2017 | Faculty of Sciences

UHASSELT

KNOWLEDGE IN ACTION

Maastricht University

Doctoral dissertation submitted to obtain the degree of
Doctor of Sciences: Information Technology, to be defended by

Dimitri Surinx

DOCTORAL DISSERTATION

A Framework for Comparing Query Languages in Their Ability to Express Boolean Queries

D/2017/2451/74

Promoter: Prof. Dr Jan Van den Bussche

Abstract

When a relational database is queried, the result is normally a relation. Some queries, however, only require a yes/no answer; such queries are often called *boolean queries*. In this thesis, we introduce a framework along which we can investigate boolean queries. We introduce three natural base modalities: testing for nonemptiness of a query; testing for emptiness; and testing for the containment of the result of one query in the result of another query. For the class of first-order queries, these three modalities have exactly the same expressive power. For other classes of queries, e.g., expressed in weaker query languages, the modalities may differ in expressiveness. The expressive power under these different modalities can be compared in several different themes, e.g., we can compare a fixed query language $\mathcal{F}$ under emptiness to $\mathcal{F}$ under nonemptiness. We introduce four general themes to compare the base modalities:

1. We identify crucial query features that enable us to go from one modality to another for a fixed query language. Furthermore, we identify semantical properties that reflect the lack of these query features to establish separations.

2. We compare the expressive power of the base modalities by comparing different query languages under fixed modalities.

3. We compare the expressive power of different query languages under different modalities.

4. We investigate the closure of the modalities under the boolean connectives.

For each of these themes, we establish subsumption as well as separation results for well known query languages such as conjunctive queries and navigational graph query languages.

Acknowledgments

First and foremost, I want to thank my advisor Jan Van den Bussche. Jan's enthusiasm and passion for research always made our meetings very inspiring and a lot of fun. His guidance throughout my PhD has always been excellent, and I have learned a lot from him.

I would also like to thank Dirk Van Gucht for giving me the inspiration I needed to solve several of my open problems. Furthermore, I would like to thank Dirk for giving me the opportunity to visit him at Indiana University, which has been an unforgettable experience.

I am also grateful for my great roomies over the years. I especially want to thank Bas Ketsman with whom I shared most of the experiences of the PhD process.

On the other hand, I want to thank my other colleagues in my research group for providing a stimulating work environment. In particular, I would like to thank Jelle Hellings, with whom I have had a lot of interesting discussions and a lot of laughs during our teaching moments.

I also would like to thank the members of my jury. The insightful comments of Bart Kuijpers and Leonid Libkin were very helpful and greatly improved my thesis.

Last but not least, I am indebted to my friends and family for the continuing support. I am especially grateful to Liesbeth for her advice and her patience during the more difficult parts of my PhD.

Contents

1
Introduction

When a relational database is queried, the result is normally a relation. Some queries, however, only require a yes/no answer; such queries are often called *Boolean queries*. We may ask, for example, "is student 14753 enrolled in course c209?" Also, every integrity constraint is essentially a Boolean query. Another application of Boolean queries is given by SQL conditions, as used in updates and triggers, or in if-then-else statements of SQL/PSM (PL/SQL) programs.

In the theory of database query languages and in finite model theory [1, 18, 31, 30], it is standard practice to express Boolean queries under what we call the *nonemptiness modality*. Under this modality, Boolean queries are expressed in the form $e \neq \emptyset$ where e is a query expression in some query language. Here, a nonempty query result is interpreted as *true* and empty is interpreted as *false*. For example, under the nonemptiness modality, the above Boolean query "is student 14753 enrolled in course c209?" is expressed by the nonemptiness of the query "give all students with id 14753 that are enrolled in course c209". The nonemptiness modality is used in practice in the query language SPARQL. In that language, the result of a Boolean query `ASK` P is true if and only if the corresponding query `SELECT *` P has a nonempty result. Another example of the nonemptiness modality in practice is given by SQL conditions of the form `EXISTS (`Q`)`.

The nonemptiness modality is by no means the only natural way of expressing Boolean queries, however. An integrity constraint is often naturally expressed by a query that looks for violations; then the constraint holds if the query returns no answers. So, here we use the *emptiness* modality rather than nonemptiness. This is exactly the mechanism provided by

SQL table-checks [23]. For example, to express the integrity constraint that an exam should be at least three hours long, we declare a table-check based on the query retrieving all exams which last strictly less than three hours. The query must return an empty result; otherwise an error is raised. Also SQL conditions of the form `NOT EXISTS (`Q`)`, instrumental in formulating nonmonotone queries, obviously use the emptiness modality.

Another natural modality is *containment* of the form $e_1 \subseteq e_2$, where e_1 and e_2 are two query expressions. This Boolean query returns true on a database D if $e_1(D)$ is a subset of $e_2(D)$.[1] For example, the integrity constraint "every student taking course c209 should have passed course c106" is naturally expressed by $e_1 \subseteq e_2$, where e_1 is the query retrieving all students taking c209 and e_2 is the query retrieving all students that passed c106. This example also illustrates the power of the containment modality: containments give us the ability to construct nonmonotone Boolean queries by using monotone queries.

When we use a query language that is powerful enough, such as having the full power of first-order logic, it does not really matter which of the above modalities we use, at least as far as expressive power is concerned. Using first-order queries, the Boolean query $\{\bar{x} \mid \varphi(\bar{x})\} = \emptyset$ can be equivalently expressed by $\{() \mid \neg\exists\bar{x}\,\varphi(\bar{x})\} \neq \emptyset$. Likewise, the Boolean query $\{\bar{x} \mid \varphi_1(\bar{x})\} \subseteq \{\bar{x} \mid \varphi_2(\bar{x})\}$ can be expressed as $\{() \mid \forall\bar{x}(\varphi_1 \to \varphi_2)(\bar{x})\} \neq \emptyset$.

Nevertheless, the choice of modality may still be important for reasons of efficiency and ease of use. For example, a functional dependency (FD) $A \to B$ on a relation $R(A, B)$ is readily expressed as the emptiness of a simple conjunctive query with nonequalities that looks for violations of the FD:

$$\{(a, b1, b2) \mid R(a, b1) \wedge R(a, b2) \wedge b1 \neq b2\} = \emptyset.$$

Here, a nonempty query result thus corresponds to a violation of the FD. Under nonemptiness however, FDs cannot be expressed using any monotone query language such as the conjunctive queries. Hence, more powerful query language features would have to be used, potentially harming effi-

[1] In this thesis, $e_1 \subseteq e_2$ stands for a Boolean query which, in general, may return true on some databases and return false on the other databases. Thus $e_1 \subseteq e_2$, as considered in this thesis, should not be misconstrued as an instance of the famous query containment problem [14, 1], where the task would be to verify statically if $e_1(D)$ is a subset of $e_2(D)$ on *every* database D. Indeed, if e_1 is contained in e_2 in this latter sense, then the Boolean query $e_1 \subseteq e_2$ is entirely uninteresting since it would just return true on every database.

ciency and ease of use. We thus see that the emptiness modality provides a way to express nonmonotone queries using monotone query languages.[2]

A similar situation occurs for inclusion dependencies (INDs [1]), which are easy to express as the containment of two conjunctive queries, but not as the nonemptiness of such a query. Under the emptiness modality, INDs are still not expressible using conjunctive queries, but become expressible when conjunctive queries are extended with negation.

We thus find it worthwhile to investigate how the different modalities for expressing Boolean queries compare to each other.

In this thesis, we introduce a framework along which we can investigate Boolean queries. All the results in this thesis fit into this framework. This framework consists of several themes.

In the first theme, we fix the query language $\mathcal{F}$ and vary the different modalities. In this thesis, we identify the crucial features that enable one modality to be expressible by another modality. Features that turn out to be relevant are the ability to express the constant empty query; set difference; cylindrification; complementation; and tests. Ideally one would like results that go in both ways, showing that one modality is expressible by another modality for some family of queries $\mathcal{F}$, *precisely* when particular query features are available in $\mathcal{F}$. This requires negative results of the kind that one modality can express Boolean queries that the other cannot, whenever these features are lacking. Since languages $\mathcal{F}$ bear no restrictions, and thus could be very pathological, this is not possible in general. Instead, we try to identify general semantical properties of families of queries, such as *monotonicity* or *additivity*, that reflect a degree of weakness or a lack of certain crucial query features. We then obtain results that show, for example, that the nonemptiness and containment modalities have incomparable expressive power for any family of additive queries. Next, we apply these results to popular query languages.

First, we look at families of queries belonging to popular query languages weaker than first-order logic, in particular, the conjunctive queries possibly extended with union. For example, we have identified tests as a feature enabling nonemptiness queries to be transformed into containment queries. Since the conjunctive queries are closed under tests, this general result can now directly be applied. On the other hand, the emptiness

[2]Under the containment modality, $A \to B$ is again expressible using conjunctive queries, as $e_1 \subseteq e_2$, where e_1 is $\{(a, b1, b2) \mid R(a, b1) \wedge R(a, b2)\}$ and e_2 is $\{(a, b, b) \mid R(a, b)\}$.

and containment modalities turn out to be incomparable for any family of unions of conjunctive queries.

Second, we consider a natural algebra of operations on database relations, and consider fragments of this algebra formed by allowing only a subset of the operations and omitting the others. By comparing the different modalities for every fixed fragment, we can investigate which features are sufficient and/or even necessary to go from one modality to another. An ideal setting for such a study is that of navigational graph queries [8, 44, 10]. Indeed, past research has identified a basic set of operations on binary relations that model navigational graph queries [34, 42, 21, 32, 7]. Our results on fragments of this algebra of binary relations are particularly satisfying in that they truly go in both ways: we show that one modality can be expressed in terms of another modality in a given fragment *precisely* when the enabling features that we identified belong to that fragment. In Chapter 3, we focus on this first theme.

In the second theme, we fix a modality and vary the query language. This is particularly interesting when a query language has a lot of different operators that can be included or left out. The task of understanding and comparing language fragments that include some needed features, but omit unneeded ones, makes sense. For example, in database query processing, we could use data structures or query optimization strategies that work well for some operators but not for others. Moreover, some automated reasoning tasks, such as satisfiability or subsumption testing, are decidable in some fragments but not in the full languages. Again, an ideal setting for such a study is that of navigational graph queries [8, 44, 10]. Under the nonemptiness modality, the primitivity[3] has already been characterized [19, 41, 22]. It turns out, for example, that the converse operator is not always primitive in that setting. In contrast, we show that under (conjunctions of) containments, every operator is primitive. In Chapter 4, we focus on this second theme.

In the third theme, we vary both the modality and the query language. We focus on this theme in Chapter 5. Combining this with the first and second themes, we can obtain a fine picture of the effect modalities have on the query language features and vice versa. In this thesis, we attempt to provide such a picture for navigational graph queries. Even though we mostly solve this question, a few comparisons remain open. It turns out, for example, that projection under nonemptiness is not subsumed

[3] An operator is primitive if it always adds expressive power when it cannot be directly constructed from other operators.

by the full relation and converse under containment when union is not present. To prove this result, we establish a preservation style result for the more general conjunctive queries in Chapter 8. Specifically, we show that monotone Boolean queries expressible by containments are exactly the Boolean queries expressible by nonemptiness. Preservation theorems are interesting in their own right and have been studied intensively in model theory, finite model theory and database theory [15, 12, 37, 24, 3, 39].

In the fourth, and final, theme, we investigate the closure of the modalities under the Boolean connectives. Indeed, since the emptiness modality is the negation of the nonemptiness modality, comparing these two modalities for a family of queries amounts to asking whether the emptiness modality is closed under negation for that family. We can ask the same question for the containment modality, and we can also consider closure under conjunction or disjunction. Conjunctions of Boolean queries are particularly relevant in the context of integrity constraints, where typically a list (conjunction) of integrity constraints is specified. We are then interested in the question whether such a list can be equivalently specified by a single integrity constraint. Another interesting observation is that, in logic, conjunction give us a concise and elegant way to express interesting binary relation properties. For example, a binary relation R is a total order if and only if it satisfies the four containments $\mathsf{id} \subseteq R$; $R \circ R \subseteq R$; $R \cap R^{-1} \subseteq \mathsf{id}$; and $\mathsf{all} \subseteq R \cup R^{-1}$. The second chapter of Maddux his book [33] is full of such examples.

For the navigational graph query fragments, we answer the question completely under the nonemptiness and emptiness modality. For containment, the question remains largely open. For conjunctive queries and unions of conjunctive queries we answer the question for the emptiness and nonemptiness modalities. For the containment modality, we only answer the question for conjunctive queries however. For unions of conjunctive queries the question remains open. In Chapter 6, we focus on this fourth theme.

Observe that the closure under conjunction of the containment modality subsumes the *equality* modality $q_1 = q_2$, which is equivalent to $q_1 \subseteq q_2 \wedge q_2 \subseteq q_1$, as well as to $q_1 \cup q_2 \subseteq q_1 \cap q_2$. Conversely, equality always subsumes containment for any family closed under union, since $q_1 \subseteq q_2$ if and only if $q_1 \cup q_2 = q_2$. More generally, it becomes clear that there is an infinitude of modalities one may consider. A general definition of what constitutes a Boolean-query modality may be found in the formal notion of *generalized quantifier* [11, 9]. In fact, questions of the same nature as the ones studied here are also being studied by logicians interested in generalized quantifiers. For example, Hella et al. [26] shows that for every finite set of generalized quantifier there is a more powerful one (by moving to more or higher-arity relations). Obviously, the value of singling out certain generalized quantifiers for investigation in a study such as ours depends on their naturalness as query language constructs.

On a final note, we want to remark that it would be too large of a project to provide a complete picture for all relevant Boolean query families. However, we do want to provide a framework that helps to investigate them, and, furthermore, provide results for some interesting query languages that fit into this framework.

In Chapter 7 of this thesis, we prove a more detailed result connected to our second theme. We already mentioned that, under the nonemptiness modality, the primitivity of operators in navigational graph queries is well understood. In particular, we know that the converse does not always add expressive power in the presence of projection, and can thus be eliminated. In Chapter 7, we show that this elimination always leads to an exponential blowup in degree.

1.1 Publications

The main results of Chapter 4 have been presented at LICS 2017 Symposium [40]. I am also a co-author of the publication [19] which provides the starting point of Chapter 4. Chapter 7 is based on the publication [41].

2

Different ways of expressing boolean queries

2.1 Preliminaries

A database schema Γ is a finite nonempty set of relation names. Every relation name R is assigned an arity, which is a natural number. Assuming some fixed infinite universe of data elements V, an instance I of a relation name R of arity k is a finite k-ary relation over V, i.e., a subset of $V^k = V \times \cdots \times V$ (k times). More generally, an instance I of a database schema Γ assigns to each $R \in \Gamma$ an instance of R, denoted by $I(R)$. The active domain of an instance I, denoted by $\mathrm{adom}(I)$, is the set of all data elements from V that occur in I. For technical reasons, we exclude the *empty instance*, i.e., one of the relations in I must be nonempty.[1] This also implies that $\mathrm{adom}(I)$ is nonempty.

For a natural number k, a k-ary query over a database schema Γ is a computable function that maps each instance I of Γ to a k-ary relation on $\mathrm{adom}(I)$. Let q_1 and q_2 be two k-ary queries, we write $q_1 \sqsubseteq q_2$ if $q_1(I) \subseteq q_2(I)$ for any instance I of Γ.

When the arity of the query is not of importance, we will simply speak of queries instead of k-ary queries. We require queries to be generic [1].

[1] The technical reason is that we consider Boolean queries expressed by the non-emptiness of a query expression. On the empty instance, however, every generic query evaluates to the empty result. But then no Boolean nonemptiness query can ever return true on the empty instance. In order to avoid including special cases in our theorems, it is easier to exclude the empty instance.

A query q is generic if for any permutation f of the universe V, and any instance I, we have $q(f(I)) = f(q(I))$.

Tests, Cylindrification, Complementation Let q_1 and q_2 be queries over a common database schema. We define the query $(q_1 \text{ if } q_2)$ as follows:

$$(q_1 \text{ if } q_2)(I) = \begin{cases} q_1(I) & \text{if } q_2(I) \neq \emptyset; \\ \emptyset & \text{otherwise.} \end{cases}$$

Naturally, we say that a family $\mathcal{F}$ of queries over a common database schema is *closed under tests* if for any two queries q_1 and q_2 in $\mathcal{F}$, the query $(q_1 \text{ if } q_2)$ is also in $\mathcal{F}$.

Cylindrification is an operation on relations that, like projection, corresponds to existential quantification, but, unlike projection, does not reduce the arity of the relation [28, 29, 43]. We introduce an abstraction of this operation as follows. For any natural number k and query q, we define the *k-ary cylindrification of* q, denoted by $\gamma_k(q)$, as follows:

$$\gamma_k(q)(I) = \begin{cases} \mathrm{adom}(I)^k & \text{if } q(I) \neq \emptyset; \\ \emptyset & \text{otherwise.} \end{cases}$$

Finally, for a k-ary query q, the *complement* of q, denoted by q^c, is defined by $q^c(I) = \mathrm{adom}(I)^k - q(I)$. Here, $-$ is the set difference operator.

2.1.1 Navigational graph query languages

In this thesis, we will often work with graph databases, by restricting the database schema Γ to only binary relation names. Any instance I of Γ can be considered as a graph, where the elements of the active domain are considered as nodes, the pairs in the binary relations are directed edges, and the relation names are edge labels. Instances of Γ are henceforth referred to as "graphs over Γ".

The most basic language we consider for expressing queries is the algebra $\mathcal{N}_\Gamma$. The expressions of this algebra are built recursively from the relation names in Γ the primitives $\emptyset$ and id, using the operators composition $(e_1 \circ e_2)$ and union $(e_1 \cup e_2)$. Semantically, each expression $e \in \mathcal{N}_\Gamma$

denotes a query in the following way. Let G be a graph over Γ. Then

$$
\begin{aligned}
\mathsf{id}(G) &= \{(m,m) \mid m \in \mathrm{adom}(G)\};\\
R(G) &= \text{the edge relation } G(R); && \text{(for } R \in \Gamma\text{)}\\
\emptyset(G) &= \emptyset;\\
e_1 \circ e_2(G) &= \{(m,n) \mid \exists p : (m,p) \in e_1(G) \wedge (p,n) \in e_2(G)\};\\
e_1 \cup e_2(G) &= e_1(G) \cup e_2(G).
\end{aligned}
$$

Remark 2.1. The assumption of a basic language is a point of discussion. In principle, there is no reason to use a basic language at all: just consider each and every operation to be optional. For our investigation, we have chosen for a basic language for the following reasons. First, it lends structure to the investigation. Without the framework provided by a basic language, our task would include a large number of ad-hoc cases to be settled. Furthermore, the field of relation algebras identifies composition and union as the natural counterparts for multiplication and addition of binary relations. Union is a very mild operation that is computationally simple. Without composition, you can hardly say you are investigating binary relations. Adding the neutral elements (empty for union, identity for composition) provides the mathematically natural structure of a semiring. □

The basic algebra $\mathcal{N}_\Gamma$ can be extended by adding some of the following features: the primitives diversity (di), and the full relation (all); and the operators converse (e^{-1}), intersection ($e_1 \cap e_2$), set difference ($e_1 - e_2$), projections ($\pi_1(e)$ and $\pi_2(e)$), coprojections ($\overline{\pi}_1(e)$ and $\overline{\pi}_2(e)$), and transitive closure (e^+). We refer to the operators in the basic algebra $\mathcal{N}$ as *basic features*; we refer to the extensions as *nonbasic features*. The semantics of the extensions are as follows:

$$\begin{aligned}
\mathsf{di}(G) &= \{(m,n) \mid m,n \in \mathrm{adom}(G) \wedge m \neq n\};\\
\mathsf{all}(G) &= \{(m,n) \mid m,n \in \mathrm{adom}(G)\};\\
e^{-1}(G) &= \{(m,n) \mid (n,m) \in e(G)\};\\
e_1 \cap e_2(G) &= e_1(G) \cap e_2(G);\\
e_1 - e_2(G) &= e_1(G) - e_2(G);\\
\pi_1(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge \exists n : (m,n) \in e(G)\};\\
\pi_2(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge \exists n : (n,m) \in e(G)\};\\
\overline{\pi}_1(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge \neg\exists n : (m,n) \in e(G)\};\\
\overline{\pi}_2(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge \neg\exists n : (n,m) \in e(G)\};\\
e^+(G) &= \text{the transitive closure of } e(G).
\end{aligned}$$

All the above operators are well-established in so-called "navigational" graph querying [34, 42, 21, 32, 7]. Composition is the analogue of the natural join operator for binary relations and is the essential operator for navigation along the edges of the graph. The set operators are self-explanatory and well known from relational algebra. Converse serves as a kind of renaming, allowing edges to be traversed backwards. Projection allows for testing for the existence of certain nodes without having to move to these nodes; the result is a subset of the identity relation. Coprojection (also known as counterprojection) provides negative testing. Note that coprojection in this thesis should not be confused with the well established co-projection in category theory, algebraic topology, etc. The diversity and full relations are, in a sense, the most extreme, as they allow to jump to any other node, independent of the existence of edges in the graph. The transitive closure operator plays the role of Kleene star for regular expressions over graphs [2, 13]. However, note that the transitive closure operator is not reflexive, while the Kleene star is reflexive. Although we include transitive closure in our treatment, curiously, its presence has little effect for the questions considered in this work.

A *fragment* is any set of nonbasic features. We will often require that fragments F have the following two conditions:

- F contains both projections or none of them.
- F contains both coprojections or none of them.

We refer to these fragments as the *(co)projection restricted fragments*. In our initial research, we only worked with (co)projection restricted fragments. We, however, realized that these restrictions might have been to strict. Therefore, we try to remove these restrictions where possible. If F is a fragment, we denote by $\mathcal{N}_\Gamma(F)$ the language obtained by adding the features in F to $\mathcal{N}_\Gamma$. For example, $\mathcal{N}_\Gamma(\cap)$ denotes the extension with intersection, and $\mathcal{N}_\Gamma(\cap, \pi_1, \pi_2)$ denotes the extension with intersection and both projections. Note that if we write projection without an index, we actually mean that both projections are present. The same holds for coprojection.

Remark 2.2. We will omit the subscript Γ in $\mathcal{N}_\Gamma(F)$ when the precise database schema is not of importance.

Various interdependencies exist between the nonbasic features [21]:

$$
\begin{aligned}
\mathsf{all} &= \mathsf{di} \cup \mathsf{id};\\
\mathsf{di} &= \mathsf{all} - \mathsf{id};\\
e_1 \cap e_2 &= e_1 - (e_1 - e_2);\\
\pi_1(e) &= (e \circ e^{-1}) \cap \mathsf{id} = (e \circ \mathsf{all}) \cap \mathsf{id} = \overline{\pi}_1(\overline{\pi}_1(e)) = \pi_2(e^{-1});\\
\pi_2(e) &= (e^{-1} \circ e) \cap \mathsf{id} = (\mathsf{all} \circ e) \cap \mathsf{id} = \overline{\pi}_2(\overline{\pi}_2(e)) = \pi_1(e^{-1});\\
\overline{\pi}_1(e) &= \mathsf{id} - \pi_1(e);\\
\overline{\pi}_2(e) &= \mathsf{id} - \pi_2(e).
\end{aligned}
$$

For example, by the third equation, when we add difference, we get intersection for free. Hence, when we want to state that, say, intersection is present in the language $\mathcal{N}(F)$, it is not sufficient to state that $\cap$ belongs to F. To deal with this, we use the *completion* $\widetilde{F}$ of a set of nonbasic features F. Guided by the above equations, we define $\widetilde{F}$ as the smallest superset of F satisfying the following rules:

- if $\mathsf{di} \in \widetilde{F}$, then $\mathsf{all} \in \widetilde{F}$;
- if $\mathsf{all} \in \widetilde{F}$ and $- \in F$, then $\mathsf{di} \in \widetilde{F}$;
- if $- \in F$, then $\cap \in \widetilde{F}$;
- if $\cap \in \widetilde{F}$ and $\mathsf{id} \in \widetilde{F}$ and (${}^{-1} \in F$ or $\mathsf{all} \in \widetilde{F}$), then $\pi_1 \in \widetilde{F}$ and $\pi_2 \in \widetilde{F}$;
- if $\overline{\pi}_i \in \widetilde{F}$, then $\pi_i \in \widetilde{F}$ for $i = 1, 2$;
- if $\overline{\pi}_i \in \widetilde{F}$ and $\pi_{3-i} \in \widetilde{F}$, then $\overline{\pi}_{3-i} \in \widetilde{F}$ for $i = 1, 2$;

- if $\pi_i \in \widetilde{F}$ and $^{-1} \in F$, then $\pi_{3-i} \in \widetilde{F}$ for $i = 1, 2$;
- if $- \in F$ and $\pi_i \in \widetilde{F}$, then $\overline{\pi}_i \in \widetilde{F}$ for $i = 1, 2$.

For example, we have

$$\widetilde{\{\mathsf{id}, \mathsf{all}, -\}} = \widetilde{\{\mathsf{id}, \mathsf{di}, -\}} = \{\mathsf{id}, \mathsf{di}, \mathsf{all}, \cap, -, \pi, \overline{\pi}\}.$$

It is now clear that the languages $\mathcal{N}(F)$ and $\mathcal{N}(\widetilde{F})$ are equivalent in that they can express precisely the same queries. Moreover, for any two fragments F_1 and F_2, call $\mathcal{N}(F_1)$ *subsumed* by $\mathcal{N}(F_2)$, denoted by $\mathcal{N}(F_1) \leq \mathcal{N}(F_2)$, if every query in $\mathcal{N}(F_1)$ is also expressible in $\mathcal{N}(F_2)$.

It is known [21] that for every fixed database schema Γ, we have for every two fragments F_1 and F_2 that

$$\mathcal{N}(F_1) \leq \mathcal{N}(F_2) \quad \text{iff} \quad F_1 \subseteq \widetilde{F}_2. \tag{‡}$$

This holds for binary-relation queries. Hence the interdependencies are complete for navigational binary-relation queries. To capture this notion, we introduce primitivity. A feature f is *primitive* under binary-relation queries if for every fragment F such that $f \notin \widetilde{F}$, $\mathcal{N}(f) \not\leq \mathcal{N}(F)$. By (‡), every feature is primitive under binary-relation queries. Obviously, we can introduce such a primitivity notion for every family of Boolean queries based on our fragments.

Remark 2.3. In the original result [21], all is not considered an operator. Furthermore, fragments never contained just a single projection or coprojection. The result, however, can easily be generalized to include all by observing that fragments without all are additive (see the Additivity Lemma in Section 3.2), while fragments with all are not. Furthermore, the restrictions for fragments regarding projection and coprojection can also be removed by a brute-force argument.

2.1.2 Conjunctive queries

To introduce conjunctive queries (CQs) we switch over to another perspective for instances. Again, let V be some fixed infinite universe of data elements V and let R be a relation name in Γ of arity n. An R-fact is an expression of the form $R(a_1, \ldots, a_n)$ where $a_i \in V$ for $i = 1, \ldots, n$. An R-instance I is a finite set of R-facts. More generally, an instance I of a database schema Γ is a union $\bigcup_{R \in \Gamma} I(R)$, where $I(R)$ denotes an R-instance. This definition for instances corresponds to the logic-programming perspective [1]. Note that there is a one-to-one correspondence between instances under the logic-programming perspective and the perspective outlined in the beginning of Chapter 2.1. Indeed, a tuple t in the relation $I(R)$ can be seen as the R-fact $R(t)$ and vice versa.

We formalize the notion of conjunctive queries as follows. A *conjunctive query* is an expression of the form $Q : H \leftarrow B$ where the *head* H is a tuple of variables and the *body* B is a set of atoms over Γ. An *atom* is an expression of the form $R(v_1, \ldots, v_n)$ where $R \in \Gamma$ and $v_1, \ldots, v_n$ are variables. We denote the set of conjunctive queries over Γ with CQ_Γ. When the databases schema Γ is not of importance we will omit the Γ subscript and write CQ instead. For a conjunctive query Q, H_Q denotes the head and B_Q denotes the body of Q. We assume that our queries are *safe*, i.e., the variables in the head are present somewhere in the body.

Semantically, for every instance I over Γ, $Q(I)$ is defined as:

$$\{f(H_Q) \mid f \text{ is a homomorphism from } Q \text{ into } I\}.$$

Here, a homomorphism f from Q into I is a function on the variables in H_Q and B_Q to $\mathrm{adom}(I)$ such that $f(B_Q) \subseteq I$. Since our queries are safe, and thus all the variables of H_Q are present in B_Q we also write that f is a homomorphism from B_Q into I. Interchangeably, we write that B_Q maps into I.

Remark 2.4. It is convenient to assume that variables are data elements in V. Then, we can use the body of a conjunctive query as a database instance. As a consequence, an R-atom can then be thought of as an R-fact.

Remember that, for every two queries Q_1 and Q_2, we write $Q_1 \sqsubseteq Q_2$ if $Q_1(I) \subseteq Q_2(I)$ for every database instance I over Γ. When Q_1 and Q_2 are conjunctive queries, it is well know that $Q_1 \sqsubseteq Q_2$ iff $H_{Q_1} \in Q_2(B_{Q_1})$.

2.2 Boolean query modalities

A *Boolean query* over a database schema Γ is a computable mapping from instances of Γ to $\{\mathit{true}, \mathit{false}\}$. For any Boolean query q, define $\neg q$ as its negation, i.e., $\neg q$ is *true* on an instance I iff q is false on I. Furthermore, for any family of Boolean queries $\mathcal{F}$, define $\neg\mathcal{F}$ as $\{\neg q \mid q \in \mathcal{F}\}$.

As argued in the Introduction, Boolean queries can be naturally expressed in terms of the emptiness, or the nonemptiness, of an ordinary query, or by the containment of the results of two queries. We call these methods the *emptiness*, *nonemptiness* and the containment *modality*. Furthermore, we refer to these modalities as our *base modalities*. Using these modalities we can associate an array of Boolean query families to any family of queries $\mathcal{F}$ on a common database schema Γ:

family of Boolean queries	expressible in the form	with
$\mathcal{F}^{=\emptyset}$	$q = \emptyset$	$q \in \mathcal{F}$
$\mathcal{F}^{\neq\emptyset}$	$q \neq \emptyset$	$q \in \mathcal{F}$
$\mathcal{F}^{\subseteq}$	$q_1 \subseteq q_2$	$q_1, q_2 \in \mathcal{F}$

For $\mathcal{F}^{\subseteq}$, it is understood that only two queries of the same arity can form a containment Boolean query.

Remark 2.5. To simplify notation, we will introduce some extra notation for navigational query languages. For any fragment F of nonbasic features, we define $F_\Gamma^{=\emptyset}$, $F_\Gamma^{\neq\emptyset}$ and $F_\Gamma^{\subseteq}$ to be $\mathcal{N}_\Gamma(F)^{=\emptyset}$, $\mathcal{N}_\Gamma(F)^{\neq\emptyset}$ and $\mathcal{N}_\Gamma(F)^{\subseteq}$ respectively. Again, we will omit the Γ subscript if the database schema is not of importance.

Obviously, these are by no means the only way to express Boolean queries from a family of queries $\mathcal{F}$. We could, for example, allow Boolean connectives within a family of Boolean queries. Indeed, we can consider Boolean queries of the form $q_1 \neq \emptyset \wedge \ldots \wedge q_n \neq \emptyset$ where $q_i \neq \emptyset \in \mathcal{F}^{\neq\emptyset}$ for $i = 1, \ldots, n$. Furthermore, we could even combine two different families of Boolean queries by using Boolean connectives. For example, we can consider Boolean queries of the form $q_1 \neq \emptyset \wedge q_2 \subseteq q_3$ where $q_1 \neq \emptyset \in \mathcal{F}^{\neq\emptyset}$ and $q_2 \subseteq q_3 \in \mathcal{F}^{\subseteq}$.

Our goal is to devise a framework along which we can work to investigate Boolean queries. All our results in this thesis fit in this framework. The framework consists of different themes.

In the first theme, we fix the query language and compare this language under the different base modalities. For example, we can compare conjunctive queries (CQ) under the emptiness and nonemptiness modality. Notice that this surmounts to checking whether CQ is closed under negation. We devote Chapter 3 to this theme.

In the second theme, we fix one of the base modalities, and vary the query language. This is particularly interesting when a query language has a lot of different operators that can be included or be left out. For example, in this theme we could compare the navigational query fragments

$\{\mathsf{di}\}$ and $\{\overline{\pi}\}$ under the containment modality. We devote Chapter 4 to this theme.

Remark 2.6. Note that $\mathcal{F}^{=\emptyset}$ is the negation of $\mathcal{F}^{\neq\emptyset}$ for any language $\mathcal{F}$. Similarly, we can introduce the negation of $\mathcal{F}^{\subseteq}$, denoted by $\mathcal{F}^{\not\subseteq}$, which contains Boolean queries expressible in the form $q_1 \not\subseteq q_2$ where q_1, q_2 are expressions in $\mathcal{F}$. We do not consider $F^{\not\subseteq}$ as a base modality along with nonemptiness, emptiness and containment. Hence, we do not consider the noncontainment modality during themes one and two. The reason for this is that we want to consider natural and practical modalities as building blocks for our study. However, we will consider the noncontainment modality as a "derived" modality at a later stage in theme four.

In the third theme, we generalize the first and second theme so that we compare different query languages under different modalities. For example, we could compare the navigational query fragment $\{\mathsf{di}\}$ under the nonemptiness modality to $\{^{-1}\}$ under emptiness. We devote Chapter 5 to this theme.

In the fourth theme, we close a Boolean query family $\mathcal{B}$ under certain Boolean connectives and compare the obtained language to $\mathcal{B}$. For example, we can close the family $\mathrm{CQ}^{\neq\emptyset}$ under disjunction and compare this to $\mathrm{CQ}^{\neq\emptyset}$. We devote Chapter 6 to this theme.

Remark 2.7. In the Introduction, we already mentioned that navigational query languages provide an ideal setting for themes two and three. These are obviously not the only languages that fit this setting. For example, Codd his famous Relational Algebra [16] is another a suitable query language for such a study. The reason why we choose to focus on the navigational query languages is because our work initially started as a continuation of a larger project on the Boolean expressive power of navigational query languages [21, 19, 22, 41]. Nevertheless, graph databases have been an important subject of study in theory and in practice [8, 44, 10, 6].

3

Comparing different base modalities for fixed query languages

The goal of this chapter is to compare the different base modalities for fixed languages. Formally, for a particular query language $\mathcal{F}$ this amounts to making six comparisons, but we can immediately get one of them out of the way. Indeed, since $\mathcal{A} \subseteq \mathcal{B}$ if and only if $\neg\mathcal{A} \subseteq \neg\mathcal{B}$, we only have to investigate whether $\mathcal{F}^{=\emptyset} \subseteq \mathcal{F}^{\neq\emptyset}$; the other direction $\mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{=\emptyset}$ then directly follows. This amounts to investigating when the emptiness modality is *closed under negation*. Formally, a family $\mathcal{B}$ of Boolean queries is called closed under negation if $\neg\mathcal{B} = \mathcal{B}$.

We first identify query features that enable the expression of one base modality in terms of another one. We also identify general properties that reflect the absence of these query features, notably, the properties of monotonicity and additivity. We then observe how these properties indeed prevent going from one modality to another.

The announced query features are summarized in the following proposition. We leave out the comparison $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{\neq\emptyset}$, since we know of no other general way of going from containment to nonemptiness than via emptiness $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset} \subseteq \mathcal{F}^{\neq\emptyset}$. This leaves four comparisons:

Proposition 3.1. *Let $\mathcal{F}$ be a family of queries. We have:*

1. *$\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset}$ if $\mathcal{F}$ is closed under set difference ($-$).*

2. *$\mathcal{F}^{=\emptyset} \subseteq \mathcal{F}^{\neq\emptyset}$ if there exists k such that $\mathcal{F}$ is closed under*
   - *k-ary complementation, and*
   - *k-ary cylindrification.*

3. *$\mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{\subseteq}$ if*
   - *$\mathcal{F}$ contains a never-empty query (one that returns nonempty on every instance), and*
   - *$\mathcal{F}$ is closed under tests, or $\mathcal{F}$ is closed under k-ary cylindrification for some k.*

4. *$\mathcal{F}^{=\emptyset} \subseteq \mathcal{F}^{\subseteq}$ if $\mathcal{F}$ contains the* empty *query which always outputs the empty relation.*

Proof. In what follows, the proofs are labeled according to the numbers in the proposition.

1. The query $q_1 \subseteq q_2$ is expressed by $q_1 - q_2 = \emptyset$.
2. The query $q = \emptyset$ is expressed by $\gamma_k(q)^c \neq \emptyset$.
3. Let p be a never-empty query. Then $q \neq \emptyset$ is expressed by $p \subseteq (p \,\text{if}\, q)$ as well as by $\gamma_k(p) \subseteq \gamma_k(q)$.
4. The query $q = \emptyset$ is expressed by $q \subseteq \text{empty}$.

□

Obviously, the above proposition only provides sufficient conditions under which we can go from one modality to another. Since the conditions hold for any general family $\mathcal{F}$, we cannot expect the literal converses of these statements to hold in general. Indeed, one could always concoct an artificial family $\mathcal{F}$ that is not closed under difference but for which $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset}$. This is illustrated by the following proposition.

