\pi_{R^p}(PI) \subseteq \pi_{S^q}(PI)$. Let Π_i be the $\leftrightarrow_{\mathrm{FUN}}$-class of R^p, and assume to the contrary the existence of a tuple $\boldsymbol{a}$ of K_i such that $a_p \notin \pi_{S^q}(PI)$. Either we have $a_p \in \mathrm{dom}(I)$, or we have $a_p \in \mathcal{H}$.

- If $a_p \in \mathrm{dom}(I)$, as $a_p \notin \pi_{S^q}(PI)$, in particular $a_p \notin \pi_{S^q}(I)$, and as $a_p \in \pi_{R^p}(I)$, τ witnesses that $a_p \in \mathsf{Wants}(I, S^q)$. By construction of PI, then, letting $\Pi_{i'}$ be the $\leftrightarrow_{\mathrm{FUN}}$-class of S^q and letting $S^q = S^{p^{i'}_j}$, as $\varphi^{i'}_j$ is surjective, we must have $a_p \in \pi_{S^q}(K_{i'})$, that is, $a_p \in \pi_{S^q}(PI)$, a contradiction.

- If $a_p \in \mathcal{H}$, clearly by construction we have $a_p \in \pi_{T^r}(PI)$ iff $T^r \in \lambda(a_p)$, so that, given that τ witnesses $R^p \sim_{\mathsf{ID}} S^q$, if $a_p \in \pi_{R^p}(PI)$ then $a_p \in \pi_{S^q}(PI)$, a contradiction.

We conclude that PI is indeed a $\Sigma_{\mathrm{U}}^{\mathrm{rev}}$-compliant piecewise realization of P. □

11.2 Relation-Saturation

The Realizations Lemma gives us a $\Sigma_{\mathrm{U}}^{\mathrm{rev}}$-compliant piecewise realization which is a piecewise instance. To construct an actual superinstance from it, we will have to expand each tuple $\boldsymbol{t}$ of each K_i, defined on the $\leftrightarrow_{\mathrm{FUN}}$-class Π_i, to an entire fact $F_{\boldsymbol{t}}$ of the corresponding relation.

However, to fill the other positions of $F_{\boldsymbol{t}}$, we will need to reuse existing elements of I_0. To do this, it is easier to assume that I_0 contains some R-fact for every relation R of the signature.

Definition 11.2.1. A superinstance I of I_0 is *relation-saturated* if for every $R \in \sigma$ there is an R-fact in I. ◁

We illustrate why it is easier to work with relation-saturated instances:

Example 11.2.2. Suppose our schema has two binary relations R and T and a unary relation S, the UIDs $\tau : S^1 \subseteq R^1$, $\tau' : R^2 \subseteq T^1$ and their reverses, and no UFDs. Consider the non-relation-saturated instance $I_0 := \{S(a)\}$. It is balanced, so $P := (I_0, \emptyset, \lambda)$, with λ the empty function, is a pssinstance of I.

Now, a $\Sigma_{\mathrm{U}}^{\mathrm{rev}}$-compliant piecewise realization of P is $PI = (K_1, \ldots, K_5)$ with $K_2 = K_4 = K_5 = \emptyset$ and $K_1 = K_3 = \{a\}$, where Π_1 and Π_3 are the $\leftrightarrow_{\mathrm{FUN}}$-classes of R^1 and S^1. However, we cannot easily complete PI to an actual superinstance of I_0 satisfying τ and τ'. Indeed, to create the fact $R(a, \bullet)$, as indicated by K_1, we need to fill position R^2. Using an existing element would violate weak-soundness, and using a fresh element would introduce a violation of τ', which P and PI would not tell us how to solve.

Consider instead the relation-saturated instance $I_1 := I_0 \sqcup \{S(c), R(c, d), T(d, e)\}$. We can complete I_1 to a weakly-sound superinstance that satisfies τ and τ', by adding the fact $R(a, d)$. Observe how we reused d to fill position R^2: this does not violate weak-soundness or introduce new UID violations.

Relation-saturation can clearly be ensured by initial chasing, which does not violate weak-soundness. We call this a *saturation process* to ensure relation-saturation:

Lemma 11.2.3 (Relation-saturated solutions)**.** *For any reversible UIDs $\Sigma_{\mathrm{UID}}^{\mathrm{rev}}$, UFDs Σ_{UFD}, and instance I_0 satisfying Σ_{UFD}, the result of performing sufficiently many chase rounds on I_0 by $\Sigma_{\mathrm{UID}}^{\mathrm{rev}}$ is a weakly-sound relation-saturated superinstance of I_0 that satisfies Σ_{UFD}.*

This allows us to assume that I_0 was preprocessed with initial chasing if needed, so we can assume it to be relation-saturated. To show the lemma, and also for further use, we make a simple observation on weak-soundness:

Lemma 11.2.4 (Weak-soundness transitivity)**.** *If I' is a weakly-sound superinstance of I, and I is a weakly-sound superinstance of I_0, then I' is a weakly-sound superinstance of I_0.*

Proof. Let $a \in \mathrm{dom}(I')$, and let us show that it does not witness a violation of the weak-soundness of I' for I_0. We distinguish three cases:

- If $a \in \mathrm{dom}(I_0)$, then in particular $a \in \mathrm{dom}(I)$. Hence, letting S^q be any position such that $a \in \pi_{S^q}(I')$, as I' is a weakly-sound superinstance of I, either $a \in \pi_{S^q}(I)$ or we have $a \in \mathsf{Wants}(I, S^q)$. Let R^p be a position such that $a \in \pi_{R^p}(I)$, and such that $R^p = S^q$ (in the first case) or $R^p \subseteq S^q$ holds in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ (in the second case). As I is a weakly-sound superinstance of I_0, either $a \in \pi_{R^p}(I_0)$ or $a \in \mathsf{Wants}(I_0, R^p)$. As $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ is transitively closed, we conclude that $a \in \mathsf{Wants}(I_0, S^q)$ or $a \in \pi_{S^q}(I_0)$. Hence, the fact that a occurs at position S^q in I' does not cause a violation of weak-soundness in I' for I_0.

- If $a \in \mathrm{dom}(I) \setminus \mathrm{dom}(I_0)$, we must show that for any two positions R^p, S^q where a occurs in I', we have $R^p \sim_{\mathsf{ID}} S^q$. Let us fix two such positions, i.e., we have $a \in \pi_{R^p}(I')$ and $a \in \pi_{S^q}(I')$. As I' is a weakly-sound superinstance of I, we have either $a \in \pi_{R^p}(I)$ or $a \in \mathsf{Wants}(I, R^p)$, and we have either $a \in \pi_{S^q}(I)$ or $a \in \mathsf{Wants}(I, S^q)$. As in the previous case, let T^v and U^w be positions such that $a \in \pi_{T^v}(I)$ and $a \in \pi_{U^w}(I)$, and $T^v = R^p$ or the UID $\tau : T^v \subseteq R^p$ holds in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$, and $U^w = S^q$ or the UID $\tau' : U^w \subseteq S^q$ holds in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$. As I is a weakly-sound superinstance of I_0, and $a \notin \mathrm{dom}(I_0)$, we know that $T^v \sim_{\mathsf{ID}} U^w$. By assumption $\mathsf{reversible}$ and as $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ is transitively closed, we deduce (using τ and τ' if necessary) that $R^p \sim_{\mathsf{ID}} S^q$, which is what we wanted to show. Hence, the fact that a occurs at positions R^p and S^q in I' does not cause a violation of weak-soundness in I' for I_0.

- If $a \in \mathrm{dom}(I') \setminus \mathrm{dom}(I)$, then from the fact that I' is a weakly-sound superinstance of I, we deduce immediately about a what is needed to show that it does not witness a violation of the weak-soundness of I' for I_0.

So we conclude that I' is a weakly-sound instance of I_0, as desired. □

We conclude the section by proving the Relation-Saturated Solutions Lemma:

Proof of Lemma 11.2.3. Remember that the signature σ was assumed without loss of generality not to contain any useless relation. Hence, for every relation $R \in \sigma$, there is an R-fact in $\mathsf{Chase}(I_0, \Sigma^{\mathrm{rev}}_{\mathrm{UID}})$, which was generated at the n_R-th round of the chase, for some $n_R \in \mathbb{N}$. Let $n := \max_{R \in \sigma} n_R$, which is finite because the number of relations in σ is finite. We take I to be the result of applying n chase rounds to I_0.

It is clear that I is relation-saturated. The fact that I is weakly-sound is by the Weak-Soundness Transitivity Lemma, because each chase step clearly preserves weak-soundness: the exported element occurs at a position where it wants to occur, so we can use assumption $\mathsf{reversible}$, and new elements only occur at one position. □

11.3 Thrifty Chase Steps

We have explained why I_0 can be assumed to be relation-saturated, and we know we can build a $\Sigma^{\mathrm{rev}}_{\mathrm{U}}$-compliant piecewise realization PI of a balanced pssinstance. Our goal is now to satisfy the UIDs using PI. We will do so by a *completion process* that fixes each violation one by one, following PI. This section presents the tool that we use for this, and the next section describes the actual process.

Our tool is a form of chase step, a *thrifty chase step*, which adds a new fact F_{n} to satisfy a UID violation. For some of the positions, the elements of F_{n} will be defined from the realization PI, using one of its tuples. For each of these elements, either

F_n makes them occur at a position that they want to be (thus satisfying another violation) or these elements are helpers that did not occur already in the domain. At any other position S^r of F_n, we may either reuse an existing element (by relation saturation, one can always reuse an element that already occurs in that position) or create a fresh element (arguing that no UID will be violated on that element). This depends on whether S^r is *dangerous* or *non-dangerous*:

Definition 11.3.1. We say a position $S^r \in \mathsf{Pos}(\sigma)$ is *dangerous* for the position $S^q \neq S^r$ if $S^r \to S^q$ is in Σ_UFD, and write $S^r \in \mathsf{Dng}(S^q)$. Otherwise, still assuming $S^q \neq S^r$ S^r is *non-dangerous* for S^q, written $S^r \in \mathsf{NDng}(S^q)$. Note that $\{S^q\} \sqcup \mathsf{Dng}(S^q) \sqcup \mathsf{NDng}(S^q) = \mathsf{Pos}(S)$. ◁

We can now define *thrifty chase steps*. The details of the definition are designed for the completion process defined in the next section (Proposition 11.4.1), and for the specialized notions that we will introduce later in this section as well as in the following chapters.

Definition 11.3.2. Let I be a superinstance of I_0, let $\tau : R^p \subseteq S^q$ be a UID of $\Sigma^\mathrm{rev}_\mathrm{UID}$, and let $F_\mathrm{a} = R(\boldsymbol{a})$ be an active fact for τ in I. We call S^q the *exported position*, and write Π_i for its $\leftrightarrow_\mathrm{FUN}$-class.

Applying a *thrifty chase step* to F_a (or a) in I by τ yields a superinstance I' of I_0 which is I plus a single new fact $F_\mathrm{n} = S(\boldsymbol{b})$. We require the following on b_r for all $S^r \in \mathsf{Pos}(S)$:

- For $S^r = S^q$, we require $b_q = a_p$ and $b_q \in \mathsf{Wants}(I, \tau)$;
- For $S^r \in \Pi_i \backslash \{S^q\}$, we require that one of the following holds:
  - $b_r \in \mathsf{Wants}(I, S^r)$;
  - $b_r \notin \mathrm{dom}(I)$ and for all $S^s \in \Pi_i$, such that $b_r = b_s$, we have $S^r \sim_\mathsf{ID} S^s$;
- For $S^r \in \mathsf{Dng}(S^q) \backslash \Pi_i$, we require b_r to be fresh and occur only at that position;
- For $S^r \in \mathsf{NDng}(S^q)$, we require that $b_r \in \pi_{S^r}(I)$. ◁

Thrifty chase steps eliminate UID violations on the element at the exported position S^q of the new fact (which is why we call them "chase steps"), and also eliminate violations on positions in the same $\leftrightarrow_\mathrm{FUN}$-class as S^q, unless a fresh element is used there. The completion process that we will define in the next section will *only* apply thrifty chase steps (namely, *relation-thrifty steps*, which we will define shortly), and indeed this will be true of *all* completion processes used in this part of my thesis.

For now, we can observe that thrifty chase steps cannot break weak-soundness:

Lemma 11.3.3 (Thrifty preserves weak-soundness)**.** *For any weakly-sound superinstance I of an instance I_0, letting I' be the result of applying a thrifty chase step on I, I' is a weakly-sound superinstance of I_0.*

Proof. By the Weak-Soundness Transitivity Lemma, it suffices to show that I' is a weakly-sound superinstance of I. It suffices to check this for the elements occurring in the one fact $F_\mathrm{n} = S(\boldsymbol{b})$ of $I' \backslash I$, as the other elements occur at the same positions as before. Let us show for each b_r for $S^r \in \mathsf{Pos}(S)$ that b_r does not cause a violation of weak-soundness:

- For $S^r = S^q$, we have $b_r \in \mathsf{Wants}(I, S^r)$, so b_r does not violate weak-soundness;
- For $S^r \in \Pi_i \backslash \{S^q\}$, there are two possible cases:
  - $b_r \in \mathsf{Wants}(I, S^r)$, so b_r does not violate weak-soundness;
  - $b_r \notin \mathrm{dom}(I)$ and b_r occurs only at positions related by $\sim_{\mathsf{ID}}$, so b_r does not violate weak-soundness;
- For $S^r \in \mathsf{Dng}(S^q) \backslash \Pi_i$, b_r is fresh and occurs at a single position in I', so b_r does not violate weak-soundness;
- For $S^r \in \mathsf{NDng}(S^q)$, as $b_r \in \pi_{S^r}(I)$, b_r does not violate weak-soundness. □

Thrifty chase steps may introduce UFD violations. For this reason, we introduce the special case of *relation-thrifty* chase steps, which can not introduce such violations. (Relation-thrifty chase steps may still introduce FD violations; we will deal with this in Chapter 14.)

Definition 11.3.4 (Relation-thrifty)**.** A *relation-thrifty chase step* is a thrifty chase step where we choose one fact $F_\mathrm{r} = S(\boldsymbol{c})$ of I, and use F_r to define $b_r := c_r$ for all $S^r \in \mathsf{NDng}(S^q)$. ◁

Remember that relation-saturation ensures that such a fact $S(\boldsymbol{c})$ can always be found, so clearly any UID violation can be solved on a relation-saturated instance by applying some relation-thrifty chase step. Further, we can show that relation-thrifty chase steps, unlike thrifty chase steps, preserve UFDs:

Lemma 11.3.5 (Relation-thrifty preservation)**.** *For any superinstance I of an instance I_0 such that I satisfies Σ_{UFD}, letting I' be the result of applying a relation-thrifty chase step on I, then I' satisfies Σ_{UFD}. Further, if I is relation-saturated, then I' is relation-saturated.*

Proof. Assume to the contrary the existence of two facts $F = S(\boldsymbol{a})$ and $F' = S(\boldsymbol{b})$ in I' that witness a violation of some UFD $\varphi : S^r \to S^p$ of Σ_{UFD}. As $I \models \Sigma_{\mathrm{UFD}}$, we may assume without loss of generality that F' is $F_\mathrm{n} = S(\boldsymbol{b})$, the unique fact of $I' \backslash I$. Write $\tau : R^p \subseteq S^q$ the UID of $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ applied in the relation-thrifty chase step.

We first note that we must have S^r in $\mathsf{NDng}(S^q)$. Indeed, assuming to the contrary that $S^r = S^q$ or $S^r \in \mathsf{Dng}(S^q)$, the definition of thrifty chase steps requires that either $b_r \notin \mathrm{dom}(I)$ or $b_r \in \mathsf{Wants}(I, S^r)$, so that in either case $b_r \notin \pi_{S^r}(I)$. Yet, as $a_r = b_r$, F witnesses that $b_r \in \pi_{S^r}(I)$, a contradiction. Thus, $S^r \in \mathsf{NDng}(S^q)$.

Now, because φ holds in Σ_{UFD} and Σ_{UFD} is closed under the transitivity rule, unwinding the definitions we can see that $S^p \in \mathsf{NDng}(S^q)$ as well. Now, let $F_\mathrm{r} = S(\boldsymbol{c})$ be the chosen fact for the relation-thrifty chase step. Observe that we must have $F \neq F_\mathrm{r}$: this follows because we have $\pi_{S^r}(F_\mathrm{r}) = c_q = b_q$ but $\pi_{S^r}(F) = a_q$ and $a_q \neq b_q$ by definition of a UFD violation. Hence, as $b_q = c_q$ and $b_r = c_r$ by definition of F_n from F_r, as $F \neq F_\mathrm{r}$, F and F_r are also a violation of φ, which is in I, contradicting that $I \models \Sigma_{\mathrm{UFD}}$.

The second part of the claim is immediate. □

To summarize: we have defined the general tool used in our completion process, *thrifty chase steps*, along with a special case that preserves UFDs, *relation-thrifty chase steps*, which applies to relation-saturated instances. We now move to the last part of this chapter, where we use this tool to satisfy UID violations, also using the tools previously defined in this chapter.

11.4 Relation-Thrifty Completions

To prove Theorem 11.1, let us start by taking our initial finite instance I_0, which satisfies Σ_{UFD}, and use the Relation-Saturated Solutions Lemma to obtain a finite weakly-sound superinstance I_0' which is relation-saturated and still satisfies Σ_{UFD}. We now obtain our weakly-sound superinstance from I_0' by performing a *completion process* by relation-thrifty chase steps, which we phrase as follows:

Proposition 11.4.1 (Reversible relation-thrifty completion)**.** *For any reversible Σ_{UFD} and $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$, for any finite relation-saturated instance I_0' that satisfies Σ_{UFD}, we can use relation-thrifty chase steps to construct a finite weakly-sound superinstance I_{f} of I_0' that satisfies $\Sigma^{\mathrm{rev}}_{\mathrm{U}} = \Sigma^{\mathrm{rev}}_{\mathrm{UID}} \cup \Sigma_{\mathrm{UFD}}$.*

Indeed, once this result is proven, we can immediately conclude the proof of Theorem 11.1 with it, by applying it to I_0' and obtaining I_{f} which is a weakly-sound superinstance of I_0', hence of I_0 by the Weak-Soundness Transitivity Lemma. So we conclude the chapter with the proof of this proposition.

Recall that we number $\Pi_1, \ldots, \Pi_n$ the $\leftrightarrow_{\mathrm{FUN}}$-classes of $\mathsf{Pos}(\sigma)$. Let us write $\Pi_i \to \Pi_j$ to mean that all corresponding UFDs hold in Σ_{UFD} for positions in Π_i and Π_j. That is, equivalently, if one of them does (by definition of a $\leftrightarrow_{\mathrm{FUN}}$-class and the fact that Σ_{UFD} is transitively closed). We first define the *inner* classes, where creating elements may cause UID violations, and the *outer* classes, where this cannot happen because no position of the class occurs in any UID:

Definition 11.4.2. We say that Π_j is an *inner* $\leftrightarrow_{\mathrm{FUN}}$-class if it contains a position occurring in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$; otherwise, it is an *outer* $\leftrightarrow_{\mathrm{FUN}}$-class. ◁

The fundamental property is:

Lemma 11.4.3. *For any $1 \leqslant i, j \leqslant n$ with $i \neq j$, if Π_i is inner and $\Pi_j \to \Pi_i$ then Π_j is outer.*

Proof. Assume to the contrary that Π_j is inner. This means that it contains a position R^q that occurs in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$. As Π_i is inner, pick any $R^p \in \Pi_i$ that occurs in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$. As $\Pi_j \to \Pi_i$, $\varphi : R^q \to R^p$ holds in Σ_{UFD}. Hence, by assumption $\mathsf{reversible}$, φ^{-1} also does. But then we have $R^p \leftrightarrow_{\mathrm{FUN}} R^q$, contradicting the maximality of $\leftrightarrow_{\mathrm{FUN}}$-classes Π_i and Π_j. □

Let us now start the actual proof of Proposition 11.4.1, and fix the finite relation-saturated instance I_0' that satisfies Σ_{UFD}. We start by constructing a balanced pssinstance P of I_0' using the Balancing Lemma (Lemma 10.3.2), and a finite $\Sigma^{\mathrm{rev}}_{\mathrm{U}}$-compliant piecewise realization $PI = (K_1, \ldots, K_n)$ of P by the Realizations Lemma (Lemma 11.1.5). Let $\mathcal{F}$ be an infinite set of fresh elements (not in $\mathrm{dom}(P)$) from which we will take the (finitely many) fresh elements that we will introduce (only at dangerous positions, in outer classes) during the relation-thrifty chase steps.

We will use PI to construct a weakly-sound superinstance I_{f} by relation-thrifty chase steps. We maintain the following invariant when doing so:

Definition 11.4.4. A superinstance I of the instance I_0' *follows* the piecewise realization $PI = (K_1, \ldots, K_n)$ if for every inner $\leftrightarrow_{\mathrm{FUN}}$-class Π_i, we have $\pi_{\Pi_i}(I) \subseteq K_i$. ◁

We prove the Reversible Relation-Thrifty Completion Proposition by satisfying UID violations in I'_0 with relation-thrifty chase steps using the piecewise realization PI. We call I the current state of our superinstance, starting at $I := I'_0$, and we perform relation-thrifty chase steps on I to satisfy UID violations, until we reach a finite weakly-sound superinstance I_f of I'_0 such that I_f satisfies $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ and I_f follows PI. This I_f will be the final result of the Reversible Relation-Thrifty Completion Proposition.

Chasing by relation-thrifty chase steps preserves the following invariants:

sub: $I'_0 \subseteq I$ (this is clearly monotone);

wsnd: I is weakly-sound (by Lemma 11.3.3);

fun: $I \models \Sigma_{\mathrm{UFD}}$ (by Lemma 11.3.5);

rsat: I is relation-saturated (by Lemma 11.3.5).

Further, we maintain the following invariants:

fw: I follows PI;

help: For any position R^p of an outer class, $\pi_{R^p}(I)$ and $\mathcal{H}$ are disjoint.

Let us show that any UID violation in I at any stage of the construction can be solved by applying a relation-thrifty chase step that preserves these invariants. To show this, let $a \in \mathsf{Wants}(I, \tau)$ be an element to which some UID $\tau : R^p \subseteq S^q$ of $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ is applicable. Let $F_\mathrm{a} = R(\boldsymbol{a})$ be the active fact, with $a = a_p$. Let Π_i denote the $\leftrightarrow_{\mathrm{FUN}}$-classes of R^p and S^q respectively. The UID τ witnesses that Π_i is inner, so by invariant fw we have $a \in \pi_{R^p}(PI)$. As PI is $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$-compliant, we must have $a \in \pi_{S^q}(PI)$, and there is a $|\Pi_i|$-tuple $\boldsymbol{t} \in K_i$ such that $t_q = a$; in fact, by Σ_{UFD}-compliance, there is exactly one such tuple.

Let $F_\mathrm{r} = S(\boldsymbol{c})$ be an S-fact of I'_0, which is possible by invariant rsat. We create a new fact $F_\mathrm{n} = S(\boldsymbol{b})$ with the relation-thrifty chase step defined as follows:

- For the exported position S^q, we set $b_q := a_p$.
- For any $S^r \in \Pi_i$, we set $b_r := t_r$.
- For any position $S^r \in \mathsf{Dng}(S^q) \backslash \Pi_i$, we take b_r to be a fresh element f_r from $\mathcal{F}$.
- For any position $S^r \in \mathsf{NDng}(S^q)$, we set $b_r := c_r$.

We first verify that this satisfies the conditions of thrifty chase steps. We have set $b_q = a$, and by definition of F_r it is immediate that $b_r \in \pi_{S^r}(I)$ for $S^r \in \mathsf{NDng}(S^q)$. For $S^r \in \mathsf{Dng}(S^q) \backslash \Pi_i$, we use a fresh element f_r from $\mathcal{F}$ which occurs only at position S^r, as we should.

The last case to check is for $S^r \in \Pi_i \backslash \{S^q\}$. The first case is if $b_r \notin \mathrm{dom}(I)$, in which case we must show that all positions at which b_r occurs are $\sim_{\mathsf{ID}}$-equivalent. Assume that b_r occurs at some other position $S^s \in \Pi_i$. Now as b_r is in $\pi_{S^s}(PI)$, by definition of PI being a piecewise realization of P, we have $b_r \in \mathsf{Wants}(P, S^s)$. Now, as $b_r \notin \mathrm{dom}(I)$, by invariant sub we also have $b_r \notin \mathrm{dom}(I'_0)$. But as $b_r \in \mathrm{dom}(PI)$, we must have $b_r \in \mathcal{H}$. So by definition of a pssinstance we have $S^s \in \lambda(b_r)$. As $b_r \in \mathsf{Wants}(P, S^r)$ also, we have $S^r \in \lambda(b_r)$. By definition of $\lambda(b_r)$ being an $\sim_{\mathsf{ID}}$-class, this means that $S^r \sim_{\mathsf{ID}} S^s$, as required.

The second case is $b_r \in \mathrm{dom}(I)$. We will show that we have $b_r \in \mathsf{Wants}(I, S^r)$. Observe first that $b_r \notin \pi_{S^r}(I)$. Indeed, assuming to the contrary that $b_r \in \pi_{S^r}(I)$, let $F = S(\boldsymbol{d})$ be a witnessing fact in I. As Π_i is inner, by invariant fw, we deduce that $\pi_{\Pi_i}(\boldsymbol{d}) \in \pi_{\Pi_i}(PI)$. Now, as $d_r = t_r$ and PI is Σ_{UFD}-compliant, we deduce that $\boldsymbol{d} = \boldsymbol{t}$, so that F witnesses that d_q is in $\pi_{S^q}(I)$. As we have $d_q = t_q = a$, this contradicts the applicability of τ to a. Hence, we have $b_r \notin \pi_{S^r}(I)$.

Second, observe that we have $t_r \in \mathsf{Wants}(P, S^r)$. Indeed, we have $b_r = t_r$ which is in $\pi_{S^r}(PI)$, and we cannot have $\boldsymbol{t} \in \pi_{\Pi_i}(I)$, as otherwise this would contradict the applicability of τ to a, as we showed; so in particular, by invariant sub, we cannot have $\boldsymbol{t} \in \pi_{\Pi_i}(I'_0)$. Thus, by definition of a piecewise realization, we have $t_r \in \mathsf{Wants}(P, S^r)$.

Now, as $t_r \in \mathsf{Wants}(P, S^r)$, by definition of $\mathsf{Wants}(P, S^r)$, there are two cases:

- We have $t_r \in \mathrm{dom}(I'_0)$ and $t_r \in \mathsf{Wants}(I'_0, S^r)$. In this case, as we have shown that $t_r \notin \pi_{S^r}(I)$, we conclude immediately that $\mathrm{t}_r \in \mathsf{Wants}(I, S^r)$.

- We have $t_r \in \mathcal{H}$ and $S^r \in \lambda(t_r)$. In this case, consider a fact F' of I witnessing $t_r \in \mathrm{dom}(I)$, where t_r occurs at a position T^l; let $\Pi_{i'}$ be the $\leftrightarrow_{\mathrm{FUN}}$-class of T^l. As $t_r \in \mathcal{H}$, by invariant help, $\Pi_{i'}$ is inner, so by invariant fw there is a tuple $\boldsymbol{t}'$ of $K_{i'}$ such that $t'_l = t_r$. Now, as $t_r \in \mathcal{H}$, by definition of piecewise realizations, we have $T^l \in \lambda(t_r)$. Hence, either the UID $\tau' : T^l \subseteq S^r$ is in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ or we have $T^l = S^r$. As $t_r \in \pi_{T^l}(I)$ and we have shown earlier that $t_r \notin \pi_{S^r}(I)$, we know that $T^l \neq S^r$, so τ' is in $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$. Hence, as F' witnesses that $t_r \in \pi_{T^l}(I)$, and as $t_r \notin \pi_{S^r}(I)$, we conclude that $t_r \in \mathsf{Wants}(I, S^r)$.

Hence, in either case we have $t_r \in \mathsf{Wants}(I, S^r)$, as claimed. This concludes the proof of the fact that we have indeed defined a thrifty chase step. Further, the step is clearly relation-thrifty by construction. The last thing to do is to check that invariants fw and help are preserved by the relation-thrifty chase step:

- For invariant fw, τ witnesses that the class Π_i of S^q is inner. Hence, for any $S^r \in \mathsf{Dng}(S^q) \backslash \Pi_i$, by Lemma 11.4.3, the $\leftrightarrow_{\mathrm{FUN}}$-class of S^r is outer. Thus, to show that fw is preserved, it suffices to show it for the $\leftrightarrow_{\mathrm{FUN}}$-class Π_i and on the $\leftrightarrow_{\mathrm{FUN}}$-classes included in $\mathsf{NDng}(S^q)$ (clearly no $\leftrightarrow_{\mathrm{FUN}}$-class includes both a position of $\mathsf{Dng}(S^q)$ and a position of $\mathsf{NDng}(S^q)$). For Π_i, the new fact F_{n} is defined following $\boldsymbol{t}$; for the classes in $\mathsf{NDng}(S^q)$, it is defined following an existing fact of I. Hence, invariant fw is preserved.

- Invariant help is preserved because the only new elements of F_{n} that may be in $\mathcal{H}$ are those used at positions of Π_i, which is inner.

Let I_{f} be the result of the process that we have described. It satisfies Σ_{UID} by definition, and it is a finite weakly-sound superinstance of I'_0 that satisfies Σ_{UID}, by invariants wsnd, sub, and fun. Further, it follows PI by invariant fw, and PI is finite. This implies that I_{f} is finite, because we apply chase steps by $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$, so each chase step makes an element of $\mathrm{dom}(PI)$ occur at a new position, so we only applied finitely many chase steps. This concludes the proof of the Reversible Relation-Thrifty Completion Proposition, and concludes the chapter.

Chapter 12

Ensuring k-Universality

We build on the constructions of the previous chapter to replace weak-soundness by k-soundness for acyclic queries in ACQ, for some $k > 0$ fixed in this chapter. That is, we aim to prove:

Theorem 12.1. *Reversible UIDs and UFDs have finite k-universal models for ACQs.*

We first introduce the concept of *aligned superinstances*, which give us an invariant that ensures k-soundness. We then give the *fact-saturation* process that generalizes relation-saturation, and a related notion of *fact-thrifty chase step*. We then define *essentiality*, which must additionally be ensured for us to be able to reuse the weakly-sound completions of the previous chapter. We conclude by the construction of a generalized completion process that uses these chase steps to repair UID violations in the instance while preserving k-soundness.

In this chapter, we still make assumption reversible on $\Sigma_{\text{UID}}^{\text{rev}}$ and Σ_{UFD}. However, we will also be considering a superset Σ_{UID} of $\Sigma_{\text{UID}}^{\text{rev}}$, which we assume to be transitively closed, but which may not satisfy assumption reversible. To prove Theorem 12.1, it suffices to define $\Sigma_{\text{UID}} := \Sigma_{\text{UID}}^{\text{rev}}$, so *the distinction can be safely ignored on first reading*. The reason for the distinction will become apparent in the next chapter.

12.1 Aligned Superinstances

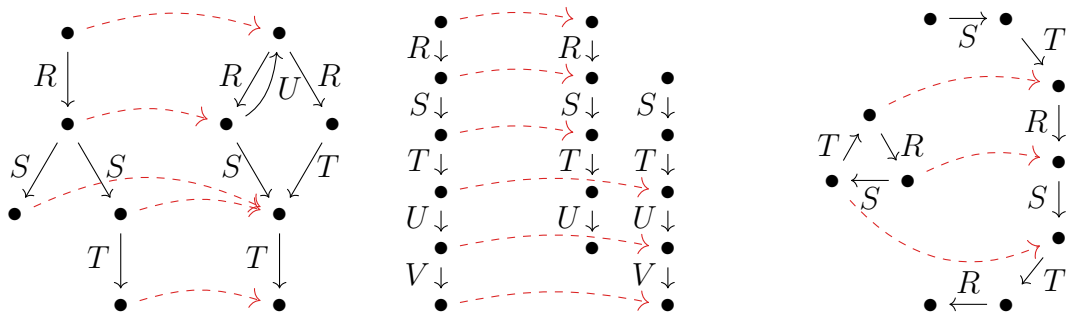
In this section, we only work with the superset Σ_{UID}, and we do not use assumption reversible. We ensure k-soundness relative to Σ_{UID} by maintaining a *k-bounded simulation* from our superinstance of I_0 to the chase $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$.

Definition 12.1.1. For I, I' two instances, $a \in \mathrm{dom}(I)$, $b \in \mathrm{dom}(I')$, and $n \in \mathbb{N}$, we write $(I, a) \leqslant_n (I', b)$ if, for any fact $R(\boldsymbol{a})$ of I with $a_p = a$ for some $R^p \in \mathsf{Pos}(R)$, there exists a fact $R(\boldsymbol{b})$ of I' such that $b_p = b$, and $(I, a_q) \leqslant_{n-1} (I', b_q)$ for all $R^q \in \mathsf{Pos}(R)$ (note that this is tautological for $R^q = R^p$). The base case $(I, a) \leqslant_0 (I', b)$ always holds.

An *n-bounded simulation* from I to I' is a mapping sim such that for all $a \in \mathrm{dom}(I)$, we have $(I, a) \leqslant_n (I', \mathsf{sim}(a))$.

We write $a \simeq_n b$ for $a, b \in \mathrm{dom}(I)$ if both $(I, a) \leqslant_n (I, b)$ and $(I, b) \leqslant_n (I, a)$; this is an equivalence relation on $\mathrm{dom}(I)$. ◁

Example 12.1.2. We illustrate in Figure 12.1 some examples of 2-bounded simulations from one instance to another, on a binary signature. For any element a in a



(a) Homomorphisms are bounded simulations (b) Bounded simulations do not preserve large ACQs (c) Bounded simulations do not preserve CQs

Figure 12.1: 2-bounded simulations (shown as dashed red lines): see Example 12.1.2

left instance I and image a' of a in the right instance I' by the 2-bounded simulation (represented by the dashed red arrows), we have $(I, a) \leqslant_2 (I', a')$. This means that, for any element b in I which is adjacent to a by some relation R, there must be an element b' in I' which is adjacent to a' by R and satisfies $(I, b) \leqslant_1 (I', b')$; however, note that b' need not be the image of b by the bounded simulation.

Figure 12.1a illustrates how a homomorphism is a special case of a 2-bounded simulation (indeed, it is an n-bounded simulation for any $n \in \mathbb{N}$).

Figure 12.1b illustrates how a 2-bounded simulation from I to I' does not guarantee that any ACQ satisfied by I is also true in I': for this example, consider the query $\exists xyzuvw\ R(x, y) \wedge S(y, z) \wedge T(z, u) \wedge U(u, v) \wedge V(v, w)$. However, we will soon see that n-bounded simulations preserve ACQ of size $\leqslant n$ (Lemma 12.1.3).

Figure 12.1c shows that a 2-bounded simulation does not preserve CQs that are not ACQs, as witnessed by $\exists xyz\ R(x, y) \wedge S(y, z) \wedge T(z, x)$.

The point of bounded simulations is that they preserve acyclic queries of size smaller than the bound:

Lemma 12.1.3 (ACQ preservation)**.** *For any instance I and ACQ q of size $\leqslant n$ such that $I \models q$, if there is an n-bounded simulation from I to I', then $I' \models q$.*

To show this lemma, we introduce a different way to write queries in ACQ. Consider the following alternate query language:

Definition 12.1.4. We inductively define a special kind of query with at most one free variable, a *pointed query*. The base case is that of a tautological query with no atoms. Inductively, pointed queries include all queries of the form:

$$q(x) : \bigwedge_i \left(\exists \boldsymbol{y^i} \left(A^i(x, \boldsymbol{y^i}) \wedge \bigwedge_{y^i_j \in \boldsymbol{y^i}} q^i_j(y^i_j) \right) \right)$$

where the $\boldsymbol{y^i}$ are vectors of pairwise distinct variables (also distinct from x), A^i are atoms with free variables as indicated and with no repeated variables (each free variable occurs at exactly one position), and the q^i_j are pointed queries.

The *size* $|q|$ of a pointed query q is the total number of atoms in q, including its subqueries. ◁

It is easily seen that, for any pointed query q', the query $q : \exists x\ q'(x)$ is an ACQ. Conversely, we can show:

Lemma 12.1.5. *For any (Boolean) ACQ q and variable x of q, we can rewrite q as $\exists x\ q'(x)$ with q' a pointed query such that $|q| = |q'|$.*

Proof. We show the claim by induction on the size of q. It is clearly true for the empty query.

Otherwise, let $\mathcal{A} = A_1, \ldots, A_m$ be the atoms of q where x occurs. Because q is an ACQ, x occurs exactly once in each of them, and each variable y occurring in one of the A_i occurs exactly once in them overall: y cannot occur twice in the same atom, nor can occur in two different atoms A_p and A_q (as in this case A_p, y, A_q, x would be a Berge cycle of q). Let $\mathcal{Y}$ be the set of the variables occurring in $\mathcal{A}$, not including x.

Consider the incidence multigraph G of q (Definition 9.2.1). Remember that we assume queries to be connected, so q is connected, and G is connected. Let $\mathcal{Z}$ be the variables of q which are not in $\mathcal{Y} \cup \{x\}$. For each $z \in \mathcal{Z}$, there must be a path p_z from x to z in G, written $x = w_1^z, \ldots, w_{n_z}^z = z$. Observe that, by definition of $\mathcal{Y}$, we must have $w_2^z \in \mathcal{Y}$ for any such path. Further, for each $z \in \mathcal{Z}$, we claim that there is a *single* $y_z \in \mathcal{Y}$ such that $w_2^z = y_z$ for any such path. Indeed, assuming to the contrary that there are $y_z \neq y_z'$ in $\mathcal{Y}$, a path p_z whose second element is y_z, and a path p_z' whose second element is y_z', we deduce from p_z and p_z' a Berge cycle in q.

Thus we can partition $\mathcal{Z}$ into sets of variables $\mathcal{Z}_y$ for $y \in \mathcal{Y}$, where $\mathcal{Z}_y$ contains all variables z of $\mathcal{Z}$ such that y is the variable used to reach z from x. Let $\mathcal{A}_y$ for $y \in \mathcal{Y}$ be the atoms of q whose variables are a subset of $\mathcal{Z}_y \cup \{y\}$. It is clear that $\mathcal{A}$ and the $\mathcal{A}_y$ are a partition of the atoms of q: no atom A can include a variable z from $\mathcal{Z}_y$ and a variable z' from $\mathcal{Z}_{y'}$ for $y \neq y'$ in $\mathcal{Y}$, as otherwise a path from x to z and a path from x to z', together with A, imply that q has a Berge cycle.

Now, we form for each $y \in \mathcal{Y}$ a query q_y as the set of atoms $\mathcal{Z}_y$, with all variables existentially quantified except for y. As the queries $\exists y\, q_y(y)$ are connected queries in ACQ which are strictly smaller than q, by induction we can rewrite q_y to a pointed query of the same size. Hence, we have shown that q can be rewritten as a pointed query built from the A_i and, for each i, the q_y for $y \in \mathcal{Y}$. □

We use this normal form to prove the ACQ Preservation Lemma:

Proof of Lemma 12.1.3. Fix the instances I and I', and the ACQ q. We show, by induction on $n \in \mathbb{N}$, the following claim: for any $n \in \mathbb{N}$, for any *pointed query* q such that $|q| \leqslant n$, for any $a \in \mathrm{dom}(I)$, if $I \models q(a)$, then for any $a' \in \mathrm{dom}(I')$ such that $(I, a) \leqslant_n (I', a')$, we have $I' \models q(a')$. Clearly this claim implies the statement of the Lemma, as by Lemma 12.1.5 any ACQ query can be written as $\exists x\ q(x)$ with q a pointed query. The case of the trivial query is immediate.

For the induction step, consider a pointed query $q(x)$ of size $n := |q|$, $n > 0$, written in the form of Definition 12.1.4, and fix $a \in \mathrm{dom}(I)$. Consider a match h of $q(a)$ on I, which must map x to a. Let $a' \in \mathrm{dom}(I')$ be such that $(I, a) \leqslant_n (I', a')$. We show that $I' \models q'(a)$.

Using notation from Definition 12.1.4, write $\boldsymbol{y}$ the (disjoint) union of the $\boldsymbol{y^i}$, write $\mathcal{A} = A^1, \ldots, A^n$, and write $q_j^i(y_j^i)$ the subqueries. Let $b_j^i := h(y_j^i)$ for all $y_j^i \in \boldsymbol{y}$. We show that there is a match $h_{\mathcal{A}}$ of $\mathcal{A}$ on I' that maps x to a' and such that every $y_j^i \in \boldsymbol{y}$ is mapped to some element $(b_j^i)'$ of I' such that $(I, b_j^i) \leqslant_{n-1} (I', (b_j^i)')$. Indeed,

start by fixing $h_{\mathcal{A}}(x) := a'$. Now, for each atom $A^i = R(x, \boldsymbol{y^i})$ of $\mathcal{A}$, x occurs at some position, say R^p, and $h(A^i) = R(\boldsymbol{b^i})$ is a fact of I where $h(x) = a$ occurs at position R^p. As each variable in $\boldsymbol{y^i}$ occurs at precisely one position of A^i, we index each of these variables by the one position in A^i where it occurs. Now, as $(I, a) \leqslant_n (I', a')$, there is a fact $(A^i)' = R((\boldsymbol{b^i})')$ of I' such that $(b^i_p)' = a'$ and, for all $1 \leqslant j \leqslant |R|$ with $p \neq j$, we have $(I, b^i_j) \leqslant_{n-1} (I', (b^i_j)')$. We define $h_{\mathcal{A}}(y^i_j) := (b^i_j)'$ for all i and j. As each variable of $\boldsymbol{y}$ occurs exactly once in $\mathcal{A}$ overall, there definitions cannot conflict, so this correctly defines a function $h_{\mathcal{A}}$ which is clearly indeed a match of $\mathcal{A}$ on I' with the claimed properties.

Now, each of the q^i_j is a pointed query which is strictly smaller than q. Further, the restriction of h to the variables of q^i_j is a match of q^i_j on I that maps each y^i_j (indexing the variables of y^i in the same way as before) to $b^i_j \in \mathrm{dom}(I)$. As we have $(I, b^i_j) \leqslant_{n-1} (I', (b^i_j)')$, then we can apply the induction hypothesis to show that each of the q^i_j has a match $h_{i,j}$ in I' that maps y^i_j to $(b^i_j)'$. As these queries have disjoint sets of variables, the range of the $h_{i,j}$ is disjoint, and the range of each $h_{i,j}$ overlaps with $h_{\mathcal{A}}$ only on $\{y^i_j\}$, where we have $h_{\mathcal{A}}(y^i_j) = h_{i,j}(y^i_j) = (b^i_j)'$. Thus, we can combine the $h_{i,j}$ and the previously defined $h_{\mathcal{A}}$ to obtain an overall match of q in I' that matches x to a'. This concludes the proof of the induction step, and proves our claim on pointed queries. □

This implies that any superinstance of I_0 that has a k-bounded simulation to $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$ must be k-sound for Σ_{U} (no matter whether it satisfies Σ_{U} or not). Indeed, the chase is a universal model for Σ_{UID}, and it satisfies Σ_{UFD} (by the Unique Witness Property, and because I_0 does). Hence, the chase is in particular k-universal for Σ_{U}. Hence, by the ACQ preservation lemma, any superinstance with a k-bounded simulation to the chase is k-sound.

