2012•2013
FACULTEIT WETENSCHAPPEN
master in de informatica: databases

Masterproef
Navigational Query Languages

Promotor :
Prof. dr. Jan VAN DEN BUSSCHE

Copromotor :
dr. Robert BRIJDER
De heer Jelle HELLINGS

Dimitri Surinx
Masterproef voorgedragen tot het bekomen van de graad van master in de informatica , afstudeerrichting databases

2012•2013
FACULTEIT WETENSCHAPPEN
master in de informatica: databases

Masterproef
Navigational Query Languages

Promotor :
Prof. dr. Jan VAN DEN BUSSCHE

Copromotor :
dr. Robert BRIJDER
De heer Jelle HELLINGS

Dimitri Surinx
Masterproef voorgedragen tot het bekomen van de graad van master in de informatica , afstudeerrichting databases

Preface

This is the ideal opportunity to thank the people without whom this thesis would never have been possible.

First and foremost, I want to thank my advisor, Prof. dr. Jan Van den Bussche, for his continuing guidance throughout the last two years. He was always prepared to help me when I was stuck, even when he had very little time. I really enjoyed our numerous meetings, in which he always was very enthusiastic. Furthermore, in these meetings he not only helped me to gain a better understanding in my thesis topic, but also tried to improve my English writing.

On the other hand I would like to thank my fellow students Bas Ketsman, Balazs Nemeth and Niki Vandesbosch for all the interesting discussions throughout the years.

Last but not least, I would like to thank my parents and my friends for the continuing guidance and support during my studies.

Abstract

In this thesis we introduce navigational query languages on graphs. Path queries in our languages are built over several operators: identity, union, composition, projection, coprojection, converse, transitive closure, diversity, intersection and difference. The smallest language we will consider only contains the first 3 operators, while the largest language contains all operators. For these query languages we will characterize their complete relative expressive power, i.e., we will compare the expressive power of languages containing different selections of operators.

In these query languages we will also model boolean queries by associating nonempty query results with *true* and empty query results with *false*. As for path queries, we will for these boolean queries characterize the complete relative expressive power of our languages. On the other hand, we will also consider other approaches to model boolean queries in our languages and characterize their expressive power.

Contents

1 Introduction

Throughout the years a wide variety of query languages have been introduced and studied extensively. Among them are query languages we have studied thoroughly in our curriculum, e.g., relational algebra and XPATH. In this thesis we will study navigational query languages on *labeled* graphs. These languages are relevant in practice since graph data is found in RDF and everywhere on the web.

In our navigational query languages we consider a selection of unary and binary operators which map graphs to graphs. The unary operators are: projection, coprojection, converse, transitive closure, identity, diversity, and an operation to retrieve edges with a certain label. On the other hand, the binary operators are: union, intersection, composition and difference. The semantics of these operators are comparable to the ones defined on binary relations. The largest language we will consider contains all operators, while the smallest language only contains identity, union and composition. Similar to standard database query languages, our queries are built recursively over a selection of these operators. Furthermore, since these operators map graphs to graphs, it is clear that queries in our languages map graphs to graphs as well. From now on, we will refer to these queries as *path* queries.

A natural question for our new query languages is how one language relates to another another language in terms of expressive power. For example we want to know whether adding converse and coprojection to a certain query language allows us to express more path queries. In Chapter 3 we will develop several techniques which will be used extensively towards answering this question in Chapter 4.

We will also consider *boolean* queries, i.e., queries which map graphs to *true* or *false*. We can express these boolean queries using our new query languages by identifying a nonempty query result with *true* and an empty result as *false*. For these kind of queries the same question arises as for path queries, e.g., we want to know whether adding diversity and projection to a certain language allows us to express more boolean queries. In Chapter 5 we will provide an answer to this question.

If a boolean or path query is not expressible in a certain language, it might be possible that the query is expressible in that language when we only consider unlabeled input graphs. Hence the expressive power of languages could change when only we consider unlabeled graphs and thus also their expressive power relative to other languages. In Chapter 6 we will show that the relative expressiveness changes between certain languages at the level of boolean queries on unlabeled graphs.

To express boolean queries in our languages we associated a nonempty query result with *true* and an empty result with *false*. This is the standard method for expressing boolean queries. We could, however, express boolean queries in a different manner. For example, we can associate *true* with empty and *false* with nonempty. On the other hand, we could also, for example, associate a boolean query q to every pair of path queries e_1, e_2 by setting $q(G) = \mathit{true}$ if and only if $e_1(G) \subseteq e_2(G)$. In Chapter 7 we will investigate the (relative) expressive power at the level of boolean queries for these two methods[1] of expressing boolean queries in our languages.

[1]We will refer to these methods as modalities.

2 Preliminaries

In this chapter we will formally introduce path and boolean queries. We will also derive some straightforward propositions related to the relative expressive power at the level of path and boolean queries. These propositions will be used extensively in the remainder of this thesis. Furthermore, we will also introduce conjunctive queries with nonequalities.

2.1 Path queries

In this thesis we want to introduce a query language to navigate over graphs. Hence we first need a formal definition for graphs.

Definition 2.1: Let $\Lambda = \{R_1, \dots, R_n\}$ be a finite nonempty set of edge labels. A graph is a relational structure $G = (V, G(R_1), \dots, G(R_n))$, where V is a set of vertices and $G(R_i) \subseteq V \times V$ is a binary relation which contains all edges with label R_i. $\diamond$

Remark 2.2: Notice that in the classical logic context, G is a structure over the signature Λ. $\diamond$

Note that by definition a graph can contain nodes which are not present in any edge relation. We want that these nodes are absent from our query results, since we only want to include nodes in our query result to which we can navigate. To achieve this, we consider an active domain construction similar to the active domain construction for the relational calculus[Gys12]. Let us define the active domain of a graph G as the set of all vertices occurring in one of the edge relations of G. More formally,

$$\operatorname{adom}(G) = \{m \mid \exists n, \exists R : (m, n) \in G(R) \vee (n, m) \in G(R)\}.$$

A *path* query e is a function, which takes any graph G over Λ as its input and outputs a binary relation $e(G) \subseteq \operatorname{adom}(G) \times \operatorname{adom}(G)$. On the other hand, a *boolean* query is a function which takes any graph G over Λ as its input and outputs either *true* or *false*.

In the remainder of this thesis, every language for navigating over graphs contains expressions build over the following operations: $G(R)$, id, $\emptyset$, union ($e_1 \cup e_2$) and composition ($e_1 \circ e_2$).

Definition 2.3: Define $\mathcal{N}$ to be the most basic algebra for navigating over graphs. Its expressions are built recursively over $G(R)$, id, $\emptyset$, union ($e_1 \cup e_2$) and composition ($e_1 \circ e_2$), which are defined as follows:

$$\begin{aligned}
R(G) &= G(R);\\
\emptyset(G) &= \emptyset;\\
id(G) &= \{(m,m) \mid m \in \mathrm{adom}(G)\};\\
e_1 \circ e_2(G) &= \{(m,n) \mid \exists p : (m,p) \in e_1(G) \wedge (p,n) \in e_2(G)\};\\
e_1 \cup e_2(G) &= e_1(G) \cup e_2(G);\\
e^k(G) &= \underbrace{e \circ \ldots \circ e}_{k \text{ times}}(G).
\end{aligned}$$

◇

Remark 2.4: Notice that the expressions in $\mathcal{N}$ depend on the given label set Λ where input graphs are built over, hence we will sometimes write $\mathcal{N}_\Lambda$ when the signature is of importance. If no set of labels is given explicitly, Λ is assumed to be some arbitrary signature. ◇

We can extend the basic algebra by adding a selection of the following operations: diversity (di), converse (e^{-1}), intersection ($e_1 \cap e_2$), difference ($e_1 \setminus e_2$), projections (π_1 and π_2), coprojections ($\overline{\pi}_1$ and $\overline{\pi}_2$) and transitive closure (e^+). These operations are defined as follows:

$$\begin{aligned}
di(G) &= \{(m,n) \mid m \neq n \wedge m,n \in \mathrm{adom}(G)\};\\
e^{-1}(G) &= \{(m,n) \mid (n,m) \in e(G)\};\\
e_1 \cap e_2(G) &= e_1(G) \cap e_2(G);\\
e_1 \setminus e_2(G) &= e_1(G) \setminus e_2(G);\\
\pi_1(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge (\exists n)(m,n) \in e(G)\};\\
\pi_2(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge (\exists n)(n,m) \in e(G)\};\\
\overline{\pi}_1(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge \neg(\exists n)(m,n) \in e(G)\};\\
\overline{\pi}_2(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge \neg(\exists n)(n,m) \in e(G)\};\\
e^+(G) &= \bigcup_{k \geq 1} e^k(G).
\end{aligned}$$

Usually, we refer to the operations in $\mathcal{N}$ as the *basic features* and the extensions as the *nonbasic features.*

If F is a set of nonbasic features, we denote $\mathcal{N}(F)$ as the language obtained by augmenting all features of F to $\mathcal{N}$.

Remark 2.5: In this thesis we will not consider extensions which only contain one of the projections (resp. coprojections). $\diamond$

Notice that all the operations apart from the transitive closure are defined in a first order fashion, hence we have the following proposition which will be used several times throughout this thesis to establish inexpressibility of queries in certain languages.

Proposition 2.6: *Let F be a set of nonbasic features such that $^+ \not\in \overline{F}$. If $e \in \mathcal{N}(F)$, then there exists a first order formula $\varphi(x, y)$ such that $(a, b) \in e(G)$ if and only if $G \models \varphi[a, b]$.*

Definition 2.7: A path query q is expressible in a language $\mathcal{N}(F)$, if there exists some expression $e \in \mathcal{N}(F)$ such that $e \equiv q$, i.e., $e(G) = q(G)$ for every graph G over Λ.

A boolean query q is expressible in $\mathcal{N}(F)$, if there exists some expression $e \in \mathcal{N}(F)$ such that for every graph G, $q(G) = \mathit{true}$ if and only if $e(G)$ is nonempty. $\diamond$

Example 2.8: The query $\mathit{id} \setminus (\pi_2((\mathit{id} \setminus R^+) \circ (\mathit{di} \cup \mathit{id})))$ expresses the boolean query connectivity.

The query $(R \setminus \mathit{id})^+ \cap \mathit{id}$ expresses whether the input graph has a cycle. $\diamond$

If every path query expressible in $\mathcal{N}_\Lambda(F_1)$ is also expressible in $\mathcal{N}_\Lambda(F_2)$, we will write $\mathcal{N}_\Lambda(F_1) \leq^{\text{path}} \mathcal{N}_\Lambda(F_2)$. Analogously, for boolean queries, we will write $\mathcal{N}_\Lambda(F_1) \leq^{\text{bool}} \mathcal{N}_\Lambda(F_2)$. Conversely, we will write $\mathcal{N}_\Lambda(F_1) \not\leq^{\text{path}} \mathcal{N}_\Lambda(F_2)$ (resp. $\not\leq^{\text{bool}}$) if there exists some path query (resp. boolean query) expressible $\mathcal{N}_\Lambda(F_1)$ which is not expressible in $\mathcal{N}_\Lambda(F_2)$.

More generally, if $\mathcal{N}_\Lambda(F_1) \leq^{\text{path}} \mathcal{N}_\Lambda(F_2)$ for every label set Λ, we will write $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$. Analogously, for boolean queries, we will write $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$. Conversely, if one label set Λ exists such that $\mathcal{N}_\Lambda(F_1) \not\leq^{\text{path}} \mathcal{N}_\Lambda(F_2)$, we will write $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$. Analogously, for boolean queries, we will write $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$.

Proposition 2.9: *If $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$, then $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$.*

Proof: Suppose that e is a boolean query expressible in $\mathcal{N}(F_1)$, then there exists an expression $q \in \mathcal{N}(F_1)$, such that for every graph G, $e(G)$ is true if and only if $q(G)$ is nonempty. Since q considered as a path query is also expressible in $\mathcal{N}(F_2)$ by hypothesis, e is also expressible in $\mathcal{N}(F_2)$. $\blacksquare$

As will be clear later on in this thesis, the converse of the previous proposition is not necessarily true.

Definition 2.10: A language $\mathcal{N}(F_1)$ is said to be strongly separable from the language $\mathcal{N}(F_2)$ at the level of path queries, written $\mathcal{N}(F_1) \leq^{\text{path}}_{\text{strong}} \mathcal{N}(F_2)$, if there exists a path query q expressible in $\mathcal{N}(F_1)$ and a finite graph G, such that for every expression $e \in \mathcal{N}(F_2)$, we have $q(G) \neq e(G)$. $\diamond$

At the level of boolean queries, we cannot use an analogous definition due to following argument. Let q be a boolean query expressible in $\mathcal{N}(F_1)$ and assume that $q(G) \neq \emptyset$ for some graph G. Also, assume that for every $e \in \mathcal{N}(F_2)$, $e(G) = \emptyset$, in particular $id(G) = \emptyset$. However, since $id(G) = \emptyset$ if and only if $\text{adom}(G) = \emptyset$, $q(G)$ is also empty, which contradicts our assumption.

Definition 2.11: A language $\mathcal{N}(F_1)$ is said to be strongly separable from the language $\mathcal{N}(F_2)$ at the level of boolean queries, written $\mathcal{N}(F_1) \leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$, if there exists two graphs which can be distinguished by an expression in $\mathcal{N}(F_1)$, but are indistinguishable in $\mathcal{N}(F_2)$. That is, there exists graphs G_1 and G_2 and a boolean query q expressible in $\mathcal{N}(F_1)$, such that $q(G_1) = \textit{true}$ and $q(G_2) = \textit{false}$, and for every $e \in \mathcal{N}(F_2)$, $e(G_1)$ is nonempty if and only if $e(G_2)$ is nonempty. ◇

2.2 Conjunctive queries

Definition 2.12: An atom is an expression of the form $R(x_1, \ldots, x_n)$ where R is a relation name and $x_1, \ldots, x_n$ are terms. A term is either a variable or a constant. ◇

Definition 2.13: A database is a set of atoms which only contain constants (these atoms are also known as facts). ◇

Definition 2.14: A conjunctive query with nonequalities is a 3-tuple $Q = (H, B, \textit{noneq})$, where

- The body B is a set of atoms;
- $\textit{noneq}$ is a set of nonequalities built over variables present in H or B;
- The head H is a tuple of variables. ◇

Example 2.15: The following two queries are conjunctive queries with nonequalities.

$$\begin{aligned} Q_1 &= ((x, y), \{R(x, z), R(z, y)\}, \{x \neq z, z \neq y\}) \\ Q_2 &= ((x), \{R(x, x), R(x, y), R(y, z)\}, \{x \neq y\}) \end{aligned}$$ ◇

For simplicity, we will also use the datalog notation for conjunctive queries, which we will introduce with an example.

Example 2.16: The queries of Example 2.15 in datalog notation:

$$\begin{aligned} Q_1(x, y) &\leftarrow R(x, z), R(z, y), x \neq z, z \neq y \\ Q_2(x) &\leftarrow R(x, x), R(x, y), R(y, z), x \neq y \end{aligned}$$ ◇

To define the semantics of conjunctive queries, we first need some machinery to assign values from a database to variables of a query.

Definition 2.17: A substitution f from a conjunctive query $Q = (H, B, \mathit{noneq})$ into a database D, written $f : Q \to D$, is a function that maps every variable of Q onto the active domain of D. $\diamond$

These substitutions can also be applied to atoms by applying f onto its terms, i.e., $f(R(x_1, \ldots, x_n)) = R(f(x_1), \ldots, f(x_n))$. Similarly, one can also apply f to a set of atoms, by applying f to every atom in that set.

Definition 2.18: A substitution f from a conjunctive query $Q = (H, B, \mathit{noneq})$ into a database D is called a matching if it has the following two properties:

– If $x \neq y \in \mathit{noneq}$ then $f(x) \neq f(y)$ in D;

– $f(B) \subseteq D$ (The homomorphism property). $\diamond$

Now, we are ready to define the semantics of conjunctive queries. Suppose that $Q = (H, B, \mathit{noneq})$ is a conjunctive query, and suppose that D is a database, then the result of applying Q to D, denoted by $Q(D)$ is defined as

$$Q(D) = \{f(H) \mid f \text{ is a matching from } Q \text{ in } D\}.$$

A partition π of a set V induces an equivalence class denoted by $[x]_\pi$ for every $x \in V$. The set of all these equivalence classes is denoted by V/π and is called the quotient of V by π.

Analogous to the 'construction' of V/π, one can, given a partition of the variables of a conjunctive query with nonequalities $Q = (H, B, \mathit{noneq})$, construct a 'quotient' query $Q/\pi = (H/\pi, B/\pi, \mathit{noneq}/\pi)$ by replacing each variable in Q by its equivalence class.

Definition 2.19: Let V be a set of variables and let *noneq* be a set of nonequalities over V. We say that a partition π of V is compatible with *noneq* if $x \neq y \in \mathit{noneq}$ implies $[x]_\pi \neq [y]_\pi$. $\diamond$

Definition 2.20: Let Q be a conjunctive query with nonequalities. We call a partition π of $\mathit{vars}(Q)$ *legal* for Q if π is compatible with *noneq*. $\diamond$

Definition 2.21: A homomorphism f from a conjunctive query $Q_1 = (H_1, B_1, \mathit{noneq}_1)$ to a conjunctive query $Q_2 = (H_2, B_2, \mathit{noneq}_2)$, written $f : Q_1 \to Q_2$, is a function which maps the variables from Q_1 onto variables of Q_2 such that $f(H_1) = H_2$ and $f(B_1) \subseteq B_2$. $\diamond$

Note that such a homomorphism also induces a homomorphism of relations from B_1 to B_2, which we will denote by $f : B_1 \to B_2$.

Theorem 2.22: *Let $Q_1 = (H_1, B_1, \mathit{noneq}_1)$ and $Q_2 = (H_2, B_2, \mathit{noneq}_2)$ be conjunctive queries with nonequalities. Then, $Q_1 \subseteq Q_2$ if and only if for every legal partition π for Q_1, there exists a homomorphism $h : Q_2 \to Q_1/\pi$.*

Proof: First, we will show the only if direction. Suppose that π is a valid partition of B_1. Since $x \mapsto [x]_\pi$ is a matching, we know that $H_1/\pi \in Q_1(B_1/\pi)$. Furthermore, since it is given that $Q_1 \subseteq Q_2$, we know that $H_1/\pi \in Q_2(B_1/\pi)$. Thus there exists a matching f such that $f(H_2) = H_1/\pi$ and $f(B_2) \subseteq B_1/\pi$, which by definition is a homomorphism from Q_2 to Q_1/π.

Conversely, we will now show the if direction. Let D be a database. If $t \in Q_1(D)$, then there exists a matching f of Q_1 in D such that $f(H_1) = t$. Now, we choose π such that $[x]_\pi = [y]_\pi$ if and only if $f(x) = f(y)$. Clearly, π is legal for Q_1 since f agrees with the inequalities of Q_1. Furthermore, due to our assumption, there exists a homomorphism $h : Q_2 \to Q_1/\pi$. Let us now define a substitution $g : Q_1/\pi \to D : [x]_\pi \mapsto f(x)$. Observe that this function is well defined. We will now show that $g \circ h$ is a matching of Q_2 in D such that $g \circ h(H_2) = t$. First, notice that $g(H_1/\pi) = f(H_1)$, hence

$$g \circ h(H_2) = g(H_1/\pi) = f(H_1) = t.$$

Moreover, $g(B_1/\pi) = f(B_1)$, hence

$$g \circ h(B_2) \subseteq g(B_1/\pi) = f(B_1) \subseteq D.$$

The only thing left to show is that $g \circ h$ is compatible with noneq_2. Suppose that $h(x) = [u]_\pi$ and that $h(y) = [v]_\pi$, where $u, v \in \mathit{vars}(Q_1)$. Then,

$$\begin{aligned} x \neq y \in \mathit{noneq}_2 &\implies [u]_\pi \neq [v]_\pi \\ &\implies f(u) \neq f(v) \\ &\implies g([u]_\pi) \neq g([v]_\pi) \\ &\implies g \circ h(x) = g([u]_\pi) \neq g([v]_\pi) = g \circ h(y) \end{aligned}$$

which concludes our proof. ■

In the remainder of this thesis we will only consider conjunctive queries with bodies which only contain binary relations.

Definition 2.23: The graph of a conjunctive query with nonequalities Q is a graph G with two sets of edges, a set of directed edges and a set of undirected edges, labelled $\neq$ (also referred to as $\neq$-edges). The set of directed edges contains an edge (x, y) for every atom $R(x, y)$ in the body of Q. Furthermore, for each nonequality $x \neq y$ in Q, we add a $\neq$-edge to G. ◇

3 Separation techniques

The main goal of this thesis is to gain an understanding in the relative expressive power of certain languages, this at the level of boolean and path queries. To do so, we will try to separate languages from one another using techniques described in this chapter.

Notice that the contrapositive of Proposition 2.9 tells us that separation on the level of boolean queries implies separation on the level of path queries. Hence when trying to establish separation on the level of path queries, we should try to prove theorems which also establish separation on the level of boolean queries.

In this chapter we will describe some of the techniques used to establish separation.

3.1 Brute-force method

The main technique to establish separation between two languages $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$, is to find two graphs G_1 and G_2, and a query $q \in \mathcal{N}(F_1)$ such that $q(G_1)$ is true, $q(G_2)$ is false, and for every $q' \in \mathcal{N}(F_2)$: $q'(G_1) = \emptyset$ if and only if $q'(G_2) = \emptyset$. In other words, we say that the graphs G_1 and G_2 are distinguishable in $\mathcal{N}(F_1)$, but indistinguishable in $\mathcal{N}(F_2)$.

Notice that if G is finite, $e(G) \subseteq \text{adom}(G) \times \text{adom}(G)$ is also finite. Furthermore, there are only a finite number of subsets of $\text{adom}(G) \times \text{adom}(G)$ and hence there exists only a finite number of nonequivalent expressions when only considering G as input, i.e., $A = \{(e(G_1), e(G_2)) \mid e \in \mathcal{N}(F)\}$ is finite. Clearly, G_1 and G_2 are distinguishable in $\mathcal{N}(F)$ if and only if there exists a pair in A where the first component is empty and the second is not empty or vice versa. Hence computing A decides whether G_1 and G_2 are distinguishable in $\mathcal{N}(F)$. Computing A for a language $\mathcal{N}(F)$ and two graphs G_1 and G_2 can be done by Algorithm 3.1. Since the size of A is bounded, the algorithm is guaranteed to halt. However, since the size of A is exponential in the size of G_1 and G_2, the algorithm has an exponential worst case running time.

Algorithm 3.1 Brute-Force Algorithm

```
1: procedure Brute-Force($G_1, G_2, \mathcal{N}(F)$)
2:     $B \leftarrow \{id(G_1), id(G_2)\}$
3:     if $di \in F$ then
4:         $B \leftarrow B \cup \{di(G_1), di(G_2)\}$
5:     repeat
6:         for all binary operators $\otimes$ in $\mathcal{N}(F)$ do
7:             for all $(R_1, R_2), (S_1, S_2) \in B$ do
8:                 $B \leftarrow B \cup \{(R_1 \otimes S_1, R_2 \otimes S_2)\}$
9:         for all unary operators $\oplus$ in $\mathcal{N}(F)$ do
10:            for all $(R_1, R_2) \in B$ do
11:                $B \leftarrow B \cup \{(\oplus(R_1), \oplus(R_2))\}$
12:    until $B$ remains unchanged
13:    if $\exists (R_1, R_2) \in B$: ($R_1 = \emptyset$ and $R_2 \neq \emptyset$) or ($R_1 \neq \emptyset$ and $R_2 = \emptyset$) then
14:        return Distinguishable
15:    return Indistinguishable
```

3.2 Bisimulation

As mentioned in the previous section, the brute-force algorithm has an exponential worst case running time. Hence in some cases, the algorithm will not stop within a reasonable time. Also, we do not necessarily have that $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$ implies $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$. Therefore, in some cases we have to resort to other techniques to establish separation.

Another technique employs the idea that certain sets of expressions 'behave' in a similar manner for certain graphs. We will now formalize this.

Definition 3.1: A marked graph is a triple (G, a, b), where G is a graph and a, b are nodes in adom(G). ◇

Definition 3.2: Let $k \in \mathbb{N}$ 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 k, denoted $\mathbf{G_1} \simeq_k \mathbf{G_k}$, if the following 3 conditions are satisfied:

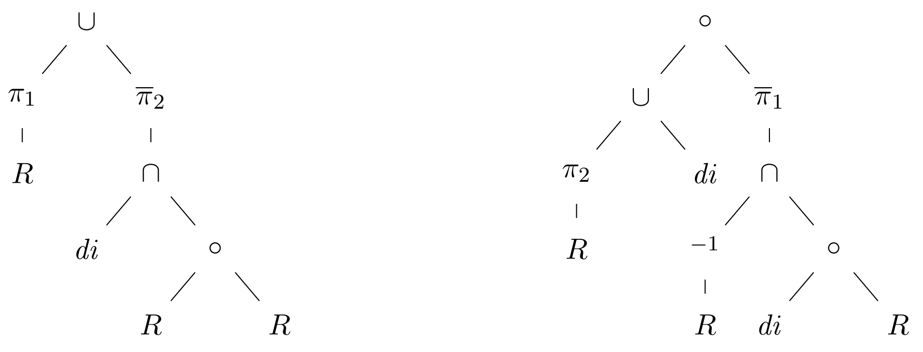
– **Atoms:** $a_1 = b_1$ iff $a_2 = b_2$; and $(a_1, b_1) \in G_1(R)$ iff $(a_2, b_2) \in G_2(R)$, for every $R \in \Lambda$;

– **Forth:** if $k > 0$, then, $\forall c_1 \in \text{adom}(G_1), \exists c_2 \in \text{adom}(G_2)$:

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

– **Back:** if $k > 0$, then, $\forall c_2 \in \text{adom}(G_2), \exists c_1 \in \text{adom}(G_1)$:

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



(a) Degree 2 (b) Degree 3

Figure 3.1: Expressions with a different degree.

As we will see later, bisimilar marked graphs behave in a similar way for certain expressions. To capture the class of such expressions we need the following definition.

Definition 3.3: The degree of an expression e is the maximum depth of nested applications of composition, transitive closure, projection and coprojection operations in e. That is, the maximum number of transitive closure, composition, projection and coprojection applications on a path from the root to a leaf in e's tree representation. More formally, we can define the degree inductively as follows,

$$\begin{aligned}
&\mathrm{degr}(id) = \mathrm{degr}(di) = \mathrm{degr}(\emptyset) = \mathrm{degr}(R) = 0 \qquad \text{for every } R \in \Lambda;\\
&\mathrm{degr}(e_1^{-1}) = \mathrm{degr}(e_1);\\
&\mathrm{degr}(\pi_i(e_1)) = \mathrm{degr}(\overline{\pi}_i(e_1)) = \mathrm{degr}(e_1^{+}) = 1 + \mathrm{degr}(e_1);\\
&\mathrm{degr}(e_1 \cup e_2) = \mathrm{degr}(e_1 \cap e_2) = \mathrm{degr}(e_1 \setminus e_2) = \max(\mathrm{degr}(e_1), \mathrm{degr}(e_2));\\
&\mathrm{degr}(e_1 \circ e_2) = \max(\mathrm{degr}(e_1), \mathrm{degr}(e_2)) + 1.
\end{aligned}$$

For a set of nonbasic features F, define $\mathcal{N}(F)_k$ as the expressions in $\mathcal{N}(F)$ with a degree of at most k. ◇

For example, in Figure 3.1a, the tree representation of the query

$$\pi_1(R) \cup \overline{\pi}_2(di \cap (R \circ R))$$

is displayed. This expression has degree 2. In Figure 3.1b the tree representation of query

$$(\pi_2(R) \cup di) \circ \overline{\pi}_1(R^{-1} \cap (di \circ R))$$

is shown. This expression has degree 3.

As mentioned in the beginning of this section, bisimilar marked graphs up to a certain degree behave in a similar manner for certain expressions. The following proposition tells us exactly how bisimilar graphs up to level k behave for expressions in $\mathcal{N}(\setminus, di)_k$. We also say that expressions in $\mathcal{N}(\setminus, di)$ of degree at most k are invariant under bisimulation up to level k.

Proposition 3.4: *Let k be a natural number; let $e \in \mathcal{N}(\setminus, di)_k$ and let $\mathbf{G_1} = (G_1, a_1, b_1)$ and $\mathbf{G_2} = (G_2, a_2, b_2)$ be marked graphs. If $\mathbf{G_1} \simeq_k \mathbf{G_2}$, then $(a_1, b_1) \in e(G_1) \iff (a_2, b_2) \in e(G_2)$.*

Proof: We will prove this proposition by induction on the structure of e.
Induction Basis:

- Let $e = R$, with $R \in \Lambda$. Then,

$$(a_1, b_1) \in R(G_1) \iff (a_2, b_2) \in R(G_2) \qquad \text{(see Definition 3.2)}$$

- Let $e = \emptyset$. Then, clearly, $\emptyset(G_1) = \emptyset = \emptyset(G_2)$.

- Let $e = id$. Then,

$$\begin{aligned} (a_1, b_1) \in id(G_1) &\iff a_1 = b_1 \\ &\iff a_2 = b_2 && \text{(see Definition 3.2(Atoms))} \\ &\iff (a_2, b_2) \in id(G_2) \end{aligned}$$

- Let $e = di$. Then,

$$\begin{aligned} (a_1, b_1) \in di(G_1) &\iff a_1 \neq b_1 \\ &\iff a_2 \neq b_2 && \text{(see Definition 3.2(Atoms))} \\ &\iff (a_2, b_2) \in di(G_2) \end{aligned}$$

Induction Hypothesis: Suppose for each subexpression e' of e that: if $\mathbf{G_1} \simeq_k \mathbf{G_2}$ with $k \geq \text{degree}(e')$, then $(a_1, b_1) \in e'(G_1) \iff (a_2, b_2) \in e'(G_2)$. **Induction Step:** We will now show that our proposition holds for e.

- Let $e = e_1 \cup e_2 \in \mathcal{N}(\setminus, di)_k$. Then,

$$\begin{aligned} (a_1, b_1) \in e_1 \cup e_2(G_1) &\iff (a_1, b_1) \in e_1(G_1) \vee (a_1, b_1) \in e_2(G_1) \\ &\iff (a_2, b_2) \in e_1(G_2) \vee (a_2, b_2) \in e_2(G_2) && \text{(hypothesis)} \\ &\iff (a_2, b_2) \in e_1 \cup e_2(G_1) \end{aligned}$$

- Let $e = e_1 \circ e_2 \in \mathcal{N}(\setminus, di)_k$. Then, since composition increases the degree of expressions, we know that $degree(e_1)$, $degree(e_2) < degree(e) \leq k$, which implies that $degree(e_1)$, $degree(e_2) \leq k - 1$.

  Assume that $(a_1, b_1) \in e_1 \circ e_2(G_1)$. Then, we know that there exists some p, such that $(a_1, p) \in e_1(G_1)$ and $(p, b_1) \in e_2(G_1)$. Furthermore, since $\mathbf{G_1} \simeq_k \mathbf{G_2}$, we know that for this p, there exists a q, such that $(G_1, a_1, p) \simeq_{k-1} (G_2, a_2, q)$ and

$(G_1, p, b_1) \simeq_{k-1} (G_2, q, b_2)$ (see Definition 3.2(Forth)). According to our hypothesis this implies that $(a_2, q) \in e_1(G_2)$ and $(q, b_2) \in e_2(G_2)$, hence $(a_2, b_2) \in e_1 \circ e_2(G_2)$.

Conversely, suppose that $(a_2, b_2) \in e_1 \circ e_2(G_2)$. Then there exists a q, such that $(a_2, q) \in e_1(G_2)$ and $(q, b_2) \in e_2(G_2)$. Furthermore, since $\mathbf{G_1} \simeq_k \mathbf{G_2}$, we know that for this q, there exists a p such that $(G_1, a_1, p) \simeq_{k-1} (G_2, a_2, q)$ and $(G_1, p, b_1) \simeq_{k-1} (G_2, q, b_2)$ (see Definition 3.2(Back)). Using our induction hypothesis, we know that $(a_1, p) \in e_1(G_1)$ and $(p, b_1) \in e_2(G_1)$, and hence $(a_1, b_1) \in e_1 \circ e_2(G_1)$.

– Let $e = e_1 \setminus e_2(G)$. Then,

$$\begin{aligned}(a_1, b_1) \in e_1 \setminus e_2(G) &\iff (a_1, b_1) \in e_1(G_1) \wedge (a_1, b_1) \notin e_2(G_1)\\ &\iff (a_2, b_2) \in e_1(G_2) \wedge (a_2, b_2) \notin e_2(G_2) \qquad \text{(hypothesis)}\\ &\iff (a_2, b_2) \in e_1 \setminus e_2(G_2)\end{aligned}$$ ■

The previous proposition can now be used to show that certain queries are not expressible in $\mathcal{N}(\setminus, di)_k$.

Proposition 3.5: *Let k be a natural number. A boolean query q is not expressible in $\mathcal{N}(\setminus, di)_k$, if there exist two graphs G_1 and G_2 such that $q(G_1)$ is true and $q(G_2)$ is false, and, for each pair $(a_1, b_1) \in \text{adom}(G_1)^2$, there exists $(a_2, b_2) \in \text{adom}(G_2)^2$, such that $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$.*

Proof: Suppose for the sake of contradiction that q is expressible in $\mathcal{N}(\setminus, di)_k$. Then, there exists an $e \in \mathcal{N}(\setminus, di)_k$ that expresses q.

It is given there exists two graphs G_1 and G_2 such that $q(G_1)$ is true and $q(G_2)$ is false. Since $q(G_1)$ is true, $e(G_1)$ is not empty and thus there exists $(a_1, b_2) \in e(G_1)$. Furthermore, it is also given that for (a_1, b_1) there exists $(a_2, b_2) \in \text{adom}(G_2)^2$, such that $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$. Now, using Proposition 3.4, we know that $(a_2, b_2) \in e(G_2)$, which implies that $e(G_2)$ is not empty and therefore contradicts our assumption. ■

Corollary 3.6: *A boolean query q is not expressible in $\mathcal{N}(\setminus, di)$, if for every natural number k, there exist two graphs G_1 and G_2 such that $q(G_1)$ is true and $q(G_2)$ is false, and, for each pair $(a_1, b_1) \in \text{adom}(G_1)^2$, there exists $(a_2, b_2) \in \text{adom}(G_2)^2$, such that $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$.*

Proof: Suppose for the sake of contradiction that q is expressible in $\mathcal{N}(\setminus, di)$. Then, there exists an $e \in \mathcal{N}(\setminus, di)$, that expresses q.

Clearly, $e \in \mathcal{N}(\setminus, di)_{deg(e)}$. Now, when we apply Proposition 3.5, we obtain a contradiction. ■

For more invariance results under bisimulation for other fragments see [FGL+12a].

Remember that intuitively, two marked graphs G_1 and G_2 are bisimilar up to a certain level $k+1$ when they behave the same way for expressions in $\mathcal{N}(\setminus, di)_{k+1}$. Then by the same intuitive reasoning, they also behave in a similar manner for expressions in $\mathcal{N}(\setminus, di)_k$. The following theorem justifies this intuition.

Lemma 3.7: *If* $(G_1, a_1, b_1) \simeq_{k+1} (G_2, a_2, b_2)$*, then* $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$

PROOF: Let $(G_1, a_1, b_1) \simeq_{k+1} (G_2, a_2, b_2)$. Then by 'back' condition, we know that for every $c_2 \in \text{adom}(G_2)$, there exists $c_1 \in \text{adom}(G_1)$, such that $(G_1, a_1, c_1) \simeq_k (G_2, a_2, c_2)$ and $(G_1, c_1, b_1) \simeq_k (G_2, c_2, b_2)$. Thus in particular for $c_2 = b_2$ there exists $c_1 \in \text{adom}(G_1)$, such that $(G_1, a_1, c_1) \simeq_k (G_2, a_2, b_2)$ and $(G_1, c_1, b_1) \simeq_k (G_2, b_2, b_2)$. Now by the atoms condition, $c_1 = b_1$ and thus $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$ as desired. ■

3.2.1 Connectivity is not expressible in $\mathcal{N}(\setminus, di)$

In this section we will show that the CONNECTIVITY query is not expressible in $\mathcal{N}(\setminus, di)$ to demonstrate the bisimilarity technique explained in Section 3.2. Remember that the CONNECTIVITY query expresses the directed connectivity property of a graph, i.e., it is the boolean query which outputs true if and only if there is a directed path between every pair of nodes.

We will first introduce some new notation which will be used extensively in this section.

Definition 3.8: Let G be an unlabeled graph. Define $\vec{d}_G(x, y)$ as the length of the shortest *directed* path in G from x to y; if no such path exists, $\vec{d}_G(x, y)$ is undefined. Furthermore, if $x = y$, then $\vec{d}_G(x, y) = 0$. If it is clear in which graph G we are expressing the distance we will omit the G subscript. ◇

Note that in the definition above $\vec{d}_G(x, y)$ is not necessarily equal to $\vec{d}_G(y, x)$, i.e., $\vec{d}_G$ is not the symmetric distance.

Definition 3.9: Let G be an unlabeled graph. We say that $(x, y) \in \text{adom}(G)^2$ is k-far in G, if $\vec{d}(x, y) > 2^k$. Conversely, $(x, y) \in \text{adom}(G)^2$ is k-close if it is not k-far and there is a path from x to y. ◇

Intuitively, we use this terminology because queries with a degree of at most k cannot distinguish a pair of vertices that lie too far from one another (farther than 2^k) and a pair of vertices where the distance is undefined.

Before we begin our proof we have to define a family of graphs which will be used extensively in our proof. Let n be an even natural number greater than 1. Define A_n to be the unlabeled graph with n vertices structured as displayed in Figure 3.2a and define B_n as the unlabeled graph with n vertices structured as displayed in Figure 3.2b, where both of B_n's connected components have exactly $n/2$ vertices.

The following lemma is mandatory for the proof of Lemma 3.11. Intuitively, it tells us that, no matter the positions of vertices a and b in A_n, we can always pick a vertex c, such that both (a, c) and (c, b) are far enough (k-far).

Lemma 3.10: *Let k be a natural number and let $n > 2 \cdot 2^{k+1}$. For every pair of vertices a_1, b_1 in A_n, there exists a vertex c_1 in A_n, such that both (a_1, c_1) and (c_1, b_1) are k-far.*

PROOF: We will split our proof up in several cases:

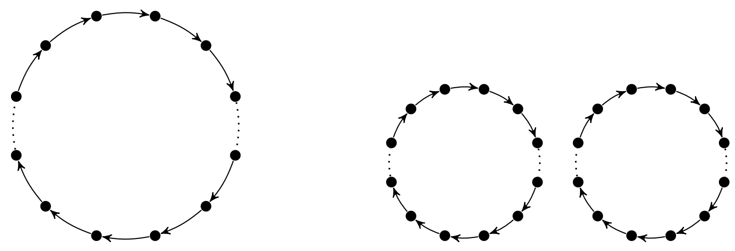

(a) Structure of graph A_n. (b) Structure of graph B_n.

Figure 3.2: Graphs used to establish inexpressibility of CONNECTIVITY in $\mathcal{N}(\backslash, di)$.

– If (a_1, b_1) is k-close, then we pick c_1 such that $\vec{d}(a_1, c_1) = 2^k + 1$. Let us now calculate $\vec{d}(c_1, b_1)$.

$$\begin{aligned}\vec{d}(c_1, b_1) &= n - \vec{d}(b_1, c_1) \\ &> 2 \cdot 2^{k+1} - (2^k + 1) \qquad \text{(since } n > 2 \cdot 2^{k+1} \text{ and } \vec{d}(b_1, c_1) \leq 2^k + 1) \\ &= 2^{k+1} + 2^k - 1 \\ &> 2^k + 1\end{aligned}$$

Thus, both (a_1, c_1) and (c_1, b_1) are k-far as desired.

– If (a_1, b_1) is k-far and the successor of b_1 is not a_1, then we pick c_1 such that a_1 is the successor of c_1. Now, $\vec{d}(c_1, b_1) = 1 + \vec{d}(a_1, b_1) > 2^k + 1$, thus (c_1, b_1) is k-far. Furthermore, $\vec{d}(a_1, c_1) > \vec{d}(a_1, b_1) > 2^k$, hence (a_1, c_1) is k-far.

On the other hand, if the successor of b_1 is a_1, then we pick c_1 such that $\vec{d}(a_1, c_1) = 2^k + 1$. Let us now calculate $\vec{d}(c_1, b_1)$.

$$\begin{aligned}\vec{d}(c_1, b_1) &= \vec{d}(a_1, b_1) - \vec{d}(a_1, c_1) \\ &= n - 1 - (2^k + 1) \\ &> 2 \cdot 2^{k+1} - 2^k - 2 \\ &> 2^k\end{aligned}$$

Thus, both (a_1, c_1) and (c_1, b_1) are k-far. ■

In the proof of the next lemma, we say that a vertex c is in *between* a and b if it is located on the shortest directed path from a to b. Conversely, we say that a vertex c is *not in between* a and b if it is on the shortest directed path from b to a.

Lemma 3.11: *Let k be a natural number and let n be an even natural number such that $n > 2 \cdot 2^k$.*

(a) If (a_1, b_1) is k-close in A_n and $a_2, b_2 \in B_n$ are located on the ***same*** *connected component of B_n such that $\vec{d}(a_1, b_1) = \vec{d}(a_2, b_2)$, then $(A_n, a_1, b_1) \simeq_k (B_n, a_2, b_2)$;*

(b) If (a_1, b_1) is k-far in A_n and (a_2, b_2) is k-far in B_n, then $(A_n, a_1, b_1) \simeq_k (B_n, a_2, b_2)$;

*(c) If (a_1, b_1) is k-far in A_n and $a_2, b_2 \in B_n$ are located on **different** connected components of B_n, then $(A_n, a_1, b_1) \simeq_k (B_n, a_2, b_2)$.*

PROOF: We will prove all three statements by induction on k simultaneously.
Induction Basis: Let $k = 0$. Then $n > 2 \cdot 2^0 = 2$. Furthermore, since $k = 0$, we only have to check the 'atoms' condition for bisimilarity.

(a) First, we will show that $a_1 = b_2$ iff $a_2 = b_2$.

$$\begin{aligned} a_1 = b_1 &\iff \vec{d}(a_1, b_1) = 0 \\ &\iff \vec{d}(a_2, b_2) = 0 \\ &\iff a_2 = b_2 \end{aligned}$$

Now, we will show that $(a_1, b_1) \in A_n$ iff $(a_2, b_2) \in B_n$.

$$\begin{aligned} (a_1, b_1) \in A_n &\iff \vec{d}(a_1, b_1) = 1 \\ &\iff \vec{d}(a_2, b_2) = 1 \\ &\iff (a_2, b_2) \in B_n \end{aligned}$$

(b) Since (a_1, b_1) and (a_2, b_2) are both k-far, we know that $a_1 \neq b_1$ and $a_2 \neq b_2$. Furthermore, both $\vec{d}(a_1, b_1)$ and $\vec{d}(a_1, b_1)$ are *strictly* greater then 1, which implies that $(a_1, b_1) \notin A_n$ and $(a_2, b_2) \notin B_n$.

(c) Since (a_1, b_1) is k-far in A_n, we know that $a_1 \neq a_2$ and $(a_1, b_1) \notin A_n$. Furthermore, it is clear that $a_2 \neq b_2$ and that $(a_2, b_2) \notin B_n$.

Induction Hypothesis: Suppose that our lemma holds for k.
Induction Step: We have to prove that our lemma holds for $k+1$. The atoms condition is verified in exactly the same way as in the proof of the induction basis, thus we only have to show that the 'forth' and 'back' conditions hold. First, we will show that 'forth' holds for (a), (b) and (c). To this end let $c_1 \in \text{adom}(A_n)$.
(a) We will split our proof up into several cases.