Proposition 3.2. *There exists a language $\mathcal{F}$ that is not closed under difference such that $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset}$.*

Proof. Define $\mathcal{F}$ as the set of queries

$$\text{if } C \text{ then } e_1 \text{ else } e_2$$

with C finite Boolean combinations of expressions $h_i \subseteq h_j$ and e_1, e_2, h_i, h_j in $\{\emptyset, R, S, R \cup S\}$.

This set is not closed under difference. Indeed, $R \in \mathcal{F}$ and $S \in \mathcal{F}$, but $R - S$ is not in $\mathcal{F}$.

We now show that $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset}$. To this end, consider the Boolean query

$$\text{if } C_1 \text{ then } e_1 \text{ else } e_2 \quad \subseteq \quad \text{if } C_2 \text{ then } e_3 \text{ else } e_4$$

in $\mathcal{F}^{\subseteq}$. This is equivalent to the emptiness of

$$\begin{aligned}
&\text{if } C_1 \wedge C_2 \wedge e_1 \subseteq e_3 \text{ then } \emptyset \\
\wedge\ &\text{if } C_1 \wedge C_2 \wedge e_1 \not\subseteq e_3 \text{ then } R \cup S \\
\wedge\ &\text{if } C_1 \wedge \neg C_2 \wedge e_1 \subseteq e_4 \text{ then } \emptyset \\
\wedge\ &\text{if } C_1 \wedge \neg C_2 \wedge e_1 \not\subseteq e_4 \text{ then } R \cup S \\
\wedge\ &\text{if } \neg C_1 \wedge C_2 \wedge e_2 \subseteq e_3 \text{ then } \emptyset \\
\wedge\ &\text{if } \neg C_1 \wedge C_2 \wedge e_2 \not\subseteq e_3 \text{ then } R \cup S \\
\wedge\ &\text{if } \neg C_1 \wedge \neg C_2 \wedge e_2 \subseteq e_4 \text{ then } \emptyset \\
\wedge\ &\text{if } \neg C_1 \wedge \neg C_2 \wedge e_2 \not\subseteq e_4 \text{ then } R \cup S.
\end{aligned}$$

This, in turn, is equivalent to the emptiness of

$$\begin{aligned}
&\text{if } (C_1 \wedge C_2 \wedge e_1 \subseteq e_3) \vee (C_1 \wedge \neg C_2 \wedge e_1 \subseteq e_4) \vee (\neg C_1 \wedge C_2 \wedge e_2 \subseteq e_3) \\
&\vee (\neg C_1 \wedge \neg C_2 \wedge e_3 \subseteq e_4) \text{ then } \emptyset \text{ else } R \cup S,
\end{aligned}$$

which proves the proposition since this query is in $\mathcal{F}$. $\square$

One approach to still find a kind of converse to the above sufficient conditions, is to come up with general semantic properties of the queries in a family that would basically prevent the sufficient conditions to hold. We can then proceed to show that the different modalities become incomparable under these properties.

More concretely, we can observe two main themes in the sufficient conditions: *negation*, in the forms of set difference and complementation, and *global access to the database*, in the forms of cylindrification and tests. A well-known semantic property of queries that runs counter to negation is *monotonicity*. For a property that prevents global access, we propose *additivity*.

Monotonicity A query q is *monotone* if $I \subseteq J$ implies $q(I) \subseteq q(J)$, where $I \subseteq J$ means that $I(R) \subseteq J(R)$ for each relation name R. We have seen that closure under negation, which typically destroys monotonicity, allows the emptiness modality to be closed under negation, as well as the containment modality to be subsumed by emptiness. We next show that both fail under monotonicity. The first failure is the strongest:

Lemma 3.3. *Let* MON *denote the family of monotone queries. The only Boolean queries in* $\mathrm{MON}^{=\emptyset} \cap \mathrm{MON}^{\neq\emptyset}$ *are the constant* true *and* false *queries.*

Proof. Suppose for the sake of contradiction that a nonconstant Boolean query $q = \emptyset \in \mathrm{MON}^{=\emptyset}$ is also in $\mathrm{MON}^{\neq\emptyset}$. Then, there exists $q' \in \mathrm{MON}^{\neq\emptyset}$ such that for any instance I, $q(I) = \emptyset$ iff $q'(I) \neq \emptyset$. Since q is nonconstant, there exist two instances I and J over Γ such that $q(I) \neq \emptyset$ and $q(J) = \emptyset$. Then, $q'(I) = \emptyset$ and $q'(J) \neq \emptyset$. Thus since q and q' are both in MON, we have $\emptyset \neq q(I) \subseteq q(I \cup J)$ and $\emptyset \neq q'(J) \subseteq q'(I \cup J)$. Therefore, $q(I \cup J) \neq \emptyset$ and $q'(I \cup J) \neq \emptyset$ which is clearly a contradiction. □

As a corollary, we obtain:

Proposition 3.4. *Let $\mathcal{F}$ be a family of monotone queries. If $\mathcal{F}^{=\emptyset}$ contains a non-constant query, then $\mathcal{F}^{=\emptyset} \not\subseteq \mathcal{F}^{\neq\emptyset}$.*

This also implies that for every monotone family of queries $\mathcal{F}$ that contains the empty query, and for MON in particular, that $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{\neq\emptyset}$ since $\mathcal{A}^{=\emptyset} \subseteq \mathcal{A}^{\subseteq}$ for every family of queries $\mathcal{A}$ that contains the empty query.

We next turn to the failure of going from containment to emptiness. Whenever q is monotone, the Boolean query $q = \emptyset$ is antimonotone (meaning that if $q(I) =$ false and $I \subseteq J$, also $q(J) =$ false). However, a Boolean containment query is typically not antimonone. The following straightforward result gives two examples.

Proposition 3.5. *Let $\mathcal{F}$ be a family of monotone queries over a database schema Γ.*

1. *If Γ contains two distinct relation names R and T of the same arity, and the two queries R and T belong to $\mathcal{F}$, then $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{=\emptyset}$. This is shown by the Boolean query $R \subseteq T$.*

2. *If R is a binary relation name in Γ and the two queries $R \circ R$ and R belong to $\mathcal{F}$, then $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{=\emptyset}$.*

Proof. In what follows, the proofs are labeled according to the numbers in the proposition.

1. The query $R \subseteq T$ is not antimonotone.

2. The query "R is transitive", or $R \circ R \subseteq R$, is not antimonotone.

$\square$

Additivity A query q is *additive* if for every two instances I and J such that adom(I) and adom(J) are disjoint, $q(I \cup J) = q(I) \cup q(J)$. Additive queries (also known as "queries distributing over components") have been recently singled out as a family of queries that are well amenable to distributed computation [4]. Indeed, additivity means that a query can be separately computed on each connected component, after which all the subresults can simply be combined by union to obtain the final result.

Both cylindrification and tests run counter to additivity. For example, just computing adom$(I) \times$ adom(I) is not additive. Also tests of the form (q_1 if q_2) are not additive, since testing if q_2 is nonempty takes part in the entire instance, across connected components. We have seen that cylindrification (together with complementation) can be used to close the emptiness modality under negation; moreover, cylindrification or tests suffice to move from nonemptiness to containment. We next show that this all fails under additivity.

The following lemma is of a similar nature as Lemma 3.3.

Lemma 3.6. *Let* ADD *denote the family of additive queries. The only Boolean queries in* $\mathrm{ADD}^{\neq\emptyset} \cap \mathrm{ADD}^{\subseteq}$ *are the constant* true *and* false *queries.*

Proof. Suppose for the sake of contradiction, that a nonconstant Boolean query $q \neq \emptyset \in \mathrm{ADD}^{\neq\emptyset}$ is also in $\mathrm{ADD}^{\subseteq}$. Then, there exist two k-ary queries q_1 and q_2 in ADD such that for any instance I we have $q_1(I) \subseteq q_2(I)$ iff $q(I) \neq \emptyset$. Since q is nonconstant, there exist two instances I and J such that $q(I) \neq \emptyset$ and $q(J) = \emptyset$. Hence $q_1(I) \subseteq q_2(I)$ and $q_1(J) \not\subseteq q_2(J)$. We may assume that adom(I) and adom(J) are disjoint since queries are defined to be generic. Therefore, since q is additive, we have $q(I \cup J) = q(I) \cup q(J) \neq \emptyset$, whence we have $q_1(I \cup J) \subseteq q_2(I \cup J)$. Thus we have $q_1(I) \cup q_1(J) \subseteq q_2(I) \cup q_2(J)$. However, this implies that $q_1(J) \subseteq q_2(J)$ since $q_1(J) \subseteq \mathrm{adom}(J)^k$ and $q_2(I) \subseteq \mathrm{adom}(I)^k$, which is a contradiction.

$\square$

$$\begin{tikzcd} \mathrm{CQ}^{\subseteq} & \\ & \mathrm{CQ}^{=\emptyset} \\ \mathrm{CQ}^{\neq\emptyset} \arrow[uu] & \end{tikzcd} \qquad \begin{tikzcd} \mathrm{UCQ}^{\subseteq} & \\ & \mathrm{UCQ}^{=\emptyset} \\ \mathrm{UCQ}^{\neq\emptyset} \arrow[uu] & \end{tikzcd}$$

Figure 3.1: These diagrams visualize Theorem 3.9. The arrows in the diagrams depict the subsumption relation of Boolean query families.

As a corollary, we obtain:

Proposition 3.7. *Let $\mathcal{F}$ be a family of additive queries. We have*

1. *If $\mathcal{F}^{\subseteq}$ contains a non-constant query, then $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{\neq\emptyset}$.*
2. *If $\mathcal{F}^{\neq\emptyset}$ contains a non-constant query, then $\mathcal{F}^{\neq\emptyset} \not\subseteq \mathcal{F}^{\subseteq}$ and $\mathcal{F}^{=\emptyset} \not\subseteq \mathcal{F}^{\neq\emptyset}$.*

Remark 3.8. Additivity and monotonicity are orthogonal properties. For example, the additive queries are closed under set difference, i.e., if q_1 and q_2 are additive, then $q_1 - q_2$ is additive. Thus, additive queries may involve negation and need not be monotone. On the other hand, computing the Cartesian product of two relations is monotone but not additive.

In the remainder of this section, we will continue our investigation on (unions) of conjunctive queries and navigational graph query languages in Section 3.1 and Section 3.2 respectively.

3.1 Conjunctive queries

In this brief section, we compare the three base modalities for the popular languages CQ (conjunctive queries) and UCQ (unions of conjunctive queries). The results are summarized in Theorem 3.9 and displayed in Figure 3.1.

Theorem 3.9. *Let $\mathcal{F}$ be* CQ *or* UCQ. *We have:*

1. *$\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{=\emptyset}$ and $\mathcal{F}^{=\emptyset} \not\subseteq \mathcal{F}^{\subseteq}$.*

2. $\mathcal{F}^{=\emptyset} \not\subseteq \mathcal{F}^{\neq\emptyset}$.

3. $\mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{\subseteq}$.

4. $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{\neq\emptyset}$.

Proof. In what follows, the proofs are labeled according to the numbers in the theorem.

1. Consider the instance Z where every relation R contains exactly one tuple $(1, 1, \dots, 1)$ of the appropriate arity. The result of every conjunctive query Q on Z contains exactly one tuple: $(1, 1, \dots, 1)$. Thus, every query in $\mathcal{F}^{\subseteq}$ returns *true* on Z, whereas every query in $\mathcal{F}^{=\emptyset}$ returns *false*.

2. This case directly follows from Proposition 3.4.

3. This case directly follows from Proposition 3.1(3). Indeed, a CQ with an empty body is never empty. CQs and UCQs are also closed under tests. Indeed, let q_1 and q_2 be UCQs. Then (q_1 if q_2) is expressed by the UCQ consisting of the following rules. Take a rule r of q_1 and a rule s of q_2. Produce the rule obtained from r by adding to the body a variable-renamed copy of the body of s. If q_1 has n rules and q_2 has m rules, we obtain nm rules. In particular, if q_1 and q_2 are CQs, we obtain a single rule so again a CQ.

4. Let R be a relation name in the database schema, and consider the two queries

$$q_1(x, y) \leftarrow R(x, \_, \dots, \_), R(y, \_, \dots, \_);$$
$$q_2(x, x) \leftarrow R(x, \_, \dots, \_).$$

   Here, the underscores stand for fresh nondistinguished variables (Prolog notation). Then $q_1 \subseteq q_2$ returns true on an instance I if and only if the first column of $R(I)$ holds at most one distinct element. This Boolean query is not monotone, and thus not in $\mathcal{F}^{\neq\emptyset}$.

□

Remark 3.10. In the proof of Theorem 3.9(4), we make convenient use of repeated variables in the head. For the version of CQs where this is disallowed, the result can still be proven by using

$$q_1(x_1,\dots,x_k) \leftarrow R(x_1,\dots,x_k);$$
$$q_2(x_1,\dots,x_k) \leftarrow R(x_1,\dots,x_k), R(x_k,\_,\dots,\_).$$

This does not work if R is unary; if there are two different relation names R and T, we can use

$$q_1(x) \leftarrow R(x,\_,\dots,\_);$$
$$q_2(x) \leftarrow T(x,\_,\dots,\_).$$

These arguments only fail when the database schema consists of just one single unary relation name, and we cannot use repeated variables in the head. In this extreme case, both $\mathrm{CQ}^{\subseteq}$ and $\mathrm{CQ}^{\neq\emptyset}$ consist only of the constant true query, so the subsumption becomes trivial. $\square$

3.2 Navigational graph query languages

In this section, we compare the three base modalities for the navigational graph query languages outlined in Section 2.1.1.

The results are summarized in the following theorem. This theorem can be seen as a version of our earlier Proposition 3.1, specialized to navigational graph query language fragments. However, now, every statement is a characterization, showing that the sufficient condition is also necessary for subsumption to hold. Particularly satisfying is that, with a few exceptions, almost the entire theorem can be proven following the simple general results in the start of Chapter 3, as we will demonstrate below.

Theorem 3.11. *Let F be a fragment of nonbasic features. We have:*

1. $F^{\subseteq} \subseteq F^{=\emptyset}$ *if and only if* $- \in F$.
2. $F^{=\emptyset} \subseteq F^{\neq\emptyset}$ *if and only if* $\mathsf{all} \in \widetilde{F}$ *and* ($- \in F$ *or* $\overline{\pi}_1 \in \widetilde{F}$ *or* $\overline{\pi}_2 \in \widetilde{F}$).
3. $F^{\neq\emptyset} \subseteq F^{\subseteq}$ *if and only if* $\mathsf{all} \in \widetilde{F}$.
4. $F^{\subseteq} \subseteq F^{\neq\emptyset}$ *if and only if* $\mathsf{all} \in \widetilde{F}$ *and* $- \in F$.

Notice that Theorem 3.11 no longer contains an adapted version for Proposition 3.1(4). This is because the empty query is in $\mathcal{N}(F)$ for every fragment F by definition, whence $F^{=\emptyset} \subseteq F^{\subseteq}$ always holds. Instead, we now do provide in item 4 an explicit characterization for when the subsumption from containment to nonemptiness holds.

In every part of the above theorem, the if-direction can be seen by showing that $\mathcal{N}(F)$ fulfills the conditions of Proposition 3.1.

1. This follows immediately from Proposition 3.1(1).
2. When set difference is present, the binary complementation of q is expressible by $\mathsf{all} - q$. Also the binary cylindrification of q is expressible by $\mathsf{all} \circ q \circ \mathsf{all}$. Hence, Proposition 3.1(2) readily applies with $k = 2$.

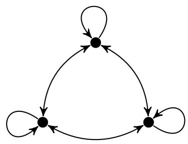
   When set difference is not present, we have coprojection. We can now apply Proposition 3.1(2) with $k = 1$. We simulate unary relations by subsets of the identity relation id. In particular, the unary cylindrification of q is expressed by $\pi_1(\mathsf{all} \circ q)$ and $\pi_2(q \circ \mathsf{all})$, and unary complement is provided by coprojection.
3. We have already seen how binary cylindrification is expressible using all. Furthermore, all also provides a never-empty query. Hence, Proposition 3.1(3) readily applies.
4. We have $F^{\subseteq} \subseteq F^{=\emptyset} \subseteq F^{\neq\emptyset}$.

To prove the only-if directions of the theorem, we will exhibit inexpressibility results.

Inexpressibility results For the first part of Theorem 3.11, it is sufficient to show that $F^{\subseteq}$ is not subsumed by $F^{=\emptyset}$ for every fragment F without set difference. Thereto, we introduce the fragment NoDiff which is defined as $\{\mathsf{id}, \mathsf{di}, ^{-1}, \cap, \overline{\pi}, ^{+}\}$. The completion of NoDiff is the maximal fragment without set difference. The following lemma establishes Theorem 3.11(1) by exhibiting, for every fragment F, a Boolean query in $F^{\subseteq}$ but not in $\mathrm{NoDiff}^{=\emptyset}$.

Lemma 3.12. *Let R be a relation schema. Then the Boolean query "R is transitive", formally, $R \circ R \subseteq R$, is neither in* $\mathrm{NoDiff}^{=\emptyset}$ *nor in* $\mathrm{NoDiff}^{\neq\emptyset}$.

Proof. Over the single relation name R, consider the complete directed graph on three nodes K_3, and a graph B in the form of a bow tie, i.e., two K_3 copies with one shared node. (Both K_3 and B are displayed in Figure 3.2.) There is a self-loop at every node. It is known [21] that K_3

(K_3) 

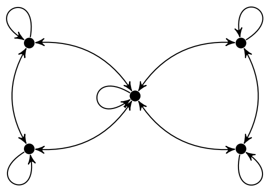
(B) 

Figure 3.2: K_3 and bow tie graphs.

and B are indistinguishable by Boolean queries in $\text{NoDiff}^{\neq\emptyset}$ [19, Proposition 5.6(1)]. This implies that both graphs are also indistinguishable by Boolean queries in $\text{NoDiff}^{=\emptyset}$. However, K_3 is transitive while B is not. $\square$

The only-if directions of the remaining parts of Theorem 3.11 all revolve around the fragment $\text{NoAll} = \{\mathsf{id}, {}^{-1}, -, {}^{+}\}$, whose completion is the largest fragment without the full relation all. This fragment lacks the only two features (di and all) that allow to jump from one connected component to another. Hence we obtain the following:

Additivity Lemma. *Every binary-relation query in* $\mathcal{N}(\text{NoAll})$ *is additive.*

Proof. Let e be an expression in $\mathcal{N}(\text{NoAll})$, and let G and H be graphs such that $\text{adom}(G) \cap \text{adom}(H) = \emptyset$. We must show that $e(G \cup H) = e(G) \cup e(H)$. We proceed by structural induction on e. The case where e is a relation name is trivial and the case where e is id is clear.

If e is of the form e_1^{-1}, then

$$\begin{aligned}
&e_1^{-1}(G \cup H)\\
&= \{(y,x) \in \mathrm{adom}(G \cup H)^2 \mid (x,y) \in e_1(G \cup H)\}\\
&\stackrel{*}{=} \{(y,x) \in \mathrm{adom}(G \cup H)^2 \mid (x,y) \in e_1(G) \cup e_1(H)\}\\
&= \{(y,x) \in \mathrm{adom}(G \cup H)^2 \mid (x,y) \in e_1(G)\}\\
&\quad \cup \{(y,x) \in \mathrm{adom}(G \cup H)^2 \mid (x,y) \in e_1(H))\}\\
&\stackrel{**}{=} \{(y,x) \in \mathrm{adom}(G)^2 \mid (x,y) \in e_1(G)\}\\
&\quad \cup \{(y,x) \in \mathrm{adom}(H)^2 \mid (x,y) \in e_1(H))\}\\
&= e_1^{-1}(G) \cup e_1^{-1}(H).
\end{aligned}$$

The equality marked with a single * follows from the induction hypothesis (IH). Furthermore, the equality marked with ** holds because $e_1(G) \subseteq \mathrm{adom}(G)^2$, $e_1(H) \subseteq \mathrm{adom}(H)^2$, and $\mathrm{adom}(G) \cap \mathrm{adom}(H) = \emptyset$.

If $e = e_1 \cup e_2$, then

$$\begin{aligned}
e_1 \cup e_2(G \cup H) &= e_1(G \cup H) \cup e_2(G \cup H)\\
&\stackrel{*}{=} e_1(G) \cup e_1(H) \cup e_2(G) \cup e_2(H)\\
&= (e_1 \cup e_2)(G) \cup (e_1 \cup e_2)(H).
\end{aligned}$$

The equality marked with a * follows from the IH.

If $e = e_1 \circ e_2$, then

$$\begin{aligned}
&(x,y) \in e_1 \circ e_2(G \cup H)\\
&\text{ iff } \exists z : (x,z) \in e_1(G \cup H) \wedge (z,y) \in e_2(G \cup H)\\
&\stackrel{*}{\text{iff}} \exists z : (x,z) \in e_1(G) \cup e_1(H)\\
&\qquad\qquad \wedge (z,y) \in e_2(G) \cup e_2(H)\\
&\text{ iff } \exists z : ((x,z) \in e_1(G) \vee (x,z) \in e_1(H))\\
&\qquad\qquad \wedge ((z,y) \in e_2(G) \vee (z,y) \in e_2(H))\\
&\stackrel{**}{\text{iff}} \exists z : ((x,z) \in e_1(G) \wedge (z,y) \in e_2(G))\\
&\qquad\qquad \vee ((x,z) \in e_1(H) \wedge (z,y) \in e_2(H))\\
&\text{ iff } (x,y) \in e_1 \circ e_2(G) \vee (x,y) \in e_1 \circ e_2(H).
\end{aligned}$$

The equivalence marked with a single * follows from the IH. Furthermore, the equivalence marked with ** holds because $e_1(G) \subseteq \mathrm{adom}(G)^2$,

$e_1(H) \subseteq \mathrm{adom}(H)^2$, and $\mathrm{adom}(G) \cap \mathrm{adom}(H) = \emptyset$. Indeed, because of this observation we can drop the cases $(x,z) \in e_1(G) \wedge (z,y) \in e_2(H)$ and $(x,z) \in e_1(H) \wedge (z,y) \in e_2(G)$.

If $e = e_1 - e_2$, then

$$\begin{aligned} e_1 - e_2(G \cup H) &= e_1(G \cup H) - e_2(G \cup H) \\ &\overset{*}{=} (e_1(G) \cup e_1(H)) - (e_2(G) \cup e_2(H)) \\ &\overset{**}{=} (e_1(G) - e_2(G)) \cup (e_1(H) - e_2(H)) \\ &= (e_1 - e_2)(G) \cup (e_1 - e_2)(H). \end{aligned}$$

The equivalence marked with a single * follows from the IH. Furthermore, the equivalence marked with ** holds because $e_1(G) \subseteq \mathrm{adom}(G)^2$, $e_1(H) \subseteq \mathrm{adom}(H)^2$, and $\mathrm{adom}(G) \cap \mathrm{adom}(H) = \emptyset$.

If $e = e_1^+$ then $e_1^+(G \cup H) = (e_1(G) \cup e_1(H))^+$ by induction. Now since $\mathrm{adom}(G) \cap \mathrm{adom}(H) = \emptyset$ we also have that $(e_1(G) \cup e_1(H))^+ = e_1^+(G) \cup e_1^+(H)$. □

The Additivity lemma allows an easy proof for Theorems 3.11(2) and 3.11(3), as we will next demonstrate. Furthermore, several other results will hinge upon the Additivity Lemma.

Remark 3.13. The Additivity Lemma also follows from the additivity of connected stratified Datalog$^\neg$ [5].

For the second part of Theorem 3.11, we must prove that $F^{=\emptyset}$ is not subsumed by $F^{\neq\emptyset}$ for every fragment F without all, as well as any fragment having neither difference nor coprojection. The latter case is clear. Indeed, difference and coprojection are the only two nonmonotone operators. Thus $\mathcal{N}(F)$ is monotone, whence Proposition 3.4 proves the result.

For a fragment F without all but possibly with difference or coprojection, we have that $\mathcal{N}(F)$ is additive. Hence, Proposition 3.7 establishes both the second and third parts of Theorem 3.11 when $\mathsf{all} \notin F$.

Finally, for the fourth part of Theorem 3.11, we must prove that $F^{\subseteq}$ is not subsumed by $F^{\neq\emptyset}$ for every fragment F without all or without set difference. The case without set difference already follows from Lemma 3.12. The case without all already follows from Theorem 3.11(2).

Remark on regular path queries The fragment $\{^+\}$ corresponds to a well known family of graph queries called regular path queries (RPQ) [17]. Thus, Theorem 3.11 directly gives us the following corollary.

Corollary 3.14. *Let RPQ be the family of regular path queries. We have:*

1. $\mathrm{RPQ}^{=\emptyset} \not\subseteq \mathrm{RPQ}^{\neq\emptyset}$;
2. $\mathrm{RPQ}^{\neq\emptyset} \not\subseteq \mathrm{RPQ}^{\subseteq}$;
3. $\mathrm{RPQ}^{\subseteq} \not\subseteq \mathrm{RPQ}^{=\emptyset}$;
4. $\mathrm{RPQ}^{\subseteq} \not\subseteq \mathrm{RPQ}^{\neq\emptyset}$.

4

Comparing different query languages under fixed base modalities

The goal of this chapter is to compare different query languages under the same base modality. Formally, for particular sets $\mathcal{C}$ of query languages, we want to answer the following questions:

1. $\mathcal{F}_1^{\neq\emptyset} \overset{?}{\subseteq} \mathcal{F}_2^{\neq\emptyset}$

2. $\mathcal{F}_1^{=\emptyset} \overset{?}{\subseteq} \mathcal{F}_2^{=\emptyset}$

3. $\mathcal{F}_1^{\subseteq} \overset{?}{\subseteq} F_2^{\subseteq}$

for every $\mathcal{F}_1$ and $\mathcal{F}_2$ in $\mathcal{C}$ such that $\mathcal{F}_1 \neq \mathcal{F}_2$.

As mentioned in Section 2.2, these questions are particularly interesting when a query language has a lot of different operators that can be included or be left out. The navigational graph query languages introduced in Section 2.1.1 are of this nature. In the remainder of this chapter, we will focus on these navigational graph query languages.

First, we look at $\mathcal{F}_1^{\neq\emptyset} \overset{?}{\subseteq} \mathcal{F}_2^{\neq\emptyset}$. This question has already been answered for (co)projection restricted fragments [21, 22]. Before we state this result, we first need the following definition.

For every fragment F, define $\widehat{F}$ as:

- The set obtained from F where we add π and remove $^{-1}$, if $^{-1} \in \widetilde{F}$, $\cap \notin \widetilde{F}$ and $^{+} \notin \widetilde{F}$;
- The set F otherwise, i.e., if $^{-1} \notin \widetilde{F}$, $\cap \in \widetilde{F}$ or $^{+} \in \widetilde{F}$.

The answer to question one can then be summarized as follows.

Theorem 4.1 ([21, 19, 22, 41])**.** *Let F_1 and F_2 be (co)projection restricted fragments. If Γ contains at least two edge labels, then:*

$$F_{1_\Gamma}^{\neq\emptyset} \subseteq F_{2_\Gamma}^{\neq\emptyset} \quad \textit{iff} \quad F_1 \subseteq \widetilde{F_2} \textit{ or } \widehat{F_1} \subseteq \widetilde{F_2}.$$

If Γ contains only one edge label, then $F_{1_\Gamma}^{\neq\emptyset} \subseteq F_{2_\Gamma}^{\neq\emptyset}$ if one of the following conditions hold:

1. $F_1 \subseteq \widetilde{F_2}$;
2. $\widehat{F_1} \subseteq \widetilde{F_2}$;
3. $^{+} \in F_1$, $^{+} \notin F_2$, $F_1 \subseteq \widetilde{\{\pi, \mathsf{di}, ^{+}\}}$ *and* $F_1 - \{^{+}\} \subseteq \widetilde{F_2}$;

Remark 4.2. In the original results [21, 19, 22, 41], all is not a considered operator. However, Theorem 4.1 can be generalized to include all by using the same reasoning as in the proof of Theorem 6.3.

Next, we look at $\mathcal{F}_1^{=\emptyset} \overset{?}{\subseteq} \mathcal{F}_2^{=\emptyset}$. This question can easily be reduced to question one. Indeed, this readily follows from the fact that $q \in F^{\neq\emptyset}$ iff $\neg q \in F^{=\emptyset}$. We thus have the following corollary.

Corollary 4.3. *Let F_1 and F_2 be fragments. Then, $F_1^{=\emptyset} \subseteq F_2^{=\emptyset}$ iff $F_1^{\neq\emptyset} \subseteq F_2^{\neq\emptyset}$.*

In the remainder of this chapter we are going devote our attention to $\mathcal{F}_1^{\subseteq} \overset{?}{\subseteq} F_2^{\subseteq}$. The result can be summarized as follows.

Theorem 4.4. *Let F_1 and F_2 be fragments. Then, $F_1^{\subseteq} \subseteq F_2^{\subseteq}$ iff $F_1 \subseteq \widetilde{F_2}$.*

Note that the fragments in Theorem 4.4 are not restricted, i.e., we also have results for fragments that contain one of the projections or one of the coprojections. By Theorem 4.4, subsumption among fragments under the containment modality behaves the same as subsumption for path queries. This is not obvious since we have already seen that under nonemptiness subsumption behaves very differently (cfr. Theorem 4.1).

4.1 Navigational graph query languages under the containment modality

In this section, we will investigate the expressive power of the various navigational features under the containment modality. Instead of working with just single containments, we work with the more general finite conjunctions of containments. For every fragment F, the family of Boolean queries expressible by finite conjunctions of containment statements using expressions from $\mathcal{N}(F)$ is denoted by $F^{\wedge\subseteq}$.

Our main result is the following:

Theorem 4.5. *For every two fragments F_1 and F_2, we have $F_1^{\wedge\subseteq} \subseteq F_2^{\wedge\subseteq}$ if and only if $F_1 \subseteq \widetilde{F_2}$. Furthermore, every separation can already be obtained with just a single containment.*

Theorem 4.4 is a direct corollary since all separations can already be obtained with just a single containment.

We call a nonbasic feature f *primitive* (under conjunctions of containments) if for every fragment F such that $f \notin \widetilde{F}$, we have $\{f\}^{\wedge\subseteq} \not\subseteq F^{\wedge\subseteq}$. In other words, just the feature, combined with the basic features, is enough to express some Boolean query that is not expressible without using the feature. We can then reformulate the above theorem as saying that *every nonbasic feature is primitive.* Next, we devote one section to every nonbasic feature.

4.1.1 Projection

We will first focus on the primitivity of the first projection. Up to completion, there are three maximal fragments lacking π_1: $\{-, \pi_2, {}^+\}$, $\{\mathsf{di}, \overline{\pi}_2, {}^+\}$, and $\{\mathsf{di}, {}^{-1}, {}^+\}$.

Let us first deal with $\{-, \pi_2, {}^+\}$. We show:

Proposition 4.6. *Let R be a relation name. The Boolean query $\pi_1(R^2) \circ R \subseteq R \circ \pi_1(R)$ is not in $\{-, \pi_2, {}^+\}^{\wedge\subseteq}$.*

To prove this proposition it suffices to reason only on the two graphs G_1 (top) and G_2 (bottom) shown in Figure 4.1.

Lemma 4.7. *Let e be a union-free expression in $\mathcal{N}(-, \pi_2)$. Then e is equivalent to $\emptyset$, id, R, R^2, $\pi_2(R)$, $\overline{\pi}_2(R)$, $\pi_2(R^2)$, $\overline{\pi}_2(R^2)$, $\pi_2(R) \circ R$, $\overline{\pi}_2(R) \circ R$, $\overline{\pi}_2(R^2) \circ \pi_2(R)$, $\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$ on the two graphs G_1 and G_2.*

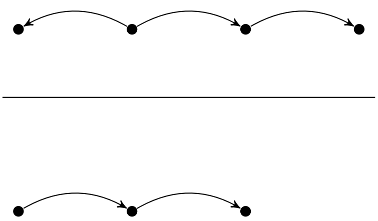


Figure 4.1: The graphs used to prove Proposition 4.6.

Proof. Table 4.1, 4.2 and 4.3 together show that $\emptyset$, id, R, R^2, $\pi_2(R)$, $\overline{\pi}_2(R)$, $\pi_2(R^2)$, $\overline{\pi}_2(R^2)$, $\pi_2(R) \circ R$, $\overline{\pi}_2(R) \circ R$, $\overline{\pi}_2(R^2) \circ \pi_2(R)$, $\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$ is closed under composition, difference and the second projection on the two graphs G_1 and G_2. This directly proves our lemma. $\square$

Observe that conjunctions of containments reduce to emptiness statements when difference is present.

Proposition 4.8. *Let F be a fragment with set difference. Then every Boolean query in $F^{\wedge\subseteq}$ can be expressed as the emptiness of an expression in $\mathcal{N}(F)$.*

We are now ready to prove Proposition 4.6.

Proof of Proposition 4.6. Let us denote the query $\pi_1(R^2) \circ R \subseteq R \circ \pi_1(R)$ by Q. Suppose for the sake of contradiction that Q is expressible in $\{-, \overline{\pi}_2, {}^+\}^{\wedge\subseteq}$. Hence, by Proposition 4.8 there exists $e = \emptyset$ in $\{-, \overline{\pi}_2, {}^+\}^{=\emptyset}$ that expresses Q. In the remainder of the proof, we will only work with G_1 and G_2, whence we can replace ${}^+$ with unions of compositions. By Lemma 4.7 we may assume that e is a union of expressions in the list: $\emptyset$, id, R, R^2, $\pi_2(R)$, $\overline{\pi}_2(R)$, $\pi_2(R^2)$, $\overline{\pi}_2(R^2)$, $\pi_2(R) \circ R$, $\overline{\pi}_2(R) \circ R$, $\overline{\pi}_2(R^2) \circ \pi_2(R)$, $\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$. All expressions in this set are nonempty on G_1 and G_2 simultaneously except $\emptyset$. Hence, $e = \emptyset$ cannot distinguish G_1 and G_2. This, however, contradicts that $Q(G_1)$ is false and $Q(G_2)$ is true. $\square$

Next we turn our attention to deal with the fragment $\{\mathsf{di}, {}^{-1}, {}^+\}$. We show the following proposition.

Proposition 4.9. *Let R be a relation name. The Boolean query $R \circ \pi_1(R) \subseteq \mathsf{id}$ is not in $\{\mathsf{di}, {}^{-1}, {}^+\}^{\wedge\subseteq}$.*

Table 4.1: The list of expressions $\emptyset$, id, R, R^2, $\pi_2(R)$, $\overline{\pi}_2(R)$, $\pi_2(R^2)$, $\overline{\pi}_2(R^2)$, $\pi_2(R) \circ R$, $\overline{\pi}_2(R) \circ R$, $\overline{\pi}_2(R^2) \circ \pi_2(R)$, $\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$ is closed under composition up to equivalence on the graphs in Figure 4.1.

$\circ$	R	R^2	$\pi_2(R)$	$\overline{\pi}_2(R)$	$\pi_2(R^2)$
R	R^2	$\emptyset$	R	$\emptyset$	$\pi_2(R) \circ R$
R^2	$\emptyset$	$\emptyset$	R^2	$\emptyset$	R^2
$\pi_2(R)$	$\pi_2(R) \circ R$	$\emptyset$	$\pi_2(R)$	$\emptyset$	$\pi_2(R^2)$
$\overline{\pi}_2(R)$	$\overline{\pi}_2(R) \circ R$	R^2	$\emptyset$	$\overline{\pi}_2(R)$	$\emptyset$
$\pi_2(R^2)$	$\emptyset$	$\emptyset$	$\pi_2(R^2)$	$\emptyset$	$\pi_2(R^2)$
$\overline{\pi}_2(R^2)$	R	R^2	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\overline{\pi}_2(R)$	$\emptyset$
$\pi_2(R) \circ R$	$\emptyset$	$\emptyset$	$\pi_2(R) \circ R$	$\emptyset$	$\pi_2(R) \circ R$
$\overline{\pi}_2(R) \circ R$	R^2	$\emptyset$	$\overline{\pi}_2(R) \circ R$	$\emptyset$	$\emptyset$
$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\pi_2(R) \circ R$	$\emptyset$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\emptyset$	$\emptyset$
$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\overline{\pi}_2(R) \circ R$	R^2	$\pi_2(R^2)$	$\overline{\pi}_2(R)$	$\pi_2(R^2)$

$\circ$	$\overline{\pi}_2(R^2)$	$\pi_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$
R	$\overline{\pi}_2(R) \circ R$	R^2	$\emptyset$	$\overline{\pi}_2(R) \circ R$	$\pi_2(R) \circ R$
R^2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	R^2
$\pi_2(R)$	$\overline{\pi}_2(R) \circ \pi_2(R)$	$\pi_2(R) \circ R$	$\emptyset$	$\overline{\pi}_2(R^2) \circ \overline{\pi}_2(R)$	$\pi_2(R^2)$
$\overline{\pi}_2(R)$	$\overline{\pi}_2(R)$	$\emptyset$	$\overline{\pi}_2(R) \circ R$	$\emptyset$	$\overline{\pi}_2(R)$
$\pi_2(R^2)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\pi_2(R^2)$
$\overline{\pi}_2(R^2)$	$\overline{\pi}_2(R^2)$	$\pi_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\overline{\pi}_2(R)$
$\pi_2(R) \circ R$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\pi_2(R) \circ R$
$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$	R^2	$\emptyset$	$\overline{\pi}_2(R) \circ R$	$\emptyset$
$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\pi_2(R) \circ R$	$\emptyset$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\emptyset$
$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\overline{\pi}_2(R)$	$\emptyset$	$\overline{\pi}_2(R) \circ R$	$\emptyset$	$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$

Table 4.2: The list of expressions $\emptyset$, id, R, R^2, $\pi_2(R)$, $\overline{\pi}_2(R)$, $\pi_2(R^2)$, $\overline{\pi}_2(R^2)$, $\pi_2(R) \circ R$, $\overline{\pi}_2(R) \circ R$, $\overline{\pi}_2(R^2) \circ \pi_2(R)$, $\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$ is closed under difference up to equivalence on the graphs in Figure 4.1.