We give a name to such superinstances. For convenience, we also require them to be finite and satisfy Σ_{UFD}. For technical reasons we require that the simulation is the identity on I_0, that it does not map other elements to I_0, and that elements occur in the superinstance at least at the position where their sim-image was introduced in the chase (the *directionality condition*):

Definition 12.1.6. An *aligned superinstance* $J = (I, \mathsf{sim})$ of I_0 (for Σ_{UFD} and Σ_{UID}) is a finite superinstance I of I_0 that satisfies Σ_{UFD}, and a k-bounded simulation sim from I to $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$ such that $\mathsf{sim}_{|I_0}$ is the identity and $\mathsf{sim}_{|(I \setminus I_0)}$ maps to $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}}) \setminus I_0$.

Further, for any $a \in \mathrm{dom}(I) \setminus \mathrm{dom}(I_0)$, letting R^p be the position where $\mathsf{sim}(a)$ was introduced in $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$, we require that $a \in \pi_{R^p}(I)$. We call this the *directionality condition*.

We write $\mathrm{dom}(J)$ to mean $\mathrm{dom}(I)$, and extend other existing notation in the same manner when relevant, e.g., $\mathsf{Wants}(J, \tau)$ means $\mathsf{Wants}(I, \tau)$. ◁

12.2 Fact-Saturation

Before we perform the *completion process* that allows us to satisfy the UIDs $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$, we need to perform a *saturation process*. Like aligned superinstances, this process is defined with respect to the superset Σ_{UID}, and does not depend on assumption $\mathsf{reversible}$. The process generalizes relation-saturation from the previous chapter:

instead of achieving all relations, we want the aligned superinstance to achieve all *fact classes*:

Definition 12.2.1. A *fact class* is a pair $(R^p, \boldsymbol{C})$ of a position $R^p \in \mathsf{Pos}(\sigma)$ and a $|R|$-tuple of $\simeq_k$-classes of elements of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$, with $\simeq_k$ as in Definition 12.1.1.

The *fact class* of a fact $F = R(\boldsymbol{a})$ of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})\backslash I_0$ is $(R^p, \boldsymbol{C})$, where a_p is the exported element of F and C_i is the $\simeq_k$-class of a_i in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ for all $R^i \in \mathsf{Pos}(R)$.

A fact class $(R^p, \boldsymbol{C})$ is *achieved* in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ if $\mathsf{NDng}(R^p) \neq \emptyset$ and if it is the fact class of some fact of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})\backslash I_0$. Such a fact is an *achiever* of the fact class. We write $\mathsf{AFactCl}$ for the set of all achieved fact classes.

For brevity, the dependence on I_0, Σ_{UID}, and k is omitted from this notation. ◁

The requirement that $\mathsf{NDng}(R^p)$ is non-empty is a technicality that will prove useful in Chapter 14. The following is easy to see:

Lemma 12.2.2. *For any initial instance I_0, set Σ_{UID} of UIDs, and $k \in \mathbb{N}$, AFactCl is finite.*

Proof. We first show that $\simeq_k$ has only a finite number of equivalence classes on $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$. Indeed, for any element $a \in \mathrm{dom}(\mathsf{Chase}(I_0, \Sigma_{\text{UID}}))$, by the Unique Witness Property, the number of facts in which a occurs is bounded by a constant depending only on I_0 and Σ_{UID}. Hence, there is a constant M depending only on I_0, Σ_{UID}, and k, such that, for any element $a \in \mathrm{dom}(\mathsf{Chase}(I_0, \Sigma_{\text{UID}}))$, the number of elements of $\mathrm{dom}(\mathsf{Chase}(I_0, \Sigma_{\text{UID}}))$ which are relevant to determine the $\simeq_k$-class of a (that is, the elements whose distance to d in the Gaifman graph of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ is $\leqslant k$) is bounded by M.

This clearly implies that $\mathsf{AFactCl}$ is finite, because the number of m-tuples of equivalence classes of $\simeq_k$ that occur in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ is then finite for any $m \leqslant \max_{R \in \sigma} |R|$, and $\mathsf{Pos}(\sigma)$ is finite. □

We define *fact-saturated* superinstances, which achieve all fact classes in $\mathsf{AFactCl}$:

Definition 12.2.3. An aligned superinstance $J = (I, \mathsf{sim})$ of I_0 is *fact-saturated* if, for any achieved fact class $D = (R^p, \boldsymbol{C})$ in $\mathsf{AFactCl}$, there is a fact $F_D = R(\boldsymbol{a})$ of $I\backslash I_0$ such that $\mathsf{sim}(a_i) \in C_i$ for all $R^i \in \mathsf{Pos}(R)$. We say that F_D *achieves* D in J.

Note that this definition does not depend on the position R^p of the fact class. ◁

The point of fact-saturation is that, when we perform thrifty chase steps, we can reuse elements from a suitable achiever at the non-dangerous positions. With relation-saturation, the facts were of the right relation; with fact-saturation, they further achieve the right *fact class*, which will be important to maintain the bounded simulation sim.

The fact-saturation completion process, which replaces the relation-saturation process of the previous chapter, works in the same way.

Lemma 12.2.4 (Fact-saturated solutions)**.** *For any UIDs Σ_{UID}, UFDs Σ_{UFD}, and instance I_0, the result I of performing sufficiently many chase rounds on I_0 is such that $J_0 = (I, \mathrm{id})$ is a fact-saturated aligned superinstance of I_0.*

Proof. For every $D \in \mathsf{AFactCl}$, let $n_D \in \mathbb{N}$ be such that D is achieved by a fact of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ created at round n_D. As $\mathsf{AFactCl}$ is finite, $n := \max_{D \in \mathsf{AFactCl}} n_D$ is finite. Hence, all classes of $\mathsf{AFactCl}$ are achieved after n chase rounds on I_0.

Consider now I'_0 obtained from the aligned superinstance I_0 by n rounds of the UID chase, and $J_0 = (I'_0, \mathrm{id})$. It is clear that for any $D \in \mathsf{AFactCl}$, considering an achiever of D in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$, the corresponding fact in J_0 is an achiever of D in J_0. Hence, J_0 is indeed fact-saturated. □

We thus obtain a fact-saturated aligned superinstance J_0 of our initial instance I_0, which we now want to complete to one that satisfies the UIDs we are interested in, namely $\Sigma^{\text{rev}}_{\text{UID}}$.

12.3 Fact-Thrifty Steps

In the previous chapter, we defined relation-thrifty chase steps, which reused non-dangerous elements from any fact of the correct relation, assuming relation-saturation. We now define *fact-thrifty* steps, which are thrifty steps that reuse elements from a fact achieving the right fact class, thanks to fact-saturation. To do so, however, we must first refine the notion of *thrifty* chase step, to make them apply to aligned superinstances. We will *always* apply them to aligned superinstances for Σ_{UID} and Σ_{UFD}; however, we will always chase by the UIDs of $\Sigma^{\text{rev}}_{\text{UID}}$.

Definition 12.3.1 (Thrifty chase steps)**.** Let $J = (I, \mathsf{sim})$ be an aligned superinstance of I_0 for Σ_{UID} and Σ_{UFD}, let $R^p \subseteq S^q$ be a UID of $\Sigma^{\text{rev}}_{\text{UID}}$, and let $a \in \mathsf{Wants}(J, \tau)$. The result of applying a *thrifty* chase step to a in J by τ is a pair (I', sim') where:

- The instance I' is the result of applying some thrifty step to a in I by τ, as in Definition 11.3.2 (note that this *only* depends on $\Sigma^{\text{rev}}_{\text{UID}}$ and Σ_{UFD}, *not* on Σ_{UID}).
- The mapping sim' extends sim to elements of $\mathrm{dom}(I') \setminus \mathrm{dom}(I)$ as follows. Because sim is a k-bounded simulation and $k > 0$, it is in particular a 1-bounded simulation, so we have $\mathsf{sim}(a) \in \pi_{R^p}(\mathsf{Chase}(I_0, \Sigma_{\text{UID}}))$. Hence, because $\tau \in \Sigma^{\text{rev}}_{\text{UID}} \subseteq \Sigma_{\text{UID}}$, there is a fact $F_{\text{w}} = S(\boldsymbol{b'})$ in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ with $b'_q = \mathsf{sim}(a)$. We call F_{w} the *chase witness*. For any $b \in \mathrm{dom}(I') \setminus \mathrm{dom}(I)$, letting S^r be some position of the new fact F_{n} where b appears (so $b = b_r$), we define $\mathsf{sim}'(b_r) := b'_r$. ◁

We do not know yet whether the result (I', sim') of a thrifty chase step on an aligned superinstance (I, sim) is still an aligned superinstance; we will investigate this later.

Now that we have defined thrifty chase steps on aligned superinstances, we can clarify the role of the directionality condition. Its goal is to ensure, intuitively, that as chase steps go "downwards" in the original chase, thrifty chase steps on aligned superinstances makes the sim mapping go "downwards" in the chase as well. Formally:

Lemma 12.3.2 (Directionality)**.** *Let J be an aligned superinstance of I_0 for Σ_{UID} and Σ_{UFD}, and consider the application of a thrifty chase step for a UID $\tau : R^p \subseteq S^q$. Consider the chase witness $F_{\text{w}} = S(\boldsymbol{b'})$. Then b'_q is the exported element of F_{w}.*

Proof. Let $F_\mathrm{a} = R(\boldsymbol{a})$ be the active fact in J, let $F_\mathrm{n} = S(\boldsymbol{b})$ be the new fact of J', and let $\tau : R^p \subseteq S^q$ be the UID, so $a_p = b_q$ is the exported element of this chase step. Let $F_\mathrm{w} = S(\boldsymbol{b'})$ be the chase witness in $\mathsf{Chase}(I_0, \Sigma_\mathrm{UID})$. Assume by way of contradiction that b'_q was not the exported element in F_w, so that it was introduced in F_w. In this case, as $\mathsf{sim}(a_p) = \mathsf{sim}(b_q) = b'_q$, by the directionality condition in the definition of aligned superinstances, we have $a_p \in \pi_{S^q}(J)$, which contradicts the fact that $a_p \in \mathsf{Wants}(J, \tau)$. Hence, we have proved by contradiction that b'_q was the exported element in F_w. □

This observation will be important to connect fact-saturation to the *fact-thrifty chase steps* that we now define:

Definition 12.3.3. We define a *fact-thrifty chase step*, using the notation of Definition 11.3.2, as follows: if $\mathsf{NDng}(S^q)$ is non-empty, choose one fact $F_\mathrm{r} = S(\boldsymbol{c})$ of $I \backslash I_0$ that achieves the fact class of $F_\mathrm{w} = S(\boldsymbol{b'})$ (that is, $\mathsf{sim}(c_i) \simeq_k b'_i$ for all i), and use F_r to define $b_r := c_r$ for all $S^r \in \mathsf{NDng}(S^q)$.

We also call a fact-thrifty chase step *fresh* if for all $S^r \in \mathsf{Dng}(S^q)$, we take b_r to be a fresh element only occurring at that position (and extend sim' accordingly). ◁

We first show that, on *fact-saturated* instances, any UID violation can be repaired by a fact-thrifty chase step; this uses Lemma 12.3.2. More specifically, we show that, for any relation-thrifty chase step that we could want to apply, we could apply a fact-thrifty chase step instead.

Lemma 12.3.4 (Fact-thrifty applicability)**.** *For any fact-saturated superinstance I of an instance I_0, for any UID $\tau : R^p \subseteq S^q$ of $\Sigma^\mathrm{rev}_\mathrm{UID}$, for any element $a \in \mathsf{Wants}(I, \tau)$, we can apply a fact-thrifty chase step on a with τ to satisfy this violation. Further, for any new fact $S(\boldsymbol{e})$ that we can create by chasing on a with τ with a relation-thrifty chase step, we can instead apply a fact-thrifty chase step on a with τ to create a fact $S(\boldsymbol{b})$ with $b_r = e_r$ for all $S^r \in \mathsf{Pos}(S) \backslash \mathsf{NDng}(S^r)$.*

Proof. We prove the first part of the statement by justifying the existence of the fact F_r, which only needs to be done if $\mathsf{NDng}(S^q)$ is non-empty. In this case, considering the fact $F_\mathrm{w} = S(\boldsymbol{b'})$ in $\mathsf{Chase}(I_0, \Sigma_\mathrm{UID})$, we know by Lemma 12.3.2 that b'_q is the exported element in F_w. Hence, letting D be the fact class of F_w, we have $D = (S^q, \boldsymbol{C})$ for some $\boldsymbol{C}$, and D is in $\mathsf{AFactCl}$ because $\mathsf{NDng}(S^q)$ is non-empty. Hence, by definition of fact-saturation, there is a fact $F_\mathrm{r} = S(\boldsymbol{c})$ in J such that, for all $S^r \in \mathsf{Pos}(R)$, we have $\mathsf{sim}(c_i) \in C_i$, i.e., $\mathsf{sim}(c_i) \simeq_k b'_i$ in $\mathsf{Chase}(I_0, \Sigma_\mathrm{UID})$. This proves the first part of the claim.

For the second part of the claim, observe that the definition of fact-thrifty chase steps only imposes conditions on the non-dangerous positions, so considering any new fact $S(\boldsymbol{e})$ created by a relation-thrifty chase step, changing its non-dangerous positions to follow the definition of fact-thrifty chase steps, we can create it with a fact-thrifty chase step. □

We now look at which properties are preserved on the result (I', sim') of fact-thrifty chase steps. First note that fact-thrifty chase steps are in particular relation-thrifty, so I' is still weakly-sound and still satisfies Σ_UFD (by Lemmas 11.3.3 and 11.3.5). However, we do not know yet whether (I', sim') is an aligned superinstance for Σ_UFD and Σ_UID.

For now, we show that it is the case for *fresh* fact-thrifty chase steps:

Lemma 12.3.5 (Fresh fact-thrifty preservation)**.** *For any fact-saturated aligned superinstance J of I_0 (for Σ_{UFD} and Σ_{UID}), the result J' of a fresh fact-thrifty chase step on J is still a fact-saturated aligned superinstance of I_0.*

We prove this result in the rest of the section. For *non-fresh* fact-thrifty chase steps, the analogous claim is not true in general: it requires us to introduce *essentiality*, the focus of the next section, and relies on the assumption reversible that we made on $\Sigma^{\text{rev}}_{\text{UID}}$ and Σ_{UFD}.

To prove the Fresh Fact-Thrifty Preservation Lemma, we first make a general claim about how we can extend a superinstance by adding a fact, and preserve bounded simulations.

Lemma 12.3.6. *Let $n \in \mathbb{N}$. Let I_1 and I be instances and sim be a n-bounded simulation from I_1 to I. Let I_2 be a superinstance of I_1 defined by adding one fact $F_{\text{n}} = R(\boldsymbol{a})$ to I_1, and let sim' be a mapping from I_2 to I such that $\mathit{sim}'_{|I_1} = \mathit{sim}$. Assume there is a fact $F_{\text{w}} = R(\boldsymbol{b})$ in I such that, for all $R^i \in \mathit{Pos}(R)$, $\mathit{sim}'(a_i) \simeq_n b_i$. Then sim' is an n-bounded simulation from I_2 to I.*

Proof. We prove the claim by induction on n. The base case of $n = 0$ is immediate.

Let $n > 0$, assume that the claim holds for $n - 1$, and show that it holds for n. As sim is an n-bounded simulation, it is an $(n-1)$-bounded simulation, so we know by the induction hypothesis that sim' is an $(n-1)$-bounded simulation.

Let us now show that it is an n-bounded simulation. Let $a \in \text{dom}(I_2)$ be an element and show that $(I_2, a) \leqslant_n (I, \mathsf{sim}'(a))$. Hence, for any $F = S(\boldsymbol{a})$ a fact of I_2 with $a_p = a$ for some p, we must show that there exists a fact $F' = S(\boldsymbol{a}')$ of I with $a'_p = \mathsf{sim}'(a_p)$ and $(I_2, a_q) \leqslant_{n-1} (I, a'_q)$ for all $S^q \in \mathsf{Pos}(S)$.

The first possibility is that F is the new fact $F_{\text{n}} = R(\boldsymbol{a})$. In this case, as we have $(I, b_p) \leqslant_n (I, \mathsf{sim}'(a_p))$, considering F_{w}, we deduce the existence of a fact $F'_{\text{w}} = R(\boldsymbol{c})$ in I such that $c_p = \mathsf{sim}'(a_p)$ and $(I, b_q) \leqslant_{n-1} (I, c_q)$ for all $1 \leqslant q \leqslant |R|$. We take $F' = F'_{\text{w}}$ as our witness fact for F. By construction we have $c_p = \mathsf{sim}'(a_p)$. Fixing $1 \leqslant q \leqslant |R|$, to show that $(I_2, a_q) \leqslant_{n-1} (I, c_q)$, we use the fact that sim' is an $(n-1)$-bounded simulation to deduce that $(I_2, a_q) \leqslant_{n-1} (I, \mathsf{sim}'(a_q))$. Now, we have $(I, \mathsf{sim}'(a_q)) \leqslant_{n-1} (I, b_q)$, and as we explained we have $(I, b_q) \leqslant_{n-1} (I, c_q)$, so we conclude by transitivity.

If F is another fact, then it is a fact of I_1, so its elements are in $\text{dom}(I_1)$, and as sim' coincides with sim on such elements, we conclude because sim is a n-bounded simulation. □

We now prove the Fresh Fact-Thrifty Preservation Lemma, which concludes the section:

Proof of Lemma 12.3.5. It is immediate that, letting $J' = (I', \mathsf{sim}')$ be the result of the fact-thrifty chase step, I' is still a finite superinstance of I_0, and it still satisfies Σ_{UFD}, because fact-thrifty chase steps are relation-thrifty chase steps, so we can still apply Lemma 11.3.5.

To show that sim' is still a k-bounded simulation, we apply Lemma 12.3.6 with $F_{\text{n}} = S(\boldsymbol{b})$ and $F_{\text{w}} = S(\boldsymbol{b}')$. Indeed, letting $\tau : R^p \subseteq S^q$ be the applied UID in $\Sigma^{\text{rev}}_{\text{UID}}$, we have $\mathsf{sim}'(b_q) = b'_q$ by definition, and have set $\mathsf{sim}'(b_r) := b'_r$ for all $S^r \in \mathsf{Dng}(S^q)$ (note that each such b_r occurs at only one position). For $S^r \in \mathsf{NDng}(S^q)$, we have

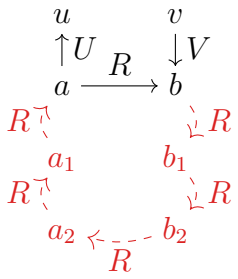


Figure 12.2: Cf. Example 12.4.1

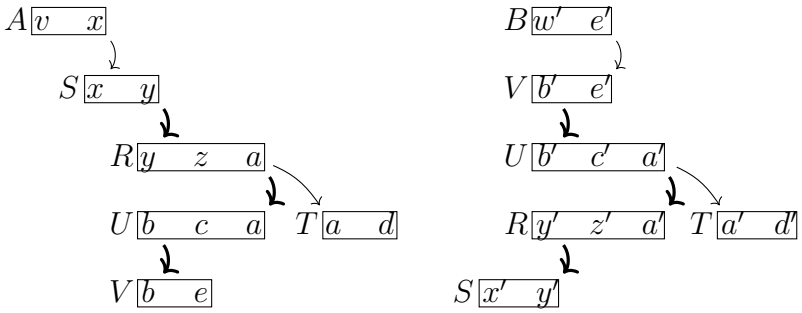


Figure 12.3: UID Chase Similarity Theorem (see Example 12.5.1)

$\mathsf{sim}'(b_r) \simeq_k b'_r$ in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ by definition of a fact-thrifty chase step. Hence, by Lemma 12.3.6, sim' is still a k-bounded simulation from I' to $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$.

We now check the directionality condition on elements of $\mathrm{dom}(I')\setminus \mathrm{dom}(I)$, namely, we show: for $S^r \neq S^q$, if $b_r \in \mathrm{dom}(I')\setminus \mathrm{dom}(I)$, then b_r occurs in J' at the position where $\mathsf{sim}'(b_r)$ was introduced in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$. By the Directionality Lemma (Lemma 12.3.2) we know that b'_q was the exported element of F_{w}. Hence, as $\mathsf{sim}'(b_r) := b'_r$, we know that b'_r was introduced at position S^r in F_{w} in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$, so the condition is respected.

Last, the preservation of fact-saturation is immediate, and the fact that sim' is the identity on I_0 is immediate because $\mathsf{sim}'_{|I_0} = \mathsf{sim}_{|I_0}$. We show that $\mathsf{sim}'_{|I'\setminus I_0}$ maps to $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})\setminus I_0$, using the directionality condition. Indeed, for all elements $b_r \in \mathrm{dom}(I')\setminus \mathrm{dom}(I)$ (with $S^r \neq S^q$), which are clearly not in I_0, we have fixed $\mathsf{sim}'(b_r) := b'_r$, and as we explained b'_r is introduced in F_{w} in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ so it cannot be an element of I_0; hence b'_r is indeed an element of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})\setminus I_0$. This is the last point we had to verify. □

12.4 Essentiality

The problem of non-fresh fact-thrifty chase steps is that, while they try to preserve k-soundness on the non-dangerous positions, they may not preserve it overall:

Example 12.4.1. Consider the instance $I_0 = \{U(a,u), R(a,b), V(v,b)\}$ depicted as the solid black elements and edges in Figure 12.2. Consider the UID $\tau : R^1 \subseteq R^2$, and the UFD $\varphi : R^1 \to R^2$. We define $\Sigma^{\text{rev}}_{\text{UID}} = \Sigma_{\text{UID}} = \{\tau, \tau^{-1}\}$ and $\Sigma_{\text{UFD}} = \{\varphi, \varphi^{-1}\}$, so that Σ_{UFD} and $\Sigma^{\text{rev}}_{\text{UID}}$ are reversible. We have $a \in \mathsf{Wants}(I, \tau^{-1})$ and $b \in \mathsf{Wants}(I, \tau)$. To satisfy these violations, we can apply a fact-thrifty chase step by τ^{-1} on a and create $F = R(b,a)$, noting that there are no non-dangerous positions. However, the superinstance $I_0 \sqcup \{F\}$ is not a k-sound superinstance of I_0 for $k \geqslant 3$. For instance, it makes the following ACQ true, which is not true in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$: $\exists xyzw\ V(x,y), R(y,z), U(z,w)$.

However, for any value of k, this problem can be avoided in the following way. First, apply k fresh fact-thrifty chase steps by τ to create the chain $R(b,b_1)$, $R(b_1,b_2), \ldots, R(b_{k-1}, b_k)$. Then apply k fresh fact-thrifty chase steps by τ^{-1} to create $R(a_1,a), R(a_2,a_1), \ldots, R(a_k, a_{k-1})$. Now we can apply a non-fresh fact-thrifty chase step by τ^{-1} on a and create $R(b_k, a_k)$, and this does not make any new ACQ of size $\leqslant k$ true. This process is illustrated with red elements and red dashed edges in Figure 12.2 for $k = 2$.

The intuition behind this example is that non-fresh fact-thrifty chase steps may connect together elements at the dangerous positions, but their image by sim may be too dissimilar, so the bounded simulation does not extend. This implies that, in general, the result of a fact-thrifty chase step may not be an aligned superinstance. As the example shows, however, we can avoid that problem if we chase for sufficiently long, so that the "history" of elements no longer contains anything specific about them.

We first formalize this notion for elements of the chase $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$, which we call *essentiality*. We will then define it for aligned superinstances using the sim mapping.

Definition 12.4.2. We define a forest structure on the facts of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$: the facts of I_0 are the roots, and the parent of a fact F not in I_0 is the fact F' that was the active fact for which F was created, so that F' and F share the exported element of F.

For $a \in \text{dom}(\mathsf{Chase}(I_0, \Sigma_{\text{UID}}))$, if a was introduced at position S^r of an S-fact $F = S(\boldsymbol{a})$ created by applying the UID $\tau : R^p \subseteq S^q$ (with $S^q \neq S^r$) to its parent fact F', we call τ the *last* UID of a. The *last two* UID*s* of a are (τ, τ') where τ' is the last UID of the exported element a_q of F (which was introduced in F'). For $n \in \mathbb{N}$, we define the *last* n UID*s* in the same way, for elements of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ introduced after sufficiently many rounds.

We say that a is *n-essential* if its last n UIDs are reversible in Σ_{UID}. This is in particular the case if these last UIDs are in $\Sigma^{\text{rev}}_{\text{UID}}$: indeed, $\Sigma^{\text{rev}}_{\text{UID}}$ satisfies assumption $\mathsf{reversible}$, so for any $\tau \in \Sigma^{\text{rev}}_{\text{UID}}$, we have $\tau^{-1} \in \Sigma^{\text{rev}}_{\text{UID}}$, so that $\tau^{-1} \in \Sigma_{\text{UID}}$. ◁

The point of this definition is the following result, which we state without proof for now. We will prove it in Section 12.5:

Theorem 12.4.3 (UID chase similarity theorem)**.** *For any instance I_0, transitively closed set of* UID*s Σ_{UID}, and $n \in \mathbb{N}$, for any two elements a and b respectively introduced at positions R^p and S^q in* $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$*, if a and b are n-essential, and if $R^p \subseteq S^q$ and $S^q \subseteq R^p$ hold in Σ_{UID}, then $a \simeq_n b$.*

In other words, in the chase, when your last n UIDs were reversible, then your $\simeq_n$-class only depends on the position where you were introduced.

We use this to define a corresponding notion on aligned superinstances: an aligned superinstance is *n-essential* if, for all elements that witness a violation of the UIDs $\Sigma^{\text{rev}}_{\text{UID}}$ that we wish to solve, their sim image is an n-essential element of the chase, introduced at a suitable position. In fact, we introduce a more general definition, which does not require the superinstance to be aligned, i.e., does not require that sim is a k-bounded simulation.

Definition 12.4.4. Let $J = (I, \mathsf{sim})$ be a pair of a superinstance I of I_0 and a mapping sim from I to $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$. Let $k \in \mathbb{N}$. We call $a \in \text{dom}(I)$ *n-essential* in J for $\Sigma^{\text{rev}}_{\text{UID}}$ if for any position $S^q \in \mathsf{Pos}(\sigma)$ such that $a \in \mathsf{Wants}_{\Sigma^{\text{rev}}_{\text{UID}}}(I, S^q)$, then:

- $\mathsf{sim}(a)$ is an n-essential element of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$;
- the position T^v where $\mathsf{sim}(a)$ was introduced in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ is such that $T^v \subseteq S^q$ and $S^q \subseteq T^v$ hold in $\Sigma^{\text{rev}}_{\text{UID}}$, which we write $T^v \sim_{\mathsf{ID}} S^q$ as in the previous chapters.

Note that if there is no UID of $\Sigma^{\text{rev}}_{\text{UID}}$ applicable to a, then a is vacuously n-essential. We call J *n-essential* for $\Sigma^{\text{rev}}_{\text{UID}}$ if, for all $a \in \text{dom}(J)$, a is n-essential in J for $\Sigma^{\text{rev}}_{\text{UID}}$. $\lhd$

We now show that, as we assumed the UIDs of $\Sigma^{\text{rev}}_{\text{UID}}$ to be reversible (assumption $\mathsf{reversible}$), fresh fact-thrifty chase steps by $\Sigma^{\text{rev}}_{\text{UID}}$ never make essentiality decrease, and even make it *increase* on new elements:

Lemma 12.4.5 (Thrifty steps and essentiality)**.** *For any n-essential aligned superinstance J, letting $J' = (I', \mathsf{sim}')$ be the result of a thrifty chase step on J by a UID of $\Sigma^{\text{rev}}_{\text{UID}}$, J' is still n-essential. Further, all elements of $\text{dom}(J')\setminus\text{dom}(J)$ are $(n+1)$-essential in J'.*

Proof. Fix J and J'; note that J' may not be an aligned superinstance. Consider first the elements of $\text{dom}(J)$ in J'. For any $a \in \text{dom}(J)$, by definition of thrifty chase steps, we know that for any $T^v \in \mathsf{Pos}(\sigma)$ such that $a \in \pi_{T^v}(J')$, we have either $a \in \pi_{T^v}(J)$ or $a \in \mathsf{Wants}_{\Sigma^{\text{rev}}_{\text{UID}}}(J, T^v)$. Hence, as $\Sigma^{\text{rev}}_{\text{UID}}$ is transitively closed, for any $U^w \in \mathsf{Pos}(\sigma)$ such that $a \in \mathsf{Wants}_{\Sigma^{\text{rev}}_{\text{UID}}}(J', U^w)$, we have also $a \in \mathsf{Wants}_{\Sigma^{\text{rev}}_{\text{UID}}}(J, U^w)$, and as J is n-essential, we conclude that a is n-essential in J'. Hence, it suffices to show that any element in $\text{dom}(J')\setminus\text{dom}(J)$ is $(n+1)$-essential in J'.

To do this, write $\tau : R^p \subseteq S^q$ the UID applied in the chase step, and let $F_{\text{n}} = S(\boldsymbol{b})$ be the new fact. By definition of thrifty chase steps, we had $b_q \in \mathsf{Wants}_{\Sigma^{\text{rev}}_{\text{UID}}}(J, S^q)$, so that b_q was n-essential because J was; hence, $\mathsf{sim}(b_q)$ is n-essential in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$. By the Directionality Lemma (Lemma 12.3.2), $b'_q := \mathsf{sim}(b_q)$ is also the exported element of the chase witness $F_{\text{w}} = S(\boldsymbol{b'})$. Now, the UID τ applied in the thrifty chase step is also the one applied to create F_{w} in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$, and as $\tau \in \Sigma^{\text{rev}}_{\text{UID}}$, by assumption $\mathsf{reversible}$, we have $\tau^{-1} \in \Sigma^{\text{rev}}_{\text{UID}}$, hence $\tau^{-1} \in \Sigma_{\text{UID}}$. Hence, for all $S^r \in \mathsf{Pos}(S)\setminus\{S^q\}$, b'_r is $(n+1)$-essential in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$, and is introduced at position S^r.

Now, let $a \in \text{dom}(J')\setminus\text{dom}(J)$, and let T^v such that $a \in \mathsf{Wants}_{\Sigma^{\text{rev}}_{\text{UID}}}(J', T^v)$. Let $U^w \in \mathsf{Pos}(\sigma)$ and $\tau' : U^w \subseteq T^v$ that witness this, i.e., $\tau' \in \Sigma^{\text{rev}}_{\text{UID}}$ and $a \in \pi_{U^w}(J')$. By assumption $\mathsf{reversible}$, we have $U^w \sim_{\mathsf{ID}} T^v$ in $\Sigma^{\text{rev}}_{\text{UID}}$. Now, by definition of thrifty chase steps on aligned superinstances, we know that we defined $\mathsf{sim}'(a) := b'_r$ for some S^r where a occurred in F_{n}. Further, by definition of thrifty chase steps, we know that all positions in which a occurs in F_{n}, and thus all positions where it occurs in J', are $\sim_{\mathsf{ID}}$-equivalent in $\Sigma^{\text{rev}}_{\text{UID}}$; in particular $S^r \sim_{\mathsf{ID}} U^w$, hence by transitivity $S^r \sim_{\mathsf{ID}} T^v$. By the previous paragraph, $\mathsf{sim}(a) = b'_r$ is an $(n+1)$-reversible element introduced in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ at position S^r, and we have $S^r \sim_{\mathsf{ID}} T^v$. This shows that a is $(n+1)$-reversible in J'.

Hence, J' is indeed n-reversible, and the elements of $\text{dom}(J')\setminus\text{dom}(J)$ are indeed $(n+1)$-reversible, which concludes the proof. $\square$

In conjunction with the Fresh Fact-Thrifty Preservation Lemma, this implies that applying sufficiently many fresh fact-thrifty chase rounds yields an n-essential aligned superinstance:

Lemma 12.4.6 (Ensuring essentiality)**.** *For any $n \in \mathbb{N}$, applying $n+1$ fresh fact-thrifty chase rounds on a fact-saturated aligned superinstance J by the UIDs of $\Sigma^{\text{rev}}_{\text{UID}}$ yields an n-essential aligned superinstance J'.*

Proof. Fix the aligned superinstance $J = (I, \mathsf{sim})$. We use the Fresh Fact-Thrifty Preservation Lemma to show that the property of being aligned is preserved, so we only show that the result is n-essential. We prove this claim by induction on n.

For the base case, we must show that the result $J' = (I', \mathsf{sim}')$ of a fresh fact-thrifty chase round on J by $\Sigma^{\text{rev}}_{\text{UID}}$ is 0-essential. Let $S^q \in \mathsf{Pos}(\sigma)$ and $a \in \mathsf{Wants}_{\Sigma^{\text{rev}}_{\text{UID}}}(J', S^q)$. As $\Sigma^{\text{rev}}_{\text{UID}}$ is transitively closed, by definition of a chase round, we have $a \in \text{dom}(J') \setminus \text{dom}(J)$, because UID violations on elements of $\text{dom}(J)$ must have been solved in J'; hence, a was created by a fact-thrifty chase step on J. By similar reasoning as in the proof of Lemma 12.4.5, considering the chase witness F_{w} for this chase step, we conclude that $\mathsf{sim}(a)$ was introduced at a position T^v in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ such that $T^v \sim_{\mathsf{ID}} S^q$ in $\Sigma^{\text{rev}}_{\text{UID}}$. Further, $\mathsf{sim}(a)$ is vacuously 0-essential. Hence, J' is indeed 0-essential.

For the induction step, let J' be the result of $n+1$ fresh fact-thrifty chase rounds on J, and show that it is n-essential. By induction hypothesis, the result $J'' = (I'', \mathsf{sim}'')$ of n fresh fact-thrifty chase rounds is $(n-1)$-essential. Now, again by definition of a chase round, for any position $S^q \in \mathsf{Pos}(\sigma)$ and $a \in \mathsf{Wants}_{\Sigma^{\text{rev}}_{\text{UID}}}(J'', S^q)$, we must have $a \in \text{dom}(J') \setminus \text{dom}(J'')$, so that a was created by applying a fact-thrifty chase step on an element a'' in J'' which witnessed a violation of a UID of $\Sigma^{\text{rev}}_{\text{UID}}$. As J'' is $(n-1)$-essential, a' was $(n-1)$-essential in J'', so we conclude by Lemma 12.4.5 that a is n-essential in J'. Hence, we conclude that J' is indeed n-essential. □

Hence, we can ensure k-essentiality. The point of essentiality is to guarantee that the result of *non-fresh* fact-thrifty chase steps on a k-essential aligned superinstance is also an aligned superinstance.

Lemma 12.4.7 (Fact-thrifty preservation)**.** *For any fact-saturated k-essential aligned superinstance J for Σ_{UID} and Σ_{UFD}, the result J' of* any *fact-thrifty chase step on J by a UID of $\Sigma^{\text{rev}}_{\text{UID}}$ is still a fact-saturated and k-essential aligned superinstance.*

Proof. Fix $J' = (I', \mathsf{sim}')$, the UID $\tau : R^p \subseteq S^q$ of $\Sigma^{\text{rev}}_{\text{UID}}$, which is reversible by assumption reversible, and the element $a \in \text{dom}(J)$ to which it is applied.

The fact that k-essentiality is preserved is by Lemma 12.4.5, and fact-saturation is clearly preserved, so we must only show that J' is still an aligned superinstance. The fact that J' is a finite superinstance of I_0 is immediate, and it still satisfies Σ_{UFD} by Lemma 11.3.5 because fact-thrifty chase steps are relation-thrifty chase steps. The directionality condition is clearly respected because any new element in $\text{dom}(J') \setminus \text{dom}(J)$ occurs at least at the position at which its sim'-image was introduced in the chase (namely, the position where it occurs in F_{w}), and the additional conditions on $\mathsf{sim}'_{|I_0}$ and $\mathsf{sim}'_{|J' \setminus I_0}$ are still verified.

The only thing to show is that sim' is still a k-bounded simulation. Let $F_{\text{n}} = S(\boldsymbol{b})$ be the new fact and $F_{\text{w}} = S(\boldsymbol{b'})$ be the chase witness. Now, as in the proof of the Thrifty Steps And Essentiality Lemma, and using the Directionality Lemma, all elements of F_{w} are n-essential (and, except for b'_q, they were introduced at their position of F_{w}).

Now, to show that sim' is a k-bounded simulation, we use Lemma 12.3.6, so it suffices to show that we have $\mathsf{sim}(b_r) \simeq_k b'_r$ for all r. This is the case whenever we have $\mathsf{sim}(b_r) = b'_r$, which is guaranteed by definition for $S^r = S^q$ and for elements in $\mathsf{Dng}(S^q)$ such that $S^r \leftrightarrow_{\text{FUN}} S^q$ does not hold. For non-dangerous elements, the fact that $\mathsf{sim}(b_r) \simeq_k b'_r$ is by definition of fact-thrifty chase steps. For the other positions, there are two cases:

- $b_r \in \text{dom}(I)$, in which case $b_r \in \mathsf{Wants}_{\Sigma^{\text{rev}}_{\text{UID}}}(I, S^r)$. As J is n-essential, $\mathsf{sim}(b_r)$ is an n-essential element of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ introduced at a position T^v such that

$T^v \sim_{\mathsf{ID}} S^r$ holds in $\Sigma^{\text{rev}}_{\text{UID}}$. Now, b'_r is an n-essential element of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ introduced at position S^r. By the UID Chase Similarity Theorem, we then have $\mathsf{sim}'(b_r) \simeq_k b'_r$ in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$

- $b_r \notin \text{dom}(I)$, in which case the claim is immediate unless it occurs at multiple positions. However, by definition of thrifty chase steps, all positions at which it occurs are related by $\sim_{\mathsf{ID}}$ in $\Sigma^{\text{rev}}_{\text{UID}}$, so the corresponding elements of F_{w} are also $\simeq_k$-equivalent by the UID Chase Similarity Theorem: hence we have $\mathsf{sim}'(b_r) \simeq_k b'_r$.

We conclude by Lemma 12.3.6 that J' is indeed an aligned superinstance, which concludes the proof. □

We can now conclude the proof of Theorem 12.1. Let I_0 be the initial instance, and consider $J_0 = (I_0, \text{id})$ which is trivially an aligned superinstance of I_0. Apply the Fact-Saturated Solutions Lemma to obtain a fact-saturated aligned superinstance $J'_0 = (I'_0, \mathsf{sim}')$. We must now show that we can complete J'_0 to a superinstance that satisfies $\Sigma^{\text{rev}}_{\text{UID}}$ as well, which we do with the following variant of the Reversible Relation-Thrifty Completion Proposition (see Proposition 11.4.1):

Proposition 12.4.8 (Reversible Fact-Thrifty Completion)**.** *For any reversible* Σ_{UFD} *and* $\Sigma^{\text{rev}}_{\text{UID}}$*, for any transitively closed* UIDs $\Sigma_{\text{UID}} \supseteq \Sigma^{\text{rev}}_{\text{UID}}$*, for any fact-saturated aligned superinstance* J'_0 *of* I_0 *(for* Σ_{UFD} *and* Σ_{UID}*), we can use fact-thrifty chase steps by* UIDs *of* $\Sigma^{\text{rev}}_{\text{UID}}$ *to construct an aligned fact-saturated superinstance* J_{f} *of* I_0 *(for* Σ_{UFD} *and* Σ_{UID}*) that satisfies* $\Sigma^{\text{rev}}_{\text{UID}}$*.*

To prove this lemma, we first apply the Ensuring Essentiality Lemma with the UIDs of $\Sigma^{\text{rev}}_{\text{UID}}$ to make J'_0 k-essential. By the Fresh Fact-Thrifty Preservation Lemma, the result $J_1 = (I_1, \mathsf{sim}_1)$ is then a fact-saturated k-essential aligned superinstance of I_0 (for Σ_{UID} and Σ_{UFD}).

To prove the Reversible Fact-Thrifty Completion Proposition, we will now use the Reversible Relation-Thrifty Completion Proposition (Proposition 11.4.1) on J_1; but we must refine it to a stronger claim. We do so using the following definition:

Definition 12.4.9. A *thrifty sequence* on an instance I for UIDs Σ_{UID} and UFDs Σ_{UFD} is a sequence L defined inductively as follows, with an *output* $L(I)$ which is a superinstance of I that we also define inductively:

- The empty sequence $L = ()$ is a thrifty sequence, with $L(I) = I$
- Let L' be a thrifty sequence, let $I' = L'(I)$ be the output of L', and let $t = (a, \tau, \boldsymbol{b})$ be a triple formed of an element $a \in \text{dom}(I')$, a UID $\tau : R^p \subseteq S^q$ of Σ_{UID}, and an $|S|$-tuple $\boldsymbol{b}$. We require that the fact $S(\boldsymbol{b})$ can be created in I' by applying a thrifty chase step to a in $L'(I)$ by τ (Definition 11.3.2). Then the concatenation L of L' and t is a thrifty sequence, and its output $L(I)$ is the result of performing this chase step on $L'(I)$, namely, $L(I) := L'(I) \sqcup \{S(\boldsymbol{b})\}$.