- Suppose that (a_1, c_1) is k-far in A_n and (c_1, b_1) is k-close in A_n. Let us pick c_2 on B_n such that $\vec{d}(c_2, b_2) = \vec{d}(c_1, b_1)$. Then, when we apply our induction hypothesis(a), it is clear that $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$.

  To prove that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$, it suffices to show that (a_2, c_2) is k-far in B_n (hypothesis(b)). We will consider two cases, each depending on the relative position of c_1 compared to a_1. If c_1 is in between a_1 and b_2, then $\vec{d}(a_2, c_2) = \vec{d}(a_1, c_1)$, and thus (a_2, c_2) is k-far, which according to induction hypothesis(b) implies that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$.

On the other hand, if c_1 does not lie between a_1 and b_1, then $0 < \vec{d}(c_1, a_1) \leq 2^k$. Remember that the connected component in B_n of a_2 and b_2 contains $n/2 > 2^{k+1}$ vertices. Now,

$$\begin{aligned}\vec{d}(a_2, c_2) &= n/2 - \vec{d}(c_2, a_2)\\ &= n/2 - \vec{d}(c_1, a_1)\\ &\geq n/2 - 2^k\\ &> 2^{k+1} - 2^k\\ &= 2^k.\end{aligned}$$

- If (a_1, c_1) is k-close in A_n and (c_1, b_1) is k-far, then the proof is analogous to the previous case, just swap the roles of a_1 and b_1.
- Suppose that (a_1, c_1) and (c_1, b_1) are both k-close. Then, c_1 is situated in between a_1 and b_1. If we choose c_2 on B_n in between a_2 and b_2 such that $\vec{d}(a_1, c_1) = \vec{d}(a_2, c_2)$ and $\vec{d}(c_1, b_1) = \vec{d}(c_2, b_2)$, then we can apply our induction hypothesis(a), which implies that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$ and $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$.
- Suppose that (a_1, c_1) and (c_1, b_1) are both k-far. Now, if we pick c_2 on the other connected component than the component of a_2 and b_2, we can apply our induction hypothesis(c), which tells us that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$ and $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$.

(b) First, notice that $\vec{d}(a_1, b_1) > 2^{k+1} = 2^k + 2^k$, hence both (a_1, c_1) and (c_1, b_1) cannot be k-close simultaneously.

- Suppose that (a_1, c_1) is k-far and (c_1, b_1) is k-close. Clearly, then c_1 is in between a_1 and b_1. Now, let us pick c_2 in B_n, such that $\vec{d}(c_1, b_1) = \vec{d}(c_2, b_2)$. Our hypothesis(a) now tells us that $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$.

  Furthermore, since (a_1, c_1) is k-far and since a_2 and c_2 are located on the same connected component, we need to show that (a_2, c_2) is also k-far, this, to be able to apply our induction hypothesis(b). Thus, we need to calculate $\vec{d}(a_2, c_2)$.

  $$\begin{aligned}\vec{d}(a_2, c_2) &= \vec{d}(a_2, b_2) - \vec{d}(c_2, b_2)\\ &> 2^{k+1} - 2^k && \text{(since } c_2 \text{ is in between } a_2 \text{ and } b_2\text{)}\\ &= 2^k\end{aligned}$$

  Thus, (a_2, c_2) is k-far in B_2 and hence $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$.
- If (a_1, c_1) is k-close and (c_1, b_1) is k-far, then the proof is analogous to the previous case.
- Suppose that both (a_1, c_1) and (c_1, b_1) are k-far. Then, if we pick c_2 to be on the other connected component than the component of a_2 and b_2, we can, according to our induction hypothesis(c), conclude that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$ and $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$.

(c) Again, $\vec{d}(a_1, c_1)$ and $\vec{d}(c_1, b_1)$ cannot be k-close simultaneously.

- Suppose that (a_1, c_1) is k-far and (c_1, b_1) is k-close. Now, let us choose c_2 on the connected component of b_2 such that $\vec{d}(c_1, b_1) = \vec{d}(c_2, b_2)$. Clearly, if we apply our induction hypothesis(a), then we can conclude that $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$. Moreover, since a_2 is located on a different connected component than b_2, we know that c_2 is also on a different component than a_2, which according to our induction hypothesis(c) implies that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$.
- If (a_1, c_1) is k-close and (c_1, b_1) is k-far, then the proof is analogous to the previous case.
- Suppose that both (a_1, c_1) and (c_1, b_1) are k-far. Then, we choose c_2 on the same connected component as a_2 such that (a_2, c_2) is k-far. Now, due to our induction hypothesis(b), we know that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$. Furthermore, since a_2 and b_2 are located on different connected components, c_2 and b_2 are located on different components. Hence $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$ (hypothesis(c)).

Let us now show that the 'back' condition holds for (a), (b) and (c). This, however, will be slightly more tricky and will require a more thorough analysis since c_2 can be picked on a different connected component than the component of both a_2 and b_2. Now let $c_2 \in \text{adom}(B_n)$.
(a) We will split our proof up into two cases:

- If c_2 is located on the same connected as a_2 and b_2, then again we have to split our proof up into two cases considering the relative position of c_2 compared to a_1 and b_2.
  - Suppose that $\vec{d}(a_2, c_2)$ is k-far and $\vec{d}(c_2, b_2)$ is k-close. Now, we pick c_1 in A_n such that $\vec{d}(c_1, b_1) = \vec{d}(c_2, b_2)$. Hence $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$ by our induction hypothesis(a). To prove that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$, by induction hypothesis(b), it suffices to show that (a_1, c_1) is k-far in A_n. To do so, we will consider two cases, each depending on the relative position of c_2 compared to a_2. If c_2 is in between a_2 and b_2, then because $\vec{d}(a_1, b_1) = \vec{d}(a_2, b_2)$, we know that $\vec{d}(a_1, c_1) = \vec{d}(a_2, c_2)$, hence $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$ (hypothesis(b)).
    On the other hand, if c_2 is not located between a_2 and b_2, then $0 < \vec{d}(c_2, a_2) \leq 2^k$. We will now calculate $\vec{d}(a_1, c_1)$. Remember that A_n has $n > 2 \cdot 2^{k+1}$ vertices. Now,

$$\begin{aligned}\vec{d}(a_1, c_1) &= n - \vec{d}(c_1, a_1) \\ &= n - \vec{d}(c_2, a_2) \\ &> 2 \cdot 2^{k+1} - \vec{d}(c_2, a_2) \\ &\geq 2 \cdot 2^{k+1} - 2^k \\ &> 2^k.\end{aligned}$$

  - If (a_2, c_2) is k-close and (c_2, b_2) is k-far or if (a_2, c_2) and (c_2, b_2) are both k-close, then the proof is analogous to the previous case.
  - If both (a_2, c_2) and (c_2, b_2) are k-far, then we pick c_2 such that both (a_1, c_1) and (c_1, b_1) are k-far, this c_2 exists due to Lemma 3.10. Now, our induction hypothesis(b) implies that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$ and $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$.

- If c_2 is not located on the same connected component as a_2 and b_2, then, we choose $c_1 \in A_n$, such that (a_1, c_1) is k-far and (c_1, b_1) is k-far; this c_1 exists due to Lemma 3.10. Now, using our induction hypothesis(c) we can conclude that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$ and $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$.

(b) Again, we will split our proof up into several cases:

- If c_2 is located on the same connected component as a_2 and b_2, then clearly both (a_2, c_2) and (c_2, b_2) cannot be k-close simultaneously.

  - Suppose that (a_2, c_2) is k-far and (c_2, b_2) is k-far. If the successor of b_1 does not equal a_1, then, we pick c_1 as the successor vertex of b_1. Clearly, both (a_1, c_1) and (c_1, b_1) are k-far. Now, when we apply our induction hypothesis(b), we know that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$.

    On the other hand, if the successor of b_1 does equal to a_1, then $\vec{d}(a_1, b_1) = 2 \cdot n - 1 > 2 \cdot 2^{k+1} - 1$. If we pick c_1 in between a_1 and b_1, such that $\vec{d}(a_1, c_1) = 2^{k+1} - 1$, then, clearly, (a_1, c_1) is k-far. Now, $\vec{d}(c_1, b_1) = \vec{d}(a_1, b_1) - \vec{d}(a_1, c_1) > 2 \cdot 2^{k+1} - 2^{k+1} = 2^{k+1}$, hence (c_1, b_1) is k-far.
  - Suppose that (a_2, c_2) is k-far and (c_2, b_2) is k-close. If we choose c_1, such that $\vec{d}(c_1, b_1) = \vec{d}(c_2, b_2)$, then since (a_1, b_1) is k-far, (a_1, c_1) is k-far. Now, our induction hypothesis(a,b) tells us that $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$ and $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$.

- If c_2 is not located on the same connected component as a_2 and b_2, then we pick c_1 on A_n in such a way that both $\vec{d}(a_1, c_1)$ and $\vec{d}(c_1, b_1)$ are k-far, this c_1 exists due to Lemma 3.10. Now, our induction hypothesis(c) implies that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$ and $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$.

(c) We have to consider several cases, each of which depend on the relative position of c_2 compared to a_2 or b_2. We will assume that c_2 belongs to the same component as a_2; the proof when c_2 belongs to the same component as b_2 is completely analogous.

- If (a_2, c_2) is k-close, then we pick c_1 on A_n, such that $\vec{d}(a_1, c_1) = \vec{d}(a_2, c_2)$. Now, since (a_1, b_1) is k-far, $\vec{d}(c_1, b_1) > 2^k$. Our induction hypothesis(a,c) now tells us that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$ and $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$.

- If (a_2, c_2) is k-far, then we pick c_1 on A_n such that both (a_1, c_1) and (c_1, b_1) are k-far, this c_1 exists due to Lemma 3.10. Now, our induction hypothesis(b,c) implies that $(A_n, a_1, c_1) \simeq_k (B_n, a_2, c_2)$ and $(A_n, c_1, b_1) \simeq_k (B_n, c_2, b_2)$. ■

Armed with the previous lemma we are now ready to show that the CONNECTIVITY query is not expressible in $\mathcal{N}(\setminus, di)$.

Theorem 3.12: CONNECTIVITY *is not expressible in* $\mathcal{N}(\setminus, di)$.

PROOF: Assume for the sake of contradiction that CONNECTIVITY is expressible in $\mathcal{N}(\setminus, di)$. Then, there exists a query $q \in \mathcal{N}(\setminus, di)$ such that for any graph G, $q(G) = \mathit{true}$ if G is connected and $q(G)$ is *false* otherwise.

Now, let k be a natural number and let $n > 2 \cdot 2^k$ be an even number. Clearly, $q(A_n) = \mathit{true}$ and $q(B_n) = \mathit{false}$. Also, suppose that $(a_1, b_1) \in \text{adom}(A_n)^2$. We will split our proof up into two cases:

- If (a_1, b_1) is k-far in A_n, then, we pick a_2 and b_2 in B_2 such that they are located on different connected components. Now, Lemma 3.11(c) tells us that $(A_n, a_1, b_1) \simeq_k (B_n, a_2, b_2)$;
- If (a_1, b_1) is k-close in A_n, then, we pick a_2 and b_2 in B_2 such that $\vec{d}(a_1, b_1) = (a_2, b_2)$. Now, Lemma 3.11(a) implies that $(A_n, a_1, b_1) \simeq_k (B_n, a_2, b_2)$.

Since the two cases above cover all tuples in $\text{adom}(A_n)^2$, Corollary 3.6 contradicts our assumption, and hence q is not expressible in $\mathcal{N}(\setminus, di)$. ∎

4 Separation of Path Queries

In the preliminaries we established that separation on the level of boolean queries implies separation on the level of path queries (Proposition 2.9). Hence it is conspicuous that we establish boolean separation in the intermediary results towards the characterization of $\leq^{\text{path}}$. We will, however, not start with the full characterization of $\leq^{\text{bool}}$ because of the following reasons: not all path separation results are implied by boolean separation, the additional theory required to establish the main theorems for $\leq^{\text{bool}}$ is more intricate, and some proofs of separation results for $\leq^{\text{bool}}$ require separation results for $\leq^{\text{path}}$. Hence why we will first devote a chapter to the characterization of $\leq^{\text{path}}$.

4.1 Interdependencies between features

We will start this section by showing that several features can be defined utilizing other features.

$$\begin{aligned}\pi_1(e)(G) &= \{(m,m) \mid m \in \text{adom}(G) \wedge \neg\neg(\exists n)((m,n) \in e(G))\} = \overline{\pi}_1(\overline{\pi}_1(e))(G)\\ &= \{(m,n) \mid \exists p : (m,p) \in e(G) \wedge (n,p) \in e(G) \wedge m = n\} = (e \circ e^{-1}) \cap id(G)\\ &= \{(m,n) \mid (m,p) \in e(G) \wedge (p,n) \in id \cup di(G) \wedge m = n\} = (e \circ (id \cup di)) \cap id(G)\end{aligned}$$

$$\begin{aligned}\pi_2(e)(G) &= \{(m,m) \mid m \in \text{adom}(G) \wedge \neg\neg(\exists n)((n,m) \in e(G))\} = \overline{\pi}_2(\overline{\pi}_2(e))(G)\\ &= \{(m,n) \mid \exists p : (p,m) \in e(G) \wedge (p,n) \in e(G) \wedge m = n\} = (e^{-1} \circ e) \cap id(G)\end{aligned}$$

$$\begin{aligned}\overline{\pi}_1(e)(G) &= \{(m,n) \mid \neg(\exists p)((m,p) \in e(G)) \wedge m = n\} = id \setminus \pi_1(e)(G)\\ \overline{\pi}_2(e)(G) &= \{(m,n) \mid \neg(\exists p)((p,m) \in e(G)) \wedge m = n\} = id \setminus \pi_2(e)(G)\\ e_1 \cap e_2(G) &= e_1 \setminus (e_1 \setminus e_2)(G)\end{aligned}$$

Clearly, these interdependencies should be taken into account when talking about separation of path queries, e.g., if $\setminus \in F$, then, we surely know that we can express expressions utilizing the $\cap$ operator. Hence it is useful to consider the set of features which can be derived from the features in F by repeatedly applying the interdependencies introduced above.

Definition 4.1: Define $\overline{F}$ as the set of features obtained by augmenting F with all nonbasic features that can be expressed in $\mathcal{N}(F)$ utilizing the above interdependencies repeatedly. ◇

Example 4.2: Two examples related to the previous definition:

$$\overline{\{\overline{\pi}\}} = \{\pi, \overline{\pi}\}$$
$$\overline{\{\backslash, di\}} = \{\backslash, di, \cap, \pi, \overline{\pi}\}$$ ◇

Intuitively, it is clear that if all features in F_1 can be derived from features in F_2, i.e., $F_1 \subseteq \overline{F_2}$, then all path queries expressible in $\mathcal{N}(F_1)$ are also expressible in $\mathcal{N}(F_2)$. The following theorem justifies this intuition.

Proposition 4.3: *If $F_1 \subseteq \overline{F_2}$, then $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$.*

PROOF: For every nonbasic feature in F_1 that is not in F_2, there exists a query in $\mathcal{N}(F_2)$ that is equivalent. Therefore, every expression $e \in \mathcal{N}(F_1)$ can be transformed into an equivalent expression $e' \in \mathcal{N}(F_2)$ by substituting the lacking features with their equivalent expressions in $\mathcal{N}(F_2)$. ■

Surprisingly, the converse of the proposition above holds as well. Hence we have the following theorem.

Theorem 4.4: $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$ *if and only if* $F_1 \subseteq \overline{F_1}$.

The proof of the only if direction of this theorem is very long and intricate, and will thus be split into several propositions and lemmas for clarity. To this end, we will first establish separation for languages in the following four classes:

$$\begin{aligned}
\mathcal{C} &= \{\mathcal{N}(F) \mid \cap \notin \overline{F}, {}^+ \notin \overline{F}\} && \text{(Section 4.2)}\\
\mathcal{C}[\cap] &= \{\mathcal{N}(F) \mid \cap \in \overline{F}, {}^+ \notin \overline{F}\} && \text{(Section 4.3)}\\
\mathcal{C}[{}^+] &= \{\mathcal{N}(F) \mid \cap \notin \overline{F}, {}^+ \in \overline{F}\} && \text{(Section 4.4)}\\
\mathcal{C}[\cap, {}^+] &= \{\mathcal{N}(F) \mid \cap \in \overline{F}, {}^+ \in \overline{F}\} && \text{(Section 4.4)}
\end{aligned}$$

4.2 Languages without $\cap$ and without $^+$

In this section we will prove Theorem 4.4 for languages in $\mathcal{C}$. To this end, we will first prove several propositions which are sometimes stronger than necessary for later purposes.

Proposition 4.5 (Primitivity of *di*): *Let F_1 and F_2 be sets of nonbasic features. If $di \in \overline{F_1}$ and $di \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

PROOF: The graphs in Figure 4.1a are distinguishable in $\mathcal{N}(F_1)$ by di. On the other hand, these graphs are indistinguishable in $\mathcal{N}(F_2)$. This result can be verified using the brute-force method discussed in Section 3.1. ■

Proposition 4.6 (Primitivity of $\overline{\pi}$): *Let F_1 and F_2 be sets of nonbasic features. If $\overline{\pi} \in \overline{F_1}$, $\overline{\pi} \notin \overline{F_2}$ and $\setminus \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$;*

PROOF: The graphs in Figure 4.1b are distinguishable in $\mathcal{N}(F_1)$ by $\overline{\pi}_2$. On the other hand, these graphs are indistinguishable in $\mathcal{N}(F_2)$. Again this result can be verified by the brute-force method. Notice that these graphs are distinguishable by $id \setminus R$, hence why we require that $\setminus$ is not present in $\overline{F_2}$. ■

Proposition 4.7 (Primitivity of $^{-1}$): *Let F_1 and F_2 be sets of nonbasic features. If $^{-1} \in \overline{F_1}$ and $^{-1} \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{path}}_{\text{strong}} \mathcal{N}(F_2)$.*

PROOF: Let G be the graph in Figure 4.1c. We can compute all the possible query results for queries in $\mathcal{N}(F_2)$ in a similar manner as the brute-force method discussed in Section 3.1. Utilizing this technique we can verify that G^{-1} cannot be computed in $\mathcal{N}(F_2)$. ■

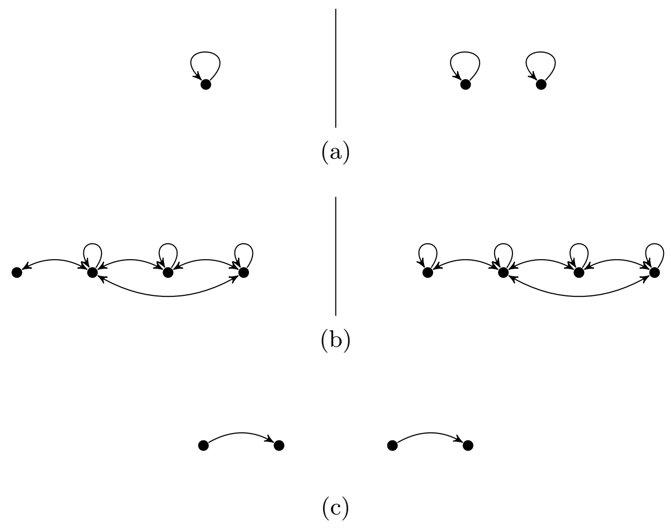


Figure 4.1: Graphs used to establish separation in Proposition 4.5, Proposition 4.6 and Proposition 4.7.

Proposition 4.8 (Primitivity of π): *Let F_1 and F_2 be sets of nonbasic features. If $\pi \in \overline{F_1}$ and $F_2 \subseteq \{^{-1}, di, ^+\}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$.*

The proof of this proposition is technical and will be split into several lemmas in Section 4.2.1.

Now we are ready to prove Theorem 4.4 for languages in $\mathcal{C}$.

Proposition 4.9: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ in $\mathcal{C}$. Then, $F_1 \not\subseteq \overline{F_2}$ implies $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$.*

PROOF: First, suppose that $\overline{\pi} \in F_2$. Then, $F_1 \not\subseteq \overline{F_2}$ if and only if $F_1 \cap \{di, ^{-1}\} \not\subseteq F_2 \cap \{di, ^{-1}\}$. Hence we have the following possible scenarios: $di \in F_1$ and $di \notin \overline{F_2}$; or $^{-1} \in F_1$ and $^{-1} \notin \overline{F_2}$. If $di \in F_1$ and $di \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$ due to Proposition 4.5. Otherwise, we achieved the result due to Proposition 4.7.

Now, suppose that $\overline{\pi} \notin F_2$. Then, $F_2 = \overline{F_2}$. Thus, $F_1 \not\subseteq \overline{F_2}$ if and only if $F_1 \not\subseteq F_2$. Hence there has to exists some $x \in F_1$ such that $x \notin F_2$. Furthermore, since $F_1 \subseteq \{di, \pi, \overline{\pi}, ^{-1}\}$ and $F_2 \subseteq \{di, \pi, ^{-1}\}$, either Proposition 4.5, Proposition 4.6, Proposition 4.7 or Proposition 4.8 gives us the required result. Notice that we cannot apply either proposition directly since Proposition 4.5, Proposition 4.6, Proposition 4.7 and Proposition 4.8 use $\overline{F_1}$ instead of F_1. However, this is no issue since $F_1 \subseteq \overline{F_1}$. ∎

4.2.1 Proof of Proposition 4.8.

Towards the proof of this proposition, we first need several technical propositions and lemmas.

First, notice that the result of a path query is a binary relation, hence we can obviously interpret it as an unlabeled graph. This insight will be exploited in the proof of the following lemma, which we will need several times throughout this thesis.

Lemma 4.10: *Let G be a graph and let $n = |\text{adom}(G)|$. For every path query e,*

$$e^+(G) = \bigcup_{i=1}^{n} e^i(G).$$

PROOF: It is sufficient to show that for $(a, b) \in e^m$ with $m > n$, there exists an $l \leq n$, such that $(a, b) \in e^l$. We will prove this by induction on m.

Induction Basis: Let $m = n + 1$. If $(a, b) \in e^{n+1}(G)$, then there exists a path of $n + 2$ vertices, $p_1, \ldots, p_{n+2} \in \text{adom}(G)$, such that,

- $p_1 = a$, and $p_{n+2} = b$;
- $\forall 1 \leq i < n + 2$: $(p_i, p_{i+1}) \in e(G)$.

Since there are only n different vertices in adom(G), we know there exists i and j, where $i \neq j$, such that $p_i = p_j$ (Pigeonhole principle). Clearly, the path along the vertices $p_{i+1}, \ldots, p_j$ is redundant and can be removed from the path to create a shorter path from a to b. Furthermore, at least one vertex is removed from the path, thus, the new path from a to b consists of at most $n + 1$ vertices, which contains at most n edges, hence there exists $l \leq n$, such that $(a, b) \in e^l(G)$. Figure 4.2 provides a graphical representation of this proof, where the red path is the path along $p_{i+1}, \ldots, p_j$. Clearly, this red path can be removed without breaking the path from p_1 to p_{n+2}.

Induction Hypothesis: Let $m = k$. Assume that if $(c, d) \in e^k$, then there exists

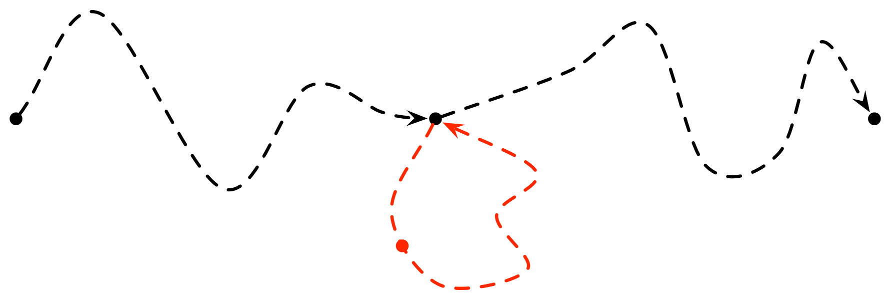


Figure 4.2: Graphical sketch of the proof of lemma 4.10. The red path can obviously be removed without breaking the path from the left hand node to the right hand node.

$l \leq n$, such that $(a, b) \in e^l$.
Induction Step: Let $m = k + 1$. Remember that $e^{k+1}(G) = e^k \circ e(G)$. Notice that

$$\begin{aligned}
(a,b) \in e^{k+1}(G) &\implies \exists p : (a,p) \in e^k \wedge (p,b) \in e(G) \\
&\implies (a,p) \in e^l \text{ with } l \leq n \wedge (p,b) \in e(G) \qquad \text{(hypothesis)} \\
&\implies (a,b) \in e^{l+1}.
\end{aligned}$$

If $l < n$ we are done, if not, we can apply our induction basis. ■

In a graph theoretic context, the previous lemma tells us that every path from a node x to a node y in a graph can be reduced to a simple path from x to y.

Definition 4.11: Let n be a natural number and let e be an expression in $\mathcal{N}(^{-1}, di, ^{+})$. The set $\text{expand}_n(e)$ of expressions in $\mathcal{N}(^{-1}, di)$ *without union* is defined inductively as follows:

$$\begin{aligned}
\text{expand}_n(R) &:= R, \textit{for any } R \in \Lambda \\
\text{expand}_n(id) &:= id \\
\text{expand}_n(di) &:= di \\
\text{expand}_n(e_1^{-1}) &:= \{e^{-1} \mid e \in \text{expand}_n(e_1)\} \\
\text{expand}_n(e_1 \cup e_2) &:= \text{expand}_n(e_1) \cup \text{expand}_n(e_2) \\
\text{expand}_n(e_1 \circ e_2) &:= \{e_1' \circ e_2' \mid e_1' \in \text{expand}_n(e_1) \wedge e_2' \in \text{expand}_n(e_2)\} \\
\text{expand}_n(e_1^{+}) &:= \{\bigcup (\text{expand}_n(e_1))^k \mid k = 1, \ldots, n\}
\end{aligned} \qquad \diamond$$

Intuitively, the following lemma tells us that if we only consider input graphs with a predetermined size bound, expressions with transitive closure can be simplified to equivalent expressions without transitive closure. It also tells us that all the union operations in an expression can be pulled out to the 'top' of the expression without changing the expression.

Lemma 4.12: *Let n be a natural number; let G be a graph with $|\mathrm{adom}(G)| \leq n$ and let e be an expression in $\mathcal{N}(^{-1}, di, ^{+})$. Then, $e(G) = \bigcup_{e' \in expand_n(e)} e'(G)$.*

PROOF: We will prove this lemma by structural induction on e.
Induction Basis: Let $e = R$ with $R \in \Lambda$. Then,

$$\begin{aligned}\bigcup_{e' \in \mathrm{expand}_n(R)} e'(G) &= \bigcup_{e' \in \{R\}} e'(G) \\ &= R(G).\end{aligned}$$

For $e = id(G)$ and $e = di(G)$ the proof is exactly the same.
Induction Hypothesis: Assume that our lemma holds for $e_1, e_2 \in \mathcal{N}(^{-1}, di)$ without union.
Induction Step:

– Let $e = e_1 \cup e_2$. Then,

$$\begin{aligned}e_1 \cup e_2(G) &= e_1(G) \cup e_2(G) \\ &= \textstyle\bigcup_{e' \in \mathrm{expand}_n(e_1)} e'(G) \cup \bigcup_{e' \in \mathrm{expand}_n(e_2)} e'(G) && \text{(hypothesis)} \\ &= \textstyle\bigcup_{e' \in \mathrm{expand}_n(e_1 \cup e_2)} e'(G)\end{aligned}$$

– Let $e = e_1^{-1}$. Then,

$$\begin{aligned}e_1^{-1}(G) &= \{(a,b) \mid (b,a) \in e_1(G)\} \\ &= \{(a,b) \mid (b,a) \in \textstyle\bigcup_{e' \in \mathrm{expand}_n(e_1)} e'(G)\} && \text{(hypothesis)} \\ &= \textstyle\bigcup_{e' \in \mathrm{expand}_n(e_1)} e'^{-1}(G) \\ &= \textstyle\bigcup_{e' \in \mathrm{expand}_n(e_1^{-1})} e'(G)\end{aligned}$$

– Let $e = e_1 \circ e_2$. Then,

$$\begin{aligned}e_1 \circ e_2(G) &= \{(a,b) \mid \exists p : (a,p) \in e_1(G) \wedge (p,b) \in e_2(G)\} \\ &= \{(a,b) \mid \exists p : (a,p) \in \textstyle\bigcup_{e' \in \mathrm{expand}_n(e_1)} e'(G) \\ &\qquad \wedge (p,b) \in \textstyle\bigcup_{e' \in \mathrm{expand}_n(e_2)} e'(G)\} && \text{(hypothesis)} \\ &= \{(a,b) \mid \exists p, \exists e_1' \in \mathrm{expand}_n(e_1), \\ &\qquad \exists e_2' \in \mathrm{expand}_n(e_2)\colon (a,p) \in e_1'(G) \wedge (p,b) \in e_2'(G)\} \\ &= \{(a,b) \mid \exists e_1' \in \mathrm{expand}_n(e_1), \exists e_2' \in \mathrm{expand}_n(e_2)\colon (a,b) \in e_1' \circ e_2'\} \\ &= \textstyle\bigcup_{e' \in \mathrm{expand}_n(e_1 \circ e_2)} e'(G)\end{aligned}$$

– Let $e = e_1^+$. Then,

$$\begin{aligned}
e_1^+(G) &= \bigcup_{i=1}^{n} e_1^i(G) && \text{(see Lemma 4.10)}\\
&= \bigcup_{i=1}^{n} \left(\bigcup_{e' \in \text{expand}_n(e_1)} e'(G) \right)^i && \text{(hypothesis)}\\
&= \bigcup_{i=1}^{n} \left(\bigcup_{e' \in \text{expand}_n(e_1)^i} e'(G) \right) \\
&= \bigcup_{e' \in \text{expand}_n(e_1^+)} e'(G) && \blacksquare
\end{aligned}$$

The previous theorem gave us an equivalent expression for every $e \in \mathcal{N}(^{-1}, di, ^+)$ in terms of $\text{expand}_n(e)$ if the size of the input graphs is bounded by a predetermined constant. If we lift this restriction, there is still a relation between the expressions in $\text{expand}_n(e)$ and e.

Lemma 4.13: *Let n be a natural number and let $e \in \mathcal{N}(^{-1}, di, ^+)$. For every expression $e' \in expand_n(e)$, $e' \subseteq e$, i.e., for every graph G, $e'(G) \subseteq e(G)$.*

PROOF: We will prove this lemma by structural induction.
Induction Basis: For $e = di, id$ or R with $R \in \Lambda$, we know that $\text{expand}_n(e)$ only contains 1 expression, e itself, thus $e' = e$.
Induction Hypothesis: Assume that for each subexpression of e our lemma is true.
Induction Step:

– Let $e = e_1^{-1} \in \mathcal{N}(^{-1}, di, ^+)$. Then,

$$\begin{aligned}
e' \in \text{expand}_n(e_1^{-1}) &\implies e'^{-1} \in \text{expand}_n(e_1)\\
&\implies e'^{-1} \subseteq e_1 && \text{(hypothesis)}\\
&\implies e' \subseteq e_1^{-1}.
\end{aligned}$$

– Let $e = e_1 \cup e_2 \in \mathcal{N}(^{-1}, di, ^+)$. Then,

$$\begin{aligned}
e' \in \text{expand}_n(e_1 \cup e_2) &\implies e' \in \text{expand}_n(e_1) \vee e' \in \text{expand}_n(e_2)\\
&\implies e' \subseteq e_1 \vee e' \subseteq e_2 && \text{(hypothesis)}\\
&\implies e' \subseteq e_1 \cup e_2.
\end{aligned}$$

The last step follows from the fact that both e_1 and e_2 are contained within $e_1 \cup e_2$.

– Let $e = e_1 \circ e_2 \in \mathcal{N}(^{-1}, di, ^{+})$. Then,

$$\begin{aligned} e' \in \text{expand}_n(e_1 \circ e_2) &\implies \exists e'_1 \in \text{expand}_n(e_1), e'_2 \in \text{expand}_n(e_2) : e' = e'_1 \circ e'_2 \\ &\implies e'_1 \subseteq e_1 \wedge e'_2 \subseteq e_2 && \text{(hypothesis)} \\ &\implies e'_1 \circ e'_2 \subseteq e_1 \circ e_2. \end{aligned}$$

– Let $e = e_1^{+} \in \mathcal{N}(^{-1}, di, ^{+})$. Then,

$$\begin{aligned} e' \in \text{expand}_n(e_1^{+}) &\implies \exists 1 \leq k \leq n : e' \in \text{expand}_n(e_1)^k \\ &\implies e' = e'_1 \circ \ldots \circ e'_k \text{ (with } e'_i \in \text{expand}_n(e_1)) \\ &\implies \forall 1 \leq i \leq k : e'_i \subseteq e_1 && \text{(hypothesis)} \\ &\implies e'_1 \circ \ldots \circ e'_k \subseteq e_1^k \subseteq e_1^{+}. \end{aligned}$$ ■

In the proof of Proposition 4.8 we argue for the sake of contradiction that if $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$, we can construct a nontrivial endomorphism on the B_{ZZZ} pattern displayed in Figure 4.3. The following lemma tells us that this contradicts our assumption as B_{ZZZ} has no nontrivial endomorphisms.

Lemma 4.14: *The B_{ZZZ} pattern has no nontrivial endomorphism.*

PROOF: Let f be an endomorphism of Q_{ZZZ}. First, we will show that $f(a) = a$ by contradiction. Define $[x, y]$ as the set of vertices along all the directed paths from x to y. The including/excluding semantics of this notation are defined in the same way as the interval notation on the real numbers.

– Since there exists a directed path starting in a of length 6 (the path from a to g), there must also exist such a path starting in $f(a)$. The only vertices in B_{ZZZ} from which a path of length 6 starts are a and j. We will now show by contradiction that $f(a) = a$. Suppose for the sake of contradiction that $f(a) = j$. Then, $f(g) = k$, and thus $f(p) = l$. Furthermore, since f is a homomorphism and $(j, g) \in B_{ZZZ}$, it is clear that $(f(j), f(g)) \in B_{ZZZ}$, which implies that $f(j) = l$. But, there starts a path of length 6 in j, hence there also has to start a path of length 6 in l. However, no such path exists.

– Suppose that $f(a) = h$ or i. Similarly, there has to be a path of length 6 in B_{ZZZ} from $f(a)$ to another vertex in B_{ZZZ}. However, again, no such path exists.

– Suppose that $f(a) = j$. Then, $f(g) = k$, and thus $f(p) = l$. Furthermore, since $(j, g) \in B_{ZZZ}$ and f is a homomorphism, $(f(j), f(g)) = (f(j), k) \in B_{ZZZ}$, which implies that $f(j) = l$. But, there starts a path of length 6 in j, hence there also starts a path of length 6 in $f(j)$. This path, however, does not exist.

Let us now prove that for every x, $f(x) = x$.

- First, we will show that for every $x \in [d, g] : f(x) = x$. Clearly, $(f(a), f(d)) = (a, f(d)) \in B_{ZZZ}$, since $(a, d) \in B_{ZZZ}$ and since f is a homomorphism. Hence the only possibilities for $f(d)$ are b, c and d. Moreover, since there starts a path of length 5 in d, there also has to start a path of length 5 in $f(d)$, which is impossible for both b and c. Now, clearly for f not to violate the homomorphism condition, every element in $[d, g]$ is fixed by f.

- To show that for every $x \in [b, e]$: $f(x) = x$; we only have to show that $f(b) = b$. Since f is a homomorphism and $f(a) = a$, we know that $f(b)$ equals b, c or d. First, notice that showing $f(b) \neq c$ is completely analogous to the previous case. We will now show that $f(b) \neq d$. Suppose for the sake of contradiction that $f(b) = d$, then $f(e) = p$ and thus $f(h) = q$. But, then there has to exist a path of length 5 starting in q. However, no such path exists in B_{ZZZ}.

- As in the previous case, to show that for every $x \in [c, m]$: $f(x) = x$, we only have to show that $f(c) = c$. Suppose for the sake of contradiction that $f(c) \neq c$, then $f(c)$ equals b or d. If $f(c) = d$, then $f(m) = q$ and thus $f(i) = z$. But, then there has to start a path of length 4 starting in z. However, no such path exists, contradiction. On the other hand, if $f(c) = b$, then $f(m) = s$ and thus $f(i) = r$. Moreover, since a directed path of length 4 starts in i and since f preserves such paths, there also has to start a directed path of length 4 in r, but no such path exists.

- Now, we will show that $f(h) = h$, $f(i) = i$ and $f(j) = j$. Due to the previous cases,

  $$(f(h), f(e)) = (f(h), e) \text{ and } (f(j), f(g)) = (f(j), g), (f(i), f(m)) \text{ and } (f(i), m)$$

  hence $(f(h), e), (f(j), g), (f(i), m) \in B_{ZZZ}$ (since f is a homomorphism). Thus, $f(j)$ equals j or p, $f(i)$ equals i or t and $f(h)$ equals h or r. Since there starts a path of length 5 in h, there also has to start a path of length 5 in $f(h)$. Hence $f(h) \neq r$, since no such path exists starting in r. The same argument can be used to show that $f(i) \neq t$ and $f(j) \neq p$, since there start no paths of length 4 in t and no paths of length 6 in p, hence $f(i) \neq t$ and $f(j) \neq p$.

  Now since $f(h) = h$, $f(i) = i$ and $f(j) = j$ and since f is a homomorphism, every element in $[h, n]$, $[i, o]$ and $[j, k]$ is fixed by f. ■

The following lemma states that the B_{ZZZ} pattern can also be recognized with an expression not using the inverse operation.

Lemma 4.15: *$\pi_1(R^k \circ R^{-1} \circ R^k)$ is equivalent to $\pi_1(R^k \circ \pi_2(\pi_1(R^k) \circ R))$.*

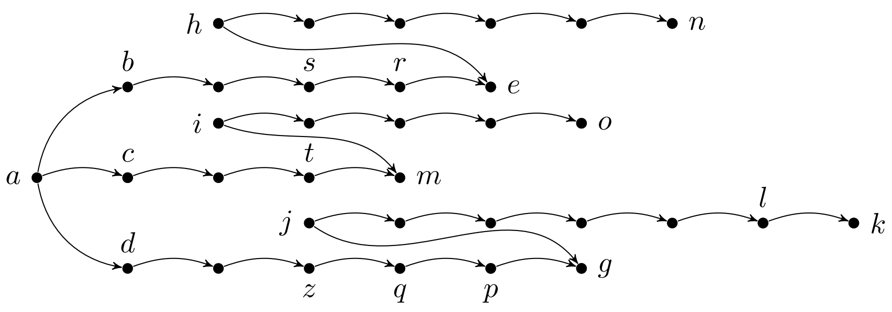


Figure 4.3: Graph with B_{ZZZ} pattern

PROOF: Let $e_1 = \pi_1(R^k \circ R^{-1} \circ R^k)$ and $e_2 = \pi_1(R^k \circ \pi_2(\pi_1(R^k) \circ R))$. We will first show that $e_1 \subseteq e_2$. Let G be an arbitrary graph. Then,

$$\begin{aligned}
(a,a) \in e_1(G) &\implies \exists b : (a,b) \in R^k \circ R^{-1} \circ R^k \\
&\implies \exists c, d : (a,c) \in R^k, (d,c) \in R \text{ and } (d,b) \in R^k \\
&\implies (d,d) \in \pi_1(R^k) \\
&\implies (d,c) \in \pi_1(R^k) \circ R \\
&\implies (c,c) \in \pi_2(\pi_1(R^k) \circ R) \\
&\implies (a,c) \in R^k \circ \pi_2(\pi_1(R^k) \circ R) \\
&\implies (a,a) \in \pi_1(R^k \circ \pi_2(\pi_1(R^k) \circ R)).
\end{aligned}$$

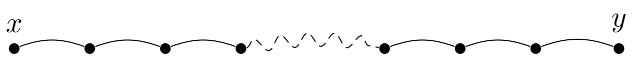
Conversely, let us show that $e_2 \subseteq e_1$.

$$\begin{aligned}
(a,a) \in e_2 &\implies \exists b : (a,b) \in R^k \circ \pi_2(\pi_1(R^k) \circ R) \\
&\implies \exists x : (a,x) \in R^k \text{ and } (x,b) \in \pi_2(\pi_1(R^k) \circ R) \\
&\implies x = b \land \exists y : (y,b) \in \pi_1(R^k) \circ R \\
&\implies (a,b) \in R^k \land (y,b) \in R \land \exists z : (y,z) \in R^k \\
&\implies (a,z) \in R^k \circ R^{-1} \circ R^k \\
&\implies (a,a) \in \pi_1(R^k \circ R^{-1} \circ R^k)
\end{aligned}$$

■

In the proof of Proposition 4.8 we will use conjunctive queries. To support this, we will first need to be able to translate expressions without union in $\mathcal{N}(^{-1}, di)$ to conjunctive queries. The following theorem tells that we are able to do so. Moreover, the translations have a very special format.

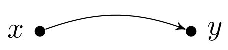
Lemma 4.16: *If $e \in \mathcal{N}(^{-1}, di)$ without union, then there exists an equivalent conjunctive query $Q = (H(x,y), B, noneq)$ with nonequalities whose graph forms a linear chain from x to y. If we would ignore the direction of the R-edges and the edge labels, the graph looks as follows.*



Importantly, notice that no vertices in this graph contain self edges.

Proof: We will prove this by structural induction on e.
Induction Basis:

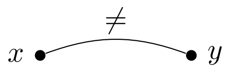
- Let $e = R$. Then, $Q = H(x, y) \leftarrow R(x, y)$. The graph of Q looks as follows,



which clearly has the required form;

- Let $e = id$. Then, $Q = H(x, x)$, where Q simply has no atoms in its body. The graph of Q is just a single vertex and thus has the required form;

- Let $e = di$. Then, $Q = H(x, y) \leftarrow x \neq y$. The graph of Q looks as follows,



where the edge between x and y is a di-edge.

Induction Hypothesis: Suppose that for $e_1, e_2 \in \mathcal{N}(^{-1}, di)$ without union there exists conjunctive queries with nonequalities $Q_1 = (H_1(x_1, y_1), B_1, noneq_1)$ and $Q_2 = (H_2(x_2, y_2), B_2, noneq_2)$ such that $e_1 \equiv Q_1$ and $e_2 \equiv Q_2$. Furthermore, the graphs of Q_1 and Q_2 are of the required form.
Induction Basis:

- Let $e = e_1 \circ e_2$. Now, let B'_1 (resp. $noneq'_1$ and H'_1) equal B_1 (resp. $noneq_1$ and H_1) where every occurrence of y_1 is replaced with a *fixed* new variable not yet present in Q_1 and Q_2. Let us denote this new variable with k. Furthermore, let B'_2 (resp. $noneq'_2$ and H'_2) equal B_2 (resp. $noneq_2$ and H_2) where every occurrence of x_2 is replaced with the same variable k. Furthermore, let $Q = (H(x_1, y_2), B'_1 \cup B'_2, noneq'_1 \cup noneq'_2)$. We will now show that Q is equivalent to e. First, we will show that $e \subseteq Q$. To this end, let G be an arbitrary graph. Then,

$$\begin{aligned}
(a, b) \in e(G) &\implies \exists p : (a, p) \in e_1(G) \wedge (p, b) \in e_2(G) \\
&\implies \exists p : (a, p) \in Q_1(G) \wedge (p, b) \in Q_2(G) \qquad \text{(hypothesis)} \\
&\implies \exists \text{ matchings } f : Q_1 \to G \text{ and } g : Q_2 \to G \\
&\qquad \text{where } f(x_1) = a, f(y_1) = p, g(x_2) = p \text{ and } g(y_2) = b.
\end{aligned}$$

Now, we will try to find a matching $f^* : Q_1 \to G$, such that $f^*(x_1, y_2) = (a, b)$. Let us define f^* as follows

$$f^*(x) = \begin{cases} f(x), & \text{if } x \in \mathit{vars}(B_1') \cup \mathit{vars}(H_1') \setminus \{k\} \\ g(x), & \text{if } x \in \mathit{vars}(B_2') \cup \mathit{vars}(H_2') \setminus \{k\} \\ f(y_1), & \text{if } x = k. \end{cases}$$

Now, we have to show that f^* is a matching. To this end we will first show that $f^*(B_1' \cup B_2') \subseteq G$.

$$f^*(B_1' \cup B_2') = f^*(B_1') \cup f^*(B_2') = f(B_1) \cup g(B_2) \subseteq G$$

To prove that f^* is compatible with the nonequalities, we will split our proof up into the following cases.