$-$	id	R	R^2	$\pi_2(R)$	$\overline{\pi}_2(R)$	$\pi_2(R^2)$
id	$\emptyset$	id	id	$\overline{\pi}_2(R)$	$\pi_2(R)$	$\overline{\pi}_2(R^2)$
R	R	$\emptyset$	R	R	R	R
R^2	R^2	R^2	$\emptyset$	R^2	R^2	R^2
$\pi_2(R)$	$\emptyset$	$\pi_2(R)$	$\pi_2(R)$	$\emptyset$	$\pi_2(R)$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$
$\overline{\pi}_2(R)$	$\emptyset$	$\overline{\pi}_2(R)$	$\overline{\pi}_2(R)$	$\overline{\pi}_2(R)$	$\emptyset$	$\overline{\pi}_2(R)$
$\pi_2(R^2)$	$\emptyset$	$\pi_2(R^2)$	$\pi_2(R^2)$	$\emptyset$	$\pi_2(R^2)$	$\emptyset$
$\overline{\pi}_2(R^2)$	$\emptyset$	$\overline{\pi}_2(R^2)$	$\overline{\pi}_2(R^2)$	$\overline{\pi}_2(R)$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\overline{\pi}_2(R^2)$
$\pi_2(R) \circ R$	$\pi_2(R) \circ R$	$\emptyset$	$\pi_2(R) \circ R$	$\pi_2(R) \circ R$	$\pi_2(R) \circ R$	$\pi_2(R) \circ R$
$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$	$\emptyset$	$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$
$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\emptyset$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\emptyset$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$
$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\emptyset$	$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\overline{\pi}_2(R)$	$\pi_2(R^2)$	$\overline{\pi}_2(R)$

$-$	$\overline{\pi}_2(R^2)$	$\pi_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$
id	$\pi_2(R^2)$	id	id	$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\overline{\pi}_2(R^2) \circ \pi_2(R))$
R	R	$\overline{\pi}_2(R) \circ R$	$\pi_2(R) \circ R$	R	R
R^2	R^2	R^2	R^2	R^2	R^2
$\pi_2(R)$	$\pi_2(R^2)$	$\pi_2(R)$	$\pi_2(R)$	$\pi_2(R^2)$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$
$\overline{\pi}_2(R)$	$\emptyset$	$\overline{\pi}_2(R)$	$\overline{\pi}_2(R)$	$\overline{\pi}_2(R)$	$\emptyset$
$\pi_2(R^2)$	$\pi_2(R^2)$	$\pi_2(R^2)$	$\pi_2(R^2)$	$\pi_2(R^2)$	$\emptyset$
$\overline{\pi}_2(R^2)$	$\emptyset$	$\overline{\pi}_2(R^2)$	$\overline{\pi}_2(R^2)$	$\overline{\pi}_2(R)$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$
$\pi_2(R) \circ R$	$\pi_2(R) \circ R$	$\emptyset$	$\pi_2(R) \circ R$	$\pi_2(R) \circ R$	$\pi_2(R) \circ R$
$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$	$\emptyset$	$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R) \circ R$
$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\emptyset$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\emptyset$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$
$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\pi_2(R^2)$	$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\emptyset$

Table 4.3: The list of expressions $\emptyset$, id, R, R^2, $\pi_2(R)$, $\overline{\pi}_2(R)$, $\pi_2(R^2)$, $\overline{\pi}_2(R^2)$, $\pi_2(R) \circ R$, $\overline{\pi}_2(R) \circ R$, $\overline{\pi}_2(R^2) \circ \pi_2(R)$, $\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$ is closed under the second projection up to equivalence on the graphs in Figure 4.1.

e	$\pi_2(e)$
$\emptyset$	$\emptyset$
id	id
R	$\pi_2(R)$
R^2	$\pi_2(R^2)$
$\pi_2(R)$	$\pi_2(R)$
$\overline{\pi}_2(R)$	$\overline{\pi}_2(R)$
$\pi_2(R^2)$	$\pi_2(R^2)$
$\overline{\pi}_2(R^2)$	$\overline{\pi}_2(R^2)$
$\pi_2(R) \circ R$	$\pi_2(R^2)$
$\overline{\pi}_2(R) \circ R$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$
$\overline{\pi}_2(R^2) \circ \pi_2(R)$	$\overline{\pi}_2(R^2) \circ \pi_2(R)$
$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$	$\overline{\pi}_2(\overline{\pi}_2(R^2) \circ \pi_2(R))$

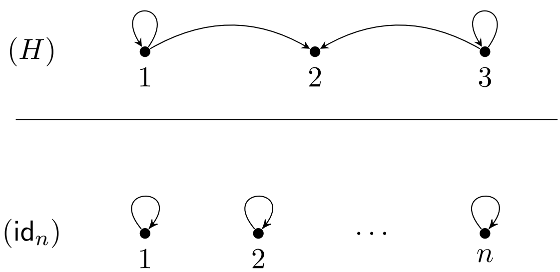


Figure 4.2: Graphs used in the proof of Proposition 4.9.

To prove this proposition it suffices to reason on the three finite graphs called K_3, H and id_3. The graphs H and id_3 are shown in Figure 4.2. The edges in these graphs are all understood to be labeled by the same relation name R. Note that K_3 is the complete graph (with loops) on 3 nodes shown in Figure 3.2; in general we use K_n to denote the complete graph on n nodes. On a complete graph, every path query invariant under isomorphisms can return only $\emptyset$, id, di, or all. Given the connection with the 3-variable fragment of first-order logic mentioned earlier, the following lemma is obvious.

Lemma 4.10. *On the class of complete graphs with at least 3 nodes, every expression in* $\mathcal{N}(\mathsf{di}, ^{-1}, -)$ *is equivalent to* $\emptyset$, id, di *or* all.

For expressions in $\mathcal{N}(\mathsf{di}, ^{-1})$ in particular, the outcome on K_3 may determine the complete behavior on all graphs, in the sense of the following lemma.

In the proof, and also later in the proof of Proposition 4.18, we frequently use monotonicity (cf. Section 3).

Lemma 4.11. *Let* e *be an expression in* $\mathcal{N}(\mathsf{di}, ^{-1})$. *We have:*

1. *If* $e(K_3) = \emptyset$ *then* $e \equiv \emptyset$;
2. *If* $e(K_3) = \mathsf{di}(K_3)$ *then* $e \equiv \mathsf{di}$;
3. *If* $e(K_3) = \mathsf{id}(K_3)$ *then* $e \equiv \mathsf{id}$.

Proof. In what follows, the proofs are labeled according to the numbers in the lemma.

1. Let G be a graph. There is a natural number $n \geq 3$ such that $G \subseteq K_n$. By Lemma 4.10, we have $e(K_n) = \emptyset$. Hence, because e is monotone, also $e(G) = \emptyset$.

2. We can write $e = \cup_{i=1}^{n} e_i$ as a union of union-free expressions, since union distributes over composition and converse. By Lemma 4.10, there must exist i such that $e_i(K_3)$ is equal to $\mathsf{di}(K_3)$. Furthermore, for every $j = 1, \ldots, n$, $e_i(K_3)$ cannot equal $\mathsf{id}(K_3)$ or $\mathsf{all}(K_3)$. If $e_j(K_3) = \emptyset$, then $e_j \equiv \emptyset$ by the previous case.

   So, we may assume that $e_j(K_3) = \mathsf{di}(K_3)$ for $j = 1, \ldots, n$. Take such an e_j. We know $e_j \not\equiv \mathsf{id}$ so e_j can be written as $H_1 \circ \ldots \circ H_l$ with $H_k \in \{R, R^{-1}, \mathsf{di}\}$. Indeed, this is possible since $(R \circ S)^{-1} \equiv S^{-1} \circ R^{-1}$. If $l \geq 2$, the first composition already yields all on K_3. Indeed, $R^{-1}(K_3) = R(K_3) = \mathsf{all}(K_3)$ and $R^2(K_3) = \mathsf{di} \circ R(K_3) = R \circ \mathsf{di}(K_3) = \mathsf{di}^2(K_3) = \mathsf{all}(K_3)$. Composing $\mathsf{all}(K_3)$ with $\mathsf{all}(K_3)$ or $\mathsf{di}(K_3)$ is again $\mathsf{all}(K_3)$. Thus $e_j(K_3) = \mathsf{all}(K_3)$ which is impossible.

   Hence, $l = 1$. Here, H_1 has to be di, because $R(K_3) = R^{-1}(K_3) = \mathsf{all}(K_3)$.

3. This case is similar to the previous case. □

Let us now look at the outcome of expressions on the graph id_3.

Lemma 4.12. *Let $e \not\equiv \emptyset$ be a union-free expression in $\mathcal{N}(\mathsf{di}, ^{-1})$.*

1. *If* di *occurs in e, then* $\mathsf{di}(\mathsf{id}_3) \cap e(\mathsf{id}_3) \neq \emptyset$*;*

2. *If* di *does not occur in e, or it occurs at least twice, then* $\mathsf{id}(\mathsf{id}_3) \cap e(\mathsf{id}_3) \neq \emptyset$*.*

Proof. In what follows, the proofs are labeled according to the numbers in the lemma.

1. Write $e = q_1 \circ \mathsf{di} \circ q_2$ where q_1 is di-free. Note that q_1 or q_2 may be id. Since di-free expressions can be evaluated in a loop, $(1,1)$ is in $q_1(\mathsf{id}_3)$, whence $(1,2)$ is in $q_1 \circ \mathsf{di}(\mathsf{id}_3)$. Furthermore, if there is an odd number of di occurrences in q_2, $(2,3) \in q_3(\mathsf{id}_3)$, and otherwise $(2,2) \in q_2(\mathsf{id}_3)$. Indeed, every di-free sub expression can be evaluated in a loop, and on every di application, one can jump from 2 to 3 and vice versa. We may thus conclude that $(1,2)$ or $(1,3)$ is in $e(\mathsf{id}_3)$.

2. If e contains no di applications, then $e = R^n$ on id_3 for some positive n, since e is not equivalent to $\emptyset$ and id_3 is symmetrical. Hence, $e(\mathsf{id}_3) = R(\mathsf{id}_3) = \mathsf{id}(\mathsf{id}_3)$.

   If e contains at least two di applications, then we can write $e = q_1 \circ \mathsf{di} \circ q_2 \circ \mathsf{di} \circ q_3$ so that q_1 and q_3 are di-free. Now $(1,1)$ is in $q_1(\mathsf{id}_3)$ and in $q_3(\mathsf{id}_3)$. Hence, $(1,2) \in q_1 \circ \mathsf{di}(\mathsf{id}_3)$ and $(3,1)$ and $(2,1)$ in $\mathsf{di} \circ q_3(\mathsf{id}_3)$. When di occurs an odd number of times in q_2, then $(2,3) \in q_3(\mathsf{id}_3)$; when it occurs an even number of times, $(2,2) \in q_2(\mathsf{id}_3)$. We may thus conclude that $(1,1) \in e(\mathsf{id}_3)$.

$\square$

We next look at the outcome of expressions on the graph H. The make the proof more readable we also use composition on the level of edges, e.g., $(1,2) \circ (2,3) = (1,3)$ and $(1,1) \circ (1,2) \circ (2,4) = (1,4)$.

Lemma 4.13. *Let $e \not\equiv \emptyset$ be a union-free expression in $\mathcal{N}(\mathsf{di}, ^{-1})$.*

1. *If $e \not\equiv \mathsf{di}$ and* di *occurs exactly once in e, then* $\mathsf{id}(H) \cap e(H) \neq \emptyset$*;*

2. *If $e \not\equiv \mathsf{id}$ and e is* di*-free, then* $\mathsf{di}(H) \cap e(H) \neq \emptyset$*.*

Proof. In what follows, the proofs are labeled according to the numbers in the lemma.

1. Write $e = e_1 \circ \mathsf{di} \circ e_2$ where e_1 and e_2 are di-free. One of e_1 or e_2 may be id, but not both, since $e \not\equiv \mathsf{di}$. We will now consider all the possible scenarios for e_1 and e_2. Note that on H, every nonempty di-free expression q can be evaluated in a loop, i.e., $(1,1),(3,3) \in q(H)$.

   - If $e_1 = q_1 \circ R$, where q_1 is di-free, then $(1,1) \circ (1,2) \circ (2,1) \circ (1,1) \in q_1 \circ R \circ \mathsf{di} \circ e_2(H)$. Hence $(1,1) \in e_1 \circ \mathsf{di} \circ e_2(H)$.
   - If $e_2 = R^{-1} \circ q_2$, where q_2 is di-free, then $(1,1) \circ (1,2) \circ (2,1) \circ (1,1) \in e_1 \circ \mathsf{di} \circ R^{-1} \circ q_2(H)$. Hence $(1,1) \in e_1 \circ \mathsf{di} \circ e_2(H)$.
   - If $e_1 = R^{-1} \circ R^{-n}$ and $e_2 = \mathsf{id}$, where n may be zero, then $(2,1) \circ (1,1) \circ (1,2) \in R^{-1} \circ R^{-n} \circ \mathsf{di}(H)$. Hence $(2,2) \in e_1 \circ \mathsf{di} \circ e_2(H)$.
   - If $e_1 = \mathsf{id}$ and $e_2 = R^n \circ R$, where n may be zero, then $(2,1) \circ (1,1) \circ (1,2) \in \mathsf{di} \circ R^n \circ R(H)$. Hence $(2,2) \in e_1 \circ \mathsf{di} \circ e_2(H)$.
   - If $e_1 = R^{-1} \circ R^{-n}$ and $e_2 = R^m \circ R$, where n and m may be zero, then $(2,1) \circ (1,1) \circ (1,3) \circ (3,3) \circ (3,2) \in R^{-1} \circ R^{-n} \circ \mathsf{di} \circ R^m \circ R(H)$. Hence $(2,2) \in e_1 \circ \mathsf{di} \circ e_2(H)$.
   - If $e_1 = q_1 \circ R \circ R^{-1} \circ R^{-n}$, where n may be zero, then $(1,1) \circ (1,2) \circ (2,3) \circ (3,3) \circ (3,1) \in q_1 \circ R \circ R^{-1} \circ R^{-n} \circ \mathsf{di}(H)$. Hence $(1,1) \in e_1 \circ \mathsf{di} \circ e_2(H)$.
   - If $e_2 = R^n \circ R \circ R^{-1} \circ q_2$, where n may be zero, then $(3,1) \circ (1,1) \circ (1,2) \circ (2,3) \circ (3,3) \in \mathsf{di} \circ R^n \circ R \circ R^{-1} \circ q_2(H)$. Hence $(3,3) \in e_1 \circ \mathsf{di} \circ e_2(H)$.

2. There are three possibilities.

   - If e can be written as $q_1 \circ R \circ R^{-1} \circ q_2$, where q_1 and q_2 may be id, then $(1,1) \circ (1,2) \circ (2,3) \circ (3,3) \in q_1 \circ R \circ R^{-1} \circ q_2(H)$. Hence, $(1,3) \in e(H)$.
   - If $e = R^n \circ R$ where n may be zero, then $(1,1) \circ (1,2) \in R^n \circ R(H)$. Hence $(1,2) \in e(H)$.
   - If e can be written as $R^{-1} \circ q$ where q may be id, then $(2,1) \circ (1,1) \in R^{-1} \circ q(H)$. Hence $(2,1) \in e(H)$.

□

We are now ready to prove Proposition 4.9.

Proof of Proposition 4.9. Let us denote the Boolean query $R \circ \pi_1(R) \subseteq \mathsf{id}$ by Q. Suppose for the sake of contradiction, that the conjunction $e_1 \subseteq$

$f_1 \wedge \cdots \wedge e_n \subseteq f_n$ expresses Q. We assume that no containment is trivial (a *trivial* containment is always true). Notice that $Q(K_3) = \mathit{false}$. Thus there exists $1 \leq i \leq n$ such that $e_i(K_3) \not\subseteq f_i(K_3)$. Hence $f_i(K_3) \neq \mathsf{all}(K_3)$. In the remainder of the proof, we will only work on the graphs K_3, H and id_3, whence we can replace $^+$ with unions of compositions. We know that $f_i(K_3)$ is either $\emptyset$, $\mathsf{id}(K_3)$, or $f_i(K_3) = \mathsf{di}(K_3)$. We will now cover each of these scenarios and obtain a contradiction.

If $f_i(K_3) = \emptyset$, then $f_i \equiv \emptyset$, by Lemma 4.11. Since $Q(\mathsf{id}_3) = \mathit{true}$, it must be that $e_i(\mathsf{id}_3) \subseteq f_i(\mathsf{id}_3)$. Thus $e_i(\mathsf{id}_3) = \emptyset$, whence e_i is equivalent to $\emptyset$ by Lemma 4.12. This, however, contradicts that $e_i \subseteq f_i$ is not trivial.

If $f_i(K_3) = \mathsf{di}(K_3)$, then $f_i \equiv \mathsf{di}$ by Lemma 4.11. Write $e_i = \cup_{j=1}^m g_j$ with g_j union-free. Since $e_i \subseteq f_i$ is not trivial, there has to exist $1 \leq j \leq m$ such that $g_j \not\equiv \mathsf{di}$. If $g_j \equiv \mathsf{id}$, then certainly $e_i(\mathsf{id}_3) \not\subseteq \mathsf{di}(\mathsf{id}_3) = f_i(\mathsf{id}_3)$. The only case left to consider is that $g_j \not\equiv \mathsf{di}$ and $g_j \not\equiv \mathsf{id}$. If g_j contains zero or more than two di applications, then $e_i(\mathsf{id}_3) \cap \mathsf{id}(\mathsf{id}_3) \neq \emptyset$ by Lemma 4.12, whence we have $e_i(\mathsf{id}_3) \not\subseteq \mathsf{di}(\mathsf{id}_3) = f_i(\mathsf{id}_3)$. This, however, contradicts that $Q(\mathsf{id}_3) = \mathit{true}$. On the other hand, if g_j contains exactly one di application, then $e_i(H) \cap \mathsf{id}(H) \neq \emptyset$ by Lemma 4.13, whence we have $e_i(H) \not\subseteq \mathsf{di}(H) = f_i(H)$. This, however, contradicts that $Q(H) = \mathit{true}$.

If $f_i(K_3) = \mathsf{id}(K_3)$, then $f_i \equiv \mathsf{id}$ by Lemma 4.11. Again write $e_i = \cup_{j=1}^m g_j$ with g_j union-free. Since $e_i \subseteq f_i$ is not trivial, there has to exist $1 \leq j \leq m$ such that $g_j \not\equiv \mathsf{id}$. If g_j contains at least one di application, then $g_j(\mathsf{id}_3) \cap \mathsf{di}(\mathsf{id}_3) \neq \emptyset$ by Lemma 4.12, whence we have $e_i(\mathsf{id}_3) \not\subseteq \mathsf{id}(\mathsf{id}_3) = f_i(\mathsf{id}_3)$. However, this contradicts that $Q(\mathsf{id}_3) = \mathit{true}$. On the other hand, if g_j is di-free, then $g_j(H) \cap \mathsf{di}(H) \neq \emptyset$ by Lemma 4.13, whence we have $e_i(H) \not\subseteq \mathsf{id}(H) = f_i(H)$. However, this contradicts that $Q(H) = \mathit{true}$. □

Finally, we are free to deal with the fragment $\{\mathsf{di}, \overline{\pi}_2, ^+\}$. First, we show that π_2 (and thus also $\overline{\pi}_2$) can be eliminated in this fragment on K_3, H and id_3.

Lemma 4.14. *Let e be an expression in $\mathcal{N}(\mathsf{di})$. Then $\pi_2(e)$ is equivalent to $\emptyset$ or id on the three graphs K_3, H and id_3.*

Proof. In this proof, whenever we write "equivalent" we mean equal on the three graphs K_3, H and id_3. We proceed by induction on e. In the base case, $\pi_2(\emptyset) = \emptyset$ and $\pi_2(R) = \pi_2(\mathsf{di}) = \mathsf{id}$ on all three graphs.

If $e = e_1 \cup e_2$, then $\pi_2(e_1 \cup e_2) = \pi_2(e_1) \cup \pi_2(e_2)$. By induction $\pi_2(e_1)$ and $\pi_2(e_2)$ are equivalent to id or $\emptyset$. Clearly, $\pi_2(e_1) \cup \pi_2(e_2)$ is equivalent to $\emptyset$ only when both $\pi_2(e_1)$ and $\pi_2(e_2)$ are equivalent to $\emptyset$. In all other cases, $\pi_2(e_1) \cup \pi_2(e_2)$ is equivalent to id.

If $e = e_1 \circ e_2$, then $\pi_2(e_1 \circ e_2) = \pi_2(\pi_2(e_1) \circ e_2)$. By induction $\pi_2(\pi_2(e_1) \circ e_2)$ equals $\pi_2(\mathsf{id} \circ e_2) = \pi_2(e_2)$ or $\pi_2(\emptyset \circ e_2) = \emptyset$. □

We may thus conclude that $\mathcal{N}(\mathsf{di}, \overline{\pi}_2, {}^+) \equiv \mathcal{N}(\mathsf{di}, {}^+)$ on K_3, H and id_3. Therefore, $R \circ \pi_1(R) \subseteq \mathsf{id}$ is not expressible in $\{\mathsf{di}, \overline{\pi}_2, {}^+\}^{\wedge\subseteq}$ by Proposition 4.9.

Proposition 4.15. *Let R be a relation name. The Boolean query $R \circ \pi_1(R) \subseteq \mathsf{id}$ is not in $\{\mathsf{di}, \overline{\pi}_2, {}^+\}^{\wedge\subseteq}$.*

Now we reduce the primitivity of π_2 to π_1. First, we show that we can pull converse up from the edge labels to the top.

Lemma 4.16. *Let F be a fragment and let F' be F where π_i is replaced by π_{3-i} and $\overline{\pi}_i$ is replaced by $\overline{\pi}_{3-i}$. Let e be an expression in $\mathcal{N}(F)$ and let e' be the expression obtained by replacing every R application with R^{-1}. Then, there exists an expression $h \in \mathcal{N}(F')$ such that $e' \equiv h^{-1}$.*

Proof. We prove this by structural induction on the expression e. In the base case, $e' = e^{-1}$.

Suppose $e = e_1 \circ e_2$. Then, $e' = e_1' \circ e_2'$. By induction, there exists h_1 and h_2 in $\mathcal{N}(F')$ such that $e_1' \circ e_2' \equiv h_1^{-1} \circ h_2^{-1}$. The result now follows from the fact that $h_1^{-1} \circ h_2^{-1} \equiv (h_2 \circ h_1)^{-1}$.

Suppose $e = e_1 \diamond e_2$ where $\diamond \in \{\cup, -\}$. Then, $e' = e_1' \diamond e_2'$. By induction there exists h_1 and h_2 in $\mathcal{N}(F')$ such that $e_1' \diamond e_2' \equiv h_1^{-1} \diamond h_2^{-1}$. The result now follows from the fact that $h_1^{-1} \diamond h_2^{-1} \equiv (h_1 \diamond h_2)^{-1}$.

Suppose $e = \pi_i(e_1)$. Then $e' = \pi_i(e_1')$. By induction there exists h_1 in $\mathcal{N}(F')$ such that $\pi_i(e_1') \equiv \pi_i(h_1^{-1})$. The result now follows from the fact that $\pi_i(h_1^{-1}) \equiv \pi_{3-i}(h_1) \equiv \pi_{3-i}(h_1)^{-1}$.

Suppose $e = e_1^+$. Then $e' = e_1'^+$. By induction there exists h_1 in $\mathcal{N}(F')$ such that $e_1'^+ \equiv (h_1^{-1})^+$. The result now follows from the fact that $(h_1^{-1})^+ \equiv (h_1^+)^{-1}$. □

Before we can reduce the primitivity of π_2 to π_1, we need the following lemma.

Lemma 4.17. *Let R be relation name, let F be a fragment and let F' be F where π_i is replaced by π_{3-i} and $\overline{\pi}_i$ is replaced by $\overline{\pi}_{3-i}$. If $e_1 \subseteq f_1 \wedge \ldots \wedge e_n \subseteq f_n \in F^{\wedge\subseteq}$ then $e_1' \subseteq f_1' \wedge \ldots \wedge e_n' \subseteq f_n' \in F'^{\wedge\subseteq}$ where e_i' and f_i' are obtained from e_i and f_i respectively by replacing R by R^{-1}.*

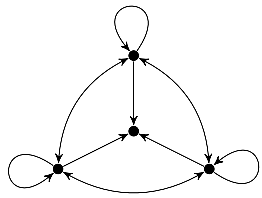


Figure 4.3: Graph used in the proof of Proposition 4.18.

Proof. By Lemma 4.16 there exists expressions $h_1, g_1, \ldots, h_n, g_n$ in $F'^{\wedge\subseteq}$ such that $e'_1 \subseteq f'_1 \wedge \ldots \wedge e'_n \subseteq f'_n \equiv h_1^{-1} \subseteq g_1^{-1} \wedge \ldots \wedge h_n^{-1} \subseteq g_n^{-1}$. The result now follows from the fact that $h_1^{-1} \subseteq g_1^{-1} \wedge \ldots \wedge h_n^{-1} \subseteq g_n^{-1} \equiv h_1 \subseteq g_1 \wedge \ldots \wedge h_n \subseteq g_n$. $\square$

Armed with the previous lemma, the primitivity of π_2 follows from the following two observations:

- By using the same notation as in Lemma 4.17, $(R \circ \pi_2(R^2))' \subseteq (\pi_2(R) \circ R)'$ is equal to $R^{-1} \circ \pi_2(R^{-1} \circ R^{-1}) \subseteq \pi_2(R^{-1}) \circ R^{-1}$ which is equivalent to $\pi_1(R^2) \circ R \subseteq R \circ \pi_1(R)$.
- Similarly, $(\pi_2(R) \circ R)' \subseteq \mathsf{id}'$ is equal to $\pi_2(R^{-1}) \circ R^{-1} \subseteq \mathsf{id}$ which in turn is equivalent to $R \circ \pi_1(R) \subseteq \mathsf{id}$.

4.1.2 Coprojection

First, we focus on the primitivity of the first coprojection. Up to completion, there are three maximal fragments lacking $\overline{\pi}_1$: $\{-, \overline{\pi}_2, ^+\}$, $\{\mathsf{di}, \overline{\pi}_2, ^+\}$ and $\{\mathsf{di}, ^{-1}, \cap, ^+\}$.[1]

For the first fragment, $\{\overline{\pi}_1\}^{\wedge\subseteq} \not\subseteq \{-, \overline{\pi}_2, ^+\}^{\wedge\subseteq}$ follows directly from Proposition 4.6, since $\pi_1 \in \widetilde{\{\overline{\pi}_1\}}$. For the second fragment, $\{\overline{\pi}_1\}^{\wedge\subseteq} \not\subseteq \{\mathsf{di}, \overline{\pi}_2, ^+\}^{\wedge\subseteq}$ follows directly from Proposition 4.15 for the same reason.

We now have our hands free for the fragment $\{\mathsf{di}, ^{-1}, \cap, ^+\}$. We are going to show:

Proposition 4.18. *Let R be a relation name. The Boolean query $\pi_1(R) \subseteq \pi_1(R \circ \overline{\pi}_1(R))$ is not in $\{\mathsf{di}, ^{-1}, \cap, ^+\}^{\wedge\subseteq}$.*

[1] Note that we do not have to consider fragments that contain $\overline{\pi}_2$ as well as π. Indeed, $\overline{\pi}_2(\pi_1(e)) \equiv \overline{\pi}_2(e)$.

To prove this proposition it suffices to reason on the complete graph K_3 and the graph G in Figure 4.3.

Lemma 4.19. *For every Boolean query $Q \in \{\mathsf{di}, {}^{-1}, \cap, {}^{+}\}^{\wedge\subseteq}$, Q cannot be true on G and false on K_3 simultaneously.*

Proof. Let Q be $e_1 \subseteq f_1 \wedge \cdots \wedge e_n \subseteq f_n \in \{\mathsf{di}, {}^{-1}, \cap, {}^{+}\}^{\wedge\subseteq}$. Suppose for the sake of contradiction that $Q(G)$ is true and $Q(K_3)$ is false. Since $Q(K_3) = \mathit{false}$ there exists $1 \leq i \leq n$ such that $e_i(K_3) \not\subseteq f_i(K_3)$. Hence $f_i(K_3) \neq \mathsf{all}(K_3)$. In the remainder of the proof, we will only work on the graphs K_3, K_4 and G, whence we can replace ${}^{+}$ with unions of compositions.

Since path queries in $\mathcal{N}(\mathsf{di}, {}^{-1}, \cap)$ are monotone, we have $f_i(K_3) \subseteq f_i(G) \subseteq f_i(K_4)$. This is used a number of times in the remainder of the proof.

Since $f_i(K_3) \neq \mathsf{all}(K_3)$, the only possibilities for $f_i(K_3)$ are $\mathsf{id}(K_3)$, $\mathsf{di}(K_3)$, and $\emptyset$.

If $f_i(K_3) = \mathsf{id}(K_3)$, then also $f_i(K_4) = \mathsf{id}(K_4)$ by Lemma 4.10. Hence, $f_i(G) \cap \mathsf{di}(G) = \emptyset$. Since $Q(K_3) = \mathit{false}$, we have $e_i(K_3) \not\subseteq f_i(K_3)$, so $e_i(K_3) \cap \mathsf{di}(K_3) \neq \emptyset$, whence we also have $e_i(G) \cap \mathsf{di}(G) \neq \emptyset$. Thus $e_i(G) \not\subseteq f_i(G)$ which contradicts that $Q(G) = \mathit{true}$.

If $f_i(K_3) = \mathsf{di}(K_3)$, then this case is analogous to the previous case.

Finally, if $f_i(K_3) = \emptyset$, then also $f_i(K_4) = \emptyset$ by Lemma 4.10, whence we also have $f_i(G) = \emptyset$. Since $Q(K_3) = \mathit{false}$, we have $e_i(K_3) \not\subseteq \emptyset$. Hence also $e_i(G) \not\subseteq \emptyset$, which contradicts that $Q(G) = \mathit{true}$. □

Proposition 4.18 is a corollary of Lemma 4.19 since $\pi_1(R) \subseteq \pi_1(R \circ \overline{\pi}_1(R))$ is true on G and false on K_3 simultaneously.

The primitivity of $\overline{\pi}_2$ follows from Lemma 4.17 and the following observation. By using the same notation as in Lemma 4.17, $(\pi_2(R))' \subseteq (\pi_2(\overline{\pi}_2(R) \circ R))'$ is equal to $\pi_2(R^{-1}) \subseteq \pi_2(\overline{\pi}_2(R^{-1}) \circ R^{-1})$, which in turn is equivalent to $\pi_1(R) \subseteq \pi_1(R \circ \overline{\pi}_1(R))$.

4.1.3 Intersection

Up to completion, the unique maximal fragment lacking intersection is $\{\mathsf{di}, \overline{\pi}, {}^{-1}, {}^{+}\}$. We now show:

Proposition 4.20. *Let R be a relation name. The Boolean query $R^2 \cap R \subseteq \mathsf{id}$ is not in $\{\mathsf{di}, \overline{\pi}, {}^{-1}, {}^{+}\}^{\wedge\subseteq}$.*

(ℓ_2)

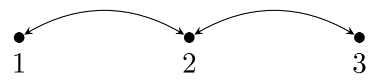


Figure 4.4: Graph used in the proof of Proposition 4.20.

To prove this proposition it suffices to reason on the finite graphs K_3 from Figure 3.2 and id_3 from Figure 4.2, and the graph ℓ_2 shown in Figure 4.4. We begin by showing that on these three graphs, projection and coprojection can be eliminated.

Lemma 4.21. *Let e be an expression in $\mathcal{N}(\mathsf{di}, \overline{\pi}, ^{-1})$. Then, $\pi_i(e)$, for $i = 1, 2$, is equivalent to $\emptyset$ or id on the three graphs K_3, ℓ_2 and id_3 simultaneously.*

Proof. In this proof, whenever we write "equivalent" we mean equal on the three graphs K_3, ℓ_2 and id_3. We proceed by induction on e. In the base case, $\pi_i(\emptyset) = \emptyset$ and $\pi_i(R) = \pi_i(R^{-1}) = \pi_i(\mathsf{di}) = \mathsf{id}$ on all three graphs.

If $e = \overline{\pi}_j(e_1)$, then $\pi_i(\overline{\pi}_j(e_1)) \equiv \mathsf{id} - \pi_j(e_1)$. By induction $\pi_j(e_1)$ is equivalent to id or $\emptyset$, whence $\mathsf{id} - \pi_j(e_1)$ also.

If $e = e_1 \cup e_2$, then $\pi_i(e_1 \cup e_2) = \pi_i(e_1) \cup \pi_i(e_2)$. By induction $\pi_i(e_1)$ and $\pi_i(e_2)$ are equivalent to id or $\emptyset$. Clearly, $\pi_i(e_1) \cup \pi_i(e_2)$ is equivalent to $\emptyset$ only when both $\pi_i(e_1)$ and $\pi_i(e_2)$ are equivalent to $\emptyset$. In all other cases, $\pi_i(e_1) \cup \pi_i(e_2)$ is equivalent to id.

If $e = e_1 \circ e_2$, there are two cases:

- Clearly, $\pi_1(e_1 \circ e_2) = \pi_1(e_1 \circ \pi_1(e_2))$. By induction, $\pi_1(e_1 \circ \pi_1(e_2))$ equals $\pi_1(e_1 \circ \mathsf{id}) = \pi_1(e_1)$ or $\pi_1(e_1 \circ \emptyset) = \emptyset$.
- Clearly, $\pi_2(e_1 \circ e_2) = \pi_2(\pi_2(e_1) \circ e_2)$. By induction $\pi_2(\pi_2(e_1) \circ e_2)$ equals $\pi_2(\mathsf{id} \circ e_2) = \pi_2(e_2)$ or $\pi_2(\emptyset \circ e_2) = \emptyset$.

□

Note that, since $\overline{\pi}(e) \equiv \mathsf{id} - \pi(e)$, the above lemma also holds for $\overline{\pi}(e)$.

We next look at the outcome of expressions on the graph ℓ_2.

Lemma 4.22. *Let e be a union-free expression in $\mathcal{N}(\mathsf{di}, ^{-1})$.*

1. *If e is di-free, $e \not\equiv \mathsf{id}$ and $e \not\equiv \emptyset$, then $e(\ell_2) \cap \mathsf{di}(\ell_2) \neq \emptyset$;*
2. *If di occurs exactly once in e and $e \not\equiv \mathsf{di}$, then $e(\ell_2) \cap \mathsf{id}(\ell_2) \neq \emptyset$.*

Proof. In what follows, the proofs are labeled according to the numbers in the lemma.

1. Since ℓ_2 is symmetrical, the converse operator does nothing and we can write $e = R^k$, with k positive since $e \not\equiv \mathsf{id}$. If k is odd, clearly $(1, 2) \in R^k(\ell_2)$. If k is even, $(1, 2) \in R^{k-1}(\ell_2)$ so $(1, 3) \in R^k(\ell_2)$.

2. First, we describe some outcome results for R^n on ℓ_2:

   - If n is odd, then $(1, 2)$, $(2, 1)$, $(2, 3)$ and $(3, 2)$ are in $R^n(\ell_2)$;
   - If n is even, then $(1, 1)$ and $(2, 2)$ are in $R^n(\ell_2)$;
   - If $n > 1$ is even, then $(1, 3)$ and $(3, 1)$ are in $R^n(\ell_2)$.

   Now write e as $R^n \circ \mathsf{di} \circ R^m$, where n and m may be zero (but not both).

   - If n and m are both odd, then $(2, 1) \circ (1, 3) \circ (3, 2) \in R^n \circ \mathsf{di} \circ R^m(\ell_2)$. Hence $(2, 2) \in e(\ell_2)$.
   - If n is even and m is odd, then $(1, 1) \circ (1, 2) \circ (2, 1) \in R^n \circ \mathsf{di} \circ R^m(\ell_2)$, whence $(1, 1)$ is also in $e(\ell_2)$.
   - If n is odd and m is even, then this case is symmetrical to the previous case.
   - If n is even and m is even, then n or m is strictly greater than one. If $n > 1$, then $(1, 3) \circ (3, 1) \circ (1, 1) \in R^n \circ \mathsf{di} \circ R^m(\ell_2)$, whence $(1, 1)$ is in $e(\ell_2)$. The case $m > 1$ is symmetrical. □

We can now give the proof of Proposition 4.20.

Proof of Proposition 4.20. Let us denote the Boolean query $R^2 \cap R \subseteq \mathsf{id}$ by Q. Observe that Q is false on K_3 but true on ℓ_2 and id_3.

Suppose for the sake of contradiction that the conjunction $e_1 \subseteq f_1 \wedge \cdots \wedge e_n \subseteq f_n$ expresses Q. We assume no containment is trivial, in the sense that $e_i \subseteq f_i$ are not equivalent to true or false.