The *length* of L is written $|L|$ and the elements of L are indexed by $L_1, \ldots, L_{|L|}$. We define a *relation-thrifty sequence* in the same way with relation-thrifty steps, and likewise define a *fact-thrifty sequence*. ◁

With this definition, the Reversible Relation-Thrifty Completion Proposition (Proposition 11.4.1) implies that there is a relation-thrifty sequence L such that $L(I'_0)$ is a finite weakly-sound superinstance I_f of I_0 that satisfies Σ_UFD and $\Sigma^\mathrm{rev}_\mathrm{UID}$. Our goal to prove the Reversible Fact-Thrifty Completion Proposition is to rewrite L to a fact-thrifty sequence. To do this, we first need to show that any two thrifty sequences that coincide on non-dangerous positions have the same effect in terms of UID violations:

Definition 12.4.10. Let Σ_UID be UIDs and Σ_UFD be UFDs, let I_0 be an instance, and L and L' be thrifty sequences on I_0. We say that L and L' *non-dangerously match* if $|L| = |L'|$ and that for all $1 \leqslant i \leqslant |L|$, writing $L_i = (a, \tau, \boldsymbol{b})$ and $L'_i = (a', \tau', \boldsymbol{b'})$, we have $a = a'$, $\tau = \tau'$, and, writing $\tau : R^p \subseteq S^q$, we have $b_r = b'_r$ for all $S^r \in \mathsf{Pos}(S) \backslash \mathsf{NDng}(S^q)$. ◁

Lemma 12.4.11 (Thrifty sequence rewriting)**.** *Let Σ_UID be UIDs and Σ_UFD be UFDs, let I_0 be an instance, and let L and L' be thrifty sequences on I_0 that non-dangerously match. Then $L(I_0)$ satisfies Σ_UID iff $L'(I_0)$ satisfies Σ_UID.*

Proof. We prove by induction on the common length of L and L' that, if L and L' non-dangerously match, then, for all $U^v \in \mathsf{Pos}(\sigma)$, we have $\pi_{U^v}(L(I_0)) = \pi_{U^v}(L'(I_0))$. If both L and L' have length 0, the claim is trivial. For the induction step, write $I := L(I_0)$ and $I' := L'(I_0)$. Write L as the concatenation of L_2 and its last tuple $t = (a, \tau, \boldsymbol{b})$, and write similarly L' as the concatenation of L'_2 and the last tuple $t' = (a', \tau', \boldsymbol{b'})$. Let $U^v \in \mathsf{Pos}(\sigma)$ and show that $\pi_{U^v}(L(I_0)) = \pi_{U^v}(L'(I_0))$. Clearly L_2 and L'_2 non-dangerously match and are strictly shorter than L and L', respectively, so by the induction hypothesis, writing $I_2 := L_2(I_0)$ and $I'_2 := L'_2(I_0)$, we have $\pi_{U^v}(I_2) = \pi_{U^v}(I'_2)$. Further, we have $\tau = \tau'$; write them as $R^p \subseteq S^q$. We then have $I = I_2 \sqcup S(\boldsymbol{b})$, and $I' = I'_2 \sqcup S(\boldsymbol{b'})$. As we must have $b_r = b'_r$ if $U^v \notin \mathsf{NDng}(S^q)$, there is nothing to show unless we have $U^v \in \mathsf{NDng}(S^q)$. However, in this case, writing U^v as S^r, then, by definition of thrifty chase steps, we have $b_r \in \pi_{S^r}(I_2)$, so that $\pi_{S^r}(I) = \pi_{S^r}(I_2)$. Likewise, $\pi_{S^r}(I') = \pi_{S^r}(I'_2)$, hence $\pi_{S^r}(I) = \pi_{S^r}(I')$. This concludes the induction proof.

We now prove the lemma. Fix $\tau : R^p \subseteq S^q$ in Σ_UID. We have $L(I_0) \models \tau$ iff $\pi_{R^p}(L(I_0)) \backslash \pi_{S^q}(L(I_0)) = \emptyset$, and likewise for $L'(I_0)$. By the result proved in the paragraph above, these conditions are equivalent, and thus we have $L(I_0) \models \tau$ iff $L'(I_0) \models \tau$. □

Hence, considering our fact-saturated aligned superinstance $J_1 = (I_1, \mathsf{sim}_1)$ (for Σ_UID and Σ_UFD), use the rephrasing of the Reversible Relation-Thrifty Completion Proposition to obtain a relation-thrifty sequence L such that $L(I_1)$ satisfies Σ_UID. We modify L inductively to obtain a *fact-thrifty sequence* L' that non-dangerously matches L, in the following manner. Whenever L applies a relation-thrifty step $t = (a, \tau, \boldsymbol{b})$ to the previous instance $L_2(I_1)$, then observe that $L_2(I_1)$ is fact-saturated, because I_1 was fact-saturated and fact-thrifty chase steps preserve fact-saturation, by the Fact-Thrifty Preservation Lemma. Hence, by the Fact-Thrifty Applicability Lemma, instead of applying the relation-thrifty step described by t, we can choose to apply a fact-thrifty step on a with τ, defining the new fact using $\boldsymbol{b}$ except on the non-dangerous positions. By Lemma 12.4.11, the resulting L' also ensures that $L'(I_1)$ satisfies Σ_UID.

Considering now the fact-thrifty sequence L', as J_1 is a fact-saturated k-essential aligned superinstance of I_0 (for Σ_{UID} and Σ_{UFD}), letting $I_{\text{f}} := L'(I_1)$, we can use the Fact-Thrifty Preservation Lemma to define an aligned fact-saturated superinstance $J_{\text{f}} = (I_{\text{f}}, \mathsf{sim}_{\text{f}})$ (for Σ_{UID} and Σ_{UFD}), following each fact-thrifty step, and we have shown that I_{f} satisfies Σ_{UID}. Hence, we have proven the Reversible Fact-Thrifty Completion Proposition.

To prove Theorem 12.1, we can simply apply the proposition with $\Sigma_{\text{UID}} = \Sigma^{\text{rev}}_{\text{UID}}$, and the resulting aligned superinstance $J_{\text{f}} = (I_{\text{f}}, \mathsf{sim}_{\text{f}})$ of I_0 satisfies Σ_{UID} and is k-sound for Σ_{U} and ACQ. Further, it satisfies Σ_{UFD} and is finite, by definition of being an aligned superinstance. Hence, I_{f} is the desired k-universal model, which proves Theorem 12.1.

12.5 UID Chase Similarity Theorem

We conclude the chapter by proving the UID Chase Similarity Theorem:

Theorem 12.4.3 (UID chase similarity theorem)**.** *For any instance I_0, transitively closed set of UIDs Σ_{UID}, and $n \in \mathbb{N}$, for any two elements a and b respectively introduced at positions R^p and S^q in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$, if a and b are n-essential, and if $R^p \subseteq S^q$ and $S^q \subseteq R^p$ hold in Σ_{UID}, then $a \simeq_n b$.*

Note that this result does not involve FDs, and applies to any arbitrary transitively closed set of UIDs, not relying on any finite closure properties, or on assumption reversible. It only assumes that the last n dependencies used to create a and b were reversible.

Example 12.5.1. Consider Figure 12.3 on page 169, which illustrates the neighborhood of two elements, a and a', in the UID chase by some UIDs. Each rectangle represents a higher-arity fact, and edges represent the UIDs used in the chase, with thick edges representing reversible UIDs.

The last UID applied to create a was $S^2 \subseteq R^1$, and the last UID for a' is $V^1 \subseteq U^1$; they are reversible. Further, a is introduced at position R^3 and a' at position U^3, and $R^3 \subseteq U^3$ holds and is reversible. The theorem claims that a that a' are 1-bounded-bisimilar, which is easily verified; in fact, they are 2-bounded-bisimilar. This is intuitively because all child facts of the R-fact at the left must occur at the right by definition of the UID chase, and the parent fact must occur as well because of the reverse of the last UID for a; a similar argument ensures that the facts at the right must be reflected at the left.

However, note that a and a' are not 3-bounded-bisimilar: the A-fact at the left is not reflected at the right, and vice-versa for the B-fact, because these UIDs are not reversible,

To prove the theorem, fix the instance I_0 and the set Σ_{UID} of UIDs. We first show the following easy lemma:

Lemma 12.5.2. *For any $n > 0$ and position R^p, for any two elements a, b of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ introduced at position R^p in two facts F_a and F_b, letting a' and b' be the exported elements of F_a and F_b, if $a' \simeq_{n-1} b'$, then $a \simeq_n b$.*

Proof. We proceed by induction on n. By symmetry, it suffices to show that $(\mathsf{Chase}(I_0, \Sigma_{\text{UID}}), a) \leqslant_n (\mathsf{Chase}(I_0, \Sigma_{\text{UID}}), b)$.

For the base case $n = 1$, observe that, for every fact F of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ where a occurs at some position S^q, there are two cases. Either $F = F_a$, so we can pick F_b as the representative fact, or the UID $R^p \subseteq S^q$ is in Σ_{UID} so we can pick a corresponding fact for b by definition of the chase. Hence, the claim is shown for $n = 1$.

For the induction step, we proceed in the same way. If $F = F_a$, we pick F_b as representative fact, and use either the hypothesis on a' and b' or the induction hypothesis (for other elements of F_a and F_b) to justify that F_b is suitable. Otherwise, we pick the corresponding fact for b which must exist by definition of the chase, and apply the induction hypothesis to the other elements of the fact to conclude. □

We now prove the UID Chase Similarity Theorem. Throughout the proof, we write $R^p \sim_{\mathsf{ID}} S^q$ as shorthand to mean that $R^p \subseteq S^q$ and $S^q \subseteq R^p$ are in Σ_{UID}: it is still the case that $\sim_{\mathsf{ID}}$ is an equivalence relation, even without assumption reversible.

We prove the main claim by induction on n: for any positions R^p and S^q such that $R^p \sim_{\mathsf{ID}} S^q$, for any two n-essential elements a and b respectively introduced at positions R^p and S^q, we have $a \simeq_n b$. By symmetry it suffices to show that $a \leqslant_n b$ in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$, formally, $(\mathsf{Chase}(I_0, \Sigma_{\text{UID}}), a) \leqslant_n (\mathsf{Chase}(I_0, \Sigma_{\text{UID}}), b)$.

The base case of $n = 0$ is immediate.

For the induction step, fix $n > 0$, and assume that the result holds for $n - 1$. Fix R^p and S^q such that $R^p \sim_{\mathsf{ID}} S^q$, and let a, b be two n-essential elements introduced respectively at R^p and S^q in facts F_a and F_b. Note that by the induction hypothesis we already know that $a \leqslant_{n-1} b$ in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$; we must show that this holds for n.

First, observe that, as a and b are n-essential with $n > 0$, they are not elements of I_0. Hence, by definition of the chase, for each one of them, the following is true: for each fact of the chase where the element occurs, it only occurs at one position, and all other elements co-occurring with it in a fact of the chase occur only at one position and in exactly one of these facts. Thus, to prove the claim, it suffices to construct a mapping φ from the set $N_1(a)$ of the facts of $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ where a occurs, to the set $N_1(b)$ of the facts where b occurs, such that the following holds: for every fact $F = T(\boldsymbol{a})$ of $N_1(a)$, letting T^c be the one position of F such that $a_c = a$, the element b occurs at position T^c in $\varphi(F) = T(\boldsymbol{b})$, and for every i, we have $a_i \leqslant_{n-1} b_i$.

By construction of the chase (using the Unique Witness Property), $N_1(a)$ consists of exactly the following facts:

- The fact $F_a = R(\boldsymbol{a})$, where $a_d = a'$ is the exported element (for a certain $R^d \neq R^p$) and $a_p = a$ was introduced at R^p in F_a. Further, for $i \notin \{p, d\}$, the element a_i was introduced at R^i in F_a.
- For every UID $\tau : R^p \subseteq V^g$ of Σ_{UID}, a V-fact F_a^τ where the element at position V^g is a. Further, for $i \neq g$, the element at position V^i in F_a^τ was introduced at this position in that fact.

A similar characterization holds for b: we write the corresponding facts F_b and F_b^τ. We construct the mapping φ as follows:

- If $R^p = S^q$ then set $\varphi(F_a) := F_b$; otherwise, as $\tau : S^q \subseteq R^p$ is in Σ_{UID}, set $\varphi(F_a) := F_b^\tau$.

- For every UID $\tau : R^p \subseteq V^g$ of Σ_{UID}, as $R^p \sim_{\mathsf{ID}} S^q$, by transitivity, either $S^q = V^g$ or the UID $\tau' : S^q \subseteq V^g$ is in Σ_{UID}. In the first case, set $\varphi(F_a^\tau) := F_b$. In the second case, set $\varphi(F_a^\tau) := F_b^{\tau'}$.

We must now show that this mapping φ from $N_1(a)$ to $N_1(b)$ satisfies the required conditions. Verify that indeed, by construction, whenever a occurs at position T^c in F, then $\varphi(F)$ is a T-fact where b occurs at position T^c. So we must show that for any $F \in N_1(a)$, writing $F = T(\boldsymbol{a})$ and $\varphi(F) = T(\boldsymbol{b})$, with $a_c = a$ and $b_c = b$ for some c, we have indeed $a_i \leqslant_{n-1} b_i$ for all $T^i \in \mathsf{Pos}(T)$. If $n = 1$ there is nothing to show and we are done, so we assume $n \geqslant 2$. If $i = c$ then the claim is immediate by the induction hypothesis; otherwise, we distinguish two cases:

1. $F = F_a$ (so that $T = R$ and $c = p$), or $F = F_a^\tau$ such that the UID $\tau : R^p \subseteq T^c$ is reversible, meaning that $\tau^{-1} \in \Sigma_{\text{UID}}$. In this case, by construction, either $\varphi(F) = F_b$ or $\varphi(F) = F_b^{\tau'}$ for $\tau' : S^q \subseteq T^c$; τ' is then reversible, because $R^p \sim_{\mathsf{ID}} S^q$ and $R^p \sim_{\mathsf{ID}} T^c$.

   We show that for all $1 \leqslant i \leqslant |T|$ such that $i \neq c$, the element a_i is $(n-1)$-essential and was introduced in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ at a position in the $\sim_{\mathsf{ID}}$-class of T^i. Once we have proved this, we can show the same for all b_i in a symmetric way, so that we can conclude that $a_i \leqslant_{n-1} b_i$ by induction hypothesis. To see why the claim holds, we distinguish two subcases. Either a_i was introduced in F, or we have $F = F_a$, $i = d$ and a_i is the exported element for a.

   In the first subcase, a_i was created by applying the reversible UID τ and the exported element was a, which is n-essential, so a_i is $(n-1)$-essential (in fact it is even $(n+1)$-essential), and a_i is introduced at position T^i.

   In the second subcase, a_i is the exported element used to create a, which is n-essential, so a_i is $(n-1)$-essential; and as $n \geqslant 2$, the last dependency applied to create a_i is reversible, so that a_i was introduced at a position in the same $\sim_{\mathsf{ID}}$-class as T^i.

   Hence, we have proved the desired claim for the first case.

2. $F = F_a^\tau$ such that $\tau : R^p \subseteq T^c$ is not reversible. In this case, we cannot have $T^c = S^q$ (because we have $R^p \sim_{\mathsf{ID}} S^q$), so we must have $\varphi(F) = F_b^{\tau'}$ with $\tau' : S^q \subseteq T^c$. Now, all a_i for $i \neq c$ were introduced in F at position T^i, and likewise for the b_i in $\varphi(F)$. Using Lemma 12.5.2, as $a \simeq_{n-1} b$, we conclude that $a_i \simeq_n b_i$, hence $a_i \leqslant_{n-1} b$.

This concludes the proof of the UID Chase Similarity Theorem, thus completing the proof of Theorem 12.1.

Chapter 13

Decomposing the Constraints

In this chapter, we lift assumption reversible, proving:

Theorem 13.1. *Finitely-closed UIDs and UFDs have finite k-universal models for ACQs.*

13.1 Partitioning the UIDs

We write the UFDs as Σ_{UFD} and the UIDs as Σ_{UID}. We will proceed by partitioning Σ_{UID} into subsets of UIDs which are either reversible or are much simpler to deal with.

Our desired notion of partition respects an order on UID, which we now define. As we will show (Lemma 13.2.3), the order is also respected by thrifty chase steps.

Definition 13.1.1. For any $\tau, \tau' \in \Sigma_{\text{UID}}$, we write $\tau \rightarrowtail \tau'$ when we can write $\tau = R^p \subseteq S^q$ and $\tau' = S^r \subseteq T^v$ with $S^q \neq S^r$, and the UFD $S^r \to S^q$ is in Σ_{UFD}. An *ordered partition* $(P_1, \ldots, P_n)$ of Σ_{UID} is a partition of Σ_{UID} (i.e., $\Sigma_{\text{UID}} = \bigsqcup_i P_i$) such that for any $\tau \in P_i$, $\tau' \in P_j$, if $\tau \rightarrowtail \tau'$ then $i \leqslant j$. ◁

The point of partitioning Σ_{UID} is to be able to control the structure of the UIDs in each class:

Definition 13.1.2. We call $P \subseteq \Sigma_{\text{UID}}$ *reversible* if P and Σ_{UFD} are reversible (Definition 9.2.4). We say $P \subseteq \Sigma_{\text{UID}}$ is *trivial* if we have $P = \{\tau\}$ for some $\tau \in \Sigma_{\text{UID}}$ such that $\tau \not\rightarrowtail \tau$. A partition is *manageable* if it is ordered and all of its classes are either reversible or trivial. ◁

As we will show in Section 13.3, we can always construct a manageable partition of Σ_{UID}:

Proposition 13.1.3. *Any conjunction Σ_{UID} of UIDs closed under finite implication has a manageable partition.*

Example 13.1.4. Consider two ternary relations R and S. Consider the UIDs $\tau_R : R^1 \subseteq R^2$, $\tau_S : S^2 \subseteq S^3$, $\tau_{RS} : R^3 \subseteq S^1$, and the UFDs $\varphi_R : R^1 \to R^2$, $\varphi_S : S^2 \to S^3$, $\varphi'_R : R^3 \to R^1$, and $\varphi'_S : S^3 \to S^1$. The UIDs τ_R^{-1} and τ_S^{-1}, and the UFDs φ_R^{-1}, φ_S^{-1}, and $R^3 \to R^2$, $S^2 \to S^1$, are finitely implied. The two relations R

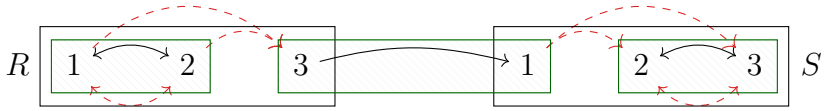


Figure 13.1: Manageable partition (see Example 13.1.4)

and S are illustrated in Figure 13.1, where UIDs as drawn as solid black edges, and UFDs as dashed red edges that are *reversed* (this follows Definition 13.3.1).

A manageable partition of the UIDs of the finite closure is $(\{\tau_R, \tau_R^{-1}\}, \{\tau_{RS}\}, \{\tau_S, \tau_S^{-1}\})$, where the first and third classes are reversible and the second is trivial. The classes of the partition are drawn as green hatched rectangles in Figure 13.1; they are intuitively related to a topological sort of the graph of the black and red edges (see Definition 13.3.3).

13.2 Using Manageable Partitions

Fix the instance I_0 and the finitely closed constraints Σ_{U} formed of UIDs Σ_{UID} and UFDs Σ_{UFD}. To prove Theorem 13.1, starting with the initial aligned superinstance $J_0 = (I_0, \mathrm{id})$ of I_0 (for Σ_{UID} and Σ_{UFD}), we first note that the Fact-Saturated Solutions Lemma (Lemma 12.2.4) does not use assumption reversible. Hence, we apply it (with Σ_{UID}) to obtain from I_0 an aligned fact-saturated superinstance J_1 of I_0 (for Σ_{UFD} and Σ_{UID}). This is the *saturation process*.

The goal is now to apply a *completion process* to satisfy Σ_{UID}, which we formalize as the following proposition. Recall the definition of thrifty sequences (Definition 12.4.9). We refine the definition below.

Definition 13.2.1. We define a *preserving fact-thrifty sequence* L (for UIDs Σ_{UID} and UFDs Σ_{UFD}) on any *fact-saturated aligned superinstance* J of I_0 in the following inductive way, with its *output* $L(J)$ also being a fact-saturated aligned superinstance:

- The empty list $L = ()$ is a preserving fact-thrifty sequence, with output $L(J) := J$.
- Let L be the concatenation of a preserving fact-thrifty sequence L' and a triple $t = (a, \tau, \boldsymbol{b})$. Let $J' := L'(J)$ be the output of L'. We call L *preserving*, if one of the following holds:
  - t is a fresh fact-thrifty chase step. In this case, by the Fresh Fact-Thrifty Preservation Lemma, J' is indeed a fact-saturated aligned superinstance of I_0.
  - J' is k-essential for some subset $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ of Σ_{UID} such that $\tau \in \Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ and $\Sigma^{\mathrm{rev}}_{\mathrm{UID}}$ and Σ_{UFD} are reversible. In this case, by the Fact-Thrifty Preservation Lemma, J is a fact-saturated aligned superinstance of I_0 (which is also k-essential for the same subset).

  In either case, the output of L is the aligned superinstance obtained as the result of applying the fact-thrifty chase step represented by t on J'. ◁

We can now state our intended result, which implies Theorem 13.1:

Proposition 13.2.2 (Fact-thrifty completion)**.** *Let $\Sigma_{\mathrm{U}} = \Sigma_{\mathrm{UFD}} \sqcup \Sigma_{\mathrm{UID}}$ be finitely closed UFDs and UIDs, and let I_0 be an instance that satisfies UFDs. For any fact-saturated aligned superinstance J of I_0 for Σ_{U}, there is a preserving fact-thrifty sequence L such that $L(J)$ satisfies Σ_{UID}.*

We prove Proposition 13.2.2, and from it Theorem 13.1, in the rest of the section. We construct a manageable partition $\boldsymbol{P} = (P_1, \ldots, P_n)$ of Σ_{UID} using Proposition 13.1.3. Now, for $1 \leqslant i \leqslant n$, we use fact-thrifty chase steps by UIDs of P_i to extend the fact-saturated aligned superinstance J_i to a larger one J_{i+1} that satisfies P_i.

The crucial point is that we can apply fact-thrifty chase steps to satisfy P_i without creating any new violations of P_j for $j < i$, and hence we can make progress following the partition. The reason for this is the following easy fact about thrifty chase steps:

Lemma 13.2.3. *Let J be an aligned superinstance of I_0 and J' be the result of applying a thrifty chase step on J for a UID τ of Σ_{UID}. Assume that a UID τ' of Σ_{UID} was satisfied by J but is not satisfied by J'. Then $\tau \rightarrowtail \tau'$.*

Proof. Fix J, J', $\tau : R^p \subseteq S^q$ and τ'. As chase steps add a single fact, the only new UID violations in J' relative to I are on elements in the newly created fact $F_{\mathrm{n}} = S(\boldsymbol{b})$, As Σ_{UID} is transitively closed, F_{n} can introduce no new violation on the exported element b_q. Now, as thrifty chase steps always reuse existing elements at non-dangerous positions, we know that if $S^r \in \mathsf{NDng}(S^q)$ then no new UID can be applicable to b_r. Hence, if a new UID is applicable to b_r for $S^r \in \mathsf{Pos}(S)$, then necessarily $S^r \in \mathsf{Dng}(S^q)$. By definition of dangerous positions, the UFD $S^r \to S^q$ is then in Σ_{UFD}, and we have $S^r \neq S^q$. Hence, writing $\tau' : S^r \subseteq T^r$, we see that $\tau \rightarrowtail \tau'$. □

The lemma justifies our definition of ordered partition, since it will allow us to do an inductive argument to prove Proposition 13.2.2. Using the fact that $\boldsymbol{P}$ is ordered ensures that we can indeed apply fact-thrifty chase steps to satisfy each P_i individually, dealing with them in the order of the partition.

Thus, to prove Proposition 13.2.2, consider each class P_i in order. As $\boldsymbol{P}$ is manageable, there are two cases: either P_i is trivial or it is reversible.

First, if P_i is trivial, it can simply be satisfied by a preserving fact-thrifty sequence L_i of fresh fact-thrifty chase steps using the one UID of P_i, as follows immediately from Lemma 13.2.3.

Lemma 13.2.4. *For any trivial class $\{\tau\}$, performing one chase round on an aligned fact-saturated superinstance J of I_0 by fresh fact-thrifty chase steps for τ yields an aligned superinstance J' of I_0 that satisfies τ.*

Proof. Fix J, J' and τ. All violations of τ in J have been satisfied in J' by definition of J', so we only have to show that no new violations of τ were introduced in J'. But by Lemma 13.2.3, as $\tau \not\rightarrowtail \tau$, each fresh fact-thrifty chase step cannot introduce such a violation, hence there is no new violation of τ in J'. Hence, $J' \models \tau$. □

Second, returning to the proof of Proposition 13.2.2, the interesting case is that of a *reversible* P_i, for which we have done the work of the last three chapters. We satisfy a reversible P_i by a preserving fact-thrifty sequence L_i obtained using the Reversible Fact-Thrifty Completion Proposition (Proposition 12.4.8). Indeed, J_i is a

fact-saturated aligned superinstance of I_0 for Σ_{UFD} and Σ_{UID}, and by definition of P_i being reversible, letting $\Sigma^{\text{rev}}_{\text{UID}} := P_i$, the constraints Σ_{UFD} and $\Sigma^{\text{rev}}_{\text{UID}}$ are reversible. By the Reversible Fact-Thrifty Completion Proposition, we can thus construct a fact-thrifty sequence L_i (by UIDs of $\Sigma^{\text{rev}}_{\text{UID}}$) such that $J_{i+1} := L_i(J_i)$ is a fact-saturated aligned superinstance of I_0 for Σ_{UFD} and Σ_{UID} that satisfies P_i. Further, from the proof, it is clear that L_i is preserving.

Hence, in either of the two cases, we construct a preserving fact-thrifty sequence L_i and $J_{i+1} := L_i(J_i)$ satisfies P_i. Further, as L_i only performs fact-thrifty chase steps by UIDs of P_i, J_{i+1} actually satisfies $\bigcup_{j \leqslant i} P_j$, thanks to Lemma 13.2.3.

The concatenation of the preserving fact-thrifty sequences L_i for each P_i is thus a preserving fact-thrifty sequence L whose final result $L(J) = J_{n+1}$ is thus an aligned superinstance of I_0 that satisfies Σ_{UID}, which proves the Fact-Thrifty Completion Proposition. As an aligned superinstance, J_{n+1} is also finite, satisfies Σ_{UFD}, and is k-sound for ACQ; so it is k-universal for Σ_{U} and ACQ. This concludes the proof of Theorem 13.1.

13.3 Building Manageable Partitions

The only missing part is to show how manageable partitions are constructed (Proposition 13.1.3), which we show in this section. We will construct the manageable partition using a *constraint graph* defined from the dependencies, inspired by the multigraph used in the proof of Theorem 8.2.1 in [Cosmadakis, Kanellakis, and Vardi 1990].

Definition 13.3.1. Given a set Σ_{U} of finitely closed UIDs and UFDs on signature σ, the *constraint graph* $G(\Sigma_{\text{U}})$ is the directed graph with vertex set $\mathsf{Pos}(\sigma)$ and with the following edges:

- For each UID $R^p \subseteq S^q$ in Σ_{U}, an edge from R^p to S^q
- For each UFD $R^a \to R^b$ in Σ_{U}, an edge from R^b to R^a.

As we forbid trivial UIDs and UFDs, $G(\Sigma_{\text{U}})$ has no self-loop, but it may contain both the edge (R^p, S^q) and (S^q, R^p). However, we do not represent multiple edges in $G(\Sigma_{\text{U}})$: for instance, if the UID $R^a \subseteq R^b$ and the UFD $R^b \to R^a$ hold in $G(\Sigma_{\text{U}})$, we only create a single copy of the edge (R^a, R^b). ◁

Hence, fix the finitely closed UIDs and UFDs $\Sigma_{\text{U}} := \Sigma_{\text{UID}} \wedge \Sigma_{\text{UFD}}$, and construct the graph $G(\Sigma_{\text{U}})$. As observed by [Cosmadakis, Kanellakis, and Vardi 1990], the graph $G(\Sigma_{\text{U}})$ has the following property, which will be needed to show that classes are reversible:

Lemma 13.3.2. *For any edge e occurring in a cycle in $G(\Sigma_{\text{U}})$, for any dependency τ which caused the creation of e, the reverse τ^{-1} of τ is in Σ_{U}.*

Proof. Let e_1 be the edge, and $e_1, \ldots, e_n$ be the cycle (the first vertex of e_1 is the second vertex of e_n), and let τ be the dependency. Consider a cycle of dependencies $\tau_1, \ldots, \tau_n$, with $\tau_1 = \tau$, such that each τ_i caused the creation of edge e_i in $G(\Sigma_{\text{U}})$. We must show that the reverse τ^{-1} of τ is in Σ_{U}.

If all the τ_i are UIDs, then, as Σ_{UID} is closed under the transitivity rule, we apply it to $\tau_2, \ldots, \tau_n$ and deduce that τ_1^{-1} is in Σ_{UID}. Likewise, if all the τ_i are UFDs, then we proceed in the same way because Σ_{UFD} is closed under the transitivity rule.

If the τ_i are of alternating types (alternatively UIDs and UFDs), then, recalling that Σ_{U} is closed under the *cycle rule* (see Section 8.2) we deduce that τ_i^{-1} is in Σ_{U} for all i.

In the general case, consider the maximal subsequence $\tau_j, \ldots, \tau_n, \tau_1, \ldots, \tau_i$ $(i < j)$ of consecutive dependencies in the cycle that includes τ and where all dependencies are of the same type. Let τ_{m} be the result of combining these dependencies by the transitivity rule, and consider the cycle $\tau_{\text{m}}, \tau_{i+1}, \ldots, \tau_n, \tau_1, \ldots, \tau_{j-1}$. Collapsing all other consecutive sequences of dependencies to a single dependency using the transitivity rule, and applying the cycle rule as in the previous case, we deduce that τ_{m}^{-1} is in Σ_{U}. Hence, the cycle $\tau_j, \ldots, \tau_n, \tau_1, \ldots, \tau_i, \tau_{\text{m}}^{-1}$ is a cycle of dependencies of the same type as τ, and it includes τ, so we conclude as in the first two cases that τ^{-1} is in Σ_{U}.

Hence, in all cases τ^{-1} is in Σ_{U}. This concludes the proof. □

Compute the strongly connected components of $G(\Sigma_{\text{U}})$, ordered following a topological sort: we label them $V_1, \ldots, V_n$. The order of the V_i guarantees that there are no edges in $G(\Sigma_{\text{U}})$ from V_i to V_j unless $i \leqslant j$.

We will build each class of the manageable partition, either as the set of UIDs within the positions of an SCC (a *reversible* class), or as a singleton UID going from a class V_i to a class V_j with $j > i$ (a *trivial* class). Formally:

Definition 13.3.3. The topological sort of the SCCs of $G(\Sigma_{\text{U}})$, written $V_1, \ldots, V_n$, defines a partition $\boldsymbol{P}$ of the UIDs of Σ_{UID}, in the following manner. For each V_i, if there are any non-trivial UIDs of the form $R^p \subseteq S^q$ with $R^p, S^q \in V_i$, create a class of UIDs (the *main* class) containing all of them. Then, for each UID of the form $R^p \subseteq S^q$ with $R^p \in V_i$ and $S^q \in V_j$ with $j > i$, create a singleton class of UIDs containing exactly that UID (a *satellite* class). The partition $\boldsymbol{P}$ is obtained by taking the concatenation, for i from 1 to n, of the main class of V_i (if it exists) and then all satellite classes of V_i (if any) in an arbitrary order. ◁

Remember that, while the constraint graph reflects both the UIDs and the UFDs, the partition $\boldsymbol{P}$ that we define is a partition of Σ_{UID}, that is, a partition of UIDs, and does not contain UFDs. We first show that $\boldsymbol{P}$ is indeed a partition, and then that it is an ordered partition.

Lemma 13.3.4. *$\boldsymbol{P}$ is indeed a partition of Σ_{UID}.*

Proof. As the SCCs of $G(\Sigma_{\text{U}})$ partition the vertex set of $G(\Sigma_{\text{U}})$, it is clear by construction that any UID occurs in at most a single class of the partition, which must be a class for the SCC of its first position, and either a satellite class or the main class depending on the SCC of its second position.

Conversely, each UID τ is reflected in some class of the partition, for the SCC V_i of its first position: either the second position of τ is also in V_i, so τ is in the main class for V_i; or the second position of τ is in an SCC V_j with $i \neq j$, in which case $i < j$ by definition of a topological sort, and τ is in some satellite class for V_i. Hence, $\boldsymbol{P}$ is indeed a partition of Σ_{UID}. □

Lemma 13.3.5. *$\boldsymbol{P}$ is an ordered partition.*

Proof. Assume by way of contradiction that there are two classes P_i and P_j and $\tau \in P_i$, $\tau' \in P_j$, such that $\tau \rightarrowtail \tau'$ but $i > j$. Let V_p and V_q be the SCCs in which P_i and P_j were created. We must have $p \geqslant q$, so there are two possibilities.

First, if $p = q$, then the first positions of τ and τ' must both be in $V_p = V_q$, and as P_i is not the first class created for $V_p = V_q$, it must be a satellite class. Hence, the second position of τ is in another SCC, say V_r, with $r > p$. Now, as $\tau \rightarrowtail \tau'$, there is a UFD from the first position of τ' to the second position of τ, which implies that there is an edge from V_r to V_p in $G(\Sigma_{\mathrm{U}})$. As $r > p$, this contradicts the fact that the SCCs are ordered following a topological sort.

Second, if we have $p > q$, then again the first position of τ must be in V_p, and the first position of τ' is in V_q. Let V_r be the SCC of the second position of τ. As $\tau \rightarrowtail \tau'$, the UFD from the first position of τ' to the second position of τ witnesses that there is an edge in $G(\Sigma_{\mathrm{U}})$ from V_r to V_q. Hence, we must have $r \leqslant q$. But τ witnesses that there is an edge from p to r in $G(\Sigma_{\mathrm{U}})$, so that we must have $p \leqslant r$. Hence, $p \leqslant q$, but we had assumed $p > q$, a contradiction. □

We now show that $\boldsymbol{P}$ is manageable, by considering each class and showing that it is either trivial or that it is reversible:

Lemma 13.3.6. *Each satellite class in $\boldsymbol{P}$ is trivial.*

Proof. Each satellite class consists by construction of a singleton dependency $\tau = R^p \subseteq S^q$, implying the existence of an edge in the constraint graph $G(\Sigma_{\mathrm{U}})$ from R^p to S^q. Assume by way of contradiction that $\tau \rightarrowtail \tau$. This implies that $R^p \to S^q$ holds in Σ_{UFD}, so there is an edge in $G(\Sigma_{\mathrm{U}})$ from S^q to R^p. Hence, $\{R^p, S^q\}$ is strongly connected, so R^p and S^q belong to the same SCC, which contradicts the definition of a satellite class. □

Lemma 13.3.7. *Each main class in the partition is reversible.*

Proof. Let P_i be the class and V_i be the corresponding SCC. We first show that P_i is transitively closed. Consider two UIDs τ and τ' of P_i that would be combined by the transitivity rule to the UID τ''. As Σ_{UID} is transitively closed, we have $\tau'' \in \Sigma_{\mathrm{UID}}$. Now, if both τ and τ' have both positions in V_i, then so does τ'', so we also have $\tau'' \in P_i$.

Second, to see that every UID τ in P_i is reversible, consider a UID $\tau : R^p \subseteq S^q$ of P_i, with $R^p, S^q \in V_i$. We forbid trivial UIDs, so $R^p \neq S^q$. As V_i is strongly connected, consider a directed path π of edges of $G(\Sigma_{\mathrm{U}})$ from S^q to R^p. Combining π with the edge created in $G(\Sigma_{\mathrm{U}})$ for the UID τ, we deduce the existence of a cycle in $G(\Sigma_{\mathrm{U}})$. Hence, by Lemma 13.3.2, the UID τ^{-1} holds in Σ_{UID}, and it also has both positions in V_i, so τ^{-1} is in P_i.

Third, we prove the claim about UFDs. Consider a UFD $\varphi : R^p \to R^q$ of Σ_{UFD}, with $R^p \neq R^q$. Assume that R^p and R^q occur in a UID of P_i; this implies that $R^p, R^q \in V_i$. By the same reasoning as before, we find a cycle in $G(\Sigma_{\mathrm{U}})$ that includes the edge that corresponds to φ, and deduce that φ^{-1} holds in Σ_{UFD}. □

Hence, $\boldsymbol{P}$ is an ordered partition of Σ_{UID} where each class is either reversible or trivial, i.e., it is a manageable partition. This concludes the proof of Proposition 13.1.3.

Chapter 14

Higher-Arity Functional Dependencies

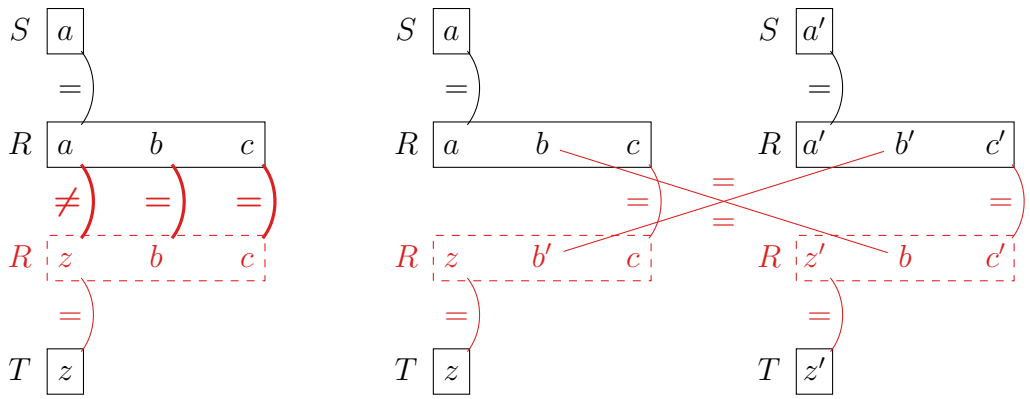
The goal of this chapter is to generalize our results to functional dependencies of arbitrary arity:

Theorem 14.1. *Finitely-closed UIDs and FDs have finite universal models for ACQs.*

We fix the finitely-closed constraints $\Sigma := \Sigma_{\text{FD}} \wedge \Sigma_{\text{UID}}$, consisting of arbitrary-arity FDs Σ_{FD} and UIDs Σ_{UID}. We denote by Σ_{UFD} the *unary* FDs among Σ_{FD}, and write $\Sigma_{\text{U}} := \Sigma_{\text{UFD}} \wedge \Sigma_{\text{UID}}$. From the definition of the finite closure (Section 8.2), it is clear that Σ_{U} is finitely closed as well, so the construction of the previous chapters applies to Σ_{U}.

The problem to address in this chapter is that our completion process to satisfy Σ_{UID} was defined with fact-thrifty chase steps. These chase steps may reuse elements from the same facts at the same positions multiple times. This may violate Σ_{FD}, and it is in fact the only point where we do so in the construction.

Example 14.1. For simplicity, we work with instances rather than aligned superinstances. Consider $I_0 := \{S(a), T(z)\}$, the UIDs $\tau : S^1 \subseteq R^1$ and $\tau' : T^1 \subseteq R^1$ for a 3-ary relation R, and the FD $\varphi : R^2R^3 \to R^1$. Consider $I := I_0 \sqcup \{R(a, b, c)\}$ obtained by one chase step of τ on $S(a)$. Figure 14.1a represents I in solid black, using edges to highlight equalities between elements.



(a) Example 14.1: an FD violation (b) Example 14.2.2: a way to avoid the FD violation

Figure 14.1: Example of a higher-arity FD violation, and the proposed solution

We can perform a fact-thrifty chase step of τ' on z to create $R(z, b, c)$, reusing (b, c) at $\mathsf{NDng}(R^1) = \{R^2, R^3\}$; this is illustrated in dashed red in Figure 14.1a. However, the two R-facts would then be a violation of φ, as shown by the patterns of equalities and inequalities illustrated as thick red edges.

The goal of this chapter is to define a new version of thrifty chase steps that preserves Σ_{FD} rather than just Σ_{UFD}; we call them *envelope-thrifty chase steps*. We first describe the new saturation process designed for them, which is much more complex because we need to saturate *sufficiently* with respect to the completion process that we do next. To saturate, we use a separate combinatorial result, of possible independent interest: Theorem 14.1.9, proved in Section 14.3. Second, we redefine the completion process of the previous chapter for this new notion of chase step, and use this new completion process to prove Theorem 14.1.

14.1 Envelopes and Envelope-Saturation

We start by defining our new notion of saturated instances. Recall the notions of fact classes (Definition 12.2.1) and thrifty chase steps (Definition 11.3.2). When a fact-thrifty chase step creates a fact F_{n} whose chase witness F_{w} has fact class $(R^p, \boldsymbol{C})$, we need elements to reuse in F_{n} at positions of $\mathsf{NDng}(R^p)$, which need to already occur at the positions where we reuse them. Further, the reused elements must have sim-images of the right class.

Fact-thrifty chase steps reuse a tuple of elements from one fact F_{r}, and thus apply to *fact-saturated instances*, where each fact class D which is achieved in the chase is also achieved by some fact (recall Definitions 12.2.1 and 12.2.3). Our new notion of envelope-thrifty chase steps will consider *multiple* tuples that achieve each class D, that we call an *envelope* for D; with the difference, however, that not all *tuples* need to actually occur in an achiever fact in the instance, though each *individual* element needs to occur in some achiever fact. Formally:

Definition 14.1.1. Consider $D = (R^p, \boldsymbol{C})$ in $\mathsf{AFactCl}$, and write $O := \mathsf{NDng}(R^p)$. Remember that O is then non-empty. An *envelope* E for D and for an aligned superinstance $J = (I, \mathsf{sim})$ of I_0 is a non-empty set of $|O|$-tuples indexed by O, with domain $\text{dom}(I)$, such that:

1. for every FD $\varphi : R^L \to R^r$ of Σ_{FD} with $R^L \subseteq O$ and $R^r \in O$, for any $\boldsymbol{t}, \boldsymbol{t}' \in E$, $\pi_{R^L}(\boldsymbol{t}) = \pi_{R^L}(\boldsymbol{t}')$ implies $t_r = t'_r$;
2. for every FD $\varphi : R^L \to R^r$ of Σ_{FD} with $R^L \subseteq O$ and $R^r \notin O$, for all $\boldsymbol{t}, \boldsymbol{t}' \in E$, $\pi_{R^L}(\boldsymbol{t}) = \pi_{R^L}(\boldsymbol{t}')$ implies $\boldsymbol{t} = \boldsymbol{t}'$;
3. for every $a \in \text{dom}(E)$, there is exactly one position $R^q \in O$ such that $a \in \pi_{R^q}(E)$, and then we also have $a \in \pi_{R^q}(J)$;
4. for any fact $F = R(\boldsymbol{a})$ of J and $R^q \in O$, if $a_q \in \pi_{R^q}(E)$, then F achieves D in J and $\pi_O(\boldsymbol{a}) \in E$. ◁

Intuitively, the tuples in the envelope E satisfy the FDs of Σ_{FD} within $\mathsf{NDng}(R^p)$ (condition 1), and never overlap on positions that determine a position out of $\mathsf{NDng}(R^p)$ (condition 2). Further, their elements already occur at the position where

they will be reused, and we require for simplicity that there is exactly one such position (condition 3). Last, the elements have the right sim-image for the fact class D, and for simplicity, whenever a fact reuses an envelope element, we require that it reuses a whole envelope tuple (condition 4).

We then extend this definition across all achieved fact classes in the natural way:

Definition 14.1.2. A *global envelope* $\mathcal{E}$ for an aligned superinstance $J = (I, \mathsf{sim})$ of I_0 is a mapping from each $D \in \mathsf{AFactCl}$ to an envelope $\mathcal{E}(D)$ for D and J, such that the envelopes have pairwise disjoint domains. ◁

It is not difficult to see that an aligned superinstance with a global envelope must be fact-saturated, as for each $D \in \mathsf{AFactCl}$, the envelope $\mathcal{E}(D)$ is a non-empty set of non-empty tuples, and any element of this tuple must occur in a fact that achieves D, by conditions 3 and 4. However, the point of envelopes is that they can contain more than a single tuple, so we have multiple choices of elements to reuse.