- If $x \neq y \in \mathit{noneq}_1'$, where x and y are not equal to k, then $x \neq y \in \mathit{noneq}_1$ and thus $f^*(x) = f(x) \neq f(y) = f^*(y)$;
- If $x \neq y \in \mathit{noneq}_2'$, where x and y are not equal to k, then $x \neq y \in \mathit{noneq}_2$ and thus $f^*(x) = g(x) \neq g(y) \neq f^*(y)$;
- Suppose that $x = k$ and $x \neq y \in \mathit{noneq}_1'$, then $y_1 \neq y \in \mathit{noneq}_1$ and thus $f^*(x) = f(y_1) \neq f(y) = f^*(y)$. On the other hand if $y = k$, the proof is completely analogous;
- Again, suppose that $x = k$ and $x \neq y \in \mathit{noneq}_2'$, then $x_2 \neq y \in \mathit{noneq}_2$. Now, $f^*(x) = f^*(k) = f(y_1) = g(x_2) \neq g(y) = f^*(y)$. On the other hand if $y = k$, the proof is analogous.

Thus f^* is a matching for Q, which implies that $(a, b) \in Q(G)$ and thus $e \subseteq Q$.

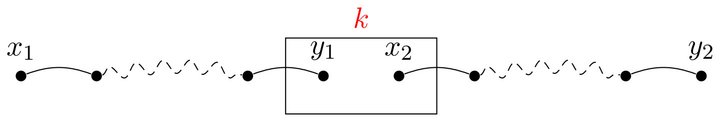
Conversely, we will show that $Q \subseteq e$. If $(a, b) \in Q(G)$, then there exists a matching $f : Q \to G$. Now, consider the two substitutions $g : Q_1 \to G$ and $h : Q_2 \to G$ which are defined as follows.

$$g(x) = \begin{cases} f(x), & \text{if } x \in \mathit{vars}(Q_1) \setminus \{y_1\} \\ p, & \text{if } x = y_1 \end{cases} \qquad h(x) = \begin{cases} f(x), & \text{if } x \in \mathit{vars}(Q_2) \setminus \{x_1\} \\ p, & \text{if } x = x_2 \end{cases}$$

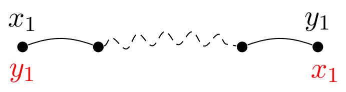
Clearly, both $g(B_1)$ and $h(B_2)$ are a subsets of G. Furthermore, for analogous reasons as in the proof for $e \subseteq Q$, g is compatible with noneq_1 and h is compatible with noneq_2. Therefore, g is a matching for Q_1 and h is a matching for Q_2, which implies that $(a, p) \in Q_1(G)$ and $(p, b) \in Q_2$. However, this implies that

$$\begin{aligned} (a, p) \in Q_1(G) \wedge (p, b) \in Q_2 &\implies (a, p) \in e_1(G) \wedge (p, b) \in e_2(G) \quad \text{(hypothesis)} \\ &\implies (a, b) \in e_1 \circ e_2(G) \end{aligned}$$

Thus, $Q \subseteq e$. Now, we will show that Q's graph has the required form. Due to our induction hypothesis we know for both Q_1 and Q_2 there exists a graph of the required form. In our construction of Q, y_1 and x_2 are both replaced by the same variable k. Thus to obtain the graph of Q we combine the graphs of Q_1 and Q_2 and 'merge' the vertices representing y_1 and x_2, and label it k. For an illustration, see the graph right below.



– Let $e = e_1^{-1}$. We define $Q = (H(y_1, x_1), B_1, noneq_1)$, where B_1 is the body of Q_1 and $noneq_1$ is the set of nonequalities of Q_1. Clearly, $e \equiv Q$ and the graph of Q equals the graph of Q_1 where the vertices representing x_1 and y_1 are swapped. For an illustration see the graph right below, where the vertex labels in black are from the graph of Q_1 and the red vertex labels are from the graph of Q.



■

Notice that every vertex in graph of the constructed query Q in Lemma 4.16 has at most two outgoing edges. This insight will be very important in the proof of Proposition 4.8.

Define Q_{ZZZ} as the conjunctive query without nonequalities $(H_{ZZZ}, B_{ZZZ}, noneq)$ where B_{ZZZ} is the graph displayed in Figure 4.3 and where $noneq = \emptyset$. Clearly, Q_{ZZZ} recognizes the B_{ZZZ} pattern.

Proof (of Proposition 4.8): Let $\pi \in \overline{F_1}$. Proposition 4.3 tells us that $\mathcal{N}(\pi) \leq^{\text{bool}} \mathcal{N}(F_1)$, hence it will suffice to show that $\mathcal{N}(\pi) \not\leq^{\text{bool}} \mathcal{N}({}^{-1}, di, {}^{+})$, as this implies that $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}({}^{-1}, di, {}^{+})$ and moreover $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$, which is exactly what we want to prove.

The query Q_{ZZZ} which recognizes the B_{ZZZ} pattern can be expressed in $\mathcal{N}({}^{-1}, \pi)$ by the query

$$\pi_1(R^4 \circ R^{-1} \circ R^4) \circ \pi_1(R^5 \circ R^{-1} \circ R^5) \circ \pi_1(R^6 \circ R^{-1} \circ R^6)$$

which according to Lemma 4.15 is equivalent with the following query in $\mathcal{N}(\pi)$

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

Now, suppose that there exists some query $Q \in \mathcal{N}({}^{-1}, di, {}^{+})$ which is equivalent to Q_{ZZZ}. Hence Q also recognizes the B_{ZZZ} pattern and thus $Q(B_{ZZZ}) \neq \emptyset$. Utilizing Lemma 4.12 we know that $Q(B_{ZZZ}) = \cup_{e \in \text{expand}_n(Q)} e(B_{ZZZ}) \neq \emptyset$ (where $n = |B_{ZZZ}|$) and thus there exists $e \in \text{expand}_n(Q)$ such that $e(B_{ZZZ}) \neq \emptyset$.

Furthermore, $\text{expand}_n(Q)$ consists of expressions in $\mathcal{N}({}^{-1}, di)$ without union, and thus there exists a conjunctive query with nonequalities $Q_e = (H_e, B_e, noneq_e)$ as in

Lemma 4.16 which is equivalent to e. Therefore, $Q_e(B_{ZZZ})$ is also nonempty and thus there exists a matching $f : B_e \to B_{ZZZ}$ which by definition is a homomorphism.

By Lemma 4.13 we know that $e \subseteq Q$ and since $Q \subseteq Q_{ZZZ}$, we know that $e \subseteq Q_{ZZZ}$. Moreover, since the trivial partition π is compatible with every set of nonequalities, we know there exists a homomorphism $h : B_{ZZZ} \to B_e/\pi \cong B_e$ due to Theorem 2.22. Remember that in the graph of Q_e every vertex has at most two outgoing edges (see Lemma 4.16), in particular $h(a)$, where a is the left most vertex labelled 'a' in Figure 4.3. Note that homomorphisms preserve edges and thus $(h(a), h(b))$, $(h(a), h(c))$ and $(h(a), h(d))$ are edges in B_e. Therefore, two of b, c and d have the same image under h, which implies that h is not injective, hence $f \circ h$ is not bijective and thus cannot be the trivial map. However, notice that $f \circ h$ is an endomorphism of B_{ZZZ}, which according to the above is nontrivial and hence contradicts Lemma 4.14. ■

4.3 Languages with $\cap$ and without $^+$

In this section, we will prove Theorem 4.4 restricted to languages in $\mathcal{C}[\cap]$, i.e., the class of languages with $\cap$ and without $^+$. First, however, we need two preliminary propositions, which are more general than needed for later purposes.

Proposition 4.17 (Primitivity of \\): *Let F_1 and F_2 be sets of nonbasic features. If $\setminus \in \overline{F_1}$ and $\setminus \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

Proof: The graphs in Figure 4.4a are distinguishable in $\mathcal{N}(F_1)$ by $R^2 \setminus (id \cup R)$. On the other hand, they are indistinguishable in $\mathcal{N}(F_2)$. This last result can be verified with the brute-force method described in Section 3.1. ■

Proposition 4.18 (Primitivity of π): *Let F_1 and F_2 be sets of nonbasic features. If $\pi \in \overline{F_1}$ and $F_2 \subseteq \{\setminus, \cap, ^+\}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

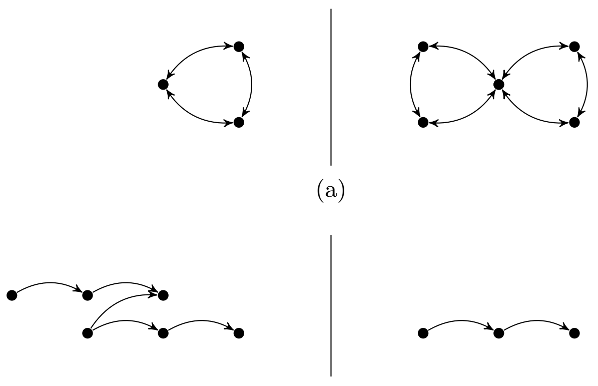
Proof: The brute-force algorithm tells us that the graphs in Figure 4.4b are indistinguishable in $\mathcal{N}(F_2)$. They are however distinguishable in $\mathcal{N}(F_1)$ by $\pi_1(R^2) \circ R \circ \pi(R^2)$. ■

Now we are ready to prove Theorem 4.4 restricted to languages in $\mathcal{C}[\cap]$. The proof will be a thorough case analysis.

Proposition 4.19: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be in $\mathcal{C}[\cap]$. If $F_1 \not\subset \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$.*

Proof: By definition $\cap \in \overline{F_1}$ and $\cap \in \overline{F_2}$ since both $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ are in $\mathcal{C}[\cap]$. Hence, $F_1 \not\subseteq \overline{F_2}$ if and only if there exists $x \in \{\pi, \overline{\pi}, di, ^{-1}, \setminus\}$ such that $x \in F_1$ and $x \notin \overline{F_2}$. We will consider every such x and show that one of Proposition 4.5, Proposition 4.6, Proposition 4.7, Proposition 4.8, Proposition 4.17 or Proposition 4.18 implies our result.

- $x = di$: Proposition 4.5 gives us the desired result;
- $x = ^{-1}$: Proposition 4.7 proves what was asked;


(a)

(b)

Figure 4.4: Graphs used to establish separation in Proposition 4.17, Proposition 4.18 and Proposition 5.10.

- $x = \setminus$: Proposition 4.17 proves our proposition in this case;
- $x = \pi$. Then, $\pi \notin \overline{F_2}$ if and only if $di, ^{-1}, \overline{\pi}, \pi \notin F_2$. Hence $F_2 \subseteq \{\cap, \setminus\}$. Now, we can apply Proposition 4.18, which proves the result;
- $x = \overline{\pi}$. Then,

  $$\overline{\pi} \notin \overline{F_2} \iff \setminus \notin F_2 \vee (\setminus \in F_2 \wedge \pi \notin \overline{F_2}).$$

  If $\setminus \notin F_2$, we can apply Proposition 4.6 to prove our result.

  On the other hand, if $\setminus \in F_2$, then we cannot apply Proposition 4.6. As said above, now π cannot be in $\overline{F_2}$. Now note that

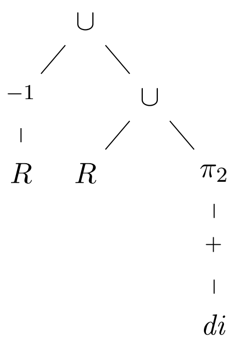
  $$\setminus \in F_2 \wedge \pi \notin \overline{F_2} \iff di \notin F_2 \wedge ^{-1} \notin F_2$$

  which implies $F_2 \subseteq \{\cap, \setminus\}$ since $F_2 \subseteq \{\cap, \pi, \overline{\pi}, di, ^{-1}, \setminus\}$. Furthermore, $\pi \in \overline{F_1}$ since $\overline{\pi} \in \overline{F_1}$. Hence, we can apply Proposition 4.18, which proves the result. ∎

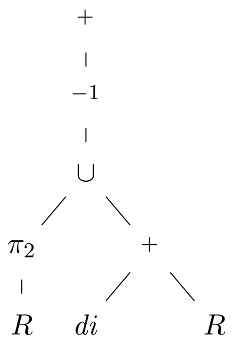
4.4 Languages without $\cap$ and with $^+$ or languages with $\cap$ and with $^+$

In this section we will prove Theorem 4.4 for languages in $\mathcal{C}[^+]$. To this end, we first need the following definition.

Definition 4.20: The transitive closure depth (TC-depth) of a query $q \in \mathcal{N}(F)$ is the maximum number of transitive closures on a path from root to a leaf in q's tree



(a) TC-depth 1



(b) TC-depth 2

Figure 4.5: Expressions with different a TC-depth.

representation. More formally

$$\begin{aligned}
&\operatorname{degr}(id) = \operatorname{degr}(di) = \operatorname{degr}(\emptyset) = \operatorname{degr}(R) = 0 \qquad \text{for every } R \in \Lambda;\\
&\operatorname{degr}(\pi_i(e_1)) = \operatorname{degr}(\overline{\pi}_i(e_1)) = \operatorname{degr}(e_1^{-1}) = \operatorname{degr}(e_1);\\
&\operatorname{degr}(e_1 \circ e_2) = \operatorname{degr}(e_1 \cup e_2) = \operatorname{degr}(e_1 \cap e_2) = \operatorname{degr}(e_1 \setminus e_2) = \max(\operatorname{degr}(e_1), \operatorname{degr}(e_2));\\
&\operatorname{degr}(e_1^{+}) = 1 + \operatorname{degr}(e_1). \qquad \diamond
\end{aligned}$$

For example, in Figure 4.5a the tree representation of an expression with TC-depth 1 is displayed. In Figure 4.5b an expression with TC-depth 2 is shown.

The following proposition tells us that if every expression in $\mathcal{N}(F_1)$ is also expressible in $\mathcal{N}(F_2)$, then adding transitive closure to both languages leaves this relation in tact.

Proposition 4.21: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be arbitrary languages. Then, $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$ implies $\mathcal{N}(F_1 \cup \{^+\}) \leq^{\text{path}} \mathcal{N}(F_2 \cup \{^+\})$.*

Proof: In this proof we will explicitly 'exploit' the set of edge labels over which input graphs are constructed. To this end, let Λ be an arbitrary set of edge labels and let $\mathcal{N}_\Lambda(F_1)$ and $\mathcal{N}_\Lambda(F_2)$ be arbitrary languages. We will show that if $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$ and if e is expressible in $\mathcal{N}_\Lambda(F_1 \cup \{^+\})$ then e is also expressible in $\mathcal{N}_\Lambda(F_2 \cup \{^+\})$ $(*)$. We will do so by induction on the transitive closure depth.
Induction Basis: If TC-DEPTH$(e) = 0$, then $e \in \mathcal{N}_\Lambda(F_1)$. Hence e is also expressible in $\mathcal{N}_\Lambda(F_2)$ by hypothesis. But since $\mathcal{N}_\Lambda(F_2) \subset \mathcal{N}_\Lambda(F_2 \cup \{^+\})$ it follows directly that e is also expressible in $\mathcal{N}(F_2 \cup \{^+\})$.
Induction Hypothesis: Suppose that $(*)$ is true if TC-DEPTH$(e) < n$ $(n > 0)$.
Induction Step: Suppose that TC-DEPTH$(e) = n$. In the tree representation of e, we will identify the top level transitive closure application on every path from the root to a leaf, i.e., every transitive closure application which is closest to the root on such a path. Every subtree starting in an identified transitive closure also represents a query,

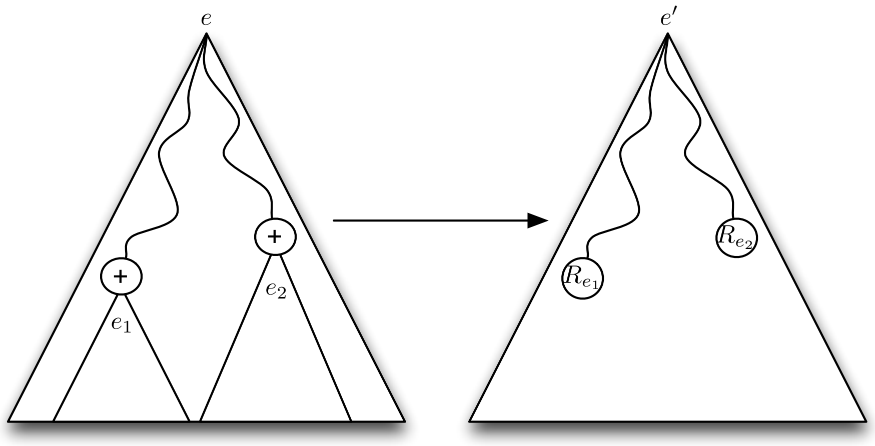


Figure 4.6: Graphical representation of the proof of Proposition 4.21

let us call these queries $e_1^+, \ldots, e_k^+$, where $e_1, \ldots, e_k$ are the expression starting under the identified transitive closures. Now, we will construct a new expression e' from e by replacing $e_1^+, \ldots, e_k^+$ in e with $R_{e_1}, \ldots, R_{e_k}$ respectively, where $R_{e_1}, \ldots, R_{e_k}$ are labels not yet present in Λ. Since e' contains labels not present in Λ we have to construct a new set of edge labels $\Lambda' = \Lambda \cup \{R_{e_1}, \ldots, R_{e_2}\}$. Clearly, e' does not contain the transitive closure operator, hence $e' \in \mathcal{N}_{\Lambda'}(F_1)$. Figure 4.6 illustrates this process; the paths from the root to the displayed transitive closures are transitive closure free, hence they are identified transitive closure operators.

By hypothesis e' is expressible in $\mathcal{N}_{\Lambda'}(F_2)$ and therefore there exists $q \in \mathcal{N}_{\Lambda'}(F_2)$, such that $e' \equiv q$.

Furthermore, for every i, TC-DEPTH$(e_i) < n$, hence we can apply our induction hypothesis, which implies that there exists $e_i' \in \mathcal{N}(F_2 \cup \{^+\})$ for every $i = 1, \ldots, k$, such that $e_i' \equiv e_i$.

Now, if we replace R_{e_i} in q with $e_i'^+$, we obtain an expression $q' \in \mathcal{N}_\Lambda(F_2 \cup \{^+\})$ which is equivalent to e. ■

Note that the proof of the previous theorem does not depend on the semantics of the transitive closure, hence it can be generalized to any nonbasic feature. We do, however, not need this generalization.

Surprisingly, the converse of the previous (non generalized) proposition also holds for languages both in $\mathcal{C}$ or both in $\mathcal{C}[\cap]$. To show this result, we first need the following lemma.

Lemma 4.22: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ languages. Then, $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$ implies $\mathcal{N}(F_1 \cup \{^+\}) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2 \cup \{^+\})$.*

Proof: Suppose that $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$. Then, by definition there exists a query $q \in \mathcal{N}(F_1)$ and two finite graphs G_1 and G_2 such that $q(G_1) = \emptyset$, $q(G_2) \neq \emptyset$, and for every $e \in \mathcal{N}(F_2)$, $e(G_1)$ and $e(G_2)$ are both empty or nonempty simultaneously. We will now show that $\mathcal{N}(F_1 \cup \{^+\}) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2 \cup \{^+\})$. Notice that $q \in \mathcal{N}(F_1 \cup \{^+\})$ since $q \in \mathcal{N}(F_1)$. Now, suppose that e' is an arbitrary expression in $\mathcal{N}(F_2 \cup \{^+\})$. The main goal of the proof is to find an expression $e'' \in \mathcal{N}(F_2)$ which is equivalent to e' on the input graphs G_1 and G_2. Hence then e'' cannot distinguish G_1 from G_2 which is exactly what we want to prove. To this end, remember that Lemma 4.10 tells us that if we only consider graphs whose active domain size is bounded by a fixed n, we can compute the transitive closure of those graphs with the expression $\cup_{i=1}^{n} R^i$. Furthermore, since the output of a query is a graph in particular, an analogue result holds for queries. Therefore, if we replace each occurrence of the transitive closure operator f^+ in the query e' with $\cup_{i=1}^{n} f^i$ (where $n = max(|G_1|, |G_2|)$), we obtain a new query $e'' \in \mathcal{N}(F_2)$, which is equivalent to e'' for input graphs G_1 and G_2. ■

Proposition 4.23: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ both in $\mathcal{C}$ or both in $\mathcal{C}[\cap]$. Then, $\mathcal{N}(F_1 \cup \{^+\}) \leq^{\text{path}} \mathcal{N}(F_2 \cup \{^+\})$ implies $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$.*

PROOF: We will prove this proposition by contraposition. To this end, assume that $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$. Since Proposition 4.5, Proposition 4.6, Proposition 4.7 and Proposition 4.8 (also Proposition 4.17 and Proposition 4.18 if the languages are in $\mathcal{C}[\cap]$) cover all cases such that $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$ — either $\mathcal{N}(F_1) \not\leq^{\text{path}}_{\text{strong}} \mathcal{N}(F_2)$, $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$ or $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$ hold.

First, let us suppose that $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$. Then, Lemma 4.22 tells us that $\mathcal{N}(F_1 \cup \{^+\}) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2 \cup \{^+\})$, which implies that $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$ as desired.

On the other hand suppose that $\mathcal{N}(F_1) \not\leq^{\text{path}}_{\text{strong}} \mathcal{N}(F_2)$. Then, there exists a query $q \in \mathcal{N}(F_1)$ and a finite graph G such that for every $e \in \mathcal{N}(F_2)$, $e(G) \neq q(G)$. Also, note that by definition $q \in \mathcal{N}(F_1 \cup \{^+\})$. Now, suppose that e' is an arbitrary expression in $\mathcal{N}(F_2 \cup \{^+\})$. As in the previous case, e' can be transformed into an expression $e'' \in \mathcal{N}(F_2)$, such that $e'(G) = e''(G)$, hence $e'(G) \neq q(G)$.

The only case left to consider is the case where $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$. Here, $\pi \in \overline{F_1}$ and $F_2 \subseteq \{^{-1}, di\}$. Therefore, $\pi \in \overline{F_1} \cup \{^+\}$ and $F_2 \cup \{^+\} \subseteq \{^{-1}, di, ^+\}$. Hence, we can use Proposition 4.8, which implies $\mathcal{N}(F_1 \cup \{^+\}) \not\leq^{\text{bool}} \mathcal{N}(F_2 \cup \{^+\})$. ■

Theorem 4.4 for languages both in $\mathcal{C}[^+]$ or both in $\mathcal{C}[^+, \cap]$ is now a direct corollary to the previous proposition.

Corollary 4.24: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be both in $\mathcal{C}[^+]$ or both in $\mathcal{C}[\cap, ^+]$. If $F_1 \not\subseteq \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$.*

PROOF: By definition $^+ \in F_1$ and $^+ \in F_2$ since $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ both are in $\mathcal{C}[^+]$ or both in $\mathcal{C}[^+, \cap]$. Hence, $F_1 \setminus \{^+\} \not\subseteq \overline{F_2 \setminus \{^+\}}$. Moreover, $\mathcal{N}(F_1 \setminus \{^+\})$ and $\mathcal{N}(F_2 \setminus \{^+\})$ are both in $\mathcal{C}$ or both in $\mathcal{C}[\cap]$, and thus $\mathcal{N}(F_1 \setminus \{^+\}) \not\leq^{\text{path}} \mathcal{N}(F_2 \setminus \{^+\})$ by Proposition 4.9 in the former and by Proposition 4.19 the latter scenario. Now Proposition 4.23 implies that $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$ as desired. ■

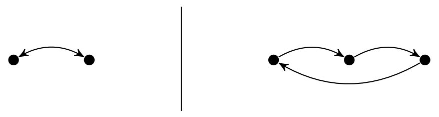


Figure 4.7: Graphs used to establish separation in Proposition 4.25(a).

4.5 Cross-relationships between $\mathcal{C}$, $\mathcal{C}[\cap]$, $\mathcal{C}[^+]$ and $\mathcal{C}[\cap, ^+]$

Now that we have established Theorem 4.4 for languages in the same classes, we have to link all those results together. Thus, in this section we will show that Theorem 4.4 also holds for languages over different classes. To this end, we first need some technical propositions and lemmas.

Proposition 4.25: *Let F_1 and F_2 be sets of nonbasic features. If $\cap \in \overline{F_1}$ and $\cap \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

Proof: The graphs in Figure 4.7 are distinguishable in $\mathcal{N}(F_1)$ by $R^2 \cap id$. They are, however, not distinguishable in $\mathcal{N}(F_2)$; this result was acquired by the brute-force method introduced in Section 3.1. ∎

We will now cite a well known theorem, which we will need to separate languages $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ such that $^+ \in F_1$ and $^+ \notin F_2$. For a proof see for example [AF90].

Theorem 4.26: *There is exists no expression $\varphi(x, y)$ in first order logic such that for all binary relations R and for all $a, b \in \text{adom}(R)$: $R \models \varphi[a, b]$ if and only if there is a path from a to b in R.*

We will use this fact to show that the boolean query $S \circ R^+ \circ T \neq \emptyset$ is not expressible in first order logic.

Lemma 4.27: *Let S, R and T be edge labels. Then the boolean query $S \circ R^+ \circ T \neq \emptyset$ is not expressible in first order logic.*

Proof: Suppose for the sake of contradiction that ψ is the first order sentence over the vocabulary $\{R, S, T\}$ that expresses $S \circ R^+ \circ T \neq \emptyset$. Now, define φ as follows: let φ be ψ and replace every occurrence of $S(u, v)$ with $u = x \wedge v = x$ and every occurrence of $T(u', v')$ with $u' = y \wedge v' = y$, where x and y are variables not yet present in ψ. Notice that φ is a first order formula with two free variables over the vocabulary $\{R\}$.

We will now show that for every binary relation R and for every $a, b \in \text{adom}(R)$:

$$R \models \varphi[a, b] \iff \{R, S = \{(a, a)\}, T = \{(b, b)\}\} \models \psi.$$

To this end let R be a binary relation, let $a, b \in \text{adom}(R)$, let $S = \{(a, a)\}$ and let $T = \{(b, b)\}$. Now, substituting every occurrence $S(u, v)$ in ψ with $u = a \wedge v = a$, and every occurrence $T(u', v')$ in ψ with $u' = b \wedge v' = b$, we get $\varphi[a, b]$. However, since

$$\begin{aligned} S(u, v) &\iff u = a \wedge v = a \\ T(u', v') &\iff u' = b \wedge v' = b \end{aligned}$$

and since the remainder of the sentence remains unaltered, the truth values of $\psi[a,b]$ and φ are equal.

Now, we will show that $\varphi(x,y)$ expresses the reachability query. To this end, let R be some binary relation, let $a, b \in \text{adom}(R)$, let $S = \{(a,a)\}$ and $T = \{(b,b)\}$. Then,

$$\begin{aligned}\text{There exists a path from } a \text{ to } b \text{ in } R &\iff (a,b) \in R^+ \\ &\iff S \circ R^+ \circ T \neq \emptyset \\ &\iff \{R, S, T\} \models \psi \\ &\iff R \models \varphi[a,b]\end{aligned}$$

which contradicts Theorem 4.26, and therefore no such ψ can exist. ■

It is a well known fact that transitive closure is not expressible in first order logic ([AU79, Gys12]). Furthermore remember that Proposition 2.6 tells us that every expression in a language not containing transitive closure can be expressed in first order logic. Therefore, it should not come as a surprise that languages with transitive closure have more expressive power than languages without transitive closure for unrestricted input graphs. The following proposition confirms this intuition.

Proposition 4.28 (Primitivity of $^+$): *Let F_1 and F_2 be sets of nonbasic features. If $^+ \in \overline{F_1}$ and $^+ \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

Proof: All queries expressible in $\mathcal{N}(F_2)$ are also expressible in first order logic by Proposition 2.6. However, $S \circ R^+ \circ T$ is a query expressible in $\mathcal{N}(F_1)$ which is not expressible in first order logic due to Lemma 4.27. Thus, we have found an expression in $\mathcal{N}(F_1)$ which is not expressible in $\mathcal{N}(F_2)$. Hence $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$ as desired. ■

Now we are ready to prove Theorem 4.4 for languages over different classes.

Proposition 4.29: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be in different classes among $\mathcal{C}$, $\mathcal{C}[\cap]$, $\mathcal{C}[^+]$ or $\mathcal{C}[\cap, ^+]$. If $F_1 \nsubseteq \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$.*

Proof: We will consider combinations of languages among different classes. To this end, we will consider all valid combinations one by one.

Let $\mathcal{N}(F_1)$ be in $\mathcal{C}$ and let $\mathcal{N}(F_2)$ be in an arbitrary other class (not $\mathcal{C}$). Clearly, $F_1 \nsubseteq \overline{F_2}$ if and only if $F_1 \nsubseteq \overline{F_2} \setminus \{^+, \setminus, \cap\}$. Hence at least one of $\pi, \overline{\pi}, di, ^{-1}$ is present in F_1 but missing in $\overline{F_2}$. We will now consider a few scenarios:

- If that feature is di or $^{-1}$, either Proposition 4.5 or Proposition 4.7 gives us the desired result.
- Suppose that π is in F_1, but missing in $\overline{F_2}$. If $\cap \in \overline{F_2}$, then $^{-1}, di, \overline{\pi} \notin \overline{F_2}$ and thus, $F_2 \subseteq \{\setminus, \cap, ^+\}$. Now, Proposition 4.18 can be applied, which proves what was asked.

  On the other hand, if $\cap \notin \overline{F_2}$, then $\setminus, \overline{\pi} \notin \overline{F_2}$. Hence, $F_2 \subseteq \{^{-1}, di, ^+\}$ and thus Proposition 4.8 can be applied, which gives us the desired result.

– Suppose that $\overline{\pi}$ is present in F_1, but lacking in $\overline{F_2}$. Then, at least one of π or $\setminus$ is missing in $\overline{F_2}$. If $\setminus$ is missing, we can apply Proposition 4.6. If not, then π is missing in $\overline{F_2}$. Since $\pi \in \overline{F_1}$, the previous case covers what has to be proven.

Let $\mathcal{N}(F_1)$ be in $\mathcal{C}[^+]$ and let $\mathcal{N}(F_2)$ be in $\mathcal{N}[\cap, ^+]$. Then, $F_1 \not\subseteq \overline{F_2}$ if and only if $F_1 \setminus \{^+\} \not\subseteq \overline{F_2} \setminus \{^+, \setminus, \cap\}$. From now on, the proof is exactly the same as the previous case.

Let $\mathcal{N}(F_1)$ be in $\mathcal{C}[\cap]$ and let $\mathcal{N}(F_2)$ be in $\mathcal{N}[\cap, ^+]$. If $\setminus \in F_1$ and $\setminus \notin \overline{F_2}$, we can apply Proposition 4.17. On the other hand if $\setminus$ is present or lacking in both F_1 and $\overline{F_2}$ simultaneously, then $F_1 \not\subseteq \overline{F_2}$ if and only if $F_1 \setminus \{\setminus, \cap\} \not\subseteq \overline{F_2} \setminus \{\setminus, \cap, ^+\}$. Here, the proof of case 1 can be used to prove the required result.

In the other not yet considered valid scenarios for $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$, either $\cap \in F_1$ and $\cap \notin \overline{F_2}$; or $^+ \in F_1$ and $^+ \notin \overline{F_2}$. In the first scenario we can apply Proposition 4.25, in the latter case we can apply Proposition 4.28, which imply our result. ∎

4.6 Proof of Theorem 4.4

Since Proposition 4.3 proves the if direction, the only thing left to prove is the only if direction of Theorem 4.4. To this end consider its contrapositive, i.e., we want to show that $F_1 \not\subseteq \overline{F_2}$ implies $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$ for arbitrary sets of nonbasic features F_1 and F_2. Throughout Sections 4.2, 4.3 and 4.4 we established separation for languages within the same classes. Furthermore, in Section 4.5 we established separation for languages over different classes.

Now, let us recite these results. In Proposition 4.9 we established path separation for languages in $\mathcal{C}$. This yields the Hasse diagram displayed in Figure 4.8a, where there is a directed path from $\mathcal{N}(F_1)$ to $\mathcal{N}(F_2)$ if and only if $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$. Note that the boxed features represent the minimal set of nonbasic features from which the other features can be derived by the appropriate interdependencies discusses in Section 4.1.

We established path separation for languages in $\mathcal{C}[\cap]$ in Proposition 4.19, which yields the Hasse diagram displayed in Figure 4.8b.

Corollary 4.24 tells us that adding $^+$ as a primitive to every language (i.e., as a boxed feature) in the Hasse diagram of $\leq^{\text{path}}$ for $\mathcal{C}$ yields the Hasse diagram for $\leq^{\text{path}}$ of $\mathcal{C}[^+]$. Doing the same with the Hasse diagram of $\leq^{\text{path}}$ for $\mathcal{C}[\cap]$ yields the Hasse diagram for $\leq^{\text{path}}$ of $\mathcal{C}[\cap, ^+]$.

Proposition 4.29 ties all these results together, yielding Theorem 4.4. We can obtain the Hasse diagram for general path queries from the Hasse diagrams of $\leq^{\text{path}}$ for $\mathcal{C}$, $\mathcal{C}[\cap]$, $\mathcal{C}[^+]$ and $\mathcal{C}[\cap, ^+]$ by adding inclusion arrows.

$$\begin{tikzcd}
 & \mathcal{N}(\boxed{di, \overline{\pi}}, \pi) \arrow[rr] & & \mathcal{N}(\boxed{^{-1}, di, \overline{\pi}}, \pi) \\
\mathcal{N}(\boxed{\overline{\pi}}, \pi) \arrow[rr] \arrow[ur] & & \mathcal{N}(\boxed{^{-1}, \overline{\pi}}, \pi) \arrow[ur] & \\
 & \mathcal{N}(\boxed{di, \pi}) \arrow[rr] \arrow[uu] & & \mathcal{N}(\boxed{^{-1}, di, \pi}) \arrow[uu] \\
\mathcal{N}(\boxed{\pi}) \arrow[rr] \arrow[ur] \arrow[uu] & & \mathcal{N}(\boxed{^{-1}, \pi}) \arrow[ur] \arrow[uu] & \\
 & \mathcal{N}(\boxed{di}) \arrow[rr] \arrow[uu] & & \mathcal{N}(\boxed{^{-1}, di}) \arrow[uu] \\
\mathcal{N} \arrow[rr] \arrow[ur] \arrow[uu] & & \mathcal{N}(\boxed{^{-1}}) \arrow[ur] \arrow[uu] &
\end{tikzcd}$$

(a) For $\mathcal{C}$.

$$\begin{tikzcd}
 & & \mathcal{N}(\boxed{\setminus, di}, \cap, \pi, \overline{\pi}) \arrow[rr] & & \mathcal{N}(\boxed{\setminus, ^{-1}, di}, \cap, \pi, \overline{\pi}) \\
 & \mathcal{N}(\boxed{\setminus, \pi}, \cap, \overline{\pi}) = \mathcal{N}(\boxed{\setminus, \overline{\pi}}, \cap, \pi) \arrow[rr] \arrow[ur] & & \mathcal{N}(\boxed{\setminus, ^{-1}}, \cap, \pi, \overline{\pi}) \arrow[ur] & \\
\mathcal{N}(\boxed{\setminus}, \cap) \arrow[ur] & & \mathcal{N}(\boxed{\cap, di, \overline{\pi}}, \pi) \arrow[rr] \arrow[uu] & & \mathcal{N}(\boxed{\cap, ^{-1}, di, \overline{\pi}}, \pi) \arrow[uu] \\
 & \mathcal{N}(\boxed{\cap, \overline{\pi}}, \pi) \arrow[rr] \arrow[ur] \arrow[uu] & & \mathcal{N}(\boxed{\cap, ^{-1}, \overline{\pi}}, \pi) \arrow[ur] \arrow[uu] & \\
 & & \mathcal{N}(\boxed{\cap, di}, \pi) \arrow[rr] \arrow[uu] & & \mathcal{N}(\boxed{\cap, ^{-1}, di}, \pi) \arrow[uu] \\
 & \mathcal{N}(\boxed{\cap, \pi}) \arrow[rr] \arrow[ur] \arrow[uu] & & \mathcal{N}(\boxed{\cap, ^{-1}}, \pi) \arrow[ur] \arrow[uu] & \\
\mathcal{N}(\boxed{\cap}) \arrow[ur] \arrow[uuuu] & & & &
\end{tikzcd}$$

(b) For $\mathcal{C}[\cap]$.

Figure 4.8: Hasse diagrams of $\leq^{\text{path}}$ characterization. The Hasse diagram of $\leq^{\text{path}}$ for $\mathcal{C}[^+]$ (resp. $\mathcal{C}[\cap, ^+]$) can be obtained by adding $^+$ as a boxed feature to every language in the Hasse diagram for $\mathcal{C}$ (resp. $\mathcal{C}[\cap]$).

5 Separation of Boolean Queries

In the previous chapter we characterized $\leq^{\text{path}}$. In this chapter we will do the same for $\leq^{\text{bool}}$. To this end, remember that $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$ implies $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$, hence we can reuse some results of the previous chapter. The converse, however, is not true as we will see later. Therefore, we have to reexamine the pair of languages where $\mathcal{N}(F_1) \not\leq^{\text{path}} \mathcal{N}(F_2)$ to conclude whether $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$.

First, however, we need a few technical preliminaries. The following proposition tells us that for every expression, we can push all the $^{-1}$ applications to the edge labels in a certain way to yield an equivalent expression. This result will simplify certain proofs. The result, however, is interesting in its own right.

Lemma 5.1: *Let F be a set of nonbasic features such that $^{-1} \in F$. For every $e \in \mathcal{N}(F)$, there exists a query $e' \in \mathcal{N}(F)$ such that $e' \equiv e$ and $^{-1}$ is only applied to edge labels.*

Proof: We will prove this proposition by induction on the size of e.
Induction Basis: If $e \in \{R, id, di\}$ we have nothing to prove.
Induction Hypothesis: Assume that our proposition is true for any expression e' that has size less than n.
Induction Step: We will now show that our proposition holds for expressions with size $e = n$.

- $e = e_1 \Delta e_2$, where Δ is $\cap$ or $\cup$. Then, $(e_1 \Delta e_2)^{-1} \equiv e_1^{-1} \Delta e_2^{-1}$. Our induction hypothesis tells us that there exists $e_1', e_2' \in \mathcal{N}(F)$ such that $^{-1}$ is only applied to the edge labels and $e_1' \equiv e_1^{-1}$ and $e_2' \equiv e_2^{-1}$. Clearly this property also holds for $e_1' \Delta e_2' \equiv e_1^{-1} \Delta e_2^{-1}$.

- $e = e_1 \circ e_2$. We will prove that $(e_1 \circ e_2)^{-1} \equiv e_2^{-1} \circ e_1^{-1}$. Let G be some arbitrary graph. Then,

$$\begin{aligned}(m,n) \in (e_1 \circ e_2)^{-1}(G) &\implies (n,m) \in e_1 \circ e_2(G)\\ &\implies \exists k : (n,k) \in e_1(G) \wedge (k,m) \in e_2(G)\\ &\implies (k,n) \in e_1^{-1}(G) \wedge (m,k) \in e_2^{-1}(G)\\ &\implies (m,n) \in e_2^{-1} \circ e_1^{-1}(G).\end{aligned}$$

On the other hand,

$$\begin{aligned}(m,n) \in e_2^{-1} \circ e_1^{-1}(G) &\implies \exists k : (m,k) \in e_2^{-1}(G) \wedge (k,n) \in e_1^{-1}(G)\\ &\implies (k,m) \in e_2(G) \wedge (n,k) \in e_1(G)\\ &\implies (n,m) \in e_1 \circ e_2(G)\\ &\implies (m,n) \in (e_1 \circ e_2)^{-1}(G).\end{aligned}$$

By our induction hypothesis there exists $e_1', e_2' \in \mathcal{N}(F)$ such that $^{-1}$ is only applied to edge labels, $e_1' \equiv e_1^{-1}$ and $e_2' \equiv e_2^{-1}$. Hence $e_2' \circ e_1' \equiv e_2^{-1} \circ e_1^{-1}$ holds the same property.

– $e = e_1 \setminus e_2$. First, we will prove that $(e_1 \setminus e_2)^{-1} \equiv e_1^{-1} \setminus e_2^{-1}$. Then,

$$\begin{aligned}(m,n) \in (e_1 \setminus e_2)^{-1}(G) &\iff (n,m) \in e_1 \setminus e_2(G)\\ &\iff (n,m) \in e_1(G) \wedge \neg\exists(n,m) \in e_2(G)\\ &\iff (m,n) \in e_1^{-1}(G) \wedge \neg\exists(m,n) \in e_2^{-1}(G)\\ &\iff (m,n) \in e_1^{-1} \setminus e_2^{-1}(G).\end{aligned}$$

By our induction hypothesis there exist $e_1', e_2' \in \mathcal{N}(F)$ that have the desired property such that $e_1' \equiv e_1^{-1}$ and $e_2' \equiv e_2^{-1}$. Hence $e_1' \setminus e_2' \equiv e_1^{-1} \setminus e_2^{-1}$ has the desired property.

– $e = \pi_i(e_1)$ or $\overline{\pi}_i(e_1)$. Clearly, $\pi_i(e_1)^{-1} \equiv \pi_i(e_1)$ (resp. $\overline{\pi}_i(e_1)^{-1} \equiv \overline{\pi}_i(e_1)$). Again, by our induction hypothesis there exists $e_1' \in \mathcal{N}(F)$ which has the desired property such that $e_1 \equiv e_1'$.

– $e = e_1^+$. For every graph G the following holds.

$$\begin{aligned}(e_1^+)^{-1}(G) &= (e_1^+(G))^{-1}\\ &= (\cup_{i\geq 1} e_1^i(G))^{-1}\\ &\equiv \cup_{i\geq 1}(e_1^i(G)^{-1})\\ &\equiv \cup_{i\geq 1}(e_1^{-1})^i(G)\end{aligned}$$

Hence $(e_1^+)^{-1} \equiv (e_1^{-1})^+$. Now, by our induction hypothesis, there exists e_1' equivalent to e_1^{-1}, such that $^{-1}$ is only applied to edge labels in e_1'. ■

Before we continue to the first separation result in this chapter for boolean queries, we need two lemmas which depict query equivalences which will be used frequently.

Lemma 5.2: *Let e_1 and e_2 be arbitrary expressions. Then,*

$$id \setminus (e_1 \cup e_2) \equiv (id \setminus e_1) \circ (id \setminus e_2).$$

PROOF: Let G be some arbitrary graph. Then,

$$\begin{aligned}
(m,m) \in id \setminus (e_1 \cup e_2)(G) &\iff (m,m) \notin e_1(G) \cup e_2(G)\\
&\iff (m,m) \notin e_1(G) \wedge (m,m) \notin e_2(G)\\
&\iff (m,m) \in id \setminus e_1(G) \wedge (m,m) \in id \setminus e_2(G)\\
&\iff (m,m) \in (id \setminus e_1) \circ (id \setminus e_2)(G). \qquad \blacksquare
\end{aligned}$$

Lemma 5.3: *Let e_1 and e_2 be arbitrary expressions. Then,*

$$id \setminus (e_1 \circ e_2) \equiv (id \setminus e_1) \cup (id \setminus e_2).$$

PROOF: Let G be some arbitrary graph. Then,

$$\begin{aligned}
(m,m) \in id \setminus (e_1 \circ e_2)(G) &\iff (m,m) \notin e_1(G) \circ e_2(G)\\
&\iff (m,m) \notin e_1(G) \vee (m,m) \notin e_2(G)\\
&\iff (m,m) \in (id \setminus e_1) \cup (id \setminus e_2)(G). \qquad \blacksquare
\end{aligned}$$

The following lemma will be key to show that under certain conditions $^{-1}$ does not add expressive power for boolean queries.