Since $Q(K_3) = \mathit{false}$, there exists $1 \leq i \leq n$ such that $e_i(K_3) \not\subseteq f_i(K_3)$. In particular, $f_i(K_3) \neq \mathsf{all}(K_3)$. In the remainder of the proof we will only work on the graphs K_3, id_3 and ℓ_2, whence we can replace $^+$ with unions of compositions. Furthermore, by Lemma 4.21, we can eliminate $\overline{\pi}$ and π. So we may assume that e_i and f_i are in $\mathcal{N}(\mathsf{di}, ^{-1})$. Since $f_i(K_3) \neq \mathsf{all}(K_3)$ the three possibilities for $f_i(K_3)$ are $\emptyset$, $\mathsf{id}(K_3)$ or $\mathsf{di}(K_3)$. We will now cover these three possibilities and obtain a contradiction.

If $f_i(K_3) = \mathsf{di}(K_3)$, then $f_i \equiv \mathsf{di}$ by Lemma 4.11. Write $e_i = \cup_{j=1}^m g_j$ with g_j union-free. Since $e_i \subseteq f_i$ is not trivial, there has to exists $1 \leq j \leq m$ such that $g_j \not\equiv \mathsf{di}$. If g_j contains exactly one di application, then $e_i(\ell_2) \cap \mathsf{id}(\ell_2) \neq \emptyset$ by Lemma 4.22, whence we have $e_i(\ell_2) \not\subseteq \mathsf{di}(\ell_2) = f_i(\ell_2)$. This, however, contradicts that $Q(\ell_2) = \mathit{true}$. On the other hand, if g_k is di-free or has more than one di application, then $e_i(\mathsf{id}_3) \cap \mathsf{id}(\mathsf{id}_3) \neq \emptyset$ by Lemma 4.12, whence $e_i(\mathsf{id}_3) \not\subseteq \mathsf{di}(\mathsf{id}_3) = f_i(\mathsf{id}_3)$. This, however, contradicts that $Q(\mathsf{id}_3) = \mathit{true}$.

If $f_i(K_3) = \mathsf{id}(K_3)$, then $f_i \equiv \mathsf{id}$ by Lemma 4.11. Again write $e_i = \cup_{j=1}^m g_j$ with g_j union-free. Since $e_i \subseteq f_i$ is not trivial, there has to exist $1 \leq j \leq m$ such that $g_j \not\equiv \mathsf{id}$ and $g_j \not\equiv \emptyset$. If g_j is di-free, then $g_j(\ell_2) \cap \mathsf{di}(\ell_2) \neq \emptyset$ by Lemma 4.22, whence $e_i(\ell_2) \not\subseteq \mathsf{id}(\ell_2) = f_i(\ell_2)$. This, however, contradicts that $Q(\ell_2) = \mathit{true}$. On the other hand, if g_j contains at least one di application, then $g_j(\mathsf{id}_3) \cap \mathsf{di}(\mathsf{id}_3) \neq \emptyset$ by Lemma 4.12, whence $e_i(\mathsf{id}_3) \not\subseteq \mathsf{id}(\mathsf{id}_3) = f_i(\mathsf{id}_3)$. This, however, contradicts that $Q(\mathsf{id}_3) = \mathit{true}$.

Finally, if $f_i(K_3) = \emptyset$, then $f_i \equiv \emptyset$. Since $Q(\mathsf{id}_3) = \mathit{true}$, we have $e_i(\mathsf{id}_3) \subseteq f_i(\mathsf{id}_3) = \emptyset$. Thus $e_i(\mathsf{id}_3) = \emptyset$, whence e_i is equivalent to $\emptyset$ by Lemma 4.12 (we can again write e_i as a union of union-free expressions). This, however, contradicts that $e_i \subseteq f_i$ is not trivial. □

4.1.4 Difference

Up to completion, the unique maximal fragment lacking difference is $\{\cap, \overline{\pi}, \mathsf{di}, {}^{-1}, {}^{+}\}$, which we denote by NoDiff. We show the following proposition.

Proposition 4.23. *Let R be a relation name. The Boolean query $\mathsf{id} \subseteq R^2 \circ (R^2 - R) \circ R^2$ is not in $\mathrm{NoDiff}^{\wedge\subseteq}$.*

To prove this proposition, it suffices to reason on the complete graph K_3 and the bow tie graph B shown in Figure 3.2.

Lemma 4.24. *Every expression in $\mathcal{N}(\mathrm{NoDiff})$ is equivalent to $\emptyset$, id, di, R, $R \cap \mathsf{di}$ or all on K_3 and B simultaneously.*

Proof. In this proof all equivalences are meant to hold on K_3 and B only. We proceed by structural induction on the expression e. For $e \in \{\emptyset, \mathsf{id}, \mathsf{di}, R\}$ the result is trivial. Note that we do not have to consider transitive closure, since on a fixed finite number of graphs, one can replace the transitive closure operator by a finite union of compositions.

Suppose $e = e_1 \cup e_2$. The only nontrivial cases are $e_1 = \mathsf{id}$ and $e_2 = R \cap \mathsf{di}$; $e_1 = \mathsf{id}$ and $e_2 = R$; and $e_1 = \mathsf{di}$ and $e_2 = R$. In the first case,

$\mathsf{id} \cup (R \cap \mathsf{di})(K_3) = R(K_3)$ and $\mathsf{id} \cup (R \cap \mathsf{di})(B) = R(B)$. In the second case, $\mathsf{id} \cup R(K_3) = R(K_3)$ and $\mathsf{id} \cup R(B) = R(B)$. In the third case, $\mathsf{di} \cup R(K_3) = \mathsf{all}(K_3)$ and $\mathsf{di} \cup R(B) = \mathsf{all}(B)$.

Suppose $e = \overline{\pi}_i(e_1)$. If $e_1 \equiv \emptyset$, then $\overline{\pi}_i(e_1)(B) = \mathsf{id}(B)$ and $\overline{\pi}_i(e_1)(K_3) = \mathsf{id}(K_3)$. In any other case, $\overline{\pi}_i(e_1)(K_3) = \emptyset$ and $\overline{\pi}_i(e_1)(B) = \emptyset$, since for any $g \in \{\mathsf{id}, \mathsf{di}, R, R \cap \mathsf{di}, \mathsf{all}\}$, we have $\overline{\pi}_i(g)(K_3) = \overline{\pi}_i(g)(B) = \emptyset$.

Suppose $e = e_1 \cap e_2$. Then the only nontrivial case occurs where $e_1 \equiv R$ and $e_2 \equiv \mathsf{id}$. Here, $R \cap \mathsf{id}(K_3) = \mathsf{id}(K_3)$ and $R \cap \mathsf{id}(B) = \mathsf{id}(B)$ since K_3 and B both contain all self-loops.

Suppose $e = e_1 \circ e_2$. Since composing with $\emptyset$ results in $\emptyset$, and composing with id does nothing, we may focus on $e_1, e_2 \in \{\mathsf{di}, R, R \cap \mathsf{di}, \mathsf{all}\}$. It is clear that $R \cap \mathsf{di}(K_3) \subseteq e_i(K_3)$ and $R \cap \mathsf{di}(B) \subseteq e_i(B)$. Hence $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(K_3) \subseteq e_1 \circ e_2(K_3)$ and $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(B) \subseteq e_1 \circ e_2(B)$. Therefore, since $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(K_3) = \mathsf{all}(K_3)$ and $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(B) = \mathsf{all}(B)$, we obtain $e_1 \circ e_2(K_3) = \mathsf{all}(K_3)$ and $e_1 \circ e_2(B) = \mathsf{all}(B)$.

The case $e = e_1^{-1}$ is trivial since all of the possible intermediate results are symmetrical. □

We are now ready to prove the crucial lemma that directly implies Proposition 4.23. Indeed, Proposition 4.23 is a corollary of Lemma 4.25, since $\mathsf{id} \subseteq R^2 \circ (R^2 - R) \circ R^2$ is *false* on K_3 and *true* on B simultaneously.

Lemma 4.25. *For every Boolean query $Q \in \mathrm{NoDiff}^{\wedge\subseteq}$, Q cannot be false on K_3 and true on B simultaneously.*

Proof. It suffices to show that a single containment $e_1 \subseteq e_2$ is never *false* on K_3 and *true* on B simultaneously. Indeed, this behavior is then preserved under conjunction.

By Lemma 4.24, e_1 and e_2 are equivalent to $\emptyset$, id, di, R, $R \cap \mathsf{di}$ or all on K_3 and B simultaneously. From now on, equivalences are understood to be on K_3 and B only. We may assume that $e_1 \not\equiv \emptyset$ and $e_2 \not\equiv \mathsf{all}$, since otherwise, the query expressed by $e_1 \subseteq e_2$ is the trivial true query.

If e_2 is $\emptyset$, id or di, then by Lemma 4.24 we have $e_1(K_3) \subseteq e_2(K_3)$ iff $e_1(B) \subseteq e_2(B)$. Hence the query $e_1 \subseteq e_2$ cannot distinguish K_3 and B.

If $e_2 \equiv R$, then again by Lemma 4.24 we have $e_1(K_3) \subseteq R(K_3)$ iff $e_1(B) \subseteq R(B)$, except for the case where $e_1 \equiv \mathsf{di}$ or $e_1 \equiv \mathsf{all}$. However, in these cases, $e_1(K_3) \subseteq e_2(K_3)$.

If $e_2 \equiv R \cap \mathsf{di}$, then again by Lemma 4.24 we clearly have $e_1(K_3) \subseteq R \cap \mathsf{di}(K_3)$ iff $e_1(B) \subseteq R \cap \mathsf{di}(B)$ except maybe for the case where $e_1 \equiv \mathsf{di}$. However, in that case, again, $e_1(K_3) \subseteq e_2(K_3)$. □

4.1.5 Transitive closure

It seems obvious that transitive closure must be primitive, as it is the only operator that is not first-order definable. However, we want to establish primitivity across all fragments and all vocabularies. Thereto, we would ideally like to find a Boolean query over a single relation name that is not first-order expressible, but is expressible as a containment statement $e \subseteq f$ with e and f in $\mathcal{N}(^+)$. Obvious candidates, such as connectivity or cyclicity, seem not expressible in this manner, however. In other contexts, transitive closure may even *not* be a primitive operator. For example, every Boolean query over a single relation name that is expressible as the nonemptiness of an expression in $\mathcal{N}(\mathsf{di}, \pi, ^+)$ is already expressible without using transitive closure [22].

Nevertheless, we have found that the simple Boolean query "every node lies on a cycle" satisfies our needs:

Proposition 4.26. *Let R be a relation name. The Boolean query* $\mathsf{id} \subseteq R^+$ *is not first-order expressible.*

It follows that transitive closure is primitive. Proving this proposition is an exercise in Hanf locality [31], which requires finding the right graphs. We found the graphs G_1^ℓ and G_2^ℓ shown in Figure 4.5. In G_1^ℓ, every node lies on a cycle, but not in G_2^ℓ. Yet, for every natural number k and every $\ell > k$, the graphs G_1^ℓ and G_2^ℓ have the same k-neighborhood types with the same multiplicities, as summarized in Figure 4.6. Since first-order logic is Hanf-local, this implies that the Boolean query $\mathsf{id} \subseteq R^+$ is not first-order expressible.

4.1.6 The full relation

Up to completion, the unique maximal fragment lacking all is $\{^{-1}, -, ^+\}$, which we denote by NoAll. We now show:

Proposition 4.27. *Let R be a relation name. The Boolean query* $\mathsf{all} \subseteq R$ *is not in* $\text{NoAll}^{\wedge\subseteq}$.

This proposition can easily be proven by using the additivity of path queries expressible in $\mathcal{N}(\text{NoAll})$ (Lemma 3.2).

Proof of Proposition 4.27. Denote the Boolean query $\mathsf{all} \subseteq R$ by Q. Let G_1 and G_2 be two disjoint graphs, each consisting of just a single self-loop. Observe that Q is true on G_1 and G_2 but false on $G_1 \cup G_2$.

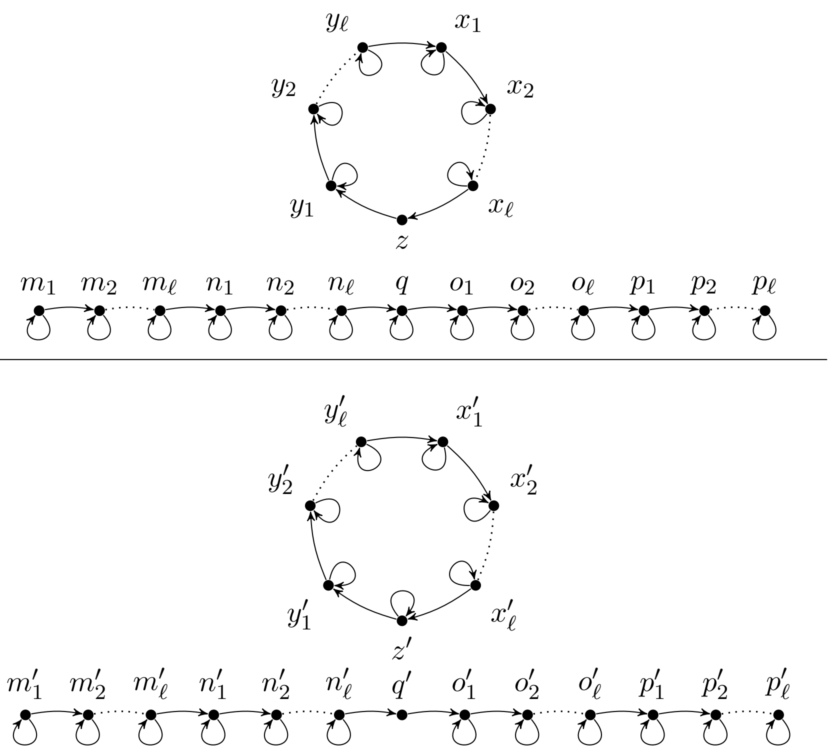


Figure 4.5: Graphs G_1^ℓ (top) and G_2^ℓ (bottom) used in the proof of Proposition 4.26.

Suppose for the sake of contradiction that the conjunction $e_1 \subseteq f_1 \wedge \cdots \wedge e_n \subseteq f_n$ expresses Q. Since $Q(G_1 \cup G_2) = \mathit{false}$, there exists $1 \leq j \leq n$ such that $e_j(G_1 \cup G_2) \not\subseteq f_j(G_1 \cup G_2)$. By additivity, $e_j(G_1) \cup e_j(G_2) \not\subseteq f_j(G_1) \cup f_j(G_2)$. Hence, $e_j(G_1) \not\subseteq f_j(G_1)$ or $e_j(G_2) \not\subseteq f_j(G_2)$, which contradicts that Q is true on both G_1 and G_2. $\square$

4.1.7 Diversity

Up to completion, there are two maximal fragments that lack diversity: $\{^{-1}, -, ^{+}\}$ and $\{^{-1}, \mathsf{all}, \overline{\pi}, \cap, ^{+}\}$. We show that in neither fragment, the Boolean query $\mathsf{di} \subseteq \emptyset$ ("there is only one node") is expressible as a conjunction of containments.

The fragment $\{^{-1}, -, ^{+}\}$ has set difference, so using Proposition 4.8, we can invoke our previous work on nonemptiness queries. Indeed, it has

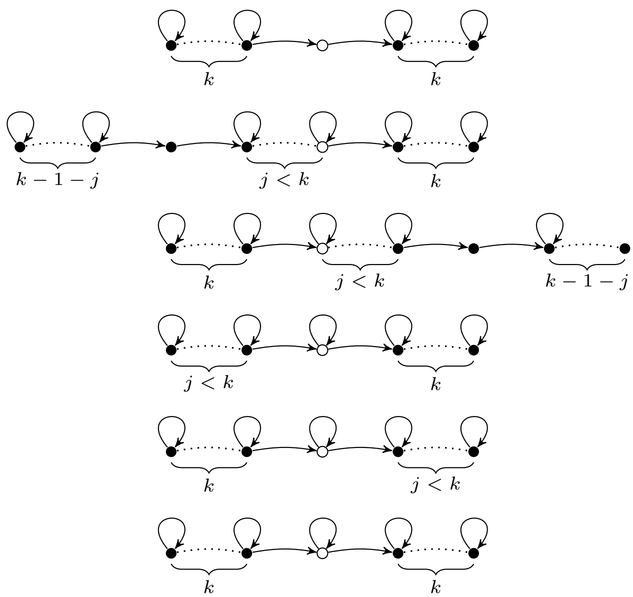


Figure 4.6: k-neighborhood types. The white node indicates the center of the neighborhood. Except for the bottom type, each type occurs exactly once in G_1^ℓ and in G_2^ℓ with $\ell > k$ (and letting j range from 0 to $k-1$). The bottom type occurs exactly $6\ell - 4k + 1$ times in both graphs.

already been shown [19, Proposition 5.4(1)] that the Boolean query $\mathsf{di} = \emptyset$ can not be expressed as the emptiness of an expression in $\mathcal{N}(^{-1}, -, ^{+})$.

For the other fragment, we show:

Proposition 4.28. *The Boolean query* $\mathsf{di} \subseteq \emptyset$ *is not in* $\{^{-1}, \mathsf{all}, \overline{\pi}, \cap, ^{+}\}^{\wedge\subseteq}$.

First, we prove the following simple lemma:

Lemma 4.29. *The graphs* id_1 *and* K_3 *are indistinguishable in* $\{^{-1}, \mathsf{all}, \overline{\pi}, \cap, ^{+}\}^{\wedge\subseteq}$.

Proof. Every expression in $\mathcal{N}(^{-1}, \mathsf{all}, \overline{\pi}, \cap, ^{+})$ is equivalent to id, all or $\emptyset$ on id_1 and K_3 simultaneously, which immediately implies the proposition.

The above claim is readily verified by induction. Indeed, the base case is trivial, and the induction step readily follows since the set $\{\mathsf{all}, \mathsf{id}, \emptyset\}$ is closed under all operators in the fragment. □

Proposition 4.28 is a direct corollary of Lemma 4.29 since $\mathsf{di} \subseteq \emptyset$ is true on id_1 and false on K_3 simultaneously.

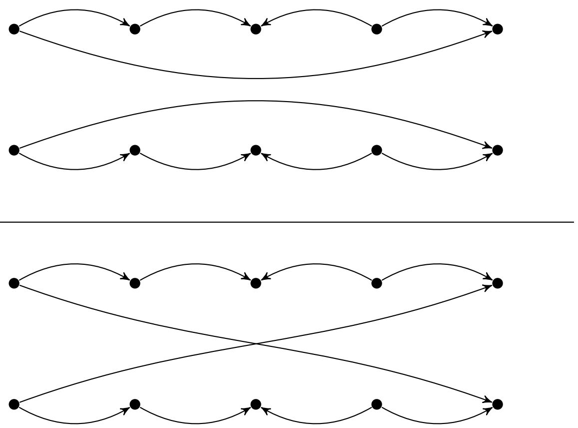


Figure 4.7: Graphs used in the proof of Proposition 4.30.

4.1.8 Converse

Up to completion, the unique maximal fragment that lacks converse is $\{\mathsf{di}, -, ^+\}$. We show:

Proposition 4.30. *Let R be a relation name. The Boolean query $R^2 \circ R^{-1} \circ R \subseteq R \cup R^2$ is not in $\{\mathsf{di}, -, ^+\}^{\wedge\subseteq}$.*

To prove this proposition it suffices to reason only on the two graphs G_1 (top) and G_2 (bottom) shown in Figure 4.7. We recall:

Lemma 4.31 ([19, Proposition 6.6])**.** *$e(G_1) \neq \emptyset$ implies $e(G_2) \neq \emptyset$ for every expression e in $\mathcal{N}(\mathsf{di}, -)$.*

With this lemma in hand we can now prove Proposition 4.30.

Proof of Proposition 4.30. Let us denote the Boolean query $R^2 \circ R^{-1} \circ R \subseteq R \cup R^2$ by Q. Observe that Q is true on G_1 but false on G_2. Suppose for the sake of contradiction that Q is in $\{\mathsf{di}, -, ^+\}^{\wedge\subseteq}$. Then by Proposition 4.8, Q is also expressible as $e = \emptyset$ with e in $\mathcal{N}(\mathsf{di}, -, ^+)$. Reasoning only on the two finite graphs G_1 and G_2, we may assume e does not use transitive closures, as we can replace these by unions of compositions. By assumption, $e(G_1)$ is empty but $e(G_2)$ is not. Equivalently, $e'(G_1) \neq \emptyset$ but $e'(G_2) = \emptyset$, with e' the expression $\mathsf{all} - (\mathsf{all} \circ e \circ \mathsf{all})$. This, however, contradicts Lemma 4.31. $\square$

5

Comparing different query languages under different base modalities

The goal of this chapter is to compare the different base modalities for different languages. Formally, for particular languages $\mathcal{F}_1, \mathcal{F}_2$ and modalities $\mathcal{M}_1, \mathcal{M}_2$ in $\{\neq\emptyset, =\emptyset, \subseteq\}$ we want to answer the following question:

$$\mathcal{F}_1^{\mathcal{M}_1} \overset{?}{\subseteq} \mathcal{F}_2^{\mathcal{M}_2}$$

Just as in Chapter 3, we only have to answer one of $\mathcal{F}_1^{=\emptyset} \overset{?}{\subseteq} \mathcal{F}_2^{\neq\emptyset}$ and $\mathcal{F}_1^{\neq\emptyset} \overset{?}{\subseteq} \mathcal{F}_2^{=\emptyset}$, since $\mathcal{A} \subseteq \mathcal{B}$ iff $\neg\mathcal{A} \subseteq \neg\mathcal{B}$ for every two families of Boolean queries $\mathcal{A}$ and $\mathcal{B}$.

Just as in Chapter 4, this question is interesting for the navigational graph query languages introduced in Section 2.1.1. In the remainder of this chapter, we will focus on the case where $\mathcal{C}$ are subsets of the set of navigational graph query fragments.

First, we compare $\neq\emptyset$ to $=\emptyset$ for (co)projection restricted fragments.

Theorem 5.1. *Let F_1 and F_2 be (co)projection restricted fragments. Then $F_1^{\neq\emptyset} \subseteq F_2^{=\emptyset}$ iff $F_1^{\neq\emptyset} \subseteq F_2^{\neq\emptyset}$ and $F_2^{\neq\emptyset} = F_2^{=\emptyset}$.*

Proof. The if direction follows by the transitivity of $\subseteq$. For the only if direction suppose that $F_2^{\neq\emptyset} \not\subseteq F_2^{=\emptyset}$ or $F_1^{\neq\emptyset} \not\subseteq F_2^{\neq\emptyset}$. In the former case, the proof follows from the proof of Theorem 3.11(2). Indeed, there is already a separating query within the most basic language. For the latter case, we may assume that $F_2^{\neq\emptyset} = F_2^{=\emptyset}$. Hence there is nothing to prove. $\square$

Next, we compare $\subseteq$ to $=\emptyset$ and $\subseteq$ to $\neq\emptyset$ for unrestricted fragments.

Theorem 5.2. *Let F_1 and F_2 be fragments. Then, we have:*

1. $F_1^{\subseteq} \subseteq F_2^{=\emptyset}$ *iff* $F_1^{\subseteq} \subseteq F_2^{\subseteq}$ *and* $F_2^{\subseteq} = F_2^{=\emptyset}$;
2. $F_1^{\subseteq} \subseteq F_2^{\neq\emptyset}$ *iff* $F_1^{\subseteq} \subseteq F_2^{\subseteq}$ *and* $F_2^{\subseteq} = F_2^{\neq\emptyset}$.

The proof of this theorem is analogous to the proof of Theorem 5.1. Here, we could drop the restrictions on the projections and coprojections since the expressive power of the containment modality has completely been characterized for all fragments.

In the remainder of this chapter we devote our attention to comparing $\neq\emptyset$ to $\subseteq$. We conjecture the following:

Conjecture 5.3. Let F_1 and F_2 be (co)projection restricted fragments. Then, $F_1^{\neq\emptyset} \subseteq F_2^{\subseteq}$ iff $F_1^{\neq\emptyset} \subseteq F_2^{\neq\emptyset}$ and $F_2^{\neq\emptyset} \subseteq F_2^{\subseteq}$.

We prove this conjecture for the most part in Section 5.1. The only open cases revolve around the fragments F_1 and F_2 where $F_1 = \{\pi\}$, $F_2 \subseteq \{^{-1}, \mathsf{di}, ^{+}\}$ and $\mathsf{all} \in \widetilde{F_2}$. In particular, if it would be true that

$$\{\pi\}^{\neq\emptyset} \nsubseteq \{\mathsf{di}, ^{-1}, ^{+}\}^{\subseteq} \tag{$\diamond$}$$

then Conjecture 5.3 would be entirely resolved. Although we have not been able to prove the equation marked with ($\diamond$), we have been able to prove it for the union-free subfragment of $\{\mathsf{all}, ^{-1}\}^{\subseteq} \subseteq \{\mathsf{di}, ^{-1}, ^{+}\}^{\subseteq}$. To do this, we observe that queries in $\mathcal{N}(\mathsf{all}, ^{-1})$ are expressible in CQ and show a monotone preservation theorem for $\mathrm{CQ}^{\subseteq}$. Using a similar strategy to prove ($\diamond$), without transitive closure on the right hand side, already involves a jump to the much more expressive UCQ with nonequalities.

We leave the comparison of emptiness to containment open. Note that this question is a more difficult version of comparing different fragments under containments. Indeed, to establish separations in this case, we have to use emptiness expressions instead of full containments. Since emptiness are special containments of the form $e \subseteq \emptyset$ for graph query languages, we thus have less power to establish the separations.

5.1 Comparing nonemptiness to containment for navigational graph query languages

This section is devoted to proving Conjecture 5.3 for nearly all fragments. The if direction clearly holds by the transitivity of $\subseteq$. For the only-if direction we consider its contrapositive. So, suppose that $F_2^{\neq\emptyset} \not\subseteq F_2^{\subseteq}$ or $F_1^{\neq\emptyset} \not\subseteq F_2^{\neq\emptyset}$. In the former case, the proof follows from the proof of Theorem 3.11(3). Indeed, there is already a separating query within the most basic language. So, we have the following:

Proposition 5.4. *Let F_1 and F_2 be (co)projection restricted fragments. If $F_2^{\neq\emptyset} \not\subseteq F_2^{\subseteq}$ then $F_1^{\neq\emptyset} \not\subseteq F_2^{\subseteq}$.*

Now we may assume that $F_2^{\neq\emptyset} \subseteq F_2^{\subseteq}$ and $F_1^{\neq\emptyset} \not\subseteq F_2^{\neq\emptyset}$. By Theorem 3.11(3), $\mathsf{all} \in \widetilde{F_2}$. Thus, if $- \in F_2$, then $F_2^{\subseteq} = F_2^{\neq\emptyset}$ by Theorem 3.11(4), whence there is nothing to prove. Hence, we do not have to consider languages with difference. Furthermore, we do not have to consider any sublanguage either since we are only trying to prove negative results here.

Since $F_1^{\neq\emptyset} \not\subseteq F_2^{\neq\emptyset}$ there must be at least one feature $f \in F_1$ that is missing in $\widetilde{F_2}$. We devote one section to each f and try to show that $\{f\}^{\neq\emptyset} \not\subseteq F_2^{\subseteq}$. In some cases this will not suffice, i.e., we need more features to establish the separation, while in a few cases, on the other hand, the result is still open.

5.1.1 Coprojection

In this section, we have a look at the case where coprojection is missing. Here, the largest F_2 we have to consider is $\{\mathsf{di}, {}^{-1}, \cap, {}^{+}\}$. Let G be the graph in Figure 4.3 and K_3 be the complete graph with three nodes. Notice that $\overline{\pi}_1(R) \neq \emptyset$ is true on G and false on K_3. By Lemma 4.19, this is not possible in $\{\mathsf{di}, {}^{-1}, \cap, {}^{+}\}^{\subseteq}$. Hence we have the following:

Proposition 5.5. *Let R be a relation name. The Boolean query $\overline{\pi}_1(R) \neq \emptyset$ is not in $\{\mathsf{di}, {}^{-1}, \cap, {}^{+}\}^{\subseteq}$.*

5.1.2 Difference

In this section, we have a look at the case where difference is missing. Here, the largest F_2 we have to consider is NoDiff. Let B be the bow tie and K_3

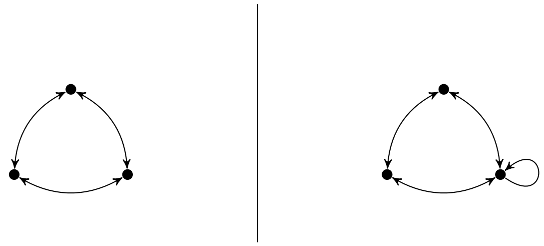


Figure 5.1: The graphs used to prove Proposition 5.7.

be the complete graph with with three nodes both displayed in Figure 3.2. Notice that $R^2 - R \neq \emptyset$ is true on B and false on K_3. By Lemma 4.25, this is not possible in $\text{NoDiff}^{\subseteq}$. Hence we have the following:

Proposition 5.6. *Let R be a relation name. The Boolean query $R^2 - R \neq \emptyset$ is not in* $\text{NoDiff}^{\subseteq}$.

5.1.3 Intersection

In this section, we have a look at the case where intersection is missing. Here, the largest F_2 we have to consider is $\{\mathsf{di}, \overline{\pi}, ^{-1}, ^{+}\}$. We are going to show:

Proposition 5.7. *Let R be a relation name. The query $R \cap id \neq \emptyset$ is not in $\{\mathsf{di}, \overline{\pi}, ^{-1}, ^{+}\}^{\subseteq}$.*

To prove this proposition it suffices to reason on the graphs A_1 (left) and A_2 (right) shown in Figure 5.1.

Lemma 5.8. *Let e be an expression in $\mathcal{N}(\mathsf{di}, ^{-1})$. On A_1 and A_2, e is equivalent to $\emptyset$, id, di, R or all simultaneously.*

Proof. We prove this lemma by structural induction on e. For id, di and R this is trivial. For R^{-1} note that A_1 and A_2 are symmetrical.

Suppose $e = e_1 \cup e_2$. Then the only troublesome cases are:

- $e_1 = \mathsf{id}$ and $e_2 = R$ or vice versa. Here, $e_1 \cup e_2(A_1) = \mathsf{all}(A_1)$ and $e_1 \cup e_2(A_2) = \mathsf{all}(A_2)$.

- $e_1 = R$ and $e_2 = \mathsf{di}$ or vice versa. Here, $e_1 \cup e_2(A_1) = R(A_1)$ and $e_1 \cup e_2(A_2) = R(A_2)$.

Suppose $e = e_1 \circ e_2$. Since composing with $\emptyset$ results in $\emptyset$, and composing with id does nothing, we may focus on $e_1, e_2 \in \{\mathsf{di}, R, \mathsf{all}\}$. It is clear that $R \cap \mathsf{di}(A_1) \subseteq e_i(A_1)$ and $R \cap \mathsf{di}(A_2) \subseteq e_i(A_2)$. Hence $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(A_1) \subseteq e_1 \circ e_2(A_1)$ and $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(A_2) \subseteq e_1 \circ e_2(A_2)$. Therefore, since $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(A_1) = \mathsf{all}(A_1)$ and $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(A_2) = \mathsf{all}(A_2)$, we obtain $e_1 \circ e_2(A_1) = \mathsf{all}(A_1)$ and $e_1 \circ e_2(A_2) = \mathsf{all}(A_2)$. □

For expressions in $\mathcal{N}(\mathsf{di}, {}^{-1})$, the outcomes on A_1 and A_2 may determine the complete behavior on all graphs, in the sense of the following lemma.

Lemma 5.9. *Let e be an expression in $\mathcal{N}(\mathsf{di}, {}^{-1})$.*

1. *If $e(A_1) = \emptyset$ then $e \equiv \emptyset$.*
2. *If $e(A_1) = \mathsf{id}(A_1)$ then $e \equiv \mathsf{id}$.*
3. *If $e(A_2) = R(A_2)$ then $e \equiv R$.*
4. *If $e(A_2) = \mathsf{di}(A_2)$ then $e \equiv \mathsf{di}$.*
5. *If $e(A_1) = \mathsf{di}(A_1)$ then $e(A_2) = \mathsf{di}(A_2)$ or $e(A_2) = R(A_2)$.*

The proof of Lemma 5.9 can be proven using the same technique as in the proof of Lemma 4.11. We are now ready for the proof of Proposition 5.7.

Proof of Proposition 5.7. Let Q be the Boolean query $R \cap \mathsf{id} \neq \emptyset$. Suppose for the sake of contradiction that Q is expressed by $e_1 \subseteq e_2 \in \{\mathsf{di}, {}^{-1}\}^{\subseteq}$. In the remainder of the proof we will only work on the graphs A_1 and A_2, whence we can replace $^{+}$ with unions of compositions. Notice that $Q(A_1) = \mathit{false}$. Thus, $e_1 \not\subseteq e_2(A_1)$, whence we have $e_2(A_1) \neq \mathsf{all}(A_1)$. Then, we know that $e_2(A_1)$ is equal to $\emptyset$, $\mathsf{id}(A_1)$ or $\mathsf{di}(A_1)$ by Lemma 5.8. We will now cover each of these scenarios and obtain a contradiction.

If $e_2(A_1) = \emptyset$, then $e_2 \equiv \emptyset$ by Lemma 5.9. Since $Q(A_1)$ is false, $e_1(A_1) \neq \emptyset$. Furthermore, since e_1 is monotone, $e_1(A_2) \neq \emptyset$, whence we have $e_1(A_2) \not\subseteq e_2(A_2)$. This, however, contradicts that $Q(A_2) = \mathit{true}$.

If $e_2(A_1) = \mathsf{id}(A_1)$, then $e_2 \equiv \mathsf{id}$ by Lemma 5.9. Since $Q(A_1)$ is false, $e_1(A_1) \cap \mathsf{di}(A_1) \neq \emptyset$. Furthermore, since e_1 is monotone, $e_1(A_2) \cap \mathsf{di}(A_2) \neq \emptyset$, whence we have $e_1(A_2) \not\subseteq e_2(A_2)$. This, however, contradicts that $Q(A_2) = \mathit{true}$.

Finally, if $e_2(A_1) = \mathsf{di}(A_1)$, then $e_2(A_2) = \mathsf{di}(A_2)$ or $e_2(A_2) = R(A_2)$ by Lemma 5.9. Suppose that $e_2(A_2) = \mathsf{di}(A_2)$, then by Lemma 5.9, $e_2 \equiv$

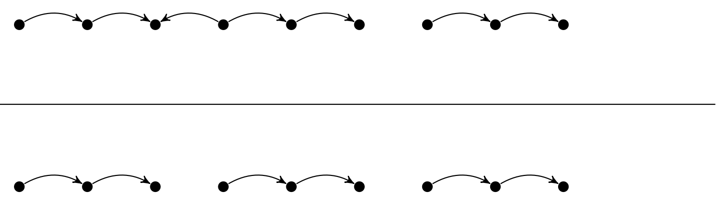


Figure 5.2: Graphs used in the proof of Proposition 5.10.

di. Since $Q(A_1)$ is false, $e_1(A_1) \cap \mathsf{id}(A_1) \neq \emptyset$. Furthermore, since e_1 is monotone, $e_1(A_2) \cap \mathsf{id}(A_2) \neq \emptyset$, whence we have $e_1(A_2) \not\subseteq e_2(A_2)$. This, however, contradicts that $Q(A_2) = \mathit{true}$. On the other hand, suppose that $e_2(A_2) = R(A_2)$. Since $Q(A_2) = \mathit{true}$, $e_1(A_2)$ equals $\emptyset$ or $R(A_2)$. In both cases, $e_1(A_1) \subseteq e_2(A_1)$ by Lemma 5.9. This, however, contradicts that $Q(A_1) = \mathit{false}$. $\square$

5.1.4 Converse

In this section, we have a look at the case where converse is missing. Here, there are two cases: $\pi \in \widetilde{F_2}$ or $\pi \notin \widetilde{F_2}$. In the former case, the largest F_2 we have to consider is $\{\pi, \cap, {}^+, -\}$, which we do not have to consider due to the presence of difference.

On the other hand, if $\pi \notin \widetilde{F_2}$, then F_2 is contained in $\{-, \cap, {}^+\}$ or $\{\mathsf{di}, {}^+\}$. Indeed, adding any other feature to F_2 adds ${}^{-1}$ or π. Since we do not have to consider $\{-, \cap, {}^+\}$, we focus on $\{\mathsf{di}, {}^+\}$.

Proposition 5.10. *Let R be a relation name. The query $R^2 \circ R^{-1} \circ R^2 \neq \emptyset$ is not in $\{\mathsf{di}, {}^+\}^{\subseteq}$.*

Proof. Let Q be the query $R^2 \circ R^{-1} \circ R^2 \neq \emptyset$. Let G_1 be the top and G_2 be the bottom graph in Figure 5.2. Since our graphs are finite, the set $\{(e(G_1), e(G_2)) \mid e \in \mathcal{N}(\mathsf{di}, {}^+)\}$ is finite. Hence, it can be computed by a computer program. Thus, we can also compute $\{(e_1 \subseteq e_2(G_1), e_1 \subseteq e_2(G_2)) \mid e_1, e_2 \in \mathcal{N}(\mathsf{di}, {}^+)\}$. We have verified that $(\mathit{true}, \mathit{false})$ is not in this set. Therefore, Q is not in $\{\mathsf{di}, {}^+\}^{\subseteq}$ since Q is true on G_1 and false on G_2. $\square$

5.1.5 Transitive closure

In this section we have a look at the case where transitive closure is missing. Here, the largest F_2 we have to consider is $\{\mathsf{di}, -, ^{-1}\}$, which we do not have to consider due to the presence of difference. Hence, for transitive closure there is nothing to prove.