For some fact classes $(R^p, \boldsymbol{C})$ it is not useful for envelopes to contain more than one tuple. This is the case if the position R^p is *safe*, meaning that no FD from positions in $O := \mathsf{NDng}(R^p)$ determines a position outside of O. (Notice that by definition of $\mathsf{NDng}(R^p)$, such an FD could never be a UFD.) Formally:

Definition 14.1.3. We call $R^p \in \mathsf{Pos}(\sigma)$ *safe* if there is no FD $R^L \to R^r$ in Σ_{FD} with $R^L \subseteq \mathsf{NDng}(R^p)$ and $R^r \notin \mathsf{NDng}(R^p)$. Otherwise, R^p is *unsafe*.

We accordingly call a fact class $(R^p, \boldsymbol{C}) \in \mathsf{AFactCl}$ *safe* or *unsafe* depending on R^p. Observe that the second condition of Definition 14.1.1 is trivial for envelopes on safe fact classes. ◁

It is not hard to see that when we apply a fact-thrifty (or even relation-thrifty) chase step, and the exported position of the new fact is safe, then the problem illustrated by Example 14.1 cannot arise. In fact, one could show that fact-thrifty or relation-thrifty chase steps cannot introduce FD violations in this case. Because of this, in envelopes for *safe* fact classes, we do not need more than one tuple, which we can reuse as we did with fact-thrifty chase steps.

For unsafe fact classes, however, it will be important to have more tuples, and to *never reuse the same tuple twice*. This motivates our definition of the *remaining tuples* of an envelope, depending on whether the fact class is safe or not; and the definition of *envelope-saturation*, which depends on the number of remaining tuples:

Definition 14.1.4. Letting E be an envelope for $(R^p, \boldsymbol{C}) \in \mathsf{AFactCl}$ and J be an aligned superinstance, the *remaining tuples* of E are $E \backslash \pi_{\mathsf{NDng}(R^p)}(J)$ if $(R^p, \boldsymbol{C})$ is unsafe, and just E if it is safe.

We call J *n-envelope-saturated* if it has a global envelope $\mathcal{E}$ such that $\mathcal{E}(D)$ has $\geqslant n$ remaining tuples for all unsafe $D \in \mathsf{AFactCl}$. J is *envelope-saturated* if it is n-envelope-saturated for some $n > 0$. ◁

In the rest of the section, inspired by the fact-saturation lemma, we will show that we can construct envelope-saturated solutions. However, there are some complications when doing so. First, we must show that we can construct *sufficiently* envelope-saturated solutions, i.e., instances with sufficiently many remaining tuples. To do this, we will need multiple copies of the chase, which explains the technical switch from I_0 to I_0' in the statement of the next result. Second, for reasons that will become clear later in this chapter, we need to ensure that the envelopes are large *relative to the resulting instance size*. This makes the result substantially harder to show.

Proposition 14.1.5 (Sufficiently envelope-saturated solutions)**.** *For any $K \in \mathbb{N}$ and instance I_0, we can construct an instance I'_0 formed of disjoint copies of I_0, and an aligned superinstance J of I'_0 that satisfies Σ_{FD} and is $(K \cdot |J|)$-envelope-saturated.*

We prove the proposition in the rest of the section. It is not hard to see that I'_0 and J can be constructed separately for each fact class in $\mathsf{AFactCl}$, and that this is difficult only for unsafe classes. In other words, the crux of the matter is to prove the following:

Lemma 14.1.6 (Single envelope)**.** *For any unsafe class D in $\mathsf{AFactCl}$, instance I_0 and constant factor $K \in \mathbb{N}$, there exists $N_0 \in \mathbb{N}$ such that, for any $N \geqslant N_0$, we can construct an instance I'_0 formed of disjoint copies of I_0, and an aligned superinstance $J = (I, \mathsf{sim})$ of I'_0 that satisfies Σ_{FD}, with an envelope E for D of size $\geqslant K \cdot N$, such that $|J| \leqslant N$.*

Indeed, let us prove Proposition 14.1.5 with this lemma, and we will prove the lemma afterwards:

Proof of Proposition 14.1.5. Fix the constant $K \in \mathbb{N}$ and the initial instance I_0, and let us build I'_0 and the aligned superinstance $J = (I, \mathsf{sim})$ of I'_0 that has a global envelope $\mathcal{E}$. As $\mathsf{AFactCl}$ is finite, we build one J_D per $D \in \mathsf{AFactCl}$ with an envelope E_D for the class D, and we will define $J := \bigsqcup_{D \in \mathsf{AFactCl}} J_D$ and define $\mathcal{E}$ by $\mathcal{E}(D) := E_D$ for all $D \in \mathsf{AFactCl}$. When $D = (R^p, \boldsymbol{C})$ is safe, we proceed as in the proof of the Fact-Saturated Solutions Lemma: take a single copy J_D of the truncated chase to achieve the class D, and take as the only fact of the envelope E_D the projection to $\mathsf{NDng}(R^p)$ of an achiever of D in J_D. When D is unsafe, we use the Single Envelope Lemma to obtain J_D and the envelope E_D. As $\mathsf{AFactCl}$ is finite and its size does not depend on I_0, we can ensure that that $|E_D| \geqslant (K+1) \cdot |J|$ for all unsafe $D \in \mathsf{AFactCl}$ by using the Single Envelope Lemma with $K' := (K+1) \cdot |\mathsf{AFactCl}|$, and taking $N \in \mathbb{N}$ which is larger than the largest N_0 of that lemma across all $D \in \mathsf{AFactCl}$. Indeed, the resulting model J then ensures that $|J| \leqslant |\mathsf{AFactCl}| \cdot N$ and $|E_D| \geqslant (K+1) \cdot |\mathsf{AFactCl}| \cdot N$.

We now check that the resulting J and E satisfy the conditions. Each J_D is an aligned superinstance of an instance $(I'_0)_D$ which is formed of disjoint copies of I_0 (for unsafe classes) or which is exactly I_0 (for safe classes), so J is an aligned superinstance of $I'_0 := \bigsqcup_{D \in \mathsf{AFactCl}} (I'_0)_D$, so I'_0 is also a union of disjoint copies of I_0. There are no violations of Σ_{FD} in J because there are none in any of the J_D. The disjointness of domains of envelopes in the global envelope $\mathcal{E}$ is because the J_D are disjoint. It is easy to see that J is $(K \cdot |I|)$-envelope-saturated, because $|\mathcal{E}(D)| \geqslant (K+1) \cdot |I|$ for all unsafe $D \in \mathsf{AFactCl}$, so the number of remaining facts of each envelope for an unsafe class is $\geqslant K \cdot |I|$ because every fact of I eliminates at most one fact in each envelope. Hence, the proposition is proven. □

So the only thing left to do is to prove the Single Envelope Lemma. Let us accordingly fix the unsafe class $D = (R^p, \boldsymbol{C})$ in $\mathsf{AFactCl}$. We will need to study more precisely the FDs implied by the definition of an envelope for D (Definition 14.1.1). We first introduce notation for them:

Definition 14.1.7. Given a set Σ_{FD} of FDs on a relation R and $O \subseteq \mathsf{Pos}(R)$, the *FD projection* Σ^O_{FD} of Σ_{FD} to O consists of the following FDs, which we close under implication:

1. the FDs $R^L \to R^r$ of Σ_{FD} such that $R^L \subseteq O$ and $R^r \in O$;
2. for every FD $R^L \to R^r$ of Σ_{FD} where $R^L \subseteq O$ and $R^r \notin O$, the key dependency $R^L \to O$. ◁

We will need to show that, as R^p is unsafe, Σ^O_{FD} cannot have a *unary key* in O, namely, there cannot be $R^q \in O$ such that, for every $R^r \in O$, either $R^q = R^r$ or the UFD $R^q \to R^r$ holds in Σ^O_{FD}. We show the contrapositive of this statement:

Lemma 14.1.8. *For any $R^p \in \mathsf{Pos}(\sigma)$, letting $O := \mathsf{NDng}(R^p)$, if O has a unary key in Σ^O_{FD}, then R^p is safe.*

Proof. Fix $R^p \in \mathsf{Pos}(\sigma)$ and let $O := \mathsf{NDng}(R^p)$. We first show that if O has a unary key $R^s \in O$ in the original FDs Σ_{FD}, then R^p is safe. Indeed, assume the existence of such an $R^s \in O$. Assume by way of contradiction that R^p is not safe, so there is an FD $R^L \to R^r$ in Σ_{FD} with $R^L \subseteq O$ and $R^r \notin O$. Then, as Σ_{FD} is closed under the transitivity rule, the UFD $\varphi : R^s \to R^r$ is in Σ_{UFD}. Now, as $R^r \notin O$, either $R^r = R^p$ or $R^r \in \mathsf{Dng}(R^p)$; in both cases, φ witnesses that $R^s \in \mathsf{Dng}(R^p)$, but we had $R^s \in O$, a contradiction.

We must now show that if O has a unary key in O according to Σ^O_{FD}, then O has a unary key in O according to Σ_{FD}. It suffices to show that for any two positions $R^q, R^s \in O$, if the UFD $\varphi' : R^q \to R^s$ holds in Σ^O_{FD} then it also does in Σ_{FD}. Hence, fix R^q in O, and consider the set S of positions in O that R^q determines according to Σ^O_{FD}. Let Φ be the FDs in the list given in Definition 14.1.7, so that Σ^O_{FD} is the result of closing Φ under FD implication. We can compute S using the well-known "attribute closure algorithm" [Abiteboul, Hull, and Vianu 1995], which starts at $S = \{R^q\}$ and iterates the following operation: whenever there is $\varphi : R^L \to R^r$ such that $R^L \subseteq S$, add R^r to S.

Assume now that there is a position R^s in S such that $\varphi' : R^q \to R^s$ does not hold in Σ_{FD}. This implies that, when computing S, we must have used some FD $R^L \to R^t$ from a key dependency κ in Φ, as they are the only FDs of Φ which are not in Σ_{FD}. The first time we did this, we had derived, using only FDs from Σ_{FD}, that $R^L \subseteq S$, so that, in Σ_{FD}, the key dependency $R^q \to R^L$ holds. Now, κ witnesses that there is an FD $R^L \to R^r$ in Σ_{FD} with $R^r \notin O$, so that, as Σ_{FD} is closed under implication, we deduce that $R^q \to R^r$ holds in Σ_{FD} with R^q in O and $R^r \notin O$. As before, this contradicts the definition of $O := \mathsf{NDng}(R^p)$. So indeed, there is no such R^s in S.

Hence, if O has a unary key in O according to Σ^O_{FD}, then it also does according to Σ_{FD}, and then, by the reasoning of the first paragraph, R^p is safe, which is the desired claim. □

We now know that O has no unary key in Σ^O_{FD}. This allows us to introduce the crucial tool needed to prove the Sufficiently Envelope-Saturated Solutions Proposition. It is the following independent result, which is proved separately in Section 14.3 using a combinatorial construction.

Theorem 14.1.9 (Dense interpretations)**.** *For any set Σ_{FD} of FDs over a relation R with no unary key, for all $K \in \mathbb{N}$, there exists $N_0 \in \mathbb{N}$ such that for all $N \geqslant N_0$, we can construct a non-empty instance I of R that satisfies Σ_{FD} and such that $|\text{dom}(I)| \leqslant N$ and $|I| \geqslant K \cdot N$.*

Further, we can impose a ***disjointness condition*** *on the result I: we can ensure that for all $a \in \mathrm{dom}(I)$, there exists exactly one $R^p \in \mathsf{Pos}(R)$ such that $a \in \pi_{R^p}(I)$.*

We can now prove the Single Envelope Lemma and conclude the section. Choose a fact $F_{\mathrm{ach}} = R(\boldsymbol{b})$ of $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}}) \backslash I_0$ that achieves the fact class D, and let I_1 be obtained from I_0 by applying UID chase steps on I_0 to obtain a finite truncation of $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$ that includes F_{ach} but no child fact of F_{ach}. Consider the aligned superinstance $J_1 = (I_1, \mathsf{sim}_1)$ of I_0, where sim_1 is the identity.

Remember that we wrote $D = (R^p, \boldsymbol{C})$, and $O = \mathsf{NDng}(R^p)$, which is non-empty. Define a $|O|$-ary relation $R_{|O}$ (with positions indexed by O for convenience), define Σ^O_{FD} as in Definition 14.1.7, and consider Σ^O_{FD} as FDs on $R_{|O}$. Because D is unsafe, by Lemma 14.1.8, $R_{|O}$ has no unary key in Σ^O_{FD}. Letting $K \in \mathbb{N}$ be our target constant for the Single Envelope Lemma, apply the Dense Interpretations Theorem (Theorem 14.1.9) to $R_{|O}$ and Σ^O_{FD}, taking $K' := 2 \cdot K \cdot |J_1|$ as the constant. Define $N_0 \in \mathbb{N}$ for the Single Envelope Lemma as $2 \cdot \max(|J_1|, 1) \cdot \max(N'_0, 1)$ where N'_0 is obtained from the Dense Interpretations Theorem for K'. Letting $N' \in \mathbb{N}$ be our target size for the Single Envelope Lemma, using $N := \lfloor N' / |J_1| \rfloor$ as the target size for the Dense Interpretations Theorem (which is $\geqslant N'_0$), we can build an instance I_{dense} of $R_{|O}$ that satisfies Σ^O_{FD} and such that $|I_{\mathrm{dense}}| \geqslant N \cdot K'$ and $|\mathrm{dom}(I_{\mathrm{dense}})| \leqslant N$.

Let $I'_{\mathrm{dense}} \subseteq I_{\mathrm{dense}}$ be a subinstance of size exactly N of I_{dense} such that we have $\mathrm{dom}(I'_{\mathrm{dense}}) = \mathrm{dom}(I_{\mathrm{dense}})$, that is, such that each element of $\mathrm{dom}(I_{\mathrm{dense}})$ occurs in some fact of I'_{dense}: we can clearly construct I'_{dense} by picking, for each element of $\mathrm{dom}(I_{\mathrm{dense}})$, one fact of I_{dense} where it occurs, removing duplicate facts, and completing with other arbitrary facts of I_{dense} so the number of facts is exactly N. Number the facts of I'_{dense} as $F'_1, \ldots, F'_N$.

Let us now create $N - 1$ disjoint copies of J_1, numbered J_2 to J_N. Let I_{pre} be the disjoint union of the underlying instances of the J_i, let I'_0 be formed of the N disjoint copies of I_0 in I_{pre}, and define a mapping $\mathsf{sim}_{\mathrm{pre}}$ from $\mathrm{dom}(I_{\mathrm{pre}})$ to $\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}})$ following the sim_i in the expected way. It is clear that J_{pre} is an aligned superinstance of I'_0. For $1 \leqslant i \leqslant N$, we call $F_i = R(\boldsymbol{a^i})$ *the fact of I_i that corresponds to the achiever F_{ach} in $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$*. In particular, for all $1 \leqslant i \leqslant N$, we have that $\mathsf{sim}(a^i_j) = b_j$ for all j, and a^i_p is the only element of F_i that also occurs in other facts of J_i, as J_i does not contain any descendent fact of F_i.

Intuitively, we will now identify elements in J_{pre} so that the restriction of the F_i to O is exactly the F'_i, and this will allow us to use the instance I_{dense} to define the envelope. Formally, as the a^i_j are pairwise distinct, we can define the function f that maps each a^i_j, for $1 \leqslant i \leqslant N$ and $R^j \in O$, to $\pi_{R^j}(F'_i)$. In other words, f is a surjective (but generally not injective) mapping, the domain of f is the projection to O of the F_i in I_i, the range of f is $\mathrm{dom}(I'_{\mathrm{dense}})$, and f maps each element of the projection to the corresponding element in F'_i. Extend f to a mapping f' with domain $\mathrm{dom}(I_{\mathrm{pre}})$ by setting $f'(a) := f(a)$ when a is in the domain of f, and $f'(a) := a$ otherwise. Now, let $I := f'(I_{\mathrm{pre}})$. In other words, I is I_{pre} except that elements in the projection to O of the facts F_i are renamed, and some are identified, so that the projection to O of $\{f'(F_i) \mid 1 \leqslant i \leqslant N\}$, seen as an instance of $R_{|O}$-facts, is exactly I'_{dense}. Because a^i_j occurs only in F_i for all $R^j \neq R^p$, nd $R^p \notin O$, this means that the elements identified by f' only occurred in the F_i in I_{pre}.

We now build $J = (I, \mathsf{sim})$ obtained by defining sim from $\mathsf{sim}_{\mathrm{pre}}$ as follows: if a is in the domain of f, then $\mathsf{sim}(a) := \mathsf{sim}_{\mathrm{pre}}(a')$ for any preimage of a' by f' (as we will

see, the choice of preimage does not matter), and if a is not in the domain of f, then $\mathsf{sim}(a) := \mathsf{sim}_{\mathrm{pre}}(a)$ because a is then the only preimage of a by f'. We have now defined the instance I'_0 formed of disjoint copies of I_0 and the final J, and we define $E := I_{\mathrm{dense}}$. We must now show that J is indeed an aligned superinstance of I'_0, and that E is an envelope for I and D, and that they satisfy the required conditions.

We note that it is immediate that $J = (I, \mathsf{sim})$ is a superinstance of I'_0. Indeed, we have $I := f(I_{\mathrm{pre}})$, and I_{pre} was a superinstance of I'_0, so it suffices to note that $\mathrm{dom}(I'_0)$ is not in the domain of f: this is because the achiever F_{ach} is not a fact of I_0, so the domain of f, namely, the projection of the F_i on O, does not intersect $\mathrm{dom}(I'_0)$. Further, it is clear that J is finite and has $N \cdot |J_1|$ facts, because this is the case of J_{pre} by definition, and f' cannot have caused any facts of J_{pre} to be identified in J, because we have $R^p \notin O$, so the projection of each F_i to R^p is a different element which is mapped to itself by f'. Hence, we have $|J| = N \cdot |J_1| \leqslant N'$. Further, we have $|E| = |I_{\mathrm{dense}}| \geqslant N \cdot K' \geqslant \lfloor N' / |J_1| \rfloor \cdot 2 \cdot K \cdot |J_1|$, and as $N' \geqslant N_0 \geqslant 2 \cdot |J_1|$ we have $\lfloor N' / |J_1| \rfloor \geqslant (1/2) \cdot (N' / |J_1|)$. Hence, $|E| \geqslant K \cdot N'$, so we have achieved the required size bound.

We now show that J is indeed an aligned superinstance of I'_0. The technical conditions on sim are clearly respected, because they were respected on J_{pre}, because f' only identifies elements in $I_{\mathrm{pre}} \backslash I_0$, and because the identified elements occur at the same positions as their preimages so the directionality condition is respected.

We show that sim is a k-bounded simulation from J to $\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}})$ by showing the stronger claim that it is actually a k'-bounded simulation for all $k' \in \mathbb{N}$, which we show by induction on k'. The case of $k' = 0$ is trivial. The induction case is trivial for all facts except for the $f'(F_i)$, because the a^i_j only occurred in I_{pre} in the facts F_i, by our assumption that the F_i have no children in the I_i, and because the exported position of F_{ach} is $R^p \notin O$. Hence, consider a fact $F' = R(\boldsymbol{c})$ of I which is the image by f' of some fact F_i. Choose $1 \leqslant p \leqslant |R|$. We wish to show that there exists a fact $F'' = R(\boldsymbol{d})$ of $\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}})$ such that $\mathsf{sim}(c_p) = d_p$ and for all $1 \leqslant q \leqslant |R|$ we have $(I, c_q) \leqslant_{k'-1} (\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}}), d_q)$. Let $a^{i_0}_{j_0}$ be the preimage of c_p used to define $\mathsf{sim}(c_p)$; by the disjointness condition of the Dense Interpretations Theorem, we must have $j_0 = p$. Observe that $\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}})$ is formed of disjoint copies of $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$, so, recalling the definition of J'_{i_0}, consider the fact $F'' = R(\boldsymbol{d})$ of $\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}})$ corresponding to F_{i_0} in I. By definition, $\mathsf{sim}(c_p) = \mathsf{sim}(a^{i_0}_{j_0}) = d_p$.

We now show that for all $1 \leqslant q \leqslant |R|$ we have $(I, c_q) \leqslant_{k'-1} (\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}}), d_q)$. Fix $1 \leqslant q \leqslant |R|$. It suffices to show that $\mathsf{sim}(c_q) \simeq_{k'} d_q$, as we can then use the induction hypothesis to know that $(I, c_q) \leqslant_{k'-1} (\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}}), \mathsf{sim}(c_q))$, so that by transitivity $(I, c_q) \leqslant_{k'-1} (\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}}), d_q)$. Hence, we show that $\mathsf{sim}(c_q) \simeq_{k'} d_q$. Let $a^{i'_0}_{j'_0}$ be the preimage of c_q used to define $\mathsf{sim}(c_q)$. Again we must have $j'_0 = q$ by the disjointness condition, and, considering the fact $F''' = R(\boldsymbol{e})$ of $\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}})$ corresponding to $F_{i'_0}$ in I, we have $\mathsf{sim}(c_q) = e_q$. But as both F''' and F'' are copies in $\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}})$ of the same fact F_{ach} of $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$, it is indeed the case that $d_q \simeq_{k'} e_q$. Hence, $\mathsf{sim}(c_q) \simeq_{k'} d_q$, from which we conclude that F'' is a suitable witness fact for F'. By induction, we have shown that sim is indeed a k'-bounded simulation from J to $\mathsf{Chase}(I'_0, \Sigma_{\mathrm{UID}})$ for any $k' \in \mathbb{N}$, so that it is in particular a k-bounded simulation.

We now show that J satisfies Σ_{FD}. For this, it will be convenient to define the *overlap* of two facts:

Definition 14.1.10. The *overlap* $\mathsf{OVL}(F, F')$ between two facts $F = R(\boldsymbol{a})$ and $F' = R(\boldsymbol{b})$ of the same relation R in an instance I is the subset O of $\mathsf{Pos}(R)$ such that $a_s = b_s$ iff $R^s \in O$. If $|O| > 0$, we say that F and F' *overlap*. ◁

As I_{pre} satisfies Σ_{FD} by the Unique Witness Property of the UID chase, any new violation of Σ_{FD} in I relative to I_{pre} must include some fact $F = f'(F'_{i_0})$, and some fact $F' \neq F$ that overlaps with F, so necessarily $F' = f'(F'_{i_1})$ for some i_1 by construction of I, and $\mathsf{OVL}(F, F') \subseteq O$. If $\mathsf{OVL}(F, F') = O$, then, by our definition of f and of the F'_i, this implies that $F'_{i_0} = F'_{i_1}$, a contradiction because $F \neq F'$. So the only case to consider is when $\mathsf{OVL}(F, F') \subsetneq O$, but we can also exclude this case:

Lemma 14.1.11. *Let I be an instance, Σ_{FD} be a conjunction of FDs, and $F \neq F'$ be two facts of I. Assume there is a position $R^p \in \mathsf{Pos}(\sigma)$ such that, writing $O := \mathsf{NDng}(R^p)$, we have $\mathsf{OVL}(F, F') \subsetneq O$, and that $\{\pi_O(F), \pi_O(F')\}$ is not a violation of Σ^O_{FD}. Then $\{F, F'\}$ is not a violation of Σ_{FD}.*

Proof. Assume by way of contradiction that F and F' violate an FD $\varphi : R^L \to R^r$ of Σ_{FD}, which implies that $R^L \subseteq \mathsf{OVL}(F, F') \subseteq O$ and $R^r \notin \mathsf{OVL}(F, F')$. Now, if $R^r \in O$, then φ is in Σ^O_{FD}, so that $\pi_O(F)$ and $\pi_O(F')$ violate Σ^O_{FD}, a contradiction. Hence, $R^r \in \mathsf{Pos}(R) \backslash O$, and the key dependency $\kappa : R^L \to O$ is in Σ^O_{FD}, so that $\pi_O(F)$ and $\pi_O(F')$ must satisfy κ. Thus, because $R^L \subseteq \mathsf{OVL}(F, F')$, we must have $\mathsf{OVL}(F, F') = O$, which is a contradiction because we assumed $\mathsf{OVL}(F, F') \subsetneq O$. □

Now, by by definition of I'_{dense}, we know that I'_{dense} satisfies Σ^O_{FD}, so that $\{\pi_O(F), \pi_O(F')\}$ is not a violation of Σ^O_{FD}. Thus, we can conclude with Lemma 14.1.11 that $\{F, F'\}$ is not a violation of Σ_{FD}, so that J satisfies Σ_{FD}. We have thus shown that J is an aligned superinstance of I'_0.

Last, we check that E is indeed an envelope for D and for J. Indeed, E satisfies Σ^O_{FD} by construction, so conditions 1 and 2 are respected. The first part of condition 3 is ensured by the disjointness condition, and its second part follows from our definition of I'_{dense} that ensures that any element in $\text{dom}(E)$ occurs in a fact F'_i of I'_{dense}, hence occurs in $f'(F_i)$ in J. Last, condition 4 is true because the elements of $\text{dom}(E)$ are only used in the $f'(F_i)$, and the sim-images of the $f'(F_i)$ are copies in $\mathsf{Chase}(I'_0, \Sigma_{\text{UID}})$ of the same fact F_{ach} in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ that achieves D, so the F_i are all achievers of D; further, by definition, their projection to O is a tuple of E because it is a fact of I'_{dense}.

Hence, J is indeed an aligned superinstance of a disjoint union I'_0 of copies of I_0, J satisfies Σ_{FD}, $|J| \leqslant N'$, and J has an envelope E of size $K \cdot N'$ for D. This concludes the proof of the Single Envelope Lemma, and hence of the Sufficiently Envelope-Thrifty Solutions Proposition.

14.2 Envelope-Thrifty Chase Steps

We have shown that we can construct sufficiently envelope-saturated superinstances of the input instance. The point of this notion is to introduce *envelope-thrifty chase steps*, namely, thrifty chase steps that use remaining tuples from the envelope to fill the non-dangerous positions:

Definition 14.2.1. *Envelope-thrifty chase steps* are thrifty chase steps (Definition 11.3.2) which apply to envelope-saturated aligned superinstances. Following Definitions 11.3.2 and 12.3.3, we write S^q for the exported position of the new fact F_n, we write $F_\mathrm{w} = S(\boldsymbol{b}')$ for the chase witness, and we let $D = (S^q, \boldsymbol{C}) \in \mathsf{AFactCl}$ be the fact class of F_w. Analogously to Definition 11.3.2, we define an *envelope-thrifty* chase step as follows: if $\mathsf{NDng}(S^q)$ is non-empty, choose one remaining tuple $\boldsymbol{t}$ of $\mathcal{E}(D)$, and set $b_r := t_r$ for all $S^r \in \mathsf{NDng}(S^q)$.

We define a *fresh envelope-thrifty step* in the same way as a fresh fact-thrifty step: all elements at dangerous positions are fresh elements only occurring at that position. ◁

Example 14.2.2. Recall I_0, τ, τ' and φ from Example 14.1. Now, consider $I_0' := \{S(a), T(z), S(a'), S(z')\}$ formed of two copies of I_0, and $I' := I_0' \sqcup \{R(a,b,c), R(a',b',c')\}$ obtained by two chase steps: this is illustrated in solid black in Figure 14.1b on page 185. The two facts $R(a,b,c)$ and $R(a',b',c')$ would achieve the same fact class D, so we can define $E(D) := \{(b,c), (b',c'), (b',c), (b,c')\}$.

We can now satisfy Σ_UID on I' without violating φ, with two envelope-thrifty chase steps that reuse the remaining tuples (b',c) and (b,c') of $E(D)$: the new facts and the pattern of equalities between them is illustrated in red in Figure 14.1b.

Recall that fact-thrifty chase steps apply to fact-saturated aligned superinstances (Lemma 12.3.4). Similarly, envelope-thrifty chase steps apply to envelope-saturated aligned superinstances:

Lemma 14.2.3 (Envelope-thrifty applicability)**.** *For any envelope-saturated superinstance I of an instance I_0, UID $\tau : R^p \subseteq S^q$ and element $a \in \mathsf{Wants}(I, \tau)$, we can apply an envelope-thrifty chase step on a with τ to satisfy this violation.*

Further, for any new fact $S(\boldsymbol{e})$ that we can create by chasing on a with τ with a fact-thrifty chase step, we can instead apply an envelope-thrifty chase step on a with τ to create a fact $S(\boldsymbol{b})$ with $b_r = e_r$ for all $S^r \in \mathsf{Pos}(S) \backslash \mathsf{NDng}(S^r)$.

Proof. For the first part of the claim, as in the proof of the Fact-Thrifty Applicability Lemma (Lemma 12.3.4), there is nothing to show unless $\mathsf{NDng}(S^q)$ is non-empty, and the fact class $D = (S^q, \boldsymbol{C})$ is then in $\mathsf{AFactCl}$, where $\boldsymbol{C}$ is the tuple of the $\simeq_k$-equivalence classes of the elements of the chase witness F_w. Hence, as J is envelope-saturated, it has some remaining tuple for the class D that we can use to define the non-dangerous positions of the new fact.

For the second part, again as in the proof of the Fact-Thrifty Applicability Lemma, observe that the definition of envelope-thrifty chase steps only poses additional conditions (relative to thrifty chase steps) on $\mathsf{NDng}(S^q)$, so that, for any fact that we would create with a fact-thrifty chase step, we can change the elements at $\mathsf{NDng}(S^q)$ to perform an envelope-thrifty chase step, using the fact that I is envelope-saturated. □

Further, recall that we showed that relation-thrifty chase steps never violate Σ_UFD (Lemma 11.3.5). We now show that envelope-thrifty chase steps never violate Σ_FD, which is their intended purpose:

Lemma 14.2.4 (Envelope-thrifty FD preservation)**.** *For any n-envelope-saturated aligned superinstance J that satisfies Σ_FD, the result of an envelope-thrifty chase step on J satisfies Σ_FD.*

Proof. Fix J and its global envelope $\mathcal{E}$. Let $F_\mathrm{n} = S(\boldsymbol{b})$ be the fact created by the envelope-thrifty step, let $\tau : R^p \subseteq S^q$ be the UID, let $J' = (I', \mathsf{sim}')$ be the result of the chase step, let F_w be the chase witness, and let D be the fact class of F_w. Write $O := \mathsf{NDng}(S^q)$. Assume by contradiction that $I' \not\models \Sigma_\mathrm{FD}$; as $I \models \Sigma_\mathrm{FD}$, any violation of Σ_FD in I' must be between the new fact F_n and an existing fact $F = S(\boldsymbol{c})$ of I. Recalling the definition of overlaps (Definition 14.1.10), note that we only have $b_r \in \pi_{S^r}(I)$ for $S^r \in O$ by definition of thrifty chase steps, so we must have $\mathsf{OVL}(F_\mathrm{n}, F) \subseteq O$. Now, as $\pi_O(F_\mathrm{n})$ was defined using elements of $\mathrm{dom}(\mathcal{E}(D))$, taking any $S^r \in \mathsf{OVL}(F_\mathrm{n}, F) \subseteq O$ (which is non-empty by definition of an FD violation), we have $c_r = b_r \in \pi_{S^r}(\mathcal{E}(D))$, so that, by condition 4 of the definition of the envelope $\mathcal{E}(D)$, we know that $\pi_O(\boldsymbol{c})$ is a tuple $\boldsymbol{t}'$ of $\mathcal{E}(D)$. Now, either $\mathsf{OVL}(F_\mathrm{n}, F) \subsetneq O$ or $\mathsf{OVL}(F_\mathrm{n}, F) = O$.

In the first case, we observe that, by conditions 1 and 2 of the definition of the envelope $\mathcal{E}(D)$, we know that $\{\pi_O(\boldsymbol{c}), \pi_O(\boldsymbol{b})\}$ is not a violation of Σ^O_FD. Together with the fact that $\mathsf{OVL}(F_\mathrm{n}, F) \subsetneq O$, this allows us to apply Lemma 14.1.11 and deduce that $\{F, F_\mathrm{n}\}$ actually does not violate Σ_FD, a contradiction.

In the second case, where $\mathsf{OVL}(F_\mathrm{n}, F) = O$, we have $\boldsymbol{t} = \boldsymbol{t}'$. Now, either D is safe or D is unsafe. If D is unsafe, we have a contradiction because F witnesses that $\boldsymbol{t}$ was not a remaining tuple of $\mathcal{E}(D)$, so we cannot have used $\boldsymbol{t}$ to define F_n. If D is safe, then by definition there is no FD $R^L \to R^r$ of Σ_FD with $R^L \subseteq O$ and $R^r \notin O$. Now, as $\mathsf{OVL}(F_\mathrm{n}, F) = O$, it is clear that F and F_n cannot violate any FD of Σ_FD, a contradiction again. □

Last, recall that we showed that fresh fact-thrifty steps preserve the property of being aligned (Lemma 12.3.5) and that non-fresh fact-thrifty steps also do when we additionally assume k-essentiality, which they also preserve (Lemma 12.4.7). We now prove the analogous claims for envelope-thrifty steps assuming envelope-saturation. The only difference is that envelope-thrifty chase steps make envelope-saturation *decrease*, unlike fact-thrifty steps which always preserved fact-saturation:

Lemma 14.2.5 (Envelope-thrifty preservation)**.** *For any $n \in \mathbb{N}$, for any n-envelope-saturated aligned superinstance J of I_0, the result J' of a fresh envelope-thrifty chase step on J is an $(n-1)$-envelope-saturated aligned superinstance of I_0. Further, if J is k-essential, the claim holds even for non-fresh envelope-thrifty chase steps, and the result J' is additionally k-essential.*

Proof. We reuse notation from Lemma 14.2.4: considering an application of an envelope-thrifty chase step: let $J = (I, \mathsf{sim})$ be the aligned superinstance of I_0, let $\tau : R^p \subseteq S^q$ be the UID, write $O := \mathsf{NDng}(S^q)$, let $F_\mathrm{w} = S(\boldsymbol{b}')$ the chase witness, let $D = (S^q, \boldsymbol{C})$ be the fact class, let $F_\mathrm{n} = S(\boldsymbol{b})$ be the new fact to be created, and let $\boldsymbol{t}$ be the remaining tuple of $\mathcal{E}(D)$ used to define F_n, and let $J' = (I', \mathsf{sim}')$ be the result.

We now prove that J' is still an aligned superinstance. We first adapt the Fresh Fact-Thrifty Preservation Lemma (Lemma 12.3.5) to work with envelope-thrifty chase steps. We can no longer use Lemma 11.3.5 to prove that $J' \models \Sigma_\mathrm{UFD}$, but we have shown already that $J' \models \Sigma_\mathrm{FD}$ in Lemma 14.2.4, so this point is already covered. The only other point specific to fact-thriftiness is proving that sim' is still a k-bounded simulation, but it actually only relies on the fact that $\mathsf{sim}'(b_r) \simeq_k b'_r$ in $\mathsf{Chase}(I_0, \Sigma_\mathrm{UID})$ for all $S^r \in \mathsf{NDng}(S^q)$, which is still ensured by envelope-thrifty

chase steps: by conditions 3 and 4 of the definition of envelopes, we know that, for any $S^r \in \mathsf{NDng}(S^q)$, the element t_r already occurs at position S^r in a fact of I that achieves D, so that $\mathsf{sim}(t_r) \simeq_k b'_r$.

Second, we adapt the Fact-Thrifty Preservation Lemma (Lemma 12.4.7) to envelope-thrifty chase steps. Again, the only condition of fact-thrifty chase steps used when proving that lemma is that $\mathsf{sim}'(b_r) \simeq_k b'_r$ in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ for all $S^r \in \mathsf{NDng}(S^q)$, which is still true. Hence, having adapted these two lemmas, we conclude that J' has the required properties.

We now prove that $\mathcal{E}$ is still a global envelope of J' after performing an envelope-thrifty chase step. The condition on the disjointness of the envelope domains only concerns $\mathcal{E}$ itself, which is unchanged. Hence, we need only show that, for any $D' \in \mathsf{AFactCl}$, $\mathcal{E}(D')$ is still an envelope. All conditions of the definition of envelopes except condition 4 are clearly true, because they were true in J, and they only depend only on $\mathcal{E}(D')$ or they are preserved when creating more facts. We now check condition 4, which only needs to be verified on the new fact F_{n}.

Consider $S^u \in \mathsf{Pos}(S)$ and $S^t \in \mathsf{NDng}(S^u)$, and assume that $b_t \in \pi_{S^t}(\mathcal{E}(D'))$. As $\mathcal{E}(D)$ is an envelope for J, by condition 3 of the definition, we have $b_t \in \pi_{S^t}(I)$ as well, so that, by definition of thrifty chase steps, we must have $S^t \in O$. Now, as the envelopes of $\mathcal{E}$ are pairwise disjoint, and as the b_r for $S^r \in O$ are all in $\mathrm{dom}(\mathcal{E}(D))$, we must have $D = D'$, and $\boldsymbol{t}$ witnesses that $\pi_O(\boldsymbol{b}) \in \mathcal{E}(D)$. Hence $\mathcal{E}$ is still a global envelope of J'.

Last, to see that the resulting J' is $(n-1)$-envelope-saturated, it suffices to observe that, for each unsafe class $D \in \mathsf{AFactCl}$, the remaining tuples of $\mathcal{E}(D)$ for J' are those of $\mathcal{E}(D)$ for J minus at most one tuple (namely, some projection of F_{n}). This concludes the proof. □

Hence, we know that envelope-thrifty chase steps preserve being aligned and also preserves Σ_{FD} (rather than Σ_{UFD} for fact-thrifty chase steps). Our goal is then to modify the Fact-Thrifty Completion Proposition of the previous chapter (Proposition 13.2.2) to use envelope-thrifty rather than fact-thrifty chase steps, relying on the previous lemmas to preserve all invariants. The problem is that unlike fact-saturation, envelope-saturation "runs out"; whenever we use a remaining tuple $\boldsymbol{t}$ in a chase step to create F_{n} and obtain a new aligned superinstance J', then we can no longer use the same $\boldsymbol{t}$ in J'. This is why the result of an envelope-thrifty chase step is less saturated than its input, and it is why we made sure in the Sufficiently Envelope-Saturated Solutions Proposition that we could construct arbitrarily saturated superinstances.

For this reason, before we modify the Fact-Thrifty Completion Proposition, we need to account for the number of chase steps that the proposition performs. We show that it is linear in the size of the input instance.

Lemma 14.2.6 (Accounting)**.** *There exists $B \in \mathbb{N}$ depending only on σ, k, and Σ_{U}, such that, for any aligned superinstance $J = (I, \mathsf{sim})$ of I_0, letting L be the preserving fact-thrifty sequence constructed in the Fact-Thrifty Completion Proposition, we have $|L| < B \cdot |I|$.*

Proof. It suffices to show that $|L(J)| < B \cdot |I|$, because, as each chase step creates one fact, we have $|L| \leqslant |L(J)|$.

Remember that the fact-thrifty completion process starts by constructing an ordered partition $\boldsymbol{P} = (P_1, \ldots, P_n)$ of Σ_{UID} (Definition 13.1.1). This $\boldsymbol{P}$ does not depend on I. Hence, as we satisfy the UIDs of each P_i in turn, if we can show that the instance size only increases by a multiplicative constant for each class, then the blow-up for the entire process is by a multiplicative constant (obtained as the product of the constants for each P_i).

For trivial classes, we apply one chase round by fresh fact-thrifty chase steps (Lemma 13.2.4), It is easy to see that applying a chase round by any form of thrifty chase step on an aligned superinstance $J_1 = (I_1, \mathsf{sim}_1)$ yield a result whose size has only increased relative to J_1 by a multiplicative constant. This is because $|\text{dom}(I_1)| \leqslant |\sigma| \cdot |I_1|$, and the number of facts created per element of I_1 in a chase round is at most $|\mathsf{Pos}(\sigma)|$. Hence, for trivial classes, we only incur a blowup by a constant multiplicative factor.

For non-trivial classes, we apply the Reversible Fact-Thrifty Completion Proposition (Proposition 12.4.8). Remember that this lemma first ensures k-essentiality by applying $k + 1$ fact-thrifty chase rounds (Lemma 12.4.6) and then makes the result satisfy Σ_{UID} using the sequence constructed by the Reversible Relation-Thrifty Completion Proposition (Proposition 11.4.1). Ensuring k-essentiality only implies a blow-up by a multiplicative constant, because it is performed by applying $k + 1$ fact-thrifty chase rounds, so we can use the same reasoning as for trivial classes. Hence, we focus on the Reversible Relation-Thrifty Completion Proposition, and show that it also causes only a blow-up by a multiplicative constant.

When we apply the Reversible Relation-Thrifty Completion Proposition to an instance I, we start by constructing a balanced pssinstance P using the Balancing Lemma (Lemma 10.3.2), and a Σ_{U}-compliant piecewise realization PI of P by the Realizations Lemma (Lemma 11.1.5), and we then apply fact-thrifty chase steps to satisfy Σ_{UID} following PI. We know that, whenever we apply a fact-thrifty chase step to an element a, the element a occurs after the chase step at a new position where it did not occur before. Hence, it suffices to show that $|\text{dom}(P)|$ is within a constant factor of $|I|$, because then we know that the final number of facts created by the sequence of the Reversible Relation-Thrifty Completion Proposition will be $\leqslant |\text{dom}(P)| \cdot |\mathsf{Pos}(\sigma)|$.

To show this, remember that $\text{dom}(P) = \text{dom}(I) \sqcup \mathcal{H}$, where $\mathcal{H}$ is the helper set. Hence, we only need to show that $|\mathcal{H}|$ is within a multiplicative constant factor of $|I|$. From the proof of the Balancing Lemma, we know that $\mathcal{H}$ is a disjoint union of $\leqslant |\mathsf{Pos}(\sigma)|$ sets whose size is linear in $|\text{dom}(I)|$ which is itself $\leqslant |\sigma| \cdot |I|$. Hence, the Reversible Relation-Thrifty Completion Proposition only causes a blowup by a constant factor. As we justified, this implies the same about the entire completion process, and concludes the proof. □

This allows us to deduce the minimal level of envelope-saturation required to adapt the Fact-Thrifty Completion Proposition:

Proposition 14.2.7 (Envelope-thrifty completion)**.** *Let $\Sigma = \Sigma_{\text{FD}} \wedge \Sigma_{\text{UID}}$ be finitely closed FDs and UIDs, let $B \in \mathbb{N}$ be as in the Accounting Lemma, and let I_0 be an instance that satisfies Σ_{FD}. For any $(B \cdot |J|)$-envelope-saturated aligned superinstance J of I_0 that satisfies Σ_{FD}, we can obtain by envelope-thrifty chase steps an aligned superinstance J_{f} of I_0 that satisfies Σ.*

Proof. We define envelope-thrifty sequences, and preserving envelope-thrifty sequences, analogously to (preserving) fact-thrifty sequences (Definition 12.4.9 and Definition 13.2.1) in the expected manner, but further requiring that all intermediate aligned superinstances remain envelope-saturated. This definition makes sense thanks to the Envelope-Thrifty Preservation Lemma.