Lemma 5.4: *Let F be a set of nonbasic features such that $\cap \notin \overline{F}$ and $^{+} \notin \overline{F}$. For any $e \in \mathcal{N}(F \cup \{^{-1}, \pi\})$ the following holds:*

- *$\pi_i(e)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$;*
- *If $\overline{\pi} \in \overline{F}$, then $\overline{\pi}_i(e)$ is expressible in $\mathcal{N}(F)$.*

PROOF: We will prove this proposition by induction on the size of e.
Induction Basis:

- $e \in \{id, di, R\}$. Clearly $\pi_i(e) \in \mathcal{N}(F \cup \{\pi\})$. Also, by definition $\overline{\pi}_i(e)$ is expressible in $\mathcal{N}(F)$.
- $e = R^{-1}$. By definition of π, it is clear that $\pi_1(R^{-1}) \equiv \pi_2(R)$ and $\pi_2(R^{-1}) \equiv \pi_1(R)$. Hence $\pi_i(e)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. On the other hand,

$$\overline{\pi}_1(R^{-1}) \equiv id \setminus \pi_1(R^{-1}) \equiv id \setminus \pi_2(R) \equiv \overline{\pi}_2(R).$$

For analogous reasons, $\overline{\pi}_2(R^{-1}) \equiv \overline{\pi}_1(R)$, and hence $\overline{\pi}_i(e)$ is expressible in $\mathcal{N}(F)$.

Induction Hypothesis: Assume that our proposition is true for any expression e' of size less than n.
Induction Step: We will show that our proposition holds for any expression e of size n.

- $e = \pi_j(e_1)$. Applying a projection operation on top of an expression whose result only contains self edges leaves the expression result invariant. Thus $\pi_i(\pi_j(e_1)) \equiv \pi_j(e_1)$, and hence by our induction hypothesis $\pi_i(e)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$.

  On the other hand, $\overline{\pi}_i(\pi_j(e_1)) \equiv id \setminus \pi_i(\pi_j(e_1)) \equiv id \setminus \pi_j(e_1) \equiv \overline{\pi}_j(e_1)$. Now, by our induction hypothesis, $\overline{\pi}_j(e_1)$ is expressible in $\mathcal{N}(F)$, hence $\overline{\pi}_i(e)$ as well.

- $e = \overline{\pi}_j(e_1)$. Note that in this case $\overline{\pi} \in F$. Using the argument in the previous case, $\pi_i(\overline{\pi}_j(e_1)) \equiv \overline{\pi}_j(e_1)$. By our induction hypothesis, $\overline{\pi}_j(e_1)$ is expressible in $\mathcal{N}(F)$, and hence it is also expressible in $\mathcal{N}(F \cup \{\pi\})$.

  On the other hand note that

  $$\overline{\pi}_i(\overline{\pi}_j(e_1)) \equiv \overline{\pi}_i(id \setminus \pi_j(e_1)) \equiv id \setminus \pi_i(id \setminus \pi_j(e_1)) \equiv id \setminus (id \setminus \pi_j(e_1)) \equiv \pi_j(e_1).$$

  Now, our hypothesis tells us that $\pi_j(e_1)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. Furthermore, the presence of $\overline{\pi}$ in $\overline{F}$ implies $\pi \in \overline{F}$, and thus $F \cup \{\pi\} \subseteq \overline{F}$. Moreover Theorem 4.4 tells us that $\mathcal{N}(F \cup \{\pi\}) \leq^{\text{path}} \mathcal{N}(F)$ and hence $\pi_j(e_1)$ is expressible in $\mathcal{N}(F)$.

- $e = e_1 \cup e_2$. Clearly $\pi_i(e_1 \cup e_2) \equiv \pi_i(e_1) \cup \pi_i(e_2)$. Now, by our induction hypothesis, both $\pi_i(e_1)$ and $\pi_i(e_2)$ are expressible in $\mathcal{N}(F \cup \{\pi\})$, hence $\pi_i(e_1) \cup \pi_i(e_2) \equiv \pi_i(e_1 \cup e_2)$ is also expressible in $\mathcal{N}(F \cup \{\pi\})$.

  On the other hand,

  $$\begin{aligned} \overline{\pi}_i(e_1 \cup e_2) &\equiv id \setminus \pi_i(e_1 \cup e_2) \\ &\equiv id \setminus (\pi_i(e_1) \cup \pi_i(e_2)) \\ &\equiv (id \setminus \pi_i(e_1)) \circ (id \setminus \pi_i(e_2)) && \text{(see Lemma 5.2)} \\ &\equiv \overline{\pi}_i(e_1) \circ \overline{\pi}_i(e_2). \end{aligned}$$

  Now, our induction hypothesis tells us that both $\overline{\pi}_i(e_1)$ and $\overline{\pi}_i(e_2)$ are expressible in $\mathcal{N}(F)$, and hence $\overline{\pi}_i(e_1) \cup \overline{\pi}_i(e_2) \equiv \overline{\pi}_i(e_1 \cup e_2)$ is also expressible in $\mathcal{N}(F)$.

- $e = e_1 \circ e_2$. First, we will consider $\pi_1(e)$ and $\overline{\pi}_1(e)$. To this end, let n be the first node in the preorder of the syntax tree of e which is not an application of the composition operator. Now, let e_3 be the expression rooted in n and let e_4 be the composition of all the right rooted subexpressions from the parent of n up to the root (in that order). Due to the associativity of the composition operator, $e_3 \circ e_4$ is equivalent to $e_1 \circ e_2$. Hence it will suffice to prove our proposition for $e_3 \circ e_4$. Notice that by construction e_3 is not a composition application, hence why we chose to continue with $e_3 \circ e_4$. We will now go over all possibilities for e_3.

– $e_3 \in \{R, di, id\}$. First, we will prove that $\pi_1(e_3 \circ e_4) \equiv \pi_1(e_3 \circ \pi_1(e_4))$. Let G be some arbitrary graph. Then,

$$\begin{aligned}
(m,m) \in \pi_1(e_3 \circ e_4)(G) &\implies \exists n : (m,n) \in e_3 \circ e_4(G)\\
&\implies \exists k : (m,k) \in e_3(G) \wedge (k,n) \in e_4(G)\\
&\implies (k,k) \in \pi_1(e_4)(G)\\
&\implies (m,k) \in e_3 \circ \pi_1(e_4)(G)\\
&\implies (m,m) \in \pi_1(e_3 \circ \pi_1(e_4))(G)
\end{aligned}$$

$$\begin{aligned}
(m,m) \in \pi_1(e_3 \circ \pi_1(e_4))(G) &\implies \exists n : (m,n) \in e_3 \circ \pi_1(e_4)(G)\\
&\implies (m,n) \in e_3(G) \wedge (n,n) \in e_4(G)\\
&\implies (m,n) \in e_3 \circ e_4(G)\\
&\implies (m,m) \in \pi_1(e_3 \circ e_4)(G).
\end{aligned}$$

By our induction hypothesis, $\pi_1(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. Furthermore, $e_3 \circ \pi_1(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$ since $e_3 \in \mathcal{N}(F)$ and thus $\pi_1(e_3 \circ \pi_1(e_4))$ is also expressible in $\mathcal{N}(F \cup \{\pi\})$.

On the other hand,

$$\overline{\pi}_1(e_3 \circ e_4) \equiv id \setminus \pi_1(e_3 \circ e_4) \equiv id \setminus \pi_1(e_3 \circ \pi_1(e_4)) \equiv \overline{\pi}_1(e_3 \circ \pi_1(e_4)).$$

Now, our induction hypothesis tells us that $\pi_1(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. However, the presence of $\overline{\pi}$ in $\overline{F}$ implies that $\mathcal{N}(F \cup \{\pi\}) \leq^{\text{path}} \mathcal{N}(F)$ and thus $\pi_1(e_4)$ is expressible in $\mathcal{N}(F)$. Therefore, $e_3 \circ \pi_1(e_4)$ is expressible in $\mathcal{N}(F)$ and hence $\overline{\pi}_1(e_3 \circ \pi_1(e_4))$ as well.

– $e_3 = R^{-1}$. We will first show that $\pi_1(R^{-1} \circ e_4) \equiv \pi_2(\pi_1(e_4) \circ R)$. Let G be an arbitrary graph. Then,

$$\begin{aligned}
(m,m) \in \pi_1(R^{-1} \circ e_4)(G) &\implies \exists n : (m,n) \in R^{-1} \circ e_4(G)\\
&\implies \exists k : (k,m) \in R(G) \wedge (k,n) \in e_4(G)\\
&\implies (k,k) \in \pi_1(e_4)(G)\\
&\implies (k,m) \in \pi_1(e_4) \circ R(G)\\
&\implies (m,m) \in \pi_2(\pi_1(e_4) \circ R)(G)
\end{aligned}$$

$$\begin{aligned}
(m,m) \in \pi_2(\pi_1(e_4) \circ R)(G) &\implies \exists n : (n,m) \in \pi_1(e_4) \circ R(G)\\
&\implies (n,n) \in \pi_1(e_4)(G) \wedge (n,m) \in R(G)\\
&\implies \exists k : (n,k) \in e_4(G) \wedge (m,n) \in R^{-1}(G)\\
&\implies (m,k) \in R^{-1} \circ e_4(G)\\
&\implies (m,m) \in \pi_1(R^{-1} \circ e_4)(G).
\end{aligned}$$

By our induction hypothesis, $\pi_1(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$, and hence $\pi_1(e_4) \circ R$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. Therefore, $\pi_2(\pi_1(e_2) \circ R) \equiv \pi_1(e)$ is also expressible in $\mathcal{N}(F \cup \{\pi\})$.

On the other hand,

$$\overline{\pi}_1(R^{-1} \circ e_4) \equiv \mathit{id} \setminus \pi_1(R^{-1} \circ e_4) \equiv \mathit{id} \setminus \pi_2(\pi_1(e_4) \circ R) \equiv \overline{\pi}_2(\pi_1(e_4) \circ R).$$

Our induction hypothesis tells us that $\pi_1(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. Therefore, $\overline{\pi}_1(\pi_1(e_4) \circ R)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. But, since $\overline{\pi} \in \overline{F}$, it follows directly from Theorem 4.4 that $\mathcal{N}(F \cup \{\pi\}) \leq^{\text{path}} \mathcal{N}(F)$ and hence $\overline{\pi}_2(\pi_1(e_4) \circ R)$ is also expressible in $\mathcal{N}(F)$.

– $e_3 = \pi_j(e_5)$. First, we will show that $\pi_1(\pi_j(e_5) \circ e_4) \equiv \pi_j(e_5) \circ \pi_1(e_4)$. Let G be some arbitrary graph. Then,

$$\begin{aligned}(m, m) \in \pi_1(\pi_j(e_5) \circ e_4)(G) &\implies \exists n : (m, n) \in \pi_j(e_5) \circ e_4(G)\\ &\implies (m, m) \in \pi_j(e_5)(G) \wedge (m, n) \in e_4(G)\\ &\implies (m, m) \in \pi_j(e_5) \circ \pi_1(e_4)(G)\end{aligned}$$

$$\begin{aligned}(m, m) \in \pi_j(e_5) \circ \pi_1(e_4)(G) &\implies (m, m) \in \pi_j(e_5)(G) \wedge (m, m) \in \pi_1(e_4)(G)\\ &\implies \exists n : (m, n) \in (e_4)(G)\\ &\implies (m, n) \in \pi_j(e_5) \circ e_4(G)\\ &\implies (m, m) \in \pi_1(\pi_j(e_5) \circ e_4)(G).\end{aligned}$$

Our induction hypothesis tells us that $\pi_j(e_5)$ and $\pi_1(e_4)$ are expressible in $\mathcal{N}(F \cup \{\pi\})$, and hence $\pi_j(e_5) \circ \pi_1(e_4)$ as well.

On the other hand,

$$\begin{aligned}\overline{\pi}_1(\pi_j(e_5) \circ e_4) &\equiv \mathit{id} \setminus \pi_1(\pi_j(e_5) \circ e_4))\\ &\equiv \mathit{id} \setminus (\pi_j(e_5) \circ \pi_1(e_4)) && \text{(see the previous case)}\\ &\equiv (\mathit{id} \setminus \pi_j(e_5)) \cup (\mathit{id} \setminus \pi_1(e_4)) && \text{(see Lemma 5.3)}\\ &\equiv \overline{\pi}_j(e_5) \cup \overline{\pi}_1(e_4).\end{aligned}$$

Our induction hypothesis tells us that $\overline{\pi}_j(e_5)$ and $\overline{\pi}_1(e_4)$ are both expressible in $\mathcal{N}(F)$, and hence $\overline{\pi}_j(e_5) \cup \overline{\pi}_1(e_4)$ as well.

– $e_3 = \overline{\pi}_j(e_5)$. By a similar argument as in the previous case, $\pi_1(\overline{\pi}_j(e_5) \circ e_4) \equiv \overline{\pi}_j(e_5) \circ \pi_1(e_4)$. Now, our induction hypothesis tells us that $\overline{\pi}_j(e_5)$ is expressible in $\mathcal{N}(F)$ and that $\pi_1(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. Hence $\overline{\pi}_j(e_5) \circ \pi_1(e_4)$ is also expressible in $\mathcal{N}(F \cup \{\pi\})$.

On the other hand,

$$\begin{aligned}
\overline{\pi}_1(\overline{\pi}_j(e_5) \circ e_4) &\equiv \mathit{id} \setminus \pi_1(\overline{\pi}_j(e_5) \circ e_4) \\
&\equiv \mathit{id} \setminus (\overline{\pi}_j(e_5) \circ \pi_1(e_4)) \\
&\equiv (\mathit{id} \setminus \overline{\pi}_j(e_5)) \cup (\mathit{id} \setminus \pi_1(e_4)) && \text{(see Lemma 5.3)} \\
&\equiv (\mathit{id} \setminus (\mathit{id} \setminus \pi_j(e_5)) \cup \overline{\pi}_1(e_4) \\
&\equiv \pi_j(e_5) \cup \overline{\pi}_1(e_4).
\end{aligned}$$

By our induction hypothesis, $\pi_j(e_5)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$ and $\overline{\pi}_1(e_4)$ is expressible in $\mathcal{N}(F)$. Moreover, the presence of $\overline{\pi}$ in $\overline{F}$ implies that $\mathcal{N}(F \cup \{\pi\}) \leq^{\text{path}} \mathcal{N}(F)$ due to Theorem 4.4 and thus $\pi_j(e_5) \cup \overline{\pi}_1(e_4)$ is also expressible in $\mathcal{N}(F)$.

– $e_3 = e_5 \cup e_6$. Clearly, $(e_5 \cup e_6) \circ e_4 \equiv (e_5 \circ e_4) \cup (e_6 \circ e_4)$. Hence,

$$\pi_1((e_5 \cup e_6) \circ e_4) \equiv \pi_1((e_5 \circ e_4) \cup (e_6 \circ e_4)).$$

Furthermore, we already know that

$$\pi_1((e_5 \circ e_4) \cup (e_6 \circ e_4)) \equiv \pi_1(e_5 \circ e_4) \cup \pi_1(e_6 \circ e_4).$$

Now, our induction hypothesis tells us that both $\pi_1(e_5 \circ e_4)$ and $\pi_1(e_6 \circ e_4)$ are expressible in $\mathcal{N}(F \cup \{\pi\})$, and hence $\pi_1(e_5 \circ e_4) \cup \pi_1(e_6 \circ e_4)$ as well.

On the other hand,

$$\begin{aligned}
\overline{\pi}_1((e_5 \cup e_6) \circ e_4) &\equiv \overline{\pi}_1((e_5 \circ e_4) \cup (e_6 \circ e_4)) \\
&\equiv \mathit{id} \setminus \pi_1((e_5 \circ e_4) \cup (e_6 \circ e_4)) \\
&\equiv \mathit{id} \setminus (\pi_1(e_5 \circ e_4) \cup \pi_1(e_6 \circ e_4)) \\
&\equiv (\mathit{id} \setminus \pi_1(e_5 \circ e_4)) \circ (\mathit{id} \setminus \pi_1(e_6 \circ e_4)) && \text{(see Lemma 5.2)} \\
&\equiv \overline{\pi}_1(e_5 \circ e_4) \circ \overline{\pi}_1(e_6 \circ e_4).
\end{aligned}$$

Since our induction hypothesis tells us that both $\overline{\pi}_1(e_5 \circ e_4)$ and $\overline{\pi}_2(e_6 \circ e_4)$ are expressible in $\mathcal{N}(F)$, their composition is also expressible in $\mathcal{N}(F)$.

For π_2 and $\overline{\pi}_2$ we will take a slightly different approach. Let n be the first node in the reverse preorder that is not an application of the composition operator. Now, let e_3 be the expression rooted in n and let e_4 be the composition of all left rooted expressions from the parent of n up to the root (in that order). Again, by the associativity of the composition operation, $e_1 \circ e_2$ is equivalent to $e_4 \circ e_3$. We will now consider all possibilities for e_3.

– $e_3 \in \{R, di, id\}$. First, we will prove that $\pi_2(e_4 \circ e_3) \equiv \pi_2(\pi_2(e_4) \circ e_3)$. Let G be some arbitrary graph. Then,

$$\begin{aligned}
(m,m) \in \pi_2(e_4 \circ e_3)(G) &\implies \exists n : (n,m) \in e_4 \circ e_3(G) \\
&\implies \exists k : (n,k) \in e_4(G) \wedge (k,m) \in e_3(G) \\
&\implies (k,k) \in \pi_2(e_4)(G) \\
&\implies (k,m) \in \pi_2(e_4) \circ e_3(G) \\
&\implies (m,m) \in \pi_2(\pi_2(e_4) \circ e_3)(G)
\end{aligned}$$

$$\begin{aligned}
(m,m) \in \pi_2(\pi_2(e_4) \circ e_3)(G) &\implies \exists n : (n,m) \in \pi_2(e_4) \circ e_3(G) \\
&\implies (n,n) \in \pi_2(e_4)(G) \wedge (n,m) \in e_3(G) \\
&\implies \exists k : (k,n) \in e_4(G) \\
&\implies \exists (k,m) \in e_4 \circ e_3(G) \\
&\implies (m,m) \in \pi_2(e_4 \circ e_3)(G).
\end{aligned}$$

Our induction hypothesis tells us that $\pi_2(e_4)$ is expressible in $\mathcal{N}(F)$, hence $\pi_2(e_4) \circ e_3$ is expressible in $\mathcal{N}(F \cup \{\pi\})$ (since $e_3 \in \mathcal{N}(F)$) and thus $\pi_2(\pi_2(e_4) \circ e_3)$ as well.

On the other hand,

$$\overline{\pi}_2(e_4 \circ e_3) \equiv id \setminus \pi_2(e_4 \circ e_3) \equiv id \setminus \pi_2(\pi_2(e_4) \circ e_3) \equiv \overline{\pi}_2(\pi_2(e_4) \circ e_3).$$

Now, our induction hypothesis tells us that $\pi_2(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. However, since $\mathcal{N}(F \cup \{\pi\}) \leq^{\text{path}} \mathcal{N}(F)$, $\pi_2(e_4)$ is expressible in $\mathcal{N}(F)$. Thus, since $e_3 \in \mathcal{N}(F)$, $\pi_2(e_4) \circ e_3$ is expressible in $\mathcal{N}(F)$. Therefore, $\overline{\pi}_2(\pi_2(e_4) \circ e_3)$ is also expressible in $\mathcal{N}(F)$.

– $e_3 = R^{-1}$. First, notice that

$$\pi_2(e_4 \circ R^{-1}) \equiv \pi_2(\pi_2(e_4) \circ R^{-1}) \equiv \pi_2(R \circ \pi_2(e_4)).$$

Now, since $R \in \mathcal{N}(F)$ and since $\pi_2(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$, our induction hypothesis tells us that $R \circ \pi_2(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$, and hence $\pi_2(R \circ \pi_2(e_4))$ as well.

On the other hand,

$$\overline{\pi}_2(e_4 \circ R^{-1}) \equiv id \setminus \pi_2(e_4 \circ R^{-1}) \equiv id \setminus \pi_2(R \circ \pi_2(e_4)) \equiv \overline{\pi}_2(R \circ \pi_2(e_4)).$$

Therefore, since $\pi_2(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$ by our induction hypothesis and since $\mathcal{N}(F \cup \{\pi\}) \leq^{\text{path}} \mathcal{N}(F)$, $\overline{\pi}_2(R \circ \pi_2(e_4))$ is also expressible in $\mathcal{N}(F)$.

– $e_3 = \pi_j(e_5)$. First, we will show that $\pi_2(e_4 \circ \pi_j(e_5)) \equiv \pi_2(e_4) \circ \pi_j(e_5)$. Let G be some arbitrary graph. Then,

$$\begin{aligned}(m,m) \in \pi_2(e_4 \circ \pi_j(e_5))(G) &\implies \exists n : (n,m) \in e_4 \circ \pi_j(e_5)(G)\\ &\implies (n,m) \in e_4(G) \wedge (m,m) \in \pi_j(e_5)(G)\\ &\implies (m,m) \in \pi_2(e_4)(G)\\ &\implies (m,m) \in \pi_2(e_4) \circ \pi_j(e_5)(G)\end{aligned}$$

$$\begin{aligned}(m,m) \in \pi_2(e_4) \circ \pi_j(e_5)(G) &\implies (m,m) \in \pi_2(e_4)(G) \wedge (m,m) \in \pi_j(e_5)(G)\\ &\implies \exists n : (n,m) \in e_4(G)\\ &\implies (n,m) \in e_4 \circ \pi_j(e_5)(G)\\ &\implies (m,m) \in \pi_2(e_4 \circ \pi_j(e_5))(G).\end{aligned}$$

By our induction hypothesis, $\pi_2(e_4)$ and $\pi_j(e_5)$ are expressible in $\mathcal{N}(F \cup \{\pi\})$, and hence $\pi_2(e_4) \circ \pi_j(e_5)$ as well.

On the other hand,

$$\begin{aligned}\overline{\pi}_2(e_4 \circ \pi_j(e_5)) &\equiv id \setminus \pi_2(e_4 \circ \pi_j(e_5))\\ &\equiv id \setminus (\pi_2(e_4) \circ \pi_j(e_5))\\ &\equiv (id \setminus \pi_2(e_4)) \cup (id \setminus \pi_j(e_5)) \qquad \text{(Lemma 5.3)}\\ &\equiv \overline{\pi}_2(e_4) \cup \overline{\pi}_j(e_5).\end{aligned}$$

Our induction hypothesis tells us that both $\overline{\pi}_2(e_4)$ and $\overline{\pi}_j(e_5)$ are expressible in $\mathcal{N}(F)$, and hence their union as well.

– $e_3 = \overline{\pi}_j(e_5)$. Using a similar argument as in the previous case, $\pi_2(e_4 \circ \overline{\pi}_j(e_5)) \equiv \pi_2(e_4) \circ \overline{\pi}_j(e_5)$. Now, by our induction hypothesis, $\pi_2(e_4)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$ and $\overline{\pi}_j(e_5)$ is expressible in $\mathcal{N}(F)$, and hence $\pi_2(e_4) \circ \overline{\pi}_j(e_5)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$.

On the other hand,

$$\begin{aligned}\overline{\pi}_2(e_4 \circ \overline{\pi}_j(e_5)) &\equiv id \setminus \pi_2(e_4 \circ \overline{\pi}_j(e_4))\\ &\equiv id \setminus (\pi_2(e_4) \circ \overline{\pi}_j(e_5))\\ &\equiv (id \setminus \pi_2(e_4)) \cup (id \setminus \overline{\pi}_j(e_5)) \qquad \text{(Lemma 5.3)}\\ &\equiv \overline{\pi}_2(e_4) \cup (id \setminus (id \setminus \pi_j(e_5)))\\ &\equiv \overline{\pi}_2(e_4) \cup \pi_j(e_5).\end{aligned}$$

Our induction hypothesis tells us that $\overline{\pi}_2(e_4)$ is expressible in $\mathcal{N}(F)$ and that $\pi_j(e_5)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. However, since $\overline{\pi} \in \overline{F}$, Theorem 4.4 implies that $\mathcal{N}(F \cup \{\pi\}) \leq^{\text{path}} \mathcal{N}(F)$. Hence $\pi_j(e_5)$ is expressible in $\mathcal{N}(F)$, and therefore $\overline{\pi}_2(e_4) \cup \pi_j(e_5)$ is also expressible in $\mathcal{N}(F)$.

– $e_3 = e_5 \cup e_6$. Obviously, $\pi_2(e_4 \circ (e_5 \cup e_6)) \equiv \pi_2((e_4 \circ e_5) \cup (e_4 \circ e_6))$. Furthermore, we already know that $\pi_2((e_4 \circ e_5) \cup (e_4 \circ e_6)) \equiv \pi_2(e_4 \circ e_5) \cup \pi_2(e_4 \circ e_6)$. Now, by our induction hypothesis, both $\pi_2(e_4 \circ e_5)$ and $\pi_2(e_4 \circ e_6)$ are expressible in $\mathcal{N}(F)$, and hence their union as well.

  On the other hand,

$$\begin{aligned}\overline{\pi}_2(e_4 \circ (e_5 \cup e_6)) &\equiv \overline{\pi}_2((e_4 \circ e_5) \cup (e_4 \circ e_6))\\ &\equiv id \setminus \pi_2((e_4 \circ e_5) \cup (e_4 \circ e_6))\\ &\equiv id \setminus (\pi_2(e_4 \circ e_5) \cup \pi_2(e_4 \circ e_6))\\ &\equiv \overline{\pi}_2(e_4 \circ e_5) \circ \overline{\pi}_2(e_4 \circ e_6). && \text{(see Lemma 5.2)}\end{aligned}$$

  Our induction hypothesis now tells us that both $\overline{\pi}_2(e_4 \circ e_5)$ and $\overline{\pi}_2(e_4 \circ e_6)$ are expressible in $\mathcal{N}(F)$, and hence their composition as well. ■

An important corollary of the previous lemma is that unless languages contain $\cap$ or $^+$, the expressive power of the language remains unaltered after including $^{-1}$ if π is present.

Proposition 5.5 (Collapse of $^{-1}$): *Let F be a set of nonbasic features such that $\cap \notin \overline{F}$ and $^+ \notin \overline{F}$. Then, $\mathcal{N}(F \cup \{^{-1}\}) \leq^{\text{bool}} \mathcal{N}(F \cup \{\pi\})$.*

PROOF: Let e be an expression in $\mathcal{N}(F \cup \{^{-1}\})$. By Lemma 5.4, $\pi_1(e)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. However, notice that for every graph G, $e(G) \neq \emptyset$ if and only if $\pi(e)(G) \neq \emptyset$, which proves the result. ■

Towards the characterization of $\leq^{\text{bool}}$ we will introduce some notation to support the collapse of $^{-1}$ mentioned in the previous proposition.

For a set of nonbasic features F, define $\widetilde{F}$ as follows.

$$\widetilde{F} = \begin{cases} \overline{F} \cup \{^{-1}\}, & \text{if } \pi \in \mathcal{N}(F), \cap \notin \overline{F} \text{ and } ^+ \notin \overline{F} \\ \overline{F}, & \text{otherwise} \end{cases}$$

With this new notation we are ready to supply the first separation result for $\leq^{\text{bool}}$.

Proposition 5.6: *Let F_1 and F_2 be sets of nonbasic features. Then, $F_1 \subseteq \widetilde{F_2}$ implies that $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$.*

PROOF: First, notice that we have two cases, either $\widetilde{F_2} = \overline{F_2}$, or $\widetilde{F_2} = \overline{F_2} \cup \{^{-1}\}$. In the first case, $F_1 \subseteq \widetilde{F_2} = \overline{F_2}$. Hence we can apply Proposition 4.3, which tells us that $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$, which implies $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$.

Let us now consider the latter case. To this end assume that $\widetilde{F_2} = \overline{F_2} \cup \{^{-1}\}$. By the definition of $\widetilde{F_2}$, $\pi \in \overline{F_2}, \cap \notin \overline{F_2}$ and $^+ \notin \overline{F_2}$. Since $\cap \notin \overline{F_2}$, we know that adding $^{-1}$ to $\overline{F_2}$ does not introduce new features in $\overline{F_2}$ other than itself, hence $\overline{F_2} \cup \{^{-1}\} = \overline{F_2 \cup \{^{-1}\}}$. Now, since $F_1 \subseteq \widetilde{F_2} = \overline{F_2} \cup \{^{-1}\} = \overline{F_2 \cup \{^{-1}\}}$, we can apply Proposition 4.3, which tells us that $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2 \cup \{^{-1}\})$. Moreover, $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(\overline{F_2} \cup \{^{-1}\})$ since $F_2 \cup \{^{-1}\} \subseteq \overline{F_2} \cup \{^{-1}\}$, and hence $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(\overline{F_2} \cup \{^{-1}\})$. Now, Proposition 5.5 implies that $\mathcal{N}(\overline{F_2} \cup \{^{-1}\}) \leq^{\text{bool}} \mathcal{N}(\overline{F_2} \cup \{\pi\}) = \mathcal{N}(\overline{F_2})$. However, since $F_2 \subseteq \overline{F_2}$, Theorem 4.4 tells us that $\mathcal{N}(F_2) = \mathcal{N}(\overline{F_2})$, and hence $\mathcal{N}(\overline{F_2} \cup \{^{-1}\}) \leq^{\text{bool}} \mathcal{N}(\overline{F_2} \cup \{\pi\}) = \mathcal{N}(F_2)$. Therefore, by transitivity, $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$ as desired. ∎

The goal is thus to obtain an 'if and only if' characterization of $\leq^{\text{bool}}$ as we did for $\leq^{\text{path}}$ in Chapter 4. Unfortunately, the converse of the previous theorem does not hold, since for example $\mathcal{N}(^{-1}) \leq^{\text{bool}} \mathcal{N}(^+, \pi)$, but $\{^{-1}\} \nsubseteq \widetilde{\{^+, \pi\}} = \{^+, \pi\}$. This problem arises because the previous proposition does not consider transitivity on the left hand side, i.e., it does not consider $\mathcal{N}(^{-1}) \leq^{\text{bool}} \mathcal{N}(\pi) \leq^{\text{bool}} \mathcal{N}(\pi, ^+)$. Thus the tilde notation we introduced before is not rich enough. To this end, we will introduce another notation which we will need alongside the tilde notation. For a set of nonbasic features F define $\widehat{F}$ as follows.

$$\widehat{F} = \begin{cases} (F \setminus \{^{-1}\}) \cup \{\pi\}, & \text{if } ^{-1} \in \overline{F}, \cap \notin \overline{F} \text{ and } ^+ \notin \overline{F} \\ F, & \text{otherwise} \end{cases}$$

Using this notation we obtain a result similar to Proposition 5.6.

Proposition 5.7: *Let F_1 and F_2 be sets of nonbasic features. If $\widehat{F_1} \subseteq \overline{F_2}$, then $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$.*

PROOF: We have two cases, either $\widehat{F_1} = F_1$ or $\widehat{F_1} = (\overline{F_1} \setminus \{^+\}) \cup \{\pi\}$. In the former case, observe that $F_1 \subseteq \overline{F_2}$. Therefore, Proposition 4.3 can be applied, which tells us that $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$, and thus $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$.

Now let us consider the latter case. By definition we know that $\cap$ and $^+$ are not present in $\overline{F_1}$. Therefore, $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(\widehat{F_1})$ by Proposition 5.5. Furthermore, since $\widehat{F_1} \subseteq \overline{F_2}$, we can apply Proposition 4.3, which implies that $\mathcal{N}(\widehat{F_1}) \leq^{\text{path}} \mathcal{N}(F_2)$ and hence also that $\mathcal{N}(\widehat{F_1}) \leq^{\text{bool}} \mathcal{N}(F_2)$. Now, by transitivity, $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$ as desired. ∎

The above proposition solves the transitivity problem of Proposition 5.6. Unfortunately, however, the converse again does not hold, see the following counter example: clearly $\mathcal{N}(di, ^{-1}) \leq^{\text{bool}} \mathcal{N}(di, ^{-1}, ^+)$, but $\widehat{\{di, ^{-1}\}} = \{di, \pi\} \nsubseteq \overline{\{di, ^{-1}, ^+\}} = \{di, ^{-1}, ^+\}$. Also, it does not consider the collapse of $^{-1}$ on the righthand size when both languages are not in $\mathcal{C}$.

It appears that if $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$, the converse of Proposition 5.6 or Proposition 5.7 holds. Thus we get the following theorem.

Theorem 5.8: *Let F_1 and F_2 be sets of nonbasic features. Then, $F_1 \subseteq \widetilde{F_2}$ or $\widehat{F_1} \subseteq \overline{F_2}$ if and only if $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$.*

Towards the proof of the if direction, we will use an approach similar to how we established the characterization of $\leq^{\text{path}}$. Thus, we will first characterize $\leq^{\text{bool}}$ for languages in the same classes

$$\begin{aligned}
\mathcal{C} &= \{\mathcal{N}(F) \mid \cap \notin \overline{F}, {}^+ \notin \overline{F}\} && \text{(Section 4.2)}\\
\mathcal{C}[\cap] &= \{\mathcal{N}(F) \mid \cap \in \overline{F}, {}^+ \notin \overline{F}\} && \text{(Section 5.2)}\\
\mathcal{C}[{}^+] &= \{\mathcal{N}(F) \mid \cap \notin \overline{F}, {}^+ \in \overline{F}\} && \text{(Section 5.3)}\\
\mathcal{C}[\cap, {}^+] &= \{\mathcal{N}(F) \mid \cap \in \overline{F}, {}^+ \in \overline{F}\} && \text{(Section 5.4)}
\end{aligned}$$

and then we will characterize $\leq^{\text{bool}}$ for languages over different classes in Section 5.5.

As we will see later, the converse of Proposition 5.7 does hold when F_1 and F_2 are in the same class. Thus, for languages within the same class we obtain the following less complicated theorem.

Theorem 5.9: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be languages in the same class. Then, $F_1 \subseteq \widetilde{F_2}$ if and only if $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$.*

We will first prove Theorem 5.9 holds.

5.1 Languages without $\cap$ and without $^+$

In this section we will characterize $\leq^{\text{bool}}$ for languages in $\mathcal{C}$, i.e., Theorem 5.9 restricted to $\mathcal{C}$. To achieve this, we first need the following lemma.

Proposition 5.10 (Primitivity of $^{-1}$): *Let F be a set of nonbasic features. If $^{-1} \in \overline{F}$, then $\mathcal{N}(F) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(di, {}^+)$.*

Proof: The graphs in Figure 4.4b are distinguishable in $\mathcal{N}(F)$ by $R^2 \circ R^{-1} \circ R^2$. They are, however, not distinguishable in $\mathcal{N}(di, {}^+)$. This result was established by the brute force method described in Section 3.1. ∎

We are now ready to prove Theorem 5.9 restricted to $\mathcal{C}$.

Proposition 5.11: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be languages in $\mathcal{C}$. If $F_1 \not\subseteq \widetilde{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$.*

Proof: First, note that $F_1 \cup F_2 \subseteq \{{}^{-1}, \overline{\pi}, \pi, di\}$ since $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ are in $\mathcal{C}$. We will consider two 'major' cases: $\pi \in \mathcal{N}(F_2)$ and $\pi \notin \mathcal{N}(F_2)$. First we will consider $\pi \in \mathcal{N}(F_2)$. In this case $\widetilde{F_2} = \mathcal{N}(F_2) \cup \{{}^{-1}\}$ since $\cap, {}^+ \notin \overline{F_2}$ by definition. Since $F_1 \not\subseteq \widetilde{F_2}$, either $\overline{\pi}$ or di is in F_1, but missing in $\widetilde{F_2}$ and thus also missing in $\overline{F_2}$. If di is missing, then Proposition 4.5 proves the result. On the other hand if $\overline{\pi}$ is missing, then Proposition 4.6 proves the desired result.

Now, we will consider the case where $\pi \notin \overline{F_2}$. Here, $\overline{\pi} \notin F_2$ and $F_1 \not\subseteq \widetilde{F_2} = \overline{F_2}$. Hence either one of $di, {}^{-1}, \pi, \overline{\pi}$ is in F_1 but missing in $\overline{F_2}$.

- $\pi \in F_1$ and $\pi \notin \overline{F_2}$. Then, $F_2 \subseteq \{{}^{-1}, di\}$ and hence Proposition 4.8 can be applied, which proves the result.

- $^{-1} \in F_1$ and $^{-1} \notin \overline{F_2}$. Now, $F_2 \subseteq \{di\}$ and thus Proposition 5.10 proves what was asked.
- $di \in F_1$ and $di \notin \overline{F_2}$. In this case Proposition 4.5 can be applied.
- $\overline{\pi} \in F_1$ and $\overline{\pi} \notin \overline{F_2}$. Here, Proposition 4.6 can be applied. ■

5.2 Languages with $\cap$ and without $^+$

In this section we will prove Theorem 5.9 for languages in $\mathcal{C}[\cap]$. To achieve this, we will first construct a new bisimulation technique.

Definition 5.12: Let G_1 en G_2 be graphs, let k be a natural number and let $\widehat{Z} = (Z_1, \dots, Z_k)$ be a tuple of relations, where $Z_i \subseteq \mathrm{adom}(G_1)^2 \times \mathrm{adom}(G_2)^2$ for $i \in \{0, \dots, k\}$. We say that $\widehat{Z}$ is a bisimulation up to k from G_1 to G_2 if the following conditions are satisfied:

- **Atoms:** Assume that $(a_1, b_1, a_2, b_2) \in Z_i$. $a_1 = b_1$ iff $a_2 = b_2$; and $(a_1, b_1) \in G_1(R)$ iff $(a_2, b_2) \in G_2(R)$ for every $R \in \Lambda$;
- **Forth:** Assume that $(a_1, b_1, a_2, b_2) \in Z_i$ where $i > 0$. For every $c_1 \in \mathrm{adom}(G_1)$, there exists $c_2 \in \mathrm{adom}(G_2)$ such that $(a_1, c_1, a_2, c_2) \in Z_{i-1}$ and $(c_1, b_1, c_2, b_2) \in Z_{i-1}$;
- **Back:** Assume that $(a_1, b_1, a_2, b_2) \in Z_i$ where $i > 0$. For every $c_2 \in \mathrm{adom}(G_2)$, there exists $c_1 \in \mathrm{adom}(G_1)$ such that $(a_1, c_1, a_2, c_2) \in Z_{i-1}$ and $(c_1, b_1, c_2, b_2) \in Z_{i-1}$. ◇

For two tuples of relations $\widehat{Z} = (Z_1, \dots, Z_k)$ and $\widehat{W} = (W_1, \dots, W_k)$ of the same length, denote with $\widehat{Z} \cup \widehat{W}$ the pairwise union of $\widehat{Z}$ and $\widehat{W}$, i.e., $\widehat{Z} \cup \widehat{W} = (Z_1 \cup W_1, \dots, Z_k \cup W_k)$. It appears that the union of two bisimulations is again a bisimulation.

Lemma 5.13: *If $\widehat{Z}$ and $\widehat{W}$ are bisimulations from G_1 to G_2 up to level k, then so is $(\widehat{Z} \cup \widehat{W})$.*

Proof: We will first verify the 'atoms' condition. Suppose that $(a_1, b_1, a_2, b_2) \in Z_i \cup W_i$. Then $(a_1, b_1, a_2, b_2) \in Z_i$ or $(a_1, b_1, a_2, b_2) \in W_i$. Thus the atoms condition holds.

For the 'forth' condition, we can use a similar argument. If $i > 0$ and $(a_1, b_1, a_2, b_2) \in Z_i \cup W_i$, then $(a_1, b_1, a_2, b_2) \in Z_i$ or $(a_1, b_1, a_2, b_2) \in W_i$. Without loss of generalization we can assume that $(a_1, b_1, a_2, b_2) \in Z_i$. Now, for every $c_1 \in \mathrm{adom}(G_1)$, there exists $c_2 \in \mathrm{adom}(G_2)$ such that $(a_1, c_1, a_2, c_2) \in Z_{i-1}$ and $(c_1, b_1, c_2, b_2) \in Z_{i-1}$ for all i, since $\widehat{Z}$ is a bisimulation from G_1 to G_2 up to level k. Therefore, $(a_1, c_1, a_2, c_2) \in Z_{i-1} \cup W_{i-1}$ and $(c_1, b_1, c_2, b_2) \in Z_{i-1} \cup W_{i-1}$, since $Z_{i-1} \subseteq Z_{i-1} \cup W_{i-1}$, which proves the 'forth' condition.

The proof for the 'back' condition is completely analogous to the proof for the 'forth' condition. ■

With the following two lemmas we will show that this new bisimulation notion is 'equivalent' with the bisimilarity notion described in Section 3.2. First we will show that if a tuple (a_1, b_1, a_2, b_2) is present in Z_k of a bisimulation $\widehat{Z}$ from G_1 to G_2, then also $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$.

Proposition 5.14: *If there exists a bisimulation $\widehat{Z}$ from G_1 to G_2 up to at least level k such that $(a_1, b_1, a_2, b_2) \in Z_k$, then $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$.*

PROOF: We will prove this proposition by induction on k.
Induction Basis: Let $k = 0$. If $(a_1, b_1, a_2, b_2) \in Z_0$, then by the 'atoms' condition of the bisimulation $a_1 = b_1$ if and only if $a_2 = b_2$; and $(a_1, b_1) \in G_1(R)$ iff $(a_2, b_2) \in G_2(R)$ for every $R \in \Lambda$, which is exactly the 'atoms' condition for bisimilarity. Therefore, since $k = 0$, this is the only condition that has to be satisfied, and hence $(G_1, a_1, b_1) \simeq_0 (G_2, a_2, b_2)$.
Induction Hypothesis: Suppose that our proposition holds for $k = n$, i.e., if there exists a bisimulation $\widehat{Z}$ from G_1 to G_2, such that $(a_1, b_1, a_2, b_2) \in Z_n$, then $(G_1, a_1, b_1) \simeq_n (G_2, a_2, b_2)$.
Induction step: We will now show that our proposition also holds for $k = n + 1$. Suppose that $(a_1, b_1, a_2, b_2) \in Z_{n+1}$. Then, using a similar argument as in our induction basis, the atoms condition for bisimilarity holds, since the argument does not depend on k.

So all that is left to verify is the 'forth' and 'back' conditions for bisimilarity. We will only consider the former, since the proof for the 'back' condition is analogous. Now, for every $c_1 \in \text{adom}(G_1)$, there exists $c_2 \in \text{adom}(G_2)$, such that $(a_1, c_1, a_2, c_2) \in Z_n$ and $(c_1, b_1, c_2, b_2) \in Z_n$, since $(a_1, b_1, a_2, b_2) \in Z_{n+1}$, and $n + 1 > 0$. When we apply our induction hypothesis we get that $(G_1, a_1, c_1) \simeq_n (G_2, a_2, c_2)$ and $(G_1, c_1, b_1) \simeq_n (G_2, c_2, b_2)$, which is exactly the 'forth' condition, and thus concludes our proof. ■

The converse of the previous proposition also holds. Hence the bisimulation and the bisimilarity notation are equivalent.