5.1.6 Diversity

In this section, we have a look at the case where diversity is missing. Here, the largest F_2 we have to consider is $\{^{-1}, \mathsf{all}, \overline{\pi}, \cap, ^{+}\}$. Let id_1 be a single self-loop displayed in Figure 4.2 and K_3 be the complete graph with three nodes displayed in Figure 3.2. Notice that $\mathsf{di} \neq \emptyset$ is *false* on id_1 and *true* on K_3. By Lemma 4.29, this is not possible in $\{^{-1}, \mathsf{all}, \overline{\pi}, \cap, ^{+}\}^{\subseteq}$. Hence we have the following:

Proposition 5.11. *Let R be a relation name. The Boolean query $\mathsf{di} \neq \emptyset$ is not in $\{^{-1}, \mathsf{all}, \overline{\pi}, \cap, ^{+}\}^{\subseteq}$.*

5.1.7 Projection

In this section, we have a look at the case where one of the projections is missing. Here, the largest F_2 we have to consider is $\{\mathsf{di}, ^{-1}, ^{+}\}$. Unfortunately, we have *not* been able to prove Conjecture 5.3 for $\{\mathsf{di}, ^{-1}, ^{+}\}$ and most of its subfragments. However, we have been able to prove results for certain subsets of $F_2{}^{\subseteq}$. These results are summarized in Propositions 5.12 and 5.13. The following proof was suggested to us by Jelle Hellings [27].

Proposition 5.12. *Let R be a relation name. The Boolean query $R \circ \pi_1(R) \circ \mathsf{di} \circ M \circ \mathsf{di} \circ M \circ \mathsf{di} \circ \pi_2(R) \circ R \neq \emptyset$ where $M = \pi_2(R) \circ \pi_2(R^2 \circ \mathsf{di})$ is not in $\{\mathsf{di}, ^{+}\}^{\subseteq}$.*

Proof. Let Q be the query $R \circ \pi_1(R) \circ \mathsf{di} \circ M \circ \mathsf{di} \circ M \circ \mathsf{di} \circ \pi_2(R) \circ R \neq \emptyset$ where $M = \pi_2(R) \circ \pi_2(R^2 \circ \mathsf{di})$. Let G_1 be the left and G_2 be the right graph in Figure 5.3. Using the same brute-force method as outlined in the proof of Proposition 5.10 we can establish that no query in $\{\mathsf{di}, ^{+}\}^{\subseteq}$ can be *false* on G_1 and *true* on G_2. Therefore, Q is not in $\{\mathsf{di}, ^{+}\}^{\subseteq}$ since Q is *false* on G_1 and *true* on G_2. $\square$

Notice that we have used projection as well as diversity to establish the separation in Proposition 5.12. The case without using diversity in the separating query is still open.

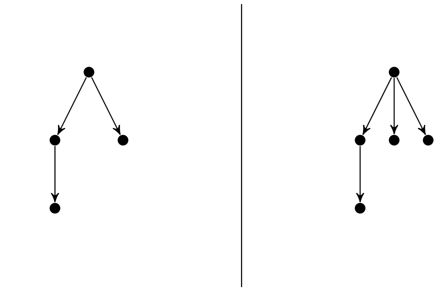


Figure 5.3: Graphs used in the proof of Proposition 5.12.

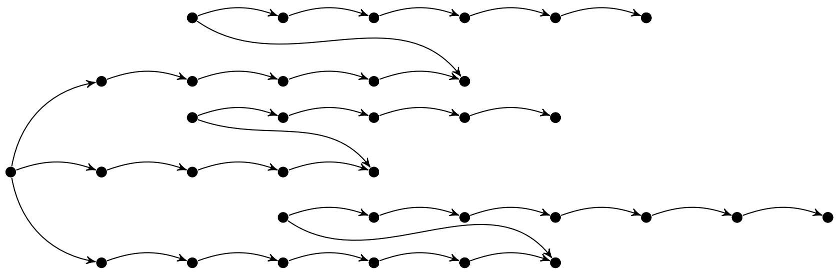


Figure 5.4: The Boolean query in Proposition 5.13 matches this graph pattern.

Next, we turn our attention to results where only projection is needed to establish separation.

Proposition 5.13. *Let R be a relation name. The Boolean query*

$$\pi_1(R^4 \circ \pi_2(\pi_1(R^4) \circ R)) \circ \pi_1(R^5 \circ \pi_2(\pi_1(R^5) \circ R)) \circ \pi_1(R^6 \circ \pi_2(\pi_1(R^6) \circ R)) \neq \emptyset$$

is not expressible by $e_1 \subseteq e_2$ where e_1 and e_2 have one of the following properties:

1. *e_1 and e_2 are binary relation queries where e_1 is monotone and e_2 is additive.*
2. *e_1 and e_2 are both union-free expressions in $\mathcal{N}(\mathsf{all}, ^{-1})$.*

The query in the above proposition seems rather complicated. However, it simply matches the graph pattern in Figure 5.4.

Since expressions in $\mathcal{N}({}^{-1}, {}^{+})$ are monotone as well as additive, we directly we have the following corollary:

Corollary 5.14. *Let R be a relation name. The Boolean query*

$$\pi_1(R^4 \circ \pi_2(\pi_1(R^4) \circ R)) \circ \pi_1(R^5 \circ \pi_2(\pi_1(R^5) \circ R)) \circ \pi_1(R^6 \circ \pi_2(\pi_1(R^6) \circ R)) \neq \emptyset$$

is not in $\{{}^{-1}, {}^{+}\}^{\subseteq}$.

To prove Proposition 5.13 we first need several technical lemmas:

Lemma 5.15. *Let e_1 and e_2 be binary relation queries. If e_1 is monotone, e_2 is additive and $e_1 \subseteq e_2$ is not constant, then $e_1 \subseteq e_2$ is not monotone.*

Proof. Since $e_1 \subseteq e_2$ is not constant, there exist two domain disjoint instances G_1 and G_2 such that $e_1(G_1) \subseteq e_2(G_1)$ and $e_1(G_2) \not\subseteq e_2(G_2)$. Since e_1 is monotone, $e_1(G_1) \cup e_1(G_2) \subseteq e_1(G_1 \cup G_2)$. Furthermore, since e_2 is additive, $e_2(G_1 \cup G_2) = e_2(G_1) \cup e_2(G_2)$. Hence, we have

$$e_1(G_1 \cup G_2) \subseteq e_2(G_1 \cup G_2) \Rightarrow e_1(G_1) \cup e_1(G_2) \subseteq e_2(G_1) \cup e_2(G_2).$$

The right hand containment is not possible, since G_1 and G_2 are domain disjoint and $e(G) \subseteq \mathrm{adom}(G)^2$ for any graph G and expression e. Therefore, $e_1(G_1 \cup G_2) \not\subseteq e_2(G_1 \cup G_2)$, whence $e_1 \subseteq e_2$ is not monotone. □

We have a similar lemma for union-free expressions in $\mathcal{N}({}^{-1}, \mathsf{all})$.

Lemma 5.16. *Let e_1 and e_2 are union-free expressions in $\mathcal{N}({}^{-1}, \mathsf{all})$. If $e_1 \subseteq e_2$ is monotone, then it is expressible in $\{{}^{-1}, \mathsf{all}\}^{\neq\emptyset}$.*

Lemma 5.16 directly follows from Theorem 8.1. Indeed, this is because union-free expressions in $\mathcal{N}({}^{-1}, \mathsf{all})$ are expressible by (unsafe) CQs.

Using Lemma 5.16 we can finally prove Proposition 5.13.

Proof of Proposition 5.13. Let Q be the Boolean query

$$\pi_1(R^4 \circ \pi_2(\pi_1(R^4) \circ R)) \circ \pi_1(R^5 \circ \pi_2(\pi_1(R^5) \circ R)) \circ \pi_1(R^6 \circ \pi_2(\pi_1(R^6) \circ R)) \neq \emptyset.$$

It is know that Q is not in $\{\mathsf{di}, {}^{-1}, {}^{+}\}^{\neq\emptyset}$ [21, Proposition 5.2].

(1) Suppose for the sake of contradiction that $e_1 \subseteq e_2 \in \{\mathsf{di}, {}^{-1}, {}^{+}\}^{\subseteq}$ expresses Q where e_1 is monotone and e_2 is additive. By Lemma 5.15, $e_1 \subseteq e_2$ has to be constant since Q is monotone. However, this is a contradiction, since Q is not constant.

(2) Suppose for the sake of contradiction that $e_1 \subseteq e_2 \in \{\mathsf{all}, {}^{-1}\}^{\subseteq}$ expresses Q where e_1 and e_2 are both union-free. Since Q is monotone, $e_1 \subseteq e_2$ is in $\{{}^{-1}, \mathsf{all}\}^{\neq\emptyset}$ by Lemma 5.16. Since $\{{}^{-1}, \mathsf{all}\}^{\neq\emptyset}$ is subsumed by $\{{}^{-1}, \mathsf{di}, {}^{+}\}^{\neq\emptyset}$, we have obtained a contradiction. $\square$

6 Closure under boolean connectives

In Section 2.2, we already observed that the question whether $\mathcal{F}^{=\emptyset}$ is subsumed by $\mathcal{F}^{\neq\emptyset}$ is equivalent to whether $\mathcal{F}^{\neq\emptyset}$ is closed under negation. A next logical step is to consider the logical negation of $\mathcal{F}^{\subseteq}$. To this end, consider the noncontainment modality: $\mathcal{F}^{\not\subseteq}$ contains the queries expressible in the form $q_1 \not\subseteq q_2$ with q_1 and q_2 k-ary queries in $\mathcal{F}$. In Section 6.1, we compare noncontainment to containment. We do not have to compare noncontainment to the other modalities since $\mathcal{A} \subseteq \mathcal{B}$ if and only if $\neg\mathcal{A} \subseteq \neg\mathcal{B}$.

Besides closure under negation, we have closure under conjunction. A family $\mathcal{B}$ of Boolean queries is *closed under conjunction* if for every pair of Boolean queries q_1 and q_2 in $\mathcal{B}$, also $q_1 \wedge q_2$ belongs to $\mathcal{B}$. We investigate this in Section 6.2.

6.1 Comparing containment to noncontainment

The goal of this chapter is to determine whether $F^{\subseteq}$ is closed under negation for fixed query languages $\mathcal{F}$. Formally, for particular query languages $\mathcal{F}$, we want to answer the following question:

$$\mathcal{F}^{\subseteq} \overset{?}{\subseteq} \mathcal{F}^{\not\subseteq}.$$

For all family of queries $\mathcal{F}$, we can infer $\mathcal{F}^{\not\subseteq} \subseteq \mathcal{F}^{\subseteq}$ if $\mathcal{F}^{\not\subseteq} \subseteq \mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{\subseteq}$. The first inequality is equivalent to $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset}$. Hence, from Proposition 3.1 we can infer that the containment modality is closed under negation if $\mathcal{F}$ has set difference, contains a never-empty query, and has tests or cylindrification. An alternative route could be taken using $\mathcal{F}^{\not\subseteq} \subseteq \mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{=\emptyset} \subseteq \mathcal{F}^{\subseteq}$, which can be done if $\mathcal{F}$ has set difference, complementation and cylindrification, and contains the empty query. Both routes suggest that closure under negation for the containment modality requires quite a strong query language. We will confirm this in the paragraphs below by showing that it does not hold for CQs or UCQs (as may be expected), and that it holds only for graph query language fragments that include both set difference and all.

In terms of a general negative result, we can only offer the straightforward inference that $\mathcal{F}^{\not\subseteq} \not\subseteq \mathcal{F}^{\subseteq}$ whenever $\mathcal{F}$ is additive and contains the empty query, and $\mathcal{F}^{\neq\emptyset}$ contains a non-constant query. Indeed, using the empty query we have $\mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{\not\subseteq}$, and Proposition 3.7 yields $\mathcal{F}^{\neq\emptyset} \not\subseteq \mathcal{F}^{\subseteq}$, whence we also have $\mathcal{F}^{\not\subseteq} \not\subseteq \mathcal{F}^{\subseteq}$.

Turning to conjunctive queries, $(\mathrm{U})\mathrm{CQ}^{\not\subseteq} \not\subseteq (\mathrm{U})\mathrm{CQ}^{\subseteq}$ follows immediately from the instance Z used in the proof of Theorem 3.9. On Z, every Boolean query in $\mathrm{UCQ}^{\subseteq}$ returns true, whereas the constant false query is easily expressed in $\mathrm{CQ}^{\not\subseteq}$.

For the navigational graph query language fragments, closure under negation of the containment modality can be characterized as follows.

Theorem 6.1. *Let F be a fragment. Then, $F^{\not\subseteq} \subseteq F^{\subseteq}$ iff $\mathsf{all} \in \widetilde{F}$ and $- \in F$.*

The if-direction directly follows from the general observations made in the beginning of this Section 6.1. To prove the only-if direction, recall that $\mathcal{N}(\mathrm{NoAll})$ is additive. Hence, $\mathrm{NoAll}^{\not\subseteq} \not\subseteq \mathrm{NoAll}^{\subseteq}$ also follows from the general observations made above. So, the only thing left to show is that $\mathrm{NoDiff}^{\not\subseteq} \subseteq \mathrm{NoDiff}^{\subseteq}$. Let B be the bow tie and K_3 be the complete graph with three nodes both displayed in Figure 3.2. Notice that $\mathsf{all} \not\subseteq R$ is true on B and false on K_3. By Lemma 4.25, this is not possible in $\mathrm{NoDiff}^{\subseteq}$. Hence, we have the following:

Lemma 6.2. *Let R be a relation schema. Then the Boolean query "R is not the full relation", formally, $\mathsf{all} \not\subseteq R$, is not in $\mathrm{NoDiff}^{\subseteq}$.*

6.2 Closure under conjunction

The goal of this chapter is to investigate the conjunctive closure of Boolean query families. Formally, for particular Boolean query families $\mathcal{B}$, we want to answer the following question:

$$\mathcal{B}^{\wedge} \overset{?}{\subseteq} \mathcal{B}$$

where $\mathcal{B}^{\wedge} = \{q_1 \wedge \ldots \wedge q_n \mid n \in \mathbb{N} \wedge q_i \in \mathcal{B} \text{ for } i = 1, \ldots, n\}$, the finite conjunctive closure of $\mathcal{B}$.

In the remainder of this section we will work with Boolean query families that stem from conjunctive queries and the navigational query languages (introduced in Section 2.1.1) under the base modalities.

For the navigational graph query languages, we will only consider the (co)projection restricted fragments, and check whether they are closed under conjunction in Section 6.2.1.

For (unions of) conjunctive queries, we will check whether the emptiness and nonemptiness modalities are closed under conjunction in Section 6.2.2. Furthermore, for conjunctive queries we also consider the same question under the containment modality. For unions of conjunctive queries, on the other hand, we leave this question open.

6.2.1 Navigational graph query languages

For the navigational graph query language fragments, which all include the union operation, closure under conjunction of the emptiness modality is trivial, since $(q_1 = \emptyset) \wedge (q_2 = \emptyset)$ is equivalent to $q_1 \cup q_2 = \emptyset$. For the nonemptiness modality, we can characterize closure under conjunction as follows.

Theorem 6.3. *Let F be a (co)projection restricted fragment. Then, $F_{\Gamma}^{\neq\emptyset}$ is closed under conjunction if and only if*

- *either* $\mathsf{all} \in \widetilde{F}$*, or*
- *the database schema Γ consists of a single binary relation name and $F \subseteq \{^+\}$.*

Proof. For the if-direction, we have two cases. If $\widetilde{F}$ has all then we can directly express $(e_1 \neq \emptyset) \wedge (e_2 \neq \emptyset)$ by $e_1 \circ \mathsf{all} \circ e_2 \neq \emptyset$. If $F \subseteq \{^+\}$ and Γ is a singleton $\{R\}$, the language $\mathcal{N}_{\Gamma}(F)$ is very simple. It is easy to see

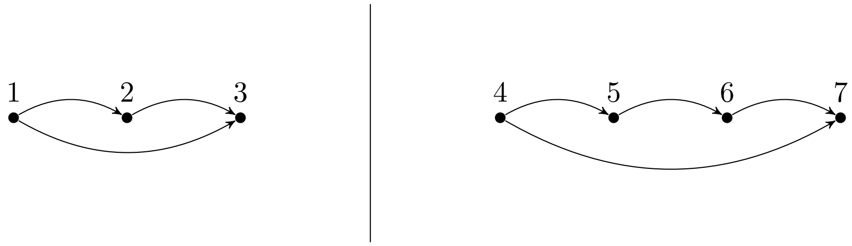


Figure 6.1: Graphs used to prove Lemma 6.5 and Theorem 6.12.

that for every expression e in this language there exists a natural number k such that $e \neq \emptyset$ is equivalent to $R^k \neq \emptyset$. The conjunction of $e_1 \neq \emptyset$ and $e_2 \neq \emptyset$ is then expressed using the maximum of the two numbers.

For the only-if direction, we also have two cases. First, if $\widetilde{F}$ does not have all and Γ is not a singleton, we can apply the following:

Lemma 6.4. *For two binary relation names R and T, the Boolean query "both R and T are nonempty", formally, $R \neq \emptyset \wedge T \neq \emptyset$, is not in* $\mathrm{NoAll}^{\neq\emptyset}$.

Proof. Denote the Boolean query $R \neq \emptyset \wedge T \neq \emptyset$ by q, and suppose q belongs to $\mathrm{NoAll}^{\neq\emptyset}$ as $e \neq \emptyset$. Let $G = \{R(a,b)\}$ and $H = \{T(c,d)\}$. Since $q(G) = q(H) = \mathit{false}$ and $q(G \cup H) = \mathit{true}$, we have $e(G) = \emptyset$, $e(H) = \emptyset$ and $e(G \cup H) \neq \emptyset$. By the Additivity Lemma, however, $e(G \cup H) = e(G) \cup e(H) = \emptyset$, a contradiction. $\square$

The second case is that $\widetilde{F}$ does not have all and $F \not\subseteq \{\mathsf{id}, {}^+\}$. Then $\widetilde{F}$ must contain at least one of the features: converse, projection, or intersection. The case with intersection is covered by the following:

Lemma 6.5. *For every binary relation name $R \in \Gamma$, the Boolean query $R^2 \cap R \neq \emptyset \wedge R^3 \cap R \neq \emptyset$ is not in* $\mathrm{NoAll}^{\neq\emptyset}$.

Proof. Denote the Boolean query $R^2 \cap R \neq \emptyset \wedge R^3 \cap R \neq \emptyset$ by q, and suppose q belongs to $\mathrm{NoAll}^{\neq\emptyset}$ as $e \neq \emptyset$. Let G be the left and H be the right graph in Figure 6.1. Since $q(G) = q(H) = \mathit{false}$ and $q(G \cup H) = \mathit{true}$, we have $e(G) = \emptyset$, $e(H) = \emptyset$ and $e(G \cup H) \neq \emptyset$. By the Additivity Lemma, however, $e(G \cup H) = e(G) \cup e(H) = \emptyset$, a contradiction. $\square$

The case with converse is covered by the following:

Lemma 6.6. *For every binary relation name $R \in \Gamma$, the Boolean query $R^2 \circ R^{-1} \circ R^3 \neq \emptyset \wedge R^3 \circ R^{-1} \circ R^2 \neq \emptyset$ is not in* $\mathrm{NoAll}^{\neq\emptyset}$.

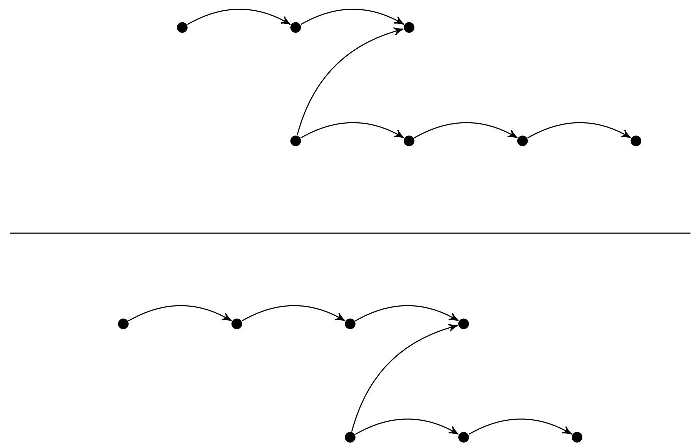


Figure 6.2: Graphs used for the proof of Lemma 6.6.

Before we can prove this we need the following technical lemma.

Lemma 6.7 ([25])**.** *There is no homomorphism from G_1 to G_2 and vice versa, where G_1 and G_2 are the top and bottom graphs of Figure 6.2.*

The lemma follows from the fact that different directed paths of the same length are cores which are not comparable with respect to homomorphisms [25].

We are now ready for the proof of Lemma 6.6.

Proof of Lemma 6.6. Denote the Boolean queries $R^2 \circ R^{-1} \circ R^3 \neq \emptyset$ and $R^3 \circ R^{-1} \circ R^2 \neq \emptyset$ by q_1 and q_2 respectively. Consider the graphs G_1 and G_2 shown at the top and bottom of Figure 6.2. For every graph G, we have $q_1(G) = \mathit{true}$ iff there is a homomorphism $G_1 \to G$, and similarly for q_2 and G_2. Hence, $q_1(G_2) = \mathit{false}$ and $q_2(G_1) = \mathit{false}$ by Lemma 6.7.

On the other hand, $q_1 \wedge q_2(G_1 \cup G_2) = \mathit{true}$. Now suppose that $q_1 \wedge q_2$ is expressed by $e \neq \emptyset \in \mathrm{NoAll}^{\neq\emptyset}$. Then, $e(G_1) = e(G_1) = \emptyset$ and $e(G_1 \cup G_2) \neq \emptyset$. By the Additivity Lemma, however, $e(G_1 \cup G_2) = e(G_1) \cup e(G_2) = \emptyset$, a contradiction. □

This lemma also covers the case with projection. Indeed, both conjuncts are in $\{^{-1}\}^{\neq\emptyset}$, which is known to be subsumed by $\{\pi\}^{\neq\emptyset}$ (cf. Theorem 4.1). Hence, the lemma also gives a conjunction of $\{\pi\}^{\neq\emptyset}$ queries that is not in $\mathrm{NoAll}^{\neq\emptyset}$. □

Under the containment modality, we can only offer the following general result:

Proposition 6.8. *Let F be a fragment. If $- \in F$, then $F^{\subseteq}$ is closed under conjunction.*

Indeed, we can express $e_1 \subseteq e_2 \wedge e_3 \subseteq e_4$ as $(e_1 - e_2) \cup (e_3 - e_4) \subseteq \emptyset$. At this point we have not been able to prove the converse of the above proposition, although we conjecture that set difference is indeed necessary. The challenge is to find a conjunction of Boolean queries in NoDiff$^{\subseteq}$ that is not in NoDiff$^{\subseteq}$. Even though we have not been able to find such a conjunction, we have been able to prove that certain subfragments NoDiff$^{\subseteq}$ are not closed under conjunction.

Proposition 6.9. *Let R be a relation name. The Boolean query $R^3 \subseteq \mathsf{id} \wedge R^2 \subseteq R$ is in $\{\mathsf{di}, ^{-1}, ^{+}\}^{\subseteq}$.*

Proof. Let Q be the Boolean query $R^3 \subseteq \mathsf{id} \wedge R^2 \subseteq R$, id_1 be the graph in Figure 4.2 and $c_2 = \{R(1,2), R(2,3)\}$ (the bottom graph in Figure 4.1). Suppose for the sake of contradiction that $e_1 \subseteq e_2 \in \{\mathsf{di}, ^{-1}, ^{+}\}^{\subseteq}$ expresses Q. Then, $e_2(K_3)$ equals $\mathsf{all}(K_3)$, $\mathsf{di}(K_3)$, $\mathsf{id}(K_3)$ or $\emptyset(K_3)$ by Lemma 4.10. In the remainder of the proof we will only work on the graphs K_3, c_2, c_2^+ and id_2, whence we can replace $^+$ with unions of compositions. We will now cover each of these scenarios and obtain a contradiction.

If $e_2(K_3) = \mathsf{all}(K_3)$, then $e_1(K_3) \subseteq e_2(K_3)$. We have thus obtained a contradiction, since $Q(K_3) = \textit{false}$.

If $e_2(K_3) = \mathsf{id}(K_3)$, then $e_2 \equiv \mathsf{id}$ by Lemma 4.11. Since $Q(c_2) = \textit{false}$, $e_1(c_2) \not\subseteq e_2(c_2)$, whence we have $e_1(c_2) \cap \mathsf{di}(c_2) \neq \emptyset$. Therefore, $e_1(c_2^+) \cap \mathsf{di}(c_2^+) \neq \emptyset$. We have thus obtained a contradiction, since $Q(c_2^+) = \textit{true}$.

The case where $e_2(K_3) = \mathsf{di}(K_3)$ is analogous.

If $e_2(K_3) = \emptyset$, then $e_2 \equiv \emptyset$. Clearly, when $e_1 \not\equiv \emptyset$, then $e_1(\mathsf{id}_2) \neq \emptyset$. We have thus contained a contradiction, since $Q(\mathsf{id}_2) = \textit{true}$. □

Unfortunately, we cannot generalize this result to include coprojection. This is because every query $e_1 \subseteq e_2 \wedge e_3 \subseteq \mathsf{id}$, such that $e_1(G) \neq \emptyset$ if $e_3(G) \not\subseteq \mathsf{id}(G)$, does not work to establish separation. Indeed, then $e_1 \subseteq e_2 \wedge e_3 \subseteq \mathsf{id}$ is equivalent with $e_1 \subseteq e_2 \circ \overline{\pi}_1(\mathit{all} \circ (e_3 \cap \mathsf{di}))$.

Next we look at another subfragment of NoDiff$^{\subseteq}$ that is not closed under conjunction.

Proposition 6.10. *Let R be a relation name. The Boolean query $R^3 \subseteq \emptyset \wedge R^2 \subseteq R$ is not expressible in $\{\cap, \mathsf{id}, \pi, ^{-1}, ^{+}\}^{\subseteq}$.*

Proof. Let Q be the Boolean query $R^3 \subseteq \emptyset \wedge R^2 \subseteq R$, id_1 be the graph in Figure 4.2, and $c_2 = \{R(1,2), R(2,3)\}$ (bottom graph in Figure 4.1). Suppose for the sake of contradiction that $e_1 \subseteq e_2 \in \{\cap, \mathsf{id}, \pi, ^{-1}, ^{+}\}^{\subseteq}$ expresses Q. In the remainder of the proof we will only work on the graphs c_2, c_2^+ and id_1, whence we can replace $^+$ with unions of compositions Remember that $e_i(\mathsf{id}_1) = R(\mathsf{id}_1)$ unless $e_i \equiv \emptyset$ for $i = 1, 2$. Therefore, if $e_2 \not\equiv \emptyset$ then $e_1(\mathsf{id}_1) \subseteq e_2(\mathsf{id}_1)$. We may thus conclude that $e_2 \equiv \emptyset$. Furthermore, if $e_1(c_2) = \emptyset$, then $e_1 \subseteq e_2$ does not express Q since $Q(c_2) = \mathit{false}$. On the other hand, if $e_1(c_2) \neq \emptyset$, then $e_1(c_2^+) \neq \emptyset$ since the language is monotone. Hence, $e_1 \subseteq e_2$ does not express Q since $Q(c_2^+) = \mathit{true}$. $\square$

Unfortunately, we cannot include coprojection here either. Every query of the form $e_1 \subseteq e_2 \wedge e_3 \subseteq \emptyset$, such that $e_1(G) \neq \emptyset$ if $e_3(G) = \emptyset$, does not work to establish separation. Indeed, then $e_1 \subseteq e_2 \wedge e_3 \subseteq \emptyset$ is equivalent with $e_1 \subseteq e_2 \circ \overline{\pi}_1(\mathit{all} \circ e_3)$.

6.2.2 Conjunctive queries

Under nonemptiness, both CQ and UCQ are clearly closed under conjunction. The construction is the same as the one used to express tests (proof of Theorem 3.9(3)).

Under emptiness, note that this modality is closed under conjunction if and only if the nonemptiness modality is closed under *disjunction*. This is clearly the case for UCQs (precisely because they are closed under union). For CQs closure under disjunction is captured by the following theorem.

Theorem 6.11. *Let Γ be a database schema. Then, $\mathrm{CQ}_\Gamma{}^{\neq\emptyset}$ is closed under disjunction if and only if Γ only contains at most two unary relations and no other n-ary relation names with $n \geq 2$.*

Proof. First, assume that Γ only contains unary relations, say $U_1, \ldots, U_n$. Then, Boolean queries in $\mathrm{CQ}_\Gamma{}^{\neq\emptyset}$ are equivalent to finite conjunctions of queries that test whether the intersection $\bigcap_{U \in A} U$ is nonempty for some $A \subseteq \Gamma$. Thus, if Γ only contains two unary relations, say U_1 and U_2, then $\mathrm{CQ}_\Gamma{}^{\neq\emptyset}$ only contains four Boolean queries.

- U_1 and U_2 are both nonempty;
- U_1 is nonempty;
- U_2 is nonempty;

- $U_1 \cap U_2$ is nonempty;

Now consider $q : q_1 \neq \emptyset \vee q_2 \neq \emptyset$ where q_1 and q_2 are both conjunctive queries over U_1 and/or U_2. Then q is equivalent to one of the following:

- U_1 and U_2 are both nonempty;
- U_1 is nonempty;
- U_2 is nonempty;
- $U_1 \cap U_2$ is nonempty;
- U_1 or U_2 is nonempty.

The first four queries are respectively expressed by $() \leftarrow U_1(x), U_2(y)$, $() \leftarrow U_1(x)$, $() \leftarrow U_2(x)$ and $() \leftarrow U_1(x), U_2(x)$. The last query, on the other hand, is equivalent to the constant true query since the empty instance is not allowed and there are only two relation names. We may thus conclude that q is also in $\mathrm{CQ}^{\neq\emptyset}$ as desired.

On the other hand, if Γ contains at least three unary relations, say U_1, U_2 and U_3, then we can consider the CQs $q_1 = () \leftarrow U_1(x), U_2(x)$ and $q_2 = () \leftarrow U_3(x)$. Clearly, $q_1 \vee q_2 \neq \emptyset$ cannot be expressed by the conjunction of intersection tests.

Finally, suppose that Γ contains an $n > 1$-ary relation name. The queries $q_1 = () \leftarrow R^3 \circ R^{-1} \circ R^2$ and $q_2 = () \leftarrow R^2 \circ R^{-1} \circ R^3$ are isomorphic to a query over Γ since we can transform them by replacing $R(x,y)$ with $R(z,\ldots,z,x,y)$. So we may assume that q_1 and q_2 are expressible in CQ_Γ. Suppose now that $q_1 \neq \emptyset \vee q_2 \neq \emptyset$ is expressible by a nonemptiness $q \neq \emptyset$ in $\mathrm{CQ}^{\neq\emptyset}$. Notice that $q_1 \neq \emptyset \vee q_2 \neq \emptyset \equiv q_1 \cup q_2 \neq \emptyset$, whence it is a UCQ. Since $q \sqsubseteq q_1 \cup q_2$, we have by the Sagiv–Yannakakis theorem [38] that, either $q \sqsubseteq q_1$ or $q \sqsubseteq q_2$. This, however, implies that $q_1(B_{q_2}) \neq \emptyset$ or $q_2(B_{q_1}) \neq \emptyset$. Hence, there is a homomorphism from B_{q_1} to B_{q_2} or vice versa. This contradicts Lemma 6.7 since B_{q_1} and B_{q_2} are isomorphic to the top and bottom graphs in Figure 6.2 respectively.

□

When Γ contains a binary relation, the result already follows from technical considerations regarding principal filters in the lattice of cores [25]. Indeed, a Boolean CQ is a principal filter and the disjunction of two Boolean CQs corresponds to the union of two principal filters. The result then follows from the fact that the union of two principal filters of incomparable cores is not principal.

Finally, we have a look at CQs under containment. We are going to show:

Theorem 6.12. *Let Γ be a database schema. Then, $\mathrm{CQ}_\Gamma{}^{\subseteq}$ is closed under conjunction if and only if Γ only contains one unary relation and no other n-ary relation names with $n \geq 2$.*

We first establish the following lemma.

Lemma 6.13. *There is no homomorphism from G_1 to G_2 and vice versa, where G_1 and G_2 are the left and right graphs of Figure 6.1.*

Proof. There cannot be homomorphism from G_2 to G_1 since there is a path of length 3 in G_2 but not in G_1.

Suppose for the sake of contradiction that there is a homomorphism $h : G_1 \to G_2$. Then h has to map 1 to 4 or 1 to 5 since only in 4 and 5 there start paths of length 2. In the former, 3 has to be mapped to 6 and in the latter 3 has to be mapped to 7. However, $(4, 6)$ and $(5, 7)$ are not in G_2. So such a homomorphism cannot exist. □

We are now ready for Theorem 6.12.

Proof of Theorem 6.12. First, suppose that $\Gamma = \{U\}$ where U is unary. We show that in this case $\mathrm{CQ}_\Gamma{}^{\subseteq}$ only contains two Boolean queries:

1. Q_1 : true
2. $Q_2 : (x, y) \leftarrow U(x), U(y) \subseteq (x, x) \leftarrow U(x)$.

Suppose that $e_1 \subseteq e_2 \in \mathrm{CQ}_\Gamma{}^{\subseteq}$ where e_1 and e_2 have heads $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ respectively. If $e_1 \subseteq e_2$ is not the constant true Boolean query, there exists an instance A such that $e_1(A) \not\subseteq e_2(A)$. Then, there exists i, j such that $x_i \neq x_j$ and $y_i = y_j$ since $(a, \ldots, a) \in Q(I)$ for any CQ over Γ, any instance I over Γ, and any $a \in \mathrm{adom}(I)$. Therefore, for any instance I with at least two elements in $\mathrm{adom}(I)$, $e_1 \not\subseteq e_2(I)$. Thus, $e_1 \subseteq e_2$ is equivalent to Q. We may thus conclude that $\mathrm{CQ}_\Gamma{}^{\subseteq}$ is closed under conjunction.

On the other hand, suppose that Γ contains at least two unary relations U_1 and U_2. We now show that $Q : U_1 \subseteq U_2 \wedge U_2 \subseteq U_1$ is not in $\mathrm{CQ}_\Gamma{}^{\subseteq}$. Suppose for the sake of contradiction that Q is expressed by $e_1 \subseteq e_2$ in $\mathrm{CQ}_\Gamma{}^{\subseteq}$. Let $I_1 = \{U_1(1)\}$ and $I_2 = \{U_2(2)\}$. Clearly, $Q(I_1) = Q(I_2) = \mathit{false}$, whence we have $e_1(I_1) \neq \emptyset$ and $e_1(I_2) \neq \emptyset$. Thus, there is a homomorphism from B_{e_1} into I_1 and B_{e_1} into I_2. Therefore, $B_{e_1} = \emptyset$. Then, due to safety of CQs, the head of e_1 is empty. Hence, Q is equivalent to $e_2 \neq \emptyset$, which is monotone. This, however, is a contradiction since $U_1 = U_2$ is not monotone.

Finally, suppose that Γ contains at least one nonunary relation R. Define

- Q_1 as $(x, y) \leftarrow R(x, z, \_, \ldots, \_), R(z, y, \_, \ldots, \_), R(x, y, \_, \ldots, \_)$
- Q_2 as $(x, y) \leftarrow R(x, z_1, \_, \ldots, \_), R(z_1, z_2, \_, \ldots, \_), R(z_2, y, \_, \ldots, \_),$
  $R(x, y, \_, \ldots, \_)$

We now show that $Q : Q_1 \subseteq Q_2 \wedge Q_2 \subseteq Q_1$ is not in $\mathrm{CQ}_\Gamma{}^{\subseteq}$. Let G_1 and G_2 be the left and right graphs in Figure 6.1 respectively. Clearly, we can identify B_{Q_1} with G_1 and B_{Q_2} with G_2. Suppose for the sake of contradiction that $e_1 \subseteq e_2 \in \mathrm{CQ}_\Gamma{}^{\subseteq}$ expresses Q. We first show that there is no homomorphism from G_1 into B_{e_1}. Suppose there is a homomorphism h from G_1 into B_{e_1}. Clearly, $e_1(G_2) \neq \emptyset$ since $Q(G_2) = \mathit{false}$. Hence, there is a homomorphism f from B_{e_1} into G_2. Then, $f \circ h$ is a homomorphism from G_1 to G_2, which contradicts Lemma 6.13. Analogously, we can establish that there is no homomorphism from G_2 into B_{e_1}.

Since there is no homomorphism from G_1 and G_2 into B_{e_1}, $Q_1(B_{e_1}) = \emptyset$ and $Q_2(B_{e_1}) = \emptyset$, whence we have $Q(B_{e_1}) = \mathit{true}$. Thus, also $e_1(B_{e_1}) \subseteq e_2(B_{e_1})$. Since B_{e_1} is the body of e_1, we have that $e_1 \sqsubseteq e_2$. This is a contradiction since Q is not the constant true query. $\square$

The question whether unions of conjunctive queries under the containment modality are closed under conjunction is still open.