By the Fact-Thrifty Completion Proposition, there exists a preserving fact-thrifty sequence L such that $L(J)$ satisfies Σ_{UID}, and $|L| < B \cdot |I|$. Construct from L an envelope-thrifty sequence L' that non-dangerously matches L, by changing each fact-thrifty chase step to an envelope-thrifty chase step, which we can do at each individual step thanks to the Envelope-Thrifty Applicability Lemma. It is clear that this is a preserving envelope-thrifty sequence, thanks to the Envelope-Thrifty Preservation Lemma, and thanks to the fact that the Ensuring Essentiality Lemma (Lemma 12.4.6) clearly adapts from fact-thrifty chase steps to envelope-thrifty chase steps: again, it only relies on the fact that, letting $F_{\text{n}} = S(\boldsymbol{b})$ be the new fact, $F_{\text{w}} = S(\boldsymbol{b'})$ the chase witness, and $\tau : R^p \subseteq S^q$ the UID, we have $\mathsf{sim'}(b_r) \simeq_k b'_r$ in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ for all $S^r \in \mathsf{NDng}(S^q)$. This also uses the fact that, by the Accounting Lemma, we have $|L| \leqslant B \cdot |I|$, so by the Envelope-Thrifty Preservation Lemma, all intermediate aligned superinstances remain envelope-saturated.

Hence, $J_{\text{f}} := L'(J)$ is an aligned superinstance of I_0. Further, by the Thrifty Sequence Rewriting Lemma (Lemma 12.4.11), as $L(J) \models \Sigma_{\text{UID}}$, so does J_{f}. Last, as $J \models \Sigma_{\text{FD}}$, by the Envelope-Thrifty FD Preservation Lemma, so does J_{f}. This concludes the proof. □

We can now conclude the proof of Theorem 14.1. Start by applying the saturation process of the Sufficiently Envelope-Saturated Solutions Proposition to obtain an aligned superinstance $J = (I, \mathsf{sim})$ of a disjoint union I'_0 of copies of I_0, such that J satisfies Σ_{FD} and is $(B \cdot |I|)$-envelope-saturated. Now, apply the Envelope-Thrifty Completion Proposition to obtain an aligned superinstance $J_{\text{f}} = (I_{\text{f}}, \mathsf{sim}_{\text{f}})$ of I'_0 that satisfies Σ. We know that I_{f} satisfies Σ and is a k-sound superinstance of I'_0 for ACQ, but clearly it is also a k-sound superinstance of I_0, as is observed by the k-bounded simulation from I' to $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ obtained by composing $\mathsf{sim'}$ with the obvious homomorphism from $\mathsf{Chase}(I'_0, \Sigma_{\text{UID}})$ to $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$. This concludes the proof.

14.3 Constructing Dense Interpretations

All that remains is to show the Dense Interpretations Theorem:

Theorem 14.1.9 (Dense interpretations)**.** *For any set Σ_{FD} of FDs over a relation R with no unary key, for all $K \in \mathbb{N}$, there exists $N_0 \in \mathbb{N}$ such that for all $N \geqslant N_0$, we can construct a non-empty instance I of R that satisfies Σ_{FD} and such that $|\text{dom}(I)| \leqslant N$ and $|I| \geqslant K \cdot N$.*

Further, we can impose a ***disjointness condition*** *on the result I: we can ensure that for all $a \in \text{dom}(I)$, there exists exactly one $R^p \in \mathsf{Pos}(R)$ such that $a \in \pi_{R^p}(I)$.*

Fix the relation R, and let Σ_{FD} be an arbitrary set of FDs which we assume is closed under FD implication. Let Σ_{UFD} be the UFDs implied by Σ_{FD}; it is also closed under FD implication. Recall the definition of OVL (Definition 14.1.10). We

introduce a notion of *tame overlaps* for Σ_{UFD}, which depends only on Σ_{UFD} but is a sufficient condition to satisfy Σ_{FD}, as we will show.

Definition 14.3.1. We say a subset $O \subseteq \mathsf{Pos}(R)$ is *tame* for Σ_{UFD} if O is empty or for every $R^p \in \mathsf{Pos}(R) \backslash O$, there exists $R^q \in \mathsf{Pos}(R)$ such that:

- for all $R^s \in O$, the UFD $R^q \to R^s$ holds in Σ_{UFD},
- the UFD $R^q \to R^p$ does not hold in Σ_{UFD}.

We say that an instance I has the *tame overlaps* property (for Σ_{UFD}) if for every $F \neq F'$ of I, $\mathsf{OVL}(F, F')$ is tame. ◁

In particular, if an instance ensures that non-empty overlaps between pairs of facts always have a unary key that determines precisely the overlap, then it has tame overlaps; we will show a refinement of this as Lemma 14.3.5. We now claim the following lemma, and its immediate corollary:

Lemma 14.3.2. *If $O \subseteq \mathsf{Pos}(R)$ is tame for Σ_{UFD} then there is no FD $\varphi : R^L \to R^r$ in Σ_{FD} such that $R^L \subseteq O$ but $R^r \notin O$.*

Proof. If O is empty the claim is immediate. Otherwise, assume to the contrary the existence of such an FD φ. As $R^r \notin O$ and O is tame, there is $R^q \in \mathsf{Pos}(R)$ such that $R^q \to R^s$ holds in Σ_{UFD} for all $R^s \in O$, but $\varphi' : R^q \to R^r$ does not hold in Σ_{UFD}. Now, as $R^L \subseteq O$, we know that $R^q \to R^s$ holds in Σ_{UFD} for all $R^s \in R^L$, so that, by transitivity with φ, as Σ_{FD} is closed by implication, φ' holds in Σ_{FD}. As φ' is a UFD, by definition of Σ_{UFD}, φ' holds in Σ_{UFD}, a contradiction. □

Corollary 14.3.3. *For any instance I, if I has the tame overlaps property for Σ_{UFD}, then I satisfies Σ_{FD}.*

Proof. Considering any two facts F and F' in I, as $O := \mathsf{OVL}(F, F')$ is tame, we know that for any FD $\varphi : R^L \to R^r$ in Σ_{FD}, we cannot have $R^L \subseteq O$ but $R^r \notin O$. Hence, F and F' cannot be a violation of φ. □

We forget for now the disjointness condition in the Dense Interpretations Theorem, which we will prove at the very end of the section (Corollary 14.3.6), and focus only on the first part. We claim the following generalization of the result:

Theorem 14.3.4. *Let R be a relation and Σ_{UFD} be a set of UFDs over R. Let D be the smallest possible cardinality of a* key *K of R (i.e., $K \subseteq \mathsf{Pos}(R)$ and for all $R^q \in \mathsf{Pos}(R)$, there is $R^p \in K$ such that $R^p \to R^q$ holds in Σ_{UFD}). Let x be $\frac{D}{D-1}$ if $D > 1$ and 1 otherwise.*

For every $N \in \mathbb{N}$, there exists a finite instance I of R such that $|\text{dom}(I)|$ is $O(N)$, $|I|$ is $\Omega(N^x)$, and I has the tame overlaps property for Σ_{UFD}.

Observe that, thanks to the use of the tame overlaps, the result does not mention higher-arity FDs, only UFDs; intuitively, tame overlaps ensures that the construction works for any FDs that have the same consequences as UFDs.

It is clear that this theorem implies the first part of the Dense Interpretations Theorem, because if R has no unary key for Σ_{FD} then $D > 1$ and thus $x > 1$, which implies that, for any K, by taking a sufficiently large N_0, we can obtain for all

$N \geqslant N_0$ an instance I for R with $\leqslant N$ elements and $\geqslant K \cdot N$ facts that has the tame overlaps property for Σ_{UFD}; now, by Lemma 14.3.3, this implies that I satisfies Σ_{FD}.

In the rest of this section, we prove Theorem 14.3.4, until the very end where we additionally show that we can enforce the disjointness condition for the Dense Interpretations Theorem. Fix the relation R and set of UFDs Σ_{UFD}. The case of $D = 1$ is vacuous and can be eliminated directly (consider the instance $\{R(a_i, \ldots, a_i) \mid 1 \leqslant i \leqslant N\}$). Hence, assume that $D > 1$, and let $x := \frac{D}{D-1}$.

We first show the claim on a specific relation R_{full} and set $\Sigma^{\mathsf{full}}_{\text{UFD}}$ of UFDs. We will then generalize the construction to arbitrary relations and UFDs. Let $T := \{1, \ldots, D\}$, and consider a bijection $\nu : \{1, \ldots, 2^D - 1\} \to \mathfrak{P}(T)\backslash\{\emptyset\}$, where $\mathfrak{P}(T)$ denotes the powerset of T. Let R_{full} be a $(2^D - 1)$-ary relation, and take $\Sigma^{\mathsf{full}}_{\text{UFD}} := \{R^i \to R^j \mid \nu(i) \subseteq \nu(j)\}$. Note that $\Sigma^{\mathsf{full}}_{\text{UFD}}$ is clearly closed under implication of UFDs. Fix $N \in \mathbb{N}$, and let us construct an instance I_{full} with $O(N)$ elements and $\Omega(N^x)$ facts.

Fix $n := \lfloor N^{1/(D-1)} \rfloor$. Let $\mathcal{F}$ be the set of partial functions from T to $\{1, \ldots, n\}$, and write $\mathcal{F} = \mathcal{F}_{\text{t}} \sqcup \mathcal{F}_{\text{p}}$, where $\mathcal{F}_{\text{t}}$ and $\mathcal{F}_{\text{p}}$ are respectively the total and the strictly partial functions. We take I_{full} to consist of one fact F_f for each $f \in \mathcal{F}_{\text{t}}$, where $F_f = R_{\mathsf{full}}(\boldsymbol{a^f})$ is defined as follows: for $1 \leqslant i \leqslant 2^D - 1$, $a^f_i := f_{|T\backslash\nu(i)}$. In particular:

- $a^f_{\nu^{-1}(T)}$, the element of F_f at the position mapped by ν to $T \in \mathfrak{P}(T)\backslash\{\emptyset\}$, is the strictly partial function that is nowhere defined; note that $R^{\nu^{-1}(T)}_{\mathsf{full}}$ is determined by *all* positions in $\Sigma^{\mathsf{full}}_{\text{UFD}}$.
- $a^f_{\nu^{-1}(\{i\})}$, the element of F_f at the position mapped by ν to $\{i\} \in \mathfrak{P}(T)\backslash\{\emptyset\}$, is the strictly partial function equal to f except that it is undefined on i; note that $R^{\nu^{-1}(\{i\})}_{\mathsf{full}}$ is determined by no other position of R_{full} in $\Sigma^{\mathsf{full}}_{\text{UFD}}$.

Hence, $\text{dom}(I_{\mathsf{full}}) = \mathcal{F}_{\text{p}}$ (because $\emptyset$ is not in the image of ν), so that $|\text{dom}(I_{\mathsf{full}})| = \sum_{0 \leqslant i < D} \binom{D}{i} n^i$. Remembering that D is a constant, this implies that $|\text{dom}(I_{\mathsf{full}})|$ is $O(n^{D-1})$, so it is $O(N)$ by definition of n. Further, we claim that $|I_{\mathsf{full}}| = |\mathcal{F}_{\text{t}}| = n^D = N^x$. To show this, consider two facts F_f and F_g. We show that $F_f = F_g$ implies $f = g$, so there are indeed $|\mathcal{F}_{\text{t}}|$ different facts in I_{full}. As $\pi_{\nu^{-1}(\{1\})}(F_f) = \pi_{\nu^{-1}(\{1\})}(F_g)$, we have $f(t) = g(t)$ for all $t \in T\backslash\{1\}$, and as $D \geqslant 2$, we can look at $\pi_{\nu^{-1}(\{2\})}(F_f)$ and $\pi_{\nu^{-1}(\{2\})}(F_g)$ to conclude that $f(1) = g(1)$, hence $f = g$ as claimed. Hence, the cardinalities of I_{full} and of its domain are suitable.

We must now show that I_{full} has the tame overlaps property for $\Sigma^{\mathsf{full}}_{\text{UFD}}$. For this we first make the following general observation:

Lemma 14.3.5. *Let Σ_{UFD} be any conjunction of UFDs and I be an instance such that $I \models \Sigma_{\text{UFD}}$. Assume that, for any pair of facts $F \neq F'$ of I that overlap, there exists $R^p \in \mathsf{OVL}(F, F')$ which is a unary key for $\mathsf{OVL}(F, F')$. Then I has the tame overlaps property for Σ_{UFD}.*

Proof. Consider $F, F' \in I$ and $O := \mathsf{OVL}(F, F')$. If $F = F'$, then $O = \mathsf{Pos}(R)$, and O is vacuously tame. Otherwise, if $F \neq F'$, let $R^p \in \mathsf{Pos}(R)\backslash O$. We take $R^q \in O$ to be the unary key of O. We know that $R^q \to R^s$ holds in Σ_{UFD} for all $R^s \in O$, so to show that O is tame it suffices to show that $\varphi : R^q \to R^p$ does not hold in Σ_{UFD}. However, if it did, then as $R^q \in O$ and $R^p \notin O$, F and F' would witness a violation of φ, contradicting the fact that I satisfies Σ_{UFD}. □

So we show that I_{full} satisfies $\Sigma^{\mathsf{full}}_{\mathrm{UFD}}$ and that every non-empty overlap between facts of I_{full} has a unary key, so we can conclude by Lemma 14.3.5 that I_{full} has tame overlaps.

First, to show that I_{full} satisfies $\Sigma^{\mathsf{full}}_{\mathrm{UFD}}$, observe that (*) whenever $\varphi : R^i_{\mathsf{full}} \to R^j_{\mathsf{full}}$ holds in $\Sigma^{\mathsf{full}}_{\mathrm{UFD}}$, then $\nu(i) \subseteq \nu(j)$, so that, for any fact F of I_{full}, for any $1 \leqslant t \leqslant D$, whenever $(\pi_j(F))(t)$ is defined, so is $(\pi_i(F))(t)$, and we have $(\pi_j(F))(t) = (\pi_i(F))(t)$. Further, by our construction, we easily see that (**) for any fact F of I_{full}, for any $1 \leqslant i \leqslant 2^D - 1$ and $1 \leqslant t \leqslant D$, the fact that $(\pi_i(F))(t)$ is defined or not only depends on i and t, not on F. Hence, consider a UFD $\varphi : R^i_{\mathsf{full}} \to R^j_{\mathsf{full}}$ in $\Sigma^{\mathsf{full}}_{\mathrm{UFD}}$, let F and F' be two facts of I_{full} such that $\pi_i(F) = \pi_i(F')$, and show that $\pi_j(F) = \pi_j(F')$. Take $1 \leqslant t \leqslant D$ and show that either $(\pi_j(F))(t)$ and $(\pi_j(F'))(t)$ are both undefined, or they are both defined and equal. By (**), either both are undefined or both are defined, so it suffices to show that if they are defined then they are equal. But then, if both are defined, by (*), we have $(\pi_j(F'))(t) = (\pi_i(F'))(t) = (\pi_i(F))(t) = (\pi_j(F))(t)$. So we conclude indeed that $\pi_j(F) = \pi_j(F')$, so that F and F' cannot witness a violation of φ. Hence, $I_{\mathsf{full}} \models \Sigma^{\mathsf{full}}_{\mathrm{UFD}}$.

Second, to show that non-empty overlaps in I_{full} have unary keys, consider two facts $F_f = R_{\mathsf{full}}(\boldsymbol{a^f})$ and $F_g = R_{\mathsf{full}}(\boldsymbol{a^g})$, with $f \neq g$ so that $F_f \neq F_g$. Assume that $\mathsf{OVL}(F_f, F_g)$ is non-empty, and let us show that it has a unary key. Let $O := \{t \in T \mid f(t) = g(t)\}$, and let $X = T \backslash O$; we have $X \neq \emptyset$, because otherwise $f = g$, so we can define $p := \nu^{-1}(X)$. We will show that

$$\mathsf{OVL}(F_f, F_g) = \{R^i \in \mathsf{Pos}(R_{\mathsf{full}}) \mid X \subseteq \nu(i)\}$$

This implies that $R^p \in \mathsf{OVL}(F_f, F_g)$ and that R^p is a unary key of $\mathsf{OVL}(F_f, F_g)$, because, for all $R^q \in \mathsf{OVL}(F_f, F_g)$, $X \subseteq \nu(q)$, so that $R^p \to R^q$ holds in $\Sigma^{\mathsf{full}}_{\mathrm{UFD}}$.

To show the equality above, consider R^i such that $X \subseteq \nu(i)$. Then $T \backslash \nu(i) \subseteq T \backslash X$. Because $a^f_i = f_{|T \backslash \nu(I)}$ and $a^g_i = g_{|T \backslash \nu(I)}$, we have $a^f_i = a^g_i$ by definition of $O = T \backslash X$. Thus $R^i \in \mathsf{OVL}(F_f, F_g)$. Conversely, if $R^i \in \mathsf{OVL}(F_f, F_g)$, then we have $a^f_i = a^g_i$, so by definition of O we must have $T \backslash \nu(i) \subseteq O = T \backslash X$, which implies $X \subseteq \nu(i)$.

Hence, I_{full} is a finite instance of $\Sigma^{\mathsf{full}}_{\mathrm{UFD}}$ which satisfies the tame overlaps property and contains $O(N)$ elements and $\Omega(N^x)$ facts. This concludes the proof of Theorem 14.3.4 for the specific case of R_{full} and $\Sigma^{\mathsf{full}}_{\mathrm{UFD}}$.

Let us now show Theorem 14.3.4 for an arbitrary relation R and set Σ_{UFD} of UFDs. Let K be a key of R of minimal cardinality, so that $|K| = D$. Let λ be a bijection from K to T. Extend λ to a function μ such that, for all $R^p \in \mathsf{Pos}(R)$, we set $\mu(R^p) := \{\lambda(R^k) \mid R^k \in K$ such that $R^k = R^p$ or $R^k \to R^p$ holds in $\Sigma_{\mathrm{UFD}}\}$; note that this set is never empty.

Consider the instance I_{full} for relation R_{full} that we defined previously, and create an instance I of R that contains, for every fact $R_{\mathsf{full}}(\boldsymbol{a})$ of I_{full}, a fact $F = R(\boldsymbol{b})$ in I, with $b_i = a_{\nu^{-1}(\mu(R^i))}$ for all $1 \leqslant i \leqslant |R|$.

We first show that $|\mathrm{dom}(I)| = O(N)$ and $|I| = \Omega(N^x)$. Indeed, for the first point, we have $\mathrm{dom}(I) \subseteq \mathrm{dom}(I_{\mathsf{full}})$, and as we had $|\mathrm{dom}(I_{\mathsf{full}})| = O(N)$, we deduce the same of $\mathrm{dom}(I)$. For the second point, it suffices to show that we never create the same fact twice in I for two different facts of I_{full}. Assume that there are two facts $F_f = R_{\mathsf{full}}(\boldsymbol{a})$ and $F_g = R_{\mathsf{full}}(\boldsymbol{a'})$ in I_{full} for which we created the same fact $F = R(\boldsymbol{b})$ in I, and let us show that we then have $f = g$ so that $F_f = F_g$. As $|K| \geqslant 2$, consider $R^{k_1} \neq R^{k_2}$ in K. We have $\mu(R^{k_1}) = \{\lambda(R^{k_1})\}$ and $\mu(R^{k_2}) = \{\lambda(R^{k_2})\}$. Hence,

let $i_j := \lambda(R^{k_j})$ for $j \in \{1, 2\}$; as λ is bijective, we deduce from $R^{k_1} \neq R^{k_2}$ that $i_1 \neq i_2$. From the definition of b_{k_1} we deduce that $a_{\nu^{-1}(\{i_1\})} = a'_{\nu^{-1}(\{i_1\})}$, and likewise $a_{\nu^{-1}(\{i_2\})} = a'_{\nu^{-1}(\{i_2\})}$. Similarly to the proof of why I_{full} has no duplicate facts, this implies that $f(t) = g(t)$ for all $t \in T \backslash \{i_1\}$ and for all $t \in T \backslash \{i_2\}$. As $i_1 \neq i_2$, we conclude that $f = g$, so that $F_f = F_g$. Hence, we have $|I| = |I_{\mathsf{full}}| = \Omega(N^x)$.

Let us now show that I has tame overlaps for Σ_{UFD}. Consider two facts F, F' of I that overlap, and let $O := \mathsf{OVL}(F, F')$. We first claim that there exists $\emptyset \subsetneq K' \subseteq K$, such that, letting $X' := \{\lambda(R^k) \mid R^k \in K'\}$, we have $\mathsf{OVL}(F, F') = \{R^i \in \mathsf{Pos}(R) \mid X' \subseteq \mu(R^i)\}$. Indeed, letting F_f and F_g be the facts of I_{full} used to create F and F', we previously showed the existence of $\emptyset \subsetneq X \subseteq T$ such that $\mathsf{OVL}(F_f, F_g) = \{R^i \in \mathsf{Pos}(R_{\mathsf{full}}) \mid X \subseteq \nu(i)\}$. Our definition of F and F' from F_f and F_g makes it clear that we can satisfy the condition by taking $K' := \lambda^{-1}(X)$, so that $X' = X$.

Consider now $R^p \in \mathsf{Pos}(R) \backslash O$. We cannot have $X' \subseteq \mu(R^p)$, otherwise $R^p \in O$. Hence, there exists $R^k \in K'$ such that $\lambda(R^k) \notin \mu(R^p)$. This implies that $R^k \to R^p$ does not hold in Σ_{UFD}. However, as $R^k \in K'$, we have $\lambda(R^k) \in \mu(R^q)$ for all $R^q \in O$, so that $R^k \to O$ holds in Σ_{UFD}. This proves that $O = \mathsf{OVL}(F, F')$ is tame. Hence, I has the tame overlaps property, which concludes the proof of Theorem 14.3.4.

The only thing left is to show that we can enforce the disjointness condition in the Dense Interpretations Theorem, namely:

Corollary 14.3.6. *We can assume in the Dense Interpretations Theorem (Theorem 14.1.9) the following* ***disjointness condition*** *on the resulting instance I: each element occurs at exactly one position of the relation R. Formally, for all $a \in \mathrm{dom}(I)$, there exists exactly one $R^p \in \mathsf{Pos}(R)$ such that $a \in \pi_{R^p}(I)$.*

Proof. Let I be the instance constructed in the proof of the Dense Interpretations Theorem, and consider the instance I' whose domain is $\{(a, R^p) \mid a \in \mathrm{dom}(I), R^p \in \mathsf{Pos}(\sigma)\}$ and which contains for every fact $F = R(\boldsymbol{a})$ of I a fact $F' = R(\boldsymbol{b})$ such that $b_p = (a_p, R^p)$ for every $R^p \in \mathsf{Pos}(\sigma)$. Clearly this defines a bijection φ from the facts of I to the facts of I', and for any facts F, F' of I', $\mathsf{OVL}(F, F') = \mathsf{OVL}(\varphi^{-1}(F), \varphi^{-1}(F'))$. Thus any violation of the FDs Σ_{FD} in I' would witness one in I. Of course, $|\mathrm{dom}(I')| = |\sigma| \cdot |\mathrm{dom}(I)|$, so to achieve a constant factor of K between the domain size and instance size with the disjointness condition, we need to use the proof of the Dense Interpretations Theorem with a constant factor of $K' := |\sigma| \cdot K$. □

Chapter 15

Blowing Up Cycles

We are now ready to prove the Universal Models Theorem, which concludes the proof of our Main Theorem (Theorem 9.3):

Theorem 9.1.3 (Universal models)**.** *The class of finitely closed* UIDs *and* FDs *has finite universal models for CQ: for every conjunction* Σ *of* FDs Σ_{FD} *and* UIDs Σ_{UID} *closed under finite implication, for any* $k \in \mathbb{N}$*, for every finite instance* I_0 *that satisfies* Σ_{FD}*, there exists a finite* k*-sound superinstance* I *of* I_0 *that satisfies* Σ*.*

To do this, we must ensure k-soundness for arbitrary Boolean CQs rather than just acyclic CQs.

Intuitively, the only cyclic CQs that hold in $\mathsf{Chase}(I_0, \Sigma_{\text{UID}})$ either have an acyclic self-homomorphic match (so they are implied by an acyclic CQ that also holds) or have all cycles matched to elements of I_0. Hence, in a k-sound instance for CQ, no other cyclic queries should be true. Our way to ensure this is by a cycle blowup process: starting with the superinstance constructed by Theorem 14.1, which satisfies Σ and is k-sound for ACQ, we build its product with a group of high girth. The standard way to do so, inspired by [Otto 2002], is presented in Section 15.1.

The problem is that this blowup process may create FD violations. We work around this problem using some additional properties ensured by our construction. In Section 15.2, we accordingly show the Cautious Models Theorem, a variant of Theorem 14.1 with additional properties. Section 15.2 is the only part of this chapter that depends on the details of the previous chapters.

We then apply a slightly different blowup construction to that model, as described in Section 15.3, which ensures that no FD violations are created. This blowup no longer depends on the specifics of the construction, and does not depend on the specific UIDs and FDs that hold; in particular, the blowup constructions do not even require that the UIDs and FDs are finitely closed.

15.1 Simple Product

We first define a simple notion of product, which we can use to extend k-soundness from ACQ to CQ, but which may introduce FD violations. Let us first introduce preliminary notions:

Definition 15.1.1. A *group* $G = (S, \cdot)$ over a finite set S consists of:

- an associative *product law* $\cdot : S \times S \to S$;

- a *neutral element* $e \in S$ such that $e \cdot x = x \cdot e = x$ for all $x \in S$;
- an *inverse law* $\cdot^{-1} : S \to S$ such that $x \cdot x^{-1} = x^{-1} \cdot x = e$ for all $x \in S$.

We say that G is *generated* by $X \subseteq S$ if all elements of S can be written as a product of elements of X and $X^{-1} := \{x^{-1} \mid x \in X\}$.

Given a group $G = (S, \cdot)$ generated by X, assuming $|S| > 2$, the *girth* of G under X is the length of the shortest non-empty word $\boldsymbol{w}$ of elements of X and X^{-1} such that $w_1 \cdots w_n = e$ and $w_i \neq w_{i+1}^{-1}$ for all $1 \leqslant i < n$. ◁

The following result, originally from [Margulis 1982], is proven for $|X| > 1$ in, e.g., [Otto 2012] (Section 2.1), and is straightforward for $|X| = 1$ (take $\mathbb{Z}/n\mathbb{Z}$):

Lemma 15.1.2. *For all $n \in \mathbb{N}$ and finite non-empty set X, there is a finite group $G = (S, \cdot)$ generated by X with girth $\geqslant n$ under X. We call G an* n-acyclic group generated by X.

In other words, in an n-acyclic group generated by X, there is no short product of elements of X and their inverses which evaluates to e, except those that include a factor $x \cdot x^{-1}$.

We now explain how to take the product of a superinstance I of I_0 with such a finite group G. This ensures that any cycles in the product instance are large, because they project to cycles in G. We use a specific generator:

Definition 15.1.3. The *fact labels* of a superinstance I of I_0 are $\Lambda(I) := \{\mathrm{l}_i^F \mid F \in I \backslash I_0, 1 \leqslant i \leqslant |F|\}$, where $|F|$ is the arity of the relation for fact F. ◁

Now, we define the product of a superinstance I of I_0 with a group generated by $\Lambda(I)$. We make sure not to blow up cycles in I_0, so the result remains a superinstance of I_0:

Definition 15.1.4. Let I be a finite superinstance of I_0 and G be a finite group generated by $\Lambda(I)$. The *product of I by G preserving I_0*, written $(I, I_0) \otimes G$, is the finite instance with domain $\mathrm{dom}(I) \times G$ consisting of the following facts, for all $g \in G$:

- For every fact $R(\boldsymbol{a})$ of I_0, the fact $R((a_1, g), \ldots, (a_{|R|}, g))$.
- For every fact $F = R(\boldsymbol{a})$ of $I \backslash I_0$, the fact $R((a_1, g \cdot \mathrm{l}_1^F), \ldots, (a_{|R|}, g \cdot \mathrm{l}_{|R|}^F))$.

We identify (a, e) to a for $a \in \mathrm{dom}(I_0)$, so $(I, I_0) \otimes G$ is still a superinstance of I_0. ◁

It will be simpler to reason about initial instances I_0 where each element has been *individualized* by the addition of a fresh fact that is unique to that element. We give a name to this notion:

Definition 15.1.5. An *individualizing* instance I_0 is such that, for each $a \in \mathrm{dom}(I_0)$, I_0 contains a fact $P_a(a)$ where P_a is a fresh unary predicate which does not occur in queries, in UIDs or in FDs.

An *individualizing superinstance* of an instance I_0 is a superinstance I_1 of I_0 that adds precisely one unary fact $P_a(a)$, for a fresh unary relation P_a, to each $a \in \mathrm{dom}(I_0)$, so that I_1 is individualizing. In particular, we have $\mathrm{dom}(I_0) = \mathrm{dom}(I_1)$, and I_0 and I_1 match for all relations of σ that occur in the query q and the constraints Σ. ◁

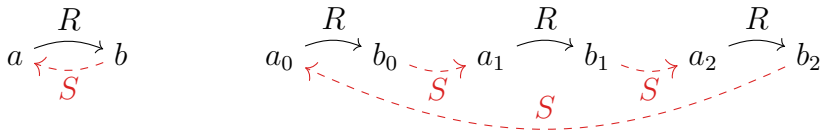


Figure 15.1: Product with a group of large girth (see Example 15.1.7)

We can now state the following property, which we will prove in the rest of this section:

Lemma 15.1.6 (Simple product)**.** *Let Σ be finitely closed FDs and UIDs, let I be a finite superinstance of an individualizing I_0 and let G be a finite $(2k+1)$-acyclic group generated by $\Lambda(I)$. If I is $(k \cdot (|\sigma|+1))$-sound for ACQ, I_0, and Σ, then $(I, I_0) \otimes G$ is k-sound for CQ, I_0, and Σ.*

The following example illustrates the idea of taking the simple product of an instance with a group of high girth:

Example 15.1.7. Consider $F_0 := R(a, b)$ and $I_0 := \{F_0\}$, illustrated in solid black in the left part of Figure 15.1. Consider Σ_{UID} consisting of $\tau : R^2 \subseteq S^1$, $\tau' : S^2 \subseteq R^1$, τ^{-1}, and $(\tau')^{-1}$. Let $F := S(b, a)$, and $I := I_0 \sqcup \{F\}$, where F is a red dashed edge in the drawing. I satisfies Σ_{UID} and is sound for ACQ, but not for CQ: take for instance $q : \exists xy\ R(x, y) \wedge S(y, x)$, which is cyclic and holds in I while $(I_0, \Sigma_{\text{UID}}) \not\models_{\text{unr}} q$.

We have $\Lambda(I) = \{\mathrm{l}_1^F, \mathrm{l}_2^F\}$. Identify l_1^F and l_2^F to 1 and 2 and consider the group $G := (\{0, 1, 2\}, +)$ where $+$ is addition modulo 3. The group G has girth 2 under $\Lambda(I)$.

The product $I_{\text{p}} := (I, I_0) \otimes G$, writing pairs as subscripts for brevity, is $\{R(a_0, b_0), R(a_1, b_1), R(a_2, b_2), S(b_1, a_2), S(b_2, a_0), S(b_0, a_1)\}$. The right part of Figure 15.1 represents I_{p}. Here, I_{p} happens to be 5-sound for CQ.

We cannot directly use the simple product for our purposes, however, because $I_{\text{p}} := (I_{\text{f}}, I_0) \otimes G$ may violate Σ_{UFD} even though our instance I_{f} satisfies Σ_{FD}. Indeed, there may be a relation R, a UFD $\varphi : R^p \to R^q$ in Σ_{UFD}, and two R-facts F and F' in $I_{\text{f}} \backslash I_0$ with $\pi_{R^p, R^q}(F) = \pi_{R^p, R^q}(F')$. In I_{p} there will be images of F and F' that overlap only on R^p, so they will violate φ.

Nevertheless, in the remainder of this section we prove the Simple Product Lemma, as it will be useful for our purposes later. Remember that a *match* of a CQ in an instance is witnessed by a homomorphism h, and that we also call the *match* the image of h. We start by proving an easy lemma:

Lemma 15.1.8. *For any CQ q and instance I, if $I \models q$ with a witnessing homomorphism h that maps two different atoms of q to the same fact, then there is a CQ q' such that:*

- $|q'| < |q|$
- q' entails q, *meaning that for any instance I, if $I \models q'$ then $I \models q$*
- $I \models q'$

Proof. Fix q, I, h, and let $A = R(\boldsymbol{x})$ and $A' = R(\boldsymbol{y})$ be the two atoms of q mapped to the same fact F by h. Necessarily A and A' are atoms for the same relation R of the fact F, and $h(A) = h(A')$ means that $h(x_i) = h(y_i)$ for all $R^i \in \mathsf{Pos}(R)$.

Let $\mathrm{dom}(q)$ be the set of variables occurring in q. Consider the map f from $\mathrm{dom}(q)$ to $\mathrm{dom}(q)$ defined by $f(y_i) = x_i$ for all i, and $f(x) = x$ if x does not occur in A'. Observe that this ensures that $h(x) = h(f(x))$ for all $x \in \mathrm{dom}(q)$. Let $q' = f(q)$ be the query obtained by replacing every variable x in q by $f(x)$, and, as $f(A') = f(A)$, removing one of those duplicate atoms so that $|q'| < |q|$. We claim that $h' := h_{|\,\mathrm{dom}(q')}$ is a match of q' in I. Indeed, observe that any atom $f(A'')$ of q' is homomorphically mapped by h' to $h(A'')$ because $h'(f(x)) = h(x)$ for all x so $h'(f(A'')) = h(A'')$.

To see why q' entails q, observe that f defines a homomorphism from q to q', so that, for any instance I', if q' has a match h'' in I', then $h'' \circ f$ is a match of q in I'. □

Let us now prove the Simple Product Lemma. Fix the constraints Σ and the superinstance I of the individualizing I_0 such that I is $((|\sigma| + 1) \cdot k)$-sound for ACQ, I_0, and Σ. Fix the $(2k + 1)$-acyclic group G generated by $\Lambda(I)$. Consider $I_\mathrm{p} := (I, I_0) \otimes G$, which is a superinstance of I_0, up to our identification of (a, e) to a for $a \in \mathrm{dom}(I_0)$, where e is the neutral element of G. We must show that I_p is k-sound for CQ, I_0, and Σ.

We call a match h of a CQ q in I_p *pure-instance-cyclic* if every atom containing two occurrences of the same variable is mapped by h to a fact of $I_0 \times G$, and every Berge cycle of q contains an atom mapped by h to a fact of $I_0 \times G$. In particular, if q is in ACQ then any match h of q in I_p is vacuously pure-instance-cyclic. Our proof consists of two claims:

1. If a CQ q with $|q| \leqslant k$ has a pure-instance-cyclic match h in I_p, then $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}}) \models q$.

2. If a CQ q with $|q| \leqslant k$ has a match h in I_p which is not pure-instance-cyclic, then there is a CQ q' with $|q'| < |q|$ such that q' entails q and q' has a match in I_p.

The fact that I_p is k-sound for CQ clearly follows from the two claims: if a CQ q with $|q| \leqslant k$ has a match in I_p, then apply the second claim repeatedly until you obtain a CQ q' with $|q'| < |q| \leqslant k$, q' entails q, and q' has a pure-instance-cyclic match in I_p: this must eventually occur because the empty query is in ACQ. Then use the first claim to deduce that $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}}) \models q'$, where it follows that $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}}) \models q$. So it suffices to prove these two claims.

We start by proving the first claim. Let q be a CQ with $|q| \leqslant k$ that has a pure-instance-cyclic match h in I_p.

We partition the atoms of q between the atoms $\mathcal{A}$ matched by h to $I_0 \times G$ and the atoms $\mathcal{A}'$ which are not: we can then write q as $\exists \boldsymbol{x}\ \mathcal{A}(x) \wedge \mathcal{A}'(x)$. Let $\mathcal{A}''$ consist of the atom $P_a(x)$ for each variable x occurring in $\mathcal{A}'$ which is mapped by h to an element $a \in \mathrm{dom}(I_0 \times G)$, and let q' be the query $\exists \boldsymbol{x}\ \mathcal{A}'(x) \wedge \mathcal{A}''(x)$. As I_0 is individualizing, it is immediate that h is a match of q' in I_p.

We first claim that q' is in ACQ. Indeed, no Berge cycle in q' can use the atoms of $\mathcal{A}''$ as they are unary, and for the same reason no atom in $\mathcal{A}''$ contains two occurrences of the same variable. Further, $\mathcal{A}'$ does not contain any Berge

cycle or atom with two occurrences of the same variable, by definition of h being pure-instance-cyclic. Hence, q' is indeed in ACQ. Further, we have $|q'| \leqslant k \cdot (|\sigma|+1)$, as $|\mathcal{A}''| \leqslant |\sigma| \cdot |\mathcal{A}'|$ and we have $|\mathcal{A}'| \leqslant |q| \leqslant k$, so that $|q'| \leqslant k \cdot (|\sigma|+1)$. Now, we know that $I \models q'$, as evidenced by the homomorphism pr from I_p to I defined by $\mathsf{pr} : (a, g) \mapsto a$ for all $a \in \mathrm{dom}(I)$ and $g \in G$. As I is $(k \cdot (|\sigma|+1))$-sound for ACQ, and q' is an ACQ query that holds in I with $|q'| \leqslant k \cdot (|\sigma|+1)$, we know that $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}}) \models q'$.

Now, as $\mathcal{A}''$ covers all variables of q', by definition of I_0 being individualizing, the only possible match of q' in the chase is the one that maps each variable x to the $a \in \mathrm{dom}(I_0)$ such that the atom $P_a(x)$ is in $\mathcal{A}''$. Further, as h matched $\mathcal{A}$ to facts of I_0 such that $h(x) = a$ where $P_a(x)$ occurs in $\mathcal{A}''$, we can clearly extend the match of q' in $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$ to a match of q in $\mathsf{Chase}(I_0, \Sigma_{\mathrm{UID}})$. This concludes the proof of the first claim.

We now prove the second claim. Let q be a CQ with $|q| \leqslant k$ that has a match h in I_p which is not pure-instance-cyclic. Consider a Berge cycle C of q, of the form $A_1, x_1, A_2, x_2, \ldots, A_n, x_n$, where the A_i are pairwise distinct atoms and the x_i pairwise distinct variables, where the A_i are mapped by h to facts not in $I_0 \times G$, and where for all $1 \leqslant i \leqslant n$, variable x_i occurs at position q_i of atom A_i and position p_{i+1} of A_{i+1}, with addition modulo $n := |C|$. We assume without loss of generality that $p_i \neq q_i$ for all i. However, we do not assume that $n \geqslant 2$: either $n \geqslant 2$ and C is really a Berge cycle according to our previous definition, or $n = 1$ and variable x_1 occurs in atom A_1 at positions $p_1 \neq q_1$, which corresponds to the case where there are multiple occurrences of the same variable in an atom.

For $1 \leqslant i \leqslant n$, we write $F_i = R_i(\boldsymbol{a^i})$ the image of A_i by h in I_p; by definition of I_p, as F_i is not a fact of $I_0 \times G$, there is a fact $F'_i = R_i(\boldsymbol{b^i})$ of I and $g_i \in G$ such that $a^i_j = (b^i_j, g_i \cdot \mathrm{l}_j^{F'_i})$ for $R^j_i \in \mathsf{Pos}(R_i)$. Now, for all $1 \leqslant i \leqslant n$, as $h(x_i) = a^i_{q_i} = a^{i+1}_{p_i+1}$ for all $1 \leqslant i \leqslant n$, we deduce by projecting on the second component that $g_i \cdot \mathrm{l}_{q_i}^{F'_i} = g_{i+1} \cdot \mathrm{l}_{p_{i+1}}^{F'_{i+1}}$, so that, by collapsing the equations of the cycle together, $\mathrm{l}_{q_1}^{F'_1} \cdot (\mathrm{l}_{p_2}^{F'_2})^{-1} \cdot \cdots \cdot \mathrm{l}_{q_{n-1}}^{F'_{n-1}} \cdot (\mathrm{l}_{p_n}^{F'_n})^{-1} \cdot \mathrm{l}_{q_n}^{F'_n} \cdot (\mathrm{l}_{p_1}^{F'_1})^{-1} = e$.

As the girth of G under $\Lambda(I)$ is $\geqslant 2k+1$, and this product contains $2n \leqslant 2k$ elements, we must have either $\mathrm{l}_{q_i}^{F'_i} = \mathrm{l}_{p_{i+1}}^{F'_{i+1}}$ for some i, or $\mathrm{l}_{p_i}^{F'_i} = \mathrm{l}_{q_i}^{F'_i}$ for some i. The second case is impossible because we assumed that $p_i \neq q_i$ for all $1 \leqslant i \leqslant n$. Hence, necessarily $\mathrm{l}_{q_i}^{F'_i} = \mathrm{l}_{p_{i+1}}^{F'_{i+1}}$, so in particular we must have $n > 1$ and $F'_i = F'_{i+1}$. Hence the atoms $A_i \neq A_{i+1}$ of q are mapped by h to the same fact $F'_i = F'_{i+1}$. We conclude by Lemma 15.1.8 that there is a strictly smaller q' which entails q and has a match in I_p, which is what we wanted to show. This concludes the proof of the second claim, and of the Simple Product Lemma.

15.2 Cautiousness

As the simple product may cause FD violations, we will define a more refined notion of product, which intuitively does not attempt to blow up cycles within fact overlaps. In order to clarify this, however, we will first need to study in more detail the instance I_f to which we will apply the process, namely, the one that we constructed to prove Theorem 14.1. We will consider a *quotient* of I_f:

Definition 15.2.1. The *quotient* $I/\sim$ of an instance I by an equivalence relation $\sim$ on $\mathrm{dom}(I)$ is defined as follows:

- $\mathrm{dom}(I/\sim)$ is the equivalence classes of $\sim$ on $\mathrm{dom}(I)$,
- $I/\sim$ contains one fact $R(\boldsymbol{A})$ for every fact $R(\boldsymbol{a})$ of I, where A_i is the $\sim$-class of a_i for all $R^i \in \mathsf{Pos}(R)$.

The *quotient homomorphism* $\chi_\sim$ is the homomorphism from I to $I/\sim$ defined by mapping each element of $\mathrm{dom}(I)$ to its $\sim$-class. $\lhd$

We quotient I_f by the equivalence relation $\simeq_k$ (recall Definition 12.1.1). The result may no longer satisfy Σ. However, it is still k-sound for ACQ, for the following reason:

Lemma 15.2.2. *Any k-bounded simulation from an instance I to an instance I' defines a k-bounded simulation from $I/\simeq_k$ to I'.*

Proof. Fix the instance I and the k-bounded simulation sim to an instance I', and consider $I'' := I/\simeq_k$. We show that there is a k-bounded simulation sim' from I'' to I, because $\mathsf{sim} \circ \mathsf{sim}'$ would then be a k-bounded simulation from I'' to I', the desired claim. We define $\mathsf{sim}'(A)$ for all $A \in I''$ to be a for any member $a \in A$ of the equivalence class A in I, and show that sim' thus defined is indeed a k-bounded simulation.