Proposition 5.15: *If $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$, then there exists a bisimulation $\widehat{Z}$ from G_1 to G_2 up to at least level k such that $(a_1, b_1, a_2, b_2) \in Z_k$.*

PROOF: We will prove this proposition by induction on k.
Induction Basis: If $(G_1, a_1, b_1) \simeq_0 (G_2, a_2, b_2)$, then the atoms condition holds, i.e., $a_1 = b_1$ 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 $R \in \Lambda$. Hence $Z = \{Z_0 = \{(a_1, b_1, a_2, b_2)\}\}$ is a bisimulation from G_1 to G_2 such that $(a_1, b_1, a_2, b_2) \in Z_0$.
Induction Hypothesis: Suppose that our proposition holds for $k = n$, that is if $(G_1, a_1, b_1) \simeq_n (G_2, a_2, b_2)$, then there exists a bisimulation $\widehat{Z}$ from G_1 to G_2 such that $(a_1, b_1, a_2, b_2) \in Z_n$.
Induction Step: We will now show that our proposition holds for $k = n + 1$. To this end, assume that $(G_1, a_1, b_1) \simeq_n (G_2, a_2, b_2)$. Then, by definition, for every $c_1 \in \text{adom}(G_1)$, there exists $c_2 \in \text{adom}(G_2)$, such that $(G_1, a_1, c_1) \simeq_n (G_2, a_2, c_2)$ and

$(G_1, c_1, b_1) \simeq_n (G_2, c_2, b_2)$. Therefore, by our induction hypothesis there exists two bisimulations $\widehat{Z}$ and $\widehat{W}$ from G_1 to G_2, such that $(a_1, c_1, a_2, c_2) \in Z_n$ and $(c_1, b_1, c_2, b_2) \in Z_n$. Now, by Lemma 5.13 $\widehat{Z} \cup \widehat{W}$ is a bisimulation from G_1 to G_2 up to level n. Furthermore, setting the $n+1$'th component to $\{(a_1, b_1, a_2, b_2)\}$ yields a bisimulation from G_1 to G_2 up to level $n+1$ by construction, such that $(a_1, b_1, a_2, b_2) \in Z_{n+1}$ as desired. ■

If two tuples of relations $\widehat{Z} = (Z_1, \ldots, Z_k)$ and $\widehat{W} = (W_1, \ldots, W_k)$ have the same length, we say that $\widehat{Z} \subseteq \widehat{W}$, if $Z_i \subseteq W_i$ for all $i \in \{1, \ldots, k\}$. Using this notation, the set of all bisimulations from G_1 to G_2 up to level k form a partially ordered set. The following proposition tells us that this partial order has a unique maximal element.

Proposition 5.16: *Let G_1 and G_2 be two finite graphs and let k be a natural number. There is a unique maximal bisimulation from G_1 to G_2 up to k, denoted by* $\mathrm{Sim}(G_1, G_2, k)$.

PROOF: The union of all bisimulations from G_1 to G_2 up to level k is a maximal bisimulation. Notice that since G_1 and G_2 are finite, there are only a finite number of bisimulations from G_1 to G_2 up to level k, and hence there is only one such maximal bisimulation from G_1 to G_2 up to level k. ■

It appears that the maximal bisimulation from G_1 to G_2 up to level k has a monotonically decreasing structure. This property will be useful later on.

Lemma 5.17: *The maximal bisimulation $\widehat{Z} = \mathrm{Sim}(G_1, G_2, k)$ up to level k from G_1 to G_2 is monotonically decreasing, i.e., $Z_i \supseteq Z_{i+1}$ for $i \in \{0, \ldots, k-1\}$.*

PROOF: Let $(a_1, b_1, a_2, b_2) \in Z_{i+1}$. Proposition 5.14 implies that $(G_1, a_1, b_1) \simeq_{i+1} (G_2, a_2, b_2)$, since $\widehat{Z}$ is a bisimulation up to level $i+1$. Now, Lemma 3.7 tells us that $(G_1, a_1, b_1) \simeq_i (G_2, a_2, b_2)$, which by Proposition 5.15 implies the existence of a bisimulation $\widehat{W}$ from G_1 to G_2 such that $(a_1, b_1, a_2, b_2) \in W_i$. Moreover, since $\mathrm{Sim}(G_1, G_2, k)$ is the unique maximal bisimulation from G_1 to G_2 up to k, it is clear that $W_i \subseteq Z_i$ and thus $(a_1, b_1, a_2, b_2) \in Z_i$ as desired. ■

Later we will have to compute the maximal bisimulation from G_1 to G_2 up to a certain level k. We will do this by refining an already existing bisimultation from G_1 to G_2 up to level $k-1$ to one of level k.

Definition 5.18: Let G_1 and G_2 be graphs, let k be a natural number, and let $Z \subseteq \mathrm{adom}(G_1)^2 \times \mathrm{adom}(G_2)^2$. Define $\mathrm{Refine}(Z)$ as the set Z' of tuples $(a_1, b_1, a_2, b_2) \in Z$ that satisfy the following conditions:

- **Forth:** For every $c_1 \in \mathrm{adom}(G_1)$, there exists $c_2 \in \mathrm{adom}(G_2)$, such that

$$(a_1, c_1, a_2, c_2) \in Z \text{ and } (c_1, b_1, c_2, b_2) \in Z.$$

– **Back:** For every $c_2 \in \mathrm{adom}(G_2)$, there exists $c_1 \in \mathrm{adom}(G_1)$, such that

$$(a_1, c_1, a_2, c_2) \in Z \text{ and } (c_1, b_1, c_2, b_2) \in Z. \qquad \diamond$$

The following proposition tells us that the refine operation does exactly what it is supposed to, i.e., it refines an already existing bisimultation up to one level higher. It also tells us that the refine operation preserves maximality.

Lemma 5.19: *Let $\widehat{Z}$ be a bisimulation from G_1 to G_2 up to level k, let $Z_{k+1} = \mathrm{Refine}(Z_k)$ and let $\mathcal{T} = (Z, Z_{k+1})$. Then,*

(a) $\mathcal{T}$ is a bisimulation from G_1 to G_2 up to $k+1$;

(b) Suppose that G_1 and G_2 are finite. If $\widehat{Z}$ is the maximal bisimulation from G_1 to G_2 up to level k, then $\mathcal{T}$ is the maximal bisimulation from G_1 to G_2 up to level $k+1$.

PROOF: The proof of (a) follows directly from the construction of $\mathrm{Refine}(Z)$.

To show (b), let $\widehat{Z}' = \{Z'_1, \ldots, Z'_{k+1}\}$ be the maximal bisimulation from G_1 to G_2 up to level $k+1$. We will show that $\widehat{Z}' \subseteq \mathcal{T}$. Note that by construction of $\mathcal{T}$, the first k components of $\mathcal{T}$ equal $\widehat{Z}$. Furthermore, since $\widehat{Z}$ is a maximal bisimulation up to level k, $Z'_i \subseteq Z_i$ for all $0 \leq i \leq k$. Thus the only thing left to show is that $Z'_{k+1} \subseteq Z_{k+1}$. By Lemma 5.17, $\widehat{Z}'$ is monotonically decreasing — which together with the previous argument implies that $Z'_{k+1} \subseteq Z'_k \subseteq Z_k$. Moreover, by definition of the refine operation, $Z'_{k+1} \subseteq \mathrm{Refine}(Z'_k)$. Now, since the refine operation is monotone, $\mathrm{Refine}(Z'_k) \subseteq \mathrm{Refine}(Z_k)$, and hence $Z'_{k+1} \subseteq \mathrm{Refine}(Z_k) = Z_{k+1}$. ■

We will now utilize the new bisimulation notion to show that it is decidable whether two marked graphs are bisimilar up to any level.

Theorem 5.20: *It is decidable in polynomial time whether two marked finite graphs (G_1, a_1, b_1) and (G_2, a_2, b_2) are bisimilar up to k for every natural number k.*

PROOF: Let Z_0 be the set of all tuples $(a_1, b_1, a_2, b_2) \in \mathrm{adom}(G_1)^2 \times \mathrm{adom}(G_2)^2$ which satisfy the bisimulation atoms condition. Note that this is the maximum bisimulation from G_1 to G_2 up to level 0. Now, for every natural number $k > 0$ define by induction $Z_k := \mathrm{Refine}(Z_{k-1})$. By Lemma 5.19, $\widehat{Z}_k = (Z_1, \ldots, Z_k)$ is the maximal bisimulation from G_1 to G_2 up to level k. Now remember that Lemma 5.17 tells us that $\widehat{Z}$ is monotonically decreasing, hence $Z_{k-1} \supseteq Z_k$. Therefore, since G_1 and G_2 are finite, there exists l, such that for every $n \geq l$, $Z_n = Z_l$. Note that $l \leq \max(|\mathrm{adom}(G_1)|, |\mathrm{adom}(G_2)|)^4$, since the refine operator removes only one tuple in the worst case scenario and since there are at most $\max(|\mathrm{adom}(G_1)|, |\mathrm{adom}(G_2)|)^4$ such tuples.

Now, by the preceding argument, if $(a_1, b_1, a_2, b_2) \in Z_l$, then $(a_1, b_1, a_2, b_2) \in Z_n$ for every $n \geq l$. Moreover, Proposition 5.14 and Proposition 5.15 tells us that this is equivalent to: if $(G_1, a_1, b_1) \simeq_l (G_2, a_2, b_2)$, then $(G_1, a_1, b_1) \simeq_n (G_2, a_2, b_2)$ for every $n \geq l$. Thus deciding whether (G_1, a_1, b_1) and (G_2, a_2, b_2) are bisimilar up to k for every k reduces to computing Z_k until the bisimulation stabilizes.

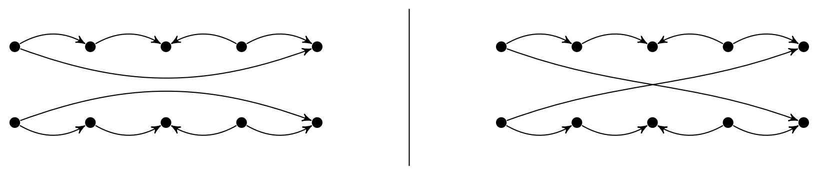


Figure 5.1: Graphs used to establish separation in Proposition 5.21

Since computing a single refine operator can be done in polynomial time and since the procedure only has to compute a polynomial number of such operators — the time complexity of this procedure is polynomial in the size of G_1 and G_2. ∎

We can now use the previous proposition to decide whether the graphs in Figure 5.1 are indistinguishable — we will need this to prove the following proposition.

Proposition 5.21 (Primitivity of $^{-1}$): *Let F_1 and F_2 be sets of nonbasic features such that $^+$ is not present in both F_1 and F_2. If $^{-1} \in F_1$, $\cap \in F_1$ and $^{-1} \notin F_2$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

PROOF: The boolean query q expressed by $(R^2 \circ R^{-1} \circ R) \cap R$ distinguishes the graphs G_1 and G_2 displayed in Figure 5.1. Thus G_1 and G_2 are distinguishable in $\mathcal{N}(F_1)$. On the other hand, using the algorithm described in Theorem 5.20, we can verify that for every $(a_1, b_1) \in \text{adom}(G_1)^2$, there exists $(a_2, b_2) \in \text{adom}(G_2)^2$, such that $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$ for every k. Hence for every k we have found two graphs $G_1' = G_1$ and $G_2' = G_2$, such that for every $(a_1, b_1) \in \text{adom}(G_1)^2$, there exists $(a_2, b_2) \in \text{adom}(G_2)^2$ such that $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$. Therefore by Corollary 3.6, q is not expressible in $\mathcal{N}(\setminus, di)$. Furthermore, since $F_2 \subseteq \{\pi, \overline{\pi}, \cap, \setminus, di\} = \overline{\{\setminus, di\}}$, Theorem 4.4 implies that $\mathcal{N}(F_2) \leq^{\text{path}} \mathcal{N}(di, \setminus)$ and hence also $\mathcal{N}(F_2) \leq^{\text{bool}} \mathcal{N}(di, \setminus)$. Now, by transitivity, $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$ as desired. ∎

We are now ready to prove that Theorem 5.8 holds for languages in $\mathcal{C}[\cap]$.

Proposition 5.22: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be in $\mathcal{C}[\cap]$. If $F_1 \not\subset \widetilde{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

PROOF: Since $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ both are in $\mathcal{C}[\cap]$, $\cap$ is present in both $\overline{F_1}$ and $\overline{F_2}$, and thus $\widetilde{F_2} = \overline{F_2}$. Hence, $F_1 \not\subseteq \overline{F_2}$ if and only if there exists $x \in \{\pi, \overline{\pi}, di, ^{-1}, \setminus\}$ such that $x \in F_1$ and $x \notin \overline{F_2}$. We will go over every possibility for x and show that one of Proposition 4.5, Proposition 4.6, Proposition 4.7, Proposition 4.8, Proposition 4.17, Proposition 4.18 or Proposition 5.21 can be applied, which then directly implies our proposition.

- $x = di$: Proposition 4.5 gives us the desired result;
- $x = {}^{-1}$: Proposition 5.21 proves what was asked;
- $x = \setminus$: Proposition 4.17 proves our proposition in this case;

– $x = \pi$: Then, $\pi \notin \overline{F_2}$ if and only if $di, {}^{-1}, \overline{\pi}, \pi \notin F_2$, and hence $F_2 \subseteq \{\cap, \setminus\}$. Now we can apply Proposition 4.18 which proves the result;

– $x = \overline{\pi}$. Notice that by the interdependencies described at the beginning of Chapter 4

$$\overline{\pi} \notin \overline{F_2} \iff \setminus \notin F_2 \vee (\setminus \in F_2 \wedge \pi \notin \overline{F_2}). \tag{5.1}$$

If $\setminus \notin F_2$, we can apply Proposition 4.6 to prove our result.

On the other hand if $\setminus \in F_2$, then we cannot apply Proposition 4.6 — hence we need another strategy. First note that by equation (5.1) π cannot be present in $\overline{F_2}$. Moreover,

$$\setminus \in F_2 \wedge \pi \notin \overline{F_2} \implies di \notin F_2 \wedge {}^{-1} \notin F_2.$$

Hence, $F_2 \subseteq \{\cap, \setminus\}$, since $F_2 \subseteq \{\cap, \pi, \overline{\pi}, di, {}^{-1}, \setminus\}$. Furthermore, since $\overline{\pi} \in \overline{F_1}$, $\pi \in \overline{F_1}$, and hence we can apply Proposition 4.18, which proves the result. ■

Notice that the proof of the proposition above is similar to the proof of Proposition 4.19. The proof only differs in the case where $x = {}^{-1}$ — this is due to the fact that unlike all the other cases, we only established path separation when $x = {}^{-1}$, which as we saw earlier, does not necessarily imply boolean separation. Also, note that we established strong boolean separation in every step of the proof — hence we get the following corollary which does not necessarily holds in general.

Corollary 5.23: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be in $\mathcal{C}[\cap]$. Then, $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_1)$ if and only if $\mathcal{N}(F_1) \leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

Remark 5.24: Note that for languages in $\mathcal{C}[\cap]$, $\widetilde{F_2} = F_2$. Hence Theorem 4.4, Proposition 5.6 and Proposition 5.22 tell us that the characterizations of $\leq^{\text{path}}$ and $\leq^{\text{bool}}$ for languages in $\mathcal{C}[\cap]$ coincide. ◇

5.3 Languages without $\cap$ and with $^+$

In this section we will characterize $\leq^{\text{bool}}$ for languages in $\mathcal{C}[^+]$, i.e., Theorem 5.9 restricted to $\mathcal{C}[^+]$.

We will now prove a proposition which will be key in the separation result for languages in $\mathcal{C}[^+]$.

Proposition 5.25 (Primitivity of $^{-1}$): *Let F_1 and F_2 be sets of nonbasic features. If ${}^1 \in F_1, {}^+ \in F_1$, and ${}^{-1} \notin F_2$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$.*

The proof of this proposition is technical and will be split into several lemmas in Section 5.3.1.

Armed with the previous lemma and the results from the prior sections and chapters, we are now ready to prove Theorem 5.8 for languages in $\mathcal{C}[^+]$.

Proposition 5.26: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be languages in $\mathcal{C}[^+]$. If $F_1 \not\subseteq \widetilde{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$.*

Proof: Notice that $\widetilde{F_2} = \overline{F_2}$, since $^+ \in \overline{F_2}$. Now, first suppose that $\overline{\pi} \in F_2$. Then, $F_1 \not\subseteq \widetilde{F_2} = \overline{F_2}$ if and only if $F_1 \cap \{di, ^{-1}\} \not\subseteq F_2 \cap \{di, ^{-1}\}$. Hence we have the following two scenarios: $di \in F_1$ and $di \notin \overline{F_2}$; $^{-1} \in F_1$ and $^{-1} \notin \overline{F_2}$. If $di \in F_1$ and $di \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$ due to Proposition 4.5. In the latter case we can apply Proposition 5.25 which tells us that $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$.

Now suppose that $\overline{\pi} \notin F_2$, then $F_2 = \overline{F_2}$. Hence, $F_1 \not\subseteq \overline{F_2}$ if and only if $F_1 \not\subseteq F_2$. Therefore, there has to exists some $x \in F_1$ such that $x \notin F_2$. Since $F_1 \subseteq \{di, \pi, \overline{\pi}, ^{-1}, ^+\}$ and $F_2 \subseteq \{di, \pi, ^{-1}, ^+\}$, either Proposition 4.5, Proposition 4.6, Proposition 4.8 or Proposition 5.25 gives us the required result. Notice that we cannot apply those propositions directly since they use $\overline{F_1}$ instead of F_1. However, here this is no issue since $F_1 \subseteq \overline{F_1}$. ∎

5.3.1 Proof of Proposition 5.25

First we will try to find a tight bound on the degree of R^n. To this end, we will examine the tree structure of R^n. First we will need two preliminary lemmas on trees.

Lemma 5.27: *A complete binary tree with $2^{n+1} - 1$ nodes has depth n.*

Proof: We will prove this by strong induction on n.
Induction Basis: When $n = 0$, the tree has only 1 node, and thus has depth 0.
Induction Hypothesis: Suppose that for $k < n$, a complete binary tree with $2^{k+1} - 1$ nodes has depth k.
Induction Basis: We will now show that our lemma holds for complete binary trees of depth n. To this end, let T be a complete binary tree with $2^{n+1} - 1$ nodes. Since T is complete it has two subtrees of size $2^n - 1$ rooted in the root node of T. By our induction hypothesis either subtree has depth $n - 1$. Therefore, since T has one level more level than either subtree, it has depth n as desired. ∎

Lemma 5.28: *A binary tree T with n nodes where every node is added in breadth first fashion has a depth of at most $\lceil \log_2 n \rceil$.*

Proof: Suppose that m is the natural number such that $2^m < n \leq 2^{m+1}$. Since the nodes of T are added in a depth first fashion — the levels of T are filled up, except possibly the last level. Therefore, T has at most as many levels as a complete binary tree with $2^{m+2} - 1$ nodes, which according to Lemma 5.27 has depth $m + 1 = \log_2(2^{m+1})$. Our lemma now follows from the fact that $\lceil \log_2 n \rceil = \log_2 2^{m+1}$. ∎

Now we will show that the previous lemma provides a bound on the degree of R^n. Recall from Definition 3.3 that $\mathcal{N}(F)_d$ denotes the set of expressions in $\mathcal{N}(F)$ with a degree of at most d.

Corollary 5.29: *Let n be a natural number, and let R be a relation. The expression R^n is equivalent to an expression in $\mathcal{N}_{\lceil \log_2 n \rceil}$.*

PROOF: We will construct a new expression $e \in \mathcal{N}$ equivalent to R^n by constructing its tree representation — we do so as follows: add n transitive closure operations in a breadth first fashion and then add two R labelled nodes to the leaf nodes labelled $\circ$. This expression clearly is equivalent to R^n since the composition operation is associative. By Lemma 5.28 this tree has at most depth $\lceil \log_2 n \rceil$. The depth of this tree obviously equals the degree of e since the root of this tree is a composition application and the last level of this tree only contains R labelled nodes. Thus we have found an expression in $\mathcal{N}$ equivalent to R^n with a degree of at most $\lceil \log_2 n \rceil$. ∎

We will now introduce some definitions which will be used until the remainder of this section. Let m be a multiple of 4, let G_1^m be the graph at the top of Figure 5.2 and G_2^m be the graph at the bottom. It is important to note that those graphs have the displayed form only when $m \in \mathbb{N}$ is a multiple of 4. Hence why we assume that m is a multiple of 4.

We say that a tuple $(x, y) \in \text{adom}(G_1) \times \text{adom}(G_2)$ is *valid* if the following conditions hold:

– $x \in \{y_i, w_i, t_i\}$, then $y \in \{u_i, v_i, w_i'\}$;

– $x = x_1$, then $y = x_1'$;

– $x = x_2$, then $y = x_2'$;

– $x = z_1$, then $y = z_1'$;

– $x = z_2$, then $y = z_2'$;

Furthermore, we say that a quadruple $(a_1, b_1, a_2, b_2) \in \text{adom}(G_1)^2 \times \text{adom}(G_2)^2$ is *valid* if the following holds:

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

(b) $(a_1, b_1) \in G_1$ if and only if $(a_2, b_2) \in G_2$; and $a_1 = b_1$ if and only if $a_2 = b_2$ (atoms);

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

(d) $(a_1, b_1, a_2, b_2) = (x_2, b_1, x_2', u_2)$, then $b_2 = y_2$;

Intuitively a valid quadruple is a potential candidate for a bisimilarity between G_1^m and G_2^m.

We will say that a node $x \in \text{adom}(G_1^m)$ is "'*Y left*" if $x \in \{x_1, x_2\}$ or $x = y_i$ with $0 \leq i \leq m/2 + 1$. Furthermore we will say that x is Y right if $x \in \{z_1, z_2\}$ or $x = y_i$ with $m/2 + 1 < i \leq m + 1$. Analogously we will say that x is W (resp. T) left if $x = w_i$ (resp. $x = t_i$) where $0 \leq i \leq m/2 + 1$ and W (resp. T) right if $m/2 + 1 < i \leq m + 1$. For nodes in $\text{adom}(G_2^m)$ we will use an analogous left and right notion for U, V and W'.

Let us now define a function f with as domain the valid tuples.

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

Later we will show that the maximum degree of bisimilarity for a valid quadruple (a_1, b_1, a_2, b_2) is $\min(f(a_1, a_2), f(b_1, b_2))$. Hence $f(x, y)$ can can be seen as a potential bound on the number of rounds in the bisimilarity game one is allowed to play.

Towards a proof of this fact, we first need a few technical lemmas. By inspecting Figure 5.2 it is clear that for every $c_1 \in \text{adom}(G_1^m)$, we can find a $c_2 \in \text{adom}(G_2^m)$ such that (c_1, c_2) is valid, and (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) satisfy the atoms condition if (a_1, b_1, a_2, b_2) is valid, $f(a_1, a_2) > 0$ and $f(b_1, b_2) > 0$. The following lemma justifies this intuition.

Lemma 5.30: *If (c_1, b_1, c_2, b_2) is valid, $f(a_1, a_2) > 0$ and $f(b_1, b_2) > 0$, then for every $c_1 \in \text{adom}(G_1^m)$, there exists $c_2 \in \text{adom}(G_2^m)$ such that (c_1, b_2) is valid, and (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) satisfy the atoms condition.*

Proof: Let $c_1 \in \text{adom}(G_1)$. We will split out proof up into several cases.

– Suppose that (a_1, c_1) is an edge, then we pick c_2 in the same column[1] as c_1 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 column of a_2 has the same outdegree as a_1. On the other hand, if $a_1 = y_{m+1}$, then $a_2 = v_{m+1}$ since $f(a_1, a_2) > 0$. Thus there is room to pick c_2. Hence (c_1, c_2) is valid and (a_1, c_1, a_2, c_2) satisfies the atoms condition.

  We will now show that (c_1, b_1, c_2, b_2) is valid. If (c_1, b_1) is *also* an edge, then c_1 has indegree and outdegree 1. Hence $c_1 \in \{x_2, y_1, z_2\}$. Let us now consider each such c_1.

[1]We say that two nodes are in the same column if they are displayed in the same column in Figure 5.2.

  - If $c_1 = x_2$, then $c_2 = x'_2$, $a_1 = x_1$ and $b_1 = y_1$. Now, $a_2 = x'_1$ since (a_1, b_1, a_2, b_2) is valid. Furthermore, $b_2 = u_1$ since $f(b_1, b_2) > 0$. Thus (c_2, b_2) is also an edge as desired;
  - If $c_1 = y_1$, then $c_2 = u_1$, $a_1 = x_2$ and $b_1 = y_2$. Now, $a_2 = x'_2$ since (a_1, b_1, a_2, b_2) is valid. Furthermore, $b_2 = u_2$, since (x_2, y_2, x'_2, b_2) is only valid when $b_2 = u_2$ by definition. Thus (c_2, b_2) is also an edge as desired;
  - If $c_1 = z_2$, then $c_2 = z'_2$, $a_1 = y_{m+1}$ and $b_2 = z_1$. Now, $b_2 = z'_1$ since (a_1, b_1, a_2, b_2) is valid. Furthermore, $a_2 = v_{m+1}$ since $f(a_1, a_2) > 0$. Thus (c_2, b_2) is also an edge as desired.

  On the other hand, if (c_2, b_2) is an edge, then c_2 has indegree and outdegree 1 and hence $c_2 \in \{x'_2, u_1, z_2\}$. Let us now consider each such c_2.

  - If $c_2 = x'_2$, then $c_1 = x_2$, $a_2 = x'_1$ and $b_2 = u_1$, and thus $a_1 = x_1$ since (a_1, b_1, a_2, b_2) is valid. Furthermore, $b_1 = y_1$ since $f(b_1, b_2) > 0$. Thus (c_1, b_1) is also an edge as desired;
  - If $c_2 = u_1$, then $c_1 = y_1$, $a_2 = x'_2$ and $b_1 = u_2$, and thus $a_1 = x_2$ since (a_1, b_1, a_2, b_2) is valid. Furthermore, $b_2 = u_2$, since (x_2, y_2, x'_2, b_2) is only valid when $b_2 = u_2$ by definition. Thus (c_1, b_1) is also an edge as desired;
  - If $c_2 = z'_2$, then $c_1 = z_2$, $a_2 = v_{m+1}$ and $b_2 = z'_1$, and thus $b_2 = z_1$ since (a_1, b_1, a_2, b_2) is valid. Furthermore, $a_2 = y_{m+1}$ since $f(a_1, a_2) > 0$. Thus (c_1, b_1) is also an edge as desired.

  Also, note that

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

  Hence (c_1, b_1, c_2, b_2) satisfies the atoms condition.

- Suppose that (c_1, b_1) is an edge and (a_1, b_1) is not an edge. Suppose that we pick c_2 such that (c_2, b_2) is an edge.

  This is clearly possible if $b_1 \neq y_1$, since in this case any node in the column of b_2 has the same indegree as b_1. On the other hand, if $b_1 = y_1$, then $b_2 = u_1$ since $f(b_1, b_2) > 0$. Thus there is room to pick c_2. Hence (c_1, c_2) is valid and (c_1, b_1, c_2, b_2) satisfies the atoms condition.

  We will now show that (a_1, c_1, a_2, c_2) satisfies the atoms condition. First, we will show that (a_2, c_2) is not an edge. To this end, suppose that (a_2, b_2) is an edge. Then c_2 has indegree and outdegree 1 and hence $c_2 \in \{x'_2, u_1, z_2\}$. Let us now consider each such c_2.

  - If $c_2 = x'_2$, then $c_1 = x_2$ and $a_2 = x'_1$, and thus $a_1 = x_1$ since (a_1, b_1, a_2, b_2) is valid. Hence (a_1, c_1) is an edge which contradicts our assumption.

  - If $c_2 = u_1$, then $c_1 = y_1$ and $a_2 = x'_2$ and thus $a_1 = x_2$ since (a_1, b_1, a_2, b_2) is valid. Thus (a_1, c_1) is also an edge which contradicts our assumption
  - If $c_2 = z'_2$, then $c_1 = z_2$ and $a_2 = v_{m+1}$, and thus $a_2 = y_{m+1}$ since $f(a_1, a_2) > 0$. Thus (a_1, c_1) is also an edge which contradicts our assumption.

  Hence (a_2, c_2) is not an edge. Furthermore, note that

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

  Thus (a_1, c_1, a_2, c_2) satisfies the atoms condition as desired.

- Suppose that $a_1 = c_1 = b_1$, then we pick $c_2 = a_2$. By construction (c_1, c_2) is valid. Furthermore, notice that now $c_2 = b_2$, because $a_2 = b_2$ due to the validity of (a_1, b_1, a_2, b_2). Hence the atoms condition holds for (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2).

- Assume that $a_1 = c_1$ and that (c_1, b_1) is not an edge. Then we pick c_2 such that $c_2 = a_2$. Since a_1 is in the same column as a_2, c_1 is in the same column as c_2. Hence by construction (c_1, c_2) is valid and (a_1, c_1, a_2, c_2) satisfies the atoms condition.

  We will now show that (c_1, b_1, c_2, b_2) satisfies the atoms condition. First notice that (c_2, b_2) is not an edge since

$$\begin{aligned} (c_1, b_1) \text{ is not an edge} &\implies (a_1, b_1) \text{ is not an edge} \\ &\implies (a_2, b_2) \text{ is not an edge} && \text{(since } (a_1, b_1, a_2, b_2) \text{ is valid)} \\ &\implies (c_2, b_2) \text{ is not an edge} && \text{(since } a_2 = c_2) \end{aligned}$$

  Furthermore, note that $c_1 = b_1$ if and only if $c_2 = b_2$ since

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

  Hence (c_1, b_1, c_2, b_2) satisfies the atoms condition.

- The case where (a_1, c_1) is not an edge and $c_1 = b_1$ is analogous to the previous case.

- Assume that (a_1, c_1) and (c_1, b_1) are not edges, and assume that $a_1 \neq c_1$ and $b_1 \neq c_1$. First we will consider every $c_1 \in \{x_1, x_2, z_1, z_2\}$.

  - Suppose $c_1 = x_1$. Then $x_2 \neq b_1 \neq x_1$ and $a_1 \neq x_1$, and thus $x'_2 \neq b_2 \neq x'_1$ and $a_1 \neq x'_1$ due to the validity of (a_1, b_1, a_2, b_2). Hence the atoms condition for (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) is satisfied.

– Suppose $c_1 = x_2$. Then $x_2 \neq b_1 \neq y_1$ and $x_1 \neq a_1 \neq x_2$, and thus $x'_2 \neq b_1$ and $x_1 \neq a_1 \neq x_2$ due to the validity of (a_1, b_1, a_2, b_2). Furthermore, $b_2 \neq u_1$ since $f(b_1, b_2) > 0$. Hence the atoms condition for (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) is satisfied.

– Suppose $c_1 = z_1$. Then $b_1 \neq z_1$ and $z_2 \neq a_1 \neq z_1$, and thus $b_1 \neq z'_1$ and $z'_2 \neq a_1 \neq z'_1$. Hence the atoms condition for (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) is satisfied.

– Suppose $c_1 = z_2$. Then $z_2 \neq b_1 \neq z_1$ and $y_{m+1} \neq a_1 \neq z_2$, and thus $z'_2 \neq a_1$ and $z'_2 \neq b_1 \neq z'_1$ due to the validity of (a_1, b_1, a_2, b_2). Furthermore, $a_1 \neq v_{m+1}$ since $f(a_1, a_2) > 0$. Hence the atoms condition for (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) is satisfied.

On the other hand, if $c_1 \notin \{x_1, x_2, z_1, z_2\}$, then we pick c_2 on the chain not containing a_2 and b_2. Notice that this is possible since there are 3 chains. Then clearly (a_1, c_1) and (c_2, b_2) are not edges, $a_2 \neq c_2$ and $c_2 \neq b_2$. Hence the atoms condition for (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) is satisfied. ∎

Intuitively, the lemma above still holds when we switch the roles of c_1 and c_2.

Lemma 5.31: *If (c_1, b_1, c_2, b_2) is valid, $f(a_1, a_2) > 0$ and $f(b_1, b_2) > 0$, then for every $c_2 \in \text{adom}(G_2^m)$, there exists $c_1 \in \text{adom}(G_1^m)$ such that (c_1, c_2) is valid, and (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) satisfy the atoms condition.*

Notice that in the proof of Lemma 5.30 we only inspected the graph locally, i.e., we only inspected neighborhoods with radius 2 of valid node pairs (x, y) in G_1^m and G_2^m. Hence the proof of the previous lemma is analogous to the proof of Lemma 5.30, since the lefthand side (resp. righthand) of G_1^m is completely the same as the lefthand (resp. righthand) side G_2^m.

Again by inspecting Figure 5.2 it is clear that for every $c_1 \in \text{adom}(G_1^m)$, we can find a $c_2 \in \text{adom}(G_2^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are potential candidates to continue our game, if (a_1, b_1, a_2, b_2) is valid, $f(a_1, a_2) > 0$ and $f(b_1, b_2) > 1$. The following lemma justifies this intuition.

Lemma 5.32: *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 $c_1 \in \text{adom}(G_1^m)$, there exists $c_2 \in \text{adom}(G_2^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid.*

Proof: Let $c_1 \in \text{adom}(G_1)$. Note that since $f(b_1, b_2) > 1$, $b_1 = y_2$ if and only if $b_2 = u_2$. Hence conditions c and d for the validity of (c_1, b_1, c_2, b_2) hold for whatever c_2 we pick such that (c_1, c_2) is valid. Hence we only need to find a c_2 such that (c_1, c_2) is valid, (c_1, b_1, c_2, b_2) satisfies the atoms condition and (a_1, c_1, a_2, c_2) is valid. Lemma 5.30 tells us that there exists c_2 such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) satisfy the atoms condition. Furthermore, by the previous argument (c_1, b_1, c_2, b_2) is valid. Now, if (a_1, c_1, a_2, c_2) is valid we are done.

On the other hand, suppose that (a_1, c_1, a_2, c_2) is not valid. Then condition c or d for the validity of (a_1, c_1, a_2, c_2) is not satisfied, since the atoms condition is trivially satisfied. To this end suppose that condition c is not satisfied, then $(a_1, c_1, a_2, c_2) = (x_2, y_2, x'_2, c_2)$ where $c_2 \neq u_2$. We will now show that picking $c_2 = u_2$ does not break the atoms condition for (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2). Since $(a_1, c_1, a_2, c_2) = (x_2, y_2, x'_2, u_2)$, we know that (a_1, c_1, a_2, c_2) is valid. Let us now show that (c_1, b_1, c_2, b_2) satisfies the atoms condition. First note that (c_1, b_1) and (c_2, b_2) cannot be edges since the outdegree of c_1 and c_2 is 0. Furthermore, note that

$$\begin{aligned} c_1 = b_1 &\iff b_1 = y_2 \\ &\iff b_2 = u_2 && \text{(since } f(b_1, b_2) > 1\text{)} \\ &\iff c_2 = b_2. && \text{(since } c_2 = u_2\text{)} \end{aligned}$$

Hence the atoms condition for (c_1, b_1, c_2, b_2) is satisfied, and thus (c_1, b_1, c_2, b_2) is valid since d is satisfied trivially. On the other hand, suppose that condition d is not satisfied, then $(a_1, c_1, a_2, c_2) = (x_2, z, x'_2, u_2)$ where $c_1 \neq y_2$. We will now show that we can pick a new c_2 such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid. First note that every node in the column of c_1 has outdegree 0, thus (c_1, b_1) and (c_2, b_2) are not edges for whatever new c_2 we pick. Hence we only need to pick a new c_2 that satisfies: $b_1 = c_1$ if and only $b_2 = c_2$. To this end suppose that $b_1 = c_1$. Then $b_1 \neq y_2$, and hence $b_2 \neq u_2$ since $f(b_1, b_2) > 1$. Thus if we now pick $c_2 = b_2$, then (c_1, b_1, c_2, b_2) satisfies the atoms condition, and thus (c_1, b_1, c_2, b_2) is valid. Also, (a_1, c_1, a_2, c_2) is valid, since condition c is trivially satisfied and $(a_1, c_1, a_2, c_2) = (x_2, c_1, x'_2, c_2)$ where $c_1 \neq y_2$ and $c_2 \neq u_2$. On the other hand if $b_1 \neq c_1$, then we pick c_2 such that $u_2 \neq c_2 \neq b_2$, note that this is possible since there are 3 chains. Then (c_1, b_1, c_2, b_2) satisfies the atoms condition by construction and thus (c_1, b_1, c_2, b_2) is valid. Furthermore, (a_1, c_1, a_2, c_2) is valid since condition c is satisfied by construction and condition d is trivially satisfied. ■

Intuitively, the previous lemma still holds when we switch the roles of c_1 and c_2.

Lemma 5.33: *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 $c_2 \in \text{adom}(G_2^m)$, there exists $c_1 \in \text{adom}(G_1^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid.*

For the same reasons why the proof of Lemma 5.30 was analogous to the proof of Lemma 5.31, the proof of the previous lemma is analogous to the proof of Lemma 5.32.

When we start our bisimulation game on G_1^m and G_2^m in (a_1, b_1, a_2, b_2), we will show later that we are allowed to play $\min(f(a_1, a_2), f(b_1, b_2))$ rounds. The following lemma will be the key in the proof of this fact.

Lemma 5.34: *If (a_1, b_1, a_2, b_2) is valid and $f(a_1, a_2) = f(b_1, b_2) = m/2$, then for every $c_1 \in \text{adom}(G_1^m)$, there exists $c_2 \in \text{adom}(G_2^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid, and $f(c_1, c_2) \geq m/2 - 1$.*

PROOF: Let $c_1 \in \text{adom}(G_1)$. Note that if we pick a c_2 such that $f(c_1, c_2) \geq m/2 - 1$, then 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) are satisfied trivially, hence we only have to check the atoms conditions. We will split our proof up into several cases. First suppose that (a_1, c_1) is an edge. Now, by Lemma 5.32 there exists c_2 such that (a_1, c_1, a_2, b_2) and (c_1, b_1, c_2, b_2) are valid. Furthermore, since (a_1, c_1, a_2, b_2) is valid, (a_2, c_2) is an edge. We will now show that $f(c_1, c_2) \geq m/2 - 1$. First note $f(a_1, a_2)$ only equals $m/2$ when 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. We will now consider each such scenario.

- Suppose that $a_1 \in \{y_{\frac{m}{2}+1}, t_{\frac{m}{2}+1}, w_{\frac{m}{2}+1}\}$. Then clearly $f(c_1, c_2) = m/2 - 1$ since (a_1, c_1) and (a_2, c_2) are edges.
- Suppose that the column of a_1 is not $m/2 + 1$, a_1 is Y left and a_2 is U left. Then c_1 is Y left and c_2 is U left since (a_1, a_2) and (a_2, c_2) are edges, and thus $f(c_1, c_2) = m/2$ as desired.
- Suppose that the column of a_1 is not $m/2 + 1$. The cases where 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; and a_1 is T left and a_2 is V left are analogous to the previous scenario.
- If a_1 is Y right and a_2 is V right, then c_1 is Y right and c_2 is V right since (a_1, c_1) and (a_2, c_2) are edges. Hence $f(c_1, c_2) = m/2$.
- The cases where 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; and a_1 is T right and a_2 is U right are analogous to the previous scenario.

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

Suppose that $a_1 = c_1$. By Lemma 5.32 there exists c_2 such that (a_1, c_1, a_2, b_2) and (c_1, b_1, c_2, b_2) are valid. Hence $a_2 = c_2$. Furthermore, $f(c_1, c_2) = f(a_1, a_2) = m/2$ as desired.

The case where $c_1 = b_1$ is analogous to the previous case.

Now suppose that (a_1, c_1) and (c_1, b_1) are not edges, $a_1 \neq a_1$ and $c_1 \neq b_1$. If $c_1 \notin$ then we pick c_2 according to the following strategy:

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

Clearly $f(c_1, c_2) = m/2$. Hence, if (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid we are finished. Now suppose that (a_1, c_1, a_2, c_2) is not valid (the case where (c_1, b_2, c_2, b_2) is not valid is analogous). Suppose that (a_2, c_2) is an edge. We will now consider every possible scenario for c_1.

- Suppose that $c_1 = y_{\frac{m}{2}}$. By Lemma 5.32 there exists a new c_2 such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid. Furthermore $f(c_1, c_2) \geq m/2 - 1$ as desired.
- The cases where $c_1 = y_{\frac{m}{2}}$, $c_1 = w_{\frac{m}{2}}$, $c_1 = t_{\frac{m}{2}}$, $c_1 = y_{\frac{m}{2}+2}$, $c_1 = w_{\frac{m}{2}+2}$ and $c_1 = t_{\frac{m}{2}+2}$ are analogous to the previous case.
- Suppose that c_1 is Y left and $c_1 \neq y_{\frac{m}{2}-1}$. Then $c_2 = u_i$ where $i < m/2 - 1$. Hence $a_2 = u_j$ where $j \leq m/2 - 1$ since (a_2, c_2) is an edge. Now since $f(a_1, a_2) = m/2$, a_1 is Y left. Hence (a_1, c_1) is an edge, which contradicts our hypothesis. Thus this scenario cannot occur.
- Suppose that c_1 is Y right and $c_1 \neq v_{\frac{m}{2}+2}$. Then $c_2 = v_i$ where $i > m/2 + 2$. Hence $a_2 = v_j$ where $j \geq m/2 + 2$. Now since $f(a_1, a_2) = m/2$, a_1 is Y right, and thus (a_1, c_1) is an edge, which contradicts our hypothesis. Therefore, this scenario cannot occur.
- Suppose that c_1 is W left and $c_1 \neq w_{\frac{m}{2}-1}$. Then $c_2 = w'_i$ where $i < m/2$. Hence $a_2 = w'_j$ where $j \leq m/2 - 1$. Now since $f(a_1, a_2) = m/2$, a_1 is T left. Now, we pick our new $c_2 = v_i$ on V, then (a_1, c_1, a_2, c_2) is valid. We will now show that (c_1, b_1, c_2, b_2) is valid. Suppose for the sake of contradiction that $c_2 = b_2$. Then, notice that

  $$\begin{aligned} c_2 = b_2 &\implies b_2 = v_i \\ &\implies b_1 = w_i \vee b_1 = t_i \qquad \text{(since } f(b_1, b_2) = m/2 \text{ and } i < m/2) \end{aligned}$$

  If $b_1 = w_i$, then $b_1 = c_1$, which contradicts the fact that $b_1 \neq c_1$. On the other hand, if $b_1 = t_i$, then (a_1, b_1) is an edge since a_1 is also on T, a_1 is in the column next to c_1 and a_1 has at least outdegree 1. But then (a_2, b_2) is an edge since (a_1, b_1, a_2, b_2) is valid, and thus $b_2 = w'_i$ (the old position of c_2), which contradicts the fact that $c_2 = b_2$. Hence $c_2 \neq b_2$.

  We will now show that (c_2, b_2) cannot be an edge. For the sake of contradiction suppose that (c_2, b_2) is an edge. Then c_2 has indegree and outdegree 1. Hence $c_2 = \{x_2, y_1\}$, which contradicts the fact that c_2 is W left. Thus (c_1, b_1, a_2, b_2) is valid since $c_2 \neq b_2$ and (c_2, b_2) is not an edge.
- The case where c_1 is T left and $c_1 \neq t_{\frac{m}{2}-1}$ is analogous to the previous case.
- Suppose that c_1 is W right and $c_1 \neq w_{\frac{m}{2}+2}$. Then $c_2 = w'_i$ where $i > m/2 + 2$. Hence $a_2 = w'_j$ where $j \geq m/2 + 2$. Now since $f(a_1, a_2) = m/2$ and (a_2, c_2) is an edge, a_1 is T right. Let us now pick our new $c_2 = u'_i$ on U. Clearly, then (a_1, c_1, a_2, c_2) is valid. We will now show that (c_1, b_1, c_2, b_2) is valid. Notice that

  $$\begin{aligned} c_2 = b_2 &\implies b_2 = u_i \\ &\implies b_1 = w_i \vee b_1 = t_i \qquad \text{(since } f(b_1, b_2) = m/2 \text{ and } i > m/2 + 2) \end{aligned}$$

If $b_1 = w_i$, then $b_1 = c_1$, which contradicts the fact that $b_1 \neq c_1$. On the other hand, if $b_1 = t_i$, then (a_1, b_1) is an edge since a_1 is also on t_i, a_1 is in the column next to c_1 and a_1 has at least outdegree 1. But then (a_2, b_2) is an edge since (a_1, b_1, a_2, b_2) is valid, and thus $b_2 = w'_i$ (the old position of c_2). This contradicts the fact that $c_2 = b_2$ since $c_2 = u_i$. Hence $c_2 \neq b_2$.