7

Succinctness of converse elimination for graph query languages under nonemptiness

In Chapter 4, we have investigated the second theme where different query languages are compared under the same fixed base modality. In particular, we investigated this theme for navigational query languages for each of the three base modalities. For nonemptiness, the result is outlined in Theorem 4.1 for (co)projection restricted fragments. Therefore, for (co)projection restricted fragments F, where $\widetilde{F}$ contains neither intersection nor transitive closure, this theorem tells us that Boolean queries in $F^{\neq\emptyset}$ are also in $\widehat{F}^{\neq\emptyset}$, and thus effectively eliminating converse (at the expense of adding projection). We refer to this phenomenon as *converse elimination.* Furthermore, we say that a fragment F *admits converse elimination* if ${}^{-1} \in F$, $\cap \notin \widetilde{F}$ and ${}^{+} \notin F$. By Theorem 4.1, there thus exists a function that translates expressions $e \in \mathcal{N}(F)$ to equivalent expressions $e' \in \mathcal{N}(\widehat{F})$ for (co)projection restricted fragments F that admit converse elimination.

In the main result of this chapter, we prove that converse elimination always leads to at least an exponential blowup in degree. The *degree* of an expression e, denoted by *degree*(e), is the maximum depth of nested applications of composition, projection and coprojection in e. For example, the degree of $R \circ R$ is 1, while the degree of both $R \circ (R \circ R)$ and $\pi_1(R \circ R)$ is 2. Intuitively, the degree of e corresponds to the quantifier rank of the standard translation of e into FO^3. Formally, the succinctness result for converse elimination can then be summarized as follows.

Theorem 7.1. *Let F be a (co)projection restricted fragment that admits converse elimination. Furthermore, let h be a function that maps expressions in $e \in \mathcal{N}(F)$ to equivalent expressions $e' \in \mathcal{N}(\widehat{F})$. If $f : \mathbb{N} \to \mathbb{N}$ is a function such that for every $e \in \mathcal{N}(F)$ we have $\mathit{degree}(h(e)) \leq f(\mathit{degree}(e))$, then $f \neq o(2^n)$.*

To prove Theorem 7.1 we will employ invariance results under the notion of bisimulation below. In essence, this notion is based on the notion of bisimulation known from arrow logics [35]. This notion has been adapted to our setting [20]. We recall the basic definitions.

Let $\mathbf{G} = (G, a, b)$ denote a *marked graph*, i.e., a graph G with $a, b \in \mathrm{adom}(G)$. For a set of features F, $\mathcal{N}(F)_k$ denotes the set of expressions in $\mathcal{N}(F)$ of degree at most k.

In what follows, we are only concerned with bisimulation results regarding $\mathcal{N}(-, \mathsf{di})$. The following is an appropriate notion of bisimulation for this language.

Definition 7.2 (Bisimilarity)**.** Let k be a natural number and let $\mathbf{G}_1 = (G_1, a_1, b_1)$ and $\mathbf{G}_2 = (G_2, a_2, b_2)$ be marked graphs. We say that $\mathbf{G}_1$ is bisimilar to $\mathbf{G}_2$ up to depth k, denoted $\mathbf{G}_1 \simeq_k \mathbf{G}_2$, if the following conditions are satisfied:

Atoms $a_1 = b_1$ if and only if $a_2 = b_2$; and $(a_1, b_1) \in G_1(R)$ if and only if $(a_2, b_2) \in G_2(R)$, for every $R \in \Gamma$;

Forth if $k > 0$, then, for every c_1 in $\mathrm{adom}(G_1)$, there exists some c_2 in $\mathrm{adom}(G_2)$ such that

$$(G_1, a_1, c_1) \simeq_{k-1} (G_2, a_2, c_2) \quad \text{and} \quad (G_1, c_1, b_1) \simeq_{k-1} (G_2, c_2, b_2);$$

Back if $k > 0$, then, for every c_2 in $\mathrm{adom}(G_2)$, there exists some c_1 in $\mathrm{adom}(G_1)$ such that

$$(G_1, a_1, c_1) \simeq_{k-1} (G_2, a_2, c_2) \quad \text{and} \quad (G_1, c_1, b_1) \simeq_{k-1} (G_2, c_2, b_2).$$

We also say that there is a bisimulation of depth k between $\mathbf{G}_1$ and $\mathbf{G}_2$ if $\mathbf{G}_1 \simeq_k \mathbf{G}_2$.

Recall the following adequacy theorem for bisimulations.

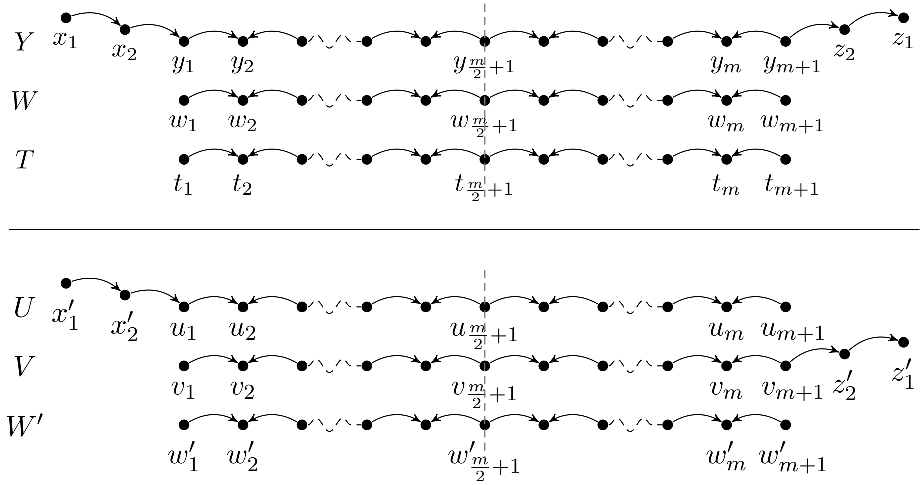


Figure 7.1: Graphs G_1^m (top) and G_2^m (bottom) used to establish the exponential blowup during converse elimination in Theorem 7.1.

Theorem 7.3 ([20])**.** *Let k be a natural number; and let $\mathbf{G}_1 = (G_1, a_1, b_1)$ and $\mathbf{G}_2 = (G_2, a_2, b_2)$ be marked graphs. We have, $\mathbf{G}_1 \simeq_k \mathbf{G}_2$ iff $(a_1, b_1) \in e(G_1) \Leftrightarrow (a_2, b_2) \in e(G_2)$ for every $e \in \mathcal{N}(-, \mathsf{di})_k$.*

Intuitively, this proposition tells us that marked graphs are indistinguishable by k-degree path queries iff these graphs are bisimilar up to depth k.

To show Theorem 7, we will establish the following bisimulations between the classes of graphs G_1^m and G_2^m displayed in Figure 7.1:

Theorem 7.4. *For every pair $(a_1, b_1) \in \mathrm{adom}(G_1^m)^2$ there exists another pair $(a_2, b_2) \in \mathrm{adom}(G_2^m)^2$ such that $(G_1^m, a_1, b_1) \simeq_{m/2-1} (G_2^m, a_2, b_2)$.*

We will prove this theorem in Section 7.1.

Armed with the bisimulations in Theorem 7.4, we are finally ready to prove Theorem 7.1.

Proof of Theorem 7.1. Let a function $f : \mathbb{N} \to \mathbb{N}$ be given as in the statement of Theorem 7.1. Now suppose for the sake of contradiction that $f(n) = o(2^n)$. Let Q be the path query $R^2 \circ (R \circ R^{-1})^+ \circ R^2$. Define $\mathcal{G}_n$ as the class of graphs with an active domain of size at most n and define Q_n as the expression Q where every subexpression of the form f^+ in Q is replaced with $\cup_{i=1}^{n} f^i$. Note that expressions of the form f^+ are equivalent to the expression $\cup_{i=1}^{n} f^i$ when we only consider graphs in $\mathcal{G}_n$. Therefore Q_n is equivalent to Q on $\mathcal{G}_n$. Notice that if we carefully arrange the compositions in f^i, we obtain that $\mathit{degree}(\cup_{i=1}^{n} f^i) = \mathit{degree}(f) + \lceil \log_2 n \rceil$. Hence we can conclude that $\mathit{degree}(Q_n) = \lceil \log_2 n \rceil + 3$.

We now show that $f(\mathit{degree}(Q_n)) = o(n)$. Since $f(n) = o(2^n)$, we have by definition that $\lim_{n\to\infty} f(n)/2^n = 0$. Notice that $\mathit{degree}(Q_n)$ goes to infinity as n goes to infinity. Therefore, we have that

$$\lim_{n\to\infty} f(\mathit{degree}(Q_n))/2^{\mathit{degree}(Q_n)} = 0$$

as well. We now show that this last limit implies that $f(\mathit{degree}(Q_n)) = o(n)$:

$$\begin{aligned} 0 = \lim_{n\to\infty} \frac{f(\mathit{degree}(Q_n))}{2^{\mathit{degree}(Q_n)}} &= \lim_{n\to\infty} \frac{f(\lceil \log_2 n \rceil + 3)}{2^{\lceil \log_2 n \rceil + 3}} \\ &\geq \lim_{n\to\infty} \frac{f(\lceil \log_2 n \rceil + 3)}{16n} \geq 0 \end{aligned}$$

Notice that Q_n is an expression in $\mathcal{N}(^{-1})$, whence by assumption $h(Q_n)$ is an expression in $\mathcal{N}(\pi)$. We now show that there exists a natural number k such that for every $m \geq k$, $h(Q_{3m+7})$ is an expression in $\mathcal{N}(\pi)_{m/2-1}$.

Since $f(\mathit{degree}(Q_n)) = o(n)$, also $f(\mathit{degree}(Q_{3m+7})) = o(3m+7)$. Furthermore, we may conclude that $f(\mathit{degree}(Q_{3m+7})) = o(m/2-1)$ since $o(3m+7) = o(m/2-1)$. Thus by definition,

$$\lim_{m\to\infty} f(\mathit{degree}(Q_n))/(m/2-1) = 0.$$

Hence

$$\forall \varepsilon > 0, \exists k \in \mathbb{N}, \forall m \in \mathbb{N} : m \geq k \Rightarrow \frac{f(\mathit{degree}(Q_{3m+7}))}{m/2-1} < \varepsilon.$$

Hence if we set $\varepsilon = 1$, we can find a k such that for every $m \geq k$ we have $f(\mathit{degree}(Q_{3m+7}))/(m/2-1) < 1$, or equivalently $f(\mathit{degree}(Q_{3m+7})) < m/2-1$. This implies that $\mathit{degree}(h(Q_{3m+7})) < m/2-1$ for any $m \geq k$ since it is given that $\mathit{degree}(h(Q_n)) \leq f(\mathit{degree}(Q_n))$ for any n. Thus we may conclude that $h(Q_{3m+7})$ is an expression in $\mathcal{N}(\pi)_{m/2-1}$ for any $m \geq k$.

Now let m be a multiple of four, greater then k, and let G_1^m be the top and G_2^m be the bottom graph in Figure 7.1. Since $|\mathrm{adom}(G_1^m)| = |\mathrm{adom}(G_2^m)| = 3m+7$, we know that Q_{3m+7} agrees with Q on G_1^m and G_2^m.

Thus $Q_{3m+7}(G_1^m) \neq \emptyset$ since $Q(G_1^m)$ is nonempty. Furthermore, because $h(Q_{3m+7})$ is equivalent to Q_{3m+7} at the level of Boolean queries, it must be that $h(Q_{3m+7})(G_1^m) \neq \emptyset$. Thus let $(a_1, b_1) \in h(Q_{3m+7})(G_1^m)$. By Theorem 7.4 there exists (a_2, b_2) such that $(G_1^m, a_1, b_1) \simeq_{m/2-1} (G_2^m, a_2, b_2)$. Then by Theorem 7.3 also $(a_2, b_2) \in h(Q_{3m+7})(G_2^m)$. However, since $Q(G_2^m)$ is clearly empty, $Q_{3m+7}(G_2^m)$ as well as $h(Q_{3m+7})(G_2^m)$ should be empty. We have thus obtained a contradiction. Thus we may conclude that $f \neq o(2^n)$. □

7.1 A bisimulation result

In this section, we prove Theorem 7.4. For the remainder of this section let $m > 4$ be an integer multiple of four, let G_1^m be the graph at the top and G_2^m be the graph at the bottom in Figure 7.1. It is important to note that these graphs have the displayed form only when m is a multiple of four.

Before we move to the proof of Theorem 7.4, we introduce some terminology. We say that a pair $(x, y) \in \text{adom}(G_1^m) \times \text{adom}(G_2^m)$ is *valid* if the following conditions hold:

- if $x \in \{y_i, w_i, t_i\}$ then $y \in \{u_i, v_i, w_i'\}$;
- if $x = x_1$ then $y = x_1'$;
- if $x = x_2$ then $y = x_2'$;
- if $x = z_1$ then $y = z_1'$;
- if $x = z_2$ then $y = z_2'$.

Intuitively, the pair (x, y) is valid if x and y are displayed in the same column in Figure 7.1, so formally, instead of saying that (x, y) is valid, we also say that x and y are *in the same column*. Moreover, we extend this terminology for nodes x and y belonging to the same graph, with the obvious meaning.

Definition 7.5. A 4-tuple $(a_1, b_1, a_2, b_2) \in \text{adom}(G_1^m)^2 \times \text{adom}(G_2^m)^2$ is *valid* if the following conditions hold:

(a) (a_1, a_2) and (b_1, b_2) are valid;

(b) $(a_1, b_1) \in G_1^m$ if and only if $(a_2, b_2) \in G_2^m$; and $a_1 = b_1$ if and only if $a_2 = b_2$. Note that this is the Atoms condition for bisimilarity;

(c) if $a_1 = x_2$, $b_1 = y_2$ and $a_2 = x_2'$, then $b_2 = u_2$;

(d) if $a_1 = x_2$, $a_2 = x_2'$ and $b_2 = u_2$, then $b_1 = y_2$.

Intuitively, a valid quadruple is a potential starting point for a bisimulation between G_1^m and G_2^m.

For any node $x \in \mathrm{adom}(G_1^m)$ we introduce the following terminology.

- If x equals x_1 or x_2, or y_i, w_i or t_i with $0 \leq i \leq m/2 + 1$, we call x a *left* element.
- If x is not a left element, i.e., x equals z_1 or z_2, or y_i, w_i or t_i with $m/2 + 1 < i \leq m + 1$, we call x a *right* element.
- If x equals y_i for any i, we call x a Y element. Analogously, if x equals w_i, t_i, x_i, or z_i for any i, we call x a W, T, X or Z element, respectively.

Clearly we can combine these adjectives and thus speak about a Y left element, for example.

For any node $y \in \mathrm{adom}(G_2^m)$ we can use the analogous terminology of left, right, U, V, W', X' and Z' elements with analogous meaning.

Let us now define a function f mapping valid pairs to natural numbers:

$$f(d,e) = \begin{cases} m/2 & \text{if } d = y_i \text{ left and } e = u_i \\ i-1 & \text{if } d = y_i \text{ left and } (e = v_i \text{ or } e = w_i') \\ m+1-i & \text{if } d = y_i \text{ right and } (e = u_i \text{ or } e = w_i') \\ m/2 & \text{if } d = y_i \text{ right and } e = v_i \\ i-1 & \text{if } (d = w_i \text{ or } d = t_i) \text{ left and } e = u_i \\ m/2 & \text{if } (d = w_i \text{ or } d = t_i) \text{ left and } (e = v_i \text{ or } e = w_i') \\ m/2 & \text{if } (d = w_i \text{ or } d = t_i) \text{ right and } (e = u_i \text{ or } e = w_i') \\ m+1-i & \text{if } (d = w_i \text{ or } d = t_i) \text{ right and } e = v_i \\ i-1 & \text{if } d = t_i \text{ left and } e = u_i \\ m/2 & \text{if } d = t_i \text{ left and } (e = v_i \text{ or } e = w_i') \\ m/2 & \text{if } d = t_i \text{ right and } (e = u_i \text{ or } e = w_i') \\ m+1-i & \text{if } d = t_i \text{ right and } e = v_i \\ m/2 & \text{if } d = x_i \text{ and } e = x_i' \\ m/2 & \text{if } d = z_i \text{ and } e = z_i' \end{cases}$$

Intuitively, $f(d,e) = m/2$ only when d and e are in the middle column or d and e are on the side of chains with similar endings in Figure 7.1, i.e., d is Y left iff e is U left, and d is Y right iff e is V right. In all other cases $f(d,e) < m/2$. For example, let us examine the values for the valid pairs (x,y), (w,z), (a_1,a_2) and (b_1,b_2) in the graphs G_1^8 and G_2^8 displayed in Figure 7.2. In this case $m = 8$, thus $f(x,y) = 2$, $f(w,z) = m+1-7 = 2$, $f(a_1,a_2) = m/2 = 4$ and $f(b_1,b_2) = 3$.

Our key idea to establish Theorem 7.4 is to show that $\min(f(a_1,a_2), f(b_1,b_2))$ is a lower bound on the bisimulation depth between (G_1^m, a_1, b_1) and (G_2^m, a_2, b_2); this will be our key Lemma 7.22. Before proving this in detail, we intuitively describe the overall strategy.

To establish a bisimulation of depth d between (G_1^m, a_1, b_1) and (G_2^m, a_2, b_2), we need that (a_1, b_1, a_2, b_2) satisfies the Atoms condition, and we need that the Forth and Back conditions hold. A first characteristic of our strategy is that we take care to maintain not just the Atoms condition, but the stronger property of *validity* from Definition 7.5. Viewing a bisimulation argument as a game, the validity property provides tighter control on the possible game situations that can arise.

For the Forth condition we need to find a node $c_2 \in \mathrm{adom}(G_2^m)$ for every node $c_1 \in \mathrm{adom}(G_1^m)$ such that there is a bisimulation of depth $d-1$

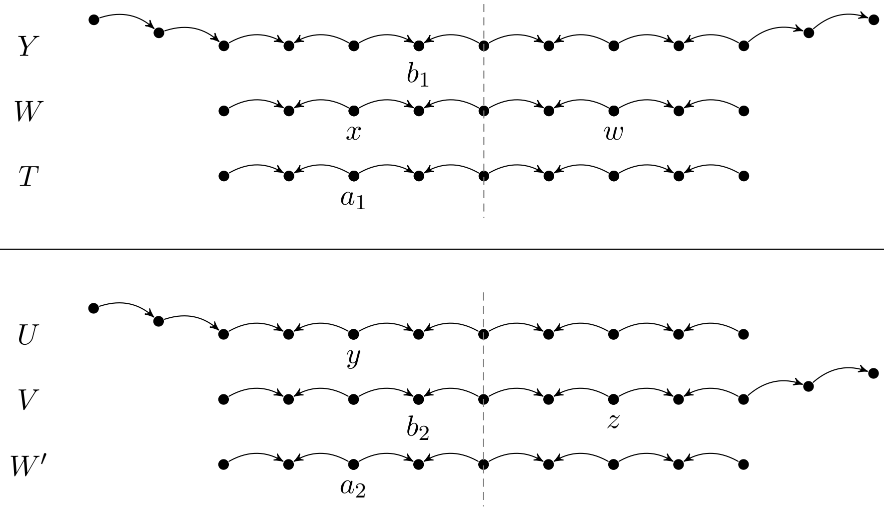


Figure 7.2: The graphs G_1^8 at the top, and G_2^8 at the bottom. Notice here that (x, y), (w, z), (a_1, a_2) and (b_1, b_2) are valid pairs. Since $m = 8$ in this case, we have that $f(x, y) = 2$, $f(w, z) = m+1-7 = 2$, $f(a_1, b_2) = m/2 = 4$ and $f(b_1, b_2) = 3$

between (G_1^m, a_1, c_1) and (G_2^m, c_1, a_2), and (G_1^m, c_1, b_1) and (G_2^m, c_2, b_2). For the Back condition we need to do the same thing except that the roles of c_1 and c_2 are switched.

Actually, instead of directly working with bisimulations with a certain depth, we will show that we can pick a $c_2 \in \text{adom}(G_2^m)$ for every $c_1 \in \text{adom}(G_1^m)$ (and vice versa) that ensures validity of (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) while providing a lower bound on $f(c_1, c_2)$. This will provide enough information to prove Lemma 7.22 by induction.

So let us now have an intuitive look at the strategy used in the technical lemmas to pick such a $c_2 \in \text{adom}(G_2^m)$ for every $c_1 \in \text{adom}(G_1^m)$. First, remember that we only work with valid quadruples, so c_2 has to be in the same column as c_1. This leaves us with three candidate nodes (or just one in case c_1 is an X or Z element). We pick one of these nodes according to the following strategy:

1. First, we check whether $a_1 = c_1$, $c_1 = b_1$, (a_1, c_1) is an edge, or (c_1, b_1) is an edge. If this is indeed the case, we say that c_1 is *related* to a_1 or b_1. Here we pick c_2 so that it is related in the same way as c_1 is related to a_1 or b_1. The relation of c_1 and to a_1 of b_1 ensures that

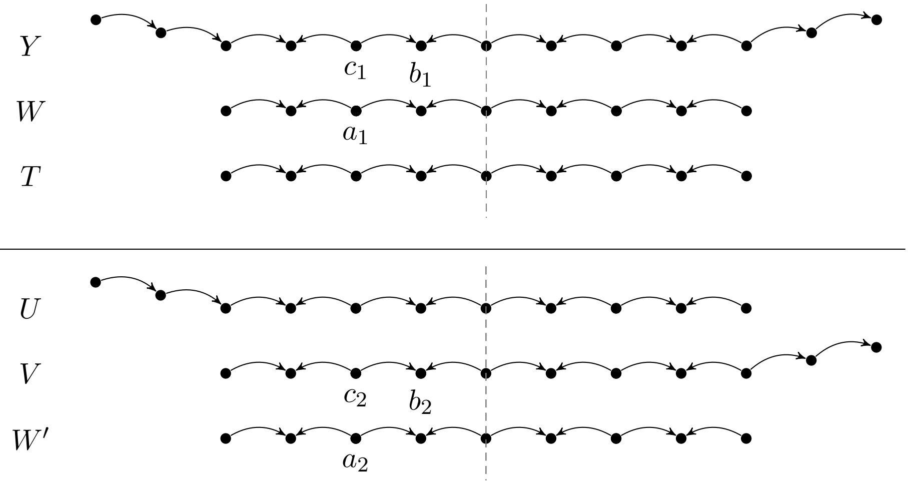


Figure 7.3: An example of the first step in our strategy on the graphs G_1^m and G_2^m with $m = 8$. Here c_1 is related to b_1, i.e, (c_1, b_1) is an edge. The node c_2 is thus picked such that (c_2, b_2) is an edge. The validity of (a_1, b_1, a_2, b_2) then ensures that a_2 is not related to c_2. Notice also that $f(c_1, c_2) = f(b_1, b_2) - 1$ by definition.

c_1 is in the column next to, or in the same column as a_1 or b_1. This implies that $f(c_1, c_2)$ is at most one lower than $f(a_1, a_2)$ or $f(b_1, b_2)$.

For example, if (c_1, b_1) is an edge, we pick c_2 in the same column as c_1 such that (c_2, b_2) is an edge (see Figure 7.3).

2. If c_1 is not related to a_1 or b_1, i.e., if $a_1 \neq c_1$, $c_1 \neq b_1$, (a_1, c_1) is not an edge, and (c_1, b_1) is not an edge, we check whether it is possible to pick c_2 such that $f(c_1, c_2) = m/2$ without breaking validity. Since $m/2$ is the maximum output of f, we can be sure that $f(c_1, c_2)$ is sufficiently large. For an example of this scenario see Figure 7.4.

3. If we cannot pick c_2 such that $f(c_1, c_2) = m/2$ without breaking validity, we just pick c_2 such that validity is ensured. It turns out that even then $f(c_1, c_2)$ is sufficiently large, i.e., at most one lower than $f(a_1, a_2)$ or $f(b_1, b_2)$. For an example of this scenario see Figure 7.5.

The strategy by itself may seem quite arbitrary. Why do we not provide a single c_2 for each c_1 without the trial and error in the second step of the strategy? The reason why we introduced the trial and error step, is because

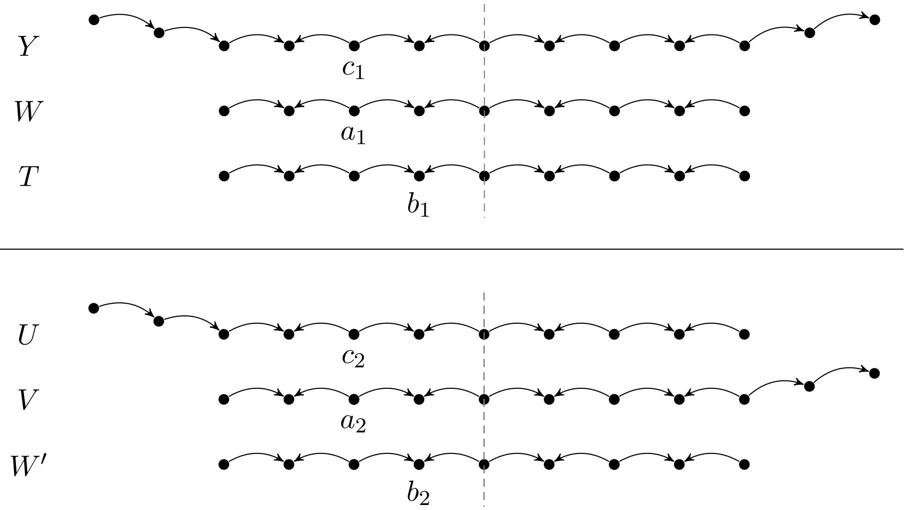


Figure 7.4: An example of the second step in our strategy on G_1^m and G_2^m with $m = 8$. Hence c_1 is not related a_1 and b_1 ($a_1 \neq c_1$, $c_1 \neq b_1$, (a_1, c_1) is not an edge, and (c_1, b_1) is not an edge), and it is possible to pick c_2 such that $f(c_1, c_2) = m/2$ without violating validity. Notice that c_2 has to be picked on U in this example since only then we have $f(c_1, c_2) = m/2$ by definition.

a failure in that step tells us something about the location of a_1 and a_2, and b_1 and b_2. Indeed, if the validity of (a_1, c_1, a_2, c_2) is broken, for example, we know that $a_2 = c_2$, or that (a_2, c_2) is an edge, which implies that a_1 and a_2 are in the same column as, or in the column next to c_1 and c_2. Using these facts, we will be able to determine the values of $f(a_1, a_2)$ and $f(b_1, b_2)$, which will appear to be sufficiently low by itself so that we can pick c_2 without having to worry about $f(c_1, c_2)$.

We will now start the technical proof with several lemmas. Lemmas 7.6 and 7.7 take care of first step of the strategy outlined above. Lemmas 7.8 and 7.17 take care of the second and third step. To establish these final two steps, we use several sublemmas for clarity (Lemmas 7.9 to 7.16).

Lemma 7.6. *Suppose that* (a_1, b_1, a_2, b_2) *is valid,* $f(a_1, a_2) > 0$*,* $f(b_1, b_2) > 1$ *and* $c_1 \in \operatorname{adom}(G_1^m)$ *such that* $a_1 = c_1$*,* $b_1 = c_1$*,* (a_1, c_1) *is an edge, or* (c_1, b_1) *is an edge. Then there exists* $c_2 \in \operatorname{adom}(G_2^m)$ *such that both* (a_1, c_1, a_2, c_2) *and* (c_1, b_1, c_2, b_2) *are valid, and* $f(c_1, c_2) \geq \min(f(a_1, a_2),$ $f(b_1, b_2)) - 1$.

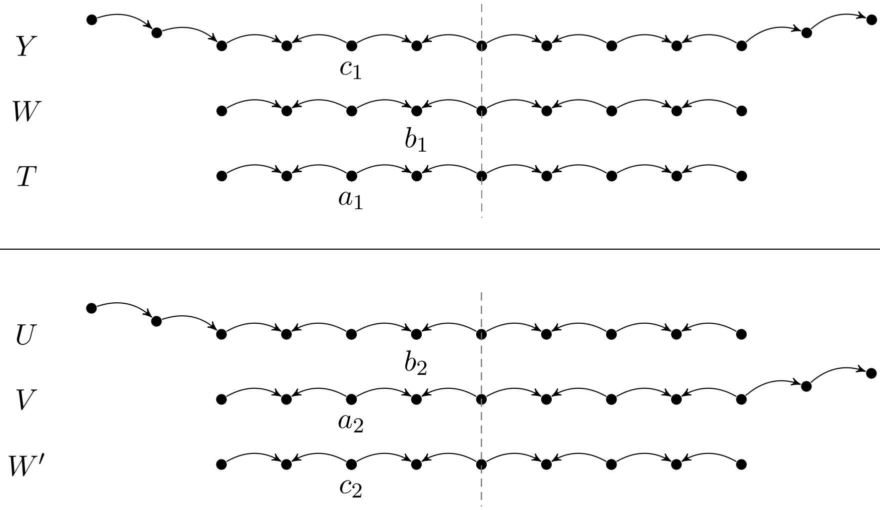


Figure 7.5: An example of the third step in our strategy on G_1^m and G_2^m with $m = 8$. Hence c_1 is not related to a_1 and b_2 ($a_1 \neq c_1$, $c_1 \neq b_1$, (a_1, c_1) is not an edge, and (c_1, b_1) is not an edge), and it is not possible to pick c_2 such that $f(c_1, c_2) = m/2$. Indeed, here $f(c_1, c_2) = m/2$ only if c_2 is on U. Thus c_2 has to be picked on the chain not containing a_2 and b_2. Clearly $f(c_1, c_2) = f(b_1, b_2) - 1$ by definition.

Proof. First suppose that $a_1 = c_1$. Then we pick $c_2 = a_2$. Clearly, (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid by construction. Furthermore, $f(c_1, c_2) = f(a_1, a_2) \geq \min((fa_1, a_2)), f(b_1, b_2)) - 1$. The case where $c_1 = b_1$ is analogous.

Now suppose that (a_1, c_1) is an edge. Then we pick c_2 in the same column as c_1 (thus (c_1, c_2) is valid) such that (a_2, c_2) is an edge. This is clearly possible if $a_1 \neq y_{m+1}$, since in that case any node in the same column of a_2 has a forward or backward outgoing edge in the same way as a_1. On the other hand, if $a_1 = y_{m+1}$, then $a_2 = v_{m+1}$ since $f(a_1, a_2) > 0$. Again y_{m+1} in G_1^m and v_{m+1} in G_2^m have similar outgoing edges. Clearly (a_1, c_1, a_2, c_2) is valid by construction. The validity of (c_1, b_1, c_2, b_2) is not so evident. Note, however, that $b_1 = y_2$ iff $b_2 = u_2$ since $f(b_1, b_2) > 1$. Thus conditions (c) and (d) for the validity of (c_1, b_1, c_2, b_2) are trivially satisfied. Thus we only have to show that (c_1, b_1, c_2, b_2) satisfies the Atoms condition.

$$\begin{aligned} b_1 = c_1 &\iff (a_1, b_1) \text{ is an edge} && \text{(since } (a_1, c_1) \text{ is an edge)} \\ &\iff (a_2, b_2) \text{ is an edge} && \text{(since } (a_1, b_1, a_2, b_2) \text{ is valid)} \\ &\iff b_2 = c_2 && \text{(since } (c_1, c_2) \text{ is valid and } (a_2, c_2) \text{ is an edge)} \end{aligned}$$

Suppose (c_1, b_1) is also an edge, then $c_1 \in \{x_2, y_1, z_2\}$ because these are the only nodes with incoming as well as outgoing edges. If $c_1 = x_2$, then $c_2 = x'_2$, and $b_1 = y_1$, whence we have $b_2 = u_1$ since $f(b_1, b_2) > 0$. On the other hand, if $c_1 = y_1$, then $a_1 = x_2$, $c_2 = u_1$, $b_1 = y_2$, and $a_2 = x'_2$. Now by conditions (c) and (d) from the validity of (a_1, b_1, a_2, b_2) we have that $b_2 = u_2$. Finally, if $c_1 = z_2$, then $c_2 = z'_2$ and $b_1 = z_1$, whence we have $b_2 = z'_1$ since (a_1, b_1, a_2, b_2) is valid. In either case, (c_2, b_2) is an edge as desired.

On the other hand suppose that (c_2, b_2) is an edge, then $c_2 \in \{x'_2, u_1, z'_2\}$ because these are the only nodes with incoming as well as outgoing edges. If $c_2 = x'_2$, then $c_1 = x_2$, and $b_2 = u_1$, whence we have $b_1 = y_1$ since $f(b_1, b_2) > 0$. On the other hand, if $c_2 = u_1$, then $c_1 = y_1$, $a_2 = x'_2$ and $b_2 = u_2$. Now by conditions (c) and (d) from the validity of (a_1, b_1, a_2, b_2) we have that $b_2 = y_2$. Finally, if $c_2 = z'_2$, then $c_1 = z_2$ and $b_2 = z'_1$, whence we have $b_2 = z_1$ since (a_1, b_1, a_2, b_2) is valid. In either case, (c_1, b_1) is an edge as desired.

So it remains to be shown that $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$. Since (a_1, c_1) is an edge, it is clear that c_1 is in the column to the left or right of a_1. Thus if $f(a_1, a_2) < m/2$, we must have that $f(c_1, c_2) \geq f(a_1, a_2) - 1 \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$. On the other hand, suppose that $f(a_1, a_2) = m/2$. Let us list the possibilities for $f(a_1, a_2)$ to equal $m/2$: the column of a_1 is $m/2+1$; a_1 is Y left and a_2 is U left; a_1 is W left and a_2 is W' left; a_1 is W left and a_2 is V left; a_1 is T left and a_2 is W' left; a_1 is T left and a_2 is V left; a_1 is Y right and a_2 is V right; a_1 is W right and a_2 is W' right; a_1 is W right and a_2 is U right; a_1 is T right and a_2 is W' right; or a_1 is T right and a_2 is U right. Therefore, unless $a_1 \in \{y_{\frac{m}{2}+1}, t_{\frac{m}{2}+1}, w_{\frac{m}{2}+1}\}$, c_1 is on the same side of the chain as a_1, and c_2 is on the same side (left or right) of the chain as a_2 since (a_1, c_1) and (a_2, c_2) are edges. The definition of f implies that $f(c_1, c_2) = m/2$. If $a_1 \in \{y_{\frac{m}{2}+1}, t_{\frac{m}{2}+1}, w_{\frac{m}{2}+1}\}$, then the column of c_1 and c_2 is $m/2$ or $m/2+2$ since (a_1, c_1) and (a_2, c_2) are edges. Therefore $f(c_1, c_2) \geq m + 1 - (m/2 + 2) = m/2 - 1$ as desired.

The case where (c_1, b_1) is an edge is analogous to the case where (a_1, c_1) is an edge. □

Notice that three consecutive columns in G_1^m are isomorphic to the three corresponding columns in G_2^m displayed in Figure 7.1. Hence we can exchange the roles of c_1 and c_2 in the proof of the previous lemma without violating the Atoms condition since the Atoms condition can only fail if there is a problem on the columns directly surrounding c_1 and c_2. Furthermore, notice that the value of $f(a_1, a_2)$ only depends on how a_1 and a_2 relate to one another on one side of the graph (the left or right-hand side). Hence, the condition on $f(c_1, c_2)$ also remains intact, since G_1^m and G_2^m look completely the same on the left-hand (right-hand) side. Thus the proof of the following lemma is analogous to the proof of Lemma 7.6.

Lemma 7.7. *Suppose that* (a_1, b_1, a_2, b_2) *is valid,* $f(a_1, a_2) > 0$, $f(b_1, b_2) > 1$ *and* $c_2 \in \mathrm{adom}(G_2^m)$ *such that* $a_2 = c_2$, $b_2 = c_2$, (a_2, c_2) *is an edge, or* (c_2, b_2) *is an edge. Then there exists* $c_1 \in \mathrm{adom}(G_1^m)$ *such that both* (a_1, c_1, a_2, c_2) *and* (c_1, b_1, c_2, b_2) *are valid, and* $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.

Let us now take care of steps two and three in the intuitive strategy outlined before Lemma 7.6, i.e., when c_1 is not related to a_1 or b_1.

Lemma 7.8. *Suppose that* (a_1, b_1, a_2, b_2) *is valid,* $f(a_1, a_2) > 0$, $f(b_1, b_2) > 1$ *and* $c_1 \in \mathrm{adom}(G_1^m)$ *such that* $a_1 \neq c_1$, $c_1 \neq b_1$, (a_1, c_1) *and* (c_1, b_1) *are not edges. Then there exists* $c_2 \in \mathrm{adom}(G_2^m)$ *such that both* (a_1, c_1, a_2, c_2) *and* (c_1, b_1, c_2, b_2) *are valid, and* $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.