We will show the stronger result that $(I'', A) \leqslant_k (I, a)$ for all $A \in \mathrm{dom}(I'')$ and for any $a \in A$. We do it by proving, by induction on $0 \leqslant k' \leqslant k$, that $(I'', A) \leqslant_{k'} (I, a)$ for all $A \in \mathrm{dom}(I'')$ and $a \in A$. The case $k' = 0$ is trivial. Hence, fix $0 < k' \leqslant k$, assume that $(I'', A) \leqslant_{k'-1} (I, a)$ for all $A \in \mathrm{dom}(I'')$ and $a \in A$, and show that this is also true for k'. Choose $A \in \mathrm{dom}(I'')$, $a \in A$, we must show that $(I'', A) \leqslant_{k'} (I, a)$. To do so, consider any fact $F = R(\boldsymbol{A})$ of I'' such that $A_p = A$ for some $R^p \in \mathsf{Pos}(R)$. Let $F' = R(\boldsymbol{a}')$ be a fact of I that is a preimage of F by $\chi_{\simeq_k}$, so that $a'_q \in A_q$ for all $R^q \in \mathsf{Pos}(R)$. We have $a'_p \in A$ and $a \in A$, so that $a'_p \simeq_k a$ holds in I. Hence, in particular we have $(I, a'_p) \leqslant_{k'} (I, a)$ because $k' \leqslant k$, so there exists a fact $F'' = R(\boldsymbol{a}'')$ of I such that $a''_p = a$ and $(I, a'_q) \leqslant_{k'-1} (I, a''_q)$ for all $R^q \in \mathsf{Pos}(R)$. We show that F'' is a witness fact for F. Indeed, we have $a''_p = a$. Let us now choose $R^q \in \mathsf{Pos}(R)$ and show that $(I'', A_q) \leqslant_{k'-1} (I, a''_q)$. By induction hypothesis, as $a'_q \in A_q$, we have $(I'', A_q) \leqslant_{k'-1} (I, a'_q)$, and as $(I, a'_q) \leqslant_{k'-1} (I, a''_q)$, by transitivity we have indeed $(I'', A_q) \leqslant_{k'-1} (I, a''_q)$. Hence, we have shown that $(I'', A) \leqslant_{k'} (I, a)$.

By induction, we conclude that $(I'', A) \leqslant_k (I, a)$ for all $A \in \mathrm{dom}(I'')$ and $a \in A$, so that there is indeed a k-bounded simulation from I'' to I, which, as we have explained, implies the desired claim. $\square$

Let us thus consider $I'_\mathrm{f} := I_\mathrm{f}/\simeq_k$ which is still k-sound for ACQ by the previous lemma, and consider the homomorphism $\chi_{\simeq_k}$ from I_f to I'_f. Our idea is to blow up cycles in I_f by a *mixed product* that only distinguishes facts that have a different image in I'_f by $\chi_{\simeq_k}$. This is sufficient to lift k-soundness from ACQ to CQ, and it will not create FD violations on facts that have the same image by $\chi_{\simeq k}$. Crucially, however, we can show from our construction that all overlapping facts of I_f have the same image by $\chi_{\simeq k}$. Let us formalize this condition:

Definition 15.2.3. Let I be an instance, let $I_1 \subseteq I$, and let f be any mapping with domain I. We say I is *cautious* for f and I_1 if for any two *overlapping* facts, namely, two facts $F = R(\boldsymbol{a})$ and $F' = R(\boldsymbol{b})$ of the same relation with $a_p = b_p$ for some $R^p \in \mathsf{Pos}(R)$, one of the following holds: $F, F' \in I_1$, or $f(a_p) = f(b_p)$ for all $R^p \in \mathsf{Pos}(R)$. ◁

We conclude the section by presenting a strengthening of Theorem 14.1. This is the only point in this chapter where we rely on the details of the process of the previous chapters:

Theorem 15.2.4 (Cautious models)**.** *For any finitely closed Σ formed of UIDs Σ_{UID} and FDs Σ_{FD}, instance I_0, and $k \in \mathbb{N}$, we can build a finite superinstance I_{f} of an instance I_1 such that:*

- *I_{f} satisfies Σ;*
- *I_{f} is k-sound for Σ, ACQ, and I_1;*
- *I_1 is an individualizing superinstance of a disjoint union of copies of I_0;*
- *I_{f} is cautious for $\chi_{\simeq_k}$ and I_1.*

We will use the Cautious Models Theorem in the next section. For now, let us show how to prove it. Fix Σ, I_0, and $k \in \mathbb{N}$. Let $I_{0,\mathrm{i}}$ be an individualizing superinstance of I_0, and apply k UID chase rounds with the UIDs of Σ_{UID} to $I_{0,\mathrm{i}}$ to obtain $I'_{0,\mathrm{i}}$. Apply the Sufficiently Envelope-Saturated Proposition to $I'_{0,\mathrm{i}}$ to obtain an aligned superinstance J of a disjoint union $I''_{0,\mathrm{i}}$ of copies of $I'_{0,\mathrm{i}}$. Now, modify J to J' and $I''_{0,\mathrm{i}}$ to I'_1 by replacing the copies of the facts of $I_{0,\mathrm{i}} \backslash I_0$ by new individualizing facts (i.e., make the individualizing facts unique across copies of $I'_{0,\mathrm{i}}$). This ensures by definition that I'_1 is the result of applying k UID chase rounds to an individualizing superinstance of a disjoint union of copies of I_0. Further, the modification to J' can be done so as to ensure that J' is an aligned superinstance of I'_1; the k chase rounds applied when defining $I'_{0,\mathrm{i}}$ ensure that the sim mapping can still be defined notwithstanding the change in the individualizing facts. Further, we have $|J'| = |J|$, so J' is still sufficiently envelope-saturated.

We now apply the Envelope-Thrifty Completion Proposition to the aligned superinstance J' of I'_1 to obtain a superinstance J_{f} of I'_1 which is k-sound for Σ, ACQ, and I'_1, and that satisfies Σ. Now, define I_1 from I'_1 by removing the facts created in the k UID chase rounds, so it is by definition an individualizing superinstance of a disjoint union of copies of I_0. As I'_1 is the result of applying chase rounds to I_1, I_{f} is also k-sound for Σ, ACQ and I_1. Hence, I_{f} satisfies the first three conditions that we have to show in the Cautious Models Theorem. The only thing left is to show the last one, namely:

Lemma 15.2.5 (Cautiousness)**.** *I_{f} is cautious for $\chi_{\simeq_k}$ and I_1.*

We show the Cautiousness Lemma in the rest of the section, which concludes the proof of the Cautious Models Theorem.

We first show that overlapping facts in $J_{\mathrm{f}} = (I_{\mathrm{f}}, \mathsf{sim}_{\mathrm{f}})$ are cautious for the sim mapping that we construct, in terms of $\simeq_k$-classes. Formally, let $I_{\mathrm{c}} := \mathsf{Chase}(I_1, \Sigma_{\mathrm{UID}})$, and let $\chi'_{\simeq_k}$ be the homomorphism from I_{c} to $I_{\mathrm{c}}/\simeq_k$. We claim:

Lemma 15.2.6. *I_f is cautious for $\chi'_{\simeq_k} \circ \mathsf{sim}$ and I_1.*

In other words, whenever two facts $F = R(\boldsymbol{a})$ and $F' = R(\boldsymbol{b})$ have non-empty overlap in I_f and are not both in I_1, then, for any position $R^p \in \mathsf{Pos}(R)$, we have $\mathsf{sim}(a_p) \simeq_k \mathsf{sim}(b_p)$ in I_c.

Proof. We first check that this claim holds on the result J' of the Sufficiently Envelope-Saturated Proposition (with our modifications to the individualizing facts). J' is a disjoint union of instances J_D for each fact class $D \in \mathsf{AFactCl}$. If D is safe, no facts overlap in J_D except possibly fact pairs in the copy of I_0, hence, in I_1. For unsafe D, in Lemma 14.1.6, the only facts with non-empty overlap in J_D are fact pairs in some copy of I_0, hence in I_1, or they are the facts $f'(F_i)$, which all map to $\simeq_k$-equivalent sim-images by construction. So the claim holds on J'.

Second, it suffices to show that the claim is preserved by envelope-thrifty chase steps. By their definition, whenever we create a new fact F_n for a fact class D, the only elements of F_n that can be part of an overlap between F_n and an existing fact are envelope elements, appearing at the one position at which they appear in $\mathcal{E}(D)$. Then, by condition 4 of the definition of envelopes (Definition 14.1.1), we deduce that the two overlapping facts achieve the same fact class. □

Returning to the proof of the Cautiousness Lemma, we now show that two elements in J_f having $\simeq_k$-equivalent sim images in I_c must themselves be $\simeq_k$-equivalent in J_f. We do it by showing that, in fact, for any $a \in \mathrm{dom}(I_\mathrm{f})$, not only do we have $(I_\mathrm{f}, a) \leqslant_k (I_\mathrm{c}, \mathsf{sim}(a))$, as required by the k-bounded simulation sim, but we also have the reverse: $(I_\mathrm{c}, \mathsf{sim}(a)) \leqslant_k (I_\mathrm{f}, a)$; in fact, we even have a *homomorphism* from I_c to I_f that maps $\mathsf{sim}(a)$ to a. The existence of this homomorphism is thanks to our specific definition of sim, and on the directionality condition of aligned superinstances; further, it only holds for the final result I_f, which satisfies Σ_UID; it is not respected at intermediate steps of the process.

To prove this, and conclude the proof of the Cautiousness Lemma, remember the forest structure on the UID chase (Definition 12.4.2). We define the *ancestry* $\mathcal{A}_F$ of a fact F in I_c as I_1 plus the facts of the path in the chase forest that leads to F; if $F \in I_1$ then $\mathcal{A}_F$ is just I_1. The *ancestry* $\mathcal{A}_a$ of $a \in \mathrm{dom}(I_\mathrm{c})$ is that of the fact where a was introduced.

We now claim the following lemma about J_f, which relies on the directionality condition:

Lemma 15.2.7. *For any $a \in \mathrm{dom}(I_\mathrm{f})$, there is a homomorphism h_a from $\mathcal{A}_{\mathsf{sim}(a)}$ to I_f such that $h_a(\mathsf{sim}(a)) = a$.*

Proof. We prove that this property holds on I_f, by first showing that it is true of J' constructed by our modification of the Sufficiently Envelope-Saturated Solutions Proposition. This is clearly the case because the instances created by Lemma 14.1.6 are just truncations of the chase where some elements are identified at the last level.

Second, we show that the property is maintained by envelope-thrifty steps; in fact, by any thrifty chase steps (Definition 12.3.1) Consider a thrifty chase step where, in a state $J_1 = (I_1, \mathsf{sim}_1)$ of the construction of our aligned superinstance, we apply a UID $\tau : R^p \subseteq S^q$ to a fact $F_\mathrm{a} = R(\boldsymbol{a})$ to create a fact $F_\mathrm{n} = S(\boldsymbol{b})$ and obtain the aligned superinstance $J_2 = (I_2, \mathsf{sim}_2)$. Consider the chase witness $F_\mathrm{w} = S(\boldsymbol{b'})$. By Lemma 12.3.2, b'_q is the exported element between F_w and its parent in $\mathsf{Chase}(I_0, \Sigma_\mathrm{UID})$. So we know that for any $S^r \neq S^q$, we have $\mathcal{A}_{b'_r} = \mathcal{A}_{b'_q} \sqcup \{F_\mathrm{w}\}$.

We must build the desired homomorphism h_a for all $a \in \mathrm{dom}(I_2) \setminus \mathrm{dom}(I_1)$. Indeed, for $a \in \mathrm{dom}(I_1)$, by hypothesis on I_1, there is a homomorphism h_a from $\mathcal{A}_{\mathsf{sim}_1(a)}$ to I_1 with $h_a(\mathsf{sim}_1(a)) = a$, and as $\mathsf{sim}_2(a) = \mathsf{sim}_1(a)$, we can use h_a as the desired homomorphism from $\mathcal{A}_{\mathsf{sim}_2(a)}$ to I_2. So let us pick $b \in \mathrm{dom}(I_2) \setminus \mathrm{dom}(I_1)$ and construct h_b. By construction of I_2, b must occur in the new fact F_n; further, by definition of thrifty chase steps, we have defined $\mathsf{sim}_2(b) := b'_r$ for some S^r where $b_r = b$. Now, as $a_p = b_q$ is in $\mathrm{dom}(I_1)$, we know that there is a homomorphism h_{b_q} from $\mathcal{A}_{\mathsf{sim}(b_q)} = \mathcal{A}_{b'_q}$ to I_1 such that we have $h_{b_q}(b'_q) = b_q$. We extend h_{b_q} to the homomorphism h_b from $\mathcal{A}_{b'_r} = \mathcal{A}_{b'_q} \sqcup \{F_\mathrm{w}\}$ to I_2 such that $h_b(b'_r) = b$, by setting $h_b(F_\mathrm{w}) := F_\mathrm{n}$ and $h_b(F) := h(F)$ for any other F of $\mathcal{A}_{b'_r}$; we can do this because, by definition of the UID chase, F_w shares no element with the other facts of $\mathcal{A}_{b'_r}$ (that is, with $\mathcal{A}_{b'_q}$), except b'_q for which our definition coincides with the existing image of b'_q by h_{b_q}. This proves the claim. □

This allows us to deduce the following, which is specific to J_f, and relates to the universality of the chase I_c:

Corollary 15.2.8. *For any $a \in \mathrm{dom}(I_\mathrm{f})$, there is a homomorphism h_a from I_c to I_f such that $h_a(\mathit{sim}(a)) = a$.*

Proof. Choose $a \in \mathrm{dom}(I_\mathrm{f})$ and let us construct h_a. Let h'_a be the homomorphism from $\mathcal{A}_{\mathsf{sim}(a)}$ to I_f with $h'_a(\mathsf{sim}(a)) = a$ whose existence was proved in Lemma 15.2.7. Now start by setting $h_a := h'_a$, and extend h'_a to be the desired homomorphism, fact by fact, using the property that $I_\mathrm{f} \models \Sigma_\mathrm{UID}$: for any $b \in \mathrm{dom}(I_\mathrm{c})$ not in the domain of h'_a but which was introduced in a fact F whose exported element c is in the current domain of h'_a, let us extend h'_a to the elements of F in the following way: consider the parent fact F' of F in I_c and its match by h'_a in I_f, let τ be the UID used to create F' from F, and $c' \in \mathrm{dom}(I_\mathrm{c})$ be the exported element between F and F' (so $h'_a(c')$ is defined). We know that $c := h'_a(c')$ occurs in I_f at all positions where c' occurs in I_c. Hence, because $I_\mathrm{f} \models \tau$, there must be a suitable fact F'' in I_f to extend h'_a to all elements of F by setting $h'_a(F) := F''$, which is consistent with the image of c previously defined in h'_a. The (generally infinite) result of this process is the desired homomorphism h_a. □

We are now ready to show our desired claim:

Lemma 15.2.9. *For any $a, b \in \mathrm{dom}(I_\mathrm{f})$, if $\mathit{sim}(a) \simeq_k \mathit{sim}(b)$ in I_c, then $a \simeq_k b$ in I_f.*

Proof. Fix $a, b \in \mathrm{dom}(I_\mathrm{f})$. We have $(I_\mathrm{f}, a) \leqslant_k (I_\mathrm{c}, \mathsf{sim}(a))$ because sim is a k-bounded simulation; we have $(I_\mathrm{c}, \mathsf{sim}(a)) \leqslant_k (I_\mathrm{c}, \mathsf{sim}(b))$ because $\mathsf{sim}(a) \simeq_k \mathsf{sim}(b)$; and we have $(I_\mathrm{c}, \mathsf{sim}(b)) \leqslant_k (I_\mathrm{f}, b)$ by Corollary 15.2.8 as witnessed by h_b. By transitivity, we have $(I_\mathrm{f}, a) \leqslant_k (I_\mathrm{f}, b)$. The other direction is symmetric, so the desired claim follows. □

The Cautiousness Lemma (Lemma 15.2.5) follows immediately from Lemma 15.2.6 and Lemma 15.2.9. This concludes the proof of the Cautious Models Theorem.

15.3 Mixed Product

Using the Cautious Models Theorem, we now define the notion of mixed product, which uses the same fact label for facts with the same image by $h := \chi_{\simeq_k}$:

Definition 15.3.1. Let I be a finite superinstance of I_1 with a homomorphism h to another finite superinstance I' of I_1 such that $h_{|I_1}$ is the identity and $h_{|(I \setminus I_1)}$ maps to $I' \setminus I_1$. Let G be a finite group generated by $\Lambda(I')$.

The *mixed product* of I by G via h preserving I_1, written $(I, I_1) \otimes^h G$, is the finite superinstance of I_1 with domain $\mathrm{dom}(I) \times G$ consisting of the following facts, for every $g \in G$:

- For every fact $R(\boldsymbol{a})$ of I_1, the fact $R((a_1, g), \ldots, (a_{|R|}, g))$.
- For every fact $F = R(\boldsymbol{a})$ of $I \setminus I_1$, the fact $R((a_1, g \cdot \mathrm{l}_1^{h(F)}), \ldots, (a_{|R|}, g \cdot \mathrm{l}_{|R|}^{h(F)}))$. $\lhd$

We now show that the mixed product preserves UIDs and FDs when cautiousness is assumed.

Lemma 15.3.2 (Mixed product preservation)**.** *For any UID or FD τ, if $I \models \tau$ and I is cautious for h, then $(I, I_1) \otimes^h G \models \tau$.*

Proof. Write $I_{\mathrm{m}} := (I, I_1) \otimes^h G$ and write I' for the range of h as before.

If τ is a UID, the claim is immediate even without the cautiousness hypothesis. (In fact, the analogous claim could even be proven for the simple product.) Indeed, for any $a \in \mathrm{dom}(I)$ and $R^p \in \mathsf{Pos}(\sigma)$, if $a \in \pi_{R^p}(I)$ then $(a, g) \in \pi_{R^p}(I_{\mathrm{m}})$ for all $g \in G$; conversely, if $a \notin \pi_{R^p}(I)$ then $(a, g) \notin \pi_{R^p}(I_{\mathrm{m}})$ for all $g \in G$. Hence, letting $\tau : R^p \subseteq S^q$ be a UID of Σ_{UID}, if there is $(a, g) \in \mathrm{dom}(I_{\mathrm{m}})$ such that $(a, g) \in \pi_{R^p}(I_{\mathrm{m}})$ but $(a, g) \notin \pi_{S^q}(I_{\mathrm{m}})$ then $a \in \pi_{R^p}(I)$ but $a \notin \pi_{S^q}(I)$. Hence any violation of τ in I_{m} implies the existence of a violation of τ in I, so we conclude because $I \models \tau$.

Assume now that τ is a FD $\varphi : R^L \to R^r$. Assume by contradiction that there are two facts $F_1 = R(\boldsymbol{a})$ and $F_2 = R(\boldsymbol{b})$ in I_{m} that violate φ, i.e., we have $a_l = b_l$ for all $l \in L$, but $a_r \neq b_r$. Write $a_i = (v_i, f_i)$ and $b_i = (w_i, g_i)$ for all $R^i \in \mathsf{Pos}(R)$. Consider $F_1' := R(\boldsymbol{v})$ and $F_2' := R(\boldsymbol{w})$ the facts of I that are the images of F_1 and F_2 by the homomorphism from I_{m} to I that projects on the first component. As $I \models \tau$, F_1' and F_2' cannot violate φ, so as $v_l = w_l$ for all $l \in L$, we must have $v_r = w_r$. Now, as I is cautious for h and F_1' and F_2' overlap (take any $R^{l_0} \in R^L$), either $F_1', F_2' \in I_1$ or $h(F_1') = h(F_2')$.

In the first case, by definition of the mixed product, there are $f, g \in G$ such that $f_i = f$ and $g_i = g$ for all $R^i \in \mathsf{Pos}(R)$. Thus, taking any $l_0 \in L$, as we have $a_{l_0} = b_{l_0}$, we have $f_{l_0} = g_{l_0}$, so $f = g$, which implies that $f_r = g_r$. Hence, as $v_r = w_r$, we have $(v_r, f_r) = (w_r, g_r)$, contradicting the fact that $a_r \neq b_r$.

In the second case, as h is the identity on I_1 and maps $I \setminus I_1$ to $I' \setminus I_1$, $h(F_1') = h(F_2')$ implies that either F_1' and F_2' are both facts of I_1 or they are both facts of $I \setminus I_1$; but we have already excluded the former possibility in the first case, so we assume the latter. By definition of the mixed product, there are $f, g \in G$ such that $f_i = f \cdot \mathrm{l}_i^{h(F_1')}$ and $g_i = g \cdot \mathrm{l}_i^{h(F_2')}$ for all $R^i \in \mathsf{Pos}(R)$. Picking any $l_0 \in L$, from $a_{l_0} = b_{l_0}$, we deduce that $f \cdot \mathrm{l}_{l_0}^{h(F_1')} = g \cdot \mathrm{l}_{l_0}^{h(F_2')}$; as $h(F_1') = h(F_2')$, this simplifies to $f = g$. Hence, $f_r = g_r$ and we conclude like in the first case. $\square$

Second, we show that $h : I \to I'$ lifts to a homomorphism from the mixed product to the simple product, so we can rely on the result of the Simple Product Lemma.

Lemma 15.3.3 (Mixed product homomorphism)**.** *There is a homomorphism from $(I, I_1) \otimes^h G$ to $(I, I_1) \otimes G$.*

Proof. We use the homomorphism $h : I \to I_1$ to define the homomorphism h' from $I_\mathrm{m} := (I, I_1) \otimes^h G$ to $I_\mathrm{p} := (I, I_1) \otimes G$ by $h'((a, g)) := (h(a), g)$ for every $(a, g) \in \mathrm{dom}(I) \times G$.

Consider a fact $F = R(\boldsymbol{a})$ of I_m, with $a_i = (v_i, g_i)$ for all $R^i \in \mathsf{Pos}(R)$. Consider its image $F' = R(\boldsymbol{v})$ by the homomorphism from I_m to I obtained by projecting to the first component, and the image $h(F')$ of F' by the homomorphism h. As $h_{|I_1}$ is the identity and $h_{|(I \setminus I_1)}$ maps to $I_1 \setminus I_1$, $h(F')$ is a fact of I_1 iff F' is. Now by definition of the simple product it is clear that I_p contains the fact $h'(F)$: it was created in I_p from $h(F')$ for the same choice of $g \in G$. This shows that h' is indeed a homomorphism, which concludes the proof. □

We can now conclude our proof of the Universal Models Theorem (Theorem 9.1.3). Let I_1 be the individualizing union of disjoint copies of I_0 and I_f be the superinstance of I_1 given by the Cautious Models Theorem applied to $k' := k \cdot (|\sigma| + 1)$. As I_1 is individualizing, we know that each element of I_1 is alone in its $\simeq_{k'}$-class in I_f, so the restriction of $I_\mathrm{f}/\simeq_{k'}$ to $\chi_{\simeq_{k'}}(I_1)$ is actually I_1 up to isomorphism; so we define I'_f to be $I_\mathrm{f}/\simeq_{k'}$ modified by identifying $\chi_{\simeq_{k'}}(I_1)$ to I_1; it is a finite superinstance of I_1. Let h be the homomorphism from I_f to I'_f obtained by modifying $\chi_{\simeq_{k'}}$ accordingly, which ensures that $h_{|I_1}$ is the identity and $h_{|I_\mathrm{f} \setminus I_1}$ maps to $I'_\mathrm{f} \setminus I_1$.

Let G be a $(2k + 1)$-acyclic group generated by $\Lambda(I'_\mathrm{f})$, and consider $I_\mathrm{p} := (I'_\mathrm{f}, I_1) \otimes G$. As I_f was k'-sound for ACQ, I_1 and Σ, so is I'_f by Lemma 15.2.2, so, as I_1 is individualizing, I_p is k-sound for CQ, I_1 and Σ by the Simple Product Lemma. However, as we explained, it may be the case that $I_\mathrm{p} \not\models \Sigma$. We therefore construct $I_\mathrm{m} := (I_\mathrm{f}, I_1) \otimes^h G$. By the Mixed Product Homomorphism Lemma, I_m has a homomorphism to I_p, so it is also k-sound for CQ, I_1 and Σ. Now, as I_1 is an individualizing superinstance of a disjoint union of copies of I_0, and as the fresh relations in the individualizing superinstance I_1 do not occur in queries or in constraints, it is clear that I_m is also k-sound for CQ, I_0 and Σ. Further, by the conditions ensured by the Cautious Models Theorem, I_f is cautious for h and I_1. So, by the Mixed Product Preservation Lemma, we have $I_\mathrm{m} \models \Sigma$ because $I_\mathrm{f} \models \Sigma$.

Hence, the mixed product I_m is a finite k-universal instance for CQ, I_0 and Σ. This concludes the proof of the Universal Models Theorem, and hence of our main theorem (Theorem 9.3).

Conclusion. *We have thus shown that UIDs and FDs are finitely controllable up to finite closure, which concludes the second part of the manuscript. To prove this result, we have developed the first tools to build finite models on arbitrary arity schemas that satisfy both referential constraints and number restrictions, while controlling which CQs are satisfied.*

A general conclusion including perspectives for future work is given in the next chapter of this manuscript; see in particular the second paragraph of Section 2 of the Conclusion.

Conclusion and Perspectives

My PhD research has investigated the field of uncertain data management from multiple angles. Its general principle was to leverage structure to work around the intractability or undecidability of tasks such as probabilistic query evaluation, open-world query answering, or possible and certain answer computation.

Specifically, I have followed three main axes. The first one, presented in Part I of this manuscript, is the study of probabilistic query evaluation, under a structural restriction on instances that bounds their treewidth. The second axis, not presented here, was to investigate the structures needed to represent uncertainty on ordered data, be it ordered facts or ordered numerical values. The third was the study of open-world query answering, where we constrain the structure of possible completions of an incomplete instance: I have studied the impact of assuming that structures are finite for some constraint languages (Part II of this manuscript), and the decidability in the infinite context of hybrid languages based on existential rules and description logics (not presented here).

The first axis of my research integrates with the well-established line of research on *probabilistic databases* [Suciu, Olteanu, Ré, and Koch 2011], in particular with the well-known dichotomy result on queries [Dalvi and Suciu 2012], but I approached the problem from the uncommon perspective of instance-based restrictions. For this reason, my work is also connected to the field of *algorithmic metatheorems* [Kreutzer 2008] pioneered by [Courcelle 1990]; but I have also tried to link it to the different area of *semiring provenance* [Green and Tannen 2006]. The second axis of my research is more prospective and does not follow an established research trend; part of it [Amarilli, Amsterdamer, Milo, and Senellart 2016] relates to *crowdsourcing*, which is one of the recent fields today where uncertainty management techniques are needed; part of it [Amarilli, Ba, Deutch, and Senellart 2016] relates to the old problem of *possible and certain answers* but in an unfamiliar order-based context. The third axis relates to a long line of work on *open-world query answering*, which I tried to investigate across community borders: I followed both the perspective of *description logics* and *existential rules* [Amarilli and Benedikt 2015a], as well as traditional *database approaches* in Part II of this manuscript. This latter result connects to the specific line of works that has studied open-world query answering *in the finite* (whose most recent representatives are [Bárány, Gottlob, and Otto 2010; Rosati 2011; Gogacz and Marcinkowski 2013; Ibáñez-García, Lutz, and Schneider 2014]) and to the specific technical tools needed to address this.

I now conclude this thesis by presenting some remaining short-term challenges for axes 1 and 3, and then presenting a long-term vision for provenance management and reasoning.

1 Perspectives on Probabilistic Query Evaluation

There are many possible directions for future work about our instance-based dichotomy for tractable probabilistic query evaluation (presented in Part I), as well as the other directions explored there.

Lower bounds. Our lower bounds (Chapter 6) show the intractability of probabilistic query evaluation whenever instance treewidth is unbounded. The most intriguing open question in this context is perhaps to understand whether the results extend *beyond arity-two signatures*, i.e., whether minor extraction techniques can be used on the primal graphs of arbitrary-arity instances. The main problem when doing this is that multiple edges of the Gaifman graph may then be correlated, because they correspond to the same higher-arity fact, so we cannot extract the exact minor that we want. Maybe something can still be done, however, when the signature is fixed. A related question is whether minor extraction can be *derandomized*, so as to show #P-hardness in the usual sense and not under RP reductions (and without making non-uniform complexity-theoretic assumptions), probably by modifying [Chekuri and Chuzhoy 2014a].

Another question is about the *language of queries* needed to show hardness. I use non-monotone first-order for probabilistic evaluation in Section 6.1, but maybe the UCQ$^{\neq}$ q_{p} used in Section 6.4 would also be suitable for this, i.e., I suspect that q_{p} may really be intractable on arbitrary unbounded-treewidth families, not just in the sense of not having polynomial-size OBDDs. I do not know how to show this, however, under the arbitrary subdivisions caused by the minor extraction process. In the same spirit, there could be more to do with the meta-dichotomy result, in particular understanding the exact *complexity of recognition* for intricate queries, and study their relationship with the safe and unsafe query classes of [Jha and Suciu 2013].

Still on the topic of lower bounds, I do not know whether the bounds on *formula-based representations* (Section 3.6) can be improved, or whether superlinear lower bounds can be shown for *circuit representations* of provenance when we do not assume that treewidth is bounded. It seems very unlikely that much progress can be done on this for now, however, as it is notoriously difficult to prove conciseness gaps between Boolean circuits and formulae, and no supralinear bounds on Boolean circuits are known at all for explicit functions [Iwama and Morizumi 2002].

Provenance semirings. The main question left open by our study of semiring provenance on treelike instances (Chapter 5) is whether these constructions could extend *beyond UCQ*$^{\neq}$. Section 5.4 illustrates the problems that we face when doing so, but I do not know whether they could not be addressed with more expressive provenance representations, for instance using *formal series* [Green, Karvounarakis, and Tannen 2007] or a circuit representation thereof; or drawing a connection with *weighted* notions of tree automata or weighted versions of monadic second-order logic [Droste and Gastin 2007; Kreutzer and Riveros 2013], where one could formalize an intrinsic notion of the *multiplicity* of a fact. Alternatively, one could study more expressive queries but in more restricted semirings, e.g., *absorptive* semirings [Deutch, Milo, Roy, and Tannen 2014].

Tractability in combined complexity. Turning to the question of extending our *upper bounds* for probabilistic query evaluation, an important limitation of our current results is that they only look at *data complexity*, namely, complexity in the input instance. They do not account for *combined complexity*, which may be exponential in the instance treewidth, and non-elementary in the GSO query [Meyer 1975]. It may be possible, however, to achieve lower combined complexities when we restrict to more limited query languages. In particular, maybe this can be done in the more recent formalism of *monadic Datalog* [Gottlob, Pichler, and Wei 2010], which has been applied to counting and enumeration problems [Pichler, Rümmele, and Woltran 2010], but not yet to provenance, as far as I know.

Query-based vs instance-based methods. We now know that probabilistic query evaluation can be made tractable when the query is safe [Dalvi and Suciu 2012] or when the instance has bounded-treewidth (Part I), and is intractable otherwise. Is it possible to design probabilistic query evaluation techniques that would depend *both* on the instance and query, and identify tractable cases where neither the query nor the instance family would be tractable in isolation? Ideally, such a joint criterion should imply both the tractability of safe queries and the tractability of bounded-treewidth instances, while implying new tractability results when restricting both the instances and queries.

The notion of *unfolding* introduced in Section 4.5 could be a first tool to develop such a criterion, by understanding in which cases one can unfold the instance while respecting the query, maybe connected to a notion of *query-dependent treewidth*, i.e., treewidth that only respects instance joins that matter for the query. One could also investigate whether unfolding could not be a method for *approximate* query evaluation, by unfolding the instance to be treelike even when breaking some matches; it would be interesting to study how this idea relates to the notion of *dissociation* [Gatterbauer and Suciu 2015] from the query-based perspective.

Last, to develop practical evaluation schemes in generally intractable cases, it would be interesting to study how our exact methods, or approximations based on them, can be combined with *sampling-based* techniques, which apply to any instances and queries but are not exact. There have been initial efforts in this direction [Maniu, Cheng, and Senellart 2014] but this approach could be further extended.

2 Perspectives on Open-World Query Answering

This section presents future direction for my work on open-world query answering, both in the unrestricted case [Amarilli and Benedikt 2015a] and in the finite case (Part II).

Expressive frontier-one constraints. I have shown with Michael Benedikt [Amarilli and Benedikt 2015a] that open-world query answering (in the unrestricted case) is decidable when we allow both frontier-one existential rules with a certain restriction [Baget, Leclère, Mugnier, and Salvat 2009] and highly expressive rules on arity-two predicates: namely, the guarded two-variable fragment of first order logic with counting quantifiers, GC^2 [Pratt-Hartmann 2009], which capture many expressive description logics such as $\mathcal{ALCQI}b$ [Tobies 2001].

The resulting language, however, is not very uniform, because its syntax only allows constraints that are written in either formalism. One could then ask whether there could be a general language with a unified syntax that would still enjoy decidable open-world query answering, e.g., extending frontier-one rules with disjunction, negation, etc., or alternatively extending GC^2 to allow expressive "frontier-one" formulae on higher-arity predicates, and non-conflicting functional dependencies in some sense. We do not understand either whether our results could be extended to cover some non-GC^2 features that are decidable in the arity-two context, such as transitivity [Glimm, Horrocks, Lutz, and Sattler 2008] or nominals [Calvanese, Eiter, and Ortiz 2009; Rudolph and Glimm 2010].

Our study has also been mostly limited to the decidability of query answering, and not on complexity. We do not know which fragments of existential rules can be added to which fragments of description logics while ensuring that the complexity remains tractable.

Finite query answering. For the specific question of finite query answering, studied in Part II, the main question is whether our results could extend to more expressive languages. We have chosen to remain in the database context of functional dependencies and inclusion dependencies, but a natural question would be to study whether the language used in [Ibáñez-García, Lutz, and Schneider 2014] could be generalized to higher-arity, and whether we could show the decidability of finite query answering for such a language, capturing both our current results and theirs. In particular, could we generalize [Cosmadakis, Kanellakis, and Vardi 1990] to show a finite closure procedure for such a generalized language?

Another interesting angle would be to go back to the arity-two context, where our higher-arity functional dependencies translate to *path functional dependencies* [Weddell 1989; Toman and Weddell 2005], and study the decidability of finite implication and query answering for such dependencies. Maybe our decidability result could be rephrased and generalized in this way.

More generally, our understanding of finite open-world query answering is still quite unsatisfactory when compared to the unrestricted variant. In the unrestricted case, for instance, we can use the non-conflicting condition [Calì, Lembo, and Rosati 2003a; Calì, Gottlob, and Pieris 2012] to show that query answering is decidable. For finite query answering, we know that it does not suffice [Rosati 2006, Theorem 7], intuitively because the finite closure of non-conflicting dependencies may not itself be non-conflicting. However, could there be a requirement that would be stronger than the non-conflicting condition, and ensures that the finite closure of dependencies can be computed and still satisfies the condition? If yes, this would be a first step to show the decidability of finite query answering for more expressive constraint classes.

3 Long-Term Plans: Keeping Track of Provenance for Reasoning

I have presented several directions for future work suggested by my PhD research. I conclude this manuscript with a brief description of a more general and ambitious plan for future research: the problem of open-world query answering and reasoning while maintaining *provenance* information.

Many new structured data sources have appeared on the Web in the last few years: general-purpose sources like DBpedia [Bizer et al. 2009], YAGO [Suchanek, Kasneci, and Weikum 2007], or more recently Wikidata [Vrandečić and Krötzsch 2014]; domain-specific sources like OpenStreetMaps [Haklay and Weber 2008]; datasets extracted from semantic annotations such as Web Data Commons [Mühleisen and Bizer 2012]; open datasets such as `https://data.gov/` [GSA 2015] or `https://data.gouv.fr/` [Etalab 2015]. We can use open-world query answering to reason over such sources and compute query results under logical constraints, but we need to account for *uncertainty*: Web data is often incorrect, and the rules that we use to reason on such data may be incorrect as well, e.g., they may come from data mining or machine learning.

I believe that this problem should be addressed by extending open-world query answering techniques to incorporate *provenance management*, to represent in a generic and expressive way the initial facts and the deduction rules that were used. Indeed, if the results of open-world query answering are annotated by provenance information, we could use the annotations to estimate the reliability of results, for instance with probabilistic methods as in Part I, or with coarser methods to work around intractability issues. We could also use these annotations for the many other applications of provenance (see Chapter 5).

However, there are many new challenges that must be faced to extend provenance management techniques to the open-world query answering problem, where query evaluation is performed by reasoning on the initial data and logical constraints. How can we extend expressive provenance representations, such as semiring-based provenance, and represent the possibly complex and numerous ways through which an open-world query answer can be deduced from the initial facts and rules? Can provenance defined on Datalog proof trees be extended to more general representations of *proofs* for logical constraint languages, while preserving desirable properties such as commutation with homomorphisms? Could such an expressive provenance be computed efficiently, and which models (such as provenance circuits [Deutch, Milo, Roy, and Tannen 2014]) can be used to represent it concisely? Could such a provenance represent the dependence on missing values or `NULL`s, for incomplete data sources, or indicate how query results would change if new facts were added to our open-world sources?

There are also many challenges to address in order to *use* such provenance annotations, for instance to determine which answers are correct and relevant. Assume that the user gives us a coarse notion of preference, relevance, recency, or reliability, by defining an order relation on initial facts to indicate which ones are better than others. How can we propagate this order information to the complex annotations of the query results, to determine which ones of the results are more accurate? For probabilistic representations, can we define principled probabilistic models in open-world contexts where an arbitrarily high number of new facts can be derived, and can we perform probabilistic computations efficiently in such situations, avoiding intractability? A natural question is also to generalize such methods to cases where the *reasoning rules* themselves are probabilistic: repeated application of rules should make our confidence in the results decrease, but multiple independent derivations of an answer should make our confidence increase. Another extension would be to ask for user feedback to verify query answers or solve inconsistencies: which questions should we ask to the users, and how can we make sense of their

feedback in terms of the original data?

My vision for uncertainty management for Web data is thus that it should be attacked through general and expressive provenance representations, such as semiring-based ones; that it should apply, beyond query evaluation, to open-world query answering under expressive (yet decidable) logical constraints; and that it should be used, beyond quantitative probability evaluation, to the many other applications of provenance, in particular to avoid intractability.

Appendix A

Unrelated Work

This appendix reviews some additional work that I have performed during my PhD, but which is not directly relevant to the topic of my thesis. Please refer to the end of the Introduction on page 11 for more details.

Complexity of Taxonomy-Based Crowd Mining

I have studied with Yael Amsterdamer and Tova Milo the problem of mining frequent itemsets by asking questions to the crowd. This amounts to learning a monotone predicate on a distributive lattice, using oracle queries. We studied the computational complexity of this problem and proposed algorithms for this task.

This work was presented at the ICDT'14 conference [Amarilli, Amsterdamer, and Milo 2014a].

Web Entity Extraction Using Unique Identifiers

I have collaborated with Fabian M. Suchanek to a research work by Aliaksandr Talaika and Joanna Biega on the topic of Web entity extraction schemes relying on unique identifiers with a fixed structure, such as ISBNs, GTINs, DOIs, email addresses, etc. This comparatively simple approach, with some preprocessing steps, was able to harvest millions of unique entities from a corpus of Web pages.

The work was presented at the WebDB'15 workshop of SIGMOD [Talaika, Biega, Amarilli, and Suchanek 2015].

Possibility for Probabilistic XML

I have studied the tractability of the possibility problem for probabilistic XML documents in various formalisms, namely, determining whether a given XML document is a possible world of a probabilistic XML document, and optionally computing the probability of this outcome. My work established the tractability boundary between tractable and intractable phrasings of this problem.

The work was presented at the AMW'14 workshop [Amarilli 2014], and an extended version was published as a journal article [Amarilli 2015a].

Recent Topics of Research around YAGO

I have collaborated with Luis Galárraga, Nicoleta Preda, and Fabian M. Suchanek, to write an invited paper to APWEB'14 [Amarilli, Galárraga, Preda, and Suchanek 2014], presented by Fabian, on the topic of recent research around the YAGO knowledge base [Suchanek, Kasneci, and Weikum 2007], to present my pre-doctoral work on large-scale ontology alignment using YAGO.

XML Pricing Schemes

I have worked with Tang Ruiming, Stéphane Bressan, and Pierre Senellart on the topic of pricing schemes for XML documents. The goal of these schemes is to allow interested users to buy only a sample of the dataset, at a fraction of the cost, and we studied efficient sampling schemes for trees to address this problem.

This work was published as [Tang, Amarilli, Senellart, and Bressan 2014; Tang, Amarilli, Senellart, and Bressan 2016].

Uncertainty, Intensionality, and Structure

I have collaborated with Pierre Senellart and Silviu Maniu on an invited paper for the SIGWEB Newsletter, presenting the topic of *intensional data* [Amarilli, Maniu, and Senellart 2015].

Appendice B

Résumé en français

This appendix is a summary in French of my main contributions during my PhD. It is a translation of the General Introduction and of Appendix A. It also includes an English-to-French lexicon which summarizes the French translations used for technical terms. This appendix does not present any additional scientific content relative to the rest of the thesis and may be safely skipped.

Ce résumé en français présente de manière synthétique les principales contributions scientifiques exposées en détail dans notre manuscrit de thèse, et résume notre recherche de doctorat en un sens plus large. Il suit le plan de l'introduction générale en anglais située au début du manuscrit ainsi que l'annexe A, et se termine par un lexique qui récapitule les traductions utilisées pour les termes techniques.

La plus grande réussite de la recherche en bases de données est le modèle relationnel [Codd 1970]. Ce modèle a donné naissance à un domaine entier de recherche théorique, tout en continuant à avoir aujourd'hui une importance pratique considérable. Les systèmes de gestion de bases de données relationnelles constituent désormais une abstraction omniprésente utilisée en informatique, au même titre que les compilateurs ou les systèmes de fichiers : ils fournissent une interface déclarative de haut niveau que les applications peuvent utiliser, et s'appuient sur de nombreuses optimisations implantées dans les moteurs de base de données, qui sont efficaces et sont toutefois suffisamment génériques pour ne pas dépendre du domaine d'application.

Malheureusement, le modèle relationnel ne permet pas de gérer l'*incertitude* sur les données qu'il représente. En particulier, il ne peut pas exprimer les notions suivantes de façon satisfaisante.

Valeurs manquantes. Dans une base de données d'utilisateurs, par exemple, il est possible que certains utilisateurs n'aient pas rempli certains champs au moment de leur inscription, ou que de nouveaux champs aient été ajoutés depuis qu'ils se sont inscrits.

Données incomplètes. Les données dont on dispose ne représentent pas toujours l'intégralité des informations disponibles ; par exemple, lors de l'extraction d'une base de données à partir du Web, il peut s'avérer nécessaire de répondre à une requête à partir du sous-ensemble des données qui ont déjà été extraites.

Données périmées. Les numéros de téléphone et les adresses, par exemple, deviennent souvent incorrects avec le temps.

Données erronées. Les données peuvent être simplement inexactes en premier lieu, pour diverses raisons : elles peuvent avoir été mal saisies, s'appuyer sur des suppositions incorrectes, etc.