We will now show that (c_2, b_2) cannot be an edge. For the sake of contradiction suppose that (c_2, b_2) is an edge. Then c_2 has indegree and outdegree 1. Hence $c_2 = \{x_2, y_1\}$, which contradicts the fact that c_2 is W left. Thus (c_1, b_1, a_2, b_2) is valid since $c_2 \neq b_2$ and (c_2, b_2) is not an edge.

– The case where c_2 is T right is analogous to the precious case.

On the other hand, suppose that $a_2 = c_2$. Again we will consider every scenario for c_1.

– Suppose that $c_1 = y_{\frac{m}{2}}$. By Lemma 5.32 there exists a new c_2 such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid. Furthermore $f(c_1, c_2) \geq m/2 - 1$ as desired.

– The cases where $c_1 = y_{\frac{m}{2}}$, $c_1 = w_{\frac{m}{2}}$, $c_1 = t_{\frac{m}{2}}$, $c_1 = y_{\frac{m}{2}+2}$, $c_1 = w_{\frac{m}{2}+2}$ and $c_1 = t_{\frac{m}{2}+2}$ are analogous to the previous case.

– Suppose that c_1 is Y left and $c_1 \neq y_{\frac{m}{2}-1}$. Then $c_1 = y_i$ and $c_2 = a_2 = u_i$ where $i < m/2 - 1$. Hence $a_1 = y_i$ since $f(a_1, a_2) = m/2$, and thus $a_1 = c_1$, which contradicts our assumption that $a_1 \neq c_1$. Thus this scenario cannot occur.

– Suppose that c_1 is Y right and $c_1 \neq y_{\frac{m}{2}+2}$. Then $c_1 = y_i$ and $c_2 = a_2 = v_i$ where $i > m/2 + 2$. Hence $a_1 = y_i$ since $f(a_1, a_2) = m/2$, and thus $a_1 = c_1$, which contradicts our assumption that $a_1 \neq c_1$. Thus this scenario cannot occur.

– Suppose that c_1 is W left and $c_1 \neq w_{\frac{m}{2}-1}$. Then $c_1 = w_i$ and $c_2 = a_2 = w'_i$ where $i < m/2 - 1$. First note that a_1 cannot be on W since $a_1 \neq c_1$ and since c_1 is on W. Furthermore, since $f(a_1, a_2) = m/2$ and since $i < m/2 - 1$, we know that $a_1 = t_i$. Let us now pick a new $c_2 = v_i$ on V. Then clearly (a_1, c_1, a_2, c_2) is valid. We will now show that (c_1, b_1, c_2, b_2) is valid. We will first show that $c_2 \neq b_2$. Suppose for the sake of contradiction that $c_2 = b_2$. Notice that

$$\begin{aligned} c_2 = b_2 &\implies b_2 = w'_i \\ &\implies b_1 = w_i \vee b_1 = t_i \qquad \text{(since } f(b_1, b_2) = m/2 \text{ and } i > m/2 + 2) \end{aligned}$$

If $b_1 = w_i$, then $b_1 = c_1$, which contradicts the assumption that $b_1 \neq c_1$. On the other hand, if $b_1 = t_i$, then $a_1 = b_1$ since $a_1 = t_i$. But then $a_2 = b_2$ since (a_1, b_1, a_2, b_2) is valid, and thus $b_2 = w'_i$ (the old position of c_2). This contradicts the fact that $b_2 = c_2$ since $c_2 = v_i$

Now we will show that (c_2, b_2) is not an edge. Suppose for the sake of contradiction that (c_2, b_2) is an edge. Since (c_1, b_1) is not an edge and $f(b_1, b_2) > 1 > 0$, we know that b_1 is on T. Therefore (a_1, b_1) is an edge since a_1 is on T and b_1 is in the column next to a_1. But then (a_2, b_2) is an edge since (a_1, b_1, a_2, b_2) is valid, which implies that b_2 is on W and thus contradicts our assumption. Hence (c_2, b_2) is not an edge and thus (c_1, b_1, c_2, b_2) is valid.

– The case where c_1 is W right and $c_1 \neq w_{\frac{m}{2}+2}$; c_1 is T left and $c_1 \neq t_{\frac{m}{2}-1}$; and c_1 is T right and $c_1 \neq t_{\frac{m}{2}+2}$ are analogous. ■

An analogous lemma holds when we switch the roles of c_1 and c_2.

Lemma 5.35: *If (a_1, b_1, a_2, b_2) is valid and $f(a_1, a_2) = f(a_1, a_2) = m/2$, then for every $c_1 \in \text{adom}(G_1^m)$, there exists $c_2 \in \text{adom}(G_2^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid, and $f(c_1, c_2) \geq m/2 - 1$.*

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 righthand side). Hence the proof of the previous lemma is analogous to the proof of Lemma 5.34 since G_1^m and G_2^m look completely the same on the lefthand (resp. righthand) side.

Intuitively, Lemma 5.34 tells us that $f(a_1, a_2)$ and $f(b_1, b_2)$ provide an upper bound on the amount of rounds in game if $f(a_1, b_1) = f(b_1, b_2) = m/2$. The following lemma tells us that this is also true when one of $f(a_1, a_2)$ and $f(b_1, b_2)$ is less than $m/2$.

Lemma 5.36: *If (a_1, b_1, a_2, b_2) is valid, $f(a_1, a_2) > 0$, $f(b_1, b_2) > 1$, and one of $f(a_1, a_2)$ and $f(b_1, b_1)$ is strictly less than $m/2$, then for every $c_1 \in \text{adom}(G_1^m)$ there exists $c_2 \in \text{adom}(G_2^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid, and $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.*

Proof: Let $c_1 \in \text{adom}(G_1)$. Then we pick c_2 in the following way:

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

If (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid, we are finished since $f(c_1, c_2) = m/2 \geq f(a_1, a_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$. On the other hand, if (a_1, c_1, a_2, c_2) or $(c_1, b_!, c_2, b_2)$ is invalid, then notice that $c_1 \neq\in \{x_1, x_2, z_1, z_2\}$. Hence from here we can assume that $c_1 \neq\in \{x_1, x_2, z_2, z_2\}$. Also note that only the atoms condition for (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) can be violated since conditions c and d are satisfied by construction. We will now split our proof up into several cases.

– Suppose that (a_1, c_1, a_2, c_2) is invalid and $f(a_1, a_2) < m/2$. First, note that $f(a_1, a_2) = i - 1$ if $i \leq m/2 + 1$ and that $f(a_1, a_2) = m + 1 - i$ if $i > m/2 + 1$, where i is the column[2] of a_1. Now, since (a_1, c_1, a_2, c_2) is invalid, we know that

[2]We say that i is the column of a node x if $x \in \{y_i, w_i, t_i, u_i, v_i, w'_i\}$.

the column of c_1 is $i-1$, i or $i+1$. Hence whatever new c_2 we pick, $f(c_1, c_2) \geq i-2 = f(a_1, a_2) - 1 \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$ if $i \leq m/2+1$ and $f(c_1, c_2) \geq m+1-(i+1) = f(a_1, a_2) - 1 \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$ if $i > m/2+1$. By Lemma 5.32 there exists a new c_2 such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2). Furthermore, by the previous argument $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$ as desired.

– Suppose that (c_1, b_1, c_2, b_2) is invalid and that $f(b_1, b_1) < m/2$. First, note that $f(b_1, b_2) = i-1$ if $i \leq m/2+1$ and that $f(b_1, b_2) = m+1-i$ if $i > m/2+1$, where i is the column of b_1. Now, since (c_1, b_1, c_2, b_2) is invalid, we know that the column of c_1 is $i-1$, i or $i+1$. Hence whatever new c_2 we pick, $f(c_1, c_2) \geq i-2 = f(b_1, b_2) - 1 \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$ if $i \leq m/2+1$ and $f(c_1, c_2) \geq m+1-(i+1) = f(b_1, b_2) - 1 \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$ if $i > m/2+1$. By Lemma 5.32 there exists a new c_2 such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid as desired.

– Now suppose that (a_1, c_1, a_2, c_2) is invalid, (c_1, b_1, c_2, b_2) is valid, $f(a_1, a_2) = m/2$ and $f(b_1, b_2) < m/2$. If there is a c_2 such that (c_1, c_2) is valid and (c_1, b_1, c_2, b_2) is invalid, we can apply the previous case since the proof does not depend on what c_2 actually is. So from here we can assume that (c_1, b_1, c_2, b_2) is valid if (c_1, c_2) is valid. We will now consider every possible c_1.

  – If $c_1 = y_{\frac{m}{2}}$, then whatever new c_2 we pick, $f(c_1, c_2) = m/2 - 1 \geq f(a_1, a_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$. By Lemma 5.32 there exists a new c_2 such that (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid as desired.

  – The cases where $c_1 = y_{\frac{m}{2}}$, $c_1 = w_{\frac{m}{2}}$, $c_1 = t_{\frac{m}{2}}$, $c_1 = y_{\frac{m}{2}+2}$, $c_1 = w_{\frac{m}{2}+2}$ and $c_1 = t_{\frac{m}{2}+2}$ are analogous to the previous case.

  – If c_1 is Y left and $c_1 \neq y_{\frac{m}{2}}$, then c_2 is on U. Now since the atoms condition of (a_1, c_1, a_2, c_2) is not satisfied, either a_1 is on Y and a_2 is not on U or a_1 is not on Y and a_2 is on U. This however contradicts the fact that a_1 is Y if and only if a_2 is on U since the column of a_1 is less than $m/2+1$ and since $f(a_1, a_2) = m/2$.

  – If c_1 is Y right and $c_1 \neq y_{\frac{m}{2}+2}$, then c_2 is on V. Now since the atoms condition of (a_1, c_1, a_2, c_2) is not satisfied, either a_1 is on Y and a_2 is not on V or a_1 is not on Y and a_2 is on V. This however contradicts the fact that a_1 is on Y if and only if a_2 is on V, since the column of a_1 is greater than $m/2+1$ and since $f(a_1, a_2) = m/2$.

  – Suppose that c_1 is W left and $c_1 \neq w_{\frac{m}{2}}$. Then c_2 is on W'. If $a_1 = c_1$, then we pick $c_2 = a_2$, notice then that $f(c_1, c_2) = f(a_1, a_2) = m/2$.

    On the other hand, suppose that (a_1, c_1) is an edge, then we pick c_2 in the same column as c_1 such that (a_2, c_2) is an edge. This is possible since $f(a_1, a_2) > 0$. Notice now that $f(c_1, c_2) = f(a_1, a_2) = m/2$.

    Now suppose that $a_1 \neq c_1$ or (a_2, c_2) is not an edge. First, note that a_1 is on W or T and a_2 is on W' or V since $f(a_1, a_2) = m/2$, (a_1, c_1, a_2, c_2) is not valid and the column of a_1 is greater than $m/2+1$. Let us now pick c_2 in the same column as c_2 on W' if a_2 is on V and on V otherwise. Clearly then (a_1, c_1, a_2, c_2) is valid and $f(c_1, c_2) = m/2$.

  - The case where c_1 is T left and $c_1 \neq t_{\frac{m}{2}+2}$ is analogous to the previous case.
  - Suppose that c_1 is W right and $c_1 \neq w_{\frac{m}{2}+2}$. Then c_2 is on W'. If $a_1 = c_1$, then we pick $c_2 = a_2$, notice then that $f(c_1, c_2) = f(a_1, a_2) = m/2$.

    On the other hand, if (a_1, c_1) is an edge, then we pick c_2 in the same column as c_1 such that (a_2, c_2) is an edge. This is possible since $f(a_1, a_2) > 0$. Notice now that $f(c_1, c_2) = f(a_1, a_2) = m/2$.

    Now suppose that $a_1 \neq c_1$ or (a_2, c_2) is not an edge. First, note that a_1 is on W or T and a_2 is on W' or U since $f(a_1, a_2) = m/2$, (a_1, c_1, a_2, c_2) is not valid and the column of a_1 is greater than $m/2+1$. Let us now we pick c_2 in the same column as c_1 on W' if a_2 is on U and on U otherwise. Clearly then (a_1, c_1, a_2, c_2) is valid and $f(c_1, c_2) = m/2$.
  - The case where c_1 is T right and $c_1 \neq t_{\frac{m}{2}+2}$ is analogous to the previous case.

- The case where (c_1, b_1, c_2, b_2) is invalid, (a_1, c_1, a_2, c_2) is valid, $f(b_1, b_2) = m/2$ and $f(a_1, a_2) < m/2$ is analogous to the precious case. ∎

An analogous lemma holds when we switch the roles of c_1 and c_2.

Lemma 5.37: *If (a_1, b_1, a_2, b_2) is valid, $f(a_1, a_2) > 0$, $f(b_1, b_2) > 1$, and either one of $f(a_1, a_2), f(b_1, b_1)$ is less than $m/2$, 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 $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.*

For the same reasons why the proof of Lemma 5.35 was analogous to the proof of Lemma 5.34, the proof of the previous lemma is analogous to the proof of Lemma 5.36.

Combining Lemma 5.34 and Lemma 5.36 we get the following corollary.

Corollary 5.38: *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 directly implies the forth condition for a bisimulation game starting in (a_1, b_1, a_2, b_2).

Analogously, combining Lemma 5.35 and Lemma 5.37 we get the following corollary.

Corollary 5.39: *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 $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$.*

In the proof of the next lemma we will see that this corollary directly implies the back condition for a bisimulation game starting in (a_1, b_1, a_2, b_2).

Armed with the precious two corollaries we are finally ready to show that for a valid (a_1, b_1, a_2, b_2) we are allowed to play a bisimulation game of at most $\min(f(a_1, a_2), f(b_1, b_2))$ rounds.

Lemma 5.40: *Let s be a natural number and let m be a natural number divisible by 4. If $(a_1, b_1, a_2, b_2) \in \operatorname{adom}(G_1^m)^2 \times \operatorname{adom}(G_2^m)^2$ 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 prove this lemma by induction on s.
Induction Basis: Let $s = 0$. 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).
Induction Hypothesis: Suppose that our lemma holds for $s - 1$.
Induction Step: We will now show that our lemma holds for $s > 0$. To this end let $(a_1, b_1, a_2, b_2) \in \operatorname{adom}(G_1^m)^2 \times \operatorname{adom}(G_2^m)^2$ be valid such that $s \leq \min(f(a_1, a_2), f(b_1, b_2))$. If $s = 0$ our result follows from our induction hypothesis.

On the other hand, suppose that $s > 0$. If $f(b_1, b_2) = 1$, then Lemma 5.30 implies that for every $c_1 \in \operatorname{adom}(G_1^m)$ there exists $c_2 \in \operatorname{adom}(G_2^m)$ 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 $(G_1^m, a_1, c_1) \simeq_0 (G_2^m, a_2, c_2)$ and $(G_1^m, c_1, b_1) \simeq_0 (G_2^m, c_2, b_2)$. Hence the forth condition holds. Furthermore, Lemma 5.31 implies that for every $c_2 \in \operatorname{adom}(G_2^m)$ there exists $c_1 \in \operatorname{adom}(G_1^m)$ such that (a_1, c_1, a_2, c_2) and (c_1, b_2, c_2, b_2) 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. Now $(G_1^m, a_1, b_1) \simeq_1 (G_2^m, a_2, b_2)$ since the forth and back conditions hold.

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 5.38 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, since $s - 1 \leq \min(f(a_1, a_2), f(b_1, b_2)) - 1$, we have $f(c_1, c_2) \geq s - 1$, and thus $s - 1 \leq \min(f(c_1, c_2), f(a_1, a_2))$ and $s - 1 \leq \min(f(c_1, c_2), f(b_1, b_2))$. Now we can apply our induction hypothesis, which tells us that $(G_1, a_1, c_1) \simeq_{s-1} (G_2, a_2, c_2)$ and $(G_1, c_1, b_1) \simeq_{s-1} (G_2, c_2, b_2)$ as desired.

Let us now focus on the back condition. Suppose that $c_2 \in \operatorname{adom}(G_2^m)$. Then by Corollary 5.39 there exists $c_1 \in \operatorname{adom}(G_1^m)$ such that both (a_1, c_1, a_2, c_2) and (c_1, b_1, c_2, b_2) are valid and $f(c_1, c_2) \geq \min(f(a_1, a_2), f(b_1, b_2)) - 1$. Furthermore, since $s - 1 \leq \min(f(a_1, a_2), f(b_1, b_2)) - 1$, $f(c_1, c_2) \geq s - 1$, and thus $s - 1 \leq \min(f(c_1, c_2), f(a_1, a_2))$ and $s - 1 \leq \min(f(c_1, c_2), f(b_1, b_2))$. Now we can apply our induction hypothesis, which tells us that $(G_1, a_1, c_1) \simeq_{s-1} (G_2, a_2, c_2)$ and $(G_1, c_1, b_1) \simeq_{s-1} (G_2, c_2, b_2)$ as desired. Now $(G_1, a_1, b_1) \simeq_s (G_2, a_2, b_2)$ since the forth and back conditions hold, which concludes our proof. ■

The following corollary is at the heart of the proof of Proposition 5.25. Intuitively it states that for every (a_1, b_1) we can find (a_2, b_2) such that we can play a bisimulation game of $m/2 - 1$ rounds.

Corollary 5.41: *For every* $(a_1, b_1) \in \text{adom}(G_1^m)$ *there exists* $(a_2, b_2) \in \text{adom}(G_2^m)$ *such that* $(G_1, a_1, b_1) \simeq_{m/2-1} (G_2, a_2, b_2)$.

Proof: First if $(a_1, b_1) = (y_{m/2+1}, y_{m/2+2})$, then we pick $(a_2, b_2) = (u_{m/2+1}, u_{m/2+2})$. 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 5.40.

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

$$\begin{aligned}
a_1 = y_i \wedge 0 \leq i \leq m/2 + 1 &\implies a_2 \in U \\
a_1 = y_i \wedge m/2 + 1 \leq i \leq m + 1 &\implies a_2 \in V \\
a_1 \in W &\implies c_2 \in W' \\
a_1 = t_i \wedge 0 \leq i \leq m/2 + 1 &\implies a_2 \in V \\
a_1 = t_i \wedge m/2 + 1 < i \leq m + 1 &\implies a_2 \in U
\end{aligned}$$

We use the same strategy for b_1. Clearly in this case (a_1, b_1, a_2, b_2) is valid, and $f(a_1, a_2) = f(b_1, b_2) = m/2$. Now $(G_1, a_1, b_1) \simeq_{m/2-1} (G_2, a_2, b_2)$ due to Lemma 5.40. ■

Using the previous corollary we can finally prove Proposition 5.25.

Proof (of Proposition 5.25): First observe that $\mathcal{N}(^{-1}, ^{+}) \leq^{\text{bool}} \mathcal{N}(F_1)$. Hence it suffices to prove that $\mathcal{N}(^{-1}, ^{+}) \not\leq^{\text{bool}} \mathcal{N}(F_2)$. Furthermore, since $F_2 \subseteq \overline{\{\backslash, di, ^{+}\}}$, it follows from Theorem 5.8 that $\mathcal{N}(F_2) \leq^{\text{path}} \mathcal{N}(\backslash, di, ^{+})$ and therefore also $\mathcal{N}(F_2) \leq^{\text{bool}} \mathcal{N}(\backslash, di, ^{+})$. Thus it is sufficient to show that $\mathcal{N}(^{-1}, ^{+}) \not\leq^{\text{bool}} \mathcal{N}(\backslash, di, ^{+})$.

To this end consider the boolean query q expressed by $R^2 \circ (R \circ R^{-1})^{+} \circ R^2$. We will show that this query is not expressible in $\mathcal{N}(\backslash, di, ^{+})$.

Now, assume for the same of contradiction that q is expressible by a query $e \in \mathcal{N}(\backslash, di, ^{+})$. Furthermore, define $\mathcal{G}_n$ as the class of graphs with an active domain of size at most n. Remember that Lemma 4.10 tells us that when we only consider graphs in $\mathcal{G}_n$, expressions of the form f^{+} are equivalent with the expression $\cup_{i=1}^{n} f^i$. Now let d be the degree of e. Define e_n as the expression e where every subexpression of the form f^{+} in e is replaced with $\cup_{i=1}^{n} f^i$ where f^i is constructed as in Corollary 5.29. By the same corollary, f^i has degree degree(f) + $\lceil \log_2 i \rceil \leq$ degree(f) + $\lceil \log_2 n \rceil$. Now, since in the worst case scenario every operation which contributes to the degree of e is a transitive closure application, e_n has at most degree $d\lceil \log_2 n \rceil$. Now, notice that e and e_n are equivalent on graphs in $\mathcal{G}_n$ by Lemma 4.10. Also, if $k = |\text{adom}(G_2^m)| = |\text{adom}(G_1^m)| = 3m + 7$, then

$$\frac{k}{6} - 3 = \frac{3m + 7}{6} - \frac{18}{6} = \frac{3m - 11}{6} = \frac{m}{2} - \frac{11}{6} \leq \frac{m}{2} - 1.$$

Furthermore there exists some $l' \in \mathbb{N}$ such that for every $l \geq l' : d\lceil \log_2 l \rceil \leq \frac{l}{6} - 3$

since

$$\begin{aligned}
0 \leq \lim_{k\to+\infty} \frac{d\lceil \log_2 k\rceil}{\frac{k}{6}-3} &= d \lim_{k\to+\infty} \frac{\lceil \log_2 k\rceil}{\frac{k}{6}-3} \\
&\leq d \lim_{k\to+\infty} \frac{\log_2 k + 1}{\frac{k}{6}-3} \\
&= d\left(\lim_{k\to+\infty} \frac{\log_2 k}{\frac{k}{6}-3} + \lim_{k\to+\infty} \frac{1}{\frac{k}{6}-3}\right) \\
&= d\left(\lim_{k\to+\infty} \frac{\log_2 k}{\frac{k}{6}-3} + 0\right) \\
&= d \lim_{k\to+\infty} \frac{\frac{1}{\log_e(2)k}}{\frac{1}{6}} \qquad \text{(L'Hôpital's rule[AE09])} \\
&= d \lim_{k\to+\infty} \frac{6}{\log_e(2)k} = d \cdot 0 = 0
\end{aligned}$$

Hence there exists a natural number m divisible by 4 such that $d\lceil \log_2(3m+7)\rceil \leq \frac{3m+7}{6} - 3$. Moreover, $e_{3m+7} \in \mathcal{N}(di, \backslash)_{m/2-1}$ since $\mathrm{degr}(e_{3m+7}) \leq d\lceil \log_2(3m+7)\rceil \leq \frac{3m+7}{6} - 3 \leq m/2 - 1$. Furthermore, notice that $e_{3m+7}(G_1^m) \neq \emptyset$ and $e_{3m+7}(G_2^m) = \emptyset$, since $q(G_1^m) = \mathit{true}$, $q(G_2^m) = \mathit{false}$ and $|\mathrm{adom}(G_1^m)| = |\mathrm{adom}(G_2^m)| = 3m + 7$. Also, Corollary 5.41 tells us that for every $(a_1, b_1) \in \mathrm{adom}(G_1^m)^2$ there exists $(a_2, b_2) \in \mathrm{adom}(G_2^m)$ such that $(G_1, a_1, b_1) \simeq_{m/2-1} (G_2, a_2, b_2)$. This, however, implies that the boolean query expressed by e_{3m+7} is not expressible in $\mathcal{N}(di, \backslash)_{m/2-1}$ due to Proposition 3.5, which contradicts the existence of e_{3m+7}. Hence q is not expressible in $\mathcal{N}(di, \backslash, {}^+)_d$ and since d was arbitrary it is not expressible in $\mathcal{N}(di, \backslash, {}^+)$ either. ∎

5.4 Languages with $\cap$ and with $^+$

In this section we will prove Theorem 5.9 restricted to languages in $\mathcal{C}[\cap, {}^+]$. To this end, we will first show that the boolean counterpart of Proposition 4.21 and Lemma 4.22 hold for languages in $\mathcal{C}[\cap]$.

Proposition 5.42: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be languages in $\mathcal{C}[\cap]$. Then, $\mathcal{N}(F_1 \cup \{^+\}) \leq^{\mathrm{bool}} \mathcal{N}(F_2 \cup \{^+\})$ if and only if $\mathcal{N}(F_1) \leq^{\mathrm{bool}} \mathcal{N}(F_2)$.*

PROOF: We will first show the 'if' direction. By Remark 5.24 $\leq^{\mathrm{bool}}$ coincides with $\leq^{\mathrm{path}}$ for languages in $\mathcal{C}[\cap]$. Hence, $\mathcal{N}(F_1) \leq^{\mathrm{path}} \mathcal{N}(F_2)$, since $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ are both languages of $\mathcal{C}[\cap]$ and $\mathcal{N}(F_1) \leq^{\mathrm{bool}} \mathcal{N}(F_2)$. Now by Proposition 4.21 $\mathcal{N}(F_1 \cup \{^+\}) \leq^{\mathrm{path}} \mathcal{N}(F_2 \cup \{^+\})$, which implies $\mathcal{N}(F_1 \cup \{^+\}) \leq^{\mathrm{bool}} \mathcal{N}(F_2 \cup \{^+\})$.

For the 'only if' direction we will show that $\mathcal{N}(F_1) \not\leq^{\mathrm{bool}} \mathcal{N}(F_2)$ implies $\mathcal{N}(F_1 \cup \{^+\}) \not\leq^{\mathrm{bool}} \mathcal{N}(F_2 \cup \{^+\})$. To this end assume that $\mathcal{N}(F_1) \not\leq^{\mathrm{bool}} \mathcal{N}(F_2)$. Corollary 5.23 tells us that this implies that $\mathcal{N}(F_1) \not\leq^{\mathrm{bool}}_{\mathrm{strong}} \mathcal{N}(F_2)$. The result now follows directly from Lemma 4.22. ∎

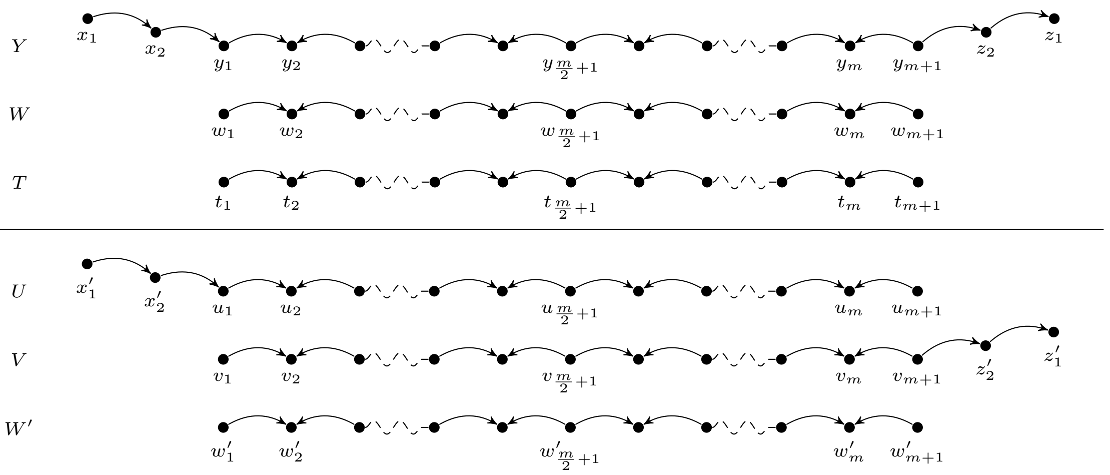


Figure 5.2: Graphs G_1^m (top) and G_2^m (bottom) used to establish boolean separation in the proof of Proposition 5.25.

Theorem 5.8 for languages in $\mathcal{C}[^+]$ is directly implied by previous proposition.

Corollary 5.43: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be languages in $\mathcal{C}[\cap, ^+]$. Then, $F_1 \not\subseteq \widetilde{F_2}$ implies $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$.*

Proof: Notice that both $\mathcal{N}(F_1 \setminus \{^+\})$ and $\mathcal{N}(F_2 \setminus \{^+\})$ are languages in $\mathcal{C}[\cap]$. Furthermore, since $^+$ is present in both F_1 and F_2,

$$F_1 \subseteq \widetilde{F_2} \iff F_1 \setminus \{^+\} \subseteq \widetilde{F_2} \setminus \{^+\} = \overline{F_2} \setminus \{^+\} = \overline{F_2 \setminus \{^+\}} = \widetilde{F_2 \setminus \{^+\}}.$$

Hence $\mathcal{N}(F_1 \setminus \{^+\}) \not\leq^{\text{bool}} \mathcal{N}(F_2 \setminus \{^+\})$ by Proposition 5.22, which by Proposition 5.42 implies that $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$ as desired. ■

For languages in $\mathcal{C}[\cap]$ and $\mathcal{C}[^+]$ we already noticed that $\leq^{\text{bool}}$ coincided with $\leq^{\text{path}}$; the previous corollary together with Proposition 5.6 tells us that this observation also holds for languages in $\mathcal{C}[\cap, ^+]$.

5.5 Cross-relationships between $\mathcal{C}$, $\mathcal{C}[\cap]$, $\mathcal{C}[^+]$ and $\mathcal{C}[\cap, ^+]$

In the previous sections we have proven Theorem 5.9. As mentioned before, that theorem does not yet cover languages over different classes. In this section we will close this final gap. First, however, we need the following proposition.

Proposition 5.44 (Primitivity of $^{-1}$): *Let F_1 and F_2 be sets of nonbasic features. If $^{-1} \in \overline{F_1}$, and $F_2 \subseteq \{\setminus, \cap, ^+\}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

Proof: The graphs in Figure 4.4b are distinguishable in $\mathcal{N}(F_1)$ by the query $R^2 \circ R^{-1} \circ R^2$. By the brute force method described in Section 3.1 they are indistinguishable in

$\mathcal{N}(F_2)$, which implies that the boolean query $R^2 \circ R^{-1} \circ R^2 \neq \emptyset$ is not expressible in $\mathcal{N}(F_2)$. ■

We are now ready to provide a separation result for languages among different classes.

Proposition 5.45: *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be in different classes among $\mathcal{C}, \mathcal{C}[\cap], \mathcal{C}[^+]$ or $\mathcal{C}[\cap, ^+]$. If $\widehat{F_1} \not\subseteq \overline{F_2}$ and $F_1 \not\subseteq \widetilde{F_2}$ then $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$.*

PROOF: We will consider all combinations of languages among different classes one by one.

Let $\mathcal{N}(F_1)$ be in $\mathcal{C}$ and let $\mathcal{N}(F_2)$ be in an arbitrary other class (not $\mathcal{C}$). Notice that $\widetilde{F_2} = \overline{F_2}$, since at least one of $^+$ or $\cap$ is present $\overline{F_2}$. Also, note that by definition $F_1 \subseteq \{\pi, \overline{\pi}, ^{-1}, di\}$. Hence $F_1 \not\subseteq \widetilde{F_2} = \overline{F_2}$ if and only if $F_1 \not\subseteq \overline{F_2} \setminus \{^+, \setminus, \cap\}$. Therefore if $F_1 \not\subseteq \overline{F_2}$, one feature $x \in \{\pi, \overline{\pi}, ^{-1}, di\}$ is present in F_1 but missing in $\overline{F_2} \setminus \{^+, \setminus, \cap\}$. We will now consider each such possibility.

- $x = \pi$. First notice that $\overline{\pi}$ cannot be present in $\overline{F_2}$. Now, if $\cap \in \overline{F_2}$, then $\overline{\pi}, di, ^{-1} \notin \overline{F_2}$ and thus $F_2 \subseteq \{^+, \cap, \setminus\}$. Hence Proposition 4.18 can be applied which proves the result.

  On the other hand, if $\cap \notin \overline{F_2}$, then $\setminus \notin F_2$, and hence $F_2 \subseteq \{di, ^{-1}, ^+\}$. The result now follows directly from Proposition 4.8.

- $x = \overline{\pi}$. First notice that since $\overline{\pi} \in F_1$, $\pi \in \overline{F_1}$. Now, if $\setminus \notin \overline{F_2}$, then the result follows directly from Proposition 4.6. On the other hand if $\setminus \in \overline{F_2}$, then $\pi \notin \overline{F_2}$. The result now follows from the case where $x = \pi$, which we have already proven in the previous case.

- $x = di$. Here Proposition 4.5 can be applied which proves the result.

- $x = ^{-1}$. In this case it will not suffice to only consider $F_1 \not\subseteq \widetilde{F_2}$. Since $^{-1}$ is present in F_1 and $\mathcal{N}(F_1) \in \mathcal{C}$, $\widehat{F_1} = (F_1 \setminus \{^{-1}\}) \cup \{\pi\}$. It is also given that $\widehat{F_1} \not\subseteq \overline{F_2}$. Thus there exits a feature $y \in \widehat{F_1}$ which is not present in $\overline{F_2}$. Notice that this feature is not $^{-1}$. Now, if $y \neq \pi$, then $y = \overline{\pi}$ or $y = di$. Since in this case y is also present in F_1, but not in $\overline{F_2}$, the result follows from one of the previous cases.

  On the other hand, if $y = \pi$, then by the interdependencies discussed at the beginning of Section 4.1

  $$\pi, ^{-1} \notin \overline{F_2} \wedge \cap \in \overline{F_2} \implies di, \overline{\pi} \notin \overline{F_2} \implies F_2 \subseteq \{\cap, \setminus, ^+\}$$

  and also

  $$\pi, ^{-1} \notin \overline{F_2} \wedge \cap \notin \overline{F_2} \implies \setminus, \overline{\pi} \notin \overline{F_2} \implies F_2 \subseteq \{di, ^+\}.$$

  Hence either Proposition 5.10 or Proposition 5.44 can be applied which concludes our proof in this case.

Let $\mathcal{N}(F_1)$ be in $\mathcal{C}[\cap]$ and $\mathcal{N}(F_2)$ be in $\mathcal{C}[\cap, ^+]$. If $\setminus \in F_1$ and $\setminus \notin \overline{F_2}$, we can apply Proposition 4.17. Otherwise, clearly, $F_1 \nsubseteq \overline{F_2}$ if and only if $F_1 \setminus \{\setminus, \cap\} \nsubseteq \overline{F_2} \setminus \{\setminus, \cap, ^+\}$. Hence one of $\pi, \overline{\pi}, di, ^{-1}$ is present in F_1, but lacking in $\overline{F_2}$. If that feature is di, π or $\overline{\pi}$ we can use the proof of case 1. On the other hand, if that feature is $^{-1}$ we can apply Proposition 5.21, since $\cap$ is present in $\overline{F_1}$.

If $\mathcal{N}(F_1)$ is in $\mathcal{C}[^+]$ and $\mathcal{N}(F_2)$ is in $\mathcal{C}[\cap, ^+]$, then $F_1 \nsubseteq \widetilde{F_2} = \overline{F_2}$ if and only if $F_1 \setminus \{^+\} \nsubseteq \overline{F_2} \setminus \{^+\}$. Hence one of $\pi, \overline{\pi}, di, ^{-1}$ is present in F_1 but missing in $\overline{F_2}$. Again, if this feature is di, π or $\overline{\pi}$ we can use the proof of case 1. On the other hand, if that feature is $^{-1}$, then we can apply Proposition 5.25, which implies our result, since $^+ \in \overline{F_2}$.

In any other scenario for $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$, one of $\cap$ or $^+$ is present in $\overline{F_1}$, but not in $\overline{F_2}$ — hence either Proposition 4.25 in the former or Proposition 4.28 latter scenario prove our proposition. ■

5.6 Proof of Theorem 5.8

Since Proposition 5.6 and Proposition 5.7 combined prove the only if direction of Theorem 5.8, the only thing left to show is the if direction. To this end consider its contrapositive, i.e., we want to show that for arbitrary sets of nonbasic features F_1 and F_2, if $F_1 \nsubseteq \widetilde{F_2}$ and $\widehat{F_1} \nsubseteq \overline{F_2}$, then, $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$. Throughout Sections 5.1, 5.2, 5.3 and 5.4 we established separation for languages within the same classes. However, as mentioned before, the characterization of $\leq^{\text{bool}}$ is less complicated in this case. Obviously, these results imply that Theorem 5.8 also holds for languages within the same classes. On the other hand, we mentioned that for languages over different classes this simplified version does not hold anymore. However, in Section 5.5 we did show that separation on the level of boolean queries for languages in different classes requires an extra condition. Let us now recite these results. Proposition 5.11 established boolean separation for languages in $\mathcal{C}$. This yields the Hasse diagram in Figure 5.3, where there is a directed path from $\mathcal{N}(F_1)$ to $\mathcal{N}(F_2)$ if and only if $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$. Note that the boxed features represent the minimal set of nonbasic features from which the other features can be derived by the appropriate interdependencies introduced at the start of Chapter 4.

$$\begin{tikzcd}
 & \mathcal{N}(\boxed{^{-1}, di, \overline{\pi}}, \pi) = \mathcal{N}(\boxed{di, \overline{\pi}}, \pi) \\
\mathcal{N}(\boxed{^{-1}, \overline{\pi}}, \pi) = \mathcal{N}(\boxed{\overline{\pi}}, \pi) \arrow[ur] & \\
 & \mathcal{N}(\boxed{^{-1}, di, \pi}) = \mathcal{N}(\boxed{di, \pi}) \arrow[uu] \\
\mathcal{N}(\boxed{^{-1}, \pi}) = \mathcal{N}(\boxed{\pi}) \arrow[uu] \arrow[ur] & \\
 & \mathcal{N}(\boxed{^{-1}, di}) \arrow[uu] \\
\mathcal{N}(\boxed{^{-1}}) \arrow[uu] \arrow[ur] & \\
 & \mathcal{N}(\boxed{di}) \arrow[uu] \\
\mathcal{N} \arrow[uu] \arrow[ur] &
\end{tikzcd}$$

Figure 5.3: The Hasse diagram of $\leq^{\text{bool}}$ for languages in $\mathcal{C}$.

We established boolean separation for languages in $\mathcal{C}[\cap]$ in Proposition 5.22; for languages in $\mathcal{C}[^+]$ in Proposition 5.26; and for languages in $\mathcal{C}[\cap, ^+]$ in Corollary 5.43. As mentioned before, these results mirror the result for $\leq^{\text{path}}$ — hence the Hasse diagrams for $\leq^{\text{bool}}$ and $\leq^{\text{path}}$ for languages in $\mathcal{C}[\cap]$, $\mathcal{C}[^+]$ and $\mathcal{C}[\cap, ^+]$ are equal. These Hasse diagrams can be found in Figure 4.8.

Proposition 5.45 ties all these results together, yielding Theorem 5.8. We can obtain the Hasse diagram for general boolean queries from the Hasse diagrams of $\leq^{\text{bool}}$ for $\mathcal{C}$, $\mathcal{C}[\cap]$, $\mathcal{C}[^+]$ and $\mathcal{C}[\cap, ^+]$ as follows:

- Add inclusion arrows from the Hasse diagram of $\leq^{\text{bool}}$ for $\mathcal{C}[\cap]$ and $\mathcal{C}[^+]$ to the Hasse diagram of $\leq^{\text{bool}}$ for $\mathcal{C}[\cap, ^+]$;

- Connect the Hasse diagram of $\mathcal{C}$ to the Hasse diagram of $\mathcal{C}[\cap]$ by adding arrows from $\mathcal{N}$ to $\mathcal{N}(\boxed{\cap})$, $\mathcal{N}(\boxed{\pi, di})$ to $\mathcal{N}(\boxed{\cap, di}, \pi)$, from $\mathcal{N}(\boxed{\pi})$ to $\mathcal{N}(\boxed{\cap, \pi})$, $\mathcal{N}(\boxed{\overline{\pi}}, \pi)$ to $\mathcal{N}(\boxed{\cap, \overline{\pi}}, \pi)$ and $\mathcal{N}(\boxed{di, \overline{\pi}}, \pi)$ to $\mathcal{N}(\boxed{\cap, di, \overline{\pi}})$;

- Connect the diagram of $\mathcal{C}$ to the diagram of $\mathcal{C}[^+]$ by adding an arrow for every $\mathcal{N}(F_1)$ in the diagram of $\mathcal{C}$ to $\mathcal{N}(F_1 \cup \{^+\})$ in the diagram of $\mathcal{C}[\cap]$.

6 Queries on Unlabeled Graphs

In the previous two chapters we considered boolean and path separation for boolean and path queries over graphs which were allowed to have multiple edge labels. We did, however, supply a lot of separation results whose proofs only relied on unlabeled graphs, i.e., graphs with a single edge label. In this chapter we will explore which of the separation results still hold when we only consider unlabeled graphs. We will write $\mathcal{N}(F_1) \leq^{\text{path}}_{\text{unl}} \mathcal{N}(F_2)$ (resp. $\mathcal{N}(F_1) \leq^{\text{bool}}_{\text{unl}} \mathcal{N}(F_2)$) if every path (resp. boolean) query expressible on unlabeled graphs in $\mathcal{N}(F_1)$ is also expressible in $\mathcal{N}(F_2)$. Conversely, we will write $\mathcal{N}(F_1) \not\leq^{\text{path}}_{\text{unl}} \mathcal{N}(F_2)$ (resp. $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{unl}} \mathcal{N}(F_2)$) if there exists a path (resp. boolean) query expressible in $\mathcal{N}(F_1)$ which is not expressible $\mathcal{N}(F_2)$ when only considering unlabeled input graphs.

Obviously, $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$ implies $\mathcal{N}(F_1) \leq^{\text{path}}_{\text{unl}} \mathcal{N}(F_2)$ and $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$ implies $\mathcal{N}(F_1) \leq^{\text{bool}}_{\text{unl}} \mathcal{N}(F_2)$. For boolean queries, the converse does not necessarily hold, as we will see later.

Notice that out of all the separation results, only the proof of Proposition 4.28 relies on multiple edge labels. Therefore all the separation results which do not rely on the presence of the transitive closure in $\mathcal{N}(F_1)$ and the lack thereof in $\mathcal{N}(F_2)$ still hold. In the remainder of this chapter we will refer to the separation results which do rely this phenomenon as the separation results involving transitive closure. We can, however, prove a version of Proposition 4.28 for $\leq^{\text{path}}_{\text{unl}}$.

Proposition 6.1 (Primitivity of $^+$): *Let F_1 and F_2 be sets of nonbasic features. If $^+ \in \overline{F_1}$ and $^+ \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{path}}_{\text{unl}} \mathcal{N}(F_2)$.*

Proof: As mentioned before, it is a well known fact that the transitive closure of binary relations is not expressible in first order logic, see e.g., [AU79, Gys12]. Since every expression in $\mathcal{N}(F_2)$ is also expressible in first order logic, it follows directly that the transitive closure is not expressible in $\mathcal{N}(F_2)$, which concludes our proof. ■

Hence $\leq^{\text{path}}$ and $\leq^{\text{path}}_{\text{unl}}$ coincide — more formally:

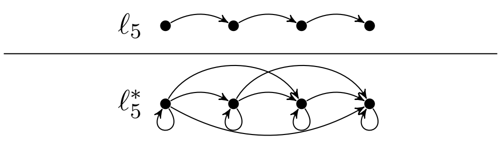


Figure 6.1: ℓ_5 (top) and ℓ_5^* (bottom).

Proposition 6.2: *Let F_1 and F_2 be sets of nonbasic features. Then, $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$ if and only if $\mathcal{N}(F_1) \leq^{\text{path}}_{\text{unl}} \mathcal{N}(F_2)$.*

6.1 Separation of boolean queries

As mentioned before, there are some separation results of $\leq^{\text{bool}}_{\text{unl}}$ involving transitive closure that do coincide with $\leq^{\text{bool}}$. Before we can prove this we need a few techniques from finite model theory.

Define ℓ_n as a linear directed chain with n nodes and define ℓ_n^* as the reflexive-transitive closure of ℓ_n, see Figure 6.1 for an example of ℓ_4 and ℓ_4^*. The graph ℓ_n^* is also called a linear order of size n.

We now cite a well known result involving linear orders, for a proof, see e.g., [AHV95, Gys12].

Theorem 6.3: *The parity query for linear orders is not expressible in first order logic.*

We will now use this result to prove that there exist no sentence in first order logic which expresses whether a graph contains a cycle. Before we do so, however, notice that we can extract ℓ_n from ℓ_n^* with the following first order query:

$$\varphi_\ell(x,y) := x \neq y \wedge R(x,y) \wedge \neg(\exists z : y \neq z \wedge x \neq z \wedge R(x,z) \wedge R(z,y))$$

i.e., $\ell_n^* \models \varphi_\ell[x,y]$ if and only if $(x,y) \in \ell_n$. We will use this first order formula to prove that the cycle query is not expressible in first order logic.