Proof. The goal is to follow the following strategy, unless it breaks the Atoms condition for (a_1, c_1, a_2, c_2) or (c_1, b_1, a_2, c_2). Henceforth we refer to this strategy as the *Greedy Strategy.*

$$\begin{aligned}
c_1 = z_i \wedge 1 \leq i \leq 2 &\implies c_2 = z_i' \\
c_1 = x_i \wedge 1 \leq i \leq 2 &\implies c_2 = x_i' \\
c_1 = y_i \wedge 0 \leq i \leq m/2 + 1 &\implies c_2 = u_i \\
c_1 = y_i \wedge m/2 + 1 < i \leq m + 1 &\implies c_2 = v_i \\
c_1 = w_i &\implies c_2 = w_i' \\
c_1 = t_i \wedge 0 \leq i \leq m/2 + 1 &\implies c_2 = v_i \\
c_1 = t_i \wedge m/2 + 1 < i \leq m + 1 &\implies c_2 = u_i.
\end{aligned}$$

The reason why we use this strategy is because in this case $f(c_1, c_2) = m/2$, in which case it is trivial that $f(c_1, c_2) \geq \min\{f(a_1, a_2), f(b_1, b_2)\} - 1$.

First, we establish that the Atoms conditions cannot be broken in the following situations: $c_1 = y_1$; $c_1 = y_{m+1}$; $c_1 = z_i$ with $i = 1, 2$; $c_1 = x_i$ with $i = 1, 2$; or $(a_1, c_1) = (x_2, y_2)$. To prove this, suppose first that $c_1 = y_1$; then by the strategy outlined above $c_2 = u_1$.

- If (a_2, c_2) is an edge then $a_2 = x'_2$, whence we have $a_1 = x_2$ since (a_1, b_1, a_2, b_2) is valid. Thus (a_1, c_1) is also an edge, which is a contradiction.

- If $a_2 = c_2$ then $a_2 = u_1$, whence we have $a_1 = y_1$ since $f(a_1, a_2) > 0$. Thus $a_1 = c_1$ which is a contradiction.

⋆ If (c_2, b_2) is an edge then $b_2 = u_2$, whence we have $b_1 = y_2$ since $f(b_1, b_2) > 1$. Thus (c_1, b_1) is also an edge, which is a contradiction. (This item is specially marked with ⋆ for later reference in the proof of Lemma 7.20.)

- If $b_2 = c_2$ then $b_2 = u_1$, whence we have $b_1 = y_1$ since $f(b_1, b_2) > 0$. Thus $c_1 = b_1$ which is a contradiction.

So, when $c_1 = y_1$ the chosen c_2 does not break the Atoms conditions.

Next suppose that $c_1 = y_{m+1}$; then by the Greedy Strategy $c_2 = v_{m+1}$.

- (a_2, c_2) cannot be an edge since v_{m+1} has no incoming edges.

- If $a_2 = c_2$ then $a_2 = v_{m+1}$, whence we have $a_1 = y_{m+1}$ since $f(a_1, a_2) > 0$. Thus $a_1 = c_1$ which is a contradiction.

- If (c_2, b_2) is an edge then $b_2 = z'_2$, whence we have $b_1 = z_2$ since (a_1, b_1, a_2, b_2) is valid. Thus (c_1, b_1) is also an edge, which is a contradiction.

- If $c_2 = b_2$ then $b_2 = v_{m+1}$. Hence, we have $b_1 = y_{m+1}$ since $f(b_1, b_2) > 0$. Thus $b_1 = c_1$ which contradicts the given.

Next suppose that $c_1 = x_2$; then by the Greedy Strategy $c_2 = x'_2$.

- If (a_2, c_2) is an edge, then $a_2 = x'_1$, whence we have $a_1 = x_1$ since (a_1, b_1, a_2, b_2). Thus (a_1, c_1) is also an edge, which is a contradiction.

- If $a_2 = c_2$ then $a_2 = x'_2$, whence we have $a_1 = x_2$. Thus $a_1 = c_1$ which is a contradiction.

- If (c_2, b_2) is an edge, then $b_2 = u_1$. whence we have $b_1 = y_1$, since $f(b_1, b_2) > 0$. Thus (c_1, b_1) is an edge which is a contradiction.

- If $b_2 = c_2$, then $b_2 = x'_2$, whence we have $b_1 = x_2$ since (a_1, b_1, a_2, b_2) is valid. Thus $b_1 = c_1$ which contradicts the given.

The situations where $c_1 = x_1$ or $c_1 = z_i$ with $i = 1, 2$ are similar to the previous case.

Finally, suppose that $(a_1, c_1) = (x_2, y_2)$; then by the Greedy Strategy $(a_2, c_2) = (x'_2, u_2)$. Now, for the Atoms condition to be broken, we must have that $c_2 = b_2$ since u_2 only has outgoing edges. Thus $(a_1, b_1, a_2, b_2) = (x_2, b_1, x'_2, u_2)$, whence we have $b_1 = y_2$ by condition (d) for the validity of (a_1, b_1, a_2, b_2). But then $c_1 = b_1$ which contradicts the given.

At this point, we may assume that the Atoms condition is broken if c_2 is picked according to the Greedy Strategy. By the arguments before, then, c_1 is not y_1, y_{m+1} or z_i, x_i for $i = 1, 2$, and $(a_1, c_1) \neq (x_2, y_2)$.

Furthermore, we do not have to consider cases where c_1 is in the middle column, or the two columns directly adjacent to it, i.e., the column directly to the left and right of the middle one. Indeed, since there are three chains in G_2^m, we can always pick another node c_2^{new} on the chain that does not contain a_2 and b_2. Thus $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ are certainly valid. Since c_1 and c_2^{new} is located on either of the three middle columns, we have that $f(c_1, c_2^{new}) \geq m/2 - 1 \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$ since $f(x, y)$ is at most $m/2$ for any pair of nodes $(x, y) \in \mathrm{adom}(G_1^m) \times \mathrm{adom}(G_2^m)$.

From here we will write c_2^{old} for the c_2 chosen by the Greedy Strategy.

We will split the proof into several sublemmas (Lemmas 7.9 to 7.16). First, in Lemmas 7.9 to 7.14 we show, for each case where the Atoms condition is broken, that we can pick a $c_2^{new} \in \mathrm{adom}(G_2^m)$ such that conditions (a) and (b) for the validity of both $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ are satisfied, and $f(c_1, c_2^{new}) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$. Then, in Lemmas 7.15 and 7.16 we show that $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ also satisfy conditions (c) and (d) for validity.

Lemma 7.9. *If $a_2 = c_2^{old}$ or (a_2, c_2^{old}) is an edge, and c_1 is on Y then there exists $c_2^{new} \in \mathrm{adom}(G_2^m)$ such that $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ both satisfy conditions (a) and (b) for validity, and $f(c_1, c_2^{new}) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.*

Proof. If c_1 is Y left (respectively Y right), c_2^{old} is U left (respectively V right). Since c_1 is not in the middle three columns, $c_1 \notin \{x_1, x_2, y_1\}$, and $a_2 = c_2^{old}$ or (a_2, c_2^{old}) is an edge, we have that a_2 is also U left (respectively

V right), whence we have $f(a_1, a_2) < m/2$ by definition. We now pick c_2^{new} on the chain that does not contain a_2 or b_2, in the same column as c_1, whence $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ both satisfy conditions (a) and (b) for validity. This is indeed possible since there are three chains. Thus we may conclude that c_2^{new} is not U left (respectively V right), and hence $f(c_1, c_2) < m/2$. Therefore, if $a_2 = c_2^{old}$, clearly $f(c_1, c_2^{new}) = f(a_1, a_2) < m/2$ by definition, since then c_1 is in the same column as a_1 and a_2. On the other hand, if (a_2, c_2^{old}) is an edge, then $f(c_1, c_2^{new}) \geq f(a_1, a_2) - 1$ by definition, since then c_1 is in one of the columns next to a_1 and a_2. Thus we may conclude that $f(c_1, c_2^{new}) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$. □

The proof of the following lemma is similar to the proof of Lemma 7.9 where the roles of a_1 and a_2 are replaced by b_1 and b_2, and (a_2, c_2) being an edge is replaced by (c_2, b_2) being an edge.

Lemma 7.10. *If $b_2 = c_2^{old}$ or (c_2^{old}, b_2) is an edge, and c_1 is on Y then there exists $c_2^{new} \in \mathrm{adom}(G_2^m)$ such that $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ both satisfy conditions (a) and (b) for validity, and $f(c_1, c_2^{new}) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.*

Lemma 7.9 and 7.10 have considered the scenarios where the Atoms condition was broken when c_1 is located on Y. The scenarios when c_1 is located on W are handled by Lemmas 7.11 and 7.12, and the scenarios when c_1 is located on T are handled by Lemmas 7.13 and 7.14. We now have a look at the scenarios where c_1 is located on W.

Lemma 7.11. *If $a_2 = c_2^{old}$ or (a_2, c_2^{old}) is an edge, and c_1 is on W then there exists $c_2^{new} \in \mathrm{adom}(G_2^m)$ such that conditions (a) and (b) for the validity of both $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ are satisfied, and $f(c_1, c_2^{new}) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.*

Proof. In this case c_2^{old} is on W', whence a_2 is also on W' since $a_2 = c_2^{old}$, or (a_2, c_2^{old}) is an edge. Since $a_1 \neq c_1$ and (a_1, c_1) is not an edge, we have that a_1 is on Y or on T. If a_1 is on Y, then $f(a_1, a_2) < m/2$ since c_1 is not in the three middle columns. Hence whatever new c_2^{new} we pick such that (c_1, c_2) is valid, we have $f(c_1, c_2^{new}) \geq f(a_1, a_2) - 1$ since c_1 and c_2^{new} are either located in the same column as, or in the column next to a_1 and a_2. Thus, if we pick c_2^{new} on the chain that does not contain a_2 and b_2, in the same column as c_1, we have that $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ both satisfy conditions (a) and (b) for validity, and $f(c_1, c_2^{new}) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.

On the other hand, suppose that a_1 is on T then $f(a_1, a_2) = m/2$. This could be problematic if a_1 is T left (respectively T right) and if we cannot put c_2^{new} on the left side of V (respectively the right side of U), in the same column as c_1, simultaneously. That is, if putting c_2^{new} on the left side of V, in the same column as c_1, (respectively right side of U) makes $b_2 = c_2^{new}$ or (c_2^{new}, b_2) an edge. If this is not the case, then we simply put c_2^{new} on V, in the same column as c_1 (respectively U). Then by construction $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ satisfy conditions (a) and (b) for validity and $f(c_1, c_2) = m/2$.

In the problematic case we will show that $f(b_1, b_2)$ is sufficiently low. So in this case putting c_2^{new} in the same column as c_1 on the left side of V (respectively right side of U) violates the Atoms condition for $(c_1, b_1, c_2^{new}, b_2)$. Then b_2 is V left (respectively U right), in the same column as, or in the column next to c_1 and c_2^{old}. Since $c_2^{old} = a_2$ or (a_2, c_2^{old}) is an edge, a_2 must be on W' as well. This implies that $a_2 \neq b_2$ and that (a_2, b_2) is not an edge, since b_2 is on V (respectively U) as mentioned before. Therefore, by the validity of (a_1, b_1, a_2, b_2), we can also conclude that $a_1 \neq b_1$ and that (a_1, b_1) is not an edge. Thus b_1 is certainly not on T since then $a_1 = b_1$ or (a_1, b_1) would be an edge. It cannot be on W either because then $c_1 = b_1$ or (c_1, b_1) would be an edge, which contradicts the given. Thus we may conclude that in this case b_1 is on Y, whence we have $f(b_1, b_2) < m/2$ since b_1 is V left (respectively U right). If we now put c_2^{new} on the chain that does not contain a_2 or b_2, in the same column as c_1, then $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ certainly satisfy conditions (a) and (b) for validity, and we have that $f(c_1, c_2^{new}) \geq f(b_1, b_2) - 1 \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$ since c_1 and c_2^{new} are either in the column next to, or in the same column as b_1 or b_2. For an example of this scenario see Figure 7.6. □

The proof of the following lemma is similar to the proof of Lemma 7.11 where the roles of a_1 and a_2 are replaced by b_1 and b_2, and (a_2, c_2) being an edge is replaced by (c_2, b_2) being an edge.

Lemma 7.12. *If $c_2^{old} = b_2$ or (c_2^{old}, b_2) is an edge, and c_1 is on W then there exists $c_2^{new} \in \mathrm{adom}(G_2^m)$ such that $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ both satisfy conditions (a) and (b) for validity, and $f(c_1, c_2^{new}) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.*

As announced we now look at the scenarios when c_1 is located on T. The reasoning used to prove the following lemma is again analogous to the proof of Lemma 7.11, but since the Greedy Strategy deviates in this

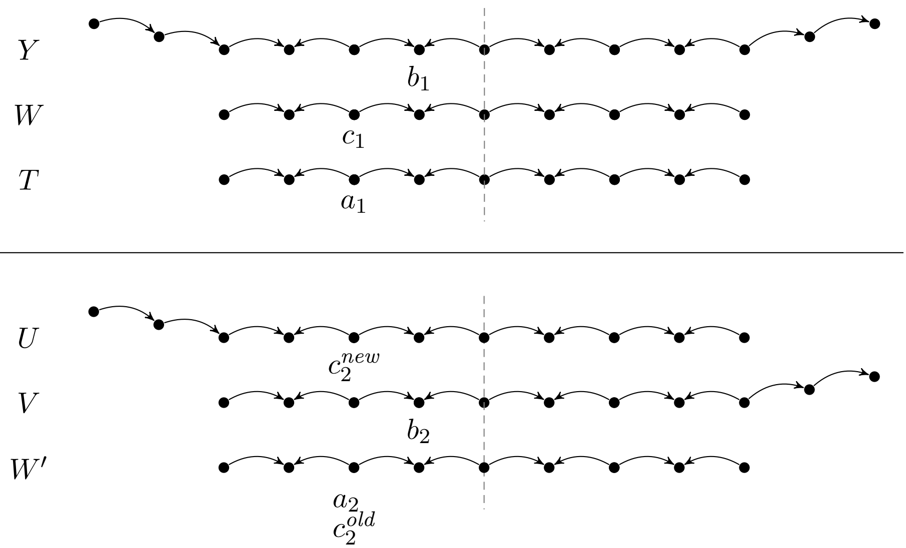


Figure 7.6: An example of a problem scenario in Lemma 7.11. Clearly c_2^{old} breaks the Atoms condition. Furthermore, if we would have picked c_2^{new} on V, (c_2^{new}, b_2) would have been an edge, which is not allowed. Thus we are forced to pick c_2^{new} on U. This, however, is no problem since in this scenario b_1 and b_2 are on sides of chains with different endings.

scenario compared to the scenario of Lemma 7.11, we need to address some detailed differences.

Lemma 7.13. *If* $a_2 = c_2^{old}$ *or* (a_2, c_2^{old}) *is an edge, and* c_1 *is on* T *then there exists* $c_2^{new} \in \mathrm{adom}(G_2^m)$ *such that* $(a_1, c_1, a_2, c_2^{new})$ *and* $(c_1, b_1, c_2^{new}, b_2)$ *both satisfy conditions (a) and (b) for validity, and* $f(c_1, c_2^{new}) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.

Proof. If c_1 is T left, then c_2^{old} is V left, while if c_1 is T right, then c_2^{old} is U right. Furthermore, if c_2^{old} is V left, then a_2 is also V left, and if c_2^{old} is U right, a_2 is also U right. This is because $a_2 = c_2^{old}$ or (a_2, c_2^{old}) is an edge, and c_1 is not located in the middle three columns. Since $a_1 \neq c_1$ and (a_1, c_1) is not an edge, we have that a_1 is on Y or on W. If a_1 is on Y then $f(a_1, a_2) < m/2$ since c_1 is not in the three middle columns. Hence whatever new c_2^{new} we pick in the same column as c_1 we have $f(c_1, c_2^{new}) \geq f(a_1, a_2) - 1$. Thus, if we pick c_2^{new} on the chain that does not contain a_2 and b_2, in the same column as c_1, we have that

$(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ both satisfy conditions (a) and (b) for validity, and $f(c_1, c_2^{new}) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.

On the other hand, suppose that a_1 is on W, then $f(a_1, a_2) = m/2$. This could be problematic if a_1 is W left (respectively W right) and if we cannot put c_2^{new} on W', in the same column as c_1, simultaneously, i.e., if putting c_2^{new} on W', in the same column as c_1, makes $b_2 = c_2^{new}$ or (c_2^{new}, b_2) an edge. If this is not the case we simply put c_2^{new} on W', in the same column as c_1. Then by construction $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ both satisfy conditions (a) and (b) for validity, and $f(c_1, c_2) = m/2$.

In the problematic case we will show that $f(b_1, b_2)$ is sufficiently low. So in this case putting c_2^{new} on W', in the same column as c_1, violates the Atoms condition for $(c_1, b_1, c_2^{new}, b_2)$. Then b_2 is on located on W', in the same column as, or in the column next to c_1 and c_2^{old}. Since $c_2^{old} = a_2$ or (a_2, c_2^{old}) is an edge, and c_1 is not in the middle three columns, a_2 must be on V if c_1 is T left, or on U if c_1 is U right. In either case, this implies that $a_2 \neq b_2$ and that (a_2, b_2) is not an edge, since b_2 is on W' as mentioned before. Therefore, by the validity of (a_1, b_1, a_2, b_2), we can also conclude that $a_1 \neq b_1$ and that (a_1, b_1) is not an edge. Thus b_1 is certainly not on W since then $a_1 = b_1$ or (a_1, b_1) would be an edge. It cannot be on T either because then $c_1 = b_1$ or (c_1, b_1) would be an edge, which contradicts the given. Thus we may conclude that in this case b_1 is on Y, whence we have $f(b_1, b_2) < m/2$ since b_1 is W'. If we now put c_2^{new} on the chain that does not contain a_2 or b_2, in the same column as c_1, then $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ certainly satisfy conditions (a) and (b) for validity, and we have that $f(c_1, c_2^{new}) \geq f(b_1, b_2) - 1 \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$ since c_1 and c_2^{new} are either in the column next to, or in the same column as b_1 or b_2. For an example of this scenario see Figure 7.7. $\square$

The proof of the following lemma is similar to the proof of Lemma 7.13 where the roles of a_1 and a_2 are replaced by b_1 and b_2, and (a_2, c_2) being an edge is replaced by (c_2, b_2) being an edge.

Lemma 7.14. *If $c_2^{old} = b_2$ or (c_2^{old}, b_2) is an edge, and c_1 is on T then there exists $c_2^{new} \in \mathrm{adom}(G_2^m)$ such that $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ both satisfy conditions (a) and (b) for validity, and $f(c_1, c_2^{new}) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.*

Together Lemma 7.9 to 7.14 cover all scenarios for c_1 where one of the Atoms conditions was broken. Thus, all that remains to establish Lemma 7.8 is to show that $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ satisfy conditions (c) and (d) for validity. Let us first take care of $(a_1, c_1, a_2, c_2^{new})$.

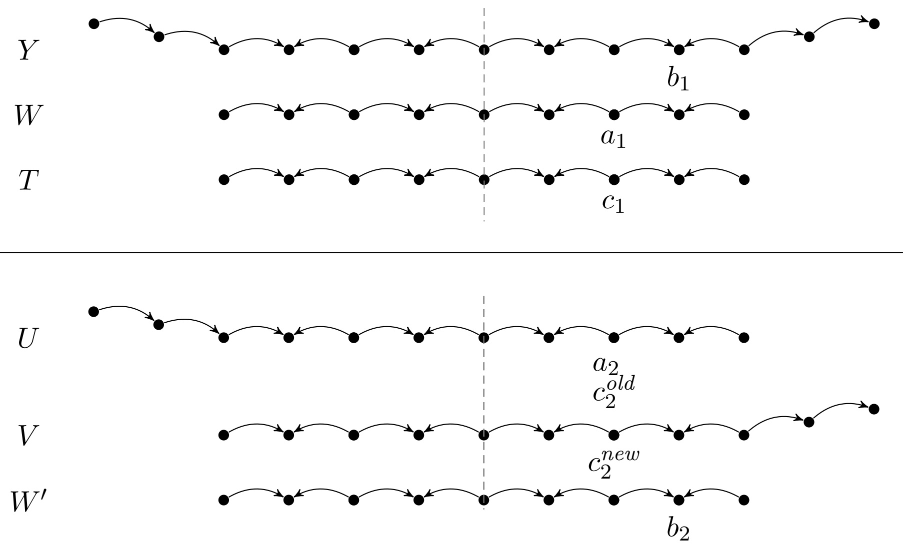


Figure 7.7: An example of a problem scenario in Lemma 7.13. Clearly c_2^{old} breaks the Atoms condition. Furthermore, if we would have picked c_2^{new} on W', (c_2^{new}, b_2) would have been an edge, which is not allowed. Thus we are forced to pick c_2^{new} on V. This, however, is no problem since in this scenario b_1 and b_2 are on sides of chains with different endings.

Lemma 7.15. *Let $c_2^{new} \in \mathrm{adom}(G_2^m)$ be the node chosen in Lemmas 7.9 to 7.14. Then $(a_1, c_1, a_2, c_2^{new})$ also satisfies conditions (c) and (d) for validity.*

Proof. Condition (c) is only involved when $(a_1, c_1) = (x_2, y_2)$, a case we have already excluded at the start of the proof.

Condition (d) is only involved when $(a_2, c_2^{new}) = (x_2', u_2)$. Since (a_1, b_1, a_2, b_2) is valid, we must have that $a_1 = x_2$, whence we have $f(a_1, a_2) = m/2$. We now show that $c_1 = y_2$. Suppose for the sake of contradiction that $c_1 \neq y_2$. Then by definition $f(c_1, c_2^{new}) = 1$. Furthermore, we have $c_2^{old} = v_2$ or $c_2^{old} = w_2'$ by the Greedy Strategy. Since $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1 = \min(m/2, f(b_1, b_2)) - 1 = f(b_1, b_2) - 1$ by assumption, we have $f(b_1, b_2) \leq 2$. Remember that the Atoms condition for either $(a_1, c_1, a_2, c_2^{old})$ or $(c_1, b_1, c_2^{old}, b_2)$ was broken. Notice that in this case the Atoms condition for $(a_1, c_1, a_2, c_2^{old})$ was not broken, since c_1 and c_2^{old} are two columns to the right of a_1 and a_2. Thus the Atoms condition for $(c_1, b_1, c_2^{old}, b_2)$ was broken. Hence $c_2^{old} = b_2$ or (c_2^{old}, b_2) is an edge (because by assumption c_1 is not related to b_1). It is not possible for (c_2^{old}, b_2) to be an edge since v_2 and w_2' have no outgoing edges. Thus we may conclude that $c_2^{old} = b_2 = v_2$ or $c_2^{old} = b_2 = w_2'$. Hence $b_1 = y_2$ in both cases since $f(b_1, b_2) \leq 2$. Therefore $(a_1, b_1, a_2, b_2) = (x_2, y_2, x_2', b_2)$ where $b_2 = v_2$ or w_2', which contradicts condition (c) for the validity of (a_1, b_1, a_2, b_2). $\square$

Finally, we take care of $(c_1, b_1, c_2^{new}, b_2)$.

Lemma 7.16. *Let $c_2^{new} \in \mathrm{adom}(G_2^m)$ be the node chosen in Lemmas 7.9 to 7.14. Then $(c_1, b_1, c_2^{new}, b_2)$ also satisfies conditions (c) and (d) for validity.*

Proof. Condition (c) is only involved when $b_1 = y_2$. Then $b_2 = u_2$ since $f(b_1, b_2) > 1$, as desired.

Condition (d) is only involved when $b_2 = u_2$. Then $b_1 = y_2$ since $f(b_1, b_2) > 1$, as desired. $\square$

Together Lemmas 7.15 and 7.16 establish that both $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ also satisfy conditions (c) and (d) for validity. Since we already established that $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ satisfy conditions (a) and (b) for validity, we may conclude that $(a_1, c_1, a_2, c_2^{new})$ and $(c_1, b_1, c_2^{new}, b_2)$ are both valid, which concludes the proof of Lemma 7.8. $\square$

The proof of next lemma is analogous to the proof of Lemma 7.8. Indeed, this is because of the same reasons why the proof of Lemma 7.7 was analogous to the proof of Lemma 7.6.

Lemma 7.17. *Suppose that (a_1, b_1, a_2, b_2) is valid, $f(a_1, a_2) > 0$, $f(b_1, b_2) > 1$ and $c_2 \in \mathrm{adom}(G_2^m)$ such that $a_1 \neq c_1$, $c_1 \neq b_1$, (a_1, c_1) and (c_1, b_1) are not edges. Then there exists $c_1 \in \mathrm{adom}(G_1^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid, and $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.*

Combining Lemmas 7.6 and 7.8 we get the following corollary.

Corollary 7.18. *If (a_1, b_1, a_2, b_2) is valid, $f(a_1, a_2) > 0$ and $f(b_1, b_2) > 1$, then for every $c_1 \in \mathrm{adom}(G_1^m)$ there exists $c_2 \in \mathrm{adom}(G_2^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid, and $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.*

We will see later that this corollary is crucial to show that the duplicator has a winning strategy starting in (a_1, b_1, a_2, b_2).

On the other hand, combining Lemmas 7.7 and 7.17 yields the following corollary.

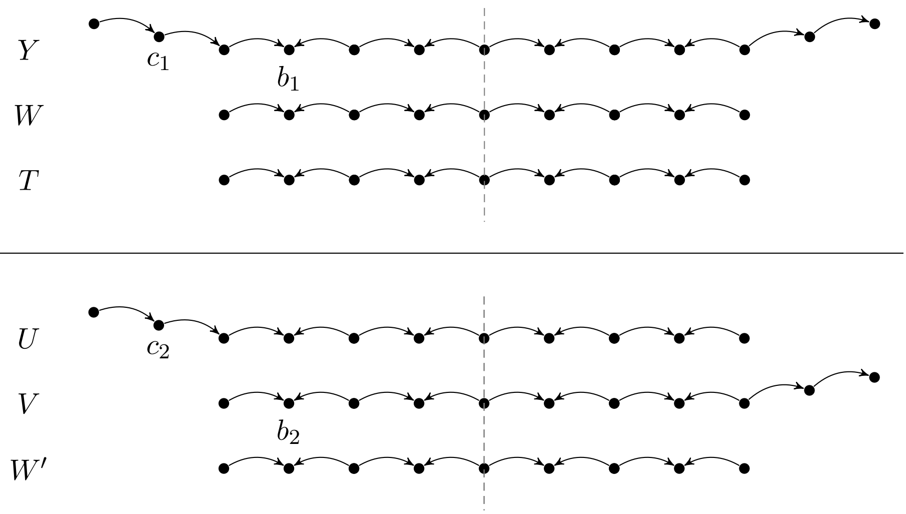


Figure 7.8: An example of a problem scenario where we are forced to pick a c_2 such that (c_1, b_1, c_2, b_2) does not satisfy condition (c) for validity. It turns out that it is sufficient to only satisfy the Atoms condition because this scenario only occurs when $f(b_1, b_2) = 1$.

Corollary 7.19. *If* (a_1, b_1, a_2, b_2) *is valid,* $f(a_1, a_2) > 0$ *and* $f(b_1, b_2) > 1$*, then for every* $c_2 \in \mathrm{adom}(G_2^m)$ *there exists* $c_1 \in \mathrm{adom}(G_1^m)$ *such that* (a_1, c_1, a_2, c_2) *and* (c_1, b_1, c_2, b_2) *are valid, and* $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$*.*

Notice that until now we have always required that $f(b_1, b_2) > 1$. The cases where $f(b_1, b_2) = 1$ are handled separately. Indeed, when $f(b_1, b_2) = 1$, we cannot necessarily guarantee that (c_1, b_1, c_2, b_2) is valid (see Figure 7.8). We can only guarantee the Atoms condition as shown Lemmas 7.20 and 7.21. This will turn out to be sufficient.

Lemma 7.20. *Suppose that* (a_1, b_1, a_2, b_2) *is valid,* $f(a_1, a_2) > 0$*,* $f(b_1, b_2) = 1$*. Then, for every* $c_1 \in \mathrm{adom}(G_1^m)$ *there exists* $c_2 \in \mathrm{adom}(G_2^m)$ *such that* (a_1, c_1, a_2, c_2) *and* (c_1, b_1, c_2, b_2) *satisfy the Atoms condition.*

Proof. Careful inspection of the proofs of Lemmas 7.6 and 7.8 reveals that $f(b_1, b_2) > 1$ is only used for showing conditions (c) and (d) for the validity of (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2), except in the case where $c_1 = y_1$ and $b_2 = u_2$ (item marked with $\star$ in the proof of Lemma 7.8). If we are not in this case, we can simply pick the same c_2 as in these proofs.

Now suppose we are in this exceptional case. Since $f(b_1, b_2) = 1$, b_1 is not on Y. Notice that (a_1, c_1) cannot be an edge, since then $a_1 = x_2$, and hence also $a_2 = x'_2$ since (a_1, b_1, a_2, b_2) is valid. Thus we have $(a_1, b_1, a_2, b_2) = (x_2, b_1, x'_2, u_2)$. Condition (d) for the validity of (a_1, b_1, a_2, b_2) then implies that $b_1 = y_2$, which contradicts the fact that b_1 is not on Y.

If $a_1 = c_1$, then we pick $a_2 = c_2$. Notice that in this case $b_2 \neq c_2$. Indeed, if $b_2 = c_2 = a_2$, then $a_1 = b_1$ by the validity of (a_1, b_1, a_2, b_2). Thus $c_1 = b_1$ which is a contradiction.

On the other hand, if $a_1 \neq c_1$, we simply pick c_2 on the chain not containing a_2 or b_2, in the same column as c_1. This is possible since there are three chains. □

The proof of the following lemma is analogous to the proof of the previous lemma. This is because of the same reasons why the proof of Lemma 7.7 was analogous to the proof of Lemma 7.6.

Lemma 7.21. *Suppose that (a_1, b_1, a_2, b_2) is valid, $f(a_1, a_2) > 0$, $f(b_1, b_2) = 1$. Then, for every $c_2 \in \mathrm{adom}(G_2^m)$ there exists $c_1 \in \mathrm{adom}(G_1^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) satisfy the Atoms condition.*

We are now ready to show our key lemma.

Lemma 7.22. *Let s be a natural number and let $m > 4$ be a natural number divisible by four. If $(a_1, b_1, a_2, b_2) \in \mathrm{adom}(G_1^m)^2 \times \mathrm{adom}(G_2^m)^2$ is valid and $s \leq \min(f(a_1, a_2), f(b_1, b_2))$, then $(G_1^m, a_1, b_1) \simeq_s (G_2^m, a_2, b_2)$.*

Proof. We proceed by induction on s. If $s = 0$ then $(G_1^m, a_1, b_1) \simeq_s (G_2^m, a_2, b_2)$ since the Atoms condition is implied by the validity of (a_1, b_1, a_2, b_2).

Now let $s > 0$, so both $f(a_1, a_2) > 0$ and $f(b_1, b_2) > 0$. If $f(b_1, b_2) = 1$ then Lemma 7.20 implies that for every $c_1 \in \mathrm{adom}(G_1^m)$, there exists $c_2 \in \mathrm{adom}(G_2^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_2, c_2, b_2) satisfy the Atoms condition. This, however, is equivalent to

$$(G_1^m, a_1, c_1) \simeq_0 (G_2^m, a_2, c_2) \quad \text{and} \quad (G_1^m, c_1, b_1) \simeq_0 (G_2^m, c_2, b_2).$$

Hence the Forth condition holds. Furthermore, Lemma 7.21 implies that for every $c_2 \in \mathrm{adom}(G_2^m)$, there exists $c_1 \in \mathrm{adom}(G_1^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_2, c_2, b_2) both satisfy the Atoms condition. Again this is equivalent to $(G_1^m, a_1, c_1) \simeq_0 (G_2^m, a_2, c_2)$ and $(G_1^m, c_1, b_1) \simeq_0 (G_2^m, c_2, b_2)$. Hence the Back condition holds. Thus $(G_1^m, a_1, b_1) \simeq_1 (G_2^m, a_2, b_2)$.

Now suppose that $f(a_1, a_2) > 0$ and $f(b_1, b_2) > 1$. We will first show that the Forth condition holds. Suppose that $c_1 \in \operatorname{adom}(G_1^m)$. Then by Corollary 7.18 there exists $c_2 \in \operatorname{adom}(G_2^m)$ such that both (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid and $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$. Furthermore, $f(c_1, c_2) \geq s - 1$ since $s - 1 \leq \min(f(a_1, a_2), f(b_1, b_2)) - 1$. Hence $s - 1 \leq \min(f(c_1, c_2), f(a_1, a_2))$ and $s - 1 \leq \min(f(c_1, c_2), f(b_1, b_2))$. Therefore we can apply our induction hypothesis, which tells us that $(G_1^m, a_1, c_1) \simeq_{s-1} (G_2^m, a_2, c_2)$ and $(G_1^m, c_1, b_1) \simeq_{s-1} (G_2^m, c_2, b_2)$ as desired.

The Back condition is verified similarly using Corollary 7.19. $\square$

Theorem 7.4 finally follows:

Proof of Theorem 7.4. First, if $(a_1, b_1) = (y_{m/2+1}, y_{m/2+2})$, then we pick the pair $(a_2, b_2) = (u_{m/2+1}, u_{m/2+2})$. In this case (a_1, b_1, a_2, b_2) is valid, $f(a_1, a_2) = m/2$ and $f(b_1, b_2) = m + 1 - (m/2 + 2) = m/2 - 1$ and thus $(G_1, a_1, b_1) \simeq_{m/2-1} (G_2, a_2, b_2)$ due to Lemma 7.22.

If $(a_1, b_1) \neq (y_{m/2+1}, y_{m/2+2})$ then we use the following strategy:

$$\begin{aligned}
a_1 = y_i \wedge 0 \leq i \leq m/2 + 1 &\implies a_2 = u_i \\
a_1 = y_i \wedge m/2 + 1 < i \leq m + 1 &\implies a_2 = v_i \\
a_1 = w_i &\implies a_2 = w_i' \\
a_1 = t_i &\implies a_2 = w_i'
\end{aligned}$$

We use the same strategy to determine b_2 from b_1. Clearly in this case (a_1, b_1, a_2, b_2) is valid, and $f(a_1, a_2) = f(b_1, b_2) = m/2$, whence we have $(G_1, a_1, b_1) \simeq_{m/2-1} (G_2, a_2, b_2)$ due to Lemma 7.22. $\square$

The bisimulations that we use always require that (a_1, b_1, a_2, b_2) is valid. There might be a a bisimulation of a larger depth when we remove this restriction. It turns out that we can find an upper bound on the depth.

Proposition 7.23. *There is no bisimulation between* $(G_1^m, y_{\frac{m}{2}+1}, y_{\frac{m}{2}+1})$ *and* (G_2^m, a, b) *for every* $(a, b) \in \operatorname{adom}(G_1^m)^2$ *of depth* $3m/4 + 2$.

Proof. By Theorem 7.3 it suffices to show that there exists an expression $e \in \mathcal{N}(-, \mathsf{di})$ of degree $3m/4 + 2$ such that $(y_{\frac{m}{2}+1}, y_{\frac{m}{2}+1}) \in e(G_1^m)$ and

$(a, b) \notin e(G_2^m)$. To this end, define the following family of expressions:

$$
\begin{aligned}
e_0 &:= \pi_2(R^3) \\
e'_0 &:= \pi_1(R^2) \\
e_1 &:= \pi_1(R \circ e_0) \\
e_{n+1} &:= \pi_1(R \cap ((R \circ \mathsf{di}) \circ (e_n \circ R))) && \text{(for } n > 1) \\
e'_{n+1} &:= \pi_1(R \cap ((R \circ \mathsf{di}) \circ (e'_n \circ R))) && \text{(for } n > 0)
\end{aligned}
$$

For $n = 1, \ldots, m/2$, we have $(y_{2n+1}, y_{2n+1}) \in e_n(G_1^m)$ and $(y_{m+1-2n}, y_{m+1-2n}) \in e'_n(G_1^m)$. Thus we may also conclude that $(y_{\frac{m}{2}+1}, y_{\frac{m}{2}+1}) \in e_{m/4} \cap e'_{m/4}(G_1^m)$.

On the other hand, $e_n(G_2^m)$ only contains pairs of nodes on U, while $e'_n(G_2^m)$ only contains nodes on V for any $n = 1 \ldots m/2$. Hence $e_n \cap e'_n(G_2^m)$ is empty for $n = 1, \ldots, m/2$. Thus we may conclude that $e_{m/4} \cap e'_{m/4}(G_2^m)$ is empty, and thus does not contain (a, b) either.

Since e_n and e'_n have degree $3n+2$, the degree of $e_{m/4} \cap e'_{m/4}$ is $3m/4+2$ as desired. □

8

A monotone preservation result for containments of conjunctive queries

In this chapter, we show the following preservation for monotone containments of conjunctive queries. Recall that MON denotes the family of all monotone Boolean queries.

Theorem 8.1. *For every database schema* Γ*,* $\mathrm{CQ}_\Gamma{}^{\subseteq} \cap \mathrm{MON} = \mathrm{CQ}_\Gamma{}^{\neq\emptyset}$*. This equality remains true in the presence of unsafe CQs.*

Note that $\mathrm{CQ}_\Gamma{}^{\neq\emptyset} \subseteq \mathrm{CQ}_\Gamma{}^{\subseteq} \cap \mathrm{MON}$ already follows from the fact that $Q \neq \emptyset$ is equivalent to $() \leftarrow \emptyset \subseteq () \leftarrow B_Q$ (cf. Theorem 3.9(3)). To prove the remaining inclusion we first establish a few technical results. First, we show that any monotone containment of conjunctive queries is equivalent to a containment of conjunctive queries with empty heads. For the remainder of this section, we write Z_a to be the instance, where there is exactly one fact $R(a, \ldots, a)$ for every $R \in \Gamma$. Note that for every CQ Q, we have $Q(Z_a) = \{(a, a, \ldots, a)\}$.