Pourtant, la gestion de données incertaines est un problème dont l'importance se fait de plus en plus sentir. En effet, aujourd'hui, de nombreux jeux de données sont extraits à partir du texte en langue naturelle disponible sur le Web, en utilisant des méthodes d'extraction automatiques et faillibles [Carlson et al. 2010] ; d'autres sont intégrés à partir de sources diverses à travers des appariements de schémas incertains [Dong, Halevy, and Yu 2007] ; d'autres encore sont créés par des utilisateurs qui contribuent à des bases de connaissances collaboratives sans être nécessairement dignes de confiance [Vrandečić and Krötzsch 2014] ; d'autres encore sont inférés à partir de réponses imprécises fournies par des utilisateurs sur les plateformes d'externalisation ouverte à partir de la foule [Amsterdamer, Grossman, Milo, and Senellart 2013; Parameswaran et al. 2012] ; plus généralement, d'autres données sont produites par des techniques de fouille de données ou d'apprentissage. Contrairement aux bases de données conçues à la main, il n'est plus possible de supposer que les données ainsi obtenues ne contiennent pas d'erreurs, ou que l'on pourra corriger manuellement chacune de leurs erreurs dès qu'on les aura identifiées. Les données sont incomplètes et contiennent des inexactitudes, et c'est une réalité dont il faut tenir compte quand on les utilise.

Pour l'instant, cependant, il n'y a guère qu'un seul aspect de la gestion de données incertaines qui soit véritablement pris en compte par les systèmes de gestion de bases de données couramment utilisés en pratique : il s'agit du problème des *valeurs manquantes*, avec la notion de `NULL`s dans le langage SQL. Malheureusement, de nombreuses difficultés se cachent dans la sémantique des `NULL`s ainsi que le standard SQL la définit [ISO 2008], et certaines de ces difficultés ont été identifiées très tôt [Grant 1977]. On peut notamment déplorer les problèmes suivants :

- Les `NULL`s peuvent uniquement représenter les valeurs manquantes dans les enregistrements : ils ne peuvent pas représenter les enregistrements manquants, ou indiquer qu'un enregistrement entier est susceptible d'être incorrect.
- Les `NULL`s ne peuvent pas indiquer que plusieurs valeurs inconnues sont les mêmes, ou qu'elles doivent obéir à certaines contraintes, par exemple, qu'elles sont choisies à partir de plusieurs valeurs possibles.
- L'évaluation des `NULL`s est fondée sur les logiques ternaires, dont la sémantique n'est pas particulièrement intuitive. Par exemple, si l'on applique l'opérateur relationnel de sélection pour retenir les valeurs qui sont soit *égales à* 42, soit *différentes de* 42, les valeurs `NULL` ne seront pas sélectionnées, même si elles devraient en principe être retenues quelle que soit la valeur réelle inconnue que la valeur `NULL` représente.

Bien entendu, de nombreuses approches ont été proposées pour définir des sémantiques améliorées pour les `NULL`s [Imieliński and Lipski 1984], et la recherche d'une solution satisfaisante se poursuit encore aujourd'hui [Libkin 2014]. Pour l'instant, cependant, aucune méthode générique n'est disponible pour les applications qui voudraient gérer des données incertaines au sens large. C'est pour cette raison que

les approches actuelles de gestion de l'incertitude sont souvent très élémentaires : une méthode fréquente en pratique consiste à définir un seuil de confiance, à éliminer toutes les réponses dont la confiance est inférieure au seuil, et à considérer qu'une réponse est certaine dès lors qu'elle atteint le seuil fixé. Des méthodes plus raffinées se rencontrent néanmoins dans certains domaines où la gestion de l'incertitude est tout à fait indispensable (la reconnaissance optique de caractères, la reconnaissance vocale, la bioinformatique, etc.), où des solutions puissantes de gestion de l'incertitude ont été développées (comme les transducteurs finis pondérés [Mohri, Pereira, and Riley 2002] pour la reconnaissance vocale). Malgré tout, ces méthodes ont le défaut d'être *spécifiques à un domaine d'application*. La vision générale de la recherche en *gestion de données incertaines* serait au contraire de développer des outils génériques pour la gestion de données incomplètes et bruitées, qui pourraient être utilisés dans tous les domaines, suivant des principes opérationnels bien définis ; cette vision est le prolongement naturel des bases de données relationnelles actuelles, qui sont soigneusement optimisées et pourtant utilisables pour d'innombrables applications.

La communauté de la recherche en base de données s'efforce aujourd'hui de parvenir à atteindre cet objectif ambitieux. De nombreux modèles théoriques ont ainsi été proposés pour généraliser les `NULL`s et gérer les données incertaines d'une manière à la fois générique et puissante. Par exemple, la gestion de l'*incomplétude* des bases de données a été formalisée comme le problème de *réponse aux requêtes en monde ouvert* : lorsque l'on suit la *sémantique du monde ouvert*, les données qui ne sont pas exprimées dans la base de données sont considérées comme *inconnues* et non comme *incorrectes*, et la requête est évaluée en déterminant quelles réponses sont impliquées par les données et par les règles logiques que l'on a imposé sur celle-ci. D'une manière analogue, pour utiliser des tuples qui sont susceptibles d'être incorrects, des formalismes de *bases de données probabilistes* [Suciu, Olteanu, Ré, and Koch 2011] ont été proposés. Par exemple, les bases de données à tuples indépendants (TID) sont des bases de données relationnelles dont les tuples sont annotés par une valeur de probabilité : celle-ci indique quelles sont les chances que ce tuple existe effectivement, indépendamment des autres tuples. Pour représenter des *corrélations* entre tuples, ou pour représenter le résultat de l'évaluation d'une requête sur des bases de données probabilistes, des formalismes plus expressifs ont été proposés, comme les *pc-tables*. D'autres formalismes ont été proposés pour d'autres types de données incertaines, comme les *documents XML probabilistes* [Kimelfeld and Senellart 2013].

Il reste cependant de nombreux défis à relever pour pouvoir réellement gérer des données incertaines. Il est encore délicat de définir des formalismes bien-fondés pour gérer des données incertaines ou probabilistes sur des structures de données inhabituelles, et il est encore plus délicat d'en définir des représentations *concises*. L'utilisation des probabilités pose des problèmes spécifiques d'*efficacité*, car l'évaluation de requêtes sur des données probabilistes est souvent beaucoup plus complexe sur des données probabilistes que sur des données déterministes [Dalvi and Suciu 2007]. Par ailleurs, dans le contexte de la réponse aux requêtes en monde ouvert avec des règles logiques, le problème de raisonnement peut même devenir *indécidable*, en fonction de l'expressivité du fragment logique dans lequel les règles sont exprimées.

Au cours de notre travail de doctorat, nous nous sommes intéressé à ces problèmes

sous un angle nouveau. Plutôt que d'étudier la gestion de données *arbitraires*, en cherchant à déterminer quelles tâches peuvent *toujours* être traitées efficacement sur de telles données (par exemple, les requêtes *prudentes* [Dalvi and Suciu 2012] pour le formalisme TID), nous avons étudié quelles sont les hypothèses que l'on pouvait faire sur la *structure* des données. Cette approche est une direction naturelle si l'on désire gérer des données probabilistes en pratique : les données utilisées dans le monde réel ne sont pas quelconques, et on peut donc s'attendre à ce que de nombreuses tâches soient faisables sur de telles données même si elles sont infaisables en général. Plus spécifiquement, au cours de notre doctorat, notre recherche s'est effectuée suivant trois axes principaux :

1. Nous avons étudié quelles conditions structurelles sur les instances relationnelles permettent d'assurer la faisabilité de la gestion de l'incertitude et des probabilités ; nous nous sommes notamment intéressé à des bornes sur la *largeur d'arbre* des données.

2. Nous avons étudié de nouveaux formalismes pour représenter l'incertitude sur des structures de données inhabituelles, plus précisément, nous avons montré comment gérer des *ordres incertains* sur des tuples et sur des valeurs.

3. Nous avons travaillé sur la *réponse aux requêtes en monde ouvert* pour les données incomplètes, et identifié de nouveaux cas où la structure des règles logiques, et le fragment logique précis utilisé, pouvait garantir la décidabilité du problème de raisonnement ; nous avons étudié entre autres l'*hypothèse de finitude* que l'on fait généralement dans le contexte des bases de données.

Dans les trois premières sections de ce résumé en français, nous présenterons le travail que nous avons effectué sur ces trois thèmes pendant notre doctorat. Le manuscrit de thèse lui-même se concentre sur deux de ces contributions : nous précisons cela au fil des prochaines sections et le récapitulons à la fin. Certaines parties de cette synthèse de notre travail de doctorat ont été publiées au congrès de doctorants SIGMOD/PODS [Amarilli 2015b].

1 Provenance et probabilités sur les instances quasi-arborescentes

Le premier axe de notre recherche concerne la faisabilité de certaines tâches de gestion des données incertaines dans le contexte des bases de données quasi-arborescentes : nous nous sommes en particulier intéressé au calcul de la provenance et à l'évaluation probabiliste. Ce travail est présenté dans la partie I de notre manuscrit ; il a été également présenté à ICALP'15 [Amarilli, Bourhis, and Senellart 2015] et a été accepté pour présentation à PODS'16 [Amarilli, Bourhis, and Senellart 2016].

Une des principales surprises en matière de données incertaines est que l'évaluation de requêtes sur des données probabilistes est considérablement plus délicate que l'évaluation de requêtes sur des instances déterministes. Ce problème se pose même pour des langages de requêtes relativement simples, comme les *requêtes conjonctives*

(CQ), et même sur les modèles probabilistes les moins expressifs, comme le modèle TID présenté plus haut.

Considérons par exemple un site Web de rencontres : le site dispose d'une table qui représente ses utilisateurs et indique leur ville de résidence, et une table qui regroupe les messages échangés par les utilisateurs. Considérons une requête conjonctive qui demande de calculer les paires d'utilisateurs qui vivent dans la même ville et qui se sont envoyé un message. L'évaluation de cette requête est une tâche faisable en fonction de la base de données (ce que l'on appelle ici la *complexité en fonction des données*) ; plus précisément, ce problème est dans la classe de complexité AC^0 [Abiteboul, Hull, and Vianu 1995], et cette borne est en réalité valable pour n'importe quelle requête qui peut être exprimée en logique du premier ordre.

Imaginons à présent que le site de rencontres utilise le formalisme TID pour représenter des données *incertaines* à propos de ses utilisateurs : le site ne sait pas déterminer quels utilisateurs sont véritablement à la recherche d'un partenaire, et quels messages échangés expriment un ressenti positif. La probabilité de ces faits pourrait par exemple être estimée, pour chaque utilisateur et pour chaque message, par des approches à base d'apprentissage ou d'analyse de sentiments, en faisant l'hypothèse que ces valeurs sont indépendantes entre utilisateurs et entre messages. Le site de rencontres désire utiliser ces informations pour calculer la *probabilité* qu'un utilisateur à la recherche d'un partenaire ait envoyé un message à un autre utilisateur dans cette situation, en imposant que le message exprime un sentiment positif, et que les deux utilisateurs habitent dans la même ville. Malheureusement, il s'avère que cette tâche n'est pas faisable avec une complexité raisonnable : plus précisément, elle est #P-difficile en général. Intuitivement, il est peu probable que cette tâche puisse être résolue beaucoup plus efficacement qu'en considérant le nombre exponentiel de mondes possibles ; ce problème provient des corrélations qui apparaissent entre les occurrences multiples des faits dans les résultats possibles de la requête.

Ce problème a déjà été étudié par des travaux antérieurs [Dalvi and Suciu 2012], qui ont abouti à un résultat de *dichotomie*. Ce résultat permet de caractériser les requêtes pour lesquelles le problème d'évaluation sur des données probabilistes (dans le formalisme TID) est faisable sur n'importe quelle instance fournie en entrée. Toutefois, ce résultat n'exclut pas que des requêtes plus expressives soient efficacement évaluables lorsque l'on impose que les instances d'entrée soient *simples*, en un sens à définir. En conséquence, comme les jeux de données réels ne sont pas arbitraires, peut-être pourrait-on interroger de telles données d'une autre manière, en contournant les résultats généraux d'infaisabilité pour l'évaluation sur des bases de données probabilistes quelconques. En fait, un résultat de ce type a déjà été démontré pour le modèle de XML probabiliste qui correspond aux bases de données TID (c'est-à-dire le modèle $\mathrm{PrXML}^{\mathsf{mux,ind}}$) : l'évaluation de requêtes est faisable dans ce contexte [Cohen, Kimelfeld, and Sagiv 2009], l'intuition étant que les choix probabilistes n'ont qu'un effet local dans la structure d'arbre.

Ceci nous conduit à la question précise que nous avons étudiée dans cette partie de notre travail de thèse : *sur quelles familles d'instances probabilistes peut-on efficacement évaluer des requêtes expressives, lorsque l'on mesure la complexité en fonction des données ?*

La partie I de notre manuscrit propose une réponse à cette question : nous y démontrons un nouveau résultat de dichotomie qui porte sur la structure des instances plutôt que sur celle des requêtes. Plus précisément, nous montrons que l'évaluation de requêtes sur les instances TID de *largeur d'arbre bornée* (qui sont intuitivement semblables à un arbre) est efficacement faisable : ce résultat s'applique à toutes les requêtes qui peuvent être exprimées dans le fragment expressif de la *logique gardée du second ordre* (GSO). À l'inverse, nous montrons l'existence de requêtes en *logique du premier ordre* telles que, pour toute famille d'instances dont la largeur d'arbre n'est pas bornée, l'évaluation de requêtes sous annotations de probabilité n'est pas efficacement faisable en fonction des données : ce résultat s'applique si l'on suppose que la signature est d'arité deux et que la famille d'instances d'entrée est efficacement constructible en un certain sens.

La borne supérieure de notre dichotomie est obtenue à partir d'une technique générale pour calculer efficacement des représentations de la *provenance*, ou du lignage, sur des instances de largeur d'arbre bornée. Cette technique est présentée dans le chapitre 3 de notre manuscrit. Notre résultat s'appuie sur la correspondance introduite par Courcelle entre les requêtes évaluées sur les instances quasi-arborescentes, et les automates d'arbres évalués sur des encodages en arbre [Courcelle 1990; Flum, Frick, and Grohe 2002]. Nous proposons une représentation générale de l'ensemble des exécutions possibles d'un automate d'arbres sur un arbre incertain, sous la forme d'un *circuit de provenance* [Deutch, Milo, Roy, and Tannen 2014], et nous généralisons cette représentation pour calculer la provenance de requêtes GSO sur les instances quasi-arborescentes.

Nous montrons ensuite comment cette provenance peut être représentée sous des formes compactes, à savoir des *OBDD* [Bryant 1992; Olteanu and Huang 2008] et des *d-DNNF* [Darwiche 2001]. Nous utilisons ces résultats dans le chapitre 4 pour montrer que l'évaluation probabiliste de requêtes GSO sur des instances TID de largeur d'arbre bornée peut être effectuée en temps ra-linéaire en fonction des données, c'est-à-dire en temps linéaire au coût des opérations arithmétiques près. Pour étendre ces résultats à des formalismes plus expressifs, nous définissons une nouvelle notion de *pcc-instances*, une variante des pc-tables [Huang, Antova, Koch, and Olteanu 2009; Green and Tannen 2006] à base de circuits, avec une définition naturelle de largeur d'arbre qui couvre à la fois l'instance et les corrélations. Nous utilisons cette notion pour obtenir des résultats de faisabilité à largeur d'arbre bornée (en un certain sens) pour les pc-tables, les tables BID [Barbará, Garcia-Molina, and Porter 1992; Ré and Suciu 2007] et le XML probabiliste, en généralisant en particulier le résultat de [Cohen, Kimelfeld, and Sagiv 2009]. Nous pouvons également identifier un lien avec les résultats de [Dalvi and Suciu 2012], en montrant que la faisabilité des unions de requêtes conjonctives *sans inversion* peut être prouvée à l'aide d'une récriture de leurs instances d'entrée qui préserve le lignage et assure que le résultat de la transformation est de *largeur linéaire* bornée.

Le chapitre 5 démontre ensuite des résultats à propos du calcul efficace, sur les instances quasi-arborescentes, de représentations de la provenance dans des semi-anneaux expressifs, au-delà des lignages booléens. Nous établissons pour cela un lien avec la provenance à base de semi-anneaux [Green, Karvounarakis, and Tannen 2007]. En effet, les représentations de la provenance ne servent pas uniquement à

l'évaluation probabiliste de requêtes : elles nous permettent également de garder trace du lien entre le résultat de la requête et l'instance de départ, et peut être utilisée pour résoudre de nombreux autres problèmes : comptage, gestion de droit d'accès, maintenance de vues, etc. À ces fins, nous généralisons nos définitions de la provenance pour les arbres et les instances quasi-arborescentes, pour leur permettre de s'appliquer à des semi-anneaux plus généraux dans lesquels nous pouvons résoudre ces problèmes. Nous montrons ensuite, à l'aide d'une variante de nos techniques, que, dans le cas des instances quasi-arborescentes et des unions de requêtes conjonctives, il est possible de calculer des circuits de provenance pour le semi-anneau universel $\mathbb{N}[X]$ en complexité linéaire en fonction des données : cette borne est meilleure que celle obtenue dans le cas d'instances quelconques, où la complexité est polynomiale.

Nous montrons la borne inférieure de notre résultat de dichotomie au chapitre 6, en exploitant les résultats récemment obtenus par [Chekuri and Chuzhoy 2014a] à savoir, une borne polynomiale pour l'extraction de graphes planaires comme mineurs de graphes de haute largeur d'arbre. Nous appliquons ce résultat à des familles arbitraires de graphes dont la largeur d'arbre n'est pas bornée, à condition qu'elles soient constructibles en un certain sens. Ceci nous permet de montrer, sur les signatures d'arité deux, que l'évaluation probabiliste pour une requête spécifique de la logique du premier ordre (FO) est #P-difficile sous réductions RP sur *n'importe quelle* famille d'instances d'entrée qui satisfasse ces conditions. Autrement dit, si l'on souhaite assurer la faisabilité de l'évaluation probabiliste de requêtes FO en imposant des conditions sur les instances d'entrée, ces conditions doivent nécessairement impliquer une borne sur la largeur d'arbre des instances, ou leur inconstructibilité. Cette utilisation de FO contraste avec les résultats similaires que nous pouvons obtenir pour l'évaluation non-probabiliste et le comptage de correspondances, où nous utilisons une requête en logique monadique du second ordre : nous nous appuyons pour ce faire sur les méthodes développées par [Kreutzer and Tazari 2010; Ganian, Hliněný, et al. 2014], et améliorons leurs résultats.

Nous étendons ensuite nos résultats à une dichotomie plus forte qui s'applique à un contexte plus exigeant : nous souhaitons imposer une condition sur les instances d'entrée qui assure que l'on pourra calculer une représentation du lignage des requêtes sous la forme d'*OBDD suffisamment concis*. Nous savons déjà que borner la largeur d'arbre suffit à ce que cette condition soit remplie pour toutes les requêtes GSO : nous montrons cette fois que, pour une certaine requête dans la classe plus restreinte $\mathrm{UCQ}^{\neq}$ (c'est-à-dire une union de requêtes conjonctives avec inégalités), il ne peut pas y avoir de représentation concise sous forme d'OBDD du lignage de la requête sur *n'importe quelle* famille d'instances d'entrée d'arité deux, du moment que la largeur d'arbre n'est pas bornée et que la famille est constructible en un certain sens. Nous terminons notre étude en caractérisant quelles requêtes $\mathrm{UCQ}^{\neq}$ *connexes* sont infaisables (en termes d'OBDD) sur n'importe quelle famille d'instances d'entrée de ce type, à travers une *méta-dichotomie* sur les requêtes.

Nous espérons que nos résultats de dichotomie pourront mener à une nouvelle étude de l'évaluation probabiliste de requêtes, qui prenne en compte à la fois les instances et les requêtes. L'objectif serait de combiner des constructions efficaces pour les instances de largeur d'arbre bornée avec les techniques connues pour l'évaluation de requêtes *prudentes*, et de parvenir à l'évaluation probabiliste efficace de requêtes

(à la fois en théorie et en pratique), dans des situations où la faisabilité peut être assurée en étudiant l'*interaction* entre la requête et l'instance, mais où il ne suffit pas de considérer individuellement les instances ou la requête.

2 Incertitude sur les données ordonnées

Le deuxième axe de notre recherche de doctorat s'intéresse à l'incertitude sur des données munies d'une relation d'ordre, qui porte sur les tuples ou sur des valeurs numériques. Ce travail n'est pas détaillé dans notre manuscrit de thèse : le contenu de la première sous-section est disponible sous la forme d'une prépublication [Amarilli, Ba, Deutch, and Senellart 2016] ; le contenu de la seconde sous-section est également disponible sous cette forme [Amarilli, Amsterdamer, Milo, and Senellart 2016] et a été esquissé à UnCrowd [Amarilli, Amsterdamer, and Milo 2014b].

De nombreux contextes de gestion de données nous demandent de prendre en compte une relation d'*ordre* sur les valeurs (par exemple des dates ou des entiers naturels) ou sur les tuples eux-mêmes (par exemple une liste d'événements dans un fichier journal). Dans cette section, nous présentons notre travail sur la représentation de l'incertitude sur des informations d'ordre. Nous étudions en premier lieu un contexte où l'incertitude porte sur l'ordre de tuples relationnels, pour nous intéresser ensuite aux situations où l'on dispose d'une relation d'ordre partiel sur des valeurs numériques.

2.1 Représentation de l'incertitude sur l'ordre de tuples relationnels

Lorsque l'on souhaite gérer des tuples ordonnés dans le modèle relationnel, une difficulté intéressante se pose : une forme d'incertitude sur l'ordre apparaît spontanément, même lorsque l'on applique les opérateurs habituels de l'algèbre relationnelle sur des relations d'entrée dont l'ordre est certain. Pour illustrer ce problème dans une situation concrète, considérons un site Web qui permet de rechercher des hôtels et de les trier par note. Un utilisateur souhaite réserver un hébergement pour quatre personnes, qui devrait proposer une chambre avec quatre lits, ou deux chambres disposant chacune de deux lits. Malheureusement, le site Web ne permet pas d'effectuer une recherche portant simultanément sur ces deux types d'hébergement. L'utilisateur va donc effectuer les deux recherches séparément, l'une après l'autre, et obtiendra ainsi deux listes triées d'hôtels correspondant à chaque critère. L'utilisateur souhaite donc intégrer ces listes, en calculant leur union au sens relationnel. Cependant, l'*ordre* sur le résultat de cette union n'est pas certain : il dépend des critères exacts de notation utilisés par le site Web, que l'utilisateur ne contrôle pas et ne connaît pas. L'ordre n'est toutefois pas complètement inconnu, car deux hôtels qui n'apparaissaient que dans les résultats de la première recherche (ou de la seconde) devraient conserver le même ordre relatif dans la liste obtenue comme résultat de l'union.

Notre travail [Amarilli, Ba, Deutch, and Senellart 2016] permet de raisonner sur de tels problèmes. Nous proposons ainsi des opérateurs pour l'algèbre relationnelle qui s'appliquent à des relations partiellement ordonnées, avec une sémantique *multiensembliste*, dans l'esprit de travaux antérieurs [Grumbach and Milo 1999]. Ces

opérateurs nous permettent de combiner des relations totalement ordonnées ou partiellement ordonnées à l'aide de l'algèbre relationnelle, afin d'intégrer les données et de calculer de nouvelles relations ordonnées, qui représentent tous les choix possibles d'ordre sur le résultat de l'intégration qui sont compatibles avec l'ordre connu sur les données d'entrée. Cette tâche ressemble au problème d'*agrégation de rangs*, qui vise à réconcilier des ordres différents sur les mêmes résultats, et pour lequel de nombreuses techniques sont déjà connues [Fagin, Lotem, and Naor 2001; Jacob, Kimelfeld, and Stoyanovich 2014; Dwork, Kumar, Naor, and Sivakumar 2001] ; la principale différence est que ces méthodes sont *quantitatives*, c'est-à-dire qu'elles tendront à faire apparaître au début du résultat global les éléments qui apparaissent *fréquemment* au début des ordres fournis en entrée. Nous souhaitons au contraire obtenir une représentation de *tous* les ordres cohérents sur le résultat. Bien sûr, ce problème pourrait en principe être résolu avec les formalismes existants de représentation de l'incertitude sur des données relationnelles, par exemple les c-tables, qui permettraient de représenter l'incertitude sur la position des tuples du résultat. Cependant, cette méthode ne serait pas commode, car ces formalismes ne permettent pas de représenter aisément les dépendances entre les rangs numériques qui sont induites par les relations de comparabilité dans l'ordre (si l'on sait, par exemple, qu'un tuple doit apparaître *avant* un autre dans le résultat).

La sémantique que nous définissons pour l'algèbre relationnelle nous permet ainsi de combiner des données ordonnées d'une façon générique et bien définie. Cependant, l'ordre n'est propagé que comme une information implicite, et les opérateurs que nous définissons ne permettent pas de l'*interroger*. Pour pallier cette lacune, nous proposons un opérateur général d'*accumulation* sur les relations à ordre incertain. Le résultat de cet opérateur dépend de l'ordre, et nous pouvons l'utiliser pour calculer ce que nous désirons savoir à propos de l'ordre du résultat. L'opérateur d'accumulation calcule simplement toutes les concaténations possibles des valeurs de tuples dans un monoïde donné, suivant tous les ordres autorisés sur le résultat. Nous pouvons utiliser cet opérateur pour répondre à des questions, par exemple, « est-il certain que tous les hôtels d'un quartier donné sont préférables à tous les hôtels d'un autre quartier ? ».

La contribution technique principale de notre travail est l'étude de la *complexité* de l'évaluation de requêtes dans ce cadre. Plus spécifiquement, nous nous intéressons à la complexité du calcul de réponses *possibles* et de réponses *certaines* en termes de possibilité et de certitude des *instances* [Antova, Koch, and Olteanu 2007], y compris pour des requêtes exprimées avec notre opérateur d'accumulation. Nous prouvons d'abord un résultat relativement surprenant : il est déjà infaisable (c'est-à-dire, respectivement, NP-difficile et coNP-difficile) de déterminer si un résultat complet (à savoir, une relation totalement ordonnée) est possible (respectivement, s'il est certain), et ce même pour des requêtes très simples. Nous montrons toutefois qu'il est faisable de déterminer la certitude (au contraire de la possibilité) pour des requêtes qui ne font pas appel à l'opérateur d'accumulation, ou qui calculent une accumulation dans un monoïde *cancellatif*.

Nous montrons ensuite comment éviter l'infaisabilité en limitant la *structure* des relations ordonnées fournies en entrée, pour des mesures de complexité définies à partir de notions existantes de la théorie des ordres partiels. Nous montrons ainsi

la faisabilité des problèmes de possibilité et de certitude que nous étudions, pour un certain sous-ensemble des opérateurs que nous définissons, lorsque les relations ordonnées d'entrée sont de *largeur* bornée au sens des ordres partiels [Brandstädt, Le, and Spinrad 1987] : nous établissons que ces bornes sont maintenues sur les résultats de l'évaluation de requêtes, et montrons la faisabilité des problèmes de possibilité et de certitude dans ce contexte, en concevant un algorithme dynamique. Nous montrons un résultat similaire lorsque les relations d'entrée ne sont presque pas ordonnées, ce que nous formalisons comme une nouvelle notion sur les ordres partiels, l'*ia-largeur*. Nous généralisons également nos résultats à un opérateur d'élimination des doublons, afin de passer de la sémantique multiensembliste à une sémantique ensembliste.

2.2 Complétion de valeurs numériques manquantes

Nous avons mené un autre travail [Amarilli, Amsterdamer, Milo, and Senellart 2016] qui étudie l'évaluation de *requêtes top-k* sur des valeurs numériques inconnues que l'on contraint par un ordre partiel et par certaines valeurs exactes connues.

Ce travail s'inspire de scénarios où l'on cherche à extraire de l'information à partir de la foule [Amsterdamer, Grossman, Milo, and Senellart 2013; Parameswaran et al. 2012], en posant des questions aux utilisateurs. Dans de nombreux cas, l'information que nous souhaitons obtenir peut être représentée comme des valeurs numériques avec des dépendances mutuelles. Imaginons par exemple que l'on cherche à catégoriser les produits du catalogue d'un site Web de vente en ligne, en déterminant la compatibilité de chaque produit avec chacune des catégories répertoriées dans une taxonomie qui nous est fournie. On suppose alors que les scores de compatibilité obéissent à un *ordre partiel* défini suivant la taxonomie : la catégorie « chemises » est plus spécifique que la catégorie « vêtements », donc si un objet est compatible avec la catégorie « chemises », il doit être également compatible avec la catégorie « vêtements ».

L'approche la plus simple pour catégoriser un objet serait de demander à la foule sa compatibilité avec chacune des catégories. Malheureusement, cette approche n'est pas raisonnable en général. En effet, chaque requête adressée à la foule a un coût, que ce soit en termes de latence (il faut attendre que les utilisateurs de la foule fournissent leurs réponses) ou en termes d'argent (il faut payer les utilisateurs pour chaque réponse). Cet écueil nous conduit à étudier des méthodes bien fondées d'*interpolation*, afin de compléter les valeurs numériques qui nous font défaut : nous pouvons utiliser à cette fin la *structure* d'ordre partiel que nous connaissons sur ces valeurs grâce à la taxonomie, ainsi que les valeurs connues que l'on a déjà récoltées à partir de la foule. L'objectif est de trouver les k éléments dont la valeur moyenne est la plus élevée (ici, les k catégories les plus compatibles avec le produit demandé) en exploitant l'information limitée dont on dispose.

Dans ce contexte, il est facile de compléter des valeurs numériques manquantes lorsqu'elles suivent un ordre *total*, car il suffit de procéder par interpolation linéaire. En revanche, la question d'interpoler suivant un ordre *partiel* sur les valeurs inconnues n'avait pas encore été étudiée. Nous proposons ainsi une définition formelle pour ce problème, qui consiste à calculer l'espérance de chaque variable inconnue, suivant la distribution uniforme, dans le polytope convexe qui est induit par les relations

de comparabilité de l'ordre partiel. Nous concevons un algorithme pour résoudre ce problème d'interpolation en $\mathrm{FP}^{\#\mathrm{P}}$, et nous montrons que cette tâche est #P-difficile, en utilisant des résultats connus de la théorie des ordres partiels [Brightwell and Winkler 1991]. Nous montrons même un résultat plus fort : il est déjà infaisable de déterminer quel objet a la plus grande espérance, même si on ne cherche pas à calculer la valeur de cette dernière. Nous montrons cependant que notre problème est faisable dans certaines situations. Nous concevons notamment un schéma d'approximation randomisé entièrement en temps polynomial pour ce problème, en utilisant une méthode connue d'échantillonnage dans les polytopes convexes [Kannan, Lovász, and Simonovits 1997]. De plus, nous montrons que l'interpolation (suivant notre définition formelle) est faisable pour les taxonomies en forme d'arbre, en concevant un algorithme dynamique adapté à cette situation où la *structure* est restreinte.

3 Décidabilité de la réponse aux requêtes en monde ouvert

Le troisième axe de notre recherche de doctorat étudie la réponse aux requêtes en monde ouvert. Ce problème porte sur des données incomplètes dont la structure est guidée par des contraintes logiques. Nous avons cherché à identifier de nouveaux fragments logiques pour lesquels cette tâche de raisonnement est décidable. Le travail présenté dans la première sous-section a été publié à IJCAI'15 [Amarilli and Benedikt 2015a] mais n'est pas présenté plus en détail dans notre manuscrit. Le travail de la deuxième sous-section est présenté dans la partie II de notre manuscrit ; il a été publié sous la forme d'un résumé étendu à LICS'15 [Amarilli and Benedikt 2015b] et une version complète est actuellement en cours de relecture par les pairs [Amarilli and Benedikt 2016].

3.1 Approches hybrides pour la réponse aux requêtes en monde ouvert

Le problème de *réponse aux requêtes en monde ouvert* (OWQA) est défini de la façon suivante : on considère une base de données I, des contraintes logiques Σ, et une requête q, et on cherche à calculer les réponses de q qui sont *certaines* sous Σ étant donné I. En d'autres termes, on cherche à déterminer quelles réponses à q sont vraies sur *toutes* les complétions possibles de I qui satisfont les contraintes Σ. Le problème OWQA nous permet ainsi de raisonner sur la base de données I en considérant qu'elle est *incomplète*, et en contraignant la *structure* de ses complétions suivant des règles logiques. Le problème OWQA est bien entendu indécidable si les contraintes logiques Σ sont arbitraires, et de nombreux travaux ont identifié des fragments logiques expressifs pour lesquels le problème OWQA était décidable, et pouvait même parfois être résolu avec une faible complexité.

Cette question de recherche a notamment été étudiée dans les communautés de recherche s'intéressant aux formalismes suivants :

Les bases de données relationnelles. Dans ce contexte, le problème OWQA a d'abord été formulé de manière équivalente comme un problème d'*inclusion de requêtes sous contraintes* [Johnson and Klug 1984], avant d'être étudié sous l'angle des bases de données incomplètes [Calì, Lembo, and Rosati 2003a].

Dans cette communauté, les règles logiques utilisées sont généralement les contraintes d'intégrité classiques utilisées en bases de données, par exemple, les dépendances génératrices de tuples (TGD) et les dépendances génératrices d'égalités (EGD) [Abiteboul, Hull, and Vianu 1995]. Il est rapidement apparu qu'il n'était pas décidable de combiner ces deux formes de dépendances [Mitchell 1983].

Les logiques de description. Ces formalismes logiques ont été définis afin de pouvoir raisonner avec une faible complexité en l'instance I. Leur principal objet est l'étude du compromis entre l'expressivité des contraintes logiques autorisées dans Σ et la complexité computationnelle du raisonnement.

En revanche, les logiques de description ne fonctionnent que sur des *données d'arité deux*, c'est-à-dire des graphes étiquetés. Cette représentation est moins expressive que les données relationnelles générales, mais elle permet d'assurer que les logiques de description restent décidables même en autorisant des contraintes logiques très expressives. Les logiques de description peuvent ainsi exprimer, entre autres, la disjonction, la négation, et les assertions de fonctionnalité (qui sont une forme de dépendances génératrices d'égalités).

Les règles existentielles. Ces règles reviennent essentiellement à exprimer des dépendances génératrices de tuples, mais des classes décidables spécifiques ont été étudiées pour elles, par exemple les règles à frontière gardée [Baget, Leclère, and Mugnier 2010].

En termes d'opérateurs, les règles existentielles ne sont pas aussi expressives que les logiques de description : elles peuvent seulement indiquer qu'une conjonction de faits implique une autre conjonction de faits. En revanche, elles s'appliquent à des faits d'arité arbitraire, au contraire des logiques de description.

Notre première contribution à l'étude du problème OWQA a été de tisser des liens entre les résultats portant sur les logiques de description et ceux portant sur les règles existentielles. Plus précisément, nous avons conçu des langages hybrides pour le raisonnement sur les données incomplètes, qui autorisent à la fois des règles existentielles et des contraintes exprimées en logique de description, afin d'obtenir le meilleur des deux mondes : nos langages hybrides peuvent exprimer des règles expressives sur les faits d'arité deux, et des règles existentielles portant sur les faits d'arité supérieure. Nous nous concentrons sur la *décidabilité* du problème OWQA, et choisissons ainsi de ne pas nous limiter à des logiques de description spécifiques : à la place, nous autorisons toutes les contraintes en arité deux qui peuvent être exprimées dans un formalisme logique très expressif. Ce formalisme est GC^2, le fragment gardé à deux variables de la logique du premier ordre avec quantificateurs à compteurs : il a été prouvé [Pratt-Hartmann 2009] que le problème OWQA pour GC^2 est décidable, ce qui couvre la décidabilité de ce même problème pour de nombreuses logiques de description (le lien entre GC^2 et les logiques de description est présenté par exemple dans [Kazakov 2004]).

Notre travail identifie quelles sont les fonctionnalités problématiques de ces deux familles de langages qui mènent à l'indécidabilité si on les combine. La principale fonctionnalité problématique en ce sens pour les logiques de description (respectivement, pour GC^2) est son support des *assertions de fonctionnalité* (respectivement,

des quantificateurs à compteurs), qui permettent d'exprimer, par exemple, qu'une personne a *un unique* lieu de naissance. Si l'on souhaite autoriser de telles contraintes, il faut renoncer à deux fonctionnalités problématiques des règles existentielles. La première est l'export de *variables multiples*, dans des règles comme, par exemple, « Si *une personne* est née dans *un pays* alors *cette personne* a vécu dans *ce pays* ». C'est pour cette raison que nous nous limitons au fragment à *frontière unaire* des règles existentielles [Baget, Leclère, Mugnier, and Salvat 2009], qui n'autorise que des règles où la tête n'utilise qu'une seule variable du corps : par exemple, « Si *quelqu'un* a gagné un prix littéraire alors *cette personne* a écrit un livre ». La seconde fonctionnalité problématique des règles existentielles est la possibilité d'écrire des motifs *cycliques* comme conséquence de règles à frontière unaire, qui mène également à l'indécidabilité ; nous introduisons un fragment de règles à frontière unaire et à *tête sans boucles* pour éviter cela.

Nous pouvons alors montrer que le problème OWQA est décidable pour le langage hybride formé par les contraintes GC^2 et les règles existentielles à frontière unaire et à tête sans boucles. Le résultat est obtenu à l'aide d'un *développement en arbre* pour récrire les règles, éliminer les boucles contenues dans leur corps s'il y en a, et les ramener dans un fragment *entièrement sans boucles* que nous définissons. Nous montrons ensuite que de telles règles peuvent être *déchiquetées* pour les traduire vers GC^2 sur une signature d'arité deux. Nous généralisons nos résultats pour exprimer également des *dépendances fonctionnelles* (FD) en arité supérieure, un type de règles génératrices d'égalités qui est courant en théorie des bases de données. Nous ajoutons ces règles aux contraintes exprimées avec GC^2 et avec les règles existentielles, et prouvons que le problème OWQA est décidable lorsque l'interaction des FD avec les règles est limitée suivant la condition existante d'*absence de conflit* [Calì, Gottlob, and Pieris 2012].

Nos résultats nous laissent penser que la combinaison des approches des logiques de description et des règles existentielles pourrait assurer la décidabilité (et peut-être la faisabilité) du problème OWQA pour des langages logiques *hybrides* plus expressifs encore.

3.2 Réponse aux requêtes en monde ouvert sous hypothèse de finitude

Notre seconde contribution à l'étude du problème OWQA concerne le contexte plus traditionnel des bases de données. Une différence importante dans ce cadre est que l'on s'intéresse traditionnellement aux complétions *finies* des données incomplètes dont on dispose : au lieu de considérer *toutes* les complétions de l'instance I qui satisfont les contraintes Σ, et de trouver les réponses certaines à la requête q sur ces complétions, on s'intéresse aux réponses (possiblement plus nombreuses) qui sont certaines sur les complétions *finies*.

Cette formulation légèrement différente du problème est cependant plus naturelle si l'on suppose, comme on voudrait généralement le faire, que les données sur lesquelles on raisonne sont intrinsèquement finies. Du reste, il s'avère que l'hypothèse de finitude peut effectivement avoir un impact sur les résultats de la requête. Imaginons par exemple une organisation sur laquelle on désire exprimer les choses suivantes :

- La base de données initiale I : « Jeanne conseille Jean. »
- Une première contrainte dans Σ : « Toute personne qui reçoit des conseils prodigue également des conseils à quelqu'un. » Suivant la terminologie des bases de données, il s'agit là d'une *dépendance d'inclusion*, un type particulier de dépendance génératrice de tuples.
- Une seconde contrainte dans Σ : « Personne ne reçoit de conseils de deux personnes différentes. » Il s'agit d'une *dépendance fonctionnelle*, un type particulier de dépendance génératrice d'égalités.

Considérons à présent le problème OWQA pour I, pour Σ, et pour la requête booléenne q qui demande si quelqu'un conseille Jeanne. La première contrainte nous permet de déduire que Jean, puisqu'il est conseillé par Jeanne, doit conseiller quelqu'un, que nous appellerons Janus ; on déduit ensuite que Janus donne des conseils à quelqu'un, par exemple Jacques. Ce processus de déduction pourrait se poursuivre indéfiniment, sans qu'on ne parvienne jamais à déduire que quelqu'un prodigue des conseils à Jeanne. En revanche, si l'on fait l'hypothèse de *finitude*, la chaîne de déductions ne peut pas s'étendre à l'infini : quelqu'un dans cette chaîne, que nous appellerons Jennifer, doit conseiller une personne que nous connaissons déjà. Nous pouvons alors déduire que c'est nécessairement Jeanne qui est conseillée par Jennifer, car tout autre choix enfreindrait nécessairement la seconde règle spécifiée dans Σ.

Cet exemple est bien entendu très simple, mais l'impact général de l'hypothèse de finitude sur le problème OWQA est encore très mal compris. En effet, les principales techniques pour résoudre le problème OWQA font appel à des *modèles universels*, obtenus par exemple à l'aide de la construction de *poursuite*, ou par des techniques de *dépliement*. Malheureusement, les modèles ainsi construits sont généralement infinis, ce qui explique pourquoi ces outils sont inopérants lorsque l'on fait l'hypothèse de finitude.

Les résultats connus à ce jour montrent que, pour certains langages logiques, le problème OWQA avec hypothèse de finitude est équivalent au problème OWQA dans sa formulation habituelle. De tels langages sont qualifiés de *finiment contrôlables*. Un résultat de ce type a été montré pour les dépendances d'inclusion dans [Rosati 2006; Rosati 2011], pour être ensuite généralisé [Bárány, Gottlob, and Otto 2010] au fragment gardé. Un autre résultat de ce type a été montré plus récemment dans [Gogacz and Marcinkowski 2013] pour le fragment de *Datalog collant* défini par [Calì, Gottlob, and Pieris 2010]. La démonstration de chacun de ces résultats est hautement technique. Pourtant, aucun d'entre eux n'autorise de dépendances fonctionnelles, ni plus généralement de dépendances génératrices d'égalité ; ces résultats ne permettent donc pas d'analyser l'exemple que nous avons développé précédemment.

Le problème OWQA sous hypothèse de finitude est mieux compris dans le cas des signatures *d'arité deux*, parce que sa décidabilité est connue même pour des langages qui ne sont pas finiment contrôlables, et même pour des langages qui autorisent des dépendances fonctionnelles. C'est en particulier le cas de GC^2, où la décidabilité sous hypothèse de finitude est démontrée par le biais d'un argument spécifique [Pratt-Hartmann 2009], mais c'est également le cas de certaines logiques de

description [Rosati 2008; Ibáñez-García, Lutz, and Schneider 2014], où les résultats sont obtenus suivant une méthode générale intéressante : compléter les contraintes par un processus de *clôture finie*, qui permet de déduire toutes les conséquences de ces contraintes sous implication finie, et démontrer ensuite que les contraintes sont finiment contrôlables une fois ce processus de clôture finie effectué. Ainsi, ces résultats permettent de comprendre l'exemple ci-dessus, mais uniquement dans le contexte de l'*arité deux* ; on ne pourrait plus les invoquer si la relation entre donneur et receveur de conseils incluait également des informations supplémentaires (par exemple une évaluation).