Proposition 6.4: *The boolean query which expresses whether a finite graph contains a cycle is not expressible in first order logic.*

PROOF: We will show that the existence of the 'cycle' query ψ contradicts Theorem 6.3. To this end, let ℓ_n be a linear chain of size $n > 2$. We now define a new graph g_n as follows:

- g_n contains n nodes, $x_1, \ldots, x_n$;
- Add an edge from x_i to x_{i+2} for every $i \in \{1, \ldots, n-2\}$.

Clearly, there is a path from x_k to x_l if and only if $k < l$, k and l are both even or both odd. Thus, g_n is a graph consisting of two distinct chains, one starting in x_1 and the

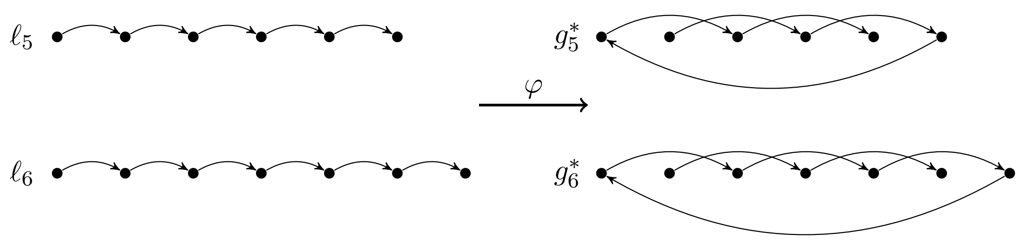


Figure 6.2: The graphs g_6^* (top) and g_7^* (bottom) constructed from ℓ_6 (top) and ℓ_7 (bottom).

other starting in x_2. By the previous observation, the first chain ends in x_n and the second chain ends in x_{n-1} if and only if n is odd. On the other hand, the first chain ends in x_{n-1} and the second chain ends in x_n if and only if n is even.

Now define g_n^* from g_n by adding an edge from x_n to x_1 (see Figure 6.2 for an example). If n is odd, there is a cycle in g_n^* around x_1 by our previous observation. If n is even, the chain starting in x_2 and ending in x_n is connected to the chain starting in x_1 and ending in x_{n-1}. Since both chains are disjoint and since both chains share the same direction, the graph g_n^* is isomorphic to ℓ_n and hence does not contain a cycle. Thus g_n^* contains a cycle if and only if n is odd.

Let us now define a first order formula $\varphi(x, y)$, such that $(a, b) \in g_n^*$ if and only if $\ell_n \models \varphi[a, b]$. Clearly

$$\varphi(x, y) = (\exists a : R(x, a) \wedge R(a, y)) \vee ((\neg\exists a : R(a, x)) \wedge \neg\exists a : R(y, a)) \qquad (6.1)$$

fulfills this property. Moreover, if we replace all the $R(x, y)$ occurrences in φ with $\varphi_\ell(x, y)$, we get $\ell_n^* \models \varphi[a, b]$ if and only if $(a, b) \in g_n^*$.

Now, if we also replace all the $R(x, y)$ occurrences in ψ with $\varphi(x, y)$ to get a new sentence ϑ we get the following: $\ell_n^* \models \vartheta$ if and only if n is odd, which contradicts Theorem 6.3. ∎

We can now prove our first separation result of $\leq_{\text{unl}}^{\text{bool}}$ for languages involving transitive closure.

Proposition 6.5: *Let F_1 and F_2 be sets of nonbasic features. If $^+ \in \overline{F_1}, \cap \in \overline{F_1}$ and $^+ \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq_{\text{unl}}^{\text{bool}} \mathcal{N}(F_2)$.*

Proof: Notice that the 'cycle' query is expressible in $\mathcal{N}(F_1)$ by the query $R^+ \cap id$. On the other hand, the 'cycle' query is not expressible $\mathcal{N}(F_2)$, since the 'cycle' query is not expressible in first order logic by Proposition 6.4 and since every query in $\mathcal{N}(F_2)$ is expressible in first order logic due to Proposition 2.6. ∎

For the second separation result of $\leq_{\text{unl}}^{\text{bool}}$ for languages involving transitive closure, we will need that the boolean query expressed by $R^2 \circ (R \circ R^{-1}) \circ R^2 \neq \emptyset$ is not expressible in first order logic. Classically this is done by showing that it is not expressible in FO$[k]$

by supplying two graphs and then show that the duplicator has a winning strategy in a k-round Ehrenfeucht-Fraïssé game played on those two graphs. This, however, is rather difficult when the structure of the graphs becomes complex. Hence why we will use another technique from finite model theory, called the Hanf-locality. First, however, we need a few preliminary definitions.

Definition 6.6: Let G be an unlabeled graph. Define $d_G(x, y)$ as the shortest distance between x and y in G when one considers G as an undirected graph. If there is no such path define $d_G(x, y)$ as infinity. Furthermore, if $x = y$, then define $d_G(x, y)$ as 0. If it is clear in which graph G we are expressing the distance we will omit the G subscript. $\diamond$

The goal is to inspect graphs locally, i.e., to consider subgraphs of a particular graph G around a node a. To this end consider the following definition.

Definition 6.7: Let G be an unlabeled graph, let $a \in \text{adom}(G)$ and let r be a natural number. The ball with radius r around a is the set

$$B_r^G(a) = \{x \in \text{adom}(G) \mid d_G(x, a) \leq r\}.$$

The r-neighborhood of a in G, denoted by $N_r^G(a)$, is the tuple (G', a) where the graph G' is defined as follows:

– Its nodes are precisely $B_r^G(a)$;

– Its edge relation is $G \cap B_r^G(a)^2$. $\diamond$

We now extend our usual definition of graph isomorphisms to (G, a). We say that (G, a) and (G', b) are isomorphic, denoted by $(G, a) \cong (G, b)$, if there exists a graph isomorphism f from G to G' such that $f(a) = b$.

A lot of winning strategies of the duplicator in an Ehrenfeucht-Fraïssé game are based on that first order formulas are local, i.e., intuitively they can only 'select' nodes which are at most a fixed number of steps away from the free variables. To formalize this, we first to have define what it means for graphs to be locally similar.

Definition 6.8: Let G_1 and G_2 be graphs, and let d be a natural number. We write $G_1 \leftrightarrows_d G_2$ if there exists a bijection $f : \text{adom}(G_1) \to \text{adom}(G_2)$ such that for every $c \in \text{adom}(A)$: $N_d^{G_1}(c) \equiv N_d^{G_2}(f(c))$. $\diamond$

We will now define what it means for a boolean query to be local.

Definition 6.9 (Hanf-locality): Let q be a boolean query on graphs. We say that q is Hanf-local if there exists a natural number d such that for every pair of finite unlabeled graphs G_1 and G_2 the following holds:

$$G_1 \leftrightarrows_d G_2 \implies (q(G_1) = \mathit{true} \iff q(G_2) = \mathit{true}) \qquad \diamond$$

In the motivation for this technique we said that intuitively first order logic is local. The following theorem justifies this intuition — for a proof see for example [Lib04].

Theorem 6.10: *Any boolean query expressible in first order logic is Hanf-local.*

To show that the query expressed by $R^2 \circ (R \circ R^{-1})^+ \circ R^2$ is not expressible in first order logic we will prove that it is not Hanf-local. To this end, we will first show that the graphs G_1^n and G_2^n displayed in Figure 6.3 are locally equivalent.

Lemma 6.11: *Let G_1^n be the graph at the top and G_2^n at the bottom of Figure 6.3. Then, $G_1^n \leftrightarrows_{n-1} G_2^n$.*

Proof: Define $f : \mathrm{adom}(G_1^n) \to \mathrm{adom}(G_2^n)$ be defined as follows:

$$f(c) = \begin{cases} v_i, & \text{if } c = x_i \text{ where } n < i \leq 2n-1 \\ u_i, & \text{if } c = y_i \text{ where } n < i \leq 2n-1 \\ u_j, & \text{if } c = x_j \text{ where } 1 \leq j \leq n \\ v_j, & \text{if } c = y_j \text{ where } 1 \leq j \leq n \\ a, & \text{if } c = x \\ b, & \text{if } c = y \end{cases}$$

We will show that $N_{n-1}^{G_1^n}(c) \cong N_{n-1}^{G_2^n}(f(c))$ for all $c \in \mathrm{adom}(G_1^n)$, which is exactly the condition for $G_1^n \leftrightarrows_{n-1} G_2^n$. First notice that if $c \in A = [x_1, x_n] \cup [y_1, y_n] \cup \{x\}$, then $d(c, y) > n-1$ and $d(f(c), b) > n-1$. Hence $y \notin \mathrm{adom}(N_{n-1}^{G_1^n}(c))$ and $b \notin \mathrm{adom}(N_{n-1}^{G_2^n}(f(c)))$. Therefore, when we consider G_1^n without y and G_2^n without b, they are isomorphic, where the isomorphism between them is defined as follows:

$$g(w) = \begin{cases} u_j, & \text{if } w = x_j \text{ where } 1 \leq j \leq 2n-1 \\ v_j, & \text{if } w = y_j \text{ where } 1 \leq j \leq 2n-1 \\ a, & \text{if } w = x \end{cases}$$

Therefore, g restricted to the appropriate nodes also yields an isomorphism between $N_{n-1}^{G_1^n}(c)$ and $N_{n-1}^{G_2^n}(f(c))$ since in this case $f(c) = g(c)$.

If on the other hand $c \notin A$, then $d(c, y) \leq n-1$ and $d(c, x) > n-1$, and thus $y \in N_{n-1}^{G_1^n}(c)$ and $x \notin N_{n-1}^{G_1^n}$. Hence without considering direction, $N_{n-1}^{G_1^n}(c)$ has one of the following chain forms: $x_i, \ldots, x_{2n-1}, y, y_{2n-1}, \ldots, y_k$ or $x_i, \ldots, x_{2n-1}, y$ or $y, y_{2n-1}, \ldots, y_k$, where c is in the middle of these chains. We will only consider the first chain form as the proof for the other chains are analogous. Since f in a sense 'mirrors' c around the horizontal dotted line, $N_{n-1}^{G_2^n}(f(c))$ has the following chain form: $v_i, \ldots, v_{2n-1}, b, u_{2n-1}, \ldots, u_k$ when we do not consider the direction of the edges. Let us now define $g : N_{n-1}^{G_1^n}(c) \to N_{n-1}^{G_2^n}(f(c))$ as follows:

$$g(w) = \begin{cases} v_k, & \text{if } w = x_k \\ u_k, & \text{if } w = y_k \\ b, & \text{if } w = y \end{cases}$$

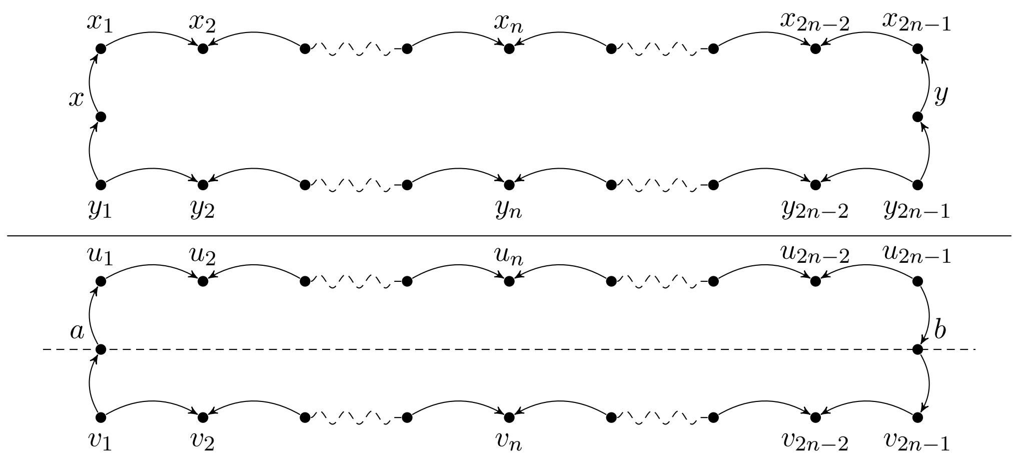


Figure 6.3: Graphs G_1^n (top) and G_2^n (bottom) used to establish separation in Proposition 6.12.

By inspecting Figure 6.3 and by the definition of g the following properties hold

$$\begin{aligned}
(x_k, x_l) \in N_{n-1}^{G_1^n}(c) &\iff (g(x_k), g(x_l)) = (v_k, v_l) \in N_{n-1}^{G_2^n}(g(c)) \\
(y_k, y_l) \in N_{n-1}^{G_1^n}(c) &\iff (g(y_k), g(y_l)) = (u_k, u_l) \in N_{n-1}^{G_2^n}(g(c)) \\
(y_{2n-1}, y) \in N_{n-1}^{G_1^n}(c) &\iff (g(y_{2n-1}), g(y)) = (u_{2n-1}, b) \in N_{n-1}^{G_2^n}(g(c)) \\
(y, x_{2n-1}) \in N_{n-1}^{G_1^n}(c) &\iff (g(y), g(x_{2n-1})) = (b, v_{2n-1}) \in N_{n-1}^{G_2^n}(g(c)).
\end{aligned}$$

Therefore, since the equations above cover all the scenarios such that $(u, v) \in N_{n-1}^{G_1^n}(c)$, and all $(u', v') \in N_{n-1}^{G_2^n}(f(c))$, g is a graph isomorphism. Furthermore, $g(c) = f(c)$ by definition of g, hence $N_{n-1}^{G_1^n}(c) \cong N_{n-1}^{G_2^n}(f(c))$ as desired. ∎

Utilizing the fact that queries in first order logic are Hanf-local, we can now prove that boolean query expressed by $R^2 \circ (R \circ R^{-1})^+ \circ R^2$ is not expressible in languages without transitive closure.

Proposition 6.12: *Let F be a set of non basic features such that $^+ \notin F$. Then, the boolean query $R^2 \circ (R \circ R^{-1})^+ \circ R^2 \neq \emptyset$ is not expressible in $\mathcal{N}(F)$.*

Proof: Suppose that the boolean query q expressed by $R^2 \circ (R \circ R^{-1})^+ \circ R^2 \neq \emptyset$ is expressible in $\mathcal{N}(F)$. Then certainly, it is also expressible in first order logic. Hence q is Hanf-local and thus by definition, there has to exists a natural number d such that for every finite graphs A and B, if $A \leftrightarrows_d B$, then q agrees on A and B. However, our previous proposition tells us that $G_1^{d+1} \leftrightarrows_d G_2^{d+1}$, but $q(G_1^{d+1})$ is true and $q(G_2^{d+1})$ is false, which contradicts that q is Hanf-local. ∎

We are now ready to prove the second separation result of $\leq_{\text{unl}}^{\text{bool}}$ for languages involving transitive closure.

Proposition 6.13: *Let F_1 and F_2 be sets of nonbasic features. If $^+ \in \overline{F_1}$, $^{-1} \in \overline{F_1}$ and $^+ \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{unl}} \mathcal{N}(F_2)$.*

PROOF: Proposition 6.12 tells us that the boolean query $R^2 \circ (R \circ R^{-1})^+ \circ R^2 \neq \emptyset$ is not expressible in first order logic. Hence it is also not expressible in $\mathcal{N}(F_2)$, since every boolean query expressible in $\mathcal{N}(F_2)$ is also expressible in first order logic. ■

Let us now move on to the third separation result of $\leq^{\text{bool}}_{\text{unl}}$ for languages involving transitive closure. We will first a few preliminary lemmas.

Lemma 6.14: *The boolean query "there is a non-sink node from which no sink node[1] can be reached" is expressible in $\mathcal{N}(\overline{\pi}, ^+)$.*

PROOF: Let G be an arbitrary graph. A node v is a sink node if there are no outgoing edges in G, i.e., for every $k \in \text{adom}(G)$: $(v, k) \notin R(G)$. Clearly, this is equivalent to $(v, v) \in \overline{\pi}_1(R)(G)$. We will now show that $\overline{\pi}_1((R^+ \circ \overline{\pi}_1(R)) \cup \overline{\pi}_1(R))(G) \neq$ if and only if there is a non-sink node in G from which no sink node can be reached. First, notice the following: $(a, s) \in R^+ \circ \overline{\pi}_1(R)(G)$ if and only if a reaches the sink node s. Moreover, a is not a sink node if and only if $(a, a) \in \overline{\pi}_1(G)$. Hence a is a non-sink node from which no sink node can be reached, if and only if, $(a, a) \in \overline{\pi}_1(R^+ \circ \overline{\pi}_1(R) \cup \overline{\pi}_1)(G)$. ■

It appears that the boolean query in the previous lemma is not expressible in first order logic.

Lemma 6.15: *The boolean query "there is a non-sink node from which no sink node can be reached" is not expressible in first order logic.*

PROOF: We will show that the existence of a first order sentence ψ boolean query which expresses the boolean query "there is a non-sink node from which no sink node can be reach" contradicts Theorem 6.3. Let us for the sake of contradiction assume that such a sentence ψ exists.

Remember that in the proof of Proposition 6.4 we established that g_n^* (see Figure 6.2) has a cycle if n is even and that g_n^* is isomorphic to ℓ^n if n is odd. Now, notice that if a graph has a cycle, it also has a non-sink node from which no sink node can be reached since there simply are no sink nodes. On the other hand, if a graph is isomorphic to ℓ_n, every non-sink node can reach a sink node. Hence, g_n^* contains a non-sink node from which no sink can be reached if and only if n is even. Hence $g_n^* \models \varphi$ if and only if n is even. Moreover, in the same proof we established that there exists a first order formula $\varphi(x, y)$ such that $\ell_n^* \models \varphi[a, b]$ if and only if $(a, b) \in g_n^*$. Now, if we replace every occurrence of $R(x, y)$ in ψ with $\varphi(x, y)$, then $\ell_n^* \models \psi$ if and only if n is even, which contradicts Theorem 6.3 as desired. ■

Armed with the previous two lemmas we are ready to prove our third separation separation result of $\leq^{\text{bool}}_{\text{unl}}$ for languages involving transitive closure.

[1]A sink node in a graph is a node in that graph with outdegree 0.

Proposition 6.16: *Let F be a set of nonbasic features such that $\overline{\pi} \in \overline{F}$ and ${}^{+} \notin \overline{F}$. Then, $\mathcal{N}(F \cup \{^{+}\}) \not\leq_{\text{unl}}^{\text{bool}} \mathcal{N}(F)$.*

Proof: The query boolean query “there is a non-sink node from which no sink node can be reached” is expressible in $\mathcal{N}(\overline{\pi}, {}^{+})$ by Lemma 6.14 and hence also in $\mathcal{N}(F \cup \{^{+}\})$. This query, however, is not expressible in first order logic by Lemma 6.15 and hence not in $\mathcal{N}(F)$ since every boolean query expressible in $\mathcal{N}(F)$ is also expressible in first order logic. ■

6.1.1 Collapse of boolean expressiveness of certain languages in $\mathcal{C}[^{+}]$

At the end of the previous section we showed that $\leq^{\text{bool}}$ and $\leq_{\text{unl}}^{\text{bool}}$ coincide for separation results involving transitive closure when ${}^{-1}$, $\cap$ or $\overline{\pi}$ is also present in the left hand language. In this section we will show that this does not hold in general. Before we can do this, however, we need several technical lemmas.

Lemma 6.17: *Let e be an expression in $\mathcal{N}(\pi, {}^{+})$, let G_1 and G_2 be graphs and let f be a homomorphism of graphs from G_1 to G_2. Then, $(a, b) \in e(G_1)$ implies that $(f(a), f(b)) \in e(G_2)$.*

Proof: We will prove this proposition by structural induction on e.
Induction Basis: If $e = R$, then the proposition follows directly from the fact that f is a homomorphism.
Induction Hypothesis: Suppose that our lemma holds for every subexpression of e.
Induction Step: We will now show that our lemma holds for e.

- $e = \pi_i(e_1)$. If $(a, a) \in \pi_1(e_1)(G_1)$, then there exists c such that $(a, c) \in e_1(G_1)$. By our induction hypothesis, $(f(a), f(c)) \in e_1(G_2)$, which by definition of π_1 implies that $(f(a), f(a)) \in \pi_1(e_1)(G_2)$ as desired. For $(a, a) \in \pi_2(e_1)(G_1)$ the proof is analogous.

- $e = e_1 \circ e_2$. Suppose that $(a, b) \in e(G_1)$. Then there exists c such that $(a, c) \in e_1(G_1)$ and $(c, b) \in e_2(G_2)$. Now, by our induction hypothesis $(f(a), f(c)) \in e_1(G_2)$ and $(f(c), f(b)) \in e_2(G_2)$, and hence $(f(a), f(b)) \in e(G)$.

- $e = e_1 \cup e_2$. If $(a, b) \in e(G_1)$, then $(a, b) \in e_1(G_1)$ or $(a, b) \in e_2(G_1)$. Now, by our induction hypothesis $(f(a), f(b)) \in e_1(G_2)$ or $(f(a), f(b)) \in e_2(G_2)$, and thus $(f(a), f(b)) \in e(G_2)$ as desired. ■

It appears that the output graph of a path query in $\mathcal{N}(\pi, {}^{+})$ is contained within the reflexive-transitive closure of the input graph, i.e., the expressions in $\mathcal{N}(\pi, {}^{+})$ literally only navigate over the paths in the input graph. Note that this is not necessarily the case for arbitrary languages $\mathcal{N}(F)$, e.g., the expression $di \cup id$ selects every pair of nodes of an input graph.

Lemma 6.18: *Let e be an expression in $\mathcal{N}(\pi, {}^+)$ and let G be an unlabeled graph. If $(a, b) \in e(G)$, then either $a = b$ or there exists a path from a to b in G.*

Proof: We will prove this proposition by structural induction on e.
Induction Basis: Clearly our lemma clearly holds for $e = R$ or $e = id$.
Induction Hypothesis: Suppose that our lemma holds for every subexpression on e.
Induction Step: We will now show that our lemma holds for e.

- $e = \pi(e_1)$. By the definition of π, $\pi(e') \subseteq id$, and thus if $(a, b) \in e(G)$, then $a = b$.

- $e = e_1 \circ e_2$. By definition of the composition operation, if $(a, b) \in e(G)$, then there exists c such that $(a, c) \in e_1(G)$ and $(c, b) \in e_2(G)$. Our hypothesis then tells us that $a = c$ or there exists a path from a to c in G and that $c = b$ or there exists a path from c to b in G. Clearly, if there is a path from a to c and from c to b, there also is a path from a to b. If $a = c$ and $c = b$, then $a = b$. On the other hand if there is a path from a to c and if $c = b$, there obviously is a path from a to b in G. In the case where there is a path from c to b and $a = c$ there also clearly is a path from a to b in G.

- $e = e_1 \cup e_2$. If $(a, b) \in e(G)$, then either $(a, b) \in e_1(G)$ or $(a, b) \in e_2(G)$. The result now follows directly from our induction hypothesis. ■

We want to show that every boolean query expressible in $\mathcal{N}(\pi, {}^+)$ can be expressed without transitive closure. It appears that we can mimic every such expression partially by a chain query, i.e., a query of the form R^m for some $m \geq 1$. Remember that ℓ_n is a linear chain with n nodes (see Figure 6.1).

Lemma 6.19: *For every expression e in $\mathcal{N}(\pi, {}^+)$, there exists a natural number m such that, for any unlabeled graph G, $e(G)$ is nonempty whenever $R^m(G)$ is nonempty.*

Proof: We will prove this by structural induction on e.
Induction Basis: If $e = R$ or $e = id$, we set $m = 1$. In the former the required condition clearly holds. On the other hand if $e = id$, then $id(G)$ is only empty when adom(G) is empty, and hence $R^1(G)$ is also nonempty as desired.
Induction Hypothesis: Suppose that our lemma holds for all subexpressions of e.
Induction Step: We will now show that our lemma holds for e.

- $e = e_1 \cup e_2$. By our induction hypothesis there exists m_1 and m_2 such that $e_1(G)$ is nonempty whenever $R^{m_1}(G)$ is nonempty and $e_2(G)$ is nonempty whenever $R^{m_2}(G)$ is nonempty. Hence setting m to m_1 would suffice since $e_1 \subseteq e_1 \cup e_2(G)$ is nonempty whenever $R^{m_1}(G)$ is nonempty.

- $e = \pi(e_1)$ or $e = e_1^+$. Here $e(G)$ is nonempty whenever $e_1(G)$ is empty — hence setting m to the m_1 associated with e_1 by our induction hypothesis is sufficient.

– $e = e_1 \circ e_2$. By our induction hypothesis there exists m_1 and m_2 such that whenever $R^{m_1}(G)$ is nonempty, $e_1(G)$ is nonempty and similarly, whenever $R^{m_2}(G)$ is nonempty, $e_2(G)$ is nonempty. We will now show that setting $m = m_1 + m_2$ does the job.

  By definition $R^{m_1}(\ell_{m_1})$ is nonempty and $R^{m_2}(\ell_{m_2})$ nonempty. Hence $e_1(\ell_{m_1})$ and $e_2(\ell_{m_2})$ are both nonempty by our induction hypothesis. Now let $(x_1, y_1) \in e_1(\ell_{m_1})$ and let $(x_2, y_2) \in e_2(\ell_{m_2})$. First, let us consider the case where $x_2 < y_1$ where the order is given by the linear chains. Then, let f_1 be the canonical injection from ℓ_{m_1} to ℓ_m such that the first node in ℓ_{m_1} is mapped to the first node in ℓ_m. Similarly, let f_2 be the canonical injection from ℓ_{m_2} to ℓ_m such that $f_2(x_2) = f_1(y_1)$. We can define f_2 in this way since $x_2 < y_1$, i.e., there are a sufficient number of nodes to the left of $f_1(y_1)$ to map the nodes in ℓ_{m_2} in front of x_2 to ℓ_m in a homomorphic fashion. Now, since f_1 and f_2 are homomorphisms by definition, Lemma 6.17 implies that $(f_1(x_1), f_1(y_1)) \in e_1(\ell_m)$ and $(f_1(y_1), f_2(y_2)) = (f_2(x_2), f_2(y_2)) \in e_2(\ell_m)$, and hence $(f_1(x_1), f_2(y_2)) \in e_1 \circ e_2(\ell_m)$. On the other hand, if $y_1 \leq x_2$, we can use a similar argument. Let f_2 be the canonical injection from ℓ_{m_2} to ℓ_m such that the first node in ℓ_{m_2} is mapped to the first node in ℓ_m. Furthermore, let f_1 be the canonical injection from ℓ_{m_1} to ℓ_m such that $f_1(y_1) = f_2(x_2)$. This construction of f_1 is possible since $y_1 \leq x_2$, i.e., there is enough room in front of $f_2(x_2)$ to map y_1 to $f_2(x_2)$ such that f_1 is a homomorphism from ℓ_{m_1} to ℓ_m. Again since f_1 and f_2 are homomorphisms, Lemma 6.17 implies that $(f_1(x_1), f_1(y_1)) \in e_1(\ell_m)$ and $(f_1(y_1), f_2(y_2)) = (f_2(x_2), f_2(y_2)) \in e_2(\ell_m)$. Hence $(f_1(x_1), f_2(y_2)) \in e_1 \circ e_2(\ell_m)$.

  Now assume that $R^m(G)$ is nonempty. Since R^m is an expression in $\mathcal{N}$ there exists a conjunctive query Q with a body isomorphic to ℓ_m which is equivalent to R^m by Lemma 4.16. Furthermore, since $Q(G)$ is nonempty there exists a matching f from the body B of Q into G which by definition is a homomorphism. Now let g be the isomorphism from ℓ_m to B, then clearly $f \circ g$ is a homomorphism from ℓ_m to G. Furthermore, by our previous discussion there exists $(a, b) \in e_1 \circ e_2(\ell_m)$, and hence $(f \circ g(a), f \circ g(b)) \in e_1 \circ e_2(G)$ as desired. ■

As said before this proposition, R^m only mimics e partially, i.e., it only mimics e when $R^m(G)$ is nonempty. Note that when $R^m(G)$ is empty, $R^{m'}(G)$ is also empty when $m' \geq m$. To categorize these graphs we need the following definition.

Definition 6.20: Let m be a natural number. We say that an unlabeled graph G is m-short if $R^m(G)$ is empty. ◇

It appears that expressions in $\mathcal{N}(\pi, {}^+)$ on m-short graphs can be mimicked exactly by an expression in $\mathcal{N}(\pi, {}^+)$.

Proposition 6.21: *Let e be an expression in $\mathcal{N}(\pi, {}^+)$ and let m be a natural number. There exists an expression $e' \in \mathcal{N}(\pi)$ such that $e'(G) = e(G)$ on any m-short graphs.*

Proof: We will show that for any expression e without transitive closure, e^+ can be expressed without transitive closure on m-short graphs. We will show that $e^+ \equiv \cup_{i=1}^{m-1} e^i$ on m-short graphs. By definition $\cup_{i=1}^{m-1} e^i \subseteq e^+$. Now let G be an arbitrary m-short graph and let $(x,y) \in e^+(G)$. If $(x,y) \in e^+(G)$ and $(x,y) \notin e(G)$, there exists $x = x_0, \ldots, x_n = y$, such that $(x_i, x_{i+1}) \in e(G)$ where $n \geq 2$. Without loss of generality we can assume that this path or cycle is simple in $e(G)$ by Lemma 4.10. Hence $x_i \neq x_{i+1}$ for all $i = 0 \ldots n-1$. Furthermore, by Lemma 6.18 there exists a path from x_i to x_{i+1}, i.e., $(x_i, x_{i+1}) \in R^j(G)$ for some $j > 0$. Now since each such path has at least length 1, there exists a path from x to y in G of at least length $n \leq k$, i.e., $(x,y) \in R^n(G)$. This implies that $n \leq k \leq m-1$ since $R^{m'}(G)$ is empty for all $m' \geq m$ and thus $(x,y) \in \cup_{i=1}^{m-1} e^i$ as desired. ∎

Armed with the two previous propositions we are now ready to show that $\leq_{\text{unl}}^{\text{bool}}$ does not necessarily coincide with $\leq^{\text{bool}}$.

Proposition 6.22: *The following collapses of languages involving $^+$ occur at the level of boolean queries on unlabeled graphs:*

(a) $\mathcal{N}(^+) \leq_{\text{unl}}^{\text{bool}} \mathcal{N}$;

(b) $\mathcal{N}(\pi, ^+) \leq_{\text{unl}}^{\text{bool}} \mathcal{N}(\pi)$;

Proof: Let us first consider (b). Let e be an expression in $\mathcal{N}(\pi, ^+)$ and let G be an arbitrary graph. Lemma 6.19 implies the existence of a natural number m such that $e(G)$ is nonempty whenever $R^m(G)$ is nonempty. On the other hand, if $R^m(G)$ is empty, G is m-short. Now, Proposition 6.21 tells us that there exists an expression $e' \in \mathcal{N}(\pi)$ such that $e'(G) = e(G)$. Hence if G is m-short $e'(G)$ is nonempty if and only if $e(G)$ is nonempty. Clearly $e' \cup R^m \in \mathcal{N}(\pi)$ is equivalent to e at the level of boolean queries as desired.

Since this proof does not depend on the presence of π, (a) also holds. ∎

The goal of this section and the previous section is to characterize $\mathcal{N}(F_1) \leq_{\text{unl}}^{\text{bool}} \mathcal{N}(F_2)$ for languages involving transitive closure. So far we already covered the scenarios where $^{-1} \in \overline{F_1}$ or $\cap \in \overline{F_1}$ or $\overline{\pi} \in \overline{F_1}$ or $F_1 = \{^+\}$ or $F_1 = \{\pi, ^+\}$. Hence the only scenarios left to consider are those where $F_1 = \{di, ^+\}$ or $F_1 = \{di, \pi, ^+\}$. It appears that a collapse also appears in these two scenarios.

Proposition 6.23: *Let $F \subseteq \{\pi, di\}$. Then, $\mathcal{N}(F \cup \{^+\}) \leq_{\text{unl}}^{\text{bool}} \mathcal{N}(F)$.*

We will not prove this proposition in this thesis, for a proof of see [FGL$^+$12b].

7 Other Boolean Query Modalities

Until now we said that a boolean query q is expressible in $\mathcal{N}(F)$ if there is an expression $e \in \mathcal{N}(F)$ such that for every graph G, $e(G) \neq \emptyset$ if and only if $q(G) = \mathit{true}$. This is just a certain interpretation of boolean queries in our language of navigational queries. Another definition could be: a boolean query q is expressible in $\mathcal{N}(F)$ if there is an expression $e \in \mathcal{N}(F)$ such that for every graph G, $e(G) = \emptyset$ if and only if $q(G) = \mathit{true}$. In this chapter we will explore the expressibility of boolean queries relative to different interpretations of boolean queries in our language of navigational queries. From now on we will refer to these interpretations as boolean query modalities.

To align our results from the previous chapters with this new notion, we will refer to the modality we used for boolean queries in the previous chapters as the $\neq\emptyset$-modality. Furthermore, we will refer to the new modality introduced above as the $=\emptyset$-modality.

The main focus of this chapter will be to characterize other modalities and compare them with the characterization of the $\neq\emptyset$-modality. To this end, we will require a new notation for $\leq^{\mathrm{bool}}$ which facilitates the use of other modalities. We will write $\mathcal{N}(F_1) \leq^{\mathrm{bool}\neq\emptyset} \mathcal{N}(F_2)$ if every boolean query q expressible in $\mathcal{N}(F_1)$ is also expressible in $\mathcal{N}(F_2)$ relative to the $\neq\emptyset$-modality. Similarly, we will write $\mathcal{N}(F_1) \leq^{\mathrm{bool}=\emptyset} \mathcal{N}(F_1)$ if every boolean query q expressible in $\mathcal{N}(F_1)$ is also expressible in $\mathcal{N}(F_2)$ relative to the $=\emptyset$-modality.

It appears that the characterization of $\leq^{\mathrm{bool}=\emptyset}$ coincides with the characterization of $\leq^{\mathrm{bool}\neq\emptyset}$.

Proposition 7.1: *Let F_1 and F_2 be arbitrary features. Then, $\mathcal{N}(F_1) \leq^{\mathrm{bool}=\emptyset} \mathcal{N}(F_2)$ if and only if $\mathcal{N}(F_1) \leq^{\mathrm{bool}\neq\emptyset} \mathcal{N}(F_2)$.*

Proof: Clearly, q is expressible in $\mathcal{N}(F)$ relative to the $=\emptyset$-modality if and only if $\neg q$ is expressible in $\mathcal{N}(F)$ relative to the $\neq\emptyset$-modality. Our proposition now follows directly from this observation. ■

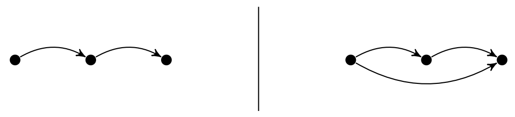


Figure 7.1: Graphs used to show that Proposition 7.3 does not necessarily hold for languages without set difference.

7.1 The $\subseteq$-modality

In this section we will introduce yet another modality and we will try characterize its relative expressiveness in our language of navigational queries.

We say that a boolean query q is expressible in $\mathcal{N}(F)$ relative to the $\subseteq$-modality if there exists two expressions $e_1, e_2 \in \mathcal{N}(F)$ such that for every graph G, $e_1(G) \subseteq e_2(G)$ if and only if $q(G) = \mathit{true}$. Furthermore, we will write $\mathcal{N}(F_1) \leq^{\text{bool}\subseteq} \mathcal{N}(F_2)$ if every boolean query expressible in $\mathcal{N}(F_1)$ relative to the $\subseteq$-modality is also expressible in $\mathcal{N}(F_2)$ relative to the $\subseteq$-modality.

The first result for the $\subseteq$-modality mirrors the result for $\neq\emptyset$-modality.

Proposition 7.2: *Let F_1 and F_2 be sets of nonbasic features. If $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$, then $\mathcal{N}(F_1) \leq^{\text{bool}\subseteq} \mathcal{N}(F_2)$.*

PROOF: Suppose q is a boolean query expressible in $\mathcal{N}(F_1)$ relative to the $\subseteq$-modality. Then there exists two expressions e_1 and e_2 in $\mathcal{N}(F_1)$, such that for every graph G, $q(G) = \mathit{true}$ if and only if $e_1(G) \subseteq e_2(G)$. Furthermore, since $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$, there exists e_1' and e_2' in $\mathcal{N}(F_2)$ such that $e_1' \equiv e_1$ and $e_2' \equiv e_2$. Now, clearly $e_1'(G) \subseteq e_2'(G)$ if and only if $q(G) = \mathit{true}$, and hence q is expressible in $\mathcal{N}(F_2)$ relative to the $\subseteq$-modality as desired. ∎

The following proposition tells us that we can mimic the containment modality in the $=\emptyset$-modality modality if we have set difference.

Proposition 7.3: *Let F be an arbitrary set of nonbasic features such that $\setminus \in F$. If q is a boolean query expressible in $\mathcal{N}(F)$ relative to the $\subseteq$-modality, then q is also expressible in $\mathcal{N}(F)$ relative to the $=\emptyset$-modality.*

PROOF: Let e_1 and e_2 be the expressions in $\mathcal{N}(F)$ such that $e_1(G) \subseteq e_2(G)$ if and only if $q(G) = \mathit{true}$. Now, the query $e_1 \setminus e_2(G) = \emptyset$ clearly expresses q relative to $=\emptyset$-modality. ∎

The graphs displayed in Figure 7.1 are distinguishable by the query $R \subseteq R^2$. They are, however, indistinguishable in $\mathcal{N}$ by the brute-force method introduced in Section 3.1. Hence the previous proposition does not hold when the set difference operator is not present in F.

The converse, however, does hold for every set of nonbasic features.

Proposition 7.4: *Let F be a set of nonbasic features. If q is expressible in $\mathcal{N}(F)$ relative to the $=\emptyset$-modality, then it is also expressible in $\mathcal{N}(F)$ relative to the $\subseteq$-modality.*

Proof: If q is expressible in $\mathcal{N}(F)$ relative to the $=\emptyset$-modality, there exists $e_1 \in \mathcal{N}(F)$ such that $e(G) = \emptyset$ if and only if $q(G) = \mathit{true}$ for every graph G. Clearly $e(G) \subseteq \emptyset(G)$ if and only if $e(G) = \emptyset$, and hence $e \subseteq \emptyset$ expresses q relative to the $\subseteq$-modality as desired.■

As in the previous chapters, we will introduce a notion of strong boolean separation relative to other modalities. We will write $\mathcal{N}(F_1) \not\leq^{\mathrm{bool}\subseteq}_{\mathrm{strong}} \mathcal{N}(F_2)$ if there exists a boolean query q expressible in $\mathcal{N}(F_1)$ relative to $\subseteq$-modality and two graphs G_1 and G_2 such that $q(G_1) = \mathit{true}$ and $q(G_2) = \mathit{false}$ and for every boolean query q' expressible in $\mathcal{N}(F_2)$ relative to $\subseteq$-modality, $q(G_1) = \mathit{true}$ if and only if $q(G_2) = \mathit{true}$. In this case we will also say that $\mathcal{N}(F_1)$ is strongly separable from $\mathcal{N}(F_2)$ at the level of boolean queries relative to the $\subseteq$-modality.

The following proposition mirrors the result for the $\neq\emptyset$-modality.

Proposition 7.5 (Primitivity of $\setminus$): *Let F_1 and F_2 be sets of nonbasic features. If $\setminus \in \overline{F_1}$ and $\setminus \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\mathrm{bool}\subseteq}_{\mathrm{strong}} \mathcal{N}(F_2)$.*

Proof: The graphs displayed in Figure 4.4a are distinguishable in $\mathcal{N}(F_1)$ relative to the $\subseteq$-modality by the query $R^2 \setminus R \subseteq \emptyset$. On the other hand, by the brute-force method introduced in Section 3.1 they are indistinguishable in $\mathcal{N}(F_2)$ relative to the $\neq\emptyset$-modality, hence also in $\mathcal{N}(F_2)$ relative to the $=\emptyset$-modality by Proposition 7.1. Therefore, q is not expressible in $\mathcal{N}(F_2)$ relative to the $=\emptyset$-modality. Proposition 7.5 now implies that q is not expressible in $\mathcal{N}(F_2)$ relative to the $\subseteq$-modality either, which concludes our proof.■

We can, as for the $\neq\emptyset$-modality (see Section 3.1), verify whether two graphs are distinguishable in $\mathcal{N}(F)$ relative to the $\subseteq$-modality by a brute-force method. We can do so as follows. Let F be a set of nonbasic features, let G_1 and G_2 be arbitrary graphs, and let $A = \{(e(G_1), e(G_2) \mid e \in \mathcal{N}(F)\}$. Now, notice that each pair of tuples (X_1, Y_1) and (X_2, Y_2) in A represent a boolean query $e_1 \subseteq e_2$ expressible in $\mathcal{N}(F)$ evaluated on G_1 and G_2, i.e., $X_1 = e_1(G_1)$, $X_2 = e_2(G_1)$, $Y_1 = e_1(G_2)$ and $Y_2 = e_2(G_2)$. Also, for every boolean query $e_1 \subseteq e_2$, it is clear that $(e_1(G_1), e_1(G_2))$ and $(e_2(G_1), e_2(G_2))$ are present in A. Hence we can use A to decide whether two graphs are distinguishable in $\mathcal{N}(F)$ relative to the $\subseteq$-modality or not (see Algorithm 7.1).

Algorithm 7.1 Brute-Force Algorithm for the $\subseteq$-modality

```
1: procedure Brute-Force-⊆-modality($G_1, G_2, \mathcal{N}(F)$)
2:     $A \leftarrow \{(e(G_1), e(G_2)) \mid e \in \mathcal{N}(F)\}$
3:     for all $(R_1, R_2), (S_1, S_2) \in B$ do
4:         if $(R_1 \subseteq S_1$ and $R_2 \not\subseteq S_2)$ or $(R_1 \not\subseteq S_1$ and $R_2 \subseteq S_2)$ then
5:             return Distinguishable
6:     return Indistinguishable
```

Using Algorithm 7.1 we can prove the following proposition which mirrors the result for the $\neq\emptyset$-modality.

Proposition 7.6 (Primitivity of *di*): *Let F_1 and F_2 be sets of nonbasic features. If $di \in \overline{F_1}$ and $di \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}\subseteq}_{\text{strong}} \mathcal{N}(F_2)$.*

PROOF: The graphs displayed in Figure 4.1a are distinguishable in $\mathcal{N}(F_1)$ relative to the $\subseteq$-modality by the query $di \subseteq \emptyset$. We can, however, verify that they are indistinguishable in $\mathcal{N}(F_2)$ relative to the $\subseteq$-modality with Algorithm 7.1. ■

It appears that π certainly adds expressive power when $F_2 \subseteq \{\backslash, \cap, {}^+\}$.

Proposition 7.7 (Primitivity of π): *Let F_1 and F_2 be sets of nonbasic features. If $\pi \in \overline{F_1}$ and $F_2 \subseteq \{\backslash, \cap, {}^+\}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

PROOF: The brute-force algorithm tells us that the graphs in Figure 4.4b are indistinguishable in $\mathcal{N}(F_2)$ relative to the $\subseteq$-modality. They are however distinguishable in $\mathcal{N}(F_1)$ by $\pi_1(R^2) \circ R \circ \pi(R^2) \subseteq \emptyset$. ■

It is still open whether the other results involving π still hold.

We will now try to prove that adding transitive closure to a language adds expressive power. To this end, we first need that every boolean query expressible in a language without transitive closure relative to $\subseteq$-modality is also expressible in first order logic. This should not come as a surprise since containment is expressible in first order logic.

Proposition 7.8: *Let F be a set of nonbasic features such that ${}^+ \notin \overline{F}$. Then every boolean query expressible in $\mathcal{N}(F)$ relative to the $\subseteq$-modality is also expressible in first order logic.*

PROOF: Let q be expressible in $\mathcal{N}(F_1)$ relative to the $\subseteq$-modality. Then there exists e_1 and e_2 in $\mathcal{N}(F_1)$ such that $e_1(G) \subseteq e_2(G)$ if and only if $q(G) = \mathit{true}$. By Proposition 2.6 there exists $\varphi_1(x, y)$ and $\varphi_2(x, y)$ such that $(a, b) \in e_1(G)$ if and only if $G \models \varphi_1[a, b]$, and $(a', b') \in e_2(G)$ if and only if $G \models \varphi_2[a', b']$. Now, define $\psi := \forall x, y : \varphi_1(x, y) \to \varphi_2(x, y)$. Clearly $G \models \psi$ if and only if $e_1(G) \subseteq e_2(G)$ if and only if $q(G) = \mathit{true}$. Hence q is expressible in first order logic as desired. ■

We are now ready to prove that transitive closure adds expressive power.

Proposition 7.9 (Primitivity of ${}^+$): *Let F_1 and F_2 be sets of nonbasic features. If ${}^+ \in \overline{F_1}$ and ${}^+ \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}\subseteq} \mathcal{N}(F_2)$.*

PROOF: The query $S \circ R \circ T = \emptyset$ is expressible in $\mathcal{N}(F_1)$ relative to the $\subseteq$-modality by the query $S \circ R \circ T \subseteq \emptyset$. Now, suppose for the sake of contradiction that $S \circ R \circ T \subseteq \emptyset$ is expressible in $\mathcal{N}(F_2)$ relative to the $\subseteq$-modality. Then by Proposition 7.8 this query is also expressible in first order logic and hence its negation $S \circ R \circ T \neq \emptyset$ as well. This query, however, is not expressible in first order logic by Proposition 2.6 and hence contradicts our assumption. ■

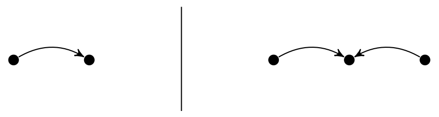


Figure 7.2: Graphs used to show that Proposition 7.3 does not hold for languages without set different.

Notice that the proof of the proposition above requires two edge labels. Hence the same question arises as for $\neq\emptyset$-modality: does this proposition still hold when we only consider unlabeled input graphs. We will explore this question in Section 7.2.

Remember that a certain collapse around $^{-1}$ occurs relative to the $\neq\emptyset$-modality. The following proposition tells us that this collapse no longer occurs for certain languages, e.g., $\mathcal{N}(^{-1}) \not\leq^{\text{bool}\subseteq} \mathcal{N}(\pi)$.

Proposition 7.10: *Let F_1 and F_2 be nonbasic features. If $^{-1} \in F_1$ and $F_2 \subseteq \{\cap, \setminus, \pi, \overline{\pi}, ^{+}\}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}\subseteq}_{\text{strong}} \mathcal{N}(F_2)$.*

Proof: The graphs displayed in Figure 7.2 are indistinguishable in $\mathcal{N}(\cap, \setminus, \pi, \overline{\pi}, ^{+})$ by the brute-force method introduced in Section 3.1. They are, however, distinguishable in $\mathcal{N}(^{-1})$ by the query $R \circ R^{-1} \subseteq id$. ■

It is still open whether we can extend the previous proposition to include diversity in $\overline{F_2}$. Notice that the previous proposition also implies the following important result.

Proposition 7.11: *$\leq^{\text{bool}\subseteq}$ and $\leq^{\text{bool}\neq\emptyset}$ do not coincide everywhere. More formally, there exists sets of nonbasic features F_1 and F_2 such that $\mathcal{N}(F_1) \not\leq^{\text{bool}\subseteq} \mathcal{N}(F_2)$.*

It is still open whether the separation results discussed in Chapter 5 involving $\overline{\pi}$ and $\cap$ still hold.

7.2 Unlabeled graphs

In the previous section we mentioned that the proof of Proposition 7.9 required 2 edge labels. Hence it is natural to ask which results still hold when we only consider unlabeled input graphs. Notice that all the other results in the previous section only require one edge label. Therefore they still hold when we only consider unlabeled input graphs.

It is still open whether Proposition 7.9 still holds in its full generality when we only consider unlabeled input graphs. We can, however, prove some less general results. The first result requires that $\cap$ is also present in $\overline{F_1}$.

Proposition 7.12: *Let F_1 and F_2 be sets of nonbasic features. If $^{+} \in \overline{F_1}, \cap \in \overline{F_1}$ and $^{+} \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}\subseteq} \mathcal{N}(F_2)$.*

Proof: Notice that the negation of the 'cycle' query is expressible in $\mathcal{N}(F_1)$ relative to the $\subseteq$-modality by the query $R^{+} \cap id \subseteq \emptyset$. On the other hand, the negation of the 'cycle' query is not expressible in first order logic since the 'cycle' query is not expressible in first order logic due to Proposition 6.4. Hence by Proposition 7.8 it is not expressible in $\mathcal{N}(F_2)$ relative to the $\subseteq$-modality either. ■

The following separation result involving transitive closure requires that $^{-1}$ is present in $\overline{F_1}$.

Proposition 7.13: *Let F_1 and F_2 be sets of nonbasic features. If $^+ \in \overline{F_1}$, $^{-1} \in \overline{F_1}$ and $^+ \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leq^{\text{bool}\subseteq} \mathcal{N}(F_2)$.*

PROOF: Proposition 6.12 tells us that the boolean query $R^2 \circ (R \circ R^{-1})^+ \circ R^2 \neq \emptyset \in \mathcal{N}(F_1)$ is not expressible in first order logic. Therefore, $R^2 \circ (R \circ R^{-1})^+ \circ R^2 \subseteq \emptyset$ is not expressible in first order logic. Hence by Proposition 7.8 it is not expressible in $\mathcal{N}(F_2)$ relative to the $\subseteq$-modality either. ■

The last separation result involving transitive closure requires that $\overline{\pi}$ is present in $\overline{F_1}$.

Proposition 7.14: *Let F be a set of nonbasic features such that $\overline{\pi} \in \overline{F}$ and $^+ \notin \overline{F}$. Then, $\mathcal{N}(F \cup \{^+\}) \not\leq^{\text{bool}}_{\text{unl}} \mathcal{N}(F)$.*

PROOF: The boolean query "there is a non-sink node from which no sink node can be reached" is expressible in $\mathcal{N}(\overline{\pi}, ^+)$ by Lemma 6.14. Hence its negation is expressible in $\mathcal{N}(\overline{\pi}, ^+)$ relative to the $\subseteq$-modality due to Proposition 7.4, and thus also in $\mathcal{N}(F \cup \{^+\})$ relative to the $\subseteq$-modality. This query, however, is not expressible in first order logic by Lemma 6.15 and hence by Proposition 7.8 not in $\mathcal{N}(F)$ relative to the $\subseteq$-modality either. ■

It is still open whether the other separation results discussed in Chapter 6 involving transitive closure still hold.

8 Conclusion

In this thesis we studied navigational query languages which contain the basic features identity (id), composition ($\circ$) and union ($\cup$), and a selection of nonbasic features: diversity (di), converse (e^{-1}), intersection ($e_1 \cap e_2$), difference ($e_1 \setminus e_2$), projections (π_1 and π_2), coprojections ($\overline{\pi}_1$ and $\overline{\pi}_2$) and transitive closure (e^+). Furthermore, we introduced path queries over these operators which map graphs to graphs and boolean queries which map graphs to *true* or *false*.

In Chapter 4 we completely characterized the relative expressiveness of our query languages at the level of path queries. It appears that this relative expressiveness has an interesting structure: every path query in $\mathcal{N}(F_1)$ is expressible $\mathcal{N}(F_2)$ if and only if every feature in F_1 is expressible in F_2.

On the other hand, in Chapter 5 we completely characterized the relative expressiveness of our query languages at the level of boolean queries. We have proven that the relative expressiveness at the level path and boolean queries do not coincide. Most notably, at the level of boolean queries $^{-1}$ does not always add expressive power, while $^{-1}$ always adds expressive power on the level of path queries.

During the characterization of the relative expressiveness at the level of boolean queries we noticed that the proof of the proposition which established that $^+$ always adds more expressive power requires more than one edge label. Hence we wondered whether the relative expressiveness at the level of boolean queries changes when we only allow unlabeled input graphs. In Chapter 6 we showed that a certain collapse in expressiveness occurs. Most notably, $^+$ does not add expressive power when there are no other nonbasic features present in the language.

To express boolean queries in our navigational query languages we associated a nonempty query result with *true* and an empty query result with *false* ($\neq\emptyset$-modality). This is the standard method to express boolean queries in database theory. We could, however, express boolean queries in our language in a different manner. In Chapter 7 we introduced two new methods to express boolean queries (also called modalities). Most notably, we linked a boolean query q to every pair of path queries e_1, e_2 as follows: $q(G) :=$ *true* if and only if $e_1(G) \subseteq e_2(G)$ ($\subseteq$-modality). During the characterization of this new modality we discovered that the collapse surrounding $^{-1}$ no longer occurs when

di is not present. It is still open whether this collapse also disappears in the presence of di. Furthermore, it is also still open whether the separation results discussed in Chapter 5 involving $\pi, \overline{\pi}$ and $\cap$ still hold.

During the characterization of the $\subseteq$-modality we again noticed that the proof of the proposition which established that $^+$ always adds more expressive power requires more than one edge label. Hence here we also wondered whether the relative expressiveness at the level of boolean queries changes when we only allow unlabeled input graphs. In Section 7.2 we started this characterization. It is unfortunately still open whether the collapse surrounding $^+$ still occurs. We did, however, establish some results which mirror the results for the $\neq\emptyset$-modality.

A Additional Material

Lemma A.1: *There are 157 pairs* (F_1, F_2) *such that* $\mathcal{N}(F_1), \mathcal{N}(F_2) \in \mathcal{C}$ *and such that* $F_1 \not\subseteq \overline{F_2}$.

PROOF: We will split our proof into several exclusive cases. First, notice that in $\mathcal{C}$, we have $\overline{F_2} = F_2 \cup \{\pi \mid \overline{\pi} \in F_2\}$.

Suppose that $\overline{\pi} \in F_2$, then $F_1 \not\subseteq \overline{F_2}$ if and only if $F_1 \cap \{di, {}^{-1}\} \not\subseteq F_2 \cap \{di, {}^{-1}\}$. Furthermore, suppose that $F_2 \cap \{di, {}^{-1}\} = \emptyset$. Then to have a valid pair, di or ${}^{-1}$ has to be in F_1, which gives us 3 possible choices for $F_1 \cap \{di, {}^{-1}\}$. Each of these choices can be combined with $\emptyset, \{\pi\}, \{\overline{\pi}\}$ or $\{\pi, \overline{\pi}\}$. Moreover, π can be in F_2 or not. Hence we have $3 \cdot 4 \cdot 2 = 24$ pairs. On the other hand, suppose that $|F_2 \cap \{di, {}^{-1}\}| = 1$. To construct a valid pair, $F_1 \cap \{di, {}^{-1}\}$ has to contain the operation not present in $F_2 \cap \{di, {}^{-1}\}$, hence we have 2 possible choices for $F_1 \cap \{di, {}^{-1}\}$. Again, these choices can be combined with $\emptyset, \{\pi\}, \{\overline{\pi}\}$ or $\{\pi, \overline{\pi}\}$ and π can be in F_2 or not. Furthermore, $F_2 \cap \{di, {}^{-1}\}$ can be either di or ${}^{-1}$. Therefore, we have $2 \cdot 4 \cdot 2 = 32$ pairs.

Now, assume that $\overline{\pi} \in F_1$ and $\overline{\pi} \notin F_2$. Since we already have one element in F_1 not in $\overline{F_2}$, the other elements of F_1 and F_2 can be arbitrary. Hence we have $2^3 \cdot 2^3 = 64$ pairs.

Suppose that $\overline{\pi} \notin F_1$ and $\overline{\pi} \notin F_2$. Now, $|F_2| \neq 3$, since $F_1, F_2 \subseteq \{di, \pi, {}^{-1}\}$. If $|F_2| = 2$, then there are 3 possible choices for F_2. Because $|F_2| = 2$, it lacks one feature x in $\{di, \pi, {}^{-1}\}$. Therefore, any F_1 which leads to a valid pair with F_2, has to contain x, there are 4 such F_1's. Hence we have $3 \cdot 4 = 12$ pairs.

Now, suppose that $|F_2| = 1$, then there are 3 possibilities for F_2. Any set having more than one element is a valid pair with F_2. Furthermore, any set with one element except one is a valid pair with F_2. Also, F_1 cannot be empty, hence we have $(2^3 - 2) \cdot 3 = 18$ pairs. Finally, if $F_2 = \emptyset$, any nonempty F_1 will do. Hence we have $2^3 - 1 = 7$ pairs.

Adding all the subtotals, we get $24 + 32 + 64 + 12 + 18 + 7 = 157$ pairs. ■

B Implementation

Throughout this thesis most of our results relied heavily on three algorithms: Algorithm 3.1, Algorithm 7.1 and the algorithm outlined in the proof of Theorem 5.20. I implemented these algorithms in java and I choose to implemented Algorithm 3.1 and Algorithm 7.1 in an object oriented fashion so the algorithms can be extended to include for example other nonbasic features like the residuation operator introduced in [FGL⁺12a]. We will devote this chapter to how I implemented Algorithm 3.1 and Algorithm 7.1.

Notice that Algorithm 3.1 and Algorithm 7.1 only differ in the distinguishability check. This distinguishability check depends only on the given boolean query modality. Hence why I chose the strategy pattern to accommodate different boolean query modalities (see Figure B.1). The Modality class plays strategy role. It contains two methods: elementSeparation and postSeparation. The elementSeparation method accommodates separation tests on individual elements in $A = \{e(G_1), e(G_2) \mid e \in \mathcal{N}(F)\}$, while the postSeparation method accommodates separation tests on the whole of A.

To accommodate a wide variety of nonbasic features I also used a strategy like

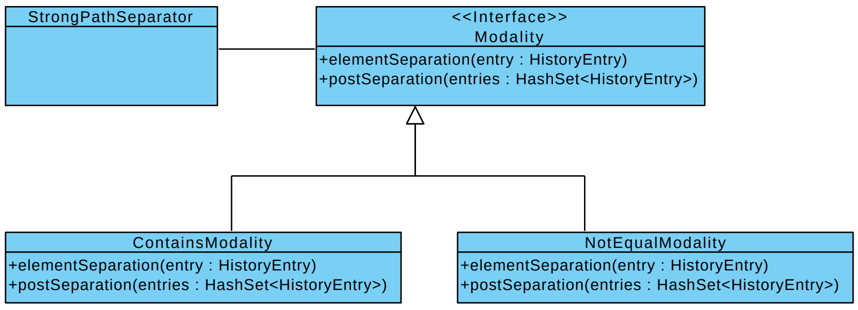

StrongPathSeparator
<<Interface>>
Modality
+elementSeparation(entry : HistoryEntry)
+postSeparation(entries : HashSet<HistoryEntry>)
ContainsModality
+elementSeparation(entry : HistoryEntry)
+postSeparation(entries : HashSet<HistoryEntry>)
NotEqualModality
+elementSeparation(entry : HistoryEntry)
+postSeparation(entries : HashSet<HistoryEntry>)

Figure B.1: Strategy pattern to accommodate different modalities.

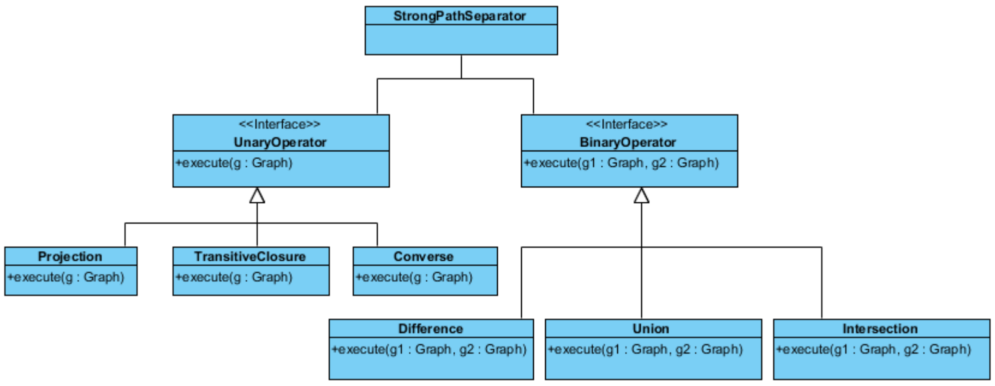


Figure B.2: Strategy like pattern to accommodate different nonbasic features.

pattern. To this end I introduced two interfaces: UnaryOperator and BinaryOperator (see Figure B.2). Either interface contains an execute method which represents the action an operator has to carry out given the input graphs. To compute A, we first initialize a set of unary and binary operators and then call the execute method of their base class repeatedly until now change occurs. This method, however, does not work for coprojection operator, since the calculation of $\overline{\pi}(e_1)(G)$ does not only depend on the $e_1(G)$ but also on G. Hence unfortunately one extra test had to be included for the coprojection operator.

C
Meeting Reports

Date: 01/08/2012

Attendees: Professor Jan Van den Bussche

Between this meeting and our last meeting I implemented the brute-force algorithm described in Section 3.1 in java. It accepts command line parameters for language and graph selections. I also finished the proof of a proposition which is needed to establish separation on the level of path queries for languages in $\mathcal{C}$. The proof utilizes the so called homomorphism approach which we discusses in our last meeting.

In this meeting I demonstrated the algorithm to my advisor. He also revised the proof mentioned above and explained me how the counting argument works to establish separation on the level of path queries. I also asked the following question which was deemed trivial by the paper I am studying: if $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$, then why does $\mathcal{N}(F_1 \cup \{^+\}) \leq^{\text{path}} \mathcal{N}(F_2 \cup \{^+\})$. My advisor explained me how to tackle this problem.

My planning: I will try to start the actual characterization proof of path for languages in the same classes and hopefully also for languages in different classes.

Date: 28/09/2012

Attendees: Professor Jan Van den Bussche

In our last meeting I asked the following question, which was deemed trivial by the paper I am studying: if $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$, then why does $\mathcal{N}(F_1 \cup \{^+\}) \leq^{\text{path}} \mathcal{N}(F_2 \cup \{^+\})$. In this meeting we reviewed the proof I wrote up for this meeting associated to this question. I also provided an alternative proof (a case analysis) for the actual path separation for every class of languages. I found this proof while trying to understand the counting argument used in the paper.

My planning: I will try to understand the collapse of $^{-1}$ for bool. I will also try to establish characterization of bool for languages in $\mathcal{C}$.

Date: 25/10/2012

Attendees: Professor Jan Van den Bussche

Since our last meeting I started with the characterization of bool. First I showed that a certain collapse occurs around $^{-1}$. However, my advisor pointed out that there was a flaw in my proof. The flaw arose because I thought that I could remove the pre-order and post-order assumption in his proof. I also proved the characterization theorem of bool for languages in the $\mathcal{C}$ class.

My planning: I will try to finish the characterization of bool for languages in $\mathcal{C}[\cap]$ by our next meeting.

Date: 7/01/2013

Attendees: Professor Jan Van den Bussche, Jelle Hellings and Robert Brijder

First, I had to give a mid term presentation of my thesis for my advisor and assessors. In my opinion it went well, although I forgot to include the difference between strong separation versus normal separation in the presentation. My advisor noticed this and asked me to explain it, which lead to an interesting discussion between me and the other attendees. This discussion made it crystal clear again why this distinction was so important.

The assessors left when my presentation was over. We then proceeded with a normal advisor-student meeting. Since our last meeting I read a part of another paper: 'Similarity and bisimilarity notions appropriate for characterizing indistinguishability in fragments of the calculus of relations'. This so I could show that two particular marked graphs are bisimilar up to every level in polynomial time. I also adapted the theory described in that paper to the language I required, as in the paper they consider another fragment of the calculus of relations. I also implemented this polynomial time algorithm. Moreover, I finished the proof of $\leq^{\text{bool}}$ characterization for languages in $\mathcal{C}[\cap]$.

As in previous meetings, I always typeset and finalize my master thesis text before starting with new concepts, hence why my writing work is not lacking behind. The previous meeting, however, my advisor pointed out to me that I should write more 'binding text', which I did by this meeting.

My planning: Finish the characterization of $\leq^{\text{bool}}$ and $\leq^{\text{bool}}_{\text{unl}}$ by the following meeting and after that start with some research of my own on other modalities for boolean queries.

Date: 25/02/2013

Attendees: Professor Jan Van den Bussche
Since the previous meeting I finished the characterization of $\leq^{\text{bool}}$ apart from one proof on which I am still stuck. This proof requires a complicated kind of pebble game to establish separation. I discussed this problem with my advisor and we tried to find a tactic for the game to finish to proof.

A few weeks back I visited my advisor for a couple of small questions since I could not directly find an EhrenfeuchtFraïssé game. My advisor then told me about another technique one can use to establish inexpressibility in first order logic, called the Hanf-locality. Since this mini-meeting I picked up the book [Lib04]. With this new technique I could finish the characterization of $\leq^{\text{bool}}_{\text{unl}}$.

In this meeting my advisor also proofread the last additions to my text. He pointed out that there was a mistake in one of my proofs, we could, however, patch the proof up right away. He also pointed out that the use of 'we have' or 'we have that' can usually be omitted in the english writing.

In another short visit to my advisor before this meeting I pointed out that there was a flaw in the characterization theorem of $\leq^{\text{bool}}$ in [FGL+11]. By this meeting I found a new theorem for which I had to introduce some new notation. The problem with the old notation (and theorem) was that it did not support the transitivity problem on the left hand side. I introduced some new notation to capture this problem and patched up the characterization theorem of $\leq^{\text{bool}}$ with this new notation.

My planning: Finish up that one last proof for the characterization of $\leq^{\text{bool}}$. This will finish up the work I am doing involving [FGL+11]. I will also try to rewrite my text to omit the overuse of 'we have' and 'we have that'. Furthermore, I will now start with the characterization of $\leq^{\text{bool}}$ for other modalities.

Date: 28/04/2013

Attendees: Professor Jan Van den Bussche
Since the previous meeting I tried to prove Proposition 5.25. Unfortunately, however, I could not finish the bisimilarity game. On the other hand, I started working on the characterization of $\leq^{\text{bool}\subseteq}$ and already found some small results.

In this meeting we found a strategy to prove Proposition 5.25.

My planning: Try to prove Proposition 5.25 with the strategy we found in this meeting. I will also try to find some more results involving $\leq^{\text{bool}\subseteq}$.

Date: 02/05/2013

Attendees: Professor Jan Van den Bussche
Since the previous meeting I found a proof for Proposition 5.25.

In this meeting my advisor proof read Proposition 5.25. There were some gaps in my proofs and my advisor found my proof very hard to follow. He asked whether I could maybe come up with a more 'high level' proof. He did say that I should first try to find some more results for $\leq^{\text{bool}\subseteq}$.

My planning: Try to prove Proposition 5.25 with the strategy we found in this meeting. I will also try to find some more results involving $\leq^{\text{bool}\subseteq}$.

D
Dutch Summery

Door de jaren heen zijn er tal van query talen geïntroduceerd en grondig bestudeerd. Enkele van deze query talen hebben we zorgvuldig bestudeerd in onze bachelor en master opleiding, bijvoorbeeld relationele algebra en XPATH. In deze thesis zullen we navigationele query talen bestuderen op gelabelde grafen. Deze talen zijn relevant voor de praktijk omdat graaf data terug gevonden kan worden in RDF en overal op het web.

In onze navigationale query talen zullen we een selectie van unaire en binary operatoren beschouwen die grafen op grafen afbeelden. De unaire operatoren zijn projectie (π), coprojectie ($\overline{\pi}$), inverteren ($^{-1}$), identiteit (id), diversiteit (di), transitieve sluiting($^+$) en een operatie om bogen met een bepaalt label terug te krijgen. Anderzijds, de binaire operatoren zijn unie ($\cup$), doorsnede ($\cap$), compositie ($\circ$) en verschil ($\setminus$). Deze operatoren zijn gedefinieerd als volgt:

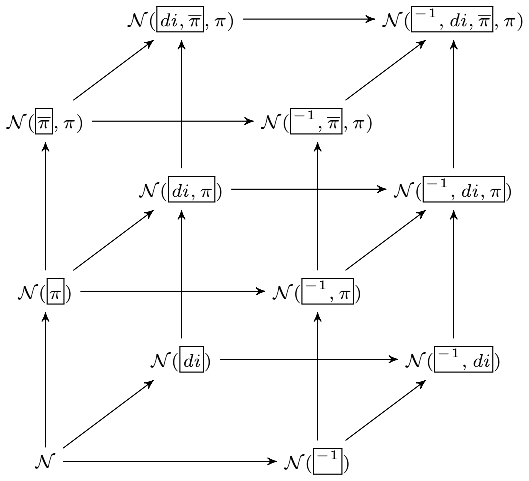
$$\begin{aligned}
R(G) &= G(R); && \text{($G(R)$ zijn de bogen met label R)}\\
\emptyset(G) &= \emptyset;\\
id(G) &= \{(m,m) \mid m \in \text{adom}(G)\};\\
e_1 \circ e_2(G) &= \{(m,n) \mid \exists p : (m,p) \in e_1(G) \wedge (p,n) \in e_2(G)\};\\
e_1 \cup e_2(G) &= e_1(G) \cup e_2(G);\\
di(G) &= \{(m,n) \mid m \neq n \wedge m,n \in \text{adom}(G)\};\\
e^{-1}(G) &= \{(m,n) \mid (n,m) \in e(G)\};\\
e_1 \cap e_2(G) &= e_1(G) \cap e_2(G);\\
e_1 \setminus e_2(G) &= e_1(G) \setminus e_2(G);\\
\pi_1(e)(G) &= \{(m,m) \mid m \in \text{adom}(G) \wedge (\exists n)(m,n) \in e(G)\};\\
\pi_2(e)(G) &= \{(m,m) \mid m \in \text{adom}(G) \wedge (\exists n)(n,m) \in e(G)\};\\
\overline{\pi}_1(e)(G) &= \{(m,m) \mid m \in \text{adom}(G) \wedge \neg(\exists n)(m,n) \in e(G)\};\\
\overline{\pi}_2(e)(G) &= \{(m,m) \mid m \in \text{adom}(G) \wedge \neg(\exists n)(n,m) \in e(G)\};\\
e^+(G) &= \bigcup_{k \geq 1} e^k(G);; && \text{(waarbij $e^k(G) = \underbrace{e \circ \ldots \circ e}_{k \text{ times}}(G)$)}
\end{aligned}$$

Merk op dat deze operatoren inwerken op het actief domein[1] van de input graaf. We doen dit omdat we enkel bogen in onze query resultaten willen hebben indien er naartoe genavigeerd kan worden. Net zoals in de relationele algebra zijn queries in onze talen recursief opgebouwd over bovenstaande operatoren (zie Voorbeeld D.1). Omdat de bovenstaande operatoren grafen op grafen afbeelden, beelden deze queries ook grafen op grafen af. Vanaf nu zullen we queries die grafen op grafen afbeelden *pad* queries noemen.

Voorbeeld D.1: De pad query $\overline{\pi}_1((R \circ R) \cup (R \circ R \circ R))$ selecteert alle paren (x, x) zodat er geen pad is van lengte twee of drie startende in x en de pad query $R^+ \cap id$ selecteert een paar (x, x) als x zich in een cykel bevindt. $\diamond$

We noemen een selectie van operatoren een taal. De kleinste taal die we zullen beschouwen bevat $\emptyset, id, \circ$ en $\cup$. De verzameling van queries over deze taal noteren we als $\mathcal{N}$. We kunnen aan deze basis taal een verzameling van operatoren F toevoegen, de verzameling queries over deze taal noteren we als $\mathcal{N}(F)$, bijvoorbeeld $\mathcal{N}(\pi)$ bevat alle queries over $\emptyset, id, \circ, \pi$ en $\cup$.

Een interessante vraag voor onze nieuwe query talen is hoe twee talen gerelateerd zijn in termen van expressieve kracht[2]. We willen bijvoorbeeld weten of het toevoegen van *di* aan een bepaalde taal ons toelaat om meer pad queries uit te drukken. Deze vraag hebben we beantwoord in Hoofdstuk 4. Het blijkt dat er een interessant patroon zit achter de relatieve expressiviteit op het niveau van pad queries: alle pad queries in $\mathcal{N}(F_1)$ zijn uitdrukbaar in $\mathcal{N}(F_2)$ als en slechts als de operatoren in F_1 uitdrukbaar zijn aan de hand van de operatoren in F_2. We zullen vanaf nu $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$ schrijven als alle pad queries in $\mathcal{N}(F_1)$ ook uitdrukbaar zijn in $\mathcal{N}(F_2)$. Zie bijvoorbeeld Figure D.1 voor de volledige karakterisatie van $\leq^{\text{path}}$ indien de talen geen doorsnede, verschil en transitieve sluiting bevatten. In deze figuur duiden de omkaderde operatoren de kleinste verzameling van operatoren aan waarvan de overblijvende operatoren afgeleid kunnen worden.


$$\begin{tikzcd}
 & \mathcal{N}(\boxed{di, \overline{\pi}}, \pi) \arrow[rr] & & \mathcal{N}(\boxed{^{-1}, di, \overline{\pi}}, \pi) \\
\mathcal{N}(\boxed{\overline{\pi}}, \pi) \arrow[rr] \arrow[ur] & & \mathcal{N}(\boxed{^{-1}, \overline{\pi}}, \pi) \arrow[ur] & \\
 & \mathcal{N}(\boxed{di, \pi}) \arrow[rr] \arrow[uu] & & \mathcal{N}(\boxed{^{-1}, di, \pi}) \arrow[uu] \\
\mathcal{N}(\boxed{\pi}) \arrow[rr] \arrow[ur] \arrow[uu] & & \mathcal{N}(\boxed{^{-1}, \pi}) \arrow[ur] \arrow[uu] & \\
 & \mathcal{N}(\boxed{di}) \arrow[rr] \arrow[uu] & & \mathcal{N}(\boxed{^{-1}, di}) \arrow[uu] \\
\mathcal{N} \arrow[rr] \arrow[ur] \arrow[uu] & & \mathcal{N}(\boxed{^{-1}}) \arrow[ur] \arrow[uu] &
\end{tikzcd}$$

Figuur D.1: Het Hasse diagram van $\leq^{\text{path}}$ for languages F als $\setminus, \cap, ^+ \notin F$. Er is een pad van $\mathcal{N}(F_1)$ naar $\mathcal{N}(F_2)$ als en slechts als $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$.

[1]Het actief domein van een graaf bevat alle knopen die voorkomen in een boog relatie.
[2]We noemen dit ook de relatieve expressiviteit van pad queries.

Voorbeeld D.2: $\mathcal{N}(\pi_1, \pi_2) \leq^{\text{path}} \mathcal{N}(\overline{\pi}_1, \overline{\pi}_2)$ omdat $\pi_i = \overline{\pi}_i(\overline{\pi}_i)$. Anderzijds $\mathcal{N}(^+) \not\leq^{\text{path}} \mathcal{N}$ omdat $^+$ niet uitdrukbaar is aan de hand van $\circ$, *id* en $\cup$. $\diamond$

We kunnen ook booleaanse queries uitdrukken in onze nieuwe query talen door een niet leeg query resultaat te associëren met *true* en een leeg query resultaat te associëren met *false* (zie Voorbeeld D.3).

Voorbeeld D.3: De booleaanse query $id \setminus (\pi_2((id \setminus R^+) \circ (di \cup id))) \neq \emptyset$ drukt graaf connectiviteit uit en de booleaanse query $R^+ \cap id \neq \emptyset$ drukt de cykel query uit. $\diamond$

Voor deze booleaanse queries kunnen we dezelfde vraag stellen als voor pad queries. We willen bijvoorbeeld weten of het toevoegen van $\overline{\pi}$ aan een bepaalde taal ons toelaat om meer booleaanse queries uit te drukken. Deze vraag hebben we beantwoord in Hoofdstuk 5. Het blijkt dat er voor boolean queries ook een interessant patroon zit achter de relatieve expressiviteit op het niveau booleaanse queries. Deze is echter te moeilijk om intuitief te formuleren omdat $^{-1}$ in tegen stelling tot bij pad queries niet altijd expressieve kracht toevoegd aan talen die nog geen $^{-1}$ bevatten (zie Voorbeeld D.4 en Figure D.2). We zullen vanaf nu $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$ schrijven als alle booleaanse queries in $\mathcal{N}(F_1)$ ook uitdrukbaar zijn in $\mathcal{N}(F_2)$.

Voorbeeld D.4: $\mathcal{N}(^{-1}) \leq^{\text{bool}} \mathcal{N}(\pi_1, \pi_2)$ maar $\mathcal{N}(^{-1}) \not\leq^{\text{path}} \mathcal{N}(\pi_1, \pi_2)$. Anderzijds $\mathcal{N}(^{-1}) \not\leq^{\text{bool}} \mathcal{N}$. $\diamond$

Als een pad of booleaanse query niet uitdrukbaar is in een taal, dan is het mogelijk dat deze query wel uitdrukbaar is in die taal indien we enkel ongelabelde input grafen toelaten. Hierdoor is het mogelijk dat de expressieve kracht van talen veranderen indien we enkel ongelabelde input grafen toelaten en dus ook de relatieve expressiviteit. In Hoofdstuk 6 hebben we aangetoond dat de relatieve expressiviteit op het niveau van pad queries niet veranderd, maar dat de relative expressiviteit op het niveau van booleaanse queries wel veranderd (zie Voorbeeld D.5).

Voorbeeld D.5: $\mathcal{N}(^+) \not\leq^{\text{bool}} \mathcal{N}$ maar elke booleaanse query op ongelabelde grafen in $\mathcal{N}(^+)$ is uitdrukbaar in $\mathcal{N}$. $\diamond$

Tot nu toe hebben we booleaanse queries uitgedrukt in onze talen door een niet leeg query resultaat te associëren met *true* en een leeg query resultaat te associëren met *false* ($\neq\emptyset$-modaliteit). Dit is de standaard methode voor uitdrukken van

$$\begin{tikzcd}
 & \mathcal{N}(\boxed{^{-1}, di, \overline{\pi}}, \pi) = \mathcal{N}(\boxed{di, \overline{\pi}}, \pi) \\
\mathcal{N}(\boxed{^{-1}, \overline{\pi}}, \pi) = \mathcal{N}(\boxed{\overline{\pi}}, \pi) \arrow[ur] & \\
 & \mathcal{N}(\boxed{^{-1}, di, \pi}) = \mathcal{N}(\boxed{di, \pi}) \arrow[uu] \\
\mathcal{N}(\boxed{^{-1}, \pi}) = \mathcal{N}(\boxed{\pi}) \arrow[uu] \arrow[ur] & \\
 & \mathcal{N}(\boxed{^{-1}, di}) \arrow[uu] \\
\mathcal{N}(\boxed{^{-1}}) \arrow[uu] \arrow[ur] & \\
 & \mathcal{N}(\boxed{di}) \arrow[uu] \\
\mathcal{N} \arrow[uu] \arrow[ur] &
\end{tikzcd}$$

Figuur D.2: Het Hasse diagram van $\leq^{\text{bool}}$ voor talen F als $\setminus, \cap, ^+ \notin F$. Er is een pad van $\mathcal{N}(F_1)$ naar $\mathcal{N}(F_2)$ als en slechts als $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$.

booleaanse queries. We kunnen booleaanse queries ook op andere manieren uitdrukken in onze query talen. We kunnen bijvoorbeeld de rollen *true* en *false* omdraaien, i.e., we associëren een niet leeg query resultaat met *false* en een leeg query resultaat met *true* ($=\emptyset$-modaliteit). Wij zouden ook bijvoorbeeld een booleaanse query q kunnen associëren aan elk paar *pad* queries e_1, e_2 als volgt: definieer $q(G) = \mathit{true}$ als en slechts als $e_1(G) \subseteq e_2(G)$ ($\subseteq$-modaliteit) (zie Voorbeeld D.6). We noemen deze alternatieve methoden voor het uitdrukken van booleaanse queries modaliteiten.

Voorbeeld D.6: De booleaanse query gekoppeld aan $R^+ \subseteq R$ is waar als en slechts als de input graaf transitief gesloten is. $\diamond$

In Hoofdstuk 7 hebben we aangetoond dat de relatieve expressiviteit voor de $=\emptyset$-modaliteit samenvalt met de relatieve expressiviteit voor de $\neq\emptyset$-modaliteit, maar de relatieve expressiviteit voor de $\subseteq$-modaliteit valt niet samen met de relatieve expressiviteit voor de $\neq\emptyset$-modaliteit (zie Voorbeeld D.7). We zullen $\leq^{\text{bool}\neq\emptyset}$ schrijven voor de relatieve expressiviteit van de $\neq\emptyset$-modaliteit en $\leq^{\text{bool}\subseteq}$ voor $\subseteq$-modaliteit.

Voorbeeld D.7: $\mathcal{N}(^{-1}) \leq^{\text{bool}\neq\emptyset} \mathcal{N}(\pi)$ maar $\mathcal{N}(^{-1}) \not\leq^{\text{bool}\subseteq} \mathcal{N}(\pi)$. $\diamond$

Er zijn nog tal van open vragen voor de $\subseteq$-modaliteit. We weten bijvoorbeeld niet of $\mathcal{N}(^{-1}) \leq^{\text{bool}\subseteq} \mathcal{N}(di)$ en $\mathcal{N}(\pi) \leq^{\text{bool}\subseteq} \mathcal{N}$.

Net zoals bij de $\neq\emptyset$-modality is het mogelijk dat een query uitdrukbaar is in een taal voor de $\subseteq$-modaliteit indien we enkel ongelabelde input grafen toelaten, terwijl deze niet uitdrukbaar is voor de $\subseteq$-modaliteit voor algemene input grafen. Hierdoor is het mogelijk dat de expressieve kracht van talen veranderd voor de $\subseteq$-modaliteit en dus ook de relatieve expressiviteit voor de $\subseteq$-modaliteit. Het is nog open of de relatieve expressiviteit effectief veranderd; de tot noch toe gevonden resultaten vallen samen met deze voor de $\subseteq$-modaliteit, bovendien vallen ze voorlopig ook nog samen met deze voor $\neq\emptyset$-modaliteit.

Voor het bewijzen van bovenstaande resultaten hebben we gebruik gemaakt van enkele technieken. We zullen deze technieken hieronder even kort toelichten.

– Omdat booleaanse queries een speciale interpretatie zijn van pad queries weten we dat: als $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$ dan $\mathcal{N}(F_1) \leq^{\text{bool}} \mathcal{N}(F_2)$ voor eender welke modaliteit die we geïntroduceerd hebben. Door de contrapositie te nemen van deze eigenschap krijgen we: als $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$ dan $\mathcal{N}(F_1) \leq^{\text{path}} \mathcal{N}(F_2)$, met andere woorden, we kunnen de expressiviteit van talen scheiden op het niveau van pad queries aan de hand van scheiding op het niveau van booleaanse queries.

– Alle pad queries en dus ook alle booleaanse queries in een taal zijn uitdrukbaar in eerste order predicaten logica indien deze taal geen transitieve sluiting bevat. Dit resultaat kunnen we gebruiken om talen F_1 die transitieve sluiting bevatten te scheiden van talen F_2 die geen transitieve sluiting bevatten als volgt: indien een query q uitdrukbaar is in $\mathcal{N}(F_1)$ en niet uitdrukbaar is in eerste orde logica, dan is deze ook niet uitdrukbaar in $\mathcal{N}(F_2)$.

Om aan te tonen dat een query niet uitdrukbaar is in eerste orde predicaten logica hebben we twee methoden gebruikt. De meest gebruikte en eenvoudigste methode is het reduceren naar queries waarvan algemeen gekend is dat ze niet uitdrukbaar zijn in eerste orde logica, e.g., om aan te tonen dat de transitieve sluiting expressiviteit toevoegt aan talen die geen transitieve sluiting bevatten gebruiken we het feit dat de 'reachability' query niet uitdrukbaar is in eerste order predicaten logica. Een andere methode die we gebruikt hebben is de Hanf-lokaliteit. Dit omdat klassieke Ehrenfeucht-Fraïssé spellen snel heel ingewikkeld worden.

– Indien we werken met eindige grafen G_1 en G_2, dan kunnen we de verzameling $A = \{(e(G_1), e(G_2)) \mid e \in \mathcal{N}(F)\}$ berekenen. Hierdoor kunnen we ook berekenen of twee eindige grafen onderscheidbaar zijn in een taal of niet. Dit kunnen we dan gebruiken voor het scheiden van twee talen op het niveau van booleaanse queries als volgt: indien we twee grafen kunnen vinden die onderscheidbaar zijn in $\mathcal{N}(F_1)$ en niet onderscheidbaar zijn in $\mathcal{N}(F_2)$, dan is duidelijk $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$. Deze methode werkt echter niet altijd omdat dergelijke grafen niet altijd bestaan indien $\mathcal{N}(F_1) \not\leq^{\text{bool}} \mathcal{N}(F_2)$. Bovendien kan het berekenen van A zeer traag zijn daar A exponentieel groot kan zijn ten opzichte van het aantal knopen in G_1 en G_2.

– In eerste orde logica worden er Ehrenfeucht-Fraïssé spel gespeeld van k rondes om aan te tonen dat een query niet uitdrukbaar is in FO[k].[3] We hebben een soort gelijk spel ontworpen voor $\mathcal{N}(di, \backslash)_k$.[4] Dit spel kunnen we net zoals in eerste orde logica gebruiken om aan te tonen dat een query niet uitdrukbaar is in $\mathcal{N}(di, \backslash)$.

[3]FO[k] is de verzameling van alle eerste order logica zinnen met kwantor diepte maximaal k.
[4]$\mathcal{N}(di, \backslash)$ staat voor de verzameling van alle queries in $\mathcal{N}(di, \backslash)$ met een graad van maximaal k.

Bibliography

[AE09] R.R.A. Adams and C. Essex. *Calculus: A Complete Course.* Pearson Canada, 2009.

[AF90] Miklos Ajtai and Ronald Fagin. Reachability is harder for directed than for undirected finite graphs. *Journal of Symbolic Logic*, 55:113–150, 1990.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[AU79] Alfred V. Aho and Jeffrey D. Ullman. Universality of data retrieval languages. In *Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '79, pages 110–119, New York, NY, USA, 1979. ACM.

[dBV12] Jan Van den Bussche and Stijn Vansummeren. Optimalisatie van select-project-join expressies. 2012.

[FGL$^+$11] George H. L. Fletcher, Marc Gyssens, Dirk Leinders, Jan Van den Bussche, Dirk Van Gucht, Stijn Vansummeren, and Yuqing Wu. Relative expressive power of navigational querying on graphs. In *Proceedings of the 14th International Conference on Database Theory*, ICDT '11, pages 197–207, New York, NY, USA, 2011. ACM.

[FGL$^+$12a] George H. L. Fletcher, Marc Gyssens, Dirk Leinders, Jan Van den Bussche, Dirk Van Gucht, and Stijn Vansummeren. Similarity and bisimilarity notions appropriate for characterizing indistinguishability in fragments of the calculus of relations. *CoRR*, abs/1210.2688, 2012.

[FGL$^+$12b] George H. L. Fletcher, Marc Gyssens, Dirk Leinders, Jan Van den Bussche, Dirk Van Gucht, Stijn Vansummeren, and Yuqing Wu. The impact of transitive closure on the boolean expressiveness of navigational query languages on graphs. In *FoIKS*, pages 124–143, 2012.

[GMUW09] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database systems - the complete book (2. ed.).* Pearson Education, 2009.

[Gys09] Marc Gyssens. *Algoritmen en Datastructuren.* Hasselt University, 2009.

[Gys12] Marc Gyssens. *Fundamenten van Databases.* Hasselt University, 2012.

[Kui10] Bart Kuijpers. *Logica en Modelleren.* Hasselt University, 2010.

[Lib04] Leonid Libkin. *Elements of Finite Model Theory.* Springer, 2004.