Lemma 8.2. *Let* Q_1 *and* Q_2 *be conjunctive queries that can be unsafe. If* $Q_1 \subseteq Q_2$ *is monotone, then it is equivalent to the conjunctive query* $() \leftarrow B_{Q_1} \subseteq () \leftarrow B_{Q_2}$*.*

Proof. Instead of working with the regular definition of CQs introduced in Section 2.1.2, we work with a slightly more general version of CQs that produce output according to the named perspective of the relational-model [1]. In this perspective, tuples are defined over a finite set of attributes, which we refer to as a *relation scheme*. Formally, *tuples*, say $H = (u_i)_{i \in S}$ on a relation scheme S, are considered as mappings, so H is a mapping on S and $H(i) = u_i$. Then, subtuples, say $H|_K$ for $K \subseteq S$ are treated as restrictions of the mapping H to K.

We now adapt conjunctive queries. In this proof, a *conjunctive query* is an expression of the form $Q : H \leftarrow B$ where the head H is a tuple over some relation scheme S and the body B is a set of atoms over Γ as defined in Section 2.1.2. We write B_Q for the body and H_Q for the head of Q. The *result scheme* of a conjunctive query Q is the relation scheme of the head H_Q. Then, semantically, for every instance I over Γ, $Q(I)$ is defined as:

$$\{f \circ H_Q \mid f \text{ is a homomorphism from } Q \text{ into } I\}.$$

Here, a homomorphism f from Q into I is a function on the variables in H_Q and B_Q to $\text{adom}(I)$ such that $f(B_Q) \subseteq I$. In this perspective, we only allow containments of conjunctive queries with the same relation scheme.

Note that CQs as defined in Section 2.1.2 can easily be expressed using our new CQs. Indeed, a tuple of variables $(v_1, \ldots, v_n)$ in the context of Section 2.1.2 can be seen as the mapping $i \mapsto v_i$ on the relation scheme $\{1, \ldots, n\}$.

Let S be the result scheme of Q_1 and Q_2. Let us write B_{Q_2} as $B_1, \ldots, B_k, B$ where the B_j are the connected components of B_{Q_2} that contain at least one variable in H_{Q_2}, and B is the collection of the remaining connected components.

Define $A_j = \{i \in S \mid H_{Q_2}(i) \in \text{adom}(B_j)\}$ for $j = 1, \ldots, k$ and let A_0 contain the remaining attributes in S. Furthermore, define $A = \bigcup_{1 \leq j \leq k} A_j$.

We first show that there is a function h such that $h \circ H_{Q_2}|_{A_0} = H_{Q_1}|_{A_0}$. Let a be a fresh data element. Define $I = Z_a \cup B_{Q_1} \cup \bigcup_{i \in C} Z_{H_{Q_1}(i)}$ where $C = \{i \in S \mid H_{Q_1}(i) \notin \text{adom}(B_{Q_1})\}$. Since, $Q_1(Z_a) = Q_2(Z_a)$ and $Q_1 \subseteq Q_2$ is monotone, we have $Q_1(I) \subseteq Q_2(I)$. Therefore, $H_{Q_1} \in Q_2(I)$ since $H_{Q_1} \in Q_1(I)$. Hence, there is a homomorphism from Q_2 into I such that $f \circ H_{Q_2} = H_{Q_1}$. In particular, $f \circ H_{Q_2}|_{A_0} = H_{Q_1}|_{A_0}$ as desired.

Next, we show for each $j = 1, \ldots, k$ that

$$(H_{Q_1}|_{A_j} \leftarrow B_{Q_1}) \sqsubseteq (H_{Q_2}|_{A_j} \leftarrow B_j). \qquad (\star)$$

Let I be an instance over Γ and let a be a fresh data element. Suppose $H \in (H_{Q_1}|_{A_j} \leftarrow B_{Q_1})(I)$. Since $(H_{Q_1}|_{A_j} \leftarrow B_{Q_1})$ and Q_1 have the same body, and $H_{Q_1}|_{A_j}$ is a subtuple of H_{Q_1}, we can extend H to H' such that $H' \in Q_1(I)$. Furthermore, since $Q_1 \subseteq Q_2$ is monotone and $Q_1(Z_a) = Q_2(Z_a)$, we have $Q_1(I \cup Z_a) \subseteq Q_2(I \cup Z_a)$. Thus, $H' \in Q_2(I \cup Z_a)$, whence we also have $H \in (H_{Q_2}|_{A_j} \leftarrow B_j)(I \cup Z_a)$. Since $H_{Q_2}|_{A_j} \leftarrow B_j$ is additive, $H \in (H_{Q_2}|_{A_j} \leftarrow B_j)(I) \cup (H_{Q_2}|_{A_j} \leftarrow B_j)(Z_a)$. This implies that $H \in (H_{Q_2}|_{A_j} \leftarrow B_j)(I)$ since H is a tuple of data elements in I.

We now show that $Q_1 \subseteq Q_2$ is equivalent to $Q_1' \subseteq Q_2'$ where $Q_1' = () \leftarrow B_{Q_1}$ and $Q_2' = () \leftarrow B_{Q_2}$, which proves our lemma. Clearly, $Q_1(I) \subseteq Q_2(I)$ implies that $Q_1'(I) \subseteq Q_2'(I)$. For the other direction, suppose that $Q_1'(I) \subseteq Q_2'(I)$ and let $H \in Q_1(I)$. Then, we have the following:

- There is a homomorphism f_1 from B_{Q_1} to I such that $f_1 \circ H_{Q_1} = H$.
- There is a homomorphism f_2 from B_{Q_2} to I since $\emptyset \neq Q_1'(I) \subseteq Q_2'(I)$.
- There is a function h such that $h \circ H_{Q_2}|_{A_0} = H_{Q_1}|_{A_0}$.
- For every $j = 1, \ldots, k$, $H|_{A_j} \in (H_{Q_2}|_{A_j} \leftarrow B_j)(I)$ by $(\star)$. Hence, there is a homomorphism h_j from B_j into I such that $h_j \circ H_{Q_2}|_{A_j} = H|_{A_j}$.

We now construct a homomorphism f from Q_2 into I such that $f \circ H_{Q_2} = H$. We define this f as follows:

$$f : x \mapsto \begin{cases} f_2(x), & \text{if } x \in B; \\ h_j(x), & \text{if } x \in \operatorname{adom}(B_j); \\ f_1 \circ h(x), & \text{otherwise.} \end{cases}$$

We first show that $f \circ H_{Q_2} = H$.

$$\begin{aligned} f \circ H_{Q_2} &= f \circ (H_{Q_2}|_{A_0} \cup \bigcup_{1 \le j \le k} H_{Q_2}|_{A_j}) \\ &= f \circ H_{Q_2}|_{A_0} \cup \bigcup_{1 \le j \le k} f \circ H_{Q_2}|_{A_j} \\ &= f_1 \circ h \circ H_{Q_2}|_{A_0} \cup \bigcup_{1 \le j \le k} h_j \circ H_{Q_2}|_{A_j} \\ &= f_1 \circ H_{Q_1}|_{A_0} \cup \bigcup_{1 \le j \le k} H|_{A_j} \\ &= H|_{A_0} \cup \bigcup_{1 \le j \le k} H|_{A_j} = H \end{aligned}$$

Finally, we show that $f(B_{Q_2}) \subseteq I$.

$$\begin{aligned} f(B_{Q_2}) = f(B \cup \bigcup_{1\le j\le k} B_j) &= f(B) \cup \bigcup_{1\le j\le k} f(B_j) \\ &= f_2(B) \cup \bigcup_{1\le j\le k} h_j(B_j) \\ &\subseteq I \end{aligned}$$

□

To prove Theorem 8.1 we may thus limit ourselves to conjunctive queries with empty heads. First, we have a look at containments of the form $Q_1 \subseteq Q_2$ where B_{Q_1} contains at least two non-redundant atoms. In what follows, when we write that a conjunctive query Q is *minimal*, we mean that B_Q does not contain redundant atoms.

Lemma 8.3. *Let Q_1 and Q_2 be* CQ*s where Q_1 is minimal and $H_{Q_1} = H_{Q_2} = ()$. If B_{Q_1} contains at least two atoms, then $Q_1 \subseteq Q_2$ is equivalent to* true *or is not monotone.*

Proof. If $Q_1 \subseteq Q_2$ is not equivalent to true, then $Q_1 \not\subseteq Q_2$. Thus, $Q_2(B_{Q_1}) = \emptyset$, whence we have $Q_1(B_{Q_1}) \not\subseteq Q_2(B_{Q_1})$. Since $|B_{Q_1}| \geq 2$, there exists a nonempty $B \subsetneq B_{Q_1}$. We have $Q_1(B) = \emptyset$ for otherwise Q_1 would not be minimal.

Clearly, $Q_1(B) = \emptyset$ implies that $Q_1(B) \subseteq Q_2(B)$. Hence, $Q_1 \subseteq Q_2$ is not monotone. □

We are now ready to prove Theorem 8.1.

Proof of Theorem 8.1. Let $Q_1 \subseteq Q_2$ be in $\mathrm{CQ}_\Gamma{}^{\subseteq} \cap \mathrm{MON}$. By Lemma 8.2 we may assume that $H_{Q_1} = H_{Q_2} = ()$. We may furthermore assume that Q_1 is minimal. The constant true query is expressed by $() \leftarrow \emptyset \neq \emptyset$, so we may assume that $Q_1 \not\subseteq Q_2$. Thus, $Q_2(B_{Q_1}) = \emptyset$.

If B_{Q_1} contains at least two atoms, then $Q_1 \subseteq Q_2$ is equivalent to true by Lemma 8.3, which we have already considered.

If $B_{Q_1} = \emptyset$, then $Q_1 \subseteq Q_2$ is equivalent to $Q_2 \neq \emptyset$ which is in $\mathrm{CQ}_\Gamma{}^{\neq\emptyset}$.

Finally, suppose that B_{Q_1} contains exactly one atom. First, let us consider $B_{Q_1} = \{R(x_1, \ldots, x_n)\}$ where there is a repetition among $x_1, \ldots, x_n$. Define $I_1 = \{R(y_1, \ldots, y_n)\}$ where $y_1, \ldots, y_n$ are all different and not equal to any of $x_1, \ldots, x_n$. Clearly, $Q_1(I_1) = \emptyset$. Since $Q_2(B_{Q_1}) = \emptyset$, there is a connected component C of B_{Q_2} that does not map in B_{Q_1}. Furthermore,

C does not map into I_1 either, whence we also have $Q_2(I_1) = \emptyset$. Indeed, if C would map into I_1, then C would also map into B_{Q_1} since I_1 maps into B_{Q_1}. It follows that C does not map into $I_1 \cup B_{Q_1}$ either, since C is connected and adom(I_1) is disjoint from adom(B_{Q_1}). Therefore, $Q_2(I_1 \cup B_{Q_1}) = \emptyset$. Hence, $Q_1(I_1 \cup B_{Q_1}) \nsubseteq Q_2(I_1 \cup B_{Q_1})$ since the head of Q_1 is in $Q_1(I_1 \cup B_{Q_1})$. This contradicts that $Q_1 \subseteq Q_2$ is monotone, since $Q_1(I_1) = \emptyset \subseteq Q_2(I_1)$. So, the only body left to consider is $B_{Q_1} = \{R(x_1, \ldots, x_n)\}$ where $x_1, \ldots, x_n$ are all different and $R \in \Gamma$. Our proof now depends on the number of relations in Γ.

1. Suppose that Γ only contains the relation name R. Then $Q_1(I) \neq \emptyset$ for any instance I over Γ since $B_{Q_1} = \{R(x_1, \ldots, x_n)\}$ where $x_1, \ldots, x_n$ are all different. We may thus conclude that $Q_1 \subseteq Q_2$ is equivalent to $Q_2 \neq \emptyset$ in $\mathrm{CQ}_{\Gamma}{}^{\neq\emptyset}$.

2. Suppose that Γ only contains R and exactly one other relation name T. Define $I_1 = \{T(y_1, \ldots, y_m)\}$ where $y_1, \ldots, y_m$ are different from each other and from $x_1, \ldots, x_n$. Since the body of Q_1 is an R-atom and I_1 only contains a T-atom, we have $Q_1(I_1) = \emptyset$. Hence, $Q_1(I_1) \subseteq Q_2(I_1)$. By the monotonicity of $Q_1 \subseteq Q_2$, we also have $Q_1(I_1 \cup B_{Q_1}) \subseteq Q_2(I_1 \cup B_{Q_1})$. Therefore, every connected component of B_{Q_2} maps in I_1 or B_{Q_1}. Indeed, $Q_2(I_1 \cup B_{Q_1}) \neq \emptyset$ since the head of Q_1 is in $Q_1(I_1 \cup B_{Q_1})$. This observation partitions the connected components of B_{Q_2} into two sets B' and B'', where B' contains the components that map into I_1, and B'' contains the components that map into B_{Q_1}.

   We now show that $Q_1 \subseteq Q_2$ is equivalent to $Q' = () \leftarrow B'$. To this end, suppose that $Q'(I) \neq \emptyset$ and $Q_1(I) \neq \emptyset$ for some instance I over Γ. Thus B' and B_{Q_1} map into I. Since B'' maps into B_{Q_1} by construction, we also have that B'' map into I. Hence, $Q_2(I) \neq \emptyset$ as desired. For the other direction, suppose that $Q_1(I) \subseteq Q_2(I)$ for some instance I over Γ. If $Q_1(I) \neq \emptyset$, then $Q_2(I) \neq \emptyset$ by assumption. Clearly, $Q'(I) \neq \emptyset$ since $B_{Q'}$ is a subset of B_{Q_2}. On the other hand, if $Q_1(I) = \emptyset$, then I has no R-facts. Since instances cannot be empty, it must contain at least one T-fact, so I_1 maps into I. Thus B' also maps into I, whence we have $Q'(I) \neq \emptyset$ as desired.

3. Finally, suppose that Γ contains at least three relation names. Since $Q_2(B_{Q_1}) = \emptyset$, there is a connected component C of B_{Q_2} that does not map into B_{Q_1}. In particular, we know that C is not empty,

whence it contains at least one atom, say a T-atom. (Note that T can be equal to R or not.) Since there are three relation names in Γ there is at least one other relation name S in Γ that is not equal to T or R. Define $I_2 = \{S(z_1, \ldots, z_l)\}$ where $z_1, \ldots, z_l$ are all different from each other and from $x_1, \ldots, x_n$. By construction, C do not map into I_2 either, since C contains an atom different from S. Thus, $Q_2(I_2 \cup B_{Q_1}) = \emptyset$, whence we have $Q_1(I_2 \cup B_{Q_1}) \nsubseteq Q_2(I_2 \cup B_{Q_1}) \neq \emptyset$ since $Q_1(I_2 \cup B_{Q_1}) \neq \emptyset$. However, $Q_1(I_2) = \emptyset$ since R and S are different, which implies that $Q_1(I_2) \subseteq Q_2(I_2)$. This contradicts the assumption that $Q_1 \subseteq Q_2$ is monotone.

$\square$

9
Conclusion

In this thesis, we have outlined a framework along which we can investigate Boolean queries. Firstly, we have identified three natural base modalities to express Boolean queries: nonemptiness, emptiness and containment. Secondly, we have outlined themes along which we investigate Boolean query families that stem from these base modalities:

- **Comparing the base modalities for fixed query languages**
  We have investigated this theme in Chapter 3. First, we have identified query features that enable the expression of one base modality in terms of another one. These query features are the constant empty query; set difference; cylindrification; complementation; and tests.

  We have also identify general properties that reflect the absence of these query features, notably, the properties of monotonicity and additivity. We have then shown how these properties indeed prevent going from one modality to another.

  We then applied these results to conjunctive queries and navigational graph query languages.

- **Comparing different query languages under fixed modalities**
  We have investigated this theme in Chapter 4. We have noted that this theme is particularly interesting when a query language has a lot of different operators that can be included or be left out. The navigational graph query languages are of this nature. We have focused on these languages in this theme.

For the (co)projection restricted fragments, subsumption under nonemptiness has already been completely characterized [21].

With a simple reduction we have shown that subsumption of navigational query fragments under emptiness coincides with the subsumption under nonemptiness.

Finally, under containment we have completely characterized subsumption for unrestricted fragments, i.e., fragments can contain just a single projection or coprojection. We have shown that every operator is primitive, i.e., every operator adds expressive power on its own. Thus, subsumption among fragments under the containment modality behaves the same as subsumption for path queries. This was not obvious, since under nonemptiness subsumption behaves very differently.

- **Comparing different query languages under different base modalities** We have investigated this theme in Chapter 5. Just as in the previous theme, this theme is particularly interesting when a query language has a lot of different operators that can be included or be left out. The navigational graph query languages are of this nature. We have focused on these languages in this theme.

  We have been able to characterize exactly when $F_1^{\subseteq} \subseteq F_2^{\neq\emptyset}$ and $F_1^{\subseteq} \subseteq F_2^{=\emptyset}$ for unrestricted graph query fragments F_1 and F_2,

  On the other hand, we have been able to characterize exactly when $F_1^{\neq\emptyset} \subseteq F_2^{=\emptyset}$ for (co)projection restricted fragments F_1 and F_2.

  We have not been able to fully characterize $F_1^{\neq\emptyset} \subseteq F_2^{\subseteq}$. We, however, conjecture that

  $$F_1^{\neq\emptyset} \subseteq F_2^{\subseteq} \text{ iff } F_1^{\neq\emptyset} \subseteq F_2^{\neq\emptyset} \text{ and } F_2^{\neq\emptyset} \subseteq F_2^{\subseteq}$$

  for (co)projection restricted fragments F_1 and F_2. We have been able to show this for the most part. Only the fragments where $\pi \in \widetilde{F_1}$ and $F_2 \subseteq \{\mathsf{di}, ^{-1}, ^{+}\}$ are still open. Even in this open case, we have been able to prove the conjecture for the union-free subfragment of $\{\mathsf{all}, ^{-1}\}^{\subseteq}$. To prove the conjecture in this case, we have proven a preservation result: $\mathrm{CQ}^{\subseteq} \cap \mathrm{MON} = \mathrm{CQ}^{\neq\emptyset}$. To prove the full conjecture when using the same proof strategy, we would have to prove a similar preservation result for conjunctive queries with union and nonequalities, but we have not been able to find such a preservation result.

The aforementioned preservation theorem is interesting in its own right. Indeed, in finite model theory, model theory and database theory, preservation theorems have been studied in detail [15, 12, 37, 24, 3, 39]. As future work, it could be interesting to find preservation theorems for larger languages and/or even other semantic properties.

- **Closure under Boolean connectives** We have investigated this theme in Chapter 6. We have already compared nonemptiness to emptiness. Remember that this comparison is equivalent to asking whether query languages are closed under negation. We have asked the same question for the containment modality. In particular, we have shown that (unions of) conjunctive queries under containment are not closed under negation. Furthermore, we have been able to characterize when graph query languages under containment are closed under negation.

  Obviously, this question can be generalized to other Boolean connectives. For this question we have again focused on conjunctive queries and navigational graph query languages. We have shown that (unions of) conjunctive queries under nonemptiness are always closed under conjunction. Under emptiness, the same holds for unions of conjunctive queries. For conjunctive queries under emptiness and containment, however, this is no longer true. We have shown that the answer depends on the database schema.

  The navigational graph query languages under emptiness are always closed under conjunction since union is always present. Under nonemptiness, however, this is no longer true. We have been able to characterize when graph query languages under nonemptiness are closed under conjunction.

  We have not been able to characterize closure under conjunction for navigational graph query languages under containment. We do, however, conjecture that a fragment under containment is closed under conjunction iff difference is present. We have been able to show this conjecture for a small number of fragments. Thus, this question remains open for the majority of fragments.

  As future work, we can investigate whether unions of conjunctive queries under the containment modality are closed under conjunction.

Another interesting line of future work would be to consider other Boolean connectives such as disjunction.

Finally, in Chapter 7 we have shown that converse elimination under nonemptiness always leads to an exponential blowup in degree. This result gives no information regarding the length of expressions. For future work, it could be interesting to establish lower bounds on the length of expressions after eliminating converse under nonemptiness.

In all of the above, it could be interesting to add two other derived operators to our graph query fragments called the residuals [36]. It expresses a natural form of universal quantification and its expressive power relative to other operators is largely unexplored. We also do not know much about basic reasoning tasks, such as deciding satisfiability or subsumption, in the basic algebra extended with residuals.

9.1 Open Questions

The following list contains a selection of interesting open problems in this thesis:

- Is $\mathrm{UCQ}^{\subseteq}$ closed under conjunction?
- Is $F^{\subseteq}$ closed under conjunction for fragments F that include $\overline{\pi}$?
- Are monotone containments in $\{\mathsf{di}, {}^{-1}, {}^{+}\}^{\subseteq}$ captured by nonemptiness expressions in $\{\mathsf{di}, {}^{-1}, {}^{+}\}^{\neq\emptyset}$?
- Is $\{\pi\}^{\neq\emptyset}$ subsumed by $\{\mathsf{di}, {}^{-1}, {}^{+}\}^{\subseteq}$?
- Find a lower bound on the length of expressions after eliminating converse.

9.2 Future work

In this section we discuss the future of this project. In our framework, we have proven results for well established languages such as CQs and navigational graph languages. Obviously, these are not the only interesting query languages. For example, one can use our framework to investigate Boolean queries constructed from relational algebra or Datalog fragments. We hope that our framework will serve as a template to investigate Boolean queries for a wide array of query languages.

Another direction for this project could be to consider other sets of modalities. The base modalities we considered stem from natural/practical problems we want answered. We realize, however, that there can be other natural modalities motivated by other practical settings. For example, Barwise and Cooper [11] consider the modality $e_1 \cap e_2 \neq \emptyset$, corresponding to the language construct "some e_1 are e_2". Even when one considers other base modalities, our framework can still serve as a guideline for the investigation of Boolean queries. However, one needs to be careful in this setting, as there are an infinitude of base modalities one can consider.

We thus hope that our framework will serve as a starting point to investigate Boolean queries in general, regardless of the specific application.

10
Dutch Summery

Wanneer een relationele database gequeryd wordt, is het resultaat normaal gezien een relatie. Dit zijn echter niet de enige interessante queries, veel interessante queries verwachten immers een ja/nee-antwoord. We kunnen bijvoorbeeld vragen “Is student 14753 ingeschreven voor het vak c209?” of “Is er een vak dat geen schriftelijk examen heeft?”. Dergelijke queries worden *Booleaanse queries* genoemd.

In database theorie en eindige model theorie is het standaard om deze Booleaanse queries uit te drukken door middel van de *nonemptiness modaliteit*. Aan de hand van deze modaliteit worden Booleaanse queries uitgedrukt met expressies van de vorm $e \neq \emptyset$, waarbij e een expressie is in een bepaalde querytaal. Hier wordt een niet-leeg queryresultaat geïnterpreteerd als *true* en een leeg query resultaat als *false*. De Booleaanse query “Is student 14753 ingeschreven voor het vak c209” wordt bijvoorbeeld uitgedrukt door het niet-leeg zijn van de query “Verzamel alle studenten met nummer 14753 die ingeschreven zijn voor het vak c209”. In de praktijk wordt de nonemptiness modaliteit gebruikt door de querytalen SPARQL (`ASK` P) en SQL (`EXISTS (`Q`)`).

De nonemptiness modaliteit is duidelijk niet de enige natuurlijke manier om Booleaanse queries uit te drukken. Een integrity constraint is bijvoorbeeld op een natuurlijke manier uitdrukbaar aan de hand van een query die slechte elementen in de database zoekt. De constraint is dan voldaan als de query geen slechte elementen in de database vindt en bijgevolg een leeg resultaat geeft. Hier gebruiken we dus de *emptiness modaliteit* waarbij een leeg resultaat geïnterpreteerd wordt als *true* en een niet-leeg resultaat als *false*. In de praktijk wordt de emptiness modaliteit gebruikt in SQL door middel van `NOT EXISTS (Q)`.

Een andere natuurlijke modaliteit is de *containment* modaliteit. Via deze modaliteit worden Booleaanse queries uitgedrukt met expressies van de vorm $e_1 \subseteq e_2$ waarbij e_1 en e_2 twee query-expressies zijn in een bepaalde querytaal. De query $e_1 \subseteq e_2$ is *true* voor een database D als $e_1(D)$ een deelverzameling is van $e_2(D)$.[1] Bijvoorbeeld, de foreign-key constraint “elke student ingeschreven voor het vak c209, moet geslaagd zijn voor c106’ wordt uitgedrukt door de containment $e_1 \subseteq e_2$ waarbij e_1 de studenten verzamelt die ingeschreven zijn voor c209 en e_2 degenen verzamelt die geslaagd waren voor c106.

Dit voorbeeld laat ons ook de sterkte zien van de containment modaliteit. Containments geven ons namelijk de kracht om niet-monotone queries uit te drukken met monotone queries. Monotone queries Q zijn queries waarbij het resultaat enkel kan groeien: als $D \subseteq D'$ dan is $Q(D)$ vervat in $Q(D')$.

Als een querytaal krachtig genoeg is, zoals bijvoorbeeld eerste-order logica, dan zijn al deze modaliteiten even krachtig. Dit wil zeggen dat we precies dezelfde ja/nee-vragen kunnen stellen. Bijvoorbeeld, $\{\bar{x} \mid \varphi(\bar{x})\} = \emptyset$ is equivalent met $\{() \mid \neg\exists\bar{x}\, \varphi(\bar{x})\} \neq \emptyset$. Eveneens is $\{\bar{x} \mid \varphi_1(\bar{x})\} \subseteq \{\bar{x} \mid \varphi_2(\bar{x})\}$ equivalent met $\{() \mid \forall\bar{x}(\varphi_1 \to \varphi_2)(\bar{x})\} \neq \emptyset$.

Desondanks kan de keuze van de modaliteit toch belangrijk zijn voor efficiëntie en gebruikersgemak. Een functionele afhankelijkheid $A \to B$ op een relatie $R(A, B)$ kan direct uitgedrukt worden door middel van de emptiness modaliteit:

$$\{(a, b1, b2) \mid R(a, b1) \wedge R(a, b2) \wedge b1 \neq b2\} = \emptyset.$$

Op basis van de nonemptiness modaliteit is dit echter niet mogelijk als we enkel monotone queries, zoals bijvoorbeeld conjunctive queries (CQ), toelaten. Een analoge situatie doet zich voor bij foreign-key constraints. Deze kunnen namelijk eenvoudig gedefinieerd worden aan de hand van containment expressies. Met de nonemptiness en emptiness modaliteiten kunnen we dergelijke constraints echter niet uitdrukken als we enkel monotone expressies toelaten.

Wij vinden het dus zeker nuttig om te onderzoeken hoe deze modaliteiten zich verhouden ten opzichte van elkaar.

---

[1] Merk op dat expressies van de vorm $e_1 \subseteq e_2$ niet verward mogen worden met het bekende containment probleem, waarbij men geïnteresseerd is in het geval waarbij $e_1(D)$ vervat zit in $e_2(D)$ voor *alle* databases D. Containment expressies waarbij e_1 volledig vervat zit in e_2 voor alle database D, zijn niet interessant als Booleaanse queries aangezien de query dan altijd *true* oplevert.

In deze thesis introduceren we een kader om Booleaanse queries te onderzoeken. Al onze resultaten passen in dit kader.

In het eerste thema fixeren we de querytaal en vergelijken we de modaliteiten. In deze thesis identificeren we enkele cruciale query operatoren die het mogelijk maken om van de ene modaliteit naar de andere te gaan. De verschil operator geeft ons bijvoorbeeld de mogelijkheid om van de containment modaliteit naar de emptiness modaliteit te gaan. Uiteraard willen we graag weten of die operatoren wel degelijk de operatoren zijn die we altijd nodig hebben. Hiervoor hebben we negatieve resultaten nodig die aantonen dat we een Booleaanse query kunnen uitdrukken met een bepaalde modaliteit, maar niet met een andere als bepaalde operatoren niet aanwezig zijn. Indien we geen restricties aan querytalen opleggen is dit echter niet mogelijk. We kunnen immers zeer pathologische querytalen constructuren. Als alternatief identificeren we bepaalde semantische eigenschappen van verzamelingen van queries, zoals bijvoorbeeld monotoniciteit, die het ontbreken van bepaalde operatoren reflecteren. We tonen bijvoorbeeld aan dat er geen gemeenschappelijke queries uitdrukbaar zijn in de emptiness en nonemptiness modaliteiten als we enkel monotone queries toelaten. Hierna passen we al onze resultaten toe op bekende querytalen zoals CQs en navigationale graaf querytalen.

In het tweede thema vergelijken we een vaste modaliteit onder verschillende querytalen. Dit soort vergelijking is interessant voor querytalen met veel verschillende operatoren. Zo kunnen we de invloed van de operatoren op de expressieve kracht bepalen voor een vaste modaliteit. Voor navigationele graaf talen is de volledige expressieve kracht al gekend onder de nonemptiness en emptiness modaliteiten [21]. In deze thesis brengen we de volledige expressieve kracht voor de navigationele graaf talen in kaart onder de containment modaliteit. We tonen in het bijzonder aan dat alle operatoren kracht toevoegen, tenzij ze letterlijk geconstrueerd kunnen worden. Dit verschilt drastisch met de expressieve kracht voor deze talen onder de nonemptiness modaliteit. Onder de nonemptiness modaliteit kunnen we namelijk in enkele gevallen de inverse operator wegwerken. Dit proces wordt ook inverse eliminatie genoemd. In deze thesis tonen we bovendien aan dat inverse eliminatie resulteert in een zeer complexe formule met exponentieel meer composities/projecties/coprojecties.

In het derde thema brengen we het eerste en tweede thema samen en vergelijken we verschillende modaliteiten onder verschillende querytalen. Net zoals bij het tweede thema, zijn talen met verschillende operatoren hier zeer geschikt voor. We vergelijken de drie modaliteiten voor alle verschillende navigationele graaftalen. Voor de meeste talen tonen we aan dat ze onvergelijkbaar zijn, tenzij de operatoren letterlijk geconstrueerd kunnen worden. In deze context zijn er echter enkele vragen open. Zo weten we bijvoorbeeld niet of alle Booleaanse queries die uitdrukbaar zijn door middel van projectie onder nonemptiness, ook uitdrukbaar zijn door middel van diversity (ongelijkheid) en inverse onder de containment modaliteit.

Merk op dat de nonemptiness en emptiness modaliteiten elkaars negatie zijn. Bijgevolg is de vergelijking van nonemptiness en emptiness voor een bepaalde querytaal $\mathcal{F}$ equivalent met de vraag of $\mathcal{F}$ onder de nonemptiness modaliteit gesloten is onder negatie. In het vierde thema bekijken we deze vraag voor de containment modaliteit. We tonen in het bijzonder aan dat CQ onder containment niet gesloten is onder negatie en navigationele talen enkel gesloten zijn onder negatie als de containment modaliteit geen extra kracht toevoegt over nonemptiness. We veralgemenen dit idee verder naar andere Booleaanse connectieven zoals conjunctie.

Bibliography

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[2] S. Abiteboul and V. Vianu. Regular path queries with constraints. *Journal of Computer and System Sciences*, 58:428–452, 1999.

[3] M. Ajtai and Y. Gurevich. Monotone versus positive. *J. ACM*, 34(4):1004–1015, October 1987.

[4] T.J. Ameloot, B. Ketsman, F. Neven, and D. Zinn. Weaker forms of monotonicity for declarative networking: A more fine-grained answer to the CALM-conjecture. *ACM Transactions on Database Systems*, 40(4):article 21, 2016.

[5] T.J. Ameloot, B. Ketsman, F. Neven, and D. Zinn. Datalog queries distributing over components. *ACM Transactions on Computational Logic*, 18(1):5:1–5:35, 2017.

[6] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. Reutter, and D. VrgočDatalog. Foundations of modern query languages for graph databases. *ACM Computing Surveys*, 50(5):68:1–68:40, September 2017.

[7] R. Angles, P. Barceló, and G. Rios. A practical query language for graph DBs. In L. Bravo and M. Lenzerini, editors, *Proceedings 7th Alberto Mendelzon International Workshop on Foundations of Data Management*, volume 1087 of *CEUR Workshop Proceedings*, 2013.

[8] R. Angles and C. Gutierrez. Survey of graph database models. *ACM Computing Surveys*, 40(1):article 1, 2008.

[9] A. Badia. *Quantifiers in Action: Generalized Quantification in Query, Logical and Natural Languages*, volume 37 of *Advances in Database Systems.* Springer, 2009.

[10] P. Barceló. Querying graph databases. In *Proceedings 32st ACM Symposium on Principles of Databases*, pages 175–188. ACM, 2013.

[11] J. Barwise and R. Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219, 1981.

[12] M. Benedikt, J. Leblay, B. ten Cate, and E. Tsamoura. *Generating Plans from Proofs: The Interpolation-based Approach to Query Reformulation*. Morgan&Claypool, 2016.

[13] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Reasoning on regular path queries. *SIGMOD Record*, 32(4):83–92, 2003.

[14] A.K. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings 9th ACM Symposium on the Theory of Computing*, pages 77–90. ACM, 1977.

[15] C.C. Chang and H.J. Keisler. *Model Theory*. North-Holland, 3rd edition, 1990.

[16] E. Codd. A relational model for large shared databanks. *Communications of the ACM*, 13(6):377–387, 1970.

[17] I.F. Cruz, A.O. Mendelzon, and P.T. Wood. A graphical query language supporting recursion. In *ACM SIGMOD Record*, volume 16, pages 323–330. ACM, 1987.

[18] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, second edition, 1999.

[19] G.H.L. Fletcher, M. Gyssens, D. Leinders, D. Surinx, J. Van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu. Relative expressive power of navigational querying on graphs. *Information Sciences*, 298:390–406, 2015.

[20] G.H.L. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, and S. Vansummeren. Similarity and bisimilarity notions appropriate for characterizing indistinguishability in fragments of the calculus of relations. *Journal of Logic and Computation*, 25(3):549–580, 2015.

[21] G.H.L. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu. Relative expressive

power of navigational querying on graphs. In *Proceedings 14th International Conference on Database Theory*, 2011.

[22] G.H.L. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu. The impact of transitive closure on the expressiveness of navigational query languages on unlabeled graphs. *Annals of Mathematics and Artificial Intelligence*, 73(1–2):167–203, 2015.

[23] H. Garcia-Molina, J.D. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice Hall, 2009.

[24] Y. Gurevich. Toward logic tailored for computational complexity. In M.M. Richter et al., editors, *Computation and Proof Theory*, volume 1104 of *Lecture Notes in Mathematics*, pages 175–216. Springer-Verlag, 1984.

[25] P. Hell and J. Nesetril. *Graphs and Homomorphisms*. Oxford Lecture Series in Mathematics and Its Applications. OUP Oxford, 2004.

[26] L. Hella, K. Luosto, and J. Väänänen. The hierarchy theorem for generalized quantifiers. *The Journal of Symbolic Logic*, 61(3):802–817, 1996.

[27] J. Hellings. Unpublished notes, Hasselt University, 2016.

[28] L. Henkin and A. Tarski. Cylindric algebras. In R.P. Dilworth, editor, *Lattice Theory*, volume 2 of *Proceedings of Symposia in Pure Mathematics*, pages 83–113. American Mathematical Society, 1961.

[29] T. Imielinski and W. Lipski. The relational model of data and cylindric algebras. *Journal of Computer and System Sciences*, 28:80–102, 1984.

[30] Ph.G. Kolaitis. On the expressive power of logics on finite models. In *Finite Model Theory and Its Applications*, chapter 2. Springer, 2007.

[31] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[32] L. Libkin, W. Martens, and D. Vrgoč. Quering graph databases with XPath. In *Proceedings 16th International Conference on Database Theory*. ACM, 2013.

[33] R.D. Maddux. *Relation Algebras*. Elsevier, 2006.

[34] M. Marx and M. de Rijke. Semantic characterizations of navigational XPath. *SIGMOD Record*, 34(2):41–46, 2005.

[35] M. Marx and Y. Venema. *Multi-Dimensional Modal Logic*. Springer, 1997.

[36] V. Pratt. Origins of the calculus of binary relations. In *Proceedings 7th Annual IEEE Symposium on Logic in Computer Science*, pages 248–254, 1992.

[37] B. Rossman. Homomorphism preservation theorems. *Journal of the ACM*, 55(3):15:1–15:53, August 2008.

[38] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *J. ACM*, 27(4):633–655, 1980.

[39] A.P. Stolboushkin. Finitely monotone properties. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science*, LICS '95, pages 324–, Washington, DC, USA, 1995. IEEE Computer Society.

[40] D. Surinx, J. Van den Bussche, and D. Van Gucht. The primitivity of operators in the algebra of binary relations under conjunctions of containments. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10, June 2017.

[41] D. Surinx, G.H.L. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu. Relative expressive power of navigational querying on graphs using transitive closure. *Logic Journal of the IGPL*, 23(5):759–788, 2015.

[42] B. ten Cate and M. Marx. Navigational XPath: Calculus and algebra. *SIGMOD Record*, 36(2):19–26, 2007.

[43] J. Van den Bussche. Applications of Alfred Tarski's ideas in database theory. In L. Fribourg, editor, *Computer Science Logic*, volume 2142 of *Lecture Notes in Computer Science*. Springer, 2001.

[44] P.T. Wood. Query languages for graph databases. *SIGMOD Record*, 41(1):50–60, March 2012.