Notre travail [Amarilli and Benedikt 2015b; Amarilli and Benedikt 2016] s'inspire de cette approche, mais l'applique à des signatures d'arité arbitraire, pour des contraintes d'intégrité qui incluent à la fois des dépendances d'inclusion (ID) et des dépendances fonctionnelles (FD). Ainsi, notre résultat permet de couvrir l'exemple décrit plus haut, quelle que soit l'arité des prédicats. Bien sûr, étant donné que le problème OWQA est indécidable en général lorsque l'on autorise des ID et des FD [Calì, Lembo, and Rosati 2003a], il est nécessaire de restreindre le langage autorisé. Nous nous limitons ainsi aux dépendances d'inclusion *unaires* (UID), c'est-à-dire à celles qui sont des règles à *frontière unaire* : comme la dépendance d'inclusion de notre exemple, elles n'exportent qu'une unique variable entre le corps et la tête de la règle. Ceci permet d'éviter l'indécidabilité, et permet en fait d'exploiter une procédure existante de clôture finie pour le langage logique correspondant [Cosmadakis, Kanellakis, and Vardi 1990]. Ainsi, notre travail montre que les UID et les FD sont finiment contrôlables une fois que l'on a appliqué ce processus de clôture finie. Ceci établit le premier résultat de décidabilité pour l'OWQA sous hypothèse de finitude pour un langage naturel sur des signatures d'arité arbitraire qui inclut à la fois des ID et des FD.

La démonstration de ce résultat est technique, et fait appel à diverses méthodes introduites par des travaux antérieurs :

- L'usage de simulations k-bornées pour préserver les requêtes acycliques de taille suffisamment faible [Ibáñez-García, Lutz, and Schneider 2014] ;
- Une partition des UID en composantes connexes qui n'interagissent que de façon limitée, de sorte que l'on peut ensuite les satisfaire composante par composante [Cosmadakis, Kanellakis, and Vardi 1990; Ibáñez-García, Lutz, and Schneider 2014] ;
- Une procédure de poursuite finie qui réutilise des éléments suffisamment similaires [Rosati 2011] ;
- Une construction de produit qui fait appel à des *groupes à grande maille* pour élargir les cycles [Otto 2002].

Cette démonstration constitue la partie II de notre manuscrit.

Structure du manuscrit

Nous avons fait le choix de restreindre ce manuscrit de thèse à un sous-ensemble de nos contributions telles que présentées plus haut. Plus précisément :

- La partie I de notre manuscrit correspond à la section 1 ci-dessus, les publications correspondantes étant [Amarilli, Bourhis, and Senellart 2015] et [Amarilli, Bourhis, and Senellart 2016] ;
- La partie II de notre manuscrit correspond à la section 3.2 ci-dessus, les publications correspondantes étant [Amarilli and Benedikt 2015b] et [Amarilli and Benedikt 2016] (en cours de relecture par les pairs).

Ces deux parties peuvent être lues indépendamment. Par ailleurs :

- Les sections 2 et 3.1 ci-dessus, même si elles concernent directement le sujet de cette thèse, ne sont pas présentées dans le manuscrit ; les publications correspondantes sont [Amarilli, Amsterdamer, and Milo 2014b; Amarilli and Benedikt 2015a] ainsi que les prépublications [Amarilli, Ba, Deutch, and Senellart 2016; Amarilli, Amsterdamer, Milo, and Senellart 2016] ;
- Certains autres travaux que nous avons effectués parallèlement à notre travail principal de thèse, sans qu'ils soient directement liés à la thématique de celui-ci, sont résumés ci-après.

Autres travaux

Cette section constitue une traduction en français de l'appendice A, et présente les autres travaux que nous avons effectués parallèlement à notre travail principal de thèse.

Complexité de la fouille de données avec une taxonomie dans le contexte de la foule

Avec Yael Amsterdamer et Tova Milo, nous avons étudié la fouille de données dans une situation où les données sont obtenues en interrogeant la foule. Ceci revient formellement à apprendre un prédicat monotone sur un treillis distributif, en utilisant des requêtes à un oracle. Nous avons étudié la complexité computationnelle de ce problème et proposé des algorithmes pour cette tâche.

Ce travail a été présenté à la conférence ICDT'14 [Amarilli, Amsterdamer, and Milo 2014a].

Extraction d'entités sur le Web à l'aide d'identifiants uniques

Nous avons collaboré avec Fabian M. Suchanek sur un travail de recherche entrepris par Aliaksandr Talaika et Joanna Biega. Celui-ci porte sur des méthodes d'extraction d'entités sur le Web qui utilisent des identifiants uniques avec une structure fixée, comme les ISBN, les GTIN, les DOI, les adresses de courrier électronique, etc. Cette approche relativement simple, raffinée suivant certaines étapes de prétraitement, est en mesure d'extraire des millions d'entités uniques à partir d'un corpus de pages Web.

Ce travail a été présenté à WebDB'15 [Talaika, Biega, Amarilli, and Suchanek 2015].

Possibilité pour le XML probabiliste

Nous avons étudié la faisabilité du problème de possibilité pour les documents XML probabilistes dans de nombreux formalismes. Le *problème de possibilité* consiste à déterminer si un document XML non-probabiliste donné est un monde possible d'un document XML probabiliste, et demande éventuellement quelle est la probabilité du document non-probabiliste comme résultat de tirage sur le document probabiliste. Notre travail a déterminé la frontière séparant les formulations faisables et les formulations infaisables de ce problème.

Ce travail a été présenté à AMW'14 [Amarilli 2014], et une version étendue a été publiée dans le journal *Ingénierie des systèmes d'information* [Amarilli 2015a].

Thématiques récentes de recherche portant sur YAGO

Nous avons collaboré avec Luis Galárraga, Nicoleta Preda et Fabian M. Suchanek pour écrire un article invité à APWEB'14 [Amarilli, Galárraga, Preda, and Suchanek 2014], présenté par M. Suchanek, sur le thème de la recherche récente portant sur la base de connaissances YAGO [Suchanek, Kasneci, and Weikum 2007].

Notre contribution porte sur l'étude de l'alignement à grande échelle de bases de connaissances (notamment YAGO) que nous avons effectuée avant notre thèse.

Politiques de tarification pour les documents XML

Nous avons travaillé avec Tang Ruiming, Stéphane Bressan et Pierre Senellart sur le sujet des politiques de tarification pour les documents XML. L'objectif de ces politiques est de permettre aux utilisateurs intéressés par un jeu de données de n'en acheter qu'un échantillon, pour un prix plus modique. Nous avons ainsi étudié des méthodes d'échantillonnage efficace sur les arbres, et les avons appliquées à ce problème.

Ce travail a été publié à la conférence DEXA'14 [Tang, Amarilli, Senellart, and Bressan 2014] et dans le journal TLDKS [Tang, Amarilli, Senellart, and Bressan 2016].

Incertitude, intensionalité et structure

Nous avons travaillé avec Pierre Senellart et Silviu Maniu sur un article que nous avons été invité à soumettre à la lettre d'information SIGWEB.

Notre article porte sur les *données intensionnelles* [Amarilli, Maniu, and Senellart 2015].

Lexique

Ce lexique résume les choix terminologiques effectués et les néologismes construits en français pour traduire les concepts techniques utilisés dans le résumé qui précède.

Par souci de lisibilité, les acronymes et sigles ont été conservés sous leur forme traditionnelle, et ne correspondent pas toujours à la forme française utilisée ici.

access right droit d'accès

acyclic query requête acyclique

arity arité

automated speech recognition (ASR) reconnaissance vocale

bag semantics sémantique multiensembliste

bounded k-simulation simulation k-bornée

cancellative monoid monoïde cancellatif

certain answer réponse certaine

chase poursuite

confidence threshold seuil de confiance

conjunctive query (CQ) requête conjonctive

connected component composante connexe

connected query requête connexe

counting quantifier quantificateur à compteur

crowd foule

crowdsourcing externalisation ouverte à partir de la foule

data complexity complexité en fonction des données

data mining fouille de données

data pricing scheme politique de tarification

database base de données

database engine moteur de base de données

database management system système de gestion de bases de données

dataset jeu de données

description logics logiques de description

distributive lattice treillis distributif

duplicate elimination élimination des doublons

entity extraction extraction d'entités

equality-generating dependency (EGD) dépendance génératrice d'égalités

existential rule règle existentielle

expected value espérance

finite chase poursuite finie

finite closure clôture finie

finitely controllable finiment contrôlable

finiteness assumption hypothèse de finitude

first-order logic (FO) logique du premier ordre

frontier-one rule règle à frontière unaire

frontier-guarded rule règle à frontière gardée

fully polynomial-time randomized approximation scheme schéma d'approximation randomisé entièrement en temps polynomial

fully-non-looping rule règle entièrement sans boucles

functional dependency dépendance fonctionnelle

functionality assertion assertion de fonctionnalité

girth maille

guarded fragment fragment gardé

guarded second-order logic (GSO) logique gardée du second ordre

head-non-looping rule règle à tête sans boucles

ia-width ia-largeur

inclusion dependency dépendance d'inclusion

intensional data données intensionnelles

intractability infaisabilité

inversion-free query requête sans inversion

knowledge base base de connaissances

labeled graph graphe étiqueté

lineage lignage

log file fichier journal

logical constraint contrainte logique

machine learning apprentissage

match counting comptage de correspondances

monadic second-order logic (MSO) logique monadique du second ordre

non-conflicting condition condition d'absence de conflit

NP-hard NP-difficile

open-world query answering (OWQA) réponse aux requêtes en monde ouvert

open-world semantics sémantique du monde ouvert

optical character recognition
reconnaissance optique de caractères

#P-hard #P-difficile

partial order ordre partiel

pathwidth largeur linéaire

planar graph graphe planaire

possible answer réponse possible

preprocessing prétraitement

probabilistic database
base de données probabiliste

probabilistic XML XML probabiliste

probability valuation
annotations de probabilité

provenance circuit circuit de provenance

query containment under constraints
inclusion de requêtes sous contraintes

query requête

ra-linear ra-linéaire

rank aggregation agrégation de rangs

record enregistrement

relational algebra algèbre relationnelle

relational model modèle relationnel

rule body corps de règle

rule head tête de règle

safe query requête prudente

sampling échantillonnage

schema mapping appariement de schémas

second-order logic (SO)
logique du second ordre

semiring semi-anneau

semiring provenance
provenance à base de semi-anneaux

set semantics sémantique ensembliste

shredding déchiquetage

sticky Datalog Datalog collant

three-valued logic logique ternaire

top-k query requête top-k

total order ordre total

tractability faisabilité

tree automaton automate d'arbres

treeification développement en arbre

treelike quasi-arborescent

treewidth largeur d'arbre

tuple-generating dependency (TGD)
dépendance génératrice de tuples

tuple-independent database (TID)
base de données à tuples indépendants

two-variable guarded fragment with counting quantifiers ($\mathbf{GC}^2$)
fragment gardé à deux variables de la logique du premier ordre avec quantificateurs à compteurs

unary inclusion dependency
dépendance d'inclusion unaire

union of conjunctive queries (UCQ)
union de requêtes conjonctives

universal model modèle universel

unraveling dépliement

view maintainance maintenance de vues

weighted finite-state transducer
transducteur fini pondéré

width largeur

Self-References

Peer-Reviewed Conference Articles

Antoine Amarilli, Yael Amsterdamer, and Tova Milo (2014a). "On the Complexity of Mining Itemsets from the Crowd Using Taxonomies". In: *Proc. ICDT*, pp. 15–25. URL: https://arxiv.org/abs/1312.3248 (cit. on pp. 221, 239).

Antoine Amarilli and Michael Benedikt (2015a). "Combining Existential Rules and Description Logics". In: *Proc. IJCAI*. AAAI Press, pp. 2691–2697. URL: https://arxiv.org/abs/1505.00326 (cit. on pp. 8, 11, 215, 217, 233, 238).

Antoine Amarilli and Michael Benedikt (2015b). "Finite Open-World Query Answering with Number Restrictions". In: *Proc. LICS*, pp. 305–316. URL: https://arxiv.org/abs/1505.04216 (cit. on pp. 8, 10, 11, 129, 233, 237, 238).

Antoine Amarilli, Pierre Bourhis, and Pierre Senellart (2015). "Provenance Circuits for Trees and Treelike Instances". In: *Proc. ICALP*. Vol. 9135. LNCS. Springer Berlin Heidelberg, pp. 56–68. URL: https://arxiv.org/abs/1511.08723 (cit. on pp. 3, 11, 15, 39, 44, 53, 226, 238).

Antoine Amarilli, Pierre Bourhis, and Pierre Senellart (2016). "Tractable Lineages on Treelike Instances: Limits and Extensions". In: *Proc. PODS*. To appear. URL: https://a3nm.net/publications/amarilli2016tractable.pdf (cit. on pp. 3, 11, 15, 226, 238).

Ruiming Tang, Antoine Amarilli, Pierre Senellart, and Stéphane Bressan (2014). "Get a Sample for a Discount". In: *Proc. DEXA*. Vol. 8644. LNCS. Springer International Publishing, pp. 20–34. URL: https://a3nm.net/publications/tang2014get.pdf (cit. on pp. 222, 240).

Peer-Reviewed Workshop Articles

Antoine Amarilli (2014). "The Possibility Problem for Probabilistic XML". In: *Proc. AMW*. URL: http://ceur-ws.org/Vol-1189/paper_2.pdf (cit. on pp. 221, 239).

Antoine Amarilli (2015b). "Structurally Tractable Uncertain Data". In: *Proc. PhD Symposium of SIGMOD/PODS*. ACM, pp. 39–44. URL: https://arxiv.org/abs/1507.04955 (cit. on pp. 3, 226).

Antoine Amarilli, Yael Amsterdamer, and Tova Milo (2014b). "Uncertainty in Crowd Data Sourcing Under Structural Constraints". In: *Proc. UnCrowd*. Vol. 8505.

LNCS. Springer Berlin Heidelberg, pp. 351–359. URL: https://arxiv.org/abs/1403.0783 (cit. on pp. 5, 11, 230, 238).

Aliaksandr Talaika, Joanna Biega, Antoine Amarilli, and Fabian M. Suchanek (2015). "IBEX: Harvesting Entities from the Web Using Unique Identifiers". In: *Proc. WebDB*. ACM, pp. 13–19. URL: https://arxiv.org/abs/1505.00841 (cit. on pp. 221, 239).

Peer-Reviewed Journal Articles

Antoine Amarilli (2015a). "Possibility for Probabilistic XML". In: *Ingénierie des Systèmes d'Information* 20.5, pp. 53–75. URL: https://arxiv.org/abs/1404.3131 (cit. on pp. 221, 239).

Ruiming Tang, Antoine Amarilli, Pierre Senellart, and Stéphane Bressan (2016). "A Framework for Sampling-Based XML Data Pricing". In: *Transactions on Large-Scale Data and Knowledge-Centered Systems* 24, pp. 116–138. URL: https://a3nm.net/publications/tang2014framework.pdf (cit. on pp. 222, 240).

Preprints

Antoine Amarilli, Yael Amsterdamer, Tova Milo, and Pierre Senellart (2016). "Top-k Queries on Unknown Values under Order Constraints". Preprint: https://a3nm.net/publications/amarilli2016top.pdf (cit. on pp. 5, 7, 11, 215, 230, 232, 238).

Antoine Amarilli, Lamine M. Ba, Daniel Deutch, and Pierre Senellart (2016). "Possible and Certain Answers for Queries over Order-Incomplete Data". Preprint: https://a3nm.net/publications/amarilli2016possible.pdf (cit. on pp. 5, 6, 11, 215, 230, 238).

Antoine Amarilli and Michael Benedikt (2016). "Finite Open-World Query Answering with Number Restrictions". Preprint: https://a3nm.net/publications/amarilli2016finite.pdf (cit. on pp. 8, 10, 11, 129, 233, 237, 238).

Other Publications

Antoine Amarilli, Luis Galárraga, Nicoleta Preda, and Fabian M. Suchanek (2014). "Recent Topics of Research around the YAGO Knowledge Base". In: *Proc. APWEB*. Vol. 8709. LNCS. Springer International Publishing, pp. 1–12. URL: https://zenodo.org/record/34912 (cit. on pp. 222, 239).

Antoine Amarilli, Silviu Maniu, and Pierre Senellart (2015). "Intensional Data on the Web". In: *SIGWEB Newsletter* Summer 2015, 4:1–4:12. URL: https://a3nm.net/publications/amarilli2015intensional.pdf (cit. on pp. 222, 240).

Antoine Amarilli (2016). "Leveraging the Structure of Uncertain Data". 2016-ENST-0021. PhD thesis. Télécom ParisTech (cit. on p. 244).

Other References

Serge Abiteboul, T.-H. Hubert Chan, Evgeny Kharlamov, Werner Nutt, and Pierre Senellart (2011). "Capturing Continuous Data and Answering Aggregate Queries in Probabilistic XML". In: *ACM TODS* 36.4 (cit. on p. 97).

Serge Abiteboul, Richard Hull, and Victor Vianu (1995). *Foundations of Databases* (cit. on pp. 3, 8, 33, 129, 130, 136, 139, 140, 189, 227, 234).

Yael Amsterdamer, Daniel Deutch, and Val Tannen (2011). "On the Limitations of Provenance for Queries with Difference." In: *Proc. TaPP* (cit. on p. 78).

Yael Amsterdamer, Yael Grossman, Tova Milo, and Pierre Senellart (2013). "Crowd Mining". In: *Proc. SIGMOD* (cit. on pp. 1, 7, 224, 232).

Lyublena Antova, Christoph Koch, and Dan Olteanu (2007). "World-Set Decompositions: Expressiveness and Efficient Algorithms". In: *Proc. ICDT* (cit. on pp. 6, 231).

William Ward Armstrong (1974). "Dependency Structure of Data Base Relationships". In: *Proc. IFIP Congress* (cit. on p. 134).

Stefan Arnborg, Jens Lagergren, and Detlef Seese (1991). "Easy Problems for Tree-Decomposable Graphs". In: *J. Algorithms* 12.2 (cit. on pp. 17, 52, 75, 109).

Jean-François Baget, Michel Leclère, and Marie-Laure Mugnier (2010). "Walking the Decidability Line for Rules with Existential Variables". In: *Proc. KR* (cit. on pp. 8, 23, 234).

Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat (2009). "Extending Decidable Cases for Rules with Existential Variables". In: *Proc. IJCAI* (cit. on pp. 9, 217, 235).

Vince Bárány, Georg Gottlob, and Martin Otto (2010). "Querying the Guarded Fragment". In: *LICS* (cit. on pp. 10, 129, 130, 215, 236).

Daniel Barbará, Hector Garcia-Molina, and Daryl Porter (1992). "The Management of Probabilistic Data". In: *TKDE* 4.5 (cit. on pp. 4, 51, 61, 228).

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann (2009). "DBpedia - A Crystallization Point for the Web of Data". In: *J. Web Semantics* 7.3 (cit. on p. 219).

H. L. Bodlaender (1996). "A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth". In: *SIAM J. Comput.* 25.6 (cit. on pp. 21, 31).

Marijke Hans L. Bodlaender (2012). "Probabilistic Inference and Monadic Second Order Logic". In: *Proc. IFIP TCS* (cit. on p. 58).

Andeas Brandstädt, Van Bang Le, and Jeremy P. Spinrad (1987). “Posets”. In: *Graph Classes. A Survey*. Chap. 6 (cit. on pp. 7, 232).

Graham Brightwell and Peter Winkler (1991). “Counting Linear Extensions”. In: *Order* 8.3 (cit. on pp. 7, 233).

Randal E. Bryant (1992). “Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams”. In: *ACM Comput. Surv.* 24.3 (cit. on pp. 4, 41, 43, 228).

Andrea Calì, Georg Gottlob, and Andreas Pieris (2012). “Towards More Expressive Ontology Languages: The Query Answering Problem”. In: *Artif. Intel.* 193 (cit. on pp. 9, 130, 139, 218, 235).

Andrea Calì, Domenico Lembo, and Riccardo Rosati (2003a). “On the Decidability and Complexity of Query Answering over Inconsistent and Incomplete Databases”. In: *Proc. PODS* (cit. on pp. 8, 10, 129, 130, 139, 218, 233, 237).

Andrea Calì, Domenico Lembo, and Riccardo Rosati (2003b). “Query Rewriting and Answering under Constraints in Data Integration Systems”. In: *Proc. IJCAI* (cit. on p. 139).

Andrea Calì, Georg Gottlob, and Andreas Pieris (2010). “Advanced Processing for Ontological Queries”. In: *PVLDB* 3.1-2 (cit. on pp. 10, 236).

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi (2000). “Containment of Conjunctive Regular Path Queries with Inverse”. In: *Proc. KR* (cit. on p. 101).

Diego Calvanese, Giuseppe De Giacomo, and Moshe Y. Vardi (2005). “Decidable Containment of Recursive Queries”. In: *TCS* 336.1 (cit. on p. 101).

Diego Calvanese, Thomas Eiter, and Magdalena Ortiz (2009). “Regular Path Queries in Expressive Description Logics with Nominals”. In: *Proc. IJCAI* (cit. on p. 218).

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell (2010). “Toward an Architecture for Never-Ending Language Learning”. In: *Proc. AAAI* (cit. on pp. 1, 224).

Marco A. Casanova, Ronald Fagin, and Christos H. Papadimitriou (1984). “Inclusion Dependencies and Their Interaction with Functional Dependencies”. In: *JCSS* 28.1 (cit. on p. 134).

Surajit Chaudhuri and Moshe Y. Vardi (1992). “On the Equivalence of Recursive and Nonrecursive Datalog Programs”. In: *Proc. PODS* (cit. on pp. 28, 30, 31).

Chandra Chekuri and Julia Chuzhoy (2014a). “Polynomial Bounds for the Grid-Minor Theorem”. In: *Proc. STOC* (cit. on pp. 5, 18, 93–96, 103, 108, 111, 216, 229).

Chandra Chekuri and Julia Chuzhoy (2014b). “Polynomial Bounds for the Grid-Minor Theorem”. In: *CoRR* abs/1305.6577 (cit. on p. 96).

James Cheney, Laura Chiticariu, and Wang Chiew Tan (2009). “Provenance in Databases: Why, How, and Where”. In: *Foundations and Trends in Databases* 1.4 (cit. on p. 17).

Edgar F. Codd (1970). “A Relational Model of Data for Large Shared Data Banks”. In: *CACM* 13.6 (cit. on pp. 1, 223).

Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv (2009). “Running Tree Automata on Probabilistic XML”. In: *Proc. PODS* (cit. on pp. 4, 15, 17, 52, 65, 71, 227, 228).

H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi (2007). *Tree Automata: Techniques and Applications*. URL: http://www.grappa.univ-lille3.fr/tata (cit. on pp. 19, 47, 88, 89).

Stavros S. Cosmadakis, Paris C. Kanellakis, and Moshe Y. Vardi (1990). “Polynomial-Time Implication Problems for Unary Inclusion Dependencies”. In: *J. ACM* 37.1 (cit. on pp. 10, 11, 131, 134, 135, 140, 143, 182, 218, 237).

B. Courcelle, J. A. Makowsky, and U. Rotics (2000). “Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width”. In: *TCS* 33.2 (cit. on p. 106).

Bruno Courcelle (1990). “The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs”. In: *Inf. Comput.* 85.1 (cit. on pp. 4, 16, 25, 28, 30, 102, 105, 215, 228).

Bruno Courcelle, Joost Engelfriet, and Grzegorz Rozenberg (1993). “Handle-Rewriting Hypergraph Grammars”. In: *JCSS* 46.2 (cit. on p. 93).

Nilesh Dalvi, Karl Schnaitter, and Dan Suciu (2010). “Computing Query Probability with Incidence Algebras”. In: *Proc. KR* (cit. on p. 73).

Nilesh Dalvi and Dan Suciu (2007). “Efficient Query Evaluation on Probabilistic Databases”. In: *VLDBJ* 16.4 (cit. on pp. 2, 24, 225).

Nilesh Dalvi and Dan Suciu (2012). “The Dichotomy of Probabilistic Inference for Unions of Conjunctive Queries”. In: *J. ACM* 59.6 (cit. on pp. 3, 4, 15–17, 24, 51, 73, 95, 215, 217, 226–228).

Adnan Darwiche (2001). “On the Tractable Counting of Theory Models and its Application to Truth Maintenance and Belief Revision”. In: *J. Applied Non-Classical Logics* 11.1-2 (cit. on pp. 4, 41, 45, 51–53, 228).

Daniel Deutch, Tova Milo, Sudeepa Roy, and Val Tannen (2014). “Circuits for Datalog Provenance.” In: *ICDT* (cit. on pp. 4, 17, 42, 47, 48, 78, 82, 216, 219, 228).

Alin Deutsch, Alan Nash, and Jeff Remmel (2008). “The Chase Revisited”. In: *Proc. SIGMOD* (cit. on p. 136).

Reinhard Diestel (2005). *Graph Theory*. Springer (cit. on p. 21).

Xin Dong, Alon Y. Halevy, and Cong Yu (2007). “Data Integration with Uncertainty”. In: *VLDBJ* (cit. on pp. 1, 224).

Feodor F. Dragan, Fedor V. Fomin, and Petr A. Golovach (2011). “Spanners in Sparse Graphs”. In: *JCSS* 77.6 (cit. on p. 116).

Manfred Droste and Paul Gastin (2007). “Weighted Automata and Weighted Logics”. In: *TCS* 380.1 (cit. on p. 216).

Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar (2001). “Rank Aggregation Methods for the Web”. In: *Proc. WWW* (cit. on pp. 6, 231).

Etalab (2015). *Plateforme ouverte des données publiques françaises*. https://www.data.gouv.fr/ (cit. on p. 219).

Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa (2003). “Data Exchange: Semantics and Query Answering”. In: *Proc. ICDT* (cit. on p. 136).

Ronald Fagin, Amnon Lotem, and Moni Naor (2001). “Optimal Aggregation Algorithms for Middleware”. In: *Proc. PODS* (cit. on pp. 6, 231).

Jörg Flum, Markus Frick, and Martin Grohe (2002). “Query Evaluation via Tree-Decompositions”. In: *J. ACM* 49.6 (cit. on pp. 4, 25, 28, 30, 32–34, 102, 228).

J. Nathan Foster, Todd J. Green, and Val Tannen (2008). “Annotated XML: Queries and Provenance”. In: *Proc. PODS* (cit. on p. 78).

Robert Ganian, Petr Hlinený, Joachim Kneis, Daniel Meister, Jan Obdrzálek, Peter Rossmanith, and Somnath Sikdar (2010). “Are There Any Good Digraph Width Measures?” In: *Proc. IPEC* (cit. on p. 103).

Robert Ganian, Petr Hlinený, Joachim Kneis, Daniel Meister, Jan Obdrzálek, Peter Rossmanith, and Somnath Sikdar (2016). “Are there any good digraph width measures?” In: *J. Comb. Theory, Ser. B* 116 (cit. on p. 105).

Robert Ganian, Petr Hliněnỳ, Alexander Langer, Jan Obdržálek, Peter Rossmanith, and Somnath Sikdar (2014). “Lower Bounds on the Complexity of MSO_1 Model-Checking”. In: *JCSS* 1.80 (cit. on pp. 5, 18, 94, 102, 103, 105–108, 229).

Wolfgang Gatterbauer and Dan Suciu (2015). “Approximate Lifted Inference with Probabilistic Databases”. In: *PVLDB* 8.5 (cit. on p. 217).

Fănică Gavril (1974). “The Intersection Graphs of Subtrees in Trees Are Exactly the Chordal Graphs”. In: *J. Combinatorial Theory* 16.1 (cit. on p. 60).

Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler (2008). “Conjunctive Query Answering for the Description Logic SHIQ”. In: *JAIR* 31 (cit. on p. 218).

Tomasz Gogacz and Jerzy Marcinkowski (2013). “Converging to the Chase – A Tool for Finite Controllability”. In: *Proc. LICS* (cit. on pp. 10, 215, 236).

Georg Gottlob, Erich Grädel, and Helmut Veith (2002). “Datalog LITE: A Deductive Query Language with Linear Time Model Checking”. In: *TOCL* 3.1 (cit. on p. 23).

Georg Gottlob, Reinhard Pichler, and Fang Wei (2010). “Monadic Datalog over Finite Structures of Bounded Treewidth”. In: *TOCL* 12.1 (cit. on p. 217).

Erich Grädel (2000). “Efficient Evaluation Methods for Guarded Logics and Datalog LITE”. In: *Proc. LPAR* (cit. on p. 23).

Erich Grädel, Colin Hirsch, and Martin Otto (2002). “Back and Forth between Guarded and Modal Logics”. In: *TOCL* 3.3 (cit. on pp. 22, 29, 33, 36).

John Grant (1977). “Null Values in a Relational Data Base”. In: *IPL* 6.5 (cit. on pp. 1, 224).

Todd J. Green, Grigoris Karvounarakis, and Val Tannen (2007). “Provenance Semirings”. In: *Proc. PODS* (cit. on pp. 5, 17, 39, 77–79, 216, 228).

Todd J. Green and Val Tannen (2006). “Models for Incomplete and Probabilistic Information”. In: *Proc. IIDB* (cit. on pp. 4, 51, 53, 215, 228).

Martin Grohe (2007). “Logic, Graphs, and Algorithms”. In: *Logic and Automata: History and Perspectives* 2 (cit. on pp. 94, 108).

Stéphane Grumbach and Tova Milo (1999). “An Algebra for Pomsets”. In: *Inf. Comput.* 150.2 (cit. on pp. 6, 230).

U.S. GSA (2015). *Data.gov*. https://www.data.gov/ (cit. on p. 219).

Mordechai Haklay and Patrick Weber (2008). “OpenStreetMap: User-Generated Street Maps”. In: *Pervasive Computing, IEEE* 7.4 (cit. on p. 219).

Georges Hansel (1964). “Nombre minimal de contacts de fermeture nécessaires pour réaliser une fonction booléenne symétrique de n variables”. In: *C. R. Acad. Sc. Paris* 258 (cit. on p. 50).

Johan Håstad (1998). “The shrinkage exponent of de Morgan formulas is 2”. In: *SIAM J. Comput.* 27.1 (cit. on p. 48).

Cecil Huang and Adnan Darwiche (1996). “Inference in Belief Networks: A Procedural Guide”. In: *Int. J. Approximate Reasoning* (cit. on pp. 44, 53).

Jiewen Huang, Lyublena Antova, Christoph Koch, and Dan Olteanu (2009). “MayBMS: a Probabilistic Database Management System”. In: *Proc. SIGMOD* (cit. on pp. 4, 51, 53, 228).

Yazmín Ibáñez-García, Carsten Lutz, and Thomas Schneider (2014). “Finite Model Reasoning in Horn Description Logics”. In: *Proc. KR* (cit. on pp. 10, 11, 129–131, 215, 218, 237).

Tomasz Imieliński and Witold Lipski Jr. (1984). “Incomplete Information in Relational Databases”. In: *J. ACM* 31.4 (cit. on pp. 2, 42, 53, 224).

ISO (2008). *ISO 9075:2008: SQL* (cit. on pp. 1, 224).

Kazuo Iwama and Hiroki Morizumi (2002). “An Explicit Lower Bound of $5n - o(n)$ for Boolean Circuits”. In: *MFCS* (cit. on p. 216).

Marie Jacob, Benny Kimelfeld, and Julia Stoyanovich (2014). “A System for Management and Analysis of Preference Data”. In: *PVLDB* 7.12 (cit. on pp. 6, 231).

Abhay Kumar Jha and Dan Suciu (2012). “On the Tractability of Query Compilation and Bounded Treewidth”. In: *ICDT* (cit. on pp. 41, 43, 45, 123).

Abhay Kumar Jha and Dan Suciu (2013). “Knowledge Compilation Meets Database Theory: Compiling Queries to Decision Diagrams”. In: *TCS* 52.3 (cit. on pp. 17, 18, 25, 41–45, 47, 52, 71–74, 111, 123, 216).

David S. Johnson and Anthony C. Klug (1984). "Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies". In: *JCSS* 28.1 (cit. on pp. 8, 129, 131, 139, 233).

Ravi Kannan, László Lovász, and Miklós Simonovits (1997). "Random Walks and an $O^*(n^5)$ Volume Algorithm for Convex Bodies". In: *Random Struct. Algorithms* 11.1 (cit. on pp. 7, 233).

Grigoris Karvounarakis and Todd J. Green (2012). "Semiring-Annotated Data: Queries and Provenance". In: *ACM SIGMOD Record* 41.3 (cit. on p. 17).

Yevgeny Kazakov (2004). "A Polynomial Translation from the Two-Variable Guarded Fragment with Number Restrictions to the Guarded Fragment". In: *Proc. JELIA* (cit. on pp. 9, 234).

Benny Kimelfeld, Yuri Kosharovsky, and Yehoshua Sagiv (2008). "Query Efficiency in Probabilistic XML Models". In: *Proc. SIGMOD* (cit. on p. 69).

Benny Kimelfeld and Pierre Senellart (2013). "Probabilistic XML: Models and Complexity". In: *Advances in Probabilistic Databases for Uncertain Information Management*. Ed. by Zongmin Ma and Li Yan. Springer (cit. on pp. 2, 17, 64, 225).

Stephan Kreutzer (2008). "Algorithmic meta-theorems". In: *Parameterized and Exact Computation*. Springer (cit. on p. 215).

Stephan Kreutzer and Cristian Riveros (2013). "Quantitative Monadic Second-Order Logic". In: *Proc. LICS* (cit. on p. 216).

Stephan Kreutzer and Siamak Tazari (2010). "Lower Bounds for the Complexity of Monadic Second-Order Logic". In: *Proc. LICS* (cit. on pp. 5, 18, 94, 106, 107, 229).

Steffen L. Lauritzen and David J. Spiegelhalter (1988). "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems". In: *J. Royal Statistical Society. Series B* (cit. on pp. 44, 53).

Leonid Libkin (2014). "Incomplete Data: What Went Wrong, and How to Fix It". In: *Proc. PODS* (cit. on pp. 2, 224).

Maciej Liskiewicz, Mitsunori Ogihara, and Seinosuke Toda (2003). "The Complexity of Counting Self-Avoiding Walks in Subgraphs of Two-Dimensional Grids and Hypercubes". In: *TCS* 1-3.304 (cit. on p. 109).

Johann A. Makowsky and J.P. Marino (2003). "Tree-Width and the Monadic Quantifier Hierarchy". In: *TCS* 303.1 (cit. on pp. 106, 107).

Silviu Maniu, Reynold Cheng, and Pierre Senellart (2014). "ProbTree: A Query-Efficient Representation of Probabilistic Graphs". In: *Proc. BUDA* (cit. on p. 217).

Grigori A. Margulis (1982). "Explicit Constructions of Graphs Without Short Cycles and Low Density Codes". In: *Combinatorica* 2 (cit. on p. 204).

Albert R. Meyer (1975). "Weak Monadic Second Order Theory of Succesor is not Elementary-Recursive". In: *Proc. Logic Colloquium* (cit. on pp. 29, 39, 217).

John C. Mitchell (1983). "The Implication Problem for Functional and Inclusion Dependencies". In: *Information and Control* 56.3 (cit. on pp. 8, 234).

Mehryar Mohri, Fernando Pereira, and Michael Riley (2002). "Weighted Finite-State Transducers in Speech Recognition". In: *Computer Speech & Language* 16.1 (cit. on pp. 2, 225).

Hannes Mühleisen and Christian Bizer (2012). "Web Data Commons – Extracting Structured Data from Two Large Web Corpora". In: *LDOW* 937 (cit. on p. 219).

Frank Neven (2002). "Automata, Logic, and XML". In: *Proc. CSL* (cit. on p. 19).

Frank Neven and Thomas Schwentick (2002). "Query Automata over Finite Trees". In: *TCS* 275.1 (cit. on p. 66).

Dan Olteanu and Jiewen Huang (2008). "Using OBDDs for Efficient Query Evaluation on Probabilistic Databases". In: *Proc. SUM* (cit. on pp. 4, 41, 43, 228).

Adrian Onet (2013). "The Chase Procedure and its Applications in Data Exchange". In: *Data Exchange, Integration, and Streams* (cit. on p. 136).

Martin Otto (2002). "Modal and Guarded Characterisation Theorems over Finite Transition Systems". In: *LICS* (cit. on pp. 11, 131, 203, 237).

Martin Otto (2012). "Highly Acyclic Groups, Hypergraph Covers, and the Guarded Fragment". In: *J. ACM* 59.1 (cit. on p. 204).

Aditya Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom (2012). "Crowdscreen: Algorithms for Filtering Data with Humans". In: *Proc. SIGMOD* (cit. on pp. 1, 7, 224, 232).

Reinhard Pichler, Stefan Rümmele, and Stefan Woltran (2010). "Counting and Enumeration Problems with Bounded Treewidth". In: *Logic for Programming, Artificial Intelligence, and Reasoning* (cit. on p. 217).

Ian Pratt-Hartmann (2009). "Data-Complexity of the Two-Variable Fragment with Counting Quantifiers". In: *Inf. Comput.* 207.8 (cit. on pp. 9, 10, 129, 130, 217, 234, 236).

J. Scott Provan and Michael O. Ball (1983). "The Complexity of Counting Cuts and of Computing the Probability That a Graph is Connected". In: *SIAM Journal on Computing* 12.4 (cit. on p. 58).

Christopher Ré and Dan Suciu (2007). "Materialized Views in Probabilistic Databases: For Information Exchange and Query Optimization". In: *Proc. VLDB* (cit. on pp. 4, 52, 61, 228).

Neil Robertson and Paul D. Seymour (1986). "Graph minors. V. Excluding a Planar Graph". In: *J. Comb. Theory, Ser. B* 41.1 (cit. on pp. 18, 96).

Riccardo Rosati (2006). "On the Decidability and Finite Controllability of Query Processing in Databases with Incomplete Information". In: *PODS* (cit. on pp. 10, 130, 137, 218, 236).

Riccardo Rosati (2008). "Finite Model Reasoning in DL-Lite". In: *Proc. ESWC* (cit. on pp. 10, 131, 137, 237).

Riccardo Rosati (2011). “On the Finite Controllability of Conjunctive Query Answering in Databases under Open-World Assumption”. In: *JCSS* 77.3 (cit. on pp. 10, 11, 129–131, 137, 215, 236, 237).

Sebastian Rudolph and Birte Glimm (2010). “Nominals, Inverses, Counting, and Conjunctive Queries or: Why Infinity is your Friend!” In: *JAIR* 39 (cit. on p. 218).

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum (2007). “YAGO: A Core of Semantic Knowledge”. In: *Proc. WWW* (cit. on pp. 219, 222, 239).

Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch (2011). *Probabilistic Databases*. Morgan & Claypool (cit. on pp. 2, 17, 51, 52, 215, 225).

James W. Thatcher and Jesse B. Wright (1968). “Generalized Finite Automata Theory with an Application to a Decision Problem of Second-Order Logic”. In: *Math. Systems Theory* 2.1 (cit. on pp. 19, 34, 35).

Stephan Tobies (2001). “Complexity Results and Practical Algorithms for Logics in Knowledge Representation”. PhD thesis (cit. on p. 217).

David Toman and Grant E. Weddell (2005). “On Path-Functional Dependencies as First-Class Citizens in Description Logics”. In: *Proc. DL* (cit. on p. 218).

Leslie G. Valiant (1979). “The Complexity of Enumeration and Reliability Problems”. In: *SIAM J. Comput.* 8.3 (cit. on p. 101).

Denny Vrandečić and Markus Krötzsch (2014). “Wikidata: a Free Collaborative Knowledgebase”. In: *CACM* 57.10 (cit. on pp. 1, 219, 224).

Grant E. Weddell (1989). “A Theory of Functional Dependencies for Object-Oriented Data Models.” In: *Proc. DOOD* (cit. on p. 218).

Ingo Wegener (1987). *The Complexity of Boolean Functions*. Wiley (cit. on pp. 47, 48, 50).

Mingji Xia, Peng Zhang, and Wenbo Zhao (2007). “Computational Complexity of Counting Problems on 3-Regular Planar Graphs”. In: *TCS* 384.1 (cit. on pp. 95, 96).

Leveraging the Structure of Uncertain Data

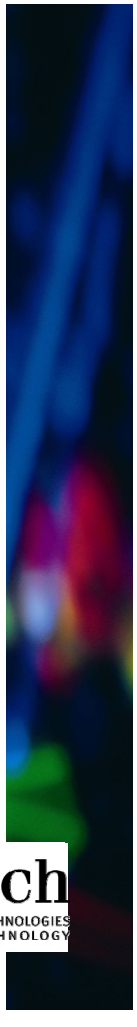
Antoine Amarilli

RÉSUMÉ : La gestion des données incertaines peut devenir infaisable, dans le cas des bases de données probabilistes, ou même indécidable, dans le cas du raisonnement en monde ouvert sous des contraintes logiques. Cette thèse étudie comment pallier ces problèmes en limitant la *structure* des données incertaines et des règles.
La première contribution présentée s'intéresse aux conditions qui permettent d'assurer la faisabilité de l'évaluation de requêtes et du calcul de lignage sur les instances relationnelles probabilistes. Nous montrons que ces tâches sont faisables, pour diverses représentations de la provenance et des probabilités, quand la *largeur d'arbre* des instances est bornée. Réciproquement, sous des hypothèses faibles, nous pouvons montrer leur infaisabilité pour *toute* autre condition imposée sur les instances.
La seconde contribution concerne l'évaluation de requêtes sur des données incomplètes et sous des contraintes logiques, sous l'hypothèse de *finitude* généralement supposée en théorie des bases de données. Nous montrons la décidabilité de cette tâche pour les dépendances d'inclusion unaires et les dépendances fonctionnelles. Ceci constitue le premier résultat positif, sous l'hypothèse de la finitude, pour la réponse aux requêtes en monde ouvert avec un langage d'arité arbitraire qui propose à la fois des contraintes d'intégrité référentielle et des contraintes de cardinalité.

MOTS-CLÉS : incertitude, probabilités, bases de données, réponse aux requêtes.

ABSTRACT: The management of data uncertainty can lead to intractability, in the case of probabilistic databases, or even undecidability, in the case of open-world reasoning under logical rules. My thesis studies how to mitigate these problems by restricting the *structure* of uncertain data and rules.
My first contribution investigates conditions on probabilistic relational instances that ensure the tractability of query evaluation and lineage computation. I show that these tasks are tractable when we bound the *treewidth* of instances, for various probabilistic frameworks and provenance representations. Conversely, I show intractability under mild assumptions for *any* other condition on instances.
The second contribution concerns query evaluation on incomplete data under logical rules, and under the *finiteness* assumption usually made in database theory. I show that this task is decidable for unary inclusion dependencies and functional dependencies. This establishes the first positive result for finite open-world query answering on an arbitrary-arity language featuring both referential constraints and number restrictions.

KEYWORDS: uncertainty, probabilities, databases, query answering.



TELECOM ